

COMPUTATIONAL FLUID DYNAMICS

Principles and Applications

COMPUTATIONAL FLUID DYNAMICS

Principles and Applications

Third Edition

JIRI BLAZEK, PhD
CFD Consulting & Analysis
Sankt Augustin, Germany



AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Butterworth-Heinemann is an imprint of Elsevier



Butterworth Heinemann is an imprint of Elsevier
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK
225 Wyman Street, Waltham, MA 02451, USA

Copyright © 2015 Elsevier Ltd. All rights reserved.

First Edition: 2005

Reprinted on: 2007

Copyright © 2005, J. Blazek. Published by Elsevier Ltd. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloguing-in-Publication Data

A catalog record for this book is available from the Library of Congress

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN: 978-0-08-099995-1

For information on all Butterworth-Heinemann publications
visit our website at <http://store.elsevier.com/>



Working together
to grow libraries in
developing countries

www.elsevier.com • www.bookaid.org

Publisher: Joe Hayton

Acquisition Editor: Hayley Gray

Editorial Project Manager: Cari Owen

Production Project Manager: Nicky Carter

Designer: Matthew Limbert

Typeset by SPI

Printed and bound in the UK

ACKNOWLEDGMENTS

My first thanks are to our Creator, without whom nothing would be possible. Furthermore, I wish to thank my father for the initial motivation to start this project, as well as for his continuous help with the text and in particular with the drawings. I also gratefully acknowledge the support of the staff at Elsevier Ltd., foremost of C. Owen and H. Gray, during the preparation of this edition.

LIST OF SYMBOLS

\bar{A}_c	Jacobian of convective fluxes
\bar{A}_v	Jacobian of viscous fluxes
b	constant depth of control volume in two dimensions
c	speed of sound
c_p	specific heat coefficient at constant pressure
c_v	specific heat coefficient at constant volume
\vec{C}	vector of characteristic variables
C_m	molar concentration of species m ($= \rho Y_m / W_m$)
C_S	Smagorinsky constant
$\text{curl } \vec{v}$	curl of \vec{v} $\left(= \vec{\nabla} \times \vec{v} = \left[\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}, \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right] \right)$
d	distance
\mathbf{D}	diagonal part of implicit operator
\vec{D}	artificial dissipation
D_m	effective binary diffusivity of species m
$\text{div } \vec{v}$	divergence of \vec{v} $\left(= \vec{\nabla} \cdot \vec{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right)$
e	internal energy per unit mass
E	total energy per unit mass
f	Fourier symbol of the time-stepping operator
\vec{f}_e	vector of external volume forces
\vec{F}	flux vector
$\overline{\overline{F}}$	flux tensor
g	amplification factor
\vec{g}	grid velocity
h	enthalpy
Δh	local grid (cell) size
H	total (stagnation) enthalpy
\tilde{H}	Hessian matrix (matrix of second derivatives)
I	imaginary unit ($I = \sqrt{-1}$)

\bar{I}	identity matrix
$\bar{\bar{I}}$	unit tensor
\hat{I}_h^{2h}	interpolation operator
I_h^{2h}	restriction operator
I_{2h}^h	prolongation operator
\bar{J}	system matrix (implicit operator)
J^{-1}	inverse of determinant of coordinate transformation Jacobian
k	thermal conductivity coefficient
K	turbulent kinetic energy
K_f, K_b	forward and backward reaction rate constants
l_T	turbulent length scale
\mathbf{L}	strictly lower part of implicit operator
L_{ij}	components of Leonard stress tensor
M	Mach number
\bar{M}	mass matrix
\vec{n}	unit normal vector (outward pointing) of control volume face
n_x, n_y, n_z	components of the unit normal vector in x -, y -, z -direction
N	number of grid points, cells, or control volumes
N_A	number of adjacent control volumes
N_F	number of control volume faces
p	static pressure
P	production term of kinetic turbulent energy
\bar{P}	transformation matrix from primitive to conservative variables
\bar{P}_L, \bar{P}_R	left and right preconditioning matrix (Krylov–subspace methods)
Pr	Prandtl number
\dot{q}_h	heat flux due to radiation, chemical reactions, etc.
Q	source term
\vec{r}	position vector (Cartesian coordinates); residual (GMRES)
\vec{r}_{ij}	vector from point i to point j
R	specific gas constant
R_u	universal gas constant ($= 8314.34 \text{ J/kg mol K}$)
\vec{R}	residual, right-hand side
\vec{R}^*	smoothed residual
$\bar{\mathcal{R}}$	rotation matrix

Re	Reynolds number
\dot{s}_m	rate of change of species m due to chemical reactions
\vec{S}	face vector ($= \vec{n} \Delta S$)
S_{ij}	components of strain-rate tensor
S_x, S_y, S_z	Cartesian components of the face vector
dS	surface element
ΔS	length/area of a face of a control volume
t	time
t_T	turbulent time scale
Δt	time step
T	static temperature
\bar{T}	matrix of right eigenvectors
\bar{T}^{-1}	matrix of left eigenvectors
u, v, w	Cartesian velocity components
u_τ	skin friction velocity ($= \sqrt{\tau_w / \rho}$)
U	general (scalar) flow variable
\mathbf{U}	strictly upper part of implicit operator
\vec{U}	vector of general flow variables
\vec{v}	velocity vector with the components u, v , and w
V	contravariant velocity
V_r	contravariant velocity relative to grid motion
V_t	contravariant velocity of a face of the control volume
W_m	molecular weight of species m
\vec{W}	vector of conservative variables ($= [\rho, \rho u, \rho v, \rho w, \rho E]^T$)
\vec{W}_p	vector of primitive variables ($= [p, u, v, w, T]^T$)
x, y, z	Cartesian coordinate system
Δx	cell size in x -direction
γ^+	nondimensional wall coordinate ($= \rho y u_\tau / \mu_w$)
Y_m	mass fraction of species m
z	Fourier symbol of the spatial operator
α	angle of attack, inlet angle
α_m	coefficient of the Runge-Kutta scheme (in stage m)
β	parameter to control time accuracy of an implicit scheme
β_m	blending coefficient (in stage m of the Runge-Kutta scheme)

γ	ratio of specific heat coefficients at constant pressure and volume
Γ	circulation
$\bar{\Gamma}$	preconditioning matrix (low Mach-number flow)
δ_{ij}	Kronecker symbol
ε	rate of turbulent energy dissipation
ϵ	smoothing coefficient (implicit residual smoothing); parameter
κ	thermal diffusivity coefficient
λ	second viscosity coefficient
Λ_c	eigenvalue of convective flux Jacobian
$\tilde{\Lambda}_c$	diagonal matrix of eigenvalues of convective flux Jacobian
$\hat{\Lambda}_c$	spectral radius of convective flux Jacobian
$\hat{\Lambda}_\nu$	spectral radius of viscous flux Jacobian
μ	dynamic viscosity coefficient
ν	kinematic viscosity coefficient ($= \mu/\rho$)
ξ, η, ζ	curvilinear coordinate system
ρ	density
σ	Courant-Friedrichs-Lowy (CFL) number
σ^*	CFL number due to residual smoothing
τ	viscous stress
$\tau_{w\!/\!w}$	wall shear stress
$\overline{\tau}$	viscous stress tensor (normal and shear stresses)
τ_{ij}	components of viscous stress tensor
τ_{ij}^F	components of Favre-averaged Reynolds stress tensor
τ_{ij}^R	components of Reynolds stress tensor
τ_{ij}^S	components of subgrid-scale stress tensor
τ_{ij}^{SF}	components of Favre-filtered subgrid-scale stress tensor
τ_{ij}^{SR}	components of subgrid-scale Reynolds stress tensor
ω	rate of dissipation per unit turbulent kinetic energy ($=\varepsilon/K$)
Υ	pressure sensor
Ω	control volume
Ω_{ij}	components of rotation-rate tensor
$\partial\Omega$	boundary of a control volume
Ψ	limiter function

$\vec{\nabla}U$	gradient of scalar $U\left(=\left[\frac{\partial U}{\partial x}, \frac{\partial U}{\partial y}, \frac{\partial U}{\partial z}\right]\right)$
∇^2U	Laplace of scalar $U\left(=\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2}\right)$
$\ \vec{U}\ _2$	2-norm of vector $\vec{U}\left(=\sqrt{\vec{U} \cdot \vec{U}}\right)$

SUBSCRIPTS

C	convective part
c	related to convection
D	diffusive part
i, j, k	nodal point index
I, J, K	index of a control volume
L	laminar; left
m	index of control volume face; species
R	right
T	turbulent
v	viscous part
V	related to volume
w	wall
x, y, z	components in the x -, y -, z -direction
∞	at infinity (far-field)

SUPERSCRIPTS

I, J, K	direction in computational space
n	previous time level
$n + 1$	new time level
T	transpose
\sim	Favre averaged mean value; Favre-filtered value (LES)
$"$	fluctuating part of Favre decomposition; subgrid scale (LES)
$-$	Reynolds averaged mean value; filtered value (LES)
$'$	fluctuating part of Reynolds decomposition; subgrid scale (LES)

ABBREVIATIONS

AGARD	Advisory Group for Aerospace Research and Development (NATO)
AIAA	American Institute of Aeronautics and Astronautics
ARC	Aeronautical Research Council, UK
ASME	The American Society of Mechanical Engineers
CERCA	Centre de Recherche en Calcul Applique (Centre for Research on Computation and its Applications), Montreal, Canada
CERFACS	Centre Europeen de Recherche et de Formation Avancee en Calcul Scientifique (European Centre for Research and Advanced Training in Scientific Computation), France
DFVLR	(now DLR) Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (German Aerospace Research Establishment)
DLR	Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center)
ERCOFTAC	European Research Community on Flow, Turbulence and Combustion
ESA	European Space Agency
FFA	Flygtekniska Försöksanstalten (The Aeronautical Research Institute of Sweden)
GAMM	Gesellschaft für Angewandte Mathematik und Mechanik (German Society of Applied Mathematics and Mechanics)
ICASE	Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, USA
INRIA	Institut National de Recherche en Informatique et en Automatique (The French National Institute for Research in Computer Science and Control)
ISABE	International Society for Air Breathing Engines
MAE	Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NY, USA

NACA	(now NASA) The National Advisory Committee for Aeronautics, USA
NASA	National Aeronautics and Space Administration, USA
NLR	Nationaal Lucht en Ruimtevaartlaboratorium (National Aerospace Laboratory), The Netherlands
ONERA	Office National d'Etudes et de Recherches Aerospatiales (National Institute for Aerospace Studies and Research), France
SIAM	Society of Industrial and Applied Mathematics, USA
VKI	Von Karman Institute for Fluid Dynamics, Belgium
ZAMM	Zeitschrift für angewandte Mathematik und Mechanik (Journal of Applied Mathematics and Mechanics), Germany
ZFW	Zeitschrift für Flugwissenschaften und Weltraumforschung (Journal of Aeronautics and Space Research), Germany
1D	one dimension
1-D	one-dimensional
2D	two dimensions
2-D	two-dimensional
3D	three dimensions
3-D	three-dimensional

CHAPTER 1

Introduction

The history of the computational fluid dynamics (CFD) started in the early 1970s. Around that time, it became an acronym for the combination of physics, numerical mathematics, and, to some extent, computer sciences—all employed to simulate fluid flows. The beginning of CFD was triggered by the availability of increasingly more powerful mainframes, and still the advances in CFD are closely linked to the evolution of the computer technology. Among the first applications of the CFD methods was the simulation of transonic flows based on the solution of the non-linear potential equation. With the beginning of the 1980s, first the solutions of two-dimensional (2-D) and later three-dimensional (3-D) Euler equations became feasible. Thanks to the rapidly increasing speed of supercomputers, and due to the development of a variety of numerical acceleration techniques like multigrid, it became possible to compute inviscid flows either past complete aircraft configurations or inside of turbomachinery. With the mid-1980s, the focus started to shift to the significantly more demanding simulations of viscous flows governed by the Navier-Stokes equations. Together with this, a variety of turbulence models evolved with different degree of numerical complexity and accuracy. The leading edge in turbulence modeling is represented by the direct numerical simulation and the large eddy simulation (LES).

With the advances of the numerical methodologies, particularly of the implicit schemes, solution of flow problems that require real gas modeling also became feasible by the end of the 1980s. Among the first large scale application, 3-D hypersonic flow past re-entry vehicles, like the European HERMES shuttle, was computed using equilibrium and later non-equilibrium chemistry models. Many research activities were and still are devoted to the numerical simulation of combustion and particularly to flame modeling. These efforts are very important for the development of low emission gas turbines and engines. Also, the modeling of steam and in particular condensation of steam became a key factor in designing efficient steam turbines.

Due to the steadily increasing demands on the complexity and the fidelity of flow simulations, grid generation methods became more and more sophisticated. The development started first with relatively simple structured meshes, constructed either by algebraic methods or by using partial differential equations. But with the increasing geometrical complexity of the configurations, the grids had to be divided into a number of topologically simpler blocks (multiblock approach). The next logical step was to allow for non-matching interfaces between the grid blocks, in order to relieve the constraints

imposed on the grid generation in a single block. Finally, solution methodologies were introduced that can deal with grids overlapping each other (Chimera technique). This allowed, for example, to simulate the flow past the complete Space Shuttle vehicle with the external tank and boosters attached. However, the generation of a structured, multiblock grid for a complicated geometry may still take weeks to accomplish. Therefore, the research also focused on the development of unstructured grid generators and flow solvers, which promise significantly reduced setup times, with only a minor user intervention. Another very important feature of the unstructured methodology is the possibility of solution-based grid adaptation. The first unstructured grids consisted exclusively of isotropic tetrahedra, which was fully sufficient for inviscid flows governed by the Euler equations. However, the solution of the Navier-Stokes equations at higher Reynolds numbers requires grids, which are highly stretched in the shear layers. Although such grids can also be constructed from tetrahedral elements, it is advisable to use prisms or hexahedra in the viscous flow regions and tetrahedra outside. This improves not only the solution accuracy, but it also saves the number of elements, faces, and edges. Thus, the memory and run-time requirements of the simulation are reduced significantly.

Nowadays, CFD methodologies are routinely employed in the fields of aircraft, turbomachinery, car, and ship design. Furthermore, CFD is also applied in meteorology, oceanography, astrophysics, biology, oil recovery, and in architecture. Many numerical techniques developed for CFD are also utilized in the solution of the Maxwell equations or in aeroacoustics. Hence, CFD has become an important design tool in engineering, and also an indispensable research tool in various sciences. Due to the advances in numerical solution methods and in the computer technology, geometrically and physically complex cases can be run even on PCs or on PC clusters. Large scale simulations of viscous flows on grids consisting of dozens of millions of elements can be accomplished within only a few hours on today's supercomputers. However, it would be completely wrong to think that CFD represents a mature technology now, like, for example, the finite-element methods in solid mechanics. No, there are still many open questions like turbulence and combustion modeling, heat transfer, efficient solution techniques for viscous flows, robust but accurate discretization methods, automated grid generators, etc. The coupling between CFD and other disciplines (like the solid mechanics) requires further research as well. Quite new opportunities also arise in the design optimization by using CFD.

The objective of this book is to provide university students with a solid foundation for understanding the numerical methods employed in today's CFD and to familiarize them with modern CFD codes by hands-on experience. The book is also intended for engineers and scientists starting to work in the field of CFD, or who are applying CFD codes. The mathematics used is always connected to the underlying physics to

facilitate the understanding of the matter. The text can serve as a reference handbook too. Each chapter contains an extensive bibliography, which may form the basis for further studies.

CFD methods are concerned with the solution of equations of fluid motion as well as with the interaction of the fluid with solid bodies. The equations governing the motion of an inviscid fluid (Euler equations) and of viscous fluid (Navier-Stokes equations) are derived in Chapter 2. Additional thermodynamic relations for a perfect gas as well as for a real gas are also discussed. Chapter 3 deals with the principles of solution of the governing equations. The most important methodologies are briefly described and the corresponding references are provided. Chapter 3 can be used together with Chapter 2 to get acquainted with the fundamental principles of CFD.

Numerous schemes were developed in the past for the spatial discretization of the Euler and the Navier-Stokes equations. A unique feature of the present book is that it deals with both the structured (Chapter 4) as well as with the unstructured finite-volume schemes (Chapter 5), because of their broad application possibilities, especially for the treatment of complex flow problems routinely encountered in an industrial environment. The attention is particularly devoted to the definition of the various types of control volumes together with spatial discretization methodologies for convective and viscous fluxes. The 3-D finite-volume formulations of the most popular central and upwind schemes are presented in detail.

The methodologies for the temporal discretization of the governing equations can be divided into two main classes. One class comprises explicit time-stepping schemes (Section 6.1), and the other one consists of implicit schemes (Section 6.2). In order to provide a more complete overview, recently developed solution methods based on the Newton-iteration as well as standard techniques like the explicit Runge-Kutta schemes are discussed.

Two qualitatively different types of viscous fluid flows are encountered in general: laminar and turbulent. The solution of the Navier-Stokes equations does not raise any fundamental difficulties in the case of laminar flows. However, the simulation of turbulent flows continues to present a significant challenge as before. A relatively simple way of modeling the turbulence is offered by the so-called Reynolds-averaged Navier-Stokes equations. On the other hand, Reynolds stress models or LES enable considerably more accurate predictions of turbulent flows. In Chapter 7, various well-proven and widely applied turbulence models of varying level of complexity are presented in detail.

In order to account for the specific features of a particular problem, and to obtain an unique solution of the governing equations, it is necessary to specify appropriate boundary conditions. Basically, there are two types of boundary conditions: physical and numerical. Chapter 8 deals with both types in different situations like solid walls,

inlet, outlet, injection, and far-field. Symmetry planes, periodic and block boundaries are treated as well.

In order to reduce the computer time required to solve the governing equations for complex flow problems, it is quite essential to employ numerical acceleration techniques. Chapter 9 deals extensively, among others, with approaches like the implicit residual smoothing and multigrid. Another important methodology which is also described in Chapter 9 is preconditioning of the governing equations. It allows the application of a single numerical scheme for flows, where the Mach number varies between nearly zero and transonic or higher values. Finally, Chapter 9 contains a section on the parallelization of numerical computer codes by using different approaches.

Each discretization of the governing equations introduces a certain error—the discretization error. Several consistency requirements have to be fulfilled by the discretization scheme, in order to ensure the solution of the discretized equations closely approximates the solution of the original equations. This problem is addressed in the first two parts of Chapter 10. Before a particular numerical solution method is implemented, it is important to know, at least approximately, how the method will influence the stability and the convergence behavior of the CFD code. It was frequently confirmed that the Von Neumann stability analysis can provide a good assessment of the properties of a numerical scheme. Therefore, the third part of Chapter 10 deals with stability analysis for various model equations.

One of the challenging tasks in CFD is the generation of structured or unstructured body-fitted grids around complex geometries. The grid is used to discretize the governing equations in space. The accuracy of the flow solution is therefore closely linked to the quality of the grid. In Chapter 11, the most important methodologies for the generation of structured as well as unstructured grids are discussed in depth.

In order to demonstrate the practical aspects of different numerical solution methodologies, various source codes are available for download. Provided are the sources of quasi 1-D Euler, as well as of 2-D Euler and Navier-Stokes structured and unstructured flow solvers. Furthermore, source codes of 2-D structured algebraic and elliptic grid generators are included together with a converter from structured to unstructured grids. Furthermore, two programs are provided to conduct the linear stability analysis of explicit and implicit time-stepping schemes. The source codes are completed by a set of worked out examples including the grids, the input files and the results. The code package also contains several programs for the demonstration of parallelization techniques. Chapter 12 describes the contents of the directories, the capabilities of the particular programs, and provides examples of their usage.

The present book is finalized with an Appendix and Index. The Appendix contains the governing equations presented in a differential form as well as their characteristic properties. Formulations of the governing equations in rotating frame of reference

and for moving grids are discussed along with some simplified forms. Furthermore, Jacobian and transformation matrices from conservative to characteristic variables are presented for two and three dimensions. The GMRES conjugate gradient method for the solution of linear equations systems is described next. The Appendix closes with a brief explanation of the tensor notation.

CHAPTER 2

Governing Equations

Contents

2.1 The Flow and Its Mathematical Description	7
2.1.1 Finite control volume	8
2.2 Conservation Laws	10
2.2.1 The continuity equation	10
2.2.2 The momentum equation	10
2.2.3 The energy equation	12
2.3 Viscous Stresses	14
2.4 Complete System of the Navier-Stokes Equations	16
2.4.1 Formulation for a perfect gas	19
2.4.2 Formulation for a real gas	19
2.4.3 Simplifications to the Navier-Stokes equations	23
<i>Thin shear layer approximation</i>	23
<i>Parabolized Navier-Stokes equations</i>	24
<i>Euler equations</i>	25
References	26

2.1 THE FLOW AND ITS MATHEMATICAL DESCRIPTION

Before we begin with the derivation of the basic equations describing the behavior of the fluid, it may be convenient to clarify what the term “*fluid dynamics*” stands for. It is, in fact, the investigation of the interactive motion of a large number of individual particles. These are in our case molecules or atoms. That means, we assume the density of the fluid is high enough, so that it can be approximated as a *continuum*. It implies that even an infinitesimally small (in the sense of differential calculus) element of the fluid still contains a sufficient number of particles, for which we can specify mean velocity and mean kinetic energy. In this way, we are able to define velocity, pressure, temperature, density, and other important quantities at each point of the fluid.

The derivation of the principal equations of fluid dynamics is based on the fact that the dynamical behavior of a fluid is determined by the following *conservation laws*, namely:

1. the conservation of mass,
2. the conservation of momentum,
3. the conservation of energy.

The conservation of a certain flow quantity means that its total variation inside an arbitrary volume can be expressed as the net effect of the amount of the quantity being transported across the boundary, of any internal forces and sources, and of external forces acting on the volume. The amount of the quantity crossing the boundary is called *flux*. The flux can be in general decomposed into two different parts: one due to the convective transport and the other one due to the molecular motion present in the fluid at rest. This second contribution is of a diffusive nature—it is proportional to the gradient of the quantity considered, and hence it will vanish for a homogeneous distribution.

The discussion of the conservation laws leads us quite naturally to the idea of dividing the flow field into a number of volumes and to concentrate on the modeling of the behavior of the fluid in one such finite region. For this purpose, we define the so-called *finite control volume* and try to develop a mathematical description of its physical properties.

2.1.1 Finite control volume

Consider a general flow field as represented by streamlines in Fig. 2.1. An arbitrary finite region of the flow, bounded by the closed surface $\partial\Omega$ and fixed in space, defines the control volume Ω . We also introduce a surface element dS and its associated, outward pointing unit normal vector \vec{n} . The conservation law applied to an exemplary scalar quantity per unit volume U says that its variation in time within Ω , that is,

$$\frac{\partial}{\partial t} \int_{\Omega} U d\Omega$$

is equal to the sum of the contributions due to the *convective flux*—amount of the quantity U entering the control volume through the boundary with the velocity \vec{v}

$$-\oint_{\partial\Omega} U(\vec{v} \cdot \vec{n}) dS,$$

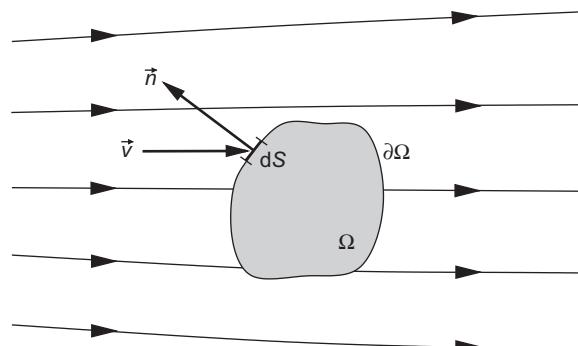


Figure 2.1 Definition of a finite control volume (fixed in space).

further due to the *diffusive flux*—expressed by the generalized Fick's gradient law

$$\oint_{\partial\Omega} \kappa\rho[\nabla(U/\rho) \cdot \vec{n}] dS,$$

where κ is the *thermal diffusivity coefficient*, and finally due to the volume as well as surface sources, Q_V , \vec{Q}_S , that is,

$$\int_{\Omega} Q_V d\Omega + \oint_{\partial\Omega} (\vec{Q}_S \cdot \vec{n}) dS.$$

After summing up the above contributions, we obtain the following general form of the conservation law for the scalar quantity U

$$\frac{\partial}{\partial t} \int_{\Omega} U d\Omega + \oint_{\partial\Omega} [U(\vec{v} \cdot \vec{n}) - \kappa\rho(\nabla U^* \cdot \vec{n})] dS = \int_{\Omega} Q_V d\Omega + \oint_{\partial\Omega} (\vec{Q}_S \cdot \vec{n}) dS, \quad (2.1)$$

where U^* denotes the quantity U per unit mass, that is, U/ρ .

It is important to note that if the conserved quantity would be a vector instead of a scalar, Eq. (2.1) would formally be still valid. But in difference, the convective and the diffusive flux would become tensors instead of vectors— $\bar{\bar{F}}_C$ the *convective flux tensor* and $\bar{\bar{F}}_D$ the *diffusive flux tensor*. The volume sources would be a vector \vec{Q}_V , and the surface sources would change into a tensor $\bar{\bar{Q}}_S$. We can therefore write the conservation law for a general vector quantity \vec{U} as

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{U} d\Omega + \oint_{\partial\Omega} [(\bar{\bar{F}}_C - \bar{\bar{F}}_D) \cdot \vec{n}] dS = \int_{\Omega} \vec{Q}_V d\Omega + \oint_{\partial\Omega} (\bar{\bar{Q}}_S \cdot \vec{n}) dS. \quad (2.2)$$

The *integral formulation* of the conservation law, as given by Eqs. (2.1) or (2.2), has two very important and desirable properties:

1. If there are no volume sources present, the variation of U depends solely on the flux across the boundary $\partial\Omega$ and **not** on any flux inside the control volume Ω .
2. This particular form remain valid in the presence of discontinuities in the flow field like shocks or contact discontinuities [1].

Because of its generality and its desirable properties, it is not surprising that the majority of the CFD codes today is based on the integral form of the governing equations.

In the following section, we shall utilize the above integral form in order to derive the corresponding expressions for the three conservation laws of the fluid dynamics.

2.2 CONSERVATION LAWS

2.2.1 The continuity equation

If we restrict our attention to single-phase fluids, the law of mass conservation expresses the fact that mass cannot be created in such a fluid system, nor can it disappear. There is also no diffusive flux contribution to the continuity equation, since for a fluid at rest, any variation of mass would imply a displacement of the fluid particles.

In order to derive the continuity equation, consider the model of a finite control volume fixed in space, as sketched in Fig. 2.1. At a point on the control surface, the flow velocity is \vec{v} , the unit normal vector is \vec{n} , and dS denotes an elemental surface area. The conserved quantity in this case is the density ρ . For the time rate of change of the total mass inside the finite volume Ω we have

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \, d\Omega.$$

The mass flow of a fluid through some surface fixed in space equals to the product of (density) \times (surface area) \times (velocity component perpendicular to the surface). Therefore, the contribution from the convective flux across each surface element dS becomes

$$\rho(\vec{v} \cdot \vec{n}) \, dS.$$

Since by convection \vec{n} always points out of the control volume, we speak of *inflow* if the product $(\vec{v} \cdot \vec{n})$ is negative, and of *outflow* if it is positive and hence the mass leaves the control volume.

As stated above, there are no volume or surface sources present. Thus, by taking into account the general formulation of Eq. (2.1), we can write

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \, d\Omega + \oint_{\partial\Omega} \rho(\vec{v} \cdot \vec{n}) \, dS = 0. \quad (2.3)$$

This represents the integral form of the continuity equation—the mass conservation law.

2.2.2 The momentum equation

We may start the derivation of the momentum equation by recalling the particular form of Newton's second law which states that the variation of momentum is caused by the net force acting on an mass element. For the momentum of an infinitesimally small portion of the control volume Ω (see Fig. 2.1) we have

$$\rho \vec{v} \, d\Omega.$$

The variation in time of momentum within the control volume equals

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \vec{v} d\Omega.$$

Hence, the conserved quantity is here the product of the density and the velocity, that is,

$$\rho \vec{v} = [\rho u, \rho v, \rho w]^T.$$

The convective flux tensor, which describes the transfer of momentum across the boundary of the control volume, consists in the Cartesian coordinate system of the following three components

$$\begin{aligned} x\text{-component: } & \rho u \vec{v} \\ y\text{-component: } & \rho v \vec{v} \\ z\text{-component: } & \rho w \vec{v}. \end{aligned}$$

The contribution of the convective flux tensor to the conservation of momentum is then given by

$$-\oint_{\partial\Omega} \rho \vec{v} (\vec{v} \cdot \vec{n}) dS.$$

The diffusive flux is zero since there is no diffusion of momentum possible for a fluid at rest. Thus, the remaining question is now, what are the forces the fluid element is exposed to? We can identify two kinds of forces acting on the control volume:

1. *External volume or body forces*, which act directly on the mass of the volume. These are, for example, gravitational, buoyancy, Coriolis, or centrifugal forces. In certain cases, there can be electromagnetic forces present as well.
2. *Surface forces*, which act directly on the surface of the control volume, result from the following two sources only:
 - (a) the pressure distribution imposed by the fluid surrounding the volume,
 - (b) the shear and normal stresses resulting from the friction between the fluid and the surface of the volume.

From the above, we can see that the body force per unit volume, further denoted as $\rho \vec{f}_e$, corresponds to the volume sources in Eq. (2.2). Thus, the contribution of the body (external) force to the momentum conservation is

$$\int_{\Omega} \rho \vec{f}_e d\Omega.$$

The surface sources consist then of two parts—of an isotropic pressure component and of a *viscous stress tensor* $\bar{\bar{\tau}}$, that is,

$$\bar{\bar{\mathbf{Q}}}_S = -p \bar{\bar{I}} + \bar{\bar{\tau}} \quad (2.4)$$

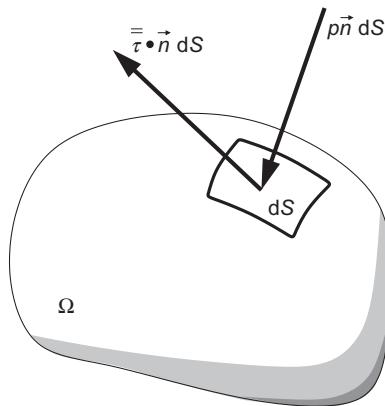


Figure 2.2 Surface forces acting on a surface element of the control volume.

with \bar{I} being the unit tensor (for tensors, see, e.g., [2]). The effect of the surface sources on the control volume is sketched in Fig. 2.2. In Section 2.3, we shall elaborate the form of the stress tensor in more detail, and in particular show how the normal and the shear stresses are connected to the flow velocity.

Hence, if we now sum up all the above contributions according to the general conservation law (Eq. (2.2)), we finally obtain the expression

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \vec{v} \, d\Omega + \oint_{\partial\Omega} \rho \vec{v} (\vec{v} \cdot \vec{n}) \, dS = \int_{\Omega} \rho \vec{f}_e \, d\Omega - \oint_{\partial\Omega} p \vec{n} \, dS + \oint_{\partial\Omega} (\bar{\tau} \cdot \vec{n}) \, dS \quad (2.5)$$

for the momentum conservation inside an arbitrary control volume Ω which is fixed in space.

2.2.3 The energy equation

The underlying principle that we will apply in the derivation of the energy equation is the first law of thermodynamics. Applied to the control volume displayed in Fig. 2.1, it states that any changes in time of the total energy inside the volume are caused by the rate of work of forces acting on the volume and by the net heat flux into it. The total energy per unit mass E of a fluid is obtained by adding its internal energy per unit mass, e , to its kinetic energy per unit mass $|\vec{v}|^2/2$. Thus, we can write for the total energy

$$E = e + \frac{|\vec{v}|^2}{2} = e + \frac{u^2 + v^2 + w^2}{2}. \quad (2.6)$$

The conserved quantity is in this case the total energy per unit volume, that is, ρE . Its variation in time within the volume Ω can be expressed as

$$\frac{\partial}{\partial t} \int_{\Omega} \rho E \, d\Omega.$$

Following the discussion in course of the derivation of the general conservation law (Eq. (2.1)), we can readily specify the contribution of the convective flux as

$$-\oint_{\partial\Omega} \rho E (\vec{v} \cdot \vec{n}) \, dS.$$

In contrast to the continuity and the momentum equation, there is now a diffusive flux. As we already stated, it is proportional to the gradient of the conserved quantity per unit mass (Fick's law). Since the diffusive flux \vec{F}_D is defined for a fluid at rest, only the internal energy becomes effective and we obtain

$$\vec{F}_D = -\gamma \rho \kappa \nabla e, \quad (2.7)$$

where $\gamma = c_p/c_v$ is the ratio of specific heat coefficients and κ denotes the *thermal diffusivity coefficient*. The diffusion flux represents one part of the heat flux into the control volume, namely the diffusion of heat due to molecular thermal conduction—heat transfer due to temperature gradients. Therefore, Eq. (2.7) is in general written in the form of Fourier's law of heat conduction, that is,

$$\vec{F}_D = -k \nabla T \quad (2.8)$$

where k is the *thermal conductivity coefficient* and T is the absolute static temperature.

The other part of the net heat flux into the finite control volume consists of volumetric heating due to the absorption or emission of radiation, or due to chemical reactions. We will denote the heat sources—the time rate of heat transfer per unit mass—as \dot{q}_h . Together with the rate of work done by the body forces \vec{f}_e , which we have introduced for the momentum equation, it completes the volume sources

$$Q_V = \rho \vec{f}_e \cdot \vec{v} + \dot{q}_h. \quad (2.9)$$

The last contribution to the conservation of energy, which we have yet to determine, are the surface sources Q_S . They correspond to the time rate of work done by the pressure as well as the shear and normal stresses on the fluid element (see Fig. 2.2), that is,

$$\vec{Q}_S = -p \vec{v} + \bar{\tau} \cdot \vec{v}. \quad (2.10)$$

Sorting now all the above contributions and terms, we obtain the following expression for the energy conservation equation

$$\begin{aligned} \frac{\partial}{\partial t} \int_{\Omega} \rho E \, d\Omega + \oint_{\partial\Omega} \rho E (\vec{v} \cdot \vec{n}) \, dS &= \oint_{\partial\Omega} k(\nabla T \cdot \vec{n}) \, dS + \int_{\Omega} (\rho \vec{f}_e \cdot \vec{v} + \dot{q}_h) \, d\Omega \\ &\quad - \oint_{\partial\Omega} p (\vec{v} \cdot \vec{n}) \, dS + \oint_{\partial\Omega} (\bar{\tau} \cdot \vec{v}) \cdot \vec{n} \, dS. \end{aligned} \quad (2.11)$$

The energy equation (2.11) is usually written in a slightly different form. For that purpose, we shall utilize the following general relation between the total enthalpy, the total energy and the pressure

$$H = h + \frac{|\vec{v}|^2}{2} = E + \frac{p}{\rho}. \quad (2.12)$$

When we gather now the convective ($\rho E \vec{v}$) and the pressure term ($p \vec{v}$) in the energy conservation law (2.11), and apply the formula (2.12), we can write the energy equation in this final form

$$\begin{aligned} \frac{\partial}{\partial t} \int_{\Omega} \rho E \, d\Omega + \oint_{\partial\Omega} \rho H (\vec{v} \cdot \vec{n}) \, dS &= \oint_{\partial\Omega} k(\nabla T \cdot \vec{n}) \, dS \\ &\quad + \int_{\Omega} (\rho \vec{f}_e \cdot \vec{v} + \dot{q}_h) \, d\Omega + \oint_{\partial\Omega} (\bar{\tau} \cdot \vec{v}) \cdot \vec{n} \, dS. \end{aligned} \quad (2.13)$$

Herewith, we have derived integral formulations of the three conservation laws: the conservation of mass (2.3), of momentum (2.5), and of energy (2.13). In the next section, we shall work out the formulation of the normal and of the shear stresses in more detail.

2.3 VISCOUS STRESSES

The viscous stresses, which originate from the friction between the fluid and the surface of an element, are described by the stress tensor $\bar{\tau}$. In Cartesian coordinates its general form is given by

$$\bar{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix}. \quad (2.14)$$

The notation τ_{ij} means by convention that the particular stress component affects a plane perpendicular to the i -axis, in the direction of the j -axis. The components τ_{xx} , τ_{yy} , and τ_{zz} represent the normal stresses, the other components of $\bar{\tau}$ stand for the shear stresses, respectively. Figure 2.3 shows the stresses for a quadrilateral fluid element. One can notice that the normal stresses (Fig. 2.3a) try to displace the faces of the element in three mutually perpendicular directions, whereas the shear stresses (Fig. 2.3b) try to shear the element.

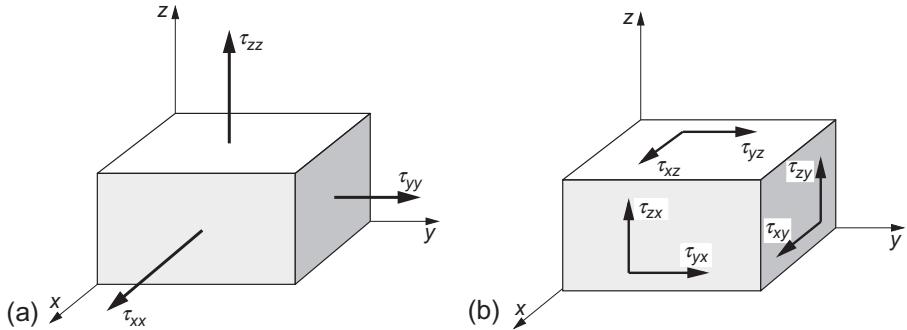


Figure 2.3 Normal (a) and shear stresses (b) acting on a finite fluid element.

You may ask now, how the viscous stresses are evaluated. First of all, they depend on the dynamical properties of the medium. For fluids like air or water, Newton stated that the shear stress is proportional to the velocity gradient. Therefore, medium of such a type is designated as *Newtonian fluid*. On the other hand, fluids like, for example, melted plastic or blood behave in a different manner—they are non-Newtonian fluids. For the vast majority of practical problems, where the fluid can be assumed to be Newtonian, the components of the viscous stress tensor are defined by the relations [3, 4]

$$\begin{aligned}
 \tau_{xx} &= \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial u}{\partial x}, \\
 \tau_{yy} &= \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial v}{\partial y}, \\
 \tau_{zz} &= \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial w}{\partial z}, \\
 \tau_{xy} = \tau_{yx} &= \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right), \\
 \tau_{xz} = \tau_{zx} &= \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right), \\
 \tau_{yz} = \tau_{zy} &= \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right)
 \end{aligned} \tag{2.15}$$

in which λ represents the *second viscosity coefficient*, and μ denotes the *dynamic viscosity coefficient*. For convenience, we can also define the *kinematic viscosity coefficient*, which is given by the formula

$$\nu = \mu/\rho. \tag{2.16}$$

The expressions in Eq. (2.15) were derived by the Englishman George Stokes in the middle of the 19th century. The terms $\mu(\partial u/\partial x)$, etc. in the normal stresses represent the rate of *linear dilatation*—a change in shape. On the other hand, the term $(\lambda \operatorname{div} \vec{v})$

in Eq. (2.15) represents *volumetric* dilatation—the rate of change in volume, which is in essence a change of the density.

In order to close the expressions for the normal stresses, Stokes introduced the hypothesis [5] that

$$\lambda + \frac{2}{3}\mu = 0. \quad (2.17)$$

The above relation (2.17) is termed as the *bulk viscosity*. Bulk viscosity represents the property that is responsible for energy dissipation in a fluid of uniform temperature during a change in volume at finite rate.

With the exception of extremely high temperatures or pressures, there is so far no experimental evidence that Stokes's hypothesis in Eq. (2.17) does not hold (see discussion in Ref. [6]). It is therefore generally used to eliminate λ from Eq. (2.15). Hence, we obtain for the normal viscous stresses

$$\begin{aligned} \tau_{xx} &= 2\mu \left(\frac{\partial u}{\partial x} - \frac{1}{3} \operatorname{div} \vec{v} \right), \\ \tau_{yy} &= 2\mu \left(\frac{\partial v}{\partial y} - \frac{1}{3} \operatorname{div} \vec{v} \right), \\ \tau_{zz} &= 2\mu \left(\frac{\partial w}{\partial z} - \frac{1}{3} \operatorname{div} \vec{v} \right). \end{aligned} \quad (2.18)$$

It should be noted that the expressions for the normal stresses in Eq. (2.18) simplify for an incompressible fluid (constant density) because of $\operatorname{div} \vec{v} = 0$ (continuity equation).

What remains to be determined are the viscosity coefficient μ and the thermal conductivity coefficient k as functions of the state of the fluid. This can be done within the framework of continuum mechanics only on the basis of empirical assumptions. We shall return to this problem in the following section.

2.4 COMPLETE SYSTEM OF THE NAVIER-STOKES EQUATIONS

In the previous sections, we have derived separately the conservation laws of mass, momentum, and energy. Now, we can collect them into one system of equations in order to obtain a better overview of the various terms involved. For this purpose, we go back to the general conservation law for a vector quantity, which is expressed in Eq. (2.2). For reasons to be explained later, we will introduce two flux vectors, namely \vec{F}_c and \vec{F}_v . The first one, \vec{F}_c , is related to the convective transport of quantities in the fluid. It is usually termed *vector of the convective fluxes*, although for the momentum and the energy equation it also includes the pressure terms $p\vec{n}$ (Eq. (2.5)) and $p(\vec{v} \cdot \vec{n})$ (Eq. (2.11)), respectively. The second flux vector—named as *vector of the viscous fluxes* \vec{F}_v , contains the

viscous stresses as well as the heat diffusion. Additionally, let us define a source term \vec{Q} , which comprises all volume sources due to body forces and volumetric heating. With all this in mind and conducting the scalar product with the unit normal vector \vec{n} , we can cast Eq. (2.2) together with Eqs. (2.3), (2.5), and (2.13) into

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega + \oint_{\partial\Omega} (\vec{F}_c - \vec{F}_v) dS = \int_{\Omega} \vec{Q} d\Omega. \quad (2.19)$$

The vector of the so-called *conservative variables* \vec{W} consists in three dimensions of the following five components

$$\vec{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}. \quad (2.20)$$

For the vector of convective fluxes we obtain

$$\vec{F}_c = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho H V \end{bmatrix} \quad (2.21)$$

with the *contravariant velocity* V —the velocity normal to the surface element dS —being defined as the scalar product of the velocity vector and the unit normal vector, that is,

$$V \equiv \vec{v} \cdot \vec{n} = n_x u + n_y v + n_z w. \quad (2.22)$$

The total enthalpy H in Eq. (2.21) is given by the formula (2.12). For the vector of viscous fluxes we have with Eq. (2.14)

$$\vec{F}_v = \begin{bmatrix} 0 \\ n_x \tau_{xx} + n_y \tau_{xy} + n_z \tau_{xz} \\ n_x \tau_{yx} + n_y \tau_{yy} + n_z \tau_{yz} \\ n_x \tau_{zx} + n_y \tau_{zy} + n_z \tau_{zz} \\ n_x \Theta_x + n_y \Theta_y + n_z \Theta_z \end{bmatrix}, \quad (2.23)$$

where

$$\begin{aligned} \Theta_x &= u \tau_{xx} + v \tau_{xy} + w \tau_{xz} + k \frac{\partial T}{\partial x}, \\ \Theta_y &= u \tau_{yx} + v \tau_{yy} + w \tau_{yz} + k \frac{\partial T}{\partial y}, \\ \Theta_z &= u \tau_{zx} + v \tau_{zy} + w \tau_{zz} + k \frac{\partial T}{\partial z} \end{aligned} \quad (2.24)$$

are terms describing the work of the viscous stresses and of the heat conduction in the fluid, respectively. Finally, the source term reads

$$\vec{Q} = \begin{bmatrix} 0 \\ \rho f_{e,x} \\ \rho f_{e,y} \\ \rho f_{e,z} \\ \vec{\rho f}_e \cdot \vec{v} + \dot{q}_h \end{bmatrix}. \quad (2.25)$$

In the case of a Newtonian fluid, that is, if the relations (2.15) for the viscous stresses are valid, the above system of equations ((2.19)-(2.25)) is called the *Navier-Stokes equations*. They describe the exchange (flux) of mass, momentum, and energy through the boundary $\partial\Omega$ of a control volume Ω , which is fixed in space (see Fig. 2.1). We have derived the Navier-Stokes equations in integral formulation, in accordance with the conservation laws. Applying Gauss's theorem, Eq. (2.19) can be re-written in differential form [7]. Since the differential form is often found in literature, for completeness it is included in Section A.1.

In some instances, for example, in turbomachinery applications or in geophysics, the control volume is rotating (usually steadily) about some axis. In such a case, the Navier-Stokes equations are transformed into a *rotating frame of reference*. As a consequence, the source term \vec{Q} has to be extended by the effects due to the Coriolis and the centrifugal force [8]. The resulting form of the Navier-Stokes equations may be found in Section A.4. In other cases, the control volume can be subjected to translation or deformation. This happens, for instance, when fluid-structure interaction is investigated. Then, the Navier-Stokes equations (2.19) have to be extended by a term, which describes the relative motion of the surface element dS with respect to the fixed coordinate system [9]. Additionally, the so-called *geometric conservation law* has to be fulfilled [10–12]. We present the appropriate formulation in Section A.5.

The Navier-Stokes equations represent in three dimensions a system of five equations for the five conservative variables ρ , ρu , ρv , ρw , and ρE . But they contain seven unknown flow field variables, namely: ρ , u , v , w , E , p , and T . Therefore, we have to provide two additional equations, which have to formulate thermodynamic relations between the state variables. For example, the pressure being expressed as a function of the density and temperature, and the internal energy or the enthalpy being given as a function of the pressure and temperature. Beyond this, we have to provide the viscosity coefficient μ and the thermal conductivity coefficient k as functions of the state of the fluid, in order to complete the entire system of equations. Clearly, these relationships depend on the kind of fluid being considered. In the following, we shall therefore show methods of closing the equations for two commonly encountered situations.

2.4.1 Formulation for a perfect gas

In pure aerodynamics, it is generally reasonable to assume that the working fluid behaves like a calorically *perfect gas*, for which the equation of state assumes the form [13, 14]

$$p = \rho RT, \quad (2.26)$$

where R denotes the specific gas constant. The enthalpy results from

$$h = c_p T. \quad (2.27)$$

It is convenient to express the pressure in terms of the conservative variables. For that purpose, we have to combine Eq. (2.12), relating the total enthalpy to the total energy, together with the equation of state (2.26). Substituting expression (2.27) for the enthalpy and using the definitions

$$R = c_p - c_v, \quad \gamma = \frac{c_p}{c_v}, \quad (2.28)$$

we finally obtain for the pressure

$$p = (\gamma - 1)\rho \left[E - \frac{u^2 + v^2 + w^2}{2} \right]. \quad (2.29)$$

The temperature is then calculated with the aid of the relationship (2.26). The coefficient of the dynamic viscosity μ , for a perfect gas, is strongly dependent on temperature but only weakly dependent on pressure. The Sutherland formula is frequently used. The result for air is (in SI units)

$$\mu = \frac{1.45 T^{3/2}}{T + 110} \cdot 10^{-6}, \quad (2.30)$$

where the temperature T is in degree Kelvin (K). Thus, at $T = 288\text{ K}$ one obtains $\mu = 1.78 \cdot 10^{-5}\text{ kg/ms}$. The temperature dependence of the thermal conductivity coefficient k resembles that of μ in the case of gases. By contrast, k is virtually constant in the case of liquids. For this reason, the relationship

$$k = c_p \frac{\mu}{Pr} \quad (2.31)$$

is generally used for air. In addition, it is commonly assumed that the Prandtl number Pr is constant in the entire flow field. For air, the Prandtl number takes the value $Pr = 0.72$.

2.4.2 Formulation for a real gas

The matter becomes more complicated when one has to deal with a *real gas*. The reason is that now we have to model a thermodynamic process and chemical reactions in addition to the fluid dynamics. Examples for real gas flows are the simulation of combustion, the hypersonic flow past re-entry vehicles, or the flow in a steam turbine.

In principle, two different methods can be pursued to solve the problem. The first methodology is applicable in cases, where the gas is in chemical and in thermodynamical equilibrium. This implies that there is a unique equation of state. Then, the governing equations (2.19) remain unchanged. Only the values of pressure, temperature, viscosity, etc. are interpolated from lookup tables using curve fits [15–18]. But in practice, the gas is more often in chemical and/or in thermodynamical non-equilibrium and has to be modeled correspondingly.

Let us for illustration consider a gas mixture consisting of N different species. For a finite Damköhler number, which is defined as the ratio of flow-residence time to chemical-reaction time, we have to include finite-rate chemistry into our model. It has to describe the generation/destruction of species due to chemical reactions. In what follows, we will furthermore assume that the temporal and the spatial scales of fluid dynamics and of chemical reactions are much larger compared to those of thermodynamics. Thus, we suppose the gas is thermodynamically in equilibrium but chemically in non-equilibrium. In order to simulate the behavior of such a gas mixture, the Navier-Stokes equations have to be augmented by $(N - 1)$ additional transport equations for the N species [19–27]. Hence, we obtain formally the same system like Eq. (2.19), but now with the vectors of the conservative variables \vec{W} , the flux vectors \vec{F}_c and \vec{F}_v , as well as with the source term \vec{Q} extended by $(N - 1)$ species equations. Recalling the expressions (2.20) to (2.25), the vector of the conservative variables reads now

$$\vec{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \\ \rho Y_1 \\ \vdots \\ \rho Y_{N-1} \end{bmatrix}. \quad (2.32)$$

The convective and the viscous flux vectors transform into

$$\vec{F}_c = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho H V \\ \rho Y_1 V \\ \vdots \\ \rho Y_{N-1} V \end{bmatrix}, \quad \vec{F}_v = \begin{bmatrix} 0 \\ n_x \tau_{xx} + n_y \tau_{xy} + n_z \tau_{xz} \\ n_x \tau_{yx} + n_y \tau_{yy} + n_z \tau_{yz} \\ n_x \tau_{zx} + n_y \tau_{zy} + n_z \tau_{zz} \\ n_x \Theta_x + n_y \Theta_y + n_z \Theta_z \\ n_x \Phi_{x,1} + n_y \Phi_{y,1} + n_z \Phi_{z,1} \\ \vdots \\ n_x \Phi_{x,N-1} + n_y \Phi_{y,N-1} + n_z \Phi_{z,N-1} \end{bmatrix}, \quad (2.33)$$

where

$$\begin{aligned}\Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k\frac{\partial T}{\partial x} + \rho \sum_{m=1}^N h_m D_m \frac{\partial Y_m}{\partial x}, \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k\frac{\partial T}{\partial y} + \rho \sum_{m=1}^N h_m D_m \frac{\partial Y_m}{\partial y}, \\ \Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + k\frac{\partial T}{\partial z} + \rho \sum_{m=1}^N h_m D_m \frac{\partial Y_m}{\partial z}, \\ \Phi_{x,m} &= \rho D_m \frac{\partial Y_m}{\partial x}, \\ \Phi_{y,m} &= \rho D_m \frac{\partial Y_m}{\partial y}, \\ \Phi_{z,m} &= \rho D_m \frac{\partial Y_m}{\partial z}.\end{aligned}\quad (2.34)$$

Finally, the source term becomes

$$\vec{Q} = \begin{bmatrix} 0 \\ \rho f_{e,x} \\ \rho f_{e,y} \\ \rho f_{e,z} \\ \rho \vec{f}_e \cdot \vec{v} + \dot{q}_h \\ \dot{s}_1 \\ \vdots \\ \dot{s}_{N-1} \end{bmatrix}. \quad (2.35)$$

In the above expressions (2.32)–(2.35), Y_m denotes the mass fraction, h_m the enthalpy, and D_m the effective binary diffusivity of species m , respectively. Furthermore, \dot{s}_m is the rate of change of species m due to chemical reactions. Note that the total density ρ of the mixture is equal to the sum of the densities of the species ρY_m . Therefore, since the total density is regarded as an independent quantity, there are only $(N - 1)$ independent densities ρY_m left. The remaining mass fraction Y_N is obtained from

$$Y_N = 1 - \sum_{m=1}^{N-1} Y_m. \quad (2.36)$$

In order to find an expression for the pressure p , we first assume that the individual species behave like ideal gases, that is,

$$p_m = \rho Y_m \frac{R_u}{W_m} T, \quad (2.37)$$

with R_u denoting the universal gas constant and W_m being the molecular weight, respectively. Together with Dalton's law,

$$p = \sum_{m=1}^N p_m, \quad (2.38)$$

we can write

$$p = \rho R_u T \sum_{m=1}^N \frac{Y_m}{W_m}. \quad (2.39)$$

It is important to notice that because the gas is in thermodynamical equilibrium, all species possess the same temperature T . The temperature has to be calculated iteratively from the expression [22, 28]

$$e = \sum_{m=1}^N \left[Y_m \left(h_{f,m}^0 + \int_{T_{\text{ref}}}^T c_{p,m} dT \right) \right] - \frac{p}{\rho}. \quad (2.40)$$

The internal energy of the gas mixture e is obtained from Eq. (2.6). The quantities $h_{f,m}^0$, $c_{p,m}$, and T_{ref} in Eq. (2.40) denote the heat of formation, the specific heat at constant pressure, and the reference temperature of the m th species, respectively. Values of the above quantities as well as of the thermal conductivity k and of the dynamic viscosity μ of the species are determined from curve fits [20, 22, 24].

The last part, which remains to be modeled, is the chemical source term \dot{s}_m in Eq. (2.35). The rate equations for a set of N_R elementary reactions involving N species can be written in the general form

$$\sum_{m=1}^N v'_{lm} C_m \xrightleftharpoons[K_{bl}]{K_{fl}} \sum_{m=1}^N v''_{lm} C_m \quad \text{for } l = 1, 2, \dots, N_R, \quad (2.41)$$

where v'_{lm} and v''_{lm} are the stoichiometric coefficients for species m in the l th forward and backward reaction, respectively. Furthermore, C_m stands for the molar concentration of species m ($C_m = \rho Y_m / W_m$), and finally K_{fl} and K_{bl} , respectively, denote the forward and the backward reaction rate constants for the l th reaction step. They are given by the empirical Arrhenius formulas

$$K_f = A_f T^{B_f} \exp(-E_f / R_u T),$$

$$K_b = A_b T^{B_b} \exp(-E_b / R_u T),$$

where A_f and A_b are the Arrhenius coefficients, E_f and E_b represent the activation energies, and B_f as well as B_b are constants, respectively. The rate of change of molar concentration of species m by the l th reaction is given by

$$\dot{C}_{lm} = (v''_{lm} - v'_{lm}) \left(K_{fl} \prod_{n=1}^N C_n^{v'_{ln}} - K_{bl} \prod_{n=1}^N C_n^{v''_{ln}} \right). \quad (2.42)$$

Hence, together with Eq. (2.42) we can calculate the total rate of change of species m from

$$\dot{s}_m = W_m \sum_{l=1}^{N_R} \dot{C}_{lm}. \quad (2.43)$$

More details can be found in the references cited above. A detailed overview of the equations governing a chemically reacting flow, together with the Jacobian matrices of the fluxes and their eigenvalues, can also be found in Ref. [28].

Another practical example of real gas is the simulation of steam or, which is more demanding, of wet steam in turbomachinery applications [29–39]. In the later case, where the steam is mixed with water droplets, so that we speak of *multiphase flow*, it is either possible to solve an additional set of transport equations, or to trace the water droplets along a number of streamlines. These simulations have very important applications in the design of modern steam turbine cascades. The analysis of flow past turbine blades can for instance help to understand the occurrence of supercritical shocks by condensation and of flow instabilities, responsible for an additional dynamic load on the blades and resulting in a loss of the efficiency.

2.4.3 Simplifications to the Navier-Stokes equations

In the following, we shall consider three common simplifications to the Navier-Stokes equations (2.19). We shall restrict our attention here to the physical reasoning behind each of the approximations. The equations for the first two simplified forms of the Navier-Stokes equations are provided in the Appendix.

Thin shear layer approximation

When simulating flows around bodies for high Reynolds numbers (i.e., when the boundary layer is thin with respect to a characteristic dimension), the Navier-Stokes equations (2.19) can be simplified. One necessary condition is that there is no large area of separated boundary layer. It can then be anticipated that only the gradients of the flow quantities in the normal direction to the surface of the body (η -direction in Fig. 2.4) contribute to the viscous stresses [40, 41]. On the other hand, the gradients in the other coordinate directions (ξ in Fig. 2.4) are neglected in the evaluation of the shear stress tensor (Eqs. (2.14) and (2.15)). We speak here of the so-called *thin shear layer* (TSL) approximation of the Navier-Stokes equations. The motivation for the TSL modification is that the numerical evaluation of the viscous terms becomes computationally less expensive, but, within the assumptions, the solution remains sufficiently accurate. The TSL approximation can also be justified from a practical point of view. In the case of high

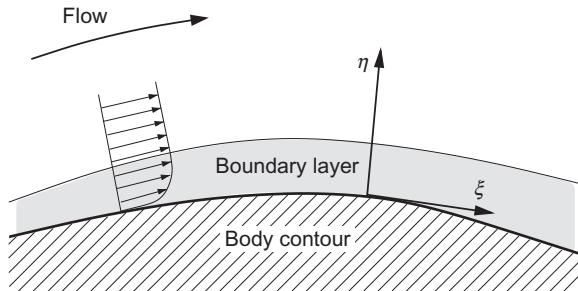


Figure 2.4 Representation of a thin boundary layer.

Reynolds number flows, the grid has to be very fine in the wall normal direction in order to resolve the boundary layer properly. Because of the limited computer memory and speed, much coarser grid has to be generated in the other directions. This in turn results in significantly lower numerical accuracy of the gradient evaluation compared to the normal direction. The TSL equations are for completeness presented in Section A.6. Due to the fact that secondary flow (e.g., like in a blade row) cannot be resolved appropriately, the TSL simplification is usually applied only in external aerodynamics.

Parabolized Navier-Stokes equations

In cases, where the following three conditions are fulfilled:

- the flow is steady (i.e., $\partial \vec{W} / \partial t = 0$),
- the fluid moves predominantly in one main direction (e.g., there must be no boundary layer separation),
- the cross-flow components are negligible,

the governing equations (2.19) can be simplified to a form called the *Parabolized Navier-Stokes* (PNS) equations [8, 42–44]. The above conditions allow us to set the derivatives of u , v , and w with respect to the stream-wise direction to zero in the viscous stress terms (Eq. (2.15)). Furthermore, the components of the viscous stress tensor $\bar{\tau}$, of the work performed by it ($\bar{\tau} \cdot \vec{v}$), and of the heat conduction $k\nabla T$ in the stream-wise direction are dropped from the viscous flux vector in Eq. (2.23). The continuity equation, as well as the convective fluxes (Eq. (2.21)) remain unchanged. For details, the reader is referred to Section A.7. Considering the situation sketched in Fig. 2.5, where the main flow direction coincides with the x coordinate, it can be shown that the PNS approximation leads to a mixed set of parabolic/elliptic equations. Namely, the momentum equation in the flow direction becomes parabolic together with the energy equation, and hence they can be solved by marching in the x -direction. The momentum equations in the y - and in the z -direction are elliptic and they have to be solved iteratively in each x -plane. Thus, the main benefit of the PNS approach is in the largely reduced complexity of the flow solution—from a complete 3-D field

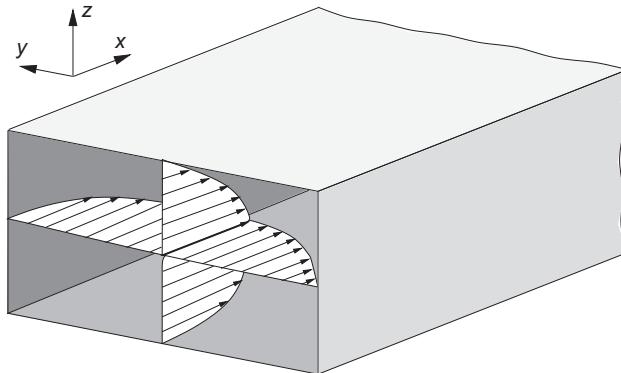


Figure 2.5 Internal flow in a duct—parabolized Navier-Stokes equations.

to a sequence of 2-D problems. A typical application of the parabolized Navier-Stokes equations is the calculation of internal flows in ducts and in pipes, and also the simulation of steady supersonic flows using the space-marching method [45–48].

Euler equations

As we have seen, the Navier-Stokes equations describe the behavior of a viscous fluid. In many instances, it is a valid approximation to neglect the viscous effects completely, like for example for high Reynolds-number flows, where the boundary layer is very thin compared to the dimensions of the body. In such cases, we can simply omit the vector of viscous fluxes, \vec{F}_v , from Eq. (2.19). Thus, we are left with

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega + \oint_{\partial\Omega} \vec{F}_c dS = \int_{\Omega} \vec{Q} d\Omega. \quad (2.44)$$

The remaining terms are given by the same relations (2.20)–(2.22) and Eq. (2.25) as before. This simplified form of the governing equations is called the *Euler equations*. They describe the pure convection of flow quantities in an inviscid fluid. If the Euler equations are formulated in conservative way (like above), they allow for accurate representation of such important phenomena like shocks, expansion waves and vortices over delta wings (with sharp leading edges). Furthermore, the Euler equations served in the past—and still do—as the basis for the development of discretization methods and boundary conditions.

However, it should be noted that today, due to the computational power of even personal computers and due to the increased demands on the quality of the simulations, the Euler equations are only relatively seldom employed for flow simulations.

REFERENCES

- [1] Lax PD. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Comm Pure Appl Math* 1954;7:159–93.
- [2] Aris R. Vectors, tensors and the basic equations of fluid mechanics. New York: Dover Publ. Inc.; 1989.
- [3] Schlichting H. Boundary layer theory. 7th ed. New York: McGraw-Hill; 1979.
- [4] White FM. Viscous fluid flow. New York: McGraw-Hill; 1991.
- [5] Stokes GG. On the theories of internal friction of fluids in motion. *Trans Camb Phil Soc* 1845;8:287–305.
- [6] Gad-el-Hak M. Questions in fluid mechanics: Stokes' hypothesis for a Newtonian, isotropic fluid. *J Fluids Eng* 1995;117:3–5.
- [7] Vinokur M. Conservation equations of gas dynamics in curvilinear coordinate systems. *J Comput Phys* 1974;14:105–25.
- [8] Hirsch C. Numerical computation of internal and external flows, vols. 1 and 2, Chichester, UK: John Wiley and Sons; 1988.
- [9] Pulliam TH, Steger JL. Recent improvements in efficiency, accuracy, and convergence for implicit approximate factorization algorithms. *AIAA Paper* 85-0360; 1985.
- [10] Thomas PD, Lombard CK. Geometric conservation law and its application to flow computations on moving grids. *AIAA J* 1979;17:1030–37.
- [11] Lesoinne M, Farhat C. Geometric conservation laws for flow problems with moving boundaries and deformable meshes and their impact on aeroelastic computations. *AIAA Paper* 95-1709; 1995 [also in *Comp Meth Appl Mech Eng* 1996;134:71–90].
- [12] Guillard H, Farhat C. On the significance of the GCL for flow computations on moving meshes. *AIAA Paper* 99-0793; 1999.
- [13] Zierep J. Vorlesungen über theoretische Gasdynamik (Lectures on theoretical gas dynamics). Karlsruhe: G. Braun Verlag; 1963.
- [14] Liepmann HW, Roshko A. Elements of gas dynamics. Mineola, NY: Dover Publications; 2002.
- [15] Srinivasan S, Weilmuenster KJ. Simplified curve fits for the thermodynamic properties of equilibrium air. *NASA RP-1181*; 1987.
- [16] Schmatz MA. Hypersonic three-dimensional Navier-Stokes calculations for equilibrium gas. *AIAA Paper* 89-2183; 1989.
- [17] Mundt Ch, Keraus R, Fischer J. New, accurate, vectorized approximations of state surfaces for the thermodynamic and transport properties of equilibrium air. *ZFW* 1991;15:179–84.
- [18] Rinaldi E, Pecnik R, Colonna P. Exact Jacobians for implicit Navier-Stokes simulations of equilibrium real gas flows. *J Comput Phys* 2014;270:459–77.
- [19] Bussing TRA, Murman EM. Finite-volume method for the calculation of compressible chemically reacting flows. *AIAA J* 1988;26:1070–78.
- [20] Molvik GA, Merkle CL. A set of strongly coupled, upwind algorithms for computing flows in chemical non-equilibrium. *AIAA Paper* 89-0199; 1989.
- [21] Slomski JF, Anderson JD, Gorski JJ. Effectiveness of multigrid in accelerating convergence of multidimensional flows in chemical non-equilibrium. *AIAA Paper* 90-1575; 1990.
- [22] Shuen JS, Liou MS, van Leer B. Inviscid flux-splitting algorithms for real gases with non-equilibrium chemistry. *J Comput Phys* 1990;90:371–95.
- [23] Li CP. Computational aspects of chemically reacting flows. *AIAA Paper* 91-1574; 1991.
- [24] Shuen J-S, Chen K-H, Choi Y. A coupled implicit method for chemical non-equilibrium flows at all speeds. *J Comput Phys* 1993;106:306–18.
- [25] Raman V, Fox RO, Harvey AD. Hybrid finite-volume/transported PDF simulations of a partially premixed methane-air flame. *Combust Flame* 2004;136:327–50.
- [26] Selle L, Lartigue G, Poinsot T, Koch R, Schildmacher K-U, Krebs W, et al. Compressible large eddy simulation of turbulent combustion in complex geometry on unstructured meshes. *Combust Flame* 2004;137:489–505.
- [27] Ajmani K, Mongia H, Lee P. CFD computations of emissions for LDI-2 combustors with simplex and airblast injectors. *AIAA Paper* 2014-3529; 2014.

- [28] Yu S-T, Chang S-C, Jorgenson PCE, Park S-J, Lai M-C. Basic equations of chemically reactive flow for computational fluid dynamics. AIAA Paper 98-1051; 1998.
- [29] Bakhtar F, Tochai MTM. An investigation of two-dimensional flows of nucleating and wet stream by the time-marching method. *Int J Heat Fluid Flow* 1980;2:5-18.
- [30] Young JB, Snoeck J. In: *Aerothermodynamics of Low Pressure Steam Turbines and Condensers*. Moore MJ, Sieverding CH, editors. New York: Springer Verlag; 1987. p. 87-133.
- [31] Bakhtar F, So KS. A study of nucleating flow of steam in a cascade of supersonic blading by the time-marching method. *Int J Heat Fluid Flow* 1991;12:54-62.
- [32] Young JB. Two-dimensional non-equilibrium wet-steam calculations for nozzles and turbine cascades. *Trans ASME J Turbomach* 1992;114:569-79.
- [33] White AJ, Young JB. Time-marching method for the prediction of two-dimensional, unsteady flows of condensing steam. *AIAA J Propul Power* 1993;9:579-87.
- [34] Liberson A, Kosolapov Y, Rieger N, Hesler S. Calculation of 3-D condensing flows in nozzles and turbine stages. In: EPRI Nucleation Workshop, Rochester, New York; October 24-26, 1995.
- [35] Bakhtar F, Mahpeykar MR, Abbas KK. An investigation of nucleating flows of steam in a cascade of turbine blading — theoretical treatment. *Trans ASME* 1995;117:138-44.
- [36] White AJ, Young JB, Walters PT. Experimental validation of condensing flow theory for a stationary cascade of steam turbine blades. *Phil Trans R Soc Lond A* 1996;354:59-88.
- [37] Fakhari K. Development of a two-phase Eulerian/Lagrangian algorithm for condensing steam flow. *AIAA Paper* 2006-597; 2006.
- [38] Fakhari K. Unsteady phenomena in the condensing steam flow of an industrial steam turbine stage. *AIAA Paper* 2008-1449; 2008.
- [39] Giordano M, Congedo P, Cinnella P. Nozzle shape optimization for wet-steam flows. *AIAA Paper* 2009-4157; 2009.
- [40] Steger JL. Implicit finite difference simulation of flows about two-dimensional arbitrary geometries. *AIAA J* 1978;17:679-86.
- [41] Pulliam TH, Steger JL. Implicit finite difference simulations of three-dimensional compressible flows. *AIAA J* 1980;18:159-67.
- [42] Patankar SV, Spalding DB. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *Int J Heat Mass Transfer* 1972;15:1787-806.
- [43] Aslan AR, Grundmann R. Computation of three-dimensional subsonic flows in ducts using the PNS approach. *ZFW* 1990;14:373-80.
- [44] Kirtley KR, Lakshminarayana B. A multiple passspace-marching method for three-dimensional incompressible viscous flow. *ZFW* 1992;16:49-59.
- [45] Hollenbäck DM, Blom GA. Application of a parabolized Navier-Stokes code to an HSCT configuration and comparison to wind tunnel test data. *AIAA Paper* 93-3537; 1993.
- [46] Krishnan RR, Eidson TM. An efficient, parallel space-marching Euler solver for HSCT research. *AIAA Paper* 95-1749; 1995.
- [47] Nakahashi K, Saitoh E. Space-marching method on unstructured grid for supersonic flows with embedded subsonic regions. *AIAA Paper* 96-0418; 1996.
- [48] Yamaleev NK, Ballmann J. Space-marching method for calculating steady supersonic flows on a grid adapted to the solution. *J Comput Phys* 1998;146:436-63.

CHAPTER 3

Principles of Solution of the Governing Equations

Contents

3.1	Spatial Discretization	32
3.1.1	Finite-difference method	36
3.1.2	Finite-volume method	37
3.1.3	Finite-element method	38
3.1.4	Other discretization methods	39
	<i>Spectral-element method</i>	39
	<i>Lattice Boltzmann method</i>	40
	<i>Gridless method</i>	41
3.1.5	Central and upwind schemes	42
	<i>Central schemes</i>	42
	<i>Upwind schemes</i>	42
	<i>Solution reconstruction</i>	45
	<i>Central versus upwind schemes</i>	46
	<i>Upwind schemes for real gas flows</i>	46
3.2	Temporal Discretization	46
3.2.1	Explicit schemes	48
3.2.2	Implicit schemes	51
3.3	Turbulence Modeling	55
3.4	Initial and Boundary Conditions	58
	References	59

In the previous chapter, we obtained the complete system of the Navier-Stokes/Euler equations. We introduced additional thermodynamic relations for a perfect gas, and we also defined additional transport equations for a chemically reacting gas. Hence, we are now ready to solve the whole system of governing equations for the flow variables. As you can imagine, there exists a vast number of solution methodologies. If we put aside analytical methods, which are applicable to simplified flow problems only, nearly all solution strategies follow the same path.

First of all, the space where the flow is to be computed, the *physical space*, is divided into a large number of geometrical elements called *grid cells*. This process is termed as *grid generation* (some authors use the term *mesh* with identical meaning). It can also be viewed as placing first *grid points* (also called nodes or vertexes) in the physical space and then connecting them by straight lines—*grid lines*. A two-dimensional (2-D) grid consists

normally of triangles and/or of quadrilaterals. In three dimensions (3D), it is usually built of tetrahedra, hexahedra, prisms, or pyramids. The most important requirements placed on a grid generation tool are that there must be no holes between the grid cells but also that the grid cells do not overlap. Additionally, the grid should be *smooth*, that is, there should be no abrupt changes in the volume of the grid cells or in the stretching ratio, and the elements should be as regular as possible. Furthermore, if the grid consists of quadrilaterals or of hexahedra, there should be no large kinks in the grid lines. Otherwise, numerical errors would increase significantly.

On the one hand, the grid can be generated to follow closely the boundaries of the physical space, in which case we speak of *body-fitted* grid (Fig. 3.1a). The main advantage of this approach is that the flow can be resolved very accurately at the

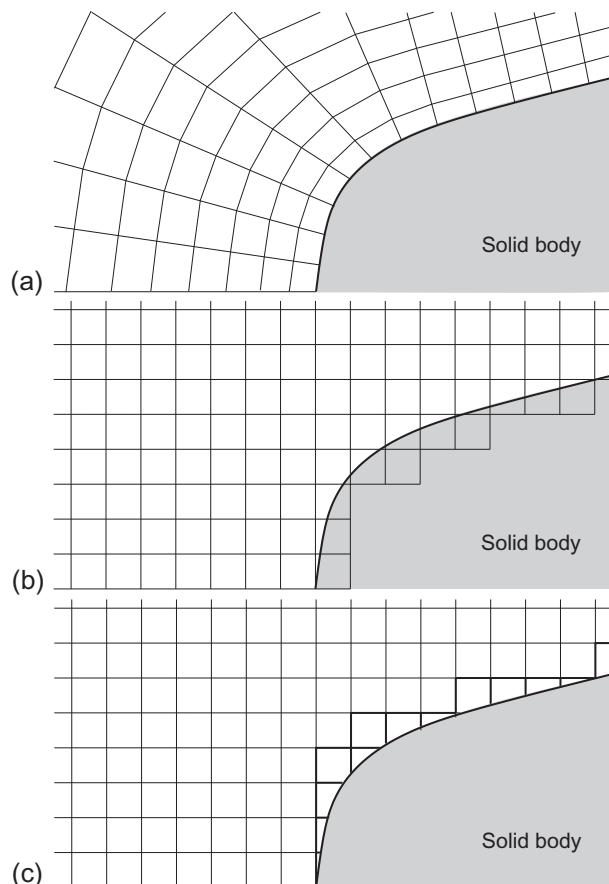


Figure 3.1 2-D body-fitted grid (a), Cartesian grid with immersed boundary (b), Cartesian grid with polygonal boundary cells (c) near a solid body. Boundary cells are marked by a thick line in (c).

boundaries, which is essential in the case of shear layers along solid bodies. The price to be paid is a high degree of complexity of the grid generation tools, especially in the case of “real-life” geometries. On the other hand, the so-called *Cartesian* grids [1–8] where the edges of the grid cells are oriented parallel to the Cartesian coordinates, can be generated very easily. Their advantage is that the evaluation of the fluxes in Eq. (2.19) is much more simple than for body-fitted grids. But when considering Fig. 3.1b or c, it becomes evident that a general and accurate treatment of the boundaries is difficult to accomplish [3]. Because of this disadvantage, body-fitted or combined approaches (see Section 11.2.4) are generally preferred, particularly in the industrial environment, where the geometrical complexity of the simulated configurations is usually very high.

Nowadays, the overwhelming number of numerical methods for the solution of the Euler- and the Navier-Stokes equations employ a separate discretization in space and in time—the so-called *method of lines* [9]. Herewith, dependent on the particular algorithm chosen, the grid is used either to construct control volumes and to evaluate the flux integrals, or to approximate the spatial derivatives of the flow quantities. In a further step, the resulting time-dependent equations are advanced in time, starting from a known initial solution, with the aid of a suitable method. Another possibility, when the flow variables do not change in time, is to find the steady-state solution of the governing equations by means of an iterative process.

The way we derived the governing equations (2.19), the continuity equation (2.3) contains a time derivative of the density. Since the density, as an independent variable, is used to calculate the pressure (Eq. (2.29)), there is a coupling between the time evolution of the density and of the pressure in the momentum equations. Due to this reason, solution methods that employ discretizations of the governing equations (2.19) are called *density-based* schemes. The problem with this formulation is that for an incompressible fluid the pressure is no longer driven by any independent variable, because the time derivative of the density vanishes from the continuity equation. Another difficulty arises from the growing disparity between acoustic and convective wave speeds with decreasing Mach number, which renders the governing equations increasingly stiff and hence hard to solve [10]. Basically, three approaches were developed to cope with the problem. The first possibility is to solve a Poisson equation in pressure, which can be derived from the momentum equations [11–17]. These methods are denoted as *pressure-based* schemes. The second approach, called the *artificial compressibility* method, is based on the idea to substitute time derivative of the pressure for that of the density in the continuity equation [18, 19]. In this way, velocity and pressure field are directly coupled. The third solution, and the most general one, is based on *preconditioning* of the governing equations [20–31]. This methodology allows for the utilization of an identical numerical scheme for very low as well as for high Mach number flows. We shall discuss this approach more extensively in Section 9.5.

In the following section, we shall learn more about the very basic principles of various solution methodologies for the numerical approximation of the governing equations in space and time, for the turbulence modeling, and also for the boundary treatment.

3.1 SPATIAL DISCRETIZATION

Let us at the beginning turn our attention to the first step—the *spatial discretization* of the Navier-Stokes equations, that is, the numerical approximation of the convective and viscous fluxes, as well as of the source term. Many different methodologies were devised for this purpose in the past and the development still continues. In order to sort them, we can at first divide the spatial discretization schemes into the following three main categories: *finite-difference*, *finite-volume*, and *finite-element* schemes. All these methods rely on some kind of grid in order to discretize the governing equations (2.19). Basically, there exist two different types of grids:

- *Structured grids* (Fig. 3.2): Each grid point (vertex, node) is uniquely identified by the indexes i , j , and k , and the corresponding Cartesian coordinates $x_{i,j,k}$, $y_{i,j,k}$, and $z_{i,j,k}$, respectively. The grid cells are quadrilaterals in 2D and hexahedra in 3D. If the

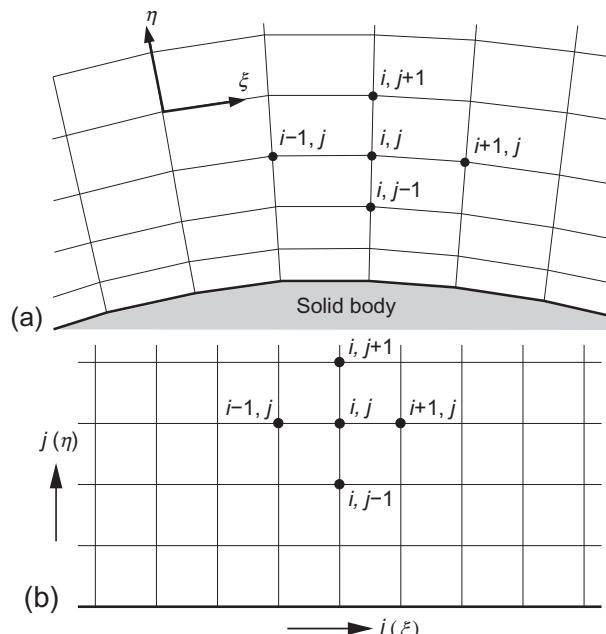


Figure 3.2 Structured, body-fitted grid approach (in 2D): (a) shows the physical space; (b) shows the computational space; ξ and η represent a curvilinear coordinate system.

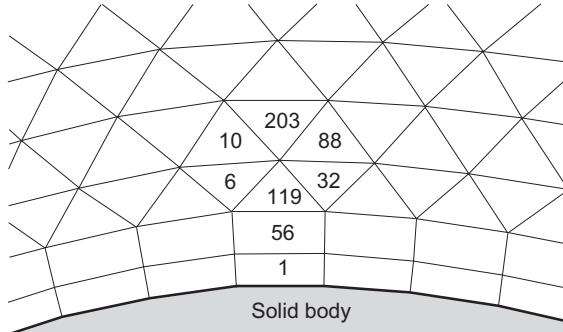


Figure 3.3 Unstructured, mixed grid approach in 2D; numbers mark the individual cells.

grid is body-fitted like the one that is shown in Fig. 3.1a, then we can also speak of *curvilinear* grid.

- *Unstructured* grids (Fig. 3.3): Grid cells as well as grid points have no particular ordering, that is, neighboring cells or grid points cannot be directly identified by their indexes (e.g., cell 6 adjacent to cell 119). In the past, the grid cells were triangles in 2D and tetrahedra in 3D. Now, unstructured grids usually consist of a mix of quadrilaterals and triangles in 2D and of hexahedra, tetrahedra, prisms, and pyramids in 3D, in order to resolve the boundary layers properly. Therefore, we speak in this case of *hybrid* or *mixed* grids.

The main advantage of structured grids follows from the property that the indexes i, j, k represent a linear address space—also called the *computational space*, since it directly corresponds to how the flow variables are stored in the computer memory. This property allows for a very quick and easy access to the neighbors of a grid node, just by adding or subtracting an integer value to or from the corresponding index (e.g., like $(i + 1)$, $(k - 3)$, etc.—see Fig. 3.2). As one can imagine, the evaluation of gradients, fluxes, and also the treatment of boundary conditions is greatly simplified by this feature. The same holds for the implementation of an implicit scheme, because of the well-ordered, banded flux Jacobian matrix. But there is also a disadvantage. It is difficult to generate structured grids for complex geometries. As sketched in Fig. 3.4, one possibility is to divide the physical space into a number of topologically simpler parts—blocks—which can be meshed more easily. We therefore speak of *multiblock* approach [32–36]. Of course, the complexity of the flow solver increases, since special logic is required in order to exchange physical quantities or fluxes between the blocks. Additional flexibility is added, if the grid points at both sides of an interface can be placed independently of each other, that is, if the grid lines are not required to meet at a block boundary (like inside C or F in Fig. 3.4). Those grid points, which are located only on one side of a block interface are called *hanging nodes*. The advantage of this approach is quite obvious—the number

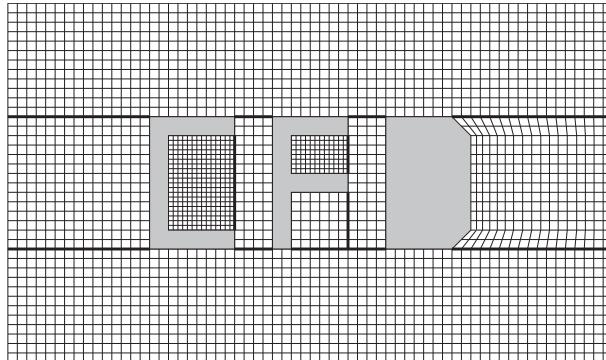


Figure 3.4 Structured, multiblock grid with conforming/non-conforming inter-block interfaces; thick lines represent block boundaries.

of grid lines can be chosen separately for each block as required. The price paid for the enhanced flexibility is an increased overhead for the conservative treatment of the hanging nodes [37, 38]. The multiblock methodology also offers interesting possibilities with respect to the implementation of the flow solver on a parallel computer by means of *domain decomposition*. However, longer time (often weeks or months) is still required for the grid generation in the case of complex configurations.

Another methodology, related to block structured grids, represents the *Chimera technique* [39–46]. The basic idea here is to generate first the grids separately around each geometrical entity in the domain. After that, the grids are combined together in such a way that they overlap each other where they meet. The situation is depicted in Fig. 3.5 for a simple configuration. The crucial operation is an accurate transfer of quantities between the different grids at the overlapping region. Therefore, the extension of the overlap is adjusted accordingly to the required

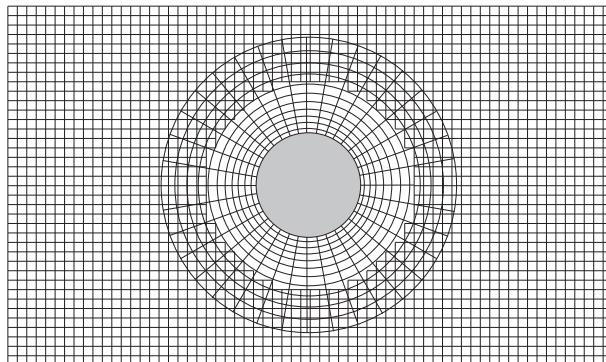


Figure 3.5 Illustration of the Chimera technique (2-D example).

interpolation order. The advantage of the Chimera technique over the multiblock approach is the possibility to generate the particular grids completely independent of each other, without having to take care of the interface between the grids. On the other hand, the problem of the Chimera technique is that the conservation properties of the governing equations are not satisfied through the overlapping region. An application of the approach to unstructured grids was reported recently in Refs. [47, 48].

The second type of grids are the unstructured grids. They offer the largest flexibility in the treatment of complex geometries [49]. The main advantage of the unstructured grids is based on the fact that triangular (in 2D) or tetrahedral grids (in 3D) can in principle be generated automatically, independent of the complexity of the domain. In practice, it is of course still necessary to set some parameters appropriately, in order to obtain a good quality grid. Furthermore, in order to resolve the boundary layers accurately, it is advisable to employ in 2D rectangular and in 3D prismatic or hexahedra elements near solid walls [50–60]. Another benefit of such mixed grids is the reduction of the number of grid cells, edges, faces, and possibly also of grid points. One should keep in mind though that the generation of mixed grids is not trivial for geometrically demanding cases. Nevertheless, the time required to built an unstructured, mixed grid for a complex configuration is still significantly lower than what is necessary for a multiblock structured grid. Since the geometrical fidelity of the flow simulations is nowadays rapidly increasing, the ability to generate grids fast and with minimum user interaction becomes more and more important. This is particularly true in an industrial environment. Further advantage of unstructured grids is that solution dependent grid refinement and coarsening can be handled in a relatively native and seamless manner. To mention also the disadvantages of unstructured grid methods, one of them is the necessity to employ sophisticated data structures within the flow solver. Such data structures work with indirect addressing, which, depending on the computer hardware, leads to more or less reduced computational efficiency. The memory requirements are in general higher as compared to the structured schemes, too. But despite all these difficulties, the capability to handle geometrically complex problems in short turn-around times still weights much more. From this point, it is not surprising that, for example, nearly all vendors of commercial CFD software switched over to unstructured flow solvers. A detailed review of various methodologies for spatial and temporal discretization on unstructured grids appeared in Ref. [61].

Having generated the grid, the next question is how to actually discretize the governing equations. As we have already mentioned, we can choose basically between three methodologies: finite differences, finite volumes, and finite elements schemes. We will discuss all of them briefly in the following sections.

3.1.1 Finite-difference method

The finite-difference method was among the first approaches applied to the numerical solution of differential equations. It was first utilized by Euler, probably in 1768. The finite-difference method is applied directly to the differential form of the governing equations. The principle is to employ a Taylor series expansion for the discretization of the derivatives of the flow variables. Let us for illustration consider the following example.

Suppose we would like to compute the first derivative of a scalar function $U(x)$ at some point x_0 . If we develop now $U(x_0 + \Delta x)$ as a Taylor series in x , we obtain

$$U(x_0 + \Delta x) = U(x_0) + \Delta x \frac{\partial U}{\partial x} \Big|_{x_0} + \frac{\Delta x^2}{2} \frac{\partial^2 U}{\partial x^2} \Big|_{x_0} + \dots \quad (3.1)$$

With this, the first derivative of U can be approximated as

$$\frac{\partial U}{\partial x} \Big|_{x_0} = \frac{U(x_0 + \Delta x) - U(x_0)}{\Delta x} + \mathcal{O}(\Delta x). \quad (3.2)$$

The above approximation is of first order, since the *truncation error* (abbreviated as $\mathcal{O}(\Delta x)$), which is proportional to the largest term of the remainder, goes to zero with the first power of Δx (for a discussion on the order of accuracy see Chapter 10). The same procedure can be applied to derive more accurate finite-difference formulas and to obtain approximations to higher-order derivatives.

An important advantage of the finite-difference methodology is its simplicity. Another advantage is the possibility to easily obtain high-order approximations, and hence to achieve high-order accuracy of the spatial discretization. On the other hand, because the method requires a structured grid, the range of application is clearly restricted. Furthermore, the finite-difference method cannot be directly applied in body-fitted (curvilinear) coordinates, but the governing equations have to be first transformed into a Cartesian coordinate system—or in other words—from the physical to the computational space (Fig. 3.2). The problem herewith is that the Jacobian coordinate transformation appears in the flow equations (see, e.g., Section A.1). This Jacobian has to be discretized consistently in order to avoid the introduction of additional numerical errors [62]. Thus, the finite-difference method is directly applicable only to rather simple geometries. Nowadays, it is utilized in the research of turbulent flows and, together with immersed boundary cells (Fig. 3.1b), in biology. More details to the finite-difference method can be found, for example, in [63], or in textbooks on the solution of partial differential equations.

3.1.2 Finite-volume method

The finite-volume method directly utilizes the conservation laws—the integral formulation of the Navier-Stokes/Euler equations. It was first employed by McDonald [64] for the simulation of 2-D inviscid flows. The finite-volume method discretizes the governing equations by first dividing the physical space into a number of arbitrary polyhedral control volumes. The surface integral on the right-hand side of Eq. (2.19) is then approximated by the sum of the fluxes crossing the individual faces of the control volume. The accuracy of the spatial discretization depends on the particular scheme with which the fluxes are evaluated.

There are several possibilities of defining the shape and position of the control volume with respect to the grid. Two basic approaches can be distinguished:

- *Cell-centered* scheme (Fig. 3.6a): Here the flow quantities are stored at the centroids of the grid cells. Thus, the control volumes are identical to the grid cells.
- *Cell-vertex* scheme (Fig. 3.6b): Here the flow variables are stored at the grid points. The control volume can then either be the union of all cells sharing the grid point, or some volume centered around the grid point. In the former case we speak of *overlapping* control volumes, in the second case of *dual* control volumes.

We shall discuss the pros and cons of cell-centered and cell-vertex formulations in the both chapters on spatial discretization (4 and 5).

The main advantage of the finite-volume method is that the spatial discretization is carried out directly in the physical space. Thus, there are no problems with any kind of transformation between the physical and the computational coordinate system, like in the case of the finite-difference method. Compared to the finite difference method, one further advantage of the finite-volume method is that it is very flexible—it can be rather easily implemented on structured as well as on unstructured grids. This renders the finite-volume method particularly suitable for the simulation of flows in or around complex geometries.

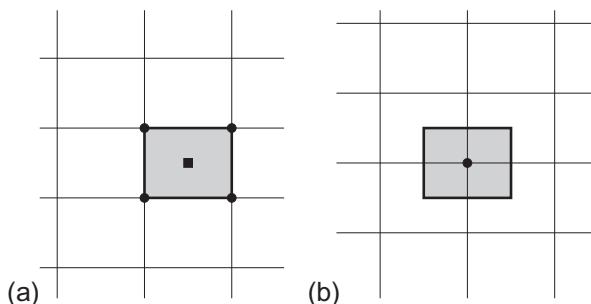


Figure 3.6 Control volume of cell-centered (a) and cell-vertex (dual control volume) scheme (b).

Since the finite-volume method is based on the direct discretization of the conservation laws, mass, momentum, and energy are also conserved by the numerical scheme. This leads to another important feature of the approach, namely the ability to compute *weak solutions* of the governing equations correctly. However, one additional condition has to be fulfilled in the case of the Euler equations. This is known as the *entropy condition*. It is necessary because of the non-uniqueness of the weak solutions. The entropy condition prevents the occurrence of unphysical features like expansion shocks, which violate the second law of thermodynamics (by decrease of the entropy). As a further consequence of the conservative discretization, the Rankine-Hugoniot relations, which must hold across a solution discontinuity (such as a shockwave or a contact discontinuity), are satisfied directly.

It is interesting to note that under certain conditions, the finite-volume method can be shown to be equivalent to the finite-difference method, or to a low-order finite-element method. Because of its attractive properties, the finite volume method is nowadays very popular and in wide use. It will be presented in detail in the following chapters.

3.1.3 Finite-element method

The finite-element method was originally employed for structural analysis only. It was first introduced by Turner et al. [65] in 1956. About 10 years later, researchers started to use the finite-element method also for the numerical solution of field equations in continuous media. However, only with the beginning of the 1990s, did the finite-element method gain popularity in the solution of the Euler and the Navier-Stokes equations. A good introduction into the classical finite-element methodology can be found in Ref. [66]. Applications to flow problems were described in Refs. [67–70]. Several years ago, the focus shifted to a variant of the method known as the *discontinuous Galerkin scheme* [71–98]. Higher-order formulations of the scheme are particularly suitable for aeroacoustic applications.

The finite-element method, as it is in general applied to the solution of the Euler/Navier-Stokes equations, starts with a subdivision of the physical space into discrete elements (mostly triangular or quadrilateral shapes in 2D, and tetrahedral or hexahedral shapes in 3D). Thus, an unstructured grid has to be generated. Depending on the element type and the required accuracy, a certain number of points at the boundaries and/or inside an element is specified, where the solution of the flow problem has to be found. The total number of points multiplied with the number of unknowns determines the number of *degrees of freedom*. Furthermore, the *shape functions* have to be defined, which represent the variation of the solution inside each element. In practical implementations, linear or higher-order elements are employed (linear elements use just the grid nodes). The shape functions are then either linear or higher-order distributions,

whose value is zero outside the corresponding element. This results in second- and higher-order accurate representation of the solution on regular grids.

Within the finite-element method, it is necessary to transform the governing equations from the differential into an equivalent integral form. This can be accomplished in two different ways. The first one is based on the *variational principle*, that is, a physical solution is sought, for which a certain functional possesses an extremum. The second possibility is known as the *method of weighted residuals* or the *weak formulation*. Here, it is required that the weighted average of the residuals is identically zero over the physical domain. The residuals can be viewed as the errors of the approximation of the solution. The weak formulation has the same advantage as the finite-volume discretization of the conservation laws—it allows the treatment of discontinuities such as shocks. Therefore, the weak formulation is preferred over the variational methodology. The discontinuous Galerkin scheme differs from the classical finite-element method with respect to the mass matrix, which is defined as being local to the generating element. This makes it possible for the discontinuous Galerkin scheme to temporally evolve the unknown solution coefficients by using simple explicit methods.

The finite-element method is attractive because of its integral formulation and the use of unstructured grids, which are both preferable for flows in or around complex geometries. Besides aeroacoustics, the method is also particularly suitable for the treatment of non-Newtonian fluids. The finite element method has a very rigorous mathematical foundation, particularly for elliptic and parabolic problems. Although it can be shown that in certain cases the method is mathematically equivalent to the finite-volume discretization, the numerical effort is noticeably higher. This may explain why the finite volume method became more popular for standard flow problems. However, both methods are sometimes combined—particularly on unstructured grids. So, for example, the treatment of the boundaries and the discretization of the viscous fluxes is usually “borrowed” from the finite-element method.

3.1.4 Other discretization methods

There are few other numerical schemes which are, in certain situations, superior to the methods discussed above. Three particular approaches should be mentioned here briefly.

Spectral-element method

The first such example is the *spectral-element method* [99–107]. This methodology combines the geometrical flexibility of the finite-element technique with the high-order spatial accuracy (e.g., 10th-order) and the rapid convergence rate of the spectral schemes [101]. The method is based on a high-order polynomial representation of the solution (usually Lagrangian interpolants), combined with a standard Galerkin

finite-element method, or the method of weighted residuals. The spectral-element method is appropriate either for a particular problem in which high-order regularity is guaranteed, or for which high-order regularity is not the exception, like in incompressible fluid mechanics. It is especially suitable for vortex flows. The advantage of the spectral-element method is primarily its non-diffusive, non-dispersive approximation of the convection operator, and its good approximation of convection-diffusion boundary layers. The method can treat geometrically and physically complex problems, supposed the condition of high-order regularity is fulfilled. Apart from the rather narrow range of applications, the principal disadvantage of the spectral-element method is its very high numerical effort as compared, for example, to the finite-volume approach.

Lattice Boltzmann method

Formulated in the late 1990s as a successor of the lattice gas methods, the *lattice Boltzmann method* (LBM) [108–129] gained popularity with the increasing availability of powerful, massively parallel computer systems. Unlike the previously discussed approaches, which are based on the conservation of macroscopic properties (i.e., mass, momentum, and energy), the LBM models the fluid as consisting of fictive particles. The particles perform repetitively microscopic propagation and collision processes over a discrete lattice of grid cells. The approach is based on kinetic theory and can be considered as a special discretized form of the continuous Boltzmann equation. In general, the LBM computes locally the effect of collisions as a relaxation process (the collision step) and the results—the discrete distribution functions—are sent to the respective neighboring points along characteristic directions (the streaming step). The macroscopic fields are then obtained by taking the discrete moments of the distribution functions. In addition, the pressure field is directly available from the density distributions. It should be mentioned that based on the Chapman-Enskog theory, one can recover the governing continuity and Navier-Stokes equations from the LBM algorithm. Recent publication also demonstrated interesting similarities between the LBM and a finite-difference method and compared both approaches in terms of computational efficiency [123].

Due to its particulate kinetics and local dynamics, the LBM offers several advantages over conventional CFD methods, especially when it comes to complex boundaries or to the incorporation of microscopic interactions. For example, the simulation of multiphase/multicomponent flows is very challenging because of the moving and deforming interfaces, and even more so because of transitional boundaries between the different phases. However, the formulation of the LBM provides a relatively easy and consistent way to tackle such problems by modifying the collision operator. Phase separations are generated automatically from the particle dynamics and no special treatment is required to manipulate the interfaces. Therefore, the LBM was successfully

applied to simulate interface instabilities, bubble/droplet dynamics, wetting on solid surfaces, or droplet electro-hydrodynamic deformations. It is also utilized for multiphase flows through porous media, in aeroacoustics, or in microfluidics.

The “collide and stream” algorithm renders the LBM highly suitable for efficient parallel processing using a domain-decomposition technique. On the other hand, traditional LBM is time marching in nature (either in physical or in pseudo time) and it is prone to a rather slow convergence rate to the desired (possibly stationary) solution. Also, there is so far no formulation of the LBM for flows with strong compressibility effects. This means that transonic or supersonic flows cannot be simulated with the LBM.

Gridless method

Another discretization scheme, which gained recently some interest, is the *gridless method* [130–136]. This approach employs only clouds of points for the spatial discretization. It does not require that the points are connected to form a grid as in conventional structured or unstructured grid schemes. The gridless method is based on the differential form of the governing equations, written in the Cartesian coordinate system. Gradients of the flow variables are determined by a least-squares reconstruction, using a specified number of neighbors surrounding the particular point. The gridless method is neither a finite-difference, finite-volume nor a finite-element approach since coordinate transformations, face areas or volumes do not have to be computed. It can be viewed as a mix between the finite-difference and the finite-element method. The principal advantages of the gridless approach are its flexibility in solving flows about complex configurations (similar to unstructured methods), and the possibility to locate or cluster the points (or the clouds of points) where it is appropriate. For example when computing gradients, it would be easily possible to select just the neighbors in the characteristic directions. However, there is one unresolved problem. Although the gridless method solves the conservation law form of the Euler or the Navier-Stokes equations, it is not quite clear whether conservation of mass, momentum, and energy is really guaranteed.

Whichever spatial discretization scheme we might select, it is important to ensure that the scheme is consistent, that is, that it converges to the solution of the **discretized** equations, when the grid is sufficiently refined. It is therefore very important to check how much the solution changes, if the grid is refined (e.g., if we double the number of grid points). If the solution improves only marginally, we speak of *grid converged* solution. Another rather self-evident requirement is that the discretization scheme should possess the order of accuracy, which is appropriate for the flow problem being solved. This rule is sometimes given up in favor of faster convergence, particularly in industrial environment (a bad solution is better than no solution). This is of course a very dangerous practice. We shall return to the question of accuracy, stability, and consistency later in Chapter 10.

3.1.5 Central and upwind schemes

So far, we discussed only the basic choices which exist for the spatial discretization. But within each of the above three main methods—finite difference, finite volume, and finite element—various numerical schemes exist to perform the spatial discretization. In this context, it is convenient to differentiate between the discretization of the convective and the viscous fluxes (\vec{F}_{mathrmc} and \vec{F}_V in Eq. (2.19), respectively). Because of the physical nature of the viscous fluxes, the only reasonable way is to employ central differences (central averaging) for their discretization. Thus, their discretization on structured grids is straightforward. On unstructured triangular or tetrahedral grids, the viscous fluxes are best approximated using the Galerkin finite element methodology, even in the case of a finite volume scheme [137]. The situation becomes more complicated for unstructured mixed grids, where a modified averaging of the gradients is more appropriate [138–142].

However, the real variety is found in the discretization of the convective fluxes. In order to classify the individual methodologies, we will restrict our attention to schemes developed for the finite-volume method, although most of the concepts are also directly applicable to the finite-difference or to the finite-element method.

Central schemes

To the first category we may count schemes, which are based solely on central difference formulas or on central averaging, respectively. These are denoted as *central schemes*. The principle is to average the conservative variables to the left and to the right in order to evaluate the flux at a side of the control volume. Since the central schemes cannot recognize and suppress an odd-even decoupling of the solution (i.e., the generation of two independent solutions of the **discretized** equations), the so-called *artificial dissipation* (because of its similarity to the viscous terms) has to be added for stabilization. The most widely known implementation is due to Jameson et al. [143]. On structured grids, it is based on a blend of 2nd- and 4th-differences scaled by the maximum eigenvalue of the convective flux Jacobian. A combination of an undivided Laplacian and biharmonic operator is employed on unstructured grids [144]. The scheme can be improved remarkably using different scaling factors for each equation. This approach is known as the *matrix dissipation* scheme [145]. It should be mentioned that on unstructured, mixed element grids the explicit Runge-Kutta time-stepping scheme can become unstable, when combined with the conventional central scheme [146].

Upwind schemes

On the other hand, there are more advanced spatial discretization schemes, which are constructed by considering the physical properties of the Euler equations. Because they

distinguish between upstream and downstream influences (wave propagation directions), they are termed *upwind schemes*. They can be divided roughly into four main groups:

- *flux-vector splitting*,
- *flux-difference splitting*,
- *total variation diminishing* (TVD),
- *fluctuation-splitting* schemes.

Each of these is described briefly in the following paragraphs.

Flux-vector splitting schemes

One class of the flux-vector splitting schemes decomposes the vector of the convective fluxes into two parts according to the sign of certain characteristic variables, which are in general similar but not identical to the eigenvalues of the convective flux Jacobian. The two parts of the flux vector are then discretized by upwind biased differences. The very first flux-vector splitting schemes of this type were developed in the beginning of the 1980s by Steger and Warming [147] and by Van Leer [148], respectively. A second class of flux-vector splitting schemes decomposes the flux vector into a convective and a pressure (an acoustic) part. This idea is utilized by schemes like AUSM (*advection upstream splitting method*) of Liou et al. [149, 150], or the CUSP scheme (*convective upwind split pressure*) of Jameson [151, 152], respectively. Further similar approaches are the *low-diffusion flux-splitting scheme* (LDFSS) introduced by Edwards [153], or the *mach number-based advection pressure splitting* (MAPS) scheme of Rossow [154, 155]. The second group of flux-vector splitting schemes gained recently larger popularity particularly because of their improved resolution of shear layers, but only a moderate computational effort. An advantage of the flux-vector splitting schemes is also that they can be quite easily extended to real gas flows, as opposed to flux-difference splitting or to TVD schemes. We shall return to real gas simulations further below.

Flux-difference splitting schemes

The second group—flux-difference splitting schemes—is based on the solution of the locally one-dimensional Euler equations for discontinuous states at an interface. This corresponds to the Riemann (shock tube) problem. The values on either side of the interface are generally termed the *left* and *right* state. The idea to solve the Riemann problem at the interface between two control volumes was first introduced by Godunov [156] back in 1959. In order to reduce the numerical effort required for an exact solution of the Riemann problem, *approximate* Riemann solvers were developed, for example, by Osher and Solomon [157] and Roe [158]. Roe's solver is often used today because of its excellent resolution of boundary layers and a crisp representation of shocks. It can be implemented easily on structured as well as on unstructured grids [159].

Further approaches based on an approximate solution of the Riemann problem are known as *Harten-Lax-van Leer* (HLL) solvers. One variant (HLLE) proposed by

Einfeldt [160] is based on a stability theory for discontinuities in fluids and takes into account the smallest and the largest wave speeds at the interface. Another version named HLLC (HLL contact) was introduced by Toro et al. [161] in order to restore the contact surface in the HLL scheme. It was later investigated and modified by other researchers [162–164]. One further approach designated as the *rotated-hybrid Riemann solver* was suggested by Nishikawa and Kitamura [165] to overcome the *carbuncle phenomenon*¹ of the original Roe scheme and the excessive diffusion of the HLL solvers at the same time. It should be noted that the carbuncle problem can be easily prevented by a simple modification to the Roe scheme known as *Harten's entropy correction*. We shall present the modification when discussing the Roe scheme in Section 4.3.3. Taking this into account, it is hard to see any real benefits from replacing the Roe upwind scheme by one of the HLL variants.

TVD Schemes

The idea of TVD schemes was first introduced by Harten [166] in 1983. TVD schemes are based on a concept aimed at preventing the generation of new extrema in the flow solution. The principal conditions for a TVD scheme are that the maxima must be non-increasing, the minima non-decreasing, and no new local extrema may be created. Such a scheme is called *monotonicity preserving*. Thus, a discretization methodology with TVD properties is capable of resolving a shock wave without any spurious oscillations of the solution. The TVD schemes are in general implemented as an average of the convective fluxes combined with an additional dissipation term. The dissipation term can either depend on the sign of the characteristic speeds or not. In the first case, we speak of an *upwind* TVD scheme [167], in the second case of a *symmetric* TVD scheme [168]. The experience shows that the upwind TVD scheme should be preferred, since it offers a better shock and boundary layer resolution than the symmetric TVD scheme. Certain disadvantage of the TVD schemes is that they cannot be easily extended to higher than second-order spatial accuracy.

Fluctuation-splitting schemes

The last group—the fluctuation-splitting schemes—provides for true multidimensional upwinding. The aim is to resolve accurately also those flow features which are not aligned with the grid. This is a significant advantage over all above upwind schemes, which split the equations according only to the orientation of the grid cells. Within the fluctuation-splitting methodology, the flow variables are associated with the grid nodes. Intermediate residuals are computed as flux balances over the grid cells, which consists of triangles in 2D and of tetrahedra in 3D. The cell-based residuals are then distributed in an upwind-biased manner to the nodes. After that, the solution is updated

¹ The carbuncle phenomenon is perturbation growing ahead of a strong bow shock along the stagnation line. This happens because the scheme fails to recognize the sonic point.

using the nodal values. In the case of systems of equations (Euler or Navier-Stokes), the cell-based residuals have to be decomposed into scalar waves. Since the decomposition is not unique in 2D and in 3D, several approaches were developed in the past. The variety reaches from the wave model of Roe [169, 170] over the algebraic scheme of Sidilkover [171] to the most advanced characteristic decomposition methods [172–176]. Despite the above mentioned advantage over the dimensionally split Riemann, TVD, etc. solvers, the fluctuation-splitting approaches are so far used only in research codes. This can be attributed to their complexity and the high numerical effort, as well as to convergence problems.

Solution reconstruction

All the above schemes require the knowledge of flow variables like density, velocity components, or pressure at the boundaries of the control volume, possibly separately for each side of the interface as the left and right states. Since only the mean values of the flow variables are known for each control volume, some kind of interpolation—we speak of *reconstruction*—is necessary. Clearly, the accuracy of the solution reconstruction determines the spatial order of the discretization scheme.

First- and second-order schemes

In the case of the central schemes, simple arithmetic averaging of the mean values from the control volumes to the left and to the right of the common interface is sufficient for second-order accuracy (cf. Section 4.3.1). In the case of the upwind schemes, setting the left and right state equal to the respective mean values results in a first-order accurate scheme. To achieve second order in space, one-sided differences are required on structured grids (see Section 4.3, Eq. (4.46)). Implementation of a second-order upwind scheme on unstructured grids requires first the computation of gradients of the flow variables. Then, using the gradients and the mean values, the left and right state are interpolated by directional derivatives (Section 5.3.3).

ENO/WENO Schemes

Higher than second-order accuracy can be achieved by using either the ENO (*essentially non-oscillatory*) or WENO (*weighted* ENO) discretization schemes [177–197]. These approaches are based on the idea to employ varying discretization stencils (as to their orientation) in order to compute different reconstruction (interpolation) polynomials for the values at the faces of the control volume. The ENO scheme then selects the polynomial with the minimal oscillation. The WENO methodology weights and combines the polynomials, the weights are inversely proportional to the amount of oscillations. In this way, the solution can be spatially of high order in smooth regions, while true discontinuities do not get excessively smoothed. A good introduction to WENO schemes was given in Ref. [185].

Central versus upwind schemes

You may ask now, what are the benefits and the drawbacks of the individual spatial discretization methods. Generally speaking, central schemes require considerably lower numerical effort, and hence significantly less CPU time per evaluation, as compared to upwind schemes. On the other hand, upwind schemes are able to capture discontinuities much more accurately than central schemes can. Furthermore, because of their lower numerical diffusion, the upwind schemes can resolve boundary layers using less grid points. Particularly, Roe's flux-difference splitting scheme and the CUSP flux-vector splitting scheme allow a very accurate computation of boundary layers. The negative side of the upwind schemes emerges for second- or higher-order spatial accuracy. The problem is that the so-called *limiter functions* (or simply *limiters*) have to be employed in order to prevent the generation of spurious oscillations near strong discontinuities. Limiters are known to stall the convergence of an iteration scheme, because of their accidental switching in smooth flow regions. A remedy was suggested by Venkatakrishnan [198–200], which works satisfactorily for most practical cases. However, small disturbances in the solution have to be accepted. Another disadvantage of the limiter functions is that they require high computational effort, particularly on unstructured grids. We shall return to the question of limiters later on when discussing the upwind discretization schemes in more detail.

Upwind schemes for real gas flows

With respect to real gas simulations, and in particular to chemically reacting flows, several extensions of the upwind discretization schemes were presented. For the case of fluids in thermodynamic and chemical equilibrium, modifications of the Van Leer flux-vector splitting [148] and of Roe's approximate Riemann solver [158] were described in Refs. [201–203]. Formulations of the both upwind methods for the more complex case of flows with non-equilibrium chemistry and thermodynamics were provided in [204–209] and in the references cited therein. In particular, the articles [203, 207, 208], respectively, give a good overview of the methodologies employed for the upwind discretization schemes. A summary of the governing equations together with Jacobian and transformation matrices, which may be required by an upwind scheme, was presented in Ref. [210] for the case of chemically reacting flows, and in Ref. [211] for equilibrium real gases.

3.2 TEMPORAL DISCRETIZATION

As already mentioned in the beginning of this chapter, the prevailing number of numerical schemes for the solution of the Euler and the Navier-Stokes equations applies the method of lines, that is, a separate discretization in space and in time. This approach offers the largest flexibility, since different levels of approximation can be easily selected

for the convective and the viscous fluxes, as well as for the time integration—just as required by the problem to be solved. Therefore, we shall follow this methodology here. For the discussion of other methods, where time and space discretizations are combined, the reader is referred to Ref. [63].

When the method of lines is applied to the governing equations (2.19), it leads, written down for each control volume, to a system of coupled ordinary differential equations in time

$$\frac{d(\Omega \bar{M} \vec{W})}{dt} = -\vec{R}. \quad (3.3)$$

For clarity, we omitted any references to cell indexes. In Eq. (3.3), Ω denotes volume of the control volume and \vec{R} stands for the complete spatial (finite-volume) discretization including the source term—the *residual*. The residual is a non-linear function of the conservative variables \vec{W} . Finally, \bar{M} represents what is termed the *mass matrix*. For a cell-vertex scheme, it relates the average value of \vec{W} in the control volume to the point values at the associated interior node and at the neighboring nodes [212, 213]. In the case of a cell-centered scheme, the mass matrix can be replaced by the identity matrix, without compromising the temporal accuracy of the scheme. The same holds for a cell-vertex scheme applied on a **uniform** grid, since then the nodes coincide with the centroids of the control volumes. The mass matrix is a function of the grid only and couples the system of differential equations (3.3). For steady-state cases, where time accuracy is of no concern, the mass matrix can be “lumped,” that is, replaced by the identity matrix. In this way, the expensive inversion of \bar{M} can be avoided and the system (3.3) is decoupled. In this respect, it is important to realize that at the steady-state, solution accuracy is determined solely by the approximation order of the residual. Thus, the mass matrix becomes important only for cell-vertex (median-dual) schemes applied to unsteady flows.

If we assume a static grid, we may take the volume Ω and the mass matrix outside the time derivative. Then, we can approximate the time derivative by the following non-linear scheme [63]

$$\frac{\Omega \bar{M}}{\Delta t} \Delta \vec{W}^n = -\frac{\beta}{1 + \omega} \vec{R}^{n+1} - \frac{1 - \beta}{1 + \omega} \vec{R}^n + \frac{\omega \Omega \bar{M}}{(1 + \omega) \Delta t} \Delta \vec{W}^{n-1} \quad (3.4)$$

with

$$\Delta \vec{W}^n = \vec{W}^{n+1} - \vec{W}^n \quad (3.5)$$

being the solution correction. The superscripts n and $(n + 1)$ denote the time levels (n means the current one). Furthermore, Δt represents the time step. The scheme in Eq. (3.4) is second-order accurate in time if the condition

$$\beta = \omega + \frac{1}{2} \quad (3.6)$$

is fulfilled, otherwise the time accuracy is reduced to first-order. Depending on the settings of the parameters β and ω , we can obtain either *explicit* ($\beta = 0$) or *implicit* time-stepping schemes. We shall discuss these two main classes briefly in the following sections, and in more detail later in Chapter 6.

3.2.1 Explicit schemes

A basic explicit time-integration scheme is obtained by setting $\beta = 0$ and $\omega = 0$ in Eq. (3.4). In this case, the time derivative is approximated by a forward difference and the residual is evaluated at the current time level only (based on known flow quantities), that is, we have

$$\Delta \vec{W}^n = -\frac{\Delta t}{\Omega} \vec{R}^n, \quad (3.7)$$

where the mass matrix was lumped. This represents a *single-stage* scheme, because a new solution \vec{W}^{n+1} results from only one evaluation of the residual. The scheme in Eq. (3.7) is of no practical value, since it is stable only if combined with a first-order upwind spatial discretization.

Very popular are *multistage* time-stepping schemes (Runge-Kutta schemes), where the solution is advanced in several stages [143] and the residual is evaluated at intermediate states. Coefficients are used to weight the residual at each stage. The coefficients can be optimized in order to expand the stability region and to improve the damping properties of the scheme and hence its convergence and robustness [143, 214, 215]. Also, depending on the stage coefficients and the number of stages, a multistage scheme can be extended to second- or higher-order accuracy in time. Special Runge-Kutta schemes were also designed to preserve the properties of the TVD and ENO spatial discretization methods, while maximizing the allowable time step [216].

Explicit multistage time-stepping schemes can be employed in connection with any spatial discretization scheme. They can be easily implemented on serial, vector, as well as on parallel computers. Explicit schemes are numerically cheap, and they require only a modest amount of computer memory. On the other hand, the maximum permissible time step is severely restricted because of stability limitations. Particularly for viscous flows and highly stretched grid cells, the convergence to steady state slows down considerably. Furthermore, in the case of stiff equation systems (e.g., real gas simulation, turbulence models), or of stiff source terms, it can take extremely long to achieve the steady state. Or even worse, an explicit scheme may become unstable or lead to spurious stationary solutions [217].

If we are interested in steady-state solutions only, we can select from (or combine) several convergence acceleration methodologies. The first, and very common, technique is *local time-stepping*. The idea is to advance the solution in each control volume with the maximum allowable time step. As a result, the convergence to the steady state is considerably accelerated. However, the transient solutions are no longer temporally accurate. Another approach is the so-called *characteristic time-stepping*. Here, not only locally varying time steps are used, but also each equation (continuity, momentum, and energy equation) is integrated with its own time step. The potential of this concept was presented for 2-D Euler equations in Ref. [218]. A further acceleration technique, which is similar to the characteristic time-stepping is *Jacobi preconditioning* [219–221]. It is basically a point-implicit Jacobi relaxation, which is carried out at each stage of a Runge-Kutta scheme. Jacobi preconditioning can be seen as a time-stepping in which all wave components (eigenvalues of the flux Jacobian) are scaled to have the same effective speed. It also adds an implicit component to the basic explicit scheme.

Another very popular acceleration method is aimed at increasing the maximum possible time step by introducing a certain amount of implicitness in the explicit scheme. It is termed *implicit residual smoothing* or *residual averaging* [222, 223]. On a structured grid, the method requires the solution of a tridiagonal matrix for each conservative variable. In the case of unstructured grids, the matrix is usually inverted by means of Jacobi iteration. The standard implicit residual smoothing allows for an increase of the time step by a factor of 2–3. Several other implicit residual smoothing techniques were developed. For example, the *upwind implicit residual smoothing* methodology [224], which was designed to be employed together with an upwind spatial discretization. In comparison to the standard technique, it allows for significantly larger time steps and it also improves the robustness of the time-stepping process [225]. One further method is the *implicit-explicit residual smoothing* [226, 227], which is intended to improve the damping properties of the time discretization at larger time steps.

The last but likely the most important convergence acceleration technique that should be mentioned here is the *multigrid method*. It was developed in the 1960s in Russia by Fedorenko [228] and Bakhvalov [229]. They applied multigrid for the solution of elliptic boundary-value problems. The methodology was further advanced and promoted by Brandt [230, 231]. The idea of multigrid is based on the observation that iterative schemes usually eliminate high-frequency errors in the solution (i.e., oscillations between the control volumes) very effectively. On the other hand, they perform quite poor in reducing low-frequency (i.e., global) solution errors. Therefore, after advancing the solution on a given grid, it is transferred to a coarser grid, where the low-frequency errors become partly high-frequency ones, and where they are again effectively damped by an iterative solver. The procedure is repeated recursively on a sequence of progressively coarser grids, where each *multigrid level* helps to suppress a certain bandwidth of error

frequencies. After the coarsest grid is reached, the solution corrections are successively collected and interpolated back to the initial fine grid, where the solution is then updated. This complete *multigrid cycle* is repeated until the solution changes less than a given threshold. In order to accelerate the convergence even further, it is possible to start the multigrid process on a coarse grid, carry out a number of cycles and then to transfer the solution to a finer grid, where the multigrid cycles are performed again. The procedure is then successively repeated until the finest grid is reached. This methodology is known as *full multigrid* (FMG) [231].

As already mentioned, the multigrid method was originally developed for the solution of elliptic boundary-value problems (Poisson equation), where it is very efficient. Jameson first proposed to employ multigrid also for the solution of the Euler equations [222, 232]. The approach was based on the so-called *full approximation storage* (FAS) scheme [231], where multigrid is applied directly to the non-linear governing equations. Nowadays, multigrid represents a standard acceleration technique for the solution of the Navier-Stokes equations. Examples of implementations can be found in Refs. [233–239] for structured grids, and in Refs. [240–250] for unstructured grids. Although not as fast as in the case of elliptic differential equations, it was often demonstrated that multigrid can accelerate the solution of the Euler or the Navier-Stokes equations by a factor between 5 and 10. An example for transonic flow is shown in Fig. 3.7. Recent research also revealed that faster convergence can be achieved if the governing equations are decomposed into hyperbolic and elliptic parts [251]. We shall return to the multigrid methodology again in Section 9.4.

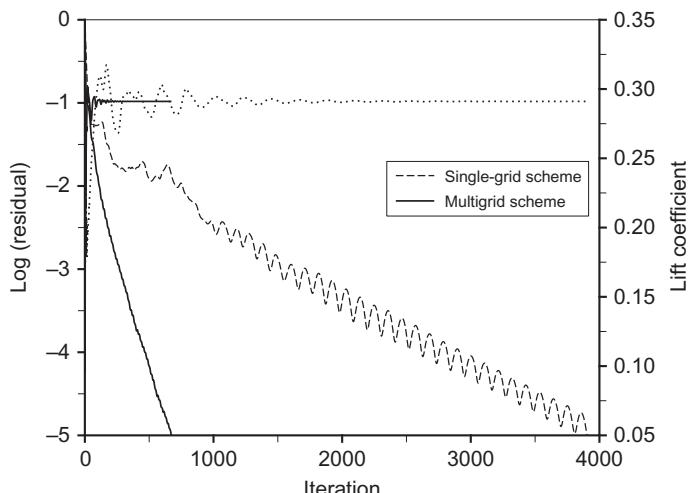


Figure 3.7 Comparison of convergence histories for inviscid flow past a transonic wing.

3.2.2 Implicit schemes

A family of implicit time integration schemes is obtained from Eq. (3.4) by setting $\beta \neq 0$. Very popular for the simulation of unsteady flows is the 3-point implicit backward-difference scheme with $\beta = 1$ and $\omega = 1/2$, which is second-order accurate in time. In this case, the scheme is mostly employed within the so-called *dual time-stepping* approach [212, 213, 252–254], where a steady-state problem is solved in pseudo-time at each physical time step.

For the solution of stationary flow problems, a scheme with $\omega = 0$ is more suitable, since it requires less computer memory. Herewith, if we linearize the residual \vec{R}^{n+1} in Eq. (3.4) about the current time level, we obtain the scheme

$$\left(\bar{M} \frac{\Omega}{\Delta t} + \beta \frac{\partial \vec{R}}{\partial \vec{W}} \right) \Delta \vec{W}^n = -\vec{R}^n. \quad (3.8)$$

The term $\partial \vec{R} / \partial \vec{W}$ is denoted as the *flux Jacobian*. It constitutes a large sparse matrix. The expression enclosed in parenthesis on the left-hand side of Eq. (3.8) is also referred to as the *implicit operator*. As already discussed above, the mass matrix \bar{M} can be replaced by the identity matrix without influencing the steady state solution. The parameter β in Eq. (3.8) is generally set to 1, which results in a first-order accurate temporal discretization. A second-order time accurate scheme is obtained for $\beta = 1/2$. However, this is not advised since the scheme with $\beta = 1$ is much more robust, and the time accuracy plays no role for steady problems anyway.

The principal advantage of implicit schemes, as compared to explicit ones, is that significantly larger time steps can be used, without hampering the stability of the time integration process. In fact, for $\Delta t \rightarrow \infty$ the scheme (3.8) transforms into standard Newton's method, which exhibits quadratic convergence. However, the condition for quadratic convergence is that the flux Jacobian contains the complete linearization of the residual. Another important advantage of implicit schemes is their superior robustness and convergence speed in the case of stiff equation systems and/or source terms, which are often encountered in real gas simulations, turbulence modeling, or in the case of highly stretched grids (high Reynolds number flows). On the other hand, the faster (in terms of time steps or iterations) and the more robust an implicit scheme is, the higher is usually the computational effort per time step or iteration. Therefore, an explicit scheme accelerated by multigrid can be equally or even more efficient. Furthermore, implicit schemes are significantly more difficult to vectorize or to parallelize than their explicit counterparts.

Written down for each control volume, the implicit scheme in Eq. (3.8) represents a large system of linear equations, which has to be solved for the update $\Delta \vec{W}^n$ at each time step Δt . This task can be accomplished using either a *direct* or an *iterative* solution method.

Direct methods are based on the exact inversion of the left-hand side of Eq. (3.8) using either the Gaussian elimination scheme or some direct sparse matrix method [255, 256]. Although quadratic convergence was demonstrated on structured [257–260] as well as on unstructured grids [261], direct methods are not an option for 3-D problems because they require an excessively high computational effort and a huge amount of computer memory.

Thus, the only practical approach for larger grids or for 3-D problems are iterative methods. Here, the linear system is solved for $\Delta \vec{W}^n$ at each time step using some iterative matrix inversion methodology. In order to reduce the memory requirements and also to increase the diagonal dominance, the flux Jacobian $\partial \vec{R} / \partial \vec{W}$ is mostly based on linearization of a first-order accurate spatial discretization of the right-hand side. The two main consequences of this approximation are that the quadratic convergence of Newton's scheme cannot be achieved and that the maximum time step becomes limited. On the other hand, the numerical effort of each iterative step is significantly reduced, which leads to a computationally efficient scheme.

In the case of structured grids, iterative methods like the *alternating direction implicit* (ADI) scheme [262–265], the (line) Jacobi or the Gauss-Seidel relaxation scheme [266–270], and particularly the *lower-upper symmetric Gauss-Seidel* (LU-SGS; also referenced to as LU-SSOR—lower-upper symmetric successive overrelaxation) scheme [271–275] are mainly employed. All these methods are based on splitting of the implicit operator into a sum or a product of parts, which can each be inverted more easily. Because of the associated *factorization error* (the difference with respect to the original matrix) and also the simplification of the flux Jacobian, it does not pay off to solve the linear system very accurately. In fact, only one iteration is carried out at each time step of the ADI and the LU-SGS method.

Implicit iterative methods for unstructured grids are in the most cases based on the Gauss-Seidel relaxation scheme [276–279]. In order to improve the convergence, it is possible to use the red-black Gauss-Seidel methodology. Its extension to unstructured grids was demonstrated in Refs. [280–282]. A particularly interesting possibility is also offered by an implementation of the LU-SGS scheme on unstructured grids [27, 283, 284], because of its very low memory requirements and numerical effort.

Because of the success of the line-implicit methods on structured grids, a few attempts were made to adopt this methodology on unstructured grids [285, 286]. The approach was to construct continuous lines such that each grid point or each grid cell (in the case of a cell-centered scheme) is visited only once—the *Hamiltonian tour* [287]. The lines were oriented primarily in coordinate directions, but they were folded at the boundaries and where else necessary (therefore they were nicknamed as “snakes”). A tri-diagonal solver was then employed to invert the left-hand side of Eq. (3.8). Later on, it was recognized that folding the lines can slow down the convergence. To overcome this, each line was broken up into multiple *linelets* [288]. However, the performance on a vector computer

was rather poor. The idea of linelets was also employed to improve the convergence of an explicit scheme on highly stretched viscous unstructured grids using an implicit solver in the direction across the boundary layer [248].

More sophisticated iterative techniques, which treat the linear equation system in a more global way, are the so-called *Krylov subspace* methods. Their development was triggered by the introduction of an efficient iterative scheme for solving large, sparse linear systems—namely the *conjugate gradient* method by Hestenes and Stiefel [289]. The original conjugate gradient method is restricted to Hermitian positive definite matrices only, but for an $n \times n$ matrix it converges in at most n iterations. Since then, a variety of Krylov subspace methods was proposed for the solution of arbitrary non-singular matrices, as they occur in CFD applications. For example, there are methods like the *conjugate gradient squared* (CGS) [290], the *bi-conjugate gradient stabilized* (Bi-CGSTAB) [291], or the *transpose-free quasi-minimum residual* (TFQMR) [292] scheme.

However, probably the most widely employed method is the *generalized minimal residual* (GMRES) scheme developed by Saad and Schultz [293]. If we rewrite the implicit scheme in Eq. (3.8) as

$$\bar{J} \Delta \vec{W}^n = -\vec{R}^n, \quad (3.9)$$

then \bar{J} represents a large, sparse, and non-symmetric matrix (the left-hand side). Starting from an initial guess $\Delta \vec{W}_0$, the GMRES(m) method seeks a solution $\Delta \vec{W}^n$ in the form $\Delta \vec{W}^n = \Delta \vec{W}_0^n + \vec{y}_m$, where \vec{y}_m belongs to the Krylov subspace

$$\mathcal{K}_m \equiv \text{span}\{\vec{r}_0, \bar{J}\vec{r}_0, \bar{J}^2\vec{r}_0, \dots, \bar{J}^{m-1}\vec{r}_0\} \quad (3.10)$$

$$\vec{r}_0 = \bar{J} \Delta \vec{W}_0^n + \vec{R}^n,$$

such that the residual $\|\bar{J} \Delta \vec{W}^n + \vec{R}^n\|$ becomes a minimum. The parameter m specifies the dimension of the Krylov subspace, or in other words the number of *search directions* ($\bar{J}^i \vec{r}_0$). Since all directions have to be stored, m is usually chosen between 10 and 40, the higher number being necessary for poorly conditioned matrices (which arise in the simulation of turbulent flows, real gas, etc.). GMRES has to be restarted, if no convergence is achieved within m sub-iterations. The GMRES method requires significantly more memory than, for example, Bi-CGSTAB or TFQMR, but it is more robust, smoothly converging and usually also faster. A very detailed comparison of the various methodologies can be found in Ref. [294].

Nevertheless, as with other conjugate gradient methods, preconditioning is absolutely essential for CFD problems. Here, we solve

$$(\bar{P}_L \bar{J}) \Delta \vec{W}^n = -\bar{P}_L \vec{R}^n, \quad \text{or} \quad \bar{J} \bar{P}_R (\bar{P}_R^{-1} \Delta \vec{W}^n) = -\vec{R}^n \quad (3.11)$$

instead of the system in Eq. (3.9). The matrices \bar{P}_L and \bar{P}_R denote left and right preconditioners, respectively. The preconditioner should approximate \bar{J}^{-1} as close as

possible, in order to cluster the eigenvalues near unity. On the other hand, it should be of course easy to invert. One particularly efficient preconditioner is the *incomplete lower upper* factorization method [295] with zero fill-in (ILU(0)). For the discussion of different preconditioning techniques in connection with GMRES, the reader is referred to [198, 296–299].

Since the GMRES method requires a considerable amount of computer memory for storing the search directions and possibly also the preconditioning matrix, it is a good idea to circumvent an explicit formation and storage of the flux Jacobian $\partial \vec{R} / \partial \vec{W}$. This is offered by the so-called *matrix-free* approach. The idea is based on the observation that GMRES (and some other Krylov subspace methods) employs only matrix vector products of the form

$$\frac{\partial \vec{R}}{\partial \vec{W}} \Delta \vec{W}^n,$$

which can be simply approximated by finite differences as

$$\frac{\partial \vec{R}}{\partial \vec{W}} \Delta \vec{W}^n = \frac{\vec{R}(\vec{W} + \epsilon \Delta \vec{W}^n) - \vec{R}(\vec{W})}{\epsilon}, \quad (3.12)$$

thus requiring only residual evaluations. The parameter ϵ has to be chosen with some care, in order to minimize the numerical error (see, e.g., [300] or [301]). Another, and even more important, advantage of the matrix-free approach is that (numerically) accurate linearization of a high-order residual \vec{R}^n can be easily utilized in the implicit scheme. Hence, the quadratic convergence of Newton's scheme can be achieved at moderate costs. In this case we speak of *Newton-Krylov* approach [212, 301–305]. Practical experience indicates that from all Krylov subspace methods, GMRES is best suited for the matrix-free implementation [306]. An interesting possibility is to utilize the LU-SGS scheme as a preconditioner for the matrix-free GMRES method. Since the LU-SGS scheme also does not require an explicit storage of the flux Jacobian, the memory requirements can be reduced even further. The computational efficiency of this approach was demonstrated for 3-D inviscid and laminar flows on unstructured grids [307].

The convergence of an implicit scheme can also be enhanced by using multigrid. There are basically two possible ways. First, we can employ multigrid inside an implicit scheme—as a solver for the linear equation system (3.9) arising at each time step, or as a preconditioner for one of the conjugate gradient methods [308, 309]. Second, the implicit scheme itself can serve as a smoother within the FAS multigrid method, which is applied directly to the governing equations [282, 310–313]. Some investigations show that at least for purely aerodynamic problems, rather “simple” implicit schemes (like Gauss-Seidel) combined with multigrid result in computationally more efficient solvers (in terms of the CPU time) than, for example, GMRES [282, 302].

3.3 TURBULENCE MODELING

The solution of the governing equations (2.19) does not raise any fundamental difficulties in the case of inviscid or laminar flows. The simulation of turbulent flows, however, presents a significant problem. Despite the performance of modern supercomputers, a direct simulation of turbulence by the time-dependent Navier-Stokes equations (2.19), called the *direct numerical simulation* (DNS), is still possible only for rather simple flow cases at low Reynolds numbers (Re). The restrictions of the DNS become quite obvious when recalling that the number of grid points needed for sufficient spatial resolution scales as $Re^{9/4}$ and the CPU-time as Re^3 . This does not mean that DNS is completely useless. It is an important tool to understand the turbulent structures and the laminar-turbulent transition. DNS also plays a vital role in the development and calibration of new or improved turbulence models. However, in engineering applications, the effects of turbulence can be taken into account only approximately, using models of various complexities.

The first level of approximation is reached for the *large-eddy simulation* (LES) approach. The development of LES is founded on the observation that the small scales of turbulent motion posses a more universal character than the large scales, which transport the turbulent energy. Thus, the idea is to resolve only the large eddies accurately and to approximate the effects of the small scales by relatively simple *subgrid-scale models*. Since LES requires significantly less grid points than DNS, the investigation of turbulent flows at much higher Reynolds numbers becomes feasible. But because LES is inherently three-dimensional and unsteady, it still remains computationally very demanding. Thus, LES is still far away from becoming an engineering tool. However, LES is well suited for detailed studies of complex flow physics including massively separated unsteady flows, large scale mixing (e.g., fuel and oxidizer), aerodynamic noise, or for the investigation of flow control strategies. LES is also very promising for more accurate computations of flows in combustion chambers or engines, heat transfer and of rotating flows. An overview of research activities in LES was recently published in Ref. [314].

The next level of approximation is represented by the *Reynolds-averaged* Navier-Stokes equations (RANS). This approach, which was presented by Reynolds in 1895, is based on the decomposition of the flow variables into mean and fluctuating parts, followed by time or ensemble averaging [315] (see also [316, 317]). In cases where the density is not constant, it is advisable to apply the *density (mass) weighted* or Favre decomposition [318, 319] to the velocity components. Otherwise, the averaged governing equations would become considerably more complicated due to additional correlations involving density fluctuations. It is common to assume that Morkovin's hypothesis [320] is valid, which states that the turbulence structure of boundary layers and wakes is not notably influenced by density fluctuations for Mach numbers below 5.

By inserting the decomposed variables (mean and fluctuating parts) into the Navier-Stokes equations (2.19) and by averaging, we obtain formally the same equations for the mean variables with the exception of two additional terms. The tensor of the viscous stresses is extended by one term—the *Reynolds-stress tensor* [315]

$$\bar{\tau}_{ij}^R = -\bar{\rho} \widetilde{v_i'' v_j''}, \quad (3.13)$$

where v_i'', v_j'' denote the density-weighted fluctuating parts of the velocity components u, v, w ; $\bar{\cdot}$ and $\widetilde{\cdot}$ stand for ensemble and density weighted averaging, respectively. The Reynolds-stress tensor represents the transport of mean momentum due to turbulent fluctuations. Furthermore, the diffusive heat flux $k\nabla T$ in the energy equation (cf. Eq. (2.8)) is enhanced by the so-called *turbulent heat-flux vector* [63]

$$\vec{F}_D^T = -\bar{\rho} h'' \widetilde{\vec{v}''}. \quad (3.14)$$

Thus, we can see that the solution of the Reynolds-averaged Navier-Stokes equations requires the modeling of the Reynolds stresses (3.13) and of the turbulent heat flux (3.14). The advantages of this approach are that considerably coarser grids can be used as compared to LES, and that stationary mean solution can be assumed (at least for attached or moderately separated flows). Clearly, both features significantly reduce the computational effort in comparison to LES or even DNS. Therefore, the RANS approach is very popular in engineering applications. Of course, because of the averaging procedure, no detailed information can be obtained about the turbulent structures.

A large variety of turbulence models was devised to close the RANS equations and the research still continues. The models can be divided into *first-* and *second-order* closures, respectively.

The most complex, but also the most flexible, are second-order closure models. The *Reynolds-stress transport* (RST) model, which was first proposed by Rotta [321], solves modeled transport equations for the Reynolds-stress tensor. The partial differential equations for the six stress components have to be closed by one additional relation. Usually, an equation for the turbulent dissipation rate is employed. The RST models are able to account for strong nonlocal and history effects. Furthermore, they are able to capture the influence of streamline curvature or system rotation on the turbulent flow.

Closely related to the RST approach are the *algebraic Reynolds-stress* (ARS) models. They can be viewed as a combination of lower level models and the RST approach. The ARS models employ only two transport equations, mostly for the turbulent kinetic energy and the dissipation rate. The components of the Reynolds-stress tensor are related to the transport quantities by non-linear algebraic equations [322]. The ARS approach is capable of predicting rotational turbulent flows and secondary flows in channels with accuracy similar to the RST models. Detailed overviews of the RST and ARS models can be found in Refs. [323, 324].

Because of numerical problems with the RST and ARS models, which are primarily caused by the stiffness of the RST and the non-linearity of the ARS equations, first-order closures are more widely used in practice. In these models, the Reynolds stresses are expressed by means of a single scalar value, the *turbulent eddy viscosity*. This approach is based on the *eddy viscosity hypothesis* of Boussinesq [325, 326], which assumes a linear relationship between the turbulent shear stress and the mean strain rate, similar to laminar flow. Herewith, the dynamic viscosity μ in the viscous stress tensor (2.15) or in the governing equations (2.19) is replaced by the sum of a laminar and a turbulent component

$$\mu = \mu_L + \mu_T. \quad (3.15)$$

As described earlier, the laminar viscosity is computed, for example, with the aid of the Sutherland formula (2.30). In analogy, the turbulent heat-flux vector (3.14) is modeled as

$$\vec{F}_D^T = -k_T \nabla T, \quad (3.16)$$

where k_T denotes the *turbulent thermal conductivity coefficient*. Hence, the thermal conductivity coefficient in Eq. (2.24) is evaluated as

$$k = k_L + k_T = c_p \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T} \right). \quad (3.17)$$

The turbulent Prandtl number is in general assumed to be constant in the flow field ($Pr_T = 0.9$ for air). The coefficient of the turbulent eddy viscosity μ_T has to be determined with the aid of a turbulence model. The limitations of the eddy viscosity approach are given by the assumption of equilibrium between the turbulence and the mean strain field, and by the independence on system rotation. The accuracy of the eddy-viscosity based models can be significantly improved either by using correction terms [327, 328], or by employing *non-linear eddy viscosity* approaches [329–331].

The first-order closures can be categorized into *zero-*, *one-*, and *multiple-equation* models, corresponding to the number of transport equations they utilize. Within the zero-equation or, as they are also denoted, *algebraic models*, the turbulent eddy viscosity is calculated from empirical relations, which employ only local mean flow variables. Therefore, no history effects can be simulated, which prevents a reliable prediction of separated flows. The most popular algebraic model, which is still in use for some applications, was developed by Baldwin and Lomax [332].

History effects are taken into account by the one- and two-equation models, where the convection and the diffusion of turbulence is modeled by transport equations. The most widely used one-equation turbulence model is due to Spalart and Allmaras [333], which is based on an eddy-viscosity like variable. The model is numerically very stable and easy to implement on structured as well as on unstructured grids.

In the case of the two-equation models, practically all approaches employ the transport equation for the turbulent kinetic energy. Among a large number of two-equation models, the $K - \varepsilon$ model of Launder and Spalding [334] and the $K - \omega$ model of Wilcox [335] are most often used in engineering applications. They offer a reasonable compromise between computational effort and accuracy. An interesting comparison between the Spalart-Allmaras model and various two-equation turbulence models was published in Ref. [336]. Another useful comparison of turbulence models can be found in Ref. [337].

3.4 INITIAL AND BOUNDARY CONDITIONS

Regardless of which numerical methodology we may choose to solve the governing equations (2.19), we have to specify suitable *initial* and *boundary* conditions. The initial conditions determine the state of the fluid at the time $t = 0$, or at the first step of an iterative scheme. Clearly, the better (the closer to the solution) the initial guess will be, the faster the final solution will be obtained. Moreover, the probability of breakdown of the numerical solution process will be reduced correspondingly. It is therefore important that the initial solution satisfies at least the governing equations as well as the additional thermodynamic relations. A common practice in external aerodynamics consists of prescribing free-stream values of pressure, density and velocity components (given as Mach number, angle of attack, and side-slip angle) in the whole flow field. In turbomachinery, it is important to specify the flow directions in the complete domain to one's best knowledge. The same holds also for the pressure field. It is therefore quite worthwhile to employ lower-order approximations (like potential methods) to generate a physically meaningful initial guess.

Any numerical flow simulation considers only a certain part of the real physical domain. The truncation of the computational domain creates artificial boundaries, where values of the physical quantities have to be specified. Examples are the far-field boundary in external aerodynamics; the inlet, outlet, and the periodic boundary in the case of internal flows; and finally the symmetry plane. The main problem when constructing such boundary conditions is of course that the solution on the truncated domain should stay as close as possible to a solution which would be obtained for the whole physical domain. In the case of the far-field, inlet and outlet boundaries, *characteristic* boundary conditions [338–340] are often used in order to suppress the generation of non-physical disturbances in the flow field. But despite this, the far-field or the inlet and outlet boundaries must still not be placed too close to the object under consideration (wing, blade, etc.). Otherwise, the accuracy of the solution would be reduced. For external flows, when a lifting body is considered, it is possible to correct the flow variables at the far-field boundary using a single vortex centered at the airfoil or the wing [340–342]. In this way, the distance between the

body and the far-field boundary can be significantly reduced without impairing the solution fidelity, or the accuracy can be improved for a given outer boundary position [264, 342, 343]. For internal flow problems, formulations for the inlet and outlet boundaries based on linearized Euler equations and Fourier series expansion of the perturbations were developed [344–346]. These formulations allow for a relatively close placement of the inlet and outlet boundaries to a blade, without influencing the solution noticeably.

A different type of boundary condition is found when the surface of a body is exposed to the fluid. In the case of inviscid flow governed by the Euler equations (2.44), the appropriate boundary condition is to require the flow to be tangential to the surface, that is,

$$\vec{v} \cdot \vec{n} = 0 \quad \text{at the surface.}$$

By contrast, no relative velocity between the surface and the fluid immediately at the surface is assumed for the Navier-Stokes equations—the *no-slip* boundary condition

$$u = v = w = 0 \quad \text{at the surface.}$$

The treatment of walls becomes more involved in cases, where, for example, a specified wall temperature distribution has to be met, or when the heat radiation has to be taken into account (see, e.g., [347, 348]).

Furthermore, boundary conditions have to be defined for surfaces where different fluids (e.g., air and water) meet together [349–353]. But apart from the physical boundary conditions and those imposed by truncating the flow domain, there can be boundaries generated by the numerical solution method itself. These are for example coordinate cuts and block or zonal boundaries [32–38].

The correct implementation of boundary condition is the crucial point of every flow solver. Not only the accuracy of the solution depends strongly on a proper physical and numerical treatment of the boundaries, but also the robustness and the convergence speed are considerably influenced. More details of various important boundary conditions are presented in Chapter 8.

REFERENCES

- [1] Frymier PD, Hassan HA, Salas MD. Navier-Stokes calculations using cartesian grids: I. Laminar flows. *AIAA J* 1988;26:1181–8.
- [2] De Zeeuw D, Powell KG. An adaptively refined Cartesian mesh solver for the Euler equations. *J Comput Phys* 1993;104:56–68.
- [3] Coirier WJ, Powell KG. An accuracy assessment of Cartesian-mesh approaches for the Euler equations. *J Comput Phys* 1995;117:121–31.
- [4] Ye T, Mittal R, Udaykumar HS, Shyy W. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *J Comput Phys* 1999;156:209–40.
- [5] Kirshman D, Liu F. Cartesian grid solution of the Euler equations using a gridless boundary condition treatment. *AIAA Paper 2003-3974*; 2003.

- [6] Carolina L, Tsai HM, Liu F. An embedded Cartesian grid Euler solver with radial basis function for boundary condition implementation. AIAA Paper 2008-532; 2008.
- [7] Ishida T, Kawai S, Nakahashi K. A high-resolution method for flow simulations with Cartesian mesh method. AIAA Paper 2011-1296; 2011.
- [8] Uddin H, Kramer RMJ, Pantano C. A Cartesian-based embedded geometry technique with adaptive high-order finite differences for compressible flow around complex geometries. *J Comput Phys* 2014;262:379-407.
- [9] Richtmyer RD, Morton KW. *Difference Methods for Initial Value Problems*. 2nd ed. London: Wiley-Interscience; 1967.
- [10] Volpe G. Performance of compressible flow codes at low mach number. *AIAA J* 1993;31:49-56.
- [11] Harlow FH, Welch JE. Numerical calculation of time-dependent viscous incompressible flow with free surface. *Phys Fluids* 1965;8:2182-9.
- [12] Patankar SV. *Numerical heat transfer and fluid flow*. New York: McGraw-Hill; 1980.
- [13] Ho Y-H, Lakshminarayana B. Computation of unsteady viscous flow using a pressure-based algorithm. *AIAA J* 1993;31:2232-40.
- [14] Javadia K, Darbandia M, Taeibi-Rahni M. Three-dimensional compressible-incompressible turbulent flow simulation using a pressure-based algorithm. *Comput Fluids* 2008;37:747-66.
- [15] Darwish M, Sraj I, Moukalled F. A coupled finite volume solver for the solution of incompressible flows on unstructured grids. *J Comput Phys* 2009;228:180-201.
- [16] Shterev KS, Stefanov SK. Pressure based finite volume method for calculation of compressible viscous gas flows. *J Comput Phys* 2010;229:461-80.
- [17] Chen ZJ, Przekwas AJ. A coupled pressure-based computational method for incompressible/compressible flows. *J Comput Phys* 2010;229:9150-65.
- [18] Chorin AJ. A numerical method for solving incompressible viscous flow problems. *J Comput Phys* 1967;2:12-26.
- [19] Turkel E. Preconditioned methods for solving the incompressible and low speed compressible equations. *J Comput Phys* 1987;72:277-98.
- [20] Van Leer B, Lee WT, Roe PL. Characteristic time-stepping or local preconditioning of the euler equations. AIAA Paper 91-1552; 1991.
- [21] Choi YH, Merkle CL. The application of preconditioning in viscous flows. *J Comput Phys* 1993;105:207-33.
- [22] Turkel E. Review of preconditioning methods for fluid dynamics. *Appl Numer Math* 1993;12:257-84.
- [23] Weiss J, Smith WA. Preconditioning applied to variable and constant density flows. *AIAA J* 1995;33:2050-7.
- [24] Lee D. Local preconditioning of the Euler and Navier-Stokes equations. PhD Thesis, University of Michigan; 1996.
- [25] Jespersen D, Pulliam T, Buning P. Recent enhancements to OVERFLOW. AIAA Paper 97-0644; 1997.
- [26] Merkle CL, Sullivan JY, Buelow PEO, Venkateswaran S. Computation of flows with arbitrary equations of state. *AIAA J* 1998;36:515-21.
- [27] Sharov D, Nakahashi K. Low speed preconditioning and LU-SGS scheme for 3-D viscous flow computations on unstructured grids. AIAA Paper 98-0614; 1998.
- [28] Mulas M, Chibbaro S, Delussu G, Di Piazza I, Talice M. Efficient parallel computations of flows of arbitrary fluids for all regimes of Reynolds, Mach and Grashof numbers. *Int J Numer Meth Heat Fluid Flow* 2002;12:637-57.
- [29] Puoti V. Preconditioning method for low-speed flows. *AIAA J* 2003;41:817-30.
- [30] Briley WR, Taylor LK, Whitfield DL. High-resolution viscous flow simulations at arbitrary Mach number. *J Comput Phys* 2003;184:79-105.
- [31] Sockol PM. Multigrid solution of the Navier-Stokes equations at low speeds with large temperature variations. *J Comput Phys* 2003;192:570-92.
- [32] Lee KD, Rubbert PE. Transonic flow computations using grid systems with block structure. *Lecture Notes in Physics*, vol. 141. Berlin: Springer Verlag; 1981. p. 266-71.

- [33] Rossow C-C. Efficient computation of inviscid flow fields around complex configurations using a multiblock multigrid method. *Commun Appl Numer Meth* 1992;8:735-47.
- [34] Kuerten H, Geurts B. Compressible turbulent flow simulation with a multigrid multiblock method. In: Proceedings of the 6th Copper Mountain conference on multigrid methods; 1993.
- [35] Rizzi A, Eliasson P, Lindblad I, Hirsch C, Lacor C, Haeuser J. The engineering of multiblock/multigrid software for navier stokes flows on structured meshes. *Comput Fluids* 1993;22:341-67.
- [36] Enander R, Sternér E. Analysis of internal boundary conditions and communication strategies for multigrid multiblock methods. Dept. of Scientific Computing, Uppsala University, Sweden, Report No. 191; 1997.
- [37] Rai MM. A conservative treatment of zonal boundaries for Euler equations calculations. *J Comput Phys* 1986;62:472-503.
- [38] Kassies A, Tognaccini R. Boundary conditions for euler equations at internal block faces of multi-block domains using local grid refinement. AIAA Paper 90-1590; 1990.
- [39] Benek JA, Buning PG, Steger JL. A 3-D chimera grid embedding technique. AIAA Paper 85-1523; 1985.
- [40] Buning PG, Chu IT, Obayashi S, Rizk YM, Steger JL. Numerical simulation of the integrated space shuttle vehicle in ascent. AIAA Paper 88-4359; 1988.
- [41] Chesshire G, Henshaw WD. Composite overlapping meshes for the solution of partial differential equations. *J Comput Phys* 1990;90:1-64.
- [42] Pearce DG, Stanley SA, Martin Jr FW, Gomez RJ, Le Beau GJ, Buning PG, et al. Development of a large scale chimera grid system for the space shuttle launch vehicle. AIAA Paper 93-0533; 1993.
- [43] Kao H-J, Liou M-S, Chow C-Y. Grid adaptation using chimera composite overlapping meshes. AIAA J 1994;32:942-9.
- [44] Nakahashi K, Togashi F, Sharov D. Intergrid-boundary definition method for overset unstructured grid approach. AIAA J 2000;38:2077-84.
- [45] Henshaw WD, Schwendeman DW. An adaptive numerical scheme for high-speed reactive flow on overlapping grids. *J Comput Phys* 2003;191:420-47.
- [46] Xu L, Weng P. High order accurate and low dissipation method for unsteady compressible viscous flow computation on helicopter rotor in forward flight. *J Comput Phys* 2014;258:470-88.
- [47] Roget B, Sitaraman J. Robust and efficient overset grid assembly for partitioned unstructured meshes. *J Comput Phys* 2014;260:1-24.
- [48] Wang G, Duchaine F, Papadogiannis D, Duran I, Moreau S, Gicquel, LYM. An overset grid method for large eddy simulation of turbomachinery stages. *J Comput Phys* 2014;274:333-55.
- [49] Thompson JF, Weatherill NP. Aspects of numerical grid generation: current science and art. AIAA Paper 93-3539; 1993.
- [50] Nakahashi K. FDM-FEM Zonal approach for computations of compressible viscous flows. Lecture Notes in Physics, vol. 264. Springer Verlag; 1986. p. 494-8.
- [51] Nakahashi K. A finite-element method on prismatic elements for the three-dimensional Navier-Stokes equations. Lecture Notes in Physics, vol. 323, Springer Verlag; 1989. p. 434-8.
- [52] Holmes DG, Connell SD. Solution of the two-dimensional Navier-Stokes equations on unstructured adaptive grids. AIAA Paper 89-1932; 1989.
- [53] Peace AJ, Shaw J. The modeling of aerodynamic flows by solution of the euler equations on mixed polyhedral grids. *Int J Numer Meth Eng* 1992;35:2003-29.
- [54] Kallinderis Y, Khawaja A, McMorris H. Hybrid prismatic/tetrahedral grid generation for viscous flows around complex geometries. AIAA J 1996;34:291-8.
- [55] Sharov D, Nakahashi K. Hybrid prismatic/tetrahedral grid generation for viscous flow applications. AIAA Paper 96-2000; 1996.
- [56] McMorris H, Kallinderis Y. Octree-advancing front method for generation of unstructured surface and volume meshes. AIAA J 1997;35:976-84.
- [57] Peraire J, Morgan K. Unstructured mesh generation for 3-d viscous flow. AIAA Paper 98-3010; 1998.
- [58] Marcum DL, Gaither JA. Mixed element type unstructured grid generation for viscous flow applications. AIAA Paper 99-3252; 1999.

- [59] Ito Y, Nakashiki K. Unstructured hybrid grid generation based on isotropic tetrahedral grids. AIAA Paper 2002-0861; 2002.
- [60] Tsoutsanis P, Antoniadis AF, Drakakis D. WENO schemes on arbitrary unstructured meshes for laminar, transitional and turbulent flows. *J Comput Phys* 2014;256:254–76.
- [61] Mavriplis DJ. Unstructured grid techniques. *Annu Rev Fluid Mech* 1997;29:473–514.
- [62] Abe Y, Nonomura T, Iizuka N, Fujii K. Geometric interpretations and spatial symmetry property of metrics in the conservative form for high-order finite-difference schemes on moving and deforming grids. *J Comput Phys* 2014;260:163–203.
- [63] Hirsch C. Numerical computation of internal and external flows, vols. 1 and 2. Chichester (UK): John Wiley and Sons; 1988.
- [64] McDonald PW. The computation of transonic flow through two-dimensional gas turbine cascades. ASME Paper 71-GT-89; 1971.
- [65] Turner MJ, Clough RW, Martin HC, Topp LP. Stiffness and deflection analysis of complex structures. *J Aeronaut Soc* 1956;23:805.
- [66] Zienkiewicz OC, Taylor RL. The finite element method. 4th ed. Maidenhead: McGraw-Hill; 1991.
- [67] Pironneau O. Finite element methods for fluids. Chichester: John Wiley; 1989.
- [68] Hassan O, Probert EJ, Morgan K, Peraire J. Adaptive finite element methods for transient compressible flow problems. In: Brebbia CA, Aliabadi MH, editors. *Adaptive finite and boundary element methods*. London: Elsevier Applied Science; 1993. p. 119–60.
- [69] Reddy JN, Gartling DK. The finite element method in heat transfer and fluid dynamics. Boca Raton, FL: CRC Press; 1994.
- [70] Shadid JN, Moffat HK, Hutchinson SA, Hennigan GL, Devine KD, Salinger AG. MPSalsa: a finite element computer program for reacting flow problems. Part 1 – Theoretical development. SAND95-2752, May 1996.
- [71] Atkins HL, Shu C-W. Quadrature-free implementation of discontinuous galerkin method for hyperbolic equations. *AIAA J* 1998;36(5):775–82.
- [72] Van der Vegt JJW, van der Ven H. Discontinuous Galerkin finite element method with anisotropic local grid refinement for inviscid compressible flows. *J Comput Phys* 1998;141:46–77.
- [73] Lomtev I, Kirby RM, Karniadakis GE. A discontinuous Galerkin ALE method for compressible viscous flows in moving domains. *J Comput Phys* 1999;155:128–59.
- [74] Burbeau A, Sagaut P, Bruneau, Ch-H. A problem-independent limiter for high-order Runge-Kutta discontinuous Galerkin methods. *J Comput Phys* 2001;169:111–50.
- [75] Van der Vegt JJW, van der Ven H. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: I. General formulation. *J Comput Phys* 2002;182:546–85.
- [76] Hartmann R, Houston P. Adaptive discontinuous Galerkin finite element methods for the compressible euler equations. *J Comput Phys* 2002;183:508–32.
- [77] Fidkowski KJ, Darmofal DL. Development of a higher-order solver for aerodynamic applications. AIAA Paper 2004-0436; 2004.
- [78] Aliabadi S, Tu S-Z, Watts M. An alternative to limiter in discontinuous Galerkin finite element method for simulation of compressible flows. AIAA Paper 2004-0076; 2004.
- [79] Özdemir H, Blom CPA, Hagmeijer R, Hoeijmakers HWM. Development of higher-order discontinuous galerkin method on hexahedral elements. AIAA Paper 2004-2961; 2004.
- [80] Dolejší V, Feistauer M. A semi-implicit discontinuous Galerkin finite element method for the numerical solution of inviscid compressible flow. *J Comput Phys* 2004;198:727–46.
- [81] Atkins HL, Helenbrook BT. Numerical evaluation of P-multigrid method for the solution of discontinuous Galerkin discretizations of diffusive equations. AIAA Paper 2005-5110; 2005.
- [82] Toulopoulos I, Ekaterinaris JA. Discontinuous-Galerkin discretizations for viscous flow problems in complex domains. AIAA Paper 2005-1264; 2005.
- [83] Bustamante R, Gatica GN. A mixed local discontinuous Galerkin method for a class of nonlinear problems in fluid mechanics. *J Comput Phys* 2005;207:427–56.
- [84] Fidkowski KJ, Oliver TA, Lu J, Darmofal DL. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *J Comput Phys* 2005;207:92–113.

- [85] Chevaugeon N, Remacle J-F, Gallez X, Ploumhans P, Caro S. Efficient discontinuous galerkin methods for solving acoustic problems. AIAA Paper 2005-2823; 2005.
- [86] Luo H, Baum JD, Löhner R. A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids. *J Comput Phys* 2006;211:767-83.
- [87] Marchandise E, Remacle J-F, Chevaugeon N. A quadrature-free discontinuous Galerkin method for the level set equation. *J Comput Phys* 2006;212:338-57.
- [88] Qiu J, Khoo BC, Shu C-W. A numerical study for the performance of the Runge-Kutta discontinuous Galerkin method based on different numerical fluxes. *J Comput Phys* 2006;212:540-65.
- [89] Nastase CR, Mavriplis DJ. High-order discontinuous Galerkin methods using hp-multigrid approach. *J Comput Phys* 2006;213:330-57.
- [90] Jacobs GB, Hesthaven JS. High-order nodal discontinuous Galerkin particle-in-cell method on unstructured grids. *J Comput Phys* 2006;214:96-121.
- [91] Lörcher F, Gassner G, Munz C-D. A discontinuous Galerkin scheme based on a space-time expansion. I. Inviscid compressible flow in one space dimension. *J Sci Comput* 2007;32:175-99.
- [92] Gassner G, Lörcher F, Munz C-D. A discontinuous Galerkin scheme based on a space-time expansion. II. Viscous flow equations in multi dimensions. *J Sci Comput* 2007;34:260-86.
- [93] Gassner G, Lörcher F, Munz C-D. A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. *J Comput Phys* 2007;224:1049-63.
- [94] Wang L, Mavriplis DJ. Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations. *J Comput Phys* 2007;225:1994-2015.
- [95] Zhu J, Qiu J, Shu C-W, Dumbser M. Runge-Kutta discontinuous Galerkin method using WENO limiters. II: Unstructured meshes. *J Comput Phys* 2008;227:4330-53.
- [96] Shahbazi K, Mavriplis DJ, Burgess NK. Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *J Comput Phys* 2009;228:7917-40.
- [97] Mavriplis DJ, Nastase CR. On the geometric conservation law for high-order discontinuous Galerkin discretizations on dynamically deforming meshes. *J Comput Phys* 2011;230:4285-300.
- [98] Zhu J, Zhong X, Shu C-W, Qiu J. Runge-Kutta discontinuous Galerkin method using a new type of WENO limiters on unstructured meshes. *J Comput Phys* 2013;248:200-20.
- [99] Patera AT. A spectral element method for fluid dynamics; laminar flow in a channel expansion. *J Comput Phys* 1984;54:468-88.
- [100] Maday Y, Patera AT. Spectral element methods for the incompressible Navier-Stokes equations. ASME, State of the art surveys on computational mechanics; 1987. p. 71-143.
- [101] Canuto C, Hussaini MY, Quarteroni A, Zang TA. Spectral methods in fluid dynamics. Berlin: Springer Verlag; 1987.
- [102] Henderson RD, Karniadakis G. Unstructured spectral element methods for simulation of turbulent flows. *J Comput Phys* 1995;122:191-217.
- [103] Amon CH. Spectral element-fourier method for unsteady conjugate heat transfer in complex geometry flows. *J Thermophys Heat Transfer* 1995;9:247-53.
- [104] Henderson RD, Meiron DI. Dynamic refinement algorithms for spectral element methods. AIAA Paper 97-1855; 1997.
- [105] Lomtev I, Quillen CB, Karniadakis GE. Spectral/hp methods for viscous compressible flows on unstructured 2D meshes. *J Comput Phys* 1998;144:325-57.
- [106] Bouffanais R, Deville MO, Fischer PF, Leriche E, Weill D. Large-eddy simulation of the lid-driven cubic cavity flow by the spectral element method. *J Sci Comput* 2006;27:151-62.
- [107] Pasquetti R, Rapetti F. Spectral element methods on unstructured meshes: comparisons and recent advances. *J Sci Comput* 2006;27:377-87.
- [108] Mei R, Luo L-S, Shyy W. An accurate curved boundary treatment in the lattice Boltzmann method. AIAA Paper 99-3353; 1999.
- [109] Hsu A, Yang T, Sun C, Lopez I. A lattice Boltzmann method for turbomachinery simulations. AIAA Paper 2002-3537; 2002.
- [110] Yu D, Mei R, Shyy W. A unified boundary treatment in lattice Boltzmann method. AIAA Paper 2003-0953; 2003.
- [111] Mavriplis DJ. Multigrid solution of the Lattice-Boltzmann equation. AIAA Paper 2005-5104; 2005.

- [112] Imamura T, Suzuki K, Nakamura T, Yoshida M. Flow simulation around an airfoil by lattice Boltzmann method on generalized coordinates. *AIAA J* 2005;43:1968-73.
- [113] Crouse B, Freed D, Balasubramanian G, Senthooran S, Lew P-T, Mongeau L. Fundamental aeroacoustics capabilities of the lattice-Boltzmann method. *AIAA Paper* 2006-2571; 2006.
- [114] Marié S, Ricot D, Sagaut P. Accuracy of lattice Boltzmann method for aeroacoustic simulations. *AIAA Paper* 2007-3515; 2007.
- [115] Najafiyazdi A, Mongeau L. A perfectly matched layer formulation for lattice Boltzmann method. *AIAA Paper* 2009-3117; 2009.
- [116] Aono H, Gupta A, Qi D, Shyy W. The lattice Boltzmann method for flapping wing aerodynamics. *AIAA Paper* 2010-4867; 2010.
- [117] So RMC, Fu SC, Leung RCK. Finite difference lattice Boltzmann method for compressible thermal fluids. *AIAA J* 2010;48:1059-71.
- [118] Eitel-Amor G, Meinke M, Schröder W. Lattice Boltzmann simulations with locally refined meshes. *AIAA Paper* 2011-3398; 2011.
- [119] Satti R, Li Y, Shock R, Noelting S. Unsteady flow analysis of a multi-element airfoil using lattice Boltzmann method. *AIAA J* 2012;50:1805-16.
- [120] Perot F, Meskine M, LeGoff V, Vidal V, Gille F, Vergne S, et al. HVAC noise predictions using a lattice Boltzmann method. *AIAA Paper* 2013-2228; 2013.
- [121] Chen L, He Y-L, Kang Q, Tao W-Q. Coupled numerical approach combining finite volume and lattice Boltzmann methods for multi-scale multi-physicochemical processes. *J Comput Phys* 2013;255:83-105.
- [122] Li Z, Yang M, Zhang Y. Hybrid lattice Boltzmann and finite volume method for natural convection. *J Thermophys Heat Transfer* 2014;28:68-77.
- [123] Löhner R, Corrigan AT, Wichmann K-R, Wall W. Comparison of lattice-Boltzmann and finite difference solvers. *AIAA Paper* 2014-1439; 2014.
- [124] Touil H, Ricot D, Lévêque E. Direct and large-eddy simulation of turbulent flows on composite multi-resolution grids by the lattice Boltzmann method. *J Comput Phys* 2014;256:220-33.
- [125] Yoshida H, Nagaoka M. Lattice Boltzmann method for the convection-diffusion equation in curvilinear coordinate systems. *J Comput Phys* 2014;257:884-900.
- [126] Dellar PJ. Lattice Boltzmann algorithms without cubic defects in Galilean invariance on standard lattices. *J Comput Phys* 2014;259:270-83.
- [127] Guzik SM, Weisgraber TH, Colella P, Alder BJ. Interpolation methods and the accuracy of lattice-Boltzmann mesh refinement. *J Comput Phys* 2014;259:461-87.
- [128] Faviera J, Revell A, Pinelli A. A lattice Boltzmann-immersed boundary method to simulate the fluid interaction with moving and slender flexible objects. *J Comput Phys* 2014;261:145-61.
- [129] Patil DV, Premnath KN, Banerjee S. Multigrid lattice Boltzmann method for accelerated solution of elliptic equations. *J Comput Phys* 2014;265:172-94.
- [130] Batina JT. A gridless Euler/Navier-Stokes solution algorithm for complex-aircraft applications. *AIAA Paper* 93-0333; 1993.
- [131] Shih S-C, Lin S-Y. A weighting least squares method for Euler and Navier-Stokes equations. *AIAA Paper* 94-0522; 1994.
- [132] Series on gridless methods. *Comput Meth Appl Mech Eng* 1996;139:1-440 [also 2004;193:933-1321].
- [133] Luo H, Baum J, Löhner R. A hybrid building-block and gridless method for compressible flows. *AIAA Paper* 2006-3710; 2006.
- [134] Tota P, Wang Z. Meshfree Euler solver using local radial basis functions for inviscid compressible flows. *AIAA Paper* 2007-4581; 2007.
- [135] Katz A, Jameson A. A comparison of various meshless schemes within a unified algorithm. *AIAA Paper* 2009-0596; 2009.
- [136] Tang L, Yang J, Lee J. Hybrid Cartesian grid/gridless algorithm for store separation prediction. *AIAA Paper* 2010-0508; 2010.
- [137] Barth TJ. Numerical aspects of computing high-Reynolds number flows on unstructured meshes. *AIAA Paper* 91-0721; 1991.

- [138] Mavriplis DJ, Venkatakrishnan V. A unified multigrid solver for the Navier-Stokes equations on mixed element meshes. ICASE Report No. 95-53; 1995.
- [139] Braaten ME, Connell SD. Three dimensional unstructured adaptive multigrid scheme for the Navier-Stokes equations. AIAA J 1996;34:281-90.
- [140] Haselbacher AC, McGuirk JJ, Page GJ. Finite volume discretisation aspects for viscous flows on mixed unstructured grids. AIAA Paper 97-1946; 1997 [also AIAA J 1999;37:177-84].
- [141] Crumpton PI, Moiner P, Giles MB. An unstructured algorithm for high Reynolds number flows on highly-stretched grids. In: 10th Int. Conf. Num. Meth. for Laminar and Turbulent Flows, Swansea, England, July 21-25; 1997.
- [142] Weiss JM, Maruszewski JP, Smith WA. Implicit solution of preconditioned Navier-Stokes equations using algebraic multigrid. AIAA J 1999;37:29-36.
- [143] Jameson A, Schmidt W, Turkel E. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. AIAA Paper 81-1259; 1981.
- [144] Jameson A, Baker TJ, Weatherill NP. Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper 86-0103; 1986.
- [145] Swanson RC, Turkel E. On central difference and upwind schemes. J Comput Phys 1992;101:292-306.
- [146] Haselbacher A, Blazek J. On the accurate and efficient discretisation of the Navier-Stokes equations on mixed grids. AIAA Paper 99-3363; 1999 [also AIAA J 2000;38:2094-102].
- [147] Steger JL, Warming RF. Flux vector splitting of the inviscid gasdynamic equations with application to finite difference methods. J Comput Phys 1981;40:263-93.
- [148] Van Leer B. Flux vector splitting for the Euler equations. In: Proc. 8th int. conf. on numerical methods in fluid dynamics, Springer Verlag; 1982. p. 507-12 [also ICASE Report 82-30; 1982].
- [149] Liou M-S, Steffen Jr CJ. A new flux splitting scheme. NASA TM-104404; 1991 [also J Comput Phys 1993;107:23-39].
- [150] Liou M-S. A sequel to AUSM: AUSM+. J Comput Phys 1996;129:364-82.
- [151] Jameson A. Positive schemes and shock modelling for compressible flow. Int J Numer Meth Fluids 1995;20:743-76.
- [152] Tatsumi S, Martinelli L, Jameson A. A new high resolution scheme for compressible viscous flow with shocks. AIAA Paper 95-0466; 1995.
- [153] Edwards JR. A low-diffusion flux-splitting scheme for Navier-Stokes calculations. Comput Fluids 1997;26:653-9.
- [154] Rossow C-C. A simple flux splitting scheme for compressible flows. In: Proc. 11th DGLR-Fach-symposium, Berlin, Germany, November 10-12; 1998.
- [155] Rossow C-C. A flux splitting scheme for compressible and incompressible flows. AIAA Paper 99-3346; 1999.
- [156] Godunov SK. A difference scheme for numerical computation discontinuous solution of hydrodynamic equations. Math. Sbornik [in Russian] 1959;47:271-306 [translated US Joint Publ. Res. Service, JPRS 7226; 1969].
- [157] Osher S, Solomon F. Upwind difference schemes for hyperbolic systems of conservation laws. Math Comput 1982;38:339-74.
- [158] Roe PL. Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 1981;43:357-72.
- [159] Barth TJ, Jespersen DC. The design and application of upwind schemes on unstructured meshes. AIAA Paper 89-0366; 1989.
- [160] Einfeldt B. On Godunov-type methods for gas dynamics. SIAM J Numer Anal 1988;25:294-318.
- [161] Toro EF, Spruce M, Speares W. Restoration of the contact surface in the HLL-Riemann solver. Shock Waves 1994;4:25-34.
- [162] Quirk JJ. A contribution to the great Riemann solver debate. Int J Numer Meth Fluids 1994;18:555-74.
- [163] Kim SD, Lee BJ, Lee HJ, Jeung I-S. Robust HLLC Riemann solver with weighted average flux scheme for strong shock. J Comput Phys 2009;228:7634-42.

- [164] Balsara DS, Dumbser M, Abgrall R. Multidimensional HLLC Riemann solver for unstructured meshes—with application to Euler and MHD flows. *J Comput Phys* 2014;261:172–208.
- [165] Nishikawa H, Kitamura K. Very simple, carbuncle-free, boundary-layer-resolving, rotated-hybrid Riemann solvers. *J Comput Phys* 2008;227:2560–81.
- [166] Harten A. High resolution schemes for hyperbolic conservation laws. *J Comput Phys* 1983;49:357–93.
- [167] Yee HC, Harten A. Implicit TVD schemes for hyperbolic conservation laws in curvilinear coordinates. *AIAA J* 1987;25:266–74.
- [168] Yee HC. Construction of implicit and explicit symmetric TVD schemes and their applications. *J Comput Phys* 1987;68:151–79.
- [169] Roe PL. Discrete models for the numerical analysis of time-dependent multidimensional gas dynamics. *J Comput Phys* 1986;63:458–76.
- [170] Powell KG, van Leer B, Roe PL. Towards a genuinely multi-dimensional upwind scheme. Rhode-St-Genèse: VKI Lecture Series 1990-03; 1990.
- [171] Sidilkover D. A genuinely multidimensional upwind scheme and efficient multigrid solver for the compressible Euler equations. ICASE Report, No. 94-84; 1994.
- [172] Struijs R, Roe PL, Deconinck H. Fluctuation splitting schemes for the 2D Euler equations. Rhode-St-Genèse: VKI Lecture Series 1991-01; 1991.
- [173] Paillère H, Deconinck H, Roe PL. Conservative upwind residual-distribution schemes based on the steady characteristics of the Euler equations. AIAA Paper 95-1700; 1995.
- [174] Issman E, Degrez G, Deconinck H. Implicit upwind residual-distribution Euler and Navier-Stokes solver on unstructured meshes. *AIAA J* 1996;34:2021–28.
- [175] Van der Weide E, Deconinck H. Compact residual-distribution scheme applied to viscous flow simulations. Rhode-St-Genèse: VKI Lecture Series 1998-03; 1998.
- [176] Abgrall R, Larat A, Ricchiuto M. Construction of very high order residual distribution schemes for steady inviscid flow problems on hybrid unstructured meshes. *J Comput Phys* 2011;230:4103–36.
- [177] Harten A, Engquist B, Osher S, Chakravarthy S. Uniformly high order accurate essentially non-oscillatory schemes III. *J Comput Phys* 1987;71:231–303 [also ICASE Report No. 86-22; 1986].
- [178] Casper J, Atkins HL. A finite-volume high-order ENO scheme for two-dimensional hyperbolic systems. *J Comput Phys* 1993;106:62–76.
- [179] Godfrey AG, Mitchell CR, Walters RW. Practical aspects of spatially high-order accurate methods. *AIAA J* 1993;31:1634–42.
- [180] Abgrall R, Lafon FC. ENO schemes on unstructured meshes. Rhode-St-Genèse: VKI Lecture Series 1993-04; 1993.
- [181] Abgrall R. On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation. *J Comput Phys* 1994;114:45–58.
- [182] Jiang G-S, Shu C-W. Efficient implementation of weighted ENO schemes. *J Comput Phys* 1996;126:202–28.
- [183] Ollivier-Gooch CF. High-Order ENO schemes for unstructured meshes based on least-squares reconstruction. AIAA Paper 97-0540; 1997.
- [184] Stanescu D, Habashi WG. Essentially nonoscillatory euler solutions on unstructured meshes using extrapolation. *AIAA J* 1998;36:1413–16.
- [185] Friedrich O. Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids. *J Comput Phys* 1998;144:194–212.
- [186] Shi J, Hu C, Shu C-W. A technique of treating negative weights in WENO schemes. *J Comput Phys* 2002;175:108–27.
- [187] Martín MP, Taylor EM, Wu M, Weirs VG. A bandwidth-optimized WENO scheme for the effective direct numerical simulation of compressible turbulence. *J Comput Phys* 2006;220:270–89.
- [188] Dumbser M, Käser M, Titarev VA, Toro EF. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. *J Comput Phys* 2007;226:204–43.
- [189] Borges R, Carmona M, Costa B, Don WS. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *J Comput Phys* 2008;227:3191–211.

- [190] Dumbser M, Enaux C, Toro EF. Finite volume schemes of very high order of accuracy for stiff hyperbolic balance laws. *J Comput Phys* 2008;227:3971–4001.
- [191] Castro M, Costa B, Don WS. High order weighted essentially non-oscillatory WENO-Z schemes for hyperbolic conservation laws. *J Comput Phys* 2011;230:1766–92.
- [192] Tsoutsanis P, Antoniadis AF, Drikakis D. WENO schemes on arbitrary unstructured meshes for laminar, transitional and turbulent flows. *J Comput Phys* 2014;256:254–76.
- [193] Ivan L, Groth, CPT. High-order solution-adaptive central essentially non-oscillatory (CENO) method for viscous flows. *J Comput Phys* 2014;257:830–62.
- [194] Xu L, Weng P. High order accurate and low dissipation method for unsteady compressible viscous flow computation on helicopter rotor in forward flight. *J Comput Phys* 2014;258:470–88.
- [195] Huang C-S, Arbogast T, Hung C-H. A re-averaged WENO reconstruction and a third order CWENO scheme for hyperbolic conservation laws. *J Comput Phys* 2014;262:291–312.
- [196] Boscheri W, Balsara DS, Dumbser M. Lagrangian ADER-WENO finite volume schemes on unstructured triangular meshes based on genuinely multidimensional HLL Riemann solvers. *J Comput Phys* 2014;267:112–38.
- [197] Fan P, Shen Y, Tian B, Yang C. A new smoothness indicator for improving the weighted essentially non-oscillatory scheme. *J Comput Phys* 2014;269:329–54.
- [198] Venkatakrishnan V. Preconditioned conjugate gradient methods for the compressible Navier-Stokes equations. *AIAA J* 1991;29:1092–110.
- [199] Venkatakrishnan V. On the accuracy of limiters and convergence to steady state solutions. *AIAA Paper* 93-0880; 1993.
- [200] Venkatakrishnan V. Convergence to steady-state solutions of the Euler equations on unstructured grids with limiters. *J Comput Phys* 1995;118:120–30.
- [201] Vinokur M. Flux Jacobian matrices and generalized roe average for an equilibrium real gas. *NASA CR-177512*; 1988.
- [202] Vinokur M, Liu Y. Equilibrium gas flow computations: II. An analysis of numerical formulations of conservation laws. *AIAA Paper* 88-0127; 1988.
- [203] Vinokur M, Montagné J-L. Generalized flux-vector splitting and Roe average for an equilibrium real gas. *J Comput Phys* 1990;89:276–300.
- [204] Vinokur M, Liu Y. Nonequilibrium flow computations: I. An analysis of numerical formulations of conservation laws. *NASA CR-177489*; 1988.
- [205] Molvik GA, Merkle CL. A set of strongly coupled, upwind algorithms for computing flows in chemical nonequilibrium. *AIAA Paper* 89-0199; 1989.
- [206] Liu Y, Vinokur M. Upwind algorithms for general thermo-chemical nonequilibrium flows. *AIAA Paper* 89-0201; 1989.
- [207] Grossman B, Cinnella P. Flux-split algorithms for flows with non-equilibrium chemistry and vibrational relaxation. *J Comput Phys* 1990;88:131–68.
- [208] Shuen J-S, Liou M-S, Van Leer B. Inviscid flux-splitting algorithms for real gases with non-equilibrium chemistry. *J Comput Phys* 1990;90:371–95.
- [209] Slomski JF, Anderson JD, Gorski JJ. Effectiveness of multigrid in accelerating convergence of multidimensional flows in chemical nonequilibrium. *AIAA Paper* 90-1575; 1990.
- [210] Yu S-T, Chang S-C, Jorgenson PCE, Park S-J, Lai M-C. Basic equations of chemically reactive flow for computational fluid dynamics. *AIAA Paper* 98-1051; 1998.
- [211] Rinaldi E, Pecnik R, Colonna P. Exact Jacobians for implicit Navier-Stokes simulations of equilibrium real gas flows. *J Comput Phys* 2014;270:459–77.
- [212] Venkatakrishnan V. Implicit schemes and parallel computing in unstructured grid CFD. ICASE Report No. 95-28; 1995.
- [213] Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. *J Comput Phys* 1996;127:380–97.
- [214] Van Leer B, Tai CH, Powell KG. Design of optimally smoothing multi-stage schemes for the Euler equations. *AIAA Paper* 89-1933; 1989.
- [215] Chang HT, Jann HS, van Leer B. Optimal multistage schemes for Euler equations with residual smoothing. *AIAA J* 1995;33:1008–16.

- [216] Shu CW, Osher S. Efficient implementation of essentially non-oscillatory shock capturing schemes. *J Comput Phys* 1988;77:439-71.
- [217] Lafon A, Yee HC. On the numerical treatment of nonlinear source terms in reaction-convection equations. AIAA Paper 92-0419; 1992.
- [218] Van Leer B, Lee WT, Roe P. Characteristic time-stepping or local preconditioning of the Euler equations. AIAA Paper 91-1552; 1991.
- [219] Riemslagh K, Dick E. A multigrid method for steady Euler equations on unstructured adaptive grids. In: Proc. 6th copper mountain conf. on multigrid methods, NASA Conf. Publ. 1993;3224:527-42.
- [220] Morano E, Dervieux A. Looking for $O(N)$ Navier-Stokes solutions on non-structured meshes. In: Proc. 6th Copper Mountain conf. on multigrid methods, NASA Conf. Publ. 1993;3224:449-64.
- [221] Ollivier-Gooch CF. Towards problem-independent multigrid convergence rates for unstructured mesh methods. In: Proc. 6th Int. Symp. CFD, Lake Tahoe, NV; 1995.
- [222] Jameson A. Solution of the Euler equations by a multigrid method. *Appl Math Comput* 1983;13:327-56.
- [223] Jameson A. Computational transonic. *Commun Pure Appl Math* 1988;41:507-49.
- [224] Blazek J, Kroll N, Radespiel R, Rossow C-C. Upwind implicit residual smoothing method for multi-stage schemes. AIAA Paper 91-1533; 1991.
- [225] Blazek J, Kroll N, Rossow C-C. A comparison of several implicit smoothing methods. In: Proc. ICFD conf. on numerical meth. for fluid dynamics, Reading; 1992. p. 451-60.
- [226] Enander E. Improved implicit residual smoothing for steady state computations of first-order hyperbolic systems. *J Comput Phys* 1993;107:291-6.
- [227] Enander E, Sjögren B. Implicit explicit residual smoothing for upwind schemes. Internal Report 96-179, Department of Scientific Computing, Uppsala University, Sweden; 1996.
- [228] Fedorenko RP. A relaxation method for solving elliptic difference equations. *USSR Comput Math Math Phys* 1962;1(5):1092-6.
- [229] Bakhvalov NS. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Comput Math Math Phys* 1966;6(5):101-35.
- [230] Brandt A. Multi-level adaptive solutions to boundary-value problems. *Math Comput* 1977;31:333-90.
- [231] Brandt A. Guide to multigrid development. *Multigrid Methods I*, Lecture Notes in Mathematics, vol. 960. Berlin: Springer Verlag; 1981.
- [232] Jameson A. Multigrid algorithms for compressible flow calculations. *Multigrid Methods II*, Lecture Notes in Mathematics, vol. 1228. Berlin: Springer Verlag; 1985. p. 166-201.
- [233] Martinelli L. Calculation of viscous flows with a multigrid method. Ph.D. Thesis, Dept. of Mech. and Aerospace Eng., Princeton University; 1987.
- [234] Mulder WA. A new multigrid approach to convection problems. *J Comput Phys* 1989;83: 303-23.
- [235] Koren B. Multigrid and defect correction for the steady Navier-Stokes equations. *J Comput Phys* 1990;87:25-46.
- [236] Turkel E, Swanson RC, Vatsa VN, White JA. Multigrid for hypersonic viscous two- and three-dimensional flows. AIAA Paper 91-1572; 1991.
- [237] Radespiel R, Swanson R.C. Progress with multigrid schemes for hypersonic flow problems. ICASE Report, No. 91-89; 1991 [also *J Comput Phys* 1995;116:103-22].
- [238] Arnone A, Pacciani R. Rotor-stator interaction analysis using the Navier-Stokes equations and a multigrid method. *Trans ASME J Turbomach* 1996;118:679-89.
- [239] Sockol PM. Multigrid solution of the Navier-Stokes equations at low speeds with large temperature variations. *J Comput Phys* 2003;192:570-92.
- [240] Mavriplis DJ. Three-dimensional multigrid for the Euler equations. *AIAA J* 1992;30:1753-61.
- [241] Peraire J, Peiró J, Morgan K. A 3D finite-element multigrid solver for the Euler equations. AIAA Paper 92-0449; 1992.
- [242] Lallemand M, Steve H, Dervieux A. Unstructured multigridding by volume agglomeration: current status. *Comput Fluids* 1992;21:397-433.

- [243] Crumpton PI, Giles MB. Aircraft computations using multigrid and an unstructured parallel library. AIAA Paper 95-0210; 1995.
- [244] Olivier-Gooch CF. Multigrid acceleration of an upwind Euler solver on unstructured meshes. AIAA J 1995;33:1822-27.
- [245] Mavriplis DJ. Multigrid techniques for unstructured meshes. ICASE Report No. 95-27; 1995.
- [246] Mavriplis DJ, Venkatakrishnan V. A 3D agglomeration multigrid solver for the Reynolds-averaged Navier-Stokes equations on unstructured meshes. Int J Numer Meth Fluids 1996;23:527-44.
- [247] Mavriplis DJ. Directional agglomeration multigrid techniques for high Reynolds number viscous flow solvers. AIAA Paper 98-0612; 1998.
- [248] Mavriplis DJ. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. J Comput Phys 1998;145:141-65.
- [249] Okamoto N, Nakahashi K, Obayashi S. A coarse grid generation algorithm for agglomeration multigrid method on unstructured grids. AIAA Paper 98-0615; 1998.
- [250] Carré G, Lanteri S. Parallel linear multigrid by agglomeration for the acceleration of 3d compressible flow calculations on unstructured meshes. Numer Algorit 2000;24:309-32.
- [251] Roberts TW, Sidilkover D, Swanson RC. Textbook multigrid efficiency for the steady Euler equations. AIAA Paper 97-1949; 1997.
- [252] Pulliam TH. Time accuracy and the use of implicit methods. AIAA Paper 93-3360; 1993.
- [253] Arnone A, Liou M-S, Povinelli LA. Multigrid time-accurate integration of Navier-Stokes equations. AIAA Paper 93-3361; 1993.
- [254] Alonso J, Martinelli L, Jameson A. Multigrid unsteady Navier-Stokes calculations with aeroelastic applications. AIAA Paper 95-0048; 1995.
- [255] George A, Liu JW. Computer solution of large sparse positive definite systems. Prentice Hall Series in Comput. Math. Englewood Cliffs, NJ: Prentice Hall; 1981.
- [256] Pothen A, Simon HD, Liou KP. Partitioning sparse matrices with eigenvectors of graphs. SIAM J Matrix Anal Appl 1990;11:430-52.
- [257] Bender EE, Khosla PK. Solution of the two-dimensional Navier-Stokes equations using sparse matrix solvers. AIAA Paper 87-0603; 1987.
- [258] Venkatakrishnan V. Newton solution of inviscid and viscous problems. AIAA J 1989;27:885-91.
- [259] Beam RM, Bailey HS. Viscous computations using a direct solver. Comput Fluids 1990;18:191-204.
- [260] Vanden KJ, Whitfield DL. Direct and iterative algorithms for the three-dimensional Euler equations. AIAA J 1995;33:851-8.
- [261] Venkatakrishnan V, Barth TJ. Application of direct solvers to unstructured meshes for the Euler and Navier-Stokes equations using upwind schemes. AIAA Paper 89-0364; 1989.
- [262] Briley WR, McDonald H. Solution of the multi-dimensional compressible Navier-Stokes equations by a generalized implicit method. J Comput Phys 1977;24:372-97.
- [263] Beam R, Warming RF. An implicit factored scheme for the compressible Navier-Stokes equations. AIAA J 1978;16:393-402.
- [264] Pulliam TH, Steger JL. Recent improvements in efficiency, accuracy and convergence for implicit approximate factorization scheme. AIAA Paper 85-0360; 1985.
- [265] Rosenfeld M, Yassour Y. The alternating direction multi-zone implicit method. J Comput Phys 1994;110:212-20.
- [266] Golub GH, Van Loan CF. Matrix computations. The Johns Hopkins University Press, Baltimore, Maryland; 1983.
- [267] Chakravarthy SR. Relaxation methods for unfactored implicit schemes. AIAA Paper 84-0165; 1984.
- [268] Napolitano M, Walters RW. An incremental line Gauss-Seidel method for the incompressible and compressible Navier-Stokes equations. AIAA Paper 85-0033; 1985.
- [269] Thomas JL, Walters RW. Upwind relaxation algorithms for the Navier-Stokes equations. AIAA J 1987;25:527-34.
- [270] Jenssen CB. Implicit multiblock Euler and Navier-Stokes calculations. AIAA J 1994;32:1808-14.
- [271] Yoon S, Jameson A. Lower-upper implicit schemes with multiple grids for the Euler equations. AIAA Paper 86-0105 [also AIAA J 1987;7:929-35].
- [272] Yoon S, Jameson A. An LU-SSOR scheme for the Euler and Navier-Stokes equations. AIAA Paper 87-0600 [also AIAA J 1988;26:1025-26].

- [273] Rieger H, Jameson A. Solution of steady three-dimensional compressible Euler and Navier-Stokes equations by an implicit LU scheme. AIAA Paper 88-0619; 1988.
- [274] Shuen J-S. Upwind differencing and LU factorization for chemical non-equilibrium Navier-Stokes equations. *J Comput Phys* 1992;99:233–50.
- [275] Blazek J. Investigations of the implicit LU-SSOR scheme. DLR Research Report, No. 93-51; 1993.
- [276] Fezoui L, Stoufflet B. A class of implicit upwind schemes for Euler simulations with unstructured meshes. *J Comput Phys* 1989;84:174–206.
- [277] Karman SL. Development of a 3D unstructured CFD method. PhD thesis, The University of Texas at Arlington; 1991.
- [278] Batina JT. Implicit upwind solution algorithms for three-dimensional unstructured meshes. *AIAA J* 1993;31:801–5.
- [279] Slack DC, Whitaker DL, Walters RW. Time integration algorithms for the two-dimensional Euler equations on unstructured meshes. *AIAA J* 1994;32:1158–66.
- [280] Anderson WK. A grid generation and flow solution method for the Euler equations on unstructured grids. *J Comput Phys* 1994;110:23–38.
- [281] Anderson WK, Bonhaus DL. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Comput Fluids* 1994;23:1–21.
- [282] Anderson WK, Rausch RD, Bonhaus DL. Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids. *J Comput Phys* 1996;128:391–408 [also AIAA Paper 95-1740; 1995].
- [283] Tomaro RF, Strang WZ, Sankar LN. An implicit algorithm for solving time dependent flows on unstructured grids. *AIAA Paper* 97-0333; 1997.
- [284] Kano S, Nakahashi K. Navier-Stokes computations of HSCT off-design aerodynamics using unstructured hybrid grids. *AIAA Paper* 98-0232; 1998.
- [285] Hassan O, Morgan K, Peraire J. An adaptive implicit/explicit finite element scheme for compressible high speed flows. *AIAA Paper* 89-0363; 1989.
- [286] Hassan O, Morgan K, Peraire J. An implicit finite element method for high speed flows. *AIAA Paper* 90-0402; 1990.
- [287] Gibbons A. Algorithmic graph theory. New York: Cambridge University Press; 1985.
- [288] Löhner R, Martin D. An implicit linelet-based solver for incompressible flows. *AIAA Paper* 92-0668; 1992.
- [289] Hestenes MR, Stiefel EL. Methods of conjugate gradients for solving linear systems. *J Res Nat Bur Stand* 1952;49:409.
- [290] Sonneveld P. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J Sci Stat Comput* 1989;10:36–52.
- [291] Van der Vorst HA. BiCGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J Sci Stat Comput* 1992;13:631–44.
- [292] Freund RW. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J Sci Comput* 1993;14:470–82.
- [293] Saad Y, Schulz MH. GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J Sci Stat Comput* 1986;7:856–69.
- [294] Meister A. Comparison of different Krylov subspace methods embedded in an implicit finite volume scheme for the computation of viscous and inviscid flow fields on unstructured grids. *J Comput Phys* 1998;140:311–45.
- [295] Meijerink JA, Van der Vorst HA. Guidelines for usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. *J Comput Phys* 1981;44:134–55.
- [296] Wigton LB, Yu NJ, Young DP. GMRES acceleration of computational fluid dynamics codes. *AIAA Paper* 85-1494; 1985.
- [297] Kadioğlu M, Mudrick S. On the implementation of the GMRES(m) method to elliptic equations in meteorology. *J Comput Phys* 1992;102:348–59.
- [298] Venkatakrishnan V, Mavriplis DJ. Implicit solvers for unstructured meshes. *J Comput Phys* 1993;105:83–91.
- [299] Ajmani K, Ng W-F, Liou M-S. Preconditioned conjugate gradient methods for low speed flow calculations. *AIAA Paper*; 93-0881; 1993.

- [300] Dennis JE, Schnabel RB. Numerical methods for unconstrained optimization and nonlinear equations. Englewood Cliffs, NJ: Prentice-Hall; 1983.
- [301] Brown PN, Saad Y. Hybrid Krylov methods for nonlinear systems of equations. SIAM J Sci Stat Comput 1990;11:450-81.
- [302] Nielsen EJ, Anderson WK, Walters RW, Keyes DE. Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code. AIAA Paper 95-1733; 1995.
- [303] Cai X, Keyes DE, Venkatakrishnan V. Newton-Krylov-Schwarz: an implicit solver for CFD. ICASE Report 95-87; 1995.
- [304] Tidriri MD. Schwarz-based algorithms for compressible flows. ICASE Report No. 96-4; 1996.
- [305] Zingg D, Pueyo A. An efficient Newton-GMRES solver for aerodynamic computations. AIAA Paper 97-1955; 1997.
- [306] McHugh PR, Knoll DA. Inexact Newton's method solutions to the incompressible Navier-Stokes and energy equations using standard and matrix-free implementations. AIAA Paper 93-3332; 1993.
- [307] Luo H, Baum JD, Löhner R. A fast, matrix-free implicit method for compressible flows on unstructured grids. AIAA Paper 99-0936; 1999.
- [308] Raw M. Robustness of coupled algebraic multigrid for the Navier-Stokes equations. AIAA Paper 96-0297; 1996.
- [309] Brieger L, Lecca G. Parallel multigrid preconditioning of the conjugate gradient method for systems of subsurface hydrology. J Comput Phys 1998;142:148-62.
- [310] Yoon S, Kwak D. Multigrid convergence of an implicit symmetric relaxation scheme. AIAA Paper 93-3357; 1993.
- [311] Blazek J. A multigrid LU-SSOR scheme for the solution of hypersonic flow problems. AIAA Paper 94-0062; 1994.
- [312] Oosterlee CW. A GMRES-based plane smoother in multigrid to solve 3D anisotropic fluid flow problems. J Comput Phys 1997;130:41-53.
- [313] Gerlinger P, Stoll P, Brüggemann D. An implicit multigrid method for the simulation of chemically reacting flows. J Comput Phys 1998;146:322-45.
- [314] Piomelli U. Large-eddy simulation: present state and future perspectives. AIAA Paper 98-0534; 1998.
- [315] Reynolds O. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. Phil Trans R Soc Lond A 1895;186:123-64.
- [316] Schlichting H. Boundary layer theory. New York: McGraw Hill; 1968.
- [317] Young AD. Boundary layers. BSP Professional Books. Oxford: Blackwell Scientific Publication Ltd.; 1989.
- [318] Favre A. Equations des gaz turbulents compressibles, part 1: formes générales. J Mécan 1965;4: 361-90.
- [319] Favre A. Equations des gaz turbulents compressibles, part 2: méthode des vitesses moyennes; méthode des vitesses moyennes pondérées par la masse volumique. J Mécan 1965;4:391-421.
- [320] Morkovin MV. Effects of compressibility on turbulent flow. In: Favre A, editor. The mechanics of turbulence. New York: Gordon & Breach; 1964.
- [321] Rotta J. Statistische theorie nichthomogener turbulenz I. Z Phys 1951;129:547-72.
- [322] Rodi W. A new algebraic relation for calculating the Reynolds stresses. ZAMM 1976;56:219-21.
- [323] Speziale CG. A review of Reynolds stress models for turbulent shear flows. ICASE Report No. 95-15; 1995.
- [324] Hallbäck M, Henningson DS, Johansson AV, Alfredsson PH, editors. Turbulence and transition modelling. ERCOFTAC Series, vol. 2, Kluwer Academic Publishers, Dordrecht, The Netherlands; 1996.
- [325] Boussinesq J. Essai sur la théorie des eaux courantes. Mem Pres Acad Sci 1877;XXIII:46.
- [326] Boussinesq J. Theorie de l' écoulement tourbillonant et tumulteur des liquides dans les lits rectilignes. Comptes Rendus Acad Sci 1896;CXXII:1293.
- [327] Spalart P, Shur M. On the sensitization of turbulence models to rotation and curvature. Aerospace Sci Tech 1997;5:297-302.
- [328] Shur M, Strelets M, Travlin A, Spalart P. Turbulence modeling in rotating and curved channels: assessment of the Spalart-Shur correction term. AIAA Paper 98-0325; 1998.

- [329] Shih TH, Zhu J, Liou WW, Chen K-H, Liu N-S, Lumley J. Modeling of turbulent swirling flows. In: Proc. 11th symposium on turbulent shear flows, Grenoble, France; 1997 [also NASA TM-113112; 1997].
- [330] Chen K-H, Liu N-S. Evaluation of a non-linear turbulence model using mixed volume unstructured grids. AIAA Paper 98-0233; 1998.
- [331] Abdel Gawad AF, Abdel Latif OE, Ragab SA, Shabaka IM. Turbulent flow and heat transfer in rotating non-circular ducts with nonlinear $k - \varepsilon$ model. AIAA Paper 98-0326; 1998.
- [332] Baldwin BS, Lomax HJ. Thin Layer Approximation and Algebraic Model for Separated Turbulent Flow. AIAA Paper 78-257; 1978.
- [333] Spalart P, Allmaras S. A one-equation turbulence model for aerodynamic flows. AIAA Paper 92-0439; 1992.
- [334] Launder BE, Spalding B. The numerical computation of turbulent flows. Comput Meth Appl Mech Eng 1974;3:269-89.
- [335] Wilcox DC. Reassessment of the scale determining equation for advanced turbulence models. AIAA J 1988;26:1299-310.
- [336] Bardina JE, Huang PG, Coakley TJ. Turbulence modeling validation, testing, and development. NASA TM-110446; 1997.
- [337] McParlin S, Adamczak D. Comparison of turbulence models for the RAE Model 2155 test cases. AIAA Paper 2003-598; 2003.
- [338] Jameson A. A non-oscillatory shock capturing scheme using flux limited dissipation. MAE Report No. 1653, Dept. of Mechanical and Aerospace Engineering, Princeton University; 1984.
- [339] Whitfield DL, Janus JM. Three-dimensional unsteady Euler equations solution using flux vector splitting. AIAA Paper 84-1552; 1984.
- [340] Thomas JL, Salas MD. Far field boundary conditions for transonic lifting solutions to the Euler equations. AIAA J 1986;24:1074-80.
- [341] Usab WJ, Murman EM. Embedded mesh solution of the Euler equations using a multiple-grid method. AIAA Paper 83-1946; 1983.
- [342] Radespiel R. A cell-vertex multigrid method for the Navier-Stokes equations. NASA TM-101557; 1989.
- [343] Kroll N, Jain RK. Solution of two-dimensional Euler equations – experience with a finite volume code. DFVLR-FB 87-41; 1987.
- [344] Verhoff A. Modeling of computational and solid surface boundary conditions for fluid dynamics calculations. AIAA Paper 85-1496; 1985.
- [345] Hirsch C, Verhoff A. Far field numerical boundary conditions for internal and cascade flow computations. AIAA Paper 89-1943; 1989.
- [346] Giles MB. Non-reflecting boundary conditions for Euler equation calculations. AIAA Paper 89-1942; 1989.
- [347] Tan Z, Wang DM, Srinivasan K, Pzekwas A, Sun R. Numerical simulation of coupled radiation and convection from complex geometries. AIAA Paper 98-2677; 1998.
- [348] Agarwal RK, Schulte P. A new computational algorithm for the solution of radiation heat transfer problems. AIAA Paper 98-2836; 1998.
- [349] Hino T. An unstructured grid method for incompressible viscous flows with a free surface. AIAA Paper 97-0862; 1997.
- [350] Löhner R, Yang C, Onate E, Idelsson S. An unstructured grid-based, parallel free surface solver. AIAA Paper 97-1830; 1997.
- [351] Cowles G, Martinelli L. A cell-centered parallel multiblock method for viscous incompressible flows with a free surface. AIAA Paper 97-1865; 1997.
- [352] Wagner CA, Davis DW, Simon SA, Hollingsworth TA. Computation of free surface flows using a hybrid multiblock/chimera approach. AIAA Paper 98-0228; 1998.
- [353] Calderer A, Kang S, Sotiropoulos F. Level set immersed boundary method for coupled simulation of air/water interaction with complex floating structures. J Comput Phys 2014;277:201-27.

CHAPTER 4

Structured Finite-Volume Schemes

Contents

4.1 Geometrical Quantities of a Control Volume	77
4.1.1 Two-dimensional case	77
4.1.2 Three-dimensional case	78
4.2 General Discretization Methodologies	80
4.2.1 Cell-centered scheme	80
4.2.2 Cell-vertex scheme: overlapping control volumes	82
4.2.3 Cell-vertex scheme: dual control volumes	85
4.2.4 Cell-centered versus cell-vertex schemes	88
4.3 Discretization of the Convective Fluxes	89
4.3.1 Central scheme with artificial dissipation	92
<i>Scalar dissipation scheme</i>	93
<i>Matrix dissipation scheme</i>	94
4.3.2 Flux-vector splitting schemes	95
<i>Van Leer's scheme</i>	96
<i>AUSM</i>	97
<i>CUSP scheme</i>	100
4.3.3 Flux-difference splitting schemes	103
<i>Roe upwind scheme</i>	103
4.3.4 Total variation diminishing schemes	106
<i>Upwind TVD scheme</i>	106
4.3.5 Limiter functions	107
<i>Limiter functions for MUSCL interpolation</i>	109
<i>Limiter for CUSP scheme</i>	112
<i>Limiter for TVD scheme</i>	112
4.4 Discretization of the Viscous Fluxes	113
4.4.1 Cell-centered scheme	114
4.4.2 Cell-vertex scheme	116
References	116

As we already mentioned in the introduction to Chapter 3, the overwhelming number of numerical schemes for the solution of the Euler- and the Navier-Stokes equations employs the *method of lines*, that is, a separate discretization in space and in time. By consequence, it allows us to use numerical approximations of different accuracy for the spatial and temporal derivatives, as it may be required by the problem to be solved. Thus, we gain a lot of flexibility by this approach. For this reason, we shall follow the method of lines here. A detailed discussion of numerical methods based on *coupled* space and

time discretization, like the Lax-Wendroff family of schemes (e.g., explicit MacCormack predictor-corrector scheme, implicit Lerat's scheme, etc.), may be found, for example, in Ref. [1].

A general, structured, finite-volume scheme is naturally based on the conservation laws, which are expressed by the Navier-Stokes (2.19) or the Euler (2.44) equations. In a pre-processing step, the physical space is subdivided into a number of grid cells—quadrilaterals in 2D and hexahedra in 3D. The grid generation is done in such a way that:

- the domain is completely covered by the grid,
- there is no free space left between the grid cells,
- the grid cells do not overlap each other.

The resulting structured grid is uniquely described by the coordinates x, y, z of the grid points (corners of the grid cells) and indexes in the computational space (see Fig. 3.2), let us call them i, j, k . Based on the grid, control volumes are defined in order to evaluate the integrals of the convective and viscous fluxes as well as of the source term. For simplicity, let us suppose that a particular control volume does not change in time (otherwise please refer to Section A.5). Then, the time derivative of the conservative variables \vec{W} can be cast in the form

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega = \Omega \frac{\partial \vec{W}}{\partial t}.$$

Herewith, Eq. (2.19) becomes

$$\frac{\partial \vec{W}}{\partial t} = -\frac{1}{\Omega} \left[\oint_{\partial\Omega} (\vec{F}_c - \vec{F}_v) dS - \int_{\Omega} \vec{Q} d\Omega \right]. \quad (4.1)$$

The surface integral on the right-hand side of Eq. (4.1) is approximated by a sum of the fluxes crossing the faces of the control volume. This approximation is called *spatial discretization*. It is usually supposed that the flux is constant along the individual face and that it is evaluated at the midpoint of the face. The source term is generally assumed to be constant inside the control volume. However, in cases where the source term becomes dominant, it is advisable to evaluate \vec{Q} as the weighted sum of values from the neighboring control volumes (see [2] and the references cited therein). If we consider a particular volume $\Omega_{I,J,K}$, as displayed in Fig. 4.1b, we obtain from Eq. (4.1)

$$\frac{d \vec{W}_{I,J,K}}{dt} = -\frac{1}{\Omega_{I,J,K}} \left[\sum_{m=1}^{N_F} (\vec{F}_c - \vec{F}_v)_m \Delta S_m - (\vec{Q}\Omega)_{I,J,K} \right]. \quad (4.2)$$

In the above expression, the indexes in capital letters (I, J, K) reference the control volume in the computational space (cf. Section 3.1). As we shall see later, the control volume does not necessarily coincide with the grid. Furthermore, N_F denotes the

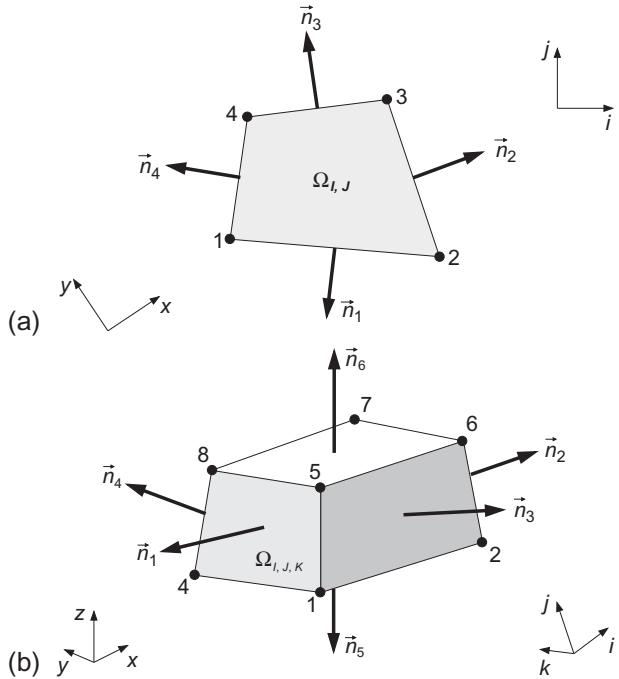


Figure 4.1 Control volume (Ω) and associated face unit normal vectors (\vec{n}_m) for a structured grid in: (a) 2D, (b) 3D. In (a), the unit normal vectors \vec{n}_2 and \vec{n}_4 are associated with the i -coordinate (direction) in computational space, \vec{n}_1 and \vec{n}_3 with the j -coordinate. In (b), the unit normal vectors \vec{n}_1 and \vec{n}_2 are associated with the i -coordinate, \vec{n}_5 and \vec{n}_6 with the j -coordinate, and \vec{n}_3 , \vec{n}_4 with the k -coordinate, respectively.

number of control volume faces (which is $N_F = 4$ in 2D and $N_F = 6$ in 3D). The variable ΔS_m stands for the area of the face m . The term in square brackets on the right-hand side of Eq. (4.2) is also generally termed the *residual*. It is denoted here by $\vec{R}_{I,J,K}$. Hence, we can abbreviate Eq. (4.2) as

$$\frac{d \vec{W}_{I,J,K}}{dt} = -\frac{1}{\Omega_{I,J,K}} \vec{R}_{I,J,K}. \quad (4.3)$$

When we write down the relationship in Eq. (4.3) for all control volumes $\Omega_{I,J,K}$, we obtain a system of ordinary differential equations of first order. The equations are hyperbolic in time. That means, we have to advance them in time by starting from a known initial solution. We have also to provide suitable boundary conditions for the viscous and the inviscid fluxes, as they are described in Chapter 8.

When solving the system of discretized governing equations (4.3) numerically, the first question is how do we define the control volumes and where do we locate the

flow variables with respect to the computational grid. In the framework of structured finite-volume schemes, three basic strategies are available:

- *Cell-centered* scheme—control volumes are identical with the grid cells and the flow variables are associated with their centroids (Fig. 4.3).
- *Cell-vertex* scheme with *overlapping* control volumes —flow quantities are assigned to the grid points (vertices, nodes) and the control volumes are defined as the union of all grid cells having the respective vertex in common (4 cells in 2D and 8 cells in 3D—see Fig. 4.4). This means that the control volumes associated with two neighboring grid points overlap each other.
- Cell-vertex scheme with *dual* control volumes: Here flow variables are again stored at the grid vertices, but the control volumes are now created by connecting the midpoints of the cells having the respective vertex in common (Fig. 4.5). In this way, the grid points are surrounded by their corresponding control volumes which do not overlap.

All three methodologies will be outlined in Section 4.2, which is devoted to discuss the general concepts of discretization schemes. It should be mentioned at this point that nowadays the cell-vertex scheme with overlapping control volumes is only seldom used. Nevertheless, it is included here for completeness.

It is important to notice that in our case **all** flow variables, that is, the conservative variables (ρ , ρu , ρv , ρw , and ρE) and the dependent variables (p , T , c , etc.), are associated with the **same** location—with the cell center or with the grid node. This approach is known as the *co-located* grid scheme. By contrast, many older pressure-based methods (cf. Section 3.1) use the *staggered* grid scheme, where the pressure and the velocity components are stored at different locations in order to suppress oscillations of the solution which arise from central differencing.

A wide range of choices exists with respect to the evaluation of the convective fluxes. The basic problem is that we have to know their values at all N_F faces of a control volume, but the flow variables are not directly available there. This means, we have to interpolate either the fluxes or the flow variables to the faces of the control volumes. In principle, this can be done in one of two ways:

- by arithmetic averaging like in *central* discretization schemes;
- by some biased interpolation like in *upwind* discretization schemes, which take care of the characteristics of the flow equations.

Besides the description, we shall treat aspects such as accuracy, range of applicability and numerical effort of the most widely used discretization schemes for the convective fluxes in Section 4.3.

A commonly applied methodology for the evaluation of the viscous fluxes at a face of the control volume is based on arithmetic averaging of the flow quantities. In contrast to the flow variables, the calculation of the velocity and the temperature gradients in Eqs. (2.15) and (2.24) is more involved. We shall present appropriate procedures in Section 4.4.

4.1 GEOMETRICAL QUANTITIES OF A CONTROL VOLUME

Before we turn our attention to the discretization methodologies, it is instructive to consider the calculation of geometrical quantities of the control volume $\Omega_{I,J,K}$ —its volume, the unit normal vector \vec{n}_m (defined as outward facing) and the area ΔS_m of a face m . The normal vector and the face area are also denoted as the *metrics* of the control volume. In the following sections, we shall consider the 2-D and the 3-D case separately for a general quadrilateral or hexahedral control volume, respectively.

4.1.1 Two-dimensional case

In general, we consider flow in a plane as a special case of a 3-D problem, where the solution is symmetric with respect to one coordinate direction (e.g., to the z -direction). Because of the symmetry and in order to obtain correct physical units for volume, pressure, etc., we set the depth of all grid cells and control volumes equal to a constant value b . The volume of a control volume results then in 2D from the product of its area with the depth b . The area of a quadrilateral can be exactly calculated by the formula of Gauss. Hence, for a control volume like that displayed in Fig. 4.1a, we get after some algebra

$$\Omega_{I,J} = \frac{b}{2} [(x_1 - x_3)(y_2 - y_4) + (x_4 - x_2)(y_1 - y_3)]. \quad (4.4)$$

where we have assumed that the control volume is located in the x - y -plane and that the z -coordinate is the symmetry axis. Since the depth b is arbitrary, we may set $b = 1$ for convenience. In 2D, the faces of a control volume are given by straight lines and therefore the unit normal vector is constant along them. When we integrate the fluxes according to the approximation of Eq. (4.2), we have to evaluate the product of the area of a face ΔS and the corresponding unit normal vector \vec{n} —the *face vector* \vec{S}

$$\vec{S}_m = \begin{bmatrix} S_{x,m} \\ S_{y,m} \end{bmatrix} = \vec{n}_m \Delta S_m. \quad (4.5)$$

Because of the symmetry, the z -component of the face vectors (and of the unit normal vector) is zero. It is therefore dropped from the expressions. The face vectors of the control volume from Fig. 4.1a are given by the relations

$$\begin{aligned} \vec{S}_1 &= b \begin{bmatrix} y_2 - y_1 \\ x_1 - x_2 \end{bmatrix}, \\ \vec{S}_2 &= b \begin{bmatrix} y_3 - y_2 \\ x_2 - x_3 \end{bmatrix}, \\ \vec{S}_3 &= b \begin{bmatrix} y_4 - y_3 \\ x_3 - x_4 \end{bmatrix}, \\ \vec{S}_4 &= b \begin{bmatrix} y_1 - y_4 \\ x_4 - x_1 \end{bmatrix}. \end{aligned} \quad (4.6)$$

The unit normal vector at face m is then obtained from Eq. (4.5) as

$$\vec{n}_m = \frac{\vec{S}_m}{\Delta S_m} \quad (4.7)$$

with

$$\Delta S_m = | \vec{S}_m | = \sqrt{S_{x,m}^2 + S_{y,m}^2}.$$

In practice, only the face vectors \vec{S}_1 and \vec{S}_4 are computed and stored for each control volume $\Omega_{I,J}$. The face vectors \vec{S}_2 as well as \vec{S}_3 are taken (with reversed signs to become outward facing) from the appropriate neighboring control volumes in order to save memory and to reduce the number of point operations.

4.1.2 Three-dimensional case

As opposed to the previous 2-D case, the calculation of face vectors and volumes poses some problems in the 3-D case. The main reason for this is that, in general, the four vertexes of the face of a control volume may not lie in a plane. Then, the normal vector is no longer constant on the face (Fig. 4.2). In order to overcome this difficulty, we could decompose all six faces of the control volume into two or more triangles each. The volume itself could then be built of tetrahedra. Performing this subdivision in an appropriate manner would lead to a discretization scheme which is at least first-order accurate on arbitrary grids [3]. Of course, the numerical effort would be increased substantially, because the fluxes would have to be integrated over each partial triangle separately. Hence, the number of point operations would be at least doubled. However, in Refs. [3, 4] it is shown that for reasonably smooth grids, where the control volume faces approach parallelograms, the decomposition into triangles does not improve the solution accuracy noticeably. Therefore, we shall employ a simplified treatment of the quadrilateral faces in the following considerations, which is based on an **averaged** normal vector.

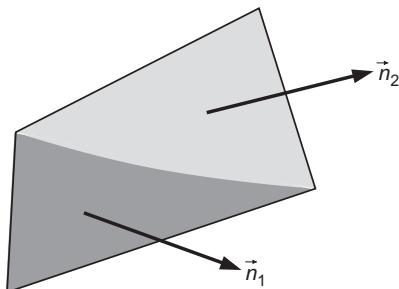


Figure 4.2 Face of a control volume with varying normal vector in 3D.

A face vector \vec{S} of an hexahedral control volume, like that rendered in Fig. 4.1b, is most conveniently computed using the same Gauss's formula as employed in 2D for the area of a quadrilateral. Thus, for example, for the face $m = 1$ (points 1, 5, 8, and 4 in Fig. 4.1b) we first define the differences

$$\begin{aligned}\Delta x_A &= x_8 - x_1, & \Delta x_B &= x_5 - x_4, \\ \Delta y_A &= y_8 - y_1, & \Delta y_B &= y_5 - y_4, \\ \Delta z_A &= z_8 - z_1, & \Delta z_B &= z_5 - z_4.\end{aligned}\quad (4.8)$$

The face vector $\vec{S}_1 = \vec{n}_1 \Delta S_1$ results then from

$$\vec{S}_1 = \frac{1}{2} \begin{bmatrix} \Delta z_A \Delta y_B - \Delta y_A \Delta z_B \\ \Delta x_A \Delta z_B - \Delta z_A \Delta x_B \\ \Delta y_A \Delta x_B - \Delta x_A \Delta y_B \end{bmatrix}. \quad (4.9)$$

The five remaining face vectors are calculated in a similar manner. It is again very convenient to store only three of the six the face vectors (e.g., \vec{S}_1 , \vec{S}_3 , and \vec{S}_5) for each control volume $\Omega_{I,J,K}$. The remaining face vectors \vec{S}_2 , \vec{S}_4 as well as \vec{S}_6 are obtained (with reversed signs to become outward facing) from the appropriate neighboring control volumes. The above expressions in Eqs. (4.8) and (4.9) deliver an average face vector. The approximation becomes exact when the face approaches a parallelogram, that is, when the vertexes of the face lie all in one plane. The unit normal vector is obtained from Eq. (4.7) with

$$\Delta S_m = \sqrt{S_{x,m}^2 + S_{y,m}^2 + S_{z,m}^2}. \quad (4.10)$$

Various, more or less accurate formulas are available for the calculation of the volume of a general hexahedron (see, e.g., [1]). One approach, which performed very well in various applications, is based on the divergence theorem [5]. This relates the volume integral of the divergence of some vector quantity to its surface integral. The key idea is to use the location in space of some point of the control volume Ω , let us call it $\vec{r} = [r_x, r_y, r_z]^T$, as the vector quantity. Herewith, the divergence theorem reads

$$\int_{\Omega} \operatorname{div}(\vec{r}) d\Omega = \oint_{\partial\Omega} (\vec{r} \cdot \vec{n}) dS. \quad (4.11)$$

We can easily evaluate the left-hand side of Eq. (4.11) which gives us the volume of Ω that we are looking for

$$\begin{aligned}\int_{\Omega} \operatorname{div}(\vec{r}) d\Omega &= \int_{\Omega} \left(\frac{\partial r_x}{\partial x} + \frac{\partial r_y}{\partial y} + \frac{\partial r_z}{\partial z} \right) d\Omega \\ &= 3 \Omega.\end{aligned}\quad (4.12)$$

If we assume now the unit normal vector is constant on all faces of the control volume, we can solve the surface integral on the right-hand side of Eq. (4.11) as follows

$$\oint_{\partial\Omega} (\vec{r} \cdot \vec{n}) dS \approx \sum_{m=1}^{m=6} (\vec{r}_{\text{mid}} \cdot \vec{n})_m \Delta S_m. \quad (4.13)$$

where $\vec{r}_{\text{mid},m}$ denotes the midpoint of the control volume face m . For example,

$$\vec{r}_{\text{mid},1} = \frac{1}{4}(\vec{r}_1 + \vec{r}_5 + \vec{r}_8 + \vec{r}_4),$$

where the vectors $\vec{r}_1, \vec{r}_5, \vec{r}_8$, and \vec{r}_4 correspond to the vertexes 1, 5, 8, and 4 of the face $m=1$ in Fig. 4.1b. Similar relations hold for the midpoints of the remaining faces. The area ΔS_m of the face m in Eq. (4.13) is obtained from Eq. (4.10). Combining Eqs. (4.12) and (4.13) together, and inserting the face vector \vec{S} for the product $\vec{n}_m \Delta S_m$, we have finally the relationship

$$\Omega_{I,J,K} = \frac{1}{3} \sum_{m=1}^{m=6} (\vec{r}_{\text{mid}} \cdot \vec{S})_m \quad (4.14)$$

for the volume of the control volume $\Omega_{I,J,K}$. Similar formula for the case with triangulated faces is provided in Ref. [6].

The origin of the coordinate system can in principle be moved to any place without affecting the volume calculation in Eq. (4.14). This leads us to the advice to locate the origin in one vertex of the control volume (e.g., point 1 in Fig. 4.1b), in order to achieve a better scaling of the numerical values. Thus, we may replace \vec{r} in the above expressions (4.11)-(4.14) by a transformed vector \vec{r}^* , which is defined as

$$\vec{r}^* = \vec{r} - \vec{r}_{\text{origin}}.$$

It is important to note that the volume computed with aid of Eq. (4.14) is exact for a control volume with planar faces.

4.2 GENERAL DISCRETIZATION METHODOLOGIES

In the introduction to this chapter, we already mentioned that the three approaches for the definition of the control volume and for the location of the flow variables. In this section, we shall present all three in more detail. We shall also discuss their advantages and shortcomings.

4.2.1 Cell-centered scheme

We speak of a cell-centered scheme if the control volumes are identical with the grid cells and if the flow variables are located at the centroids of the grid cells as indicated in Fig. 4.3. When we evaluate the discretized flow equations (4.2), we have to supply the

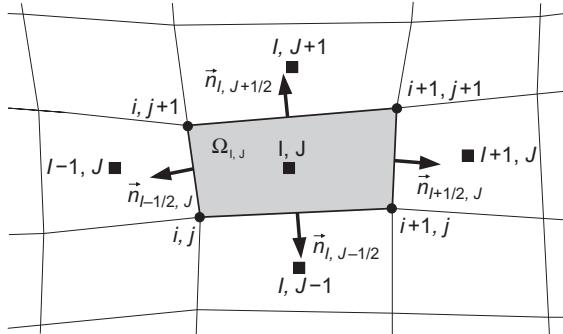


Figure 4.3 Control volume of a cell-centered scheme (in 2D).

convective and the viscous fluxes at the faces of a cell [7]. They can be approximated in one of the three following ways:

1. by the *average of fluxes* computed from values at the centroids of the grid cells to the left and to the right of the cell face, but using the same face vector (generally applied only to the convective fluxes);
2. by using an *average of variables* associated with the centroids of the grid cells to the left and to the right of the cell face;
3. by computing the fluxes from flow quantities interpolated separately to the left and to the right side of the cell face (employed only for the convective fluxes).

Thus, taking the cell face $\vec{n}_{I+1/2,J}$ in Fig. 4.3 as an example, the first approach—average of fluxes—reads in 2D

$$(\vec{F}_c \Delta S)_{I+1/2,J} \approx \frac{1}{2} [\vec{F}_c(\vec{W}_{I,J}) + \vec{F}_c(\vec{W}_{I+1,J})] \Delta S_{I+1/2,J} \quad (4.15)$$

with $\Delta S_{I+1/2,J}$ computed from Eqs. (4.6) and (4.7).

The second possible approach—average of variables—can be formulated as follows

$$(\vec{F} \Delta S)_{I+1/2,J} \approx \vec{F}(\vec{W}_{I+1/2,J}) \Delta S_{I+1/2,J}, \quad (4.16)$$

where the conservative/dependent variables at the face $\vec{n}_{I+1/2,J}$ of the control volume are defined as the arithmetic average of values at the two adjacent cells, that is,

$$\vec{W}_{I+1/2,J} = \frac{1}{2} (\vec{W}_{I,J} + \vec{W}_{I+1,J}). \quad (4.17)$$

The flux vector \vec{F} in Eq. (4.16) stands either for the convective or for the viscous fluxes.

The third methodology starts with an interpolation of flow quantities (being mostly velocity components, pressure, density and total enthalpy) separately to both sides of the cell face. The interpolated quantities—termed the *left* and the *right state* (see the beginning of Section 4.3)—differ in general between both sides. The fluxes through the

cell face are then evaluated from the difference of the left and right states using some non-linear function. Hence,

$$(\vec{F}_c \Delta S)_{I+1/2,J} \approx f_{\text{Flux}} (\vec{U}_L, \vec{U}_R, \Delta S_{I+1/2,J}), \quad (4.18)$$

where

$$\begin{aligned} \vec{U}_L &= f_{\text{Interp}} (\dots, \vec{U}_{I-1,J}, \vec{U}_{I,J}, \dots) \\ \vec{U}_R &= f_{\text{Interp}} (\dots, \vec{U}_{I,J}, \vec{U}_{I+1,J}, \dots) \end{aligned} \quad (4.19)$$

represent the interpolated states. Of course, similar relations like (4.15)–(4.19) hold also for the other cell faces.

The same approximations are employed in 3D. For example, at the cell face $\vec{n}_{I+1/2,J,K}$ (e.g., identical to \vec{n}_2 in Fig. 4.1b) the average of fluxes in Eq. (4.15) becomes

$$(\vec{F}_c \Delta S)_{I+1/2,J,K} \approx \frac{1}{2} [\vec{F}_c(\vec{W}_{I,J,K}) + \vec{F}_c(\vec{W}_{I+1,J,K})] \Delta S_{I+1/2,J,K} \quad (4.20)$$

with $\Delta S_{I+1/2,J,K}$ being defined correspondingly to Eqs. (4.8) and (4.9). The average of variables reads similarly to Eq. (4.16) as

$$(\vec{F} \Delta S)_{I+1/2,J,K} \approx \vec{F}(\vec{W}_{I+1/2,J,K}) \Delta S_{I+1/2,J,K} \quad (4.21)$$

with

$$\vec{W}_{I+1/2,J,K} = \frac{1}{2} (\vec{W}_{I,J,K} + \vec{W}_{I+1,J,K}). \quad (4.22)$$

The way over the interpolation of the flow variables results similarly to Eq. (4.18) in

$$(\vec{F}_c \Delta S)_{I+1/2,J,K} \approx f_{\text{Flux}} (\vec{U}_L, \vec{U}_R, \Delta S_{I+1/2,J,K}), \quad (4.23)$$

where \vec{U}_L and \vec{U}_R are the interpolated values at the cell face.

The last term in the discretized flow equations (4.2) which remains to be evaluated is the source term \vec{Q} . As we already stated in the introduction, the source term is usually supposed to be constant inside the control volume. For this reason, it is calculated using the flow variables from the corresponding cell center. Hence, we may define

$$(\vec{Q}\Omega)_{I,J,K} = \vec{Q}(\vec{W}_{I,J,K}) \Omega_{I,J,K}. \quad (4.24)$$

Using the above relations, the fluxes through the faces can be computed and the numerical integration over the boundary of $\Omega_{I,J,K}$ may be performed according to (4.2). In other words, the complete residual $\vec{R}_{I,J,K}$ is obtained. In Sections 4.3 and 4.4, we shall learn more in detail about the evaluation of the convective and viscous fluxes.

4.2.2 Cell-vertex scheme: overlapping control volumes

In a cell-vertex scheme, all flow variables are associated with the nodes of the computational grid. Within the approach based on overlapping control volumes, the

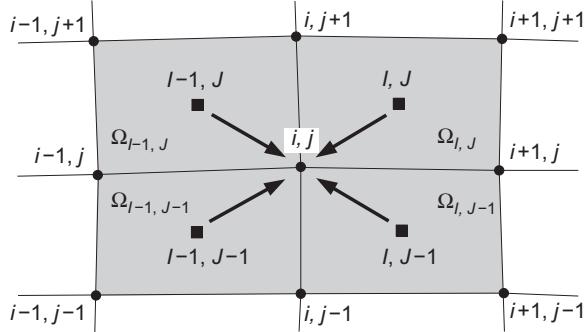


Figure 4.4 Overlapping control volumes of a cell-vertex scheme (2-D); arrows represent distribution of the residuals from cell centers to the common node (i,j) .

grid cells still represent the control volumes, just as in the case of the cell-centered scheme. The difference is that now the residuals computed for the control volumes have to be distributed to the grid points [4, 8, 9]. The situation is sketched in Fig. 4.4.

Let us consider the control volume $\Omega_{I,J}$ in Fig. 4.4, which is defined by the nodes

$$(i,j) \quad (i+1,j) \quad (i+1,j+1) \quad (i,j+1).$$

Note that the point (i,j) is located at the lower left corner of $\Omega_{I,J}$. The convective fluxes, for example, for the face $\Delta S_{I,J-1/2}$, which is given by the points (i,j) and $(i+1,j)$, are approximated as

$$(\vec{F}_c \Delta S)_{I,J-1/2} \approx \vec{F}_c(\vec{W}_{I,J-1/2}) \Delta S_{I,J-1/2}. \quad (4.25)$$

The variables at the midpoint of the face are evaluated using an arithmetic average of the variables at the nodes defining the face, that is,

$$\vec{W}_{I,J-1/2} = \frac{1}{2} [\vec{W}_{i,j} + \vec{W}_{i+1,j}]. \quad (4.26)$$

The face area $\Delta S_{I,J-1/2}$ is computed from the relations (4.6) and (4.7).

The approach remains the same in 3D. For example, for the face associated with the normal vector \vec{n}_3 in Fig. 4.1b, the averaged variables read

$$\vec{W}_{I,J,K-1/2} = \frac{1}{4} [\vec{W}_1 + \vec{W}_2 + \vec{W}_5 + \vec{W}_6]. \quad (4.27)$$

If we assume the edge 1-2 being oriented in the i -direction, edge 1-5 in the j -direction, edge 1-4 in the k -direction, and if we finally associate the point (i,j,k) with the corner 1 in Fig. 4.1b, the average in Eq. (4.27) can also be written as

$$\vec{W}_{I,J,K-1/2} = \frac{1}{4} [\vec{W}_{i,j,k} + \vec{W}_{i+1,j,k} + \vec{W}_{i,j+1,k} + \vec{W}_{i+1,j+1,k}]. \quad (4.28)$$

The convective flux is then again obtained from

$$(\vec{F}_c \Delta S)_{I,J,K-1/2} \approx \vec{F}_c (\vec{W}_{I,J,K-1/2}) \Delta S_{I,J,K-1/2}, \quad (4.29)$$

where the face area $\Delta S_{I,J,K-1/2}$ results from the formulas (4.8) and (4.9). The viscous fluxes are normally computed employing the same approach as for the dual control-volume scheme [10, 11], which results in a more compact scheme (i.e., one which involves fewer nodal values).

Summing up all face contributions given by relations (4.25), (4.26) and (4.28), (4.29), respectively, we obtain intermediate residuals $\vec{R}_{I,J,K}$ for all grid cells. In order to relate the cell-based to the node-based residuals, a further approximation is made using a *residual distribution formula*. It is basically a function, which evaluates the unknown node-based residual from a weighted sum of all cells having the particular grid node in common. The following distribution formulas were devised:

- volume weighted sum due to Ni [8];
- non-weighted sum due to Hall [9];
- characteristic (upwind) weighting procedure of Rossow [12, 13].

Theoretical investigations of the truncation error [4] suggest that Ni's scheme is more accurate than Hall's approach. However, in practice Ni's distribution formula leads to problems in places, where the grid is strongly distorted and stretched. For example, strong oscillations of the pressure field near the trailing edge of an airfoil were observed when using O-grids [4, 14]. Furthermore, convergence could only be achieved when the numerical viscosity was increased considerably. The underlying idea of the upwind weighting procedure [12, 13] is quite similar to that of the fluctuation-splitting schemes [15–18] (cf. Section 3.1.5), but the implementation is numerically much simpler. Basically, the residuals are sent only upstream in the characteristic direction.

Of the three approaches, Hall's distribution scheme proved to be the most robust. In Hall's scheme, the residual at a particular node results from a simple sum of all intermediate residuals $\vec{R}_{I,J,K}$, which cells share the node. Thus, in the 2-D case rendered in Fig. 4.4, we get

$$\vec{R}_{i,j} = \vec{R}_{I,J} + \vec{R}_{I-1,J} + \vec{R}_{I-1,J-1} + \vec{R}_{I,J-1}. \quad (4.30)$$

In 3D, in total eight cell-based residuals have to be summed up in the same way. A close inspection of (4.30) reveals that $\vec{R}_{i,j}$ is just the net flux through the boundary of the super-cell

$$\Omega_{i,j} = \Omega_{I,J} + \Omega_{I-1,J} + \Omega_{I-1,J-1} + \Omega_{I,J-1}, \quad (4.31)$$

due to the fact that the fluxes across the inner faces cancel each other. The super-cell also represents the “total” control volume, centered at the point (i,j) . In the 3-D case, the total volume consists of the cells

$$\Omega_{i,j,k} = \Omega_{I,J,K} + \Omega_{I-1,J,K} + \Omega_{I-1,J-1,K} + \Omega_{I,J-1,K} + \Omega_{I,J,K-1} + \Omega_{I-1,J,K-1} + \Omega_{I-1,J-1,K-1} + \Omega_{I,J-1,K-1} \quad (4.32)$$

As it can be seen from Fig. 4.4, the control volumes overlap by at least one cell, which gave the scheme its name.

The source term is calculated using the flow variables from the corresponding grid node, that is,

$$(\vec{Q}\Omega)_{i,j,k} = \vec{Q}(\vec{W}_{i,j,k}) \Omega_{i,j,k}. \quad (4.33)$$

With the above definitions, the time-stepping scheme in Eq. (4.2) becomes

$$\frac{d\vec{W}_{i,j,k}}{dt} = -\frac{1}{\Omega_{i,j,k}} \vec{R}_{i,j,k}. \quad (4.34)$$

This represents a system of ordinary differential equations, which has to be solved at each grid point (i, j, k) in the same way as for the cell-centered scheme. Note that the total volume (4.31) or (4.32), respectively, is utilized in Eq. (4.34).

4.2.3 Cell-vertex scheme: dual control volumes

In this scheme, the control volumes are centered around the particular grid node (vertex), where all flow variables are stored [19, 20]. As depicted in Fig. 4.5, in the 2-D case the dual control volumes are constructed by joining the midpoints of the four cells which share the node. In 3D, the centroids of eight cells have to be connected in order to form the faces of the control volume. Another possibility would be to join one cell centroid to the edge midpoint (in 2D) and then to the neighboring cell centroid again [21]. Thus, the face of the control volume would consist of two parts with different normal vectors, as it is common for unstructured grids (see Chapter 5). However, in the case of structured grids, such a definition of the control volume is justified only at boundaries, where the surface discretization would be otherwise changed. This is demonstrated in

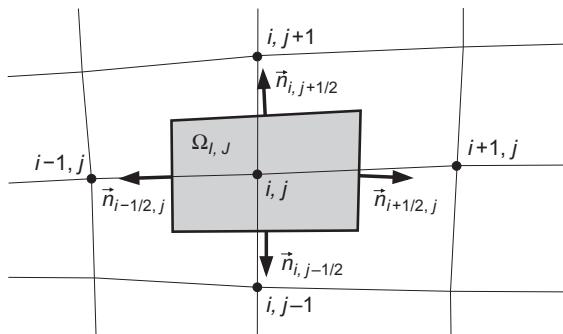


Figure 4.5 Dual control volume of a cell-vertex scheme in 2D.

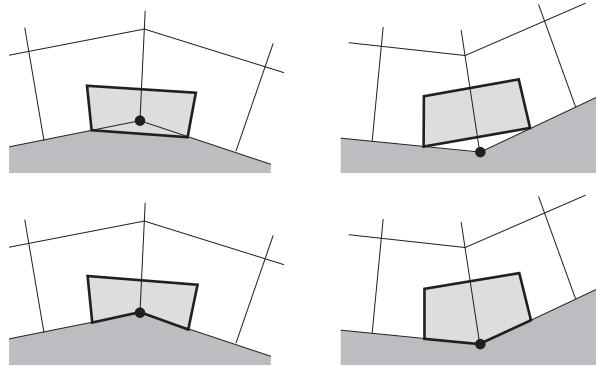


Figure 4.6 Definition of the dual control volume at a boundary (2D). Upper part: by connecting edge midpoints. Lower part: by connecting edge midpoints and the central node at the boundary.

Fig. 4.6. Significant advantages with respect to accuracy in the interior field cannot be expected from the second approach on reasonably smooth grids. Therefore, in what follows, the simpler definition of the dual control volume will be employed. The boundary treatment is discussed in Chapter 8.

When we evaluate the discretized flow equations (4.2), we have to compute the convective and the viscous fluxes at the faces of the control volume. This can be done according to one of the following three approaches:

1. by the *average of fluxes* computed from values at the nodes to the left and to the right of the face of the control volume, but using the same face vector (generally applied only to the convective fluxes);
2. by using an *average of variables* stored at the nodes to the left and to the right of the face;
3. by computing the fluxes from flow quantities interpolated separately to the left and to the right side of the face (employed only for the convective fluxes).

Thus, for example at the cell face $\vec{n}_{i+1/2,j}$ in Fig. 4.5 the first approach—average of fluxes—reads in 2D

$$(\vec{F}_c \Delta S)_{i+1/2,j} \approx \frac{1}{2} [\vec{F}_c(\vec{W}_{i,j}) + \vec{F}_c(\vec{W}_{i+1,j})] \Delta S_{i+1/2,j} \quad (4.35)$$

with $\Delta S_{i+1/2,j}$ being computed according to Eqs. (4.6) and (4.7), respectively, from the known coordinates of the cell centroids.

The second possible approach—average of variables—can be formulated as follows

$$(\vec{F} \Delta S)_{i+1/2,j} \approx \vec{F}(\vec{W}_{i+1/2,j}) \Delta S_{i+1/2,j}, \quad (4.36)$$

where the conservative (or the dependent) variables at the face $\vec{n}_{i+1/2,j}$ of the control volume result from arithmetic averaging of values at the two neighboring nodes. Hence,

$$\vec{W}_{i+1/2,j} = \frac{1}{2} (\vec{W}_{i,j} + \vec{W}_{i+1,j}). \quad (4.37)$$

The flux vector \vec{F} in Eq. (4.36) represents either the convective or the viscous fluxes.

The third methodology utilizes an interpolation of flow quantities (being mostly velocity components, pressure, density and total enthalpy) separately to both sides of the face. The interpolated quantities—termed the *left* and the *right state* (see the begin of Section 4.3)—differ in general between both sides. The fluxes through the face of the control volume are then evaluated from the difference of the left and right states using some non-linear function. Thus,

$$(\vec{F}_c \Delta S)_{i+1/2,j} \approx f_{\text{Flux}} (\vec{U}_L, \vec{U}_R, \Delta S_{i+1/2,j}), \quad (4.38)$$

where

$$\begin{aligned} \vec{U}_L &= f_{\text{Interp}} (\dots, \vec{U}_{i-1,j}, \vec{U}_{i,j}, \dots) \\ \vec{U}_R &= f_{\text{Interp}} (\dots, \vec{U}_{i,j}, \vec{U}_{i+1,j}, \dots) \end{aligned} \quad (4.39)$$

stand for the interpolated states. Of course, similar relations like (4.35)–(4.39) apply in the same way to other faces of the control volume.

The same approximations are employed in 3D. For example, at the cell face $\vec{n}_{i+1/2,j,k}$ (e.g., identical to \vec{n}_2 in Fig. 4.1b) the average of fluxes in Eq. (4.35) becomes

$$(\vec{F}_c \Delta S)_{i+1/2,j,k} \approx \frac{1}{2} [\vec{F}_c(\vec{W}_{i,j,k}) + \vec{F}_c(\vec{W}_{i+1,j,k})] \Delta S_{i+1/2,j,k}, \quad (4.40)$$

where $\Delta S_{i+1/2,j,k}$ is obtained from Eqs. (4.8) and (4.9). The average of the flow variables results similarly to Eq. (4.36) in

$$(\vec{F} \Delta S)_{i+1/2,j,k} \approx \vec{F}(\vec{W}_{i+1/2,j,k}) \Delta S_{i+1/2,j,k} \quad (4.41)$$

with

$$\vec{W}_{i+1/2,j,k} = \frac{1}{2} (\vec{W}_{i,j,k} + \vec{W}_{i+1,j,k}). \quad (4.42)$$

Finally, the interpolation of the flow variables leads correspondingly to Eq. (4.38) to

$$(\vec{F}_c \Delta S)_{i+1/2,j,k} \approx f_{\text{Flux}} (\vec{U}_L, \vec{U}_R, \Delta S_{i+1/2,j,k}), \quad (4.43)$$

where \vec{U}_L and \vec{U}_R denote the interpolated values at the face. A detailed description of several possible approaches will be presented in Section 4.3 for the convective and in Section 4.4 for the viscous fluxes.

The last term in the discretized flow equations (4.2) to be evaluated is the source term \vec{Q} . As already stated in the introduction, the source term is mostly supposed to be constant inside the control volume. For this reason, it is computed using the flow variables from the corresponding grid point. Hence, we may define

$$(\vec{Q}\Omega)_{i,j,k} = \vec{Q}(\vec{W}_{i,j,k}) \Omega_{i,j,k}. \quad (4.44)$$

Using the above relations, the fluxes through the faces can be computed and the numerical integration over the boundary of $\Omega_{i,j,k}$ can be carried out according to Eq. (4.2). In this way, we obtain the complete residual $\vec{R}_{i,j,k}$ including the source term. The change in time of the conservative variables follows then for each grid point from

$$\frac{d \vec{W}_{i,j,k}}{dt} = -\frac{1}{\Omega_{i,j,k}} \vec{R}_{i,j,k}. \quad (4.45)$$

In Chapter 6, we will present some suitable solution methods.

4.2.4 Cell-centered versus cell-vertex schemes

In the preceding three sections, cell-centered and cell-vertex discretization methodologies were outlined. Here, we compare the three schemes and give an overview of, at times controversial, the debate about their relative merits.

First, let us consider the accuracy of the discretizations. It follows from the discussion in Refs. [4, 22] that the cell-vertex scheme (either with overlapping or dual control volumes) can be made first-order accurate on distorted grids. On Cartesian or on smooth grids (i.e., where the volumes between adjacent cells vary only moderately and which are only slightly skewed), the cell-vertex scheme is second- or higher-order accurate [23], depending on the flux evaluation scheme. In the opposite, the discretization error of a cell-centered scheme depends strongly on the smoothness of the grid. For example, for an arrangement of the cells sketched in Fig. 4.7, an averaging does not provide the correct value at the midpoint of a face even for a linearly varying function. The consequence is that on a grid with slope discontinuity the discretization error will not be reduced even when the grid is infinitely refined. As demonstrated in Ref. [4], such zero-order errors manifest themselves as oscillations or kinks in isolines, whereas a cell-vertex scheme experiences no problems in the same situation. Nevertheless, on Cartesian

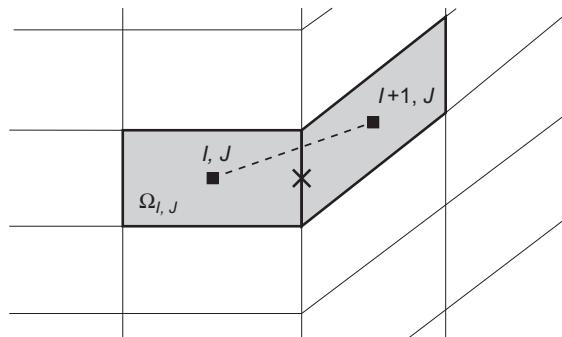


Figure 4.7 Cell-centered flux balance on a skewed grid; cross denotes the midpoint of the cell face.

or on sufficiently smooth grids, the cell-centered scheme can also reach second- or higher-order accuracy. A further analysis of the discretization errors were presented in Refs. [24–27].

Second, let us compare the three methods and their characteristics at boundaries. It is mainly at the solid wall boundary where the cell-vertex scheme with dual control volumes faces difficulties. Recalling Fig. 4.6 again, it is apparent that only about one half of the control volume is left at the boundary. The integration of fluxes around the faces results in a residual located **inside**—ideally in the **centroid**—of the control volume. But, the residual is associated with the **node** residing directly at the wall. This mismatch leads to increased discretization error in comparison to the cell-centered scheme. The definition of the dual control volume causes also problems at sharp corners (like trailing edges), which show up as unphysical peaks in pressure or density. Further complications arise, for example, at coordinate cuts or at periodic boundaries (see Chapter 8), where the fluxes from both parts of the control volume have to be summed up correctly. All cell-vertex schemes also require additional logic, in order to assure a consistent solution at boundary points shared by multiple grid blocks. No such problems appear for cell-centered schemes.

The cell-vertex scheme with overlapping control volumes has an advantage over the dual volume scheme in the treatment of wall boundaries, but it cannot be combined with the popular upwind discretization methods like TVD, AUSM, or CUSP. The discretization involves more points than those of the cell-centered and the dual control-volume schemes (27 instead of 7 in 3D), which leads to smearing of discontinuities and memory overhead in the case of an implicit time discretization.

The last main difference between the cell-centered and the cell-vertex schemes appears for unsteady flow problems. As mentioned earlier in Section 3.2, the cell-vertex schemes require at least an approximate treatment of the mass matrix [28, 29]. On the contrary, the mass matrix can be completely discarded in the case of a cell-centered scheme, because the residual is naturally associated with the centroid of the control volume.

In summary, the cell-vertex scheme with dual control volumes and the cell-centered scheme are numerically very similar in the interior of a stationary flow field. The main differences occur on distorted grids, in the boundary treatment and for unsteady flows. In the last two cases, the cell-centered approach shows advantages over the cell-vertex schemes, which result in a more straightforward implementation in a flow solver.

4.3 DISCRETIZATION OF THE CONVECTIVE FLUXES

In the previous sections, we discussed the spatial discretization methodologies in general. In this section, we shall learn more in detail about how the convective fluxes can be approximated.

As we could already see in Section 3.1.5, in the framework of the finite-volume approach, we have basically the choice between:

- central,
- flux-vector splitting,
- flux-difference splitting,
- total variation diminishing (TVD),
- fluctuation-splitting schemes.

In order to keep the amount of material bounded, we will restrict ourselves to the most important and popular methods. We will omit any detailed description of all possible modifications to the basic schemes, but instead reference the relevant literature.

Before we start to present the various discretization schemes in detail, we should explain what is meant by the designations *left* and *right state*, as well as by *stencil* or *computational molecule*, respectively.

Certain cell-centered and dual control-volume schemes require an interpolation of flow variables to the faces of the control volume. The situation is sketched in Fig. 4.8 for a grid in the i -direction. One possibility, which is employed by the central scheme (see the following section), consists of linear interpolation using the same number of values to the left and to the right of the face. In other words, the interpolation is centered at the face. Discretizations based on the characteristics of the Euler equations—upwind schemes—separately interpolate flow variables from the left and the right side of the face using non-symmetric formulas. The two values, named the **left** and the **right state**, are then utilized to compute the convective flux through the face (see Eqs. (4.18), (4.23), (4.38), or (4.43)). The interpolation formulas are almost exclusively (with the exception of TVD schemes) based on Van Leer's MUSCL (monotone upstream-centered schemes for conservation laws) approach [30]. They read for a general flow variable U

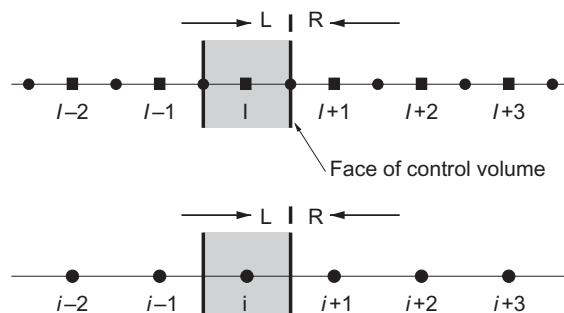


Figure 4.8 Left and right states at cell face $i+1/2$, resp. $i+1/2$. Upper part: cell-centered scheme; lower part: cell-vertex scheme with dual control volumes. Circles denote nodes, rectangles represent cell-centroids.

$$\begin{aligned} U_R &= U_{I+1} - \frac{\epsilon}{4} [(1 + \hat{\kappa})\Delta_- + (1 - \hat{\kappa})\Delta_+] U_{I+1}, \\ U_L &= U_I + \frac{\epsilon}{4} [(1 + \hat{\kappa})\Delta_+ + (1 - \hat{\kappa})\Delta_-] U_I. \end{aligned} \quad (4.46)$$

The forward (Δ_+) and the backward (Δ_-) difference operators are defined as

$$\begin{aligned} \Delta_+ U_I &= U_{I+1} - U_I, \\ \Delta_- U_I &= U_I - U_{I-1}. \end{aligned} \quad (4.47)$$

The indexes are shifted as appropriate. The above relationships remain valid for a cell-vertex scheme with dual control volumes, if the node index i is substituted for I . The parameter ϵ can be set equal to zero to obtain a first-order accurate upwind discretization. The parameter $\hat{\kappa}$ determines the spatial accuracy of the interpolation. For $\epsilon = 1$ and $\hat{\kappa} = -1$, the above interpolation formulas (4.46) give a fully one-sided interpolation of the flow variables, which results in a second-order accurate upwind approximation on uniformly spaced grid. The case $\hat{\kappa} = 0$ corresponds to a second-order accurate, upwind-biased linear interpolation. Furthermore, by setting $\hat{\kappa} = 1/3$, we obtain a three-point interpolation formula which constitutes (in a finite-volume framework—cf. Ref. [31, 49]) a second-order upwind-biased scheme with lower truncation error than the $\hat{\kappa} = -1$ and $\hat{\kappa} = 0$ schemes. Finally, if we specify $\hat{\kappa} = 1$, the MUSCL approach reduces to a purely central scheme—the average of variables. The schemes with $\hat{\kappa} = 0$ and $\hat{\kappa} = 1/3$ are the most often used ones in practice.

The MUSCL interpolation (4.46) has to be enhanced by the *limiter function* or *limiter*, if the flow region contains strong gradients. The purpose of the limiter is to suppress non-physical oscillation of the solution. Limiters will be discussed further in Section 4.3.5.

Stencil or **computational molecule** stands for the union of those cell-centroids or grid points, which are involved in the computation of the residual, the gradient, etc. For example, if we average the fluxes at the faces of the control volume according to Eqs. (4.15), (4.20), or (4.35), (4.40), respectively, we obtain in 2D a 5-point stencil, consisting of the cells/points

$$(I, J) \quad (I + 1, J) \quad (I, J + 1) \quad (I - 1, J) \quad (I, J - 1).$$

In 3D, a 7-point stencil results, which involves the cells/points

$$\begin{aligned} (I, J, K) &\quad (I + 1, J, K) \quad (I, J + 1, K) \quad (I - 1, J, K) \\ &\quad (I, J - 1, K) \quad (I, J, K - 1) \quad (I, J, K + 1). \end{aligned}$$

Both stencils are displayed in Fig. 4.9. Note that on a Cartesian grid this corresponds to the second-order accurate central-difference approximation of the first derivatives in

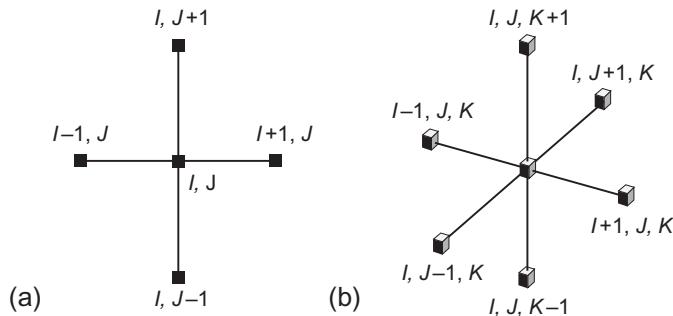


Figure 4.9 Stencil (computational molecule) of the central discretization scheme: (a) in 2-D, (b) in 3-D space.

i -, j -, and k -directions. Thus, a finite-difference scheme, applied to the differential form of the governing equations and using the central differences, would deliver the same result.

4.3.1 Central scheme with artificial dissipation

The central scheme with artificial dissipation is very simple compared to other discretization methods. It is easy to implement with either the cell-centered scheme or with both cell-vertex schemes. For these reasons the scheme became very wide-spread.

The basic idea of the central scheme is to compute the convective fluxes at a face of the control volume from the arithmetic average of the conservative variables on both sides of the face. Since this would allow for odd-even decoupling of the solution (generation of two independent solutions of the discretized equations) and for overshoots at shocks, artificial dissipation (which is similar to the viscous fluxes) has to be added for stability. The scheme was first implemented for the Euler equations by Jameson et al. [7]. Because of the names of the authors, it is also abbreviated as the JST scheme.

The central scheme is generally less accurate in the resolution of discontinuities and boundary layers than, let say, the upwind schemes. However, it is computationally considerably cheaper. Therefore, attempts were made to improve the accuracy of the scheme, while still keeping the numerical effort low. For example, improvements were devised to reduce the amount of the artificial dissipation in boundary layers [32, 33], or to enhance the shock resolution [34–36]. Another idea, followed in Ref. [36], is to utilize the Jacobian matrix of the convective fluxes, in order to scale the dissipation independently for each conservation equation. This successful approach is known as the matrix dissipation scheme. It can be viewed as a compromise between the original scalar scheme and the upwind schemes. The basic scheme also employs a single, pressure-based sensor to switch from second- to first-order accuracy at discontinuities to prevent non-physical oscillations of the flow variables. In Ref. [37], Jameson developed a concept called the SLIP (Symmetric Limited Positive) scheme, where a limiter is applied

separately for each conservation equation. A brief description of the previous approaches and comparisons for inviscid and viscous 2-D flows was presented in Ref. [38].

Scalar dissipation scheme

The convective fluxes (Eq. (2.21)) through a face of the control volume are approximated using the average of variables, according to Eqs. (4.16), (4.21), (4.36), or (4.41), respectively. Artificial dissipation is then added to the central fluxes for stability [7, 39]. Thus, the total convective flux at face $(I + 1/2, J, K)$ reads

$$(\vec{F}_c \Delta S)_{I+1/2,J,K} \approx \vec{F}_c(\vec{W}_{I+1/2,J,K}) \Delta S_{I+1/2,J,K} - \vec{D}_{I+1/2,J,K}, \quad (4.48)$$

where the flow variables are averaged as (see also Fig. 4.8)

$$\vec{W}_{I+1/2,J,K} = \frac{1}{2} (\vec{W}_{I,J,K} + \vec{W}_{I+1,J,K}). \quad (4.49)$$

In the case of the cell-vertex scheme with dual control volumes, node indexes (i, j, k) would be used instead. For simplicity, $(I + 1/2, J, K)$ will be abbreviated as $(I + 1/2)$ hereafter. The artificial dissipation flux consists of a blend of adaptive second- and fourth-order differences, which result from the sum of first- and third-order difference operators

$$\begin{aligned} \vec{D}_{I+1/2} = & \hat{\Lambda}_{I+1/2}^S \left[\epsilon_{I+1/2}^{(2)} (\vec{W}_{I+1} - \vec{W}_I) \right. \\ & \left. - \epsilon_{I+1/2}^{(4)} (\vec{W}_{I+2} - 3\vec{W}_{I+1} + 3\vec{W}_I - \vec{W}_{I-1}) \right]. \end{aligned} \quad (4.50)$$

From Eq. (4.50) we can see that the scheme possesses a compact 9-point stencil in 2D and a 13-point stencil in 3D.

The dissipation is scaled by the sum of the spectral radii of the convective flux Jacobians in all coordinate directions

$$\hat{\Lambda}_{I+1/2}^S = (\hat{\Lambda}_c^I)_{I+1/2} + (\hat{\Lambda}_c^J)_{I+1/2} + (\hat{\Lambda}_c^K)_{I+1/2}. \quad (4.51)$$

The spectral radius at the cell face $(I + 1/2)$, for example, in I -direction (represented by the superscript I), results from the average

$$(\hat{\Lambda}_c^I)_{I+1/2} = \frac{1}{2} \left[(\hat{\Lambda}_c^I)_I + (\hat{\Lambda}_c^I)_{I+1} \right]. \quad (4.52)$$

It is evaluated using the formula

$$\hat{\Lambda}_c = (|V| + c) \Delta S, \quad (4.53)$$

where V stands for the contravariant velocity (2.22) and c for the speed of sound, respectively.

A pressure-based sensor is used to switch off the fourth-order differences at shocks, where they would lead to strong oscillation of the solution. The sensor also switches

off the second-order differences in smooth parts of the flow field, in order to reduce the dissipation to the lowest possible level. Herewith, the coefficients $\epsilon^{(2)}$ and $\epsilon^{(4)}$ in Eq. (4.50) are defined as

$$\begin{aligned}\epsilon_{I+1/2}^{(2)} &= k^{(2)} \max(\Upsilon_I, \Upsilon_{I+1}), \\ \epsilon_{I+1/2}^{(4)} &= \max\left[0, (k^{(4)} - \epsilon_{I+1/2}^{(2)})\right]\end{aligned}\quad (4.54)$$

with the pressure sensor given by

$$\Upsilon_I = \frac{|p_{I+1} - 2p_I + p_{I-1}|}{p_{I+1} + 2p_I + p_{I-1}}. \quad (4.55)$$

Typical values of the parameters are $k^{(2)} = 1/2$ and $1/128 \leq k^{(4)} \leq 1/64$.

In order to reduce the amount of artificial dissipation across a viscous shear layer, we can re-define the scaling factors in Eq. (4.50) as follows [32, 33]

$$\begin{aligned}\hat{\Lambda}_{I+1/2}^S &= \frac{1}{2} \left[(\phi^I \hat{\Lambda}_c^I)_I + (\phi^I \hat{\Lambda}_c^I)_{I+1} \right], \\ \hat{\Lambda}_{J+1/2}^S &= \frac{1}{2} \left[(\phi^J \hat{\Lambda}_c^J)_J + (\phi^J \hat{\Lambda}_c^J)_{J+1} \right], \\ \hat{\Lambda}_{K+1/2}^S &= \frac{1}{2} \left[(\phi^K \hat{\Lambda}_c^K)_K + (\phi^K \hat{\Lambda}_c^K)_{K+1} \right].\end{aligned}\quad (4.56)$$

These are then employed instead of Eq. (4.51). The directionally dependent coefficients ϕ are given by the relations

$$\begin{aligned}\phi^I &= 1 + \max \left[\left(\frac{\hat{\Lambda}_c^J}{\hat{\Lambda}_c^I} \right)^\sigma, \left(\frac{\hat{\Lambda}_c^K}{\hat{\Lambda}_c^I} \right)^\sigma \right], \\ \phi^J &= 1 + \max \left[\left(\frac{\hat{\Lambda}_c^I}{\hat{\Lambda}_c^J} \right)^\sigma, \left(\frac{\hat{\Lambda}_c^K}{\hat{\Lambda}_c^J} \right)^\sigma \right], \\ \phi^K &= 1 + \max \left[\left(\frac{\hat{\Lambda}_c^I}{\hat{\Lambda}_c^K} \right)^\sigma, \left(\frac{\hat{\Lambda}_c^J}{\hat{\Lambda}_c^K} \right)^\sigma \right].\end{aligned}\quad (4.57)$$

The parameter σ is usually set equal to 1/2 or 2/3. This formulation decreases the scaling of the dissipation terms in the direction along the shorter side of a control volume, whose longer side is aligned with the flow.

Matrix dissipation scheme

In order to improve the accuracy by reducing the numerical dissipation, the preceding JST scheme can be modified to become more like an upwind scheme. The idea is to use a matrix—the convective flux Jacobian—instead of the scalar value $\hat{\Lambda}^S$ to scale the

dissipation terms [36]. In this way, each equation is scaled properly by the corresponding eigenvalue. Hence, Eq. (4.50) becomes

$$\vec{D}_{I+1/2} = |\bar{A}_c|_{I+1/2} \left[\epsilon_{I+1/2}^{(2)} (\vec{W}_{I+1} - \vec{W}_I) - \epsilon_{I+1/2}^{(4)} (\vec{W}_{I+2} - 3\vec{W}_{I+1} + 3\vec{W}_I - \vec{W}_{I-1}) \right]. \quad (4.58)$$

The scaling matrix corresponds to the convective flux Jacobian ($\bar{A}_c = \partial \vec{F}_c / \partial \vec{W}$) diagonalized with absolute values of the eigenvalues

$$|\bar{A}_c| = \bar{T} |\bar{\Lambda}_c \Delta S| \bar{T}^{-1}. \quad (4.59)$$

The matrices of right (\bar{T}) and left (\bar{T}^{-1}) eigenvectors as well as the diagonal matrix of the eigenvalues $\bar{\Lambda}_c$ can be found in Section A.11. The eigenvalues must be limited at stagnation points and sonic lines to prevent the dissipation from becoming zero. An efficient way of computing the product of $|\bar{A}_c|$ with \vec{W} is provided in Ref. [36]. It should be noted that by setting $\epsilon^{(2)} = 1/2$ and $\epsilon^{(4)} = 0$, we obtain a first-order accurate, fully upwind scheme.

As it was already stated before, the idea here was to develop a scheme which accuracy is close to that of upwind schemes, but which is still computationally only slightly more expensive (about 15–20%) than the scalar dissipation approach. Results of comparisons with flux-vector splitting schemes (CUSP and AUSM) were reported in Ref. [38].

4.3.2 Flux-vector splitting schemes

The flux-vector splitting methods can be viewed as the first level of upwind schemes, since they account only for the direction of wave propagation. The flux-vector splitting schemes decompose the vector of the convective fluxes into two parts—either according to the sign of certain characteristic variables, or into a convective and a pressure part. The well-known Van Leer's flux-vector splitting scheme [40] belongs to the first category based on characteristic decomposition. The second approach is followed by more recent methods like the advection upstream splitting method (AUSM) of Liou et al. [41, 42], or the convective upwind split pressure (CUSP) scheme of Jameson [43, 44], respectively. Further similar approaches are the low-diffusion flux-splitting scheme (LDFSS) introduced by Edwards [45], or the mach number-based advection pressure splitting (MAPS) scheme of Rossow [46, 47].

The flux-vector splitting schemes can be implemented only in the framework of the cell-centered scheme (Section 4.2.1), or the cell-vertex scheme with dual control volumes (Section 4.2.3). Their advantage can be seen in only a moderately increased numerical effort but a much better resolution of shocks and boundary layers, as compared to the central scheme with scalar artificial dissipation. However, the matrix dissipation scheme (Eq. (4.58)), can also produce results of comparable accuracy [38]. Because of certain numerical difficulties, many modifications to the basic schemes (particularly to

AUSM) were devised by various researchers and the development still continues. In the following, we shall present the basics of the Van Leer, AUSM and CUSP schemes, give some hints with respect to the most important modifications, and provide references to the corresponding literature.

Van Leer's scheme

Van Leer's flux-vector splitting scheme [40] is based on characteristic decomposition of the convective fluxes. An extension of the approach to body-fitted grids was presented in Refs. [48, 49].

The convective flux is split into a positive and a negative part, that is,

$$\vec{F}_c = \vec{F}_c^+ + \vec{F}_c^-, \quad (4.60)$$

according to the Mach number normal to the face of the control volume (e.g., at $(I + 1/2)$ —see Fig. 4.8)

$$(M_n)_{I+1/2} = \left(\frac{V}{c} \right)_{I+1/2}, \quad (4.61)$$

where V represents the contravariant velocity (2.22) and c the speed of sound, respectively. In the case of the cell-vertex scheme with dual control volumes, the cell indexes have to be replaced by node indexes.

The values of the flow variables ρ , u , v , w , and p , respectively, have to be interpolated first to the faces of the control volume correspondingly to Eq. (4.19) or Eq. (4.39). Then, the positive fluxes are computed with the left state and the negative fluxes with the right state. The advection Mach number $(M_n)_{I+1/2}$ is obtained from the relation [40]

$$(M_n)_{I+1/2} = M_L^+ + M_R^-, \quad (4.62)$$

where the split Mach numbers are defined as

$$M_L^+ = \begin{cases} M_L & \text{if } M_L \geq +1, \\ \frac{1}{4}(M_L + 1)^2 & \text{if } |M_L| < 1, \\ 0 & \text{if } M_L \leq -1, \end{cases} \quad (4.63)$$

and

$$M_R^- = \begin{cases} 0 & \text{if } M_R \geq +1, \\ \frac{1}{4}(M_R - 1)^2 & \text{if } |M_R| < 1, \\ M_R & \text{if } M_R \leq -1. \end{cases} \quad (4.64)$$

The Mach numbers M_L and M_R are evaluated using the left and right states, respectively, that is,

$$M_L = \frac{V_L}{c_L}, \quad M_R = \frac{V_R}{c_R}. \quad (4.65)$$

In the case of $|M_n| < 1$ (subsonic flow), the positive and the negative flux parts are given by

$$\vec{F}_c^\pm = \begin{bmatrix} f_{\text{mass}}^\pm \\ f_{\text{mass}}^\pm [n_x(-V \pm 2c)/\gamma + u] \\ f_{\text{mass}}^\pm [n_y(-V \pm 2c)/\gamma + v] \\ f_{\text{mass}}^\pm [n_z(-V \pm 2c)/\gamma + w] \\ f_{\text{energy}}^\pm \end{bmatrix}. \quad (4.66)$$

The mass and energy flux components are defined as

$$\begin{aligned} f_{\text{mass}}^+ &= +\rho_L c_L \frac{(M_L + 1)^2}{4}, \\ f_{\text{mass}}^- &= -\rho_R c_R \frac{(M_R - 1)^2}{4}, \\ f_{\text{energy}}^\pm &= f_{\text{mass}}^\pm \left\{ \frac{[(\gamma - 1)V \pm 2c]^2}{2(\gamma^2 - 1)} + \frac{u^2 + v^2 + w^2 - V^2}{2} \right\}_{L/R}. \end{aligned} \quad (4.67)$$

For supersonic flow, that is, for $|M_n| \geq 1$, the fluxes are evaluated from

$$\begin{aligned} \vec{F}_c^+ &= \vec{F}_c \quad \vec{F}_c^- = 0 \quad \text{if } M_n \geq +1 \\ \vec{F}_c^+ &= 0 \quad \vec{F}_c^- = \vec{F}_c \quad \text{if } M_n \leq -1. \end{aligned} \quad (4.68)$$

The evaluation of the left and right states follows generally the MUSCL approach [30], which is given by Eq. (4.46). The higher order schemes $\hat{\kappa} = -1$, $\hat{\kappa} = 0$ and $\hat{\kappa} = 1/3$, respectively, require a limiter if the flow field contains discontinuities like shocks. More details are provided in Section 4.3.5.

The flux-vector splitting scheme of Van Leer performs very well in the case of the Euler equations. But several investigations [50, 51], carried out with the Navier-Stokes equations revealed that splitting errors in the momentum and the energy equations smear the boundary layers and also lead to inaccurate stagnation and wall temperatures. A modification to the momentum flux in the direction normal to the boundary layer was therefore suggested in Ref. [52]. A similar remedy for the energy flux was proposed in Ref. [53]. Both modifications together remove the splitting errors, and hence they improve the solution accuracy considerably [54].

AUSM

The advection upstream splitting method (AUSM) was introduced by Liou and Steffen [41], and Liou [55]. It was subsequently modified by Wada and Liou [56] and renamed

as AUSMD/V. Finally, an improved version termed AUSM⁺ was presented by Liou [42, 57].

The underlying idea of the approach is based on the observation that the vector of convective fluxes (2.21) consists of two physically distinct parts, namely the convective and the pressure part

$$\vec{F}_c = V \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho H \end{bmatrix} + \begin{bmatrix} 0 \\ n_x p \\ n_y p \\ n_z p \\ 0 \end{bmatrix}. \quad (4.69)$$

The first term in Eq. (4.69) represents scalar quantities, which are convected by the contravariant velocity V . By contrast, the pressure term is governed by the acoustic wave speed. The idea now is to discretize the convective term in purely upwind manner by taking either the left or the right state, depending on the sign of V (even for subsonic flow). On the other hand, the pressure term includes both states in the subsonic case. It becomes fully upwind only for a supersonic flow.

Following the basic AUSM formulation from [41], we introduce an advection Mach number $(M_n)_{I+1/2}$ from Eq. (4.61). Herewith, we can recast the convective flux at the face $(I + 1/2)$, or $(i + 1/2)$ of the control volume, respectively, into

$$(\vec{F}_c)_{I+1/2} = (M_n)_{I+1/2} \begin{bmatrix} \rho c \\ \rho c u \\ \rho c v \\ \rho c w \\ \rho c H \end{bmatrix}_{L/R} + \begin{bmatrix} 0 \\ n_x p \\ n_y p \\ n_z p \\ 0 \end{bmatrix}_{I+1/2}, \quad (4.70)$$

where

$$(\bullet)_{L/R} = \begin{cases} (\bullet)_L & \text{if } M_{I+1/2} \geq 0, \\ (\bullet)_R & \text{otherwise.} \end{cases} \quad (4.71)$$

Similarly to Van Leer's flux-vector splitting scheme, the advection Mach number is evaluated as a sum of the left and right split Mach numbers according to the relations (4.62) and (4.63)–(4.65). The computation of the left and right states (flow quantities: ρ , u , v , w , p , H) is based again on a separate interpolation to the faces of the control volume according to Eq. (4.19) or Eq. (4.39). The interpolation follows the MUSCL methodology [30], as it is given in Eq. (4.46). All higher-order MUSCL schemes ($\hat{\kappa} = -1$, $\hat{\kappa} = 0$, and $\hat{\kappa} = 1/3$) require a limiter, if the flow field contains strong gradients like shocks. For more details readers are referred to Section 4.3.5.

The pressure at the face $(I + 1/2)$ of the control volume is obtained from the splitting [41]

$$p_{I+1/2} = p_L^+ + p_R^-, \quad (4.72)$$

with the split pressures given in Ref. [40]

$$p_L^+ = \begin{cases} p_L & \text{if } M_L \geq +1, \\ \frac{p_L}{4}(M_L + 1)^2(2 - M_L) & \text{if } |M_L| < 1, \\ 0 & \text{if } M_L \leq -1, \end{cases} \quad (4.73)$$

and

$$p_R^- = \begin{cases} 0 & \text{if } M_R \geq +1, \\ \frac{p_R}{4}(M_R - 1)^2(2 + M_R) & \text{if } |M_R| < 1, \\ p_R & \text{if } M_R \leq -1. \end{cases} \quad (4.74)$$

It is also possible to use the following lower-order expansion for $|M_{L/R}| < 1$

$$p_{L/R}^\pm = \frac{p_{L/R}}{2} (1 \pm M_{L/R}). \quad (4.75)$$

It should be noted that we can write AUSM also in the form

$$\begin{aligned} (\vec{F}_c)_{I+1/2} = & \frac{1}{2}(M_n)_{I+1/2} \left\{ \begin{bmatrix} \rho c \\ \rho cu \\ \rho cv \\ \rho cw \\ \rho cH \end{bmatrix}_L + \begin{bmatrix} \rho c \\ \rho cu \\ \rho cv \\ \rho cw \\ \rho cH \end{bmatrix}_R \right\} \\ & - \frac{1}{2} |(M_n)_{I+1/2}| \left\{ \begin{bmatrix} \rho c \\ \rho cu \\ \rho cv \\ \rho cw \\ \rho cH \end{bmatrix}_R - \begin{bmatrix} \rho c \\ \rho cu \\ \rho cv \\ \rho cw \\ \rho cH \end{bmatrix}_L \right\} \\ & + \begin{bmatrix} 0 \\ n_x(p_L^+ + p_R^-) \\ n_y(p_L^+ + p_R^-) \\ n_z(p_L^+ + p_R^-) \\ 0 \end{bmatrix}. \end{aligned} \quad (4.76)$$

The first term on the right-hand side of Eq. (4.76) represents a Mach number-weighted average of the left and right states—similar to the average of fluxes equation (4.15), (4.20) or (4.35), (4.40), respectively. The second term has a dissipative character. It is scaled by the scalar value $|(M_n)_{I+1/2}|$.

AUSM proved to deliver a crisp resolution of strong shocks and accurate results for boundary layers. However, the original AUSM [41, 55] was found to generate local pressure oscillations at shocks and in cases where the flow is aligned with the grid [58].

In Refs. [58, 59] it was therefore suggested to switch at shocks to Van Leer's scheme. When the advection Mach number $(M_n)_{I+1/2}$ tends to zero, the dissipation term in Eq. (4.76) will approach zero as well. Thus, disturbances cannot be damped by the scheme. In order to solve the flow alignment problem, it was proposed in Ref. [58] to modify the scaling of the dissipation term as follows:

$$|(M_n)_{I+1/2}| = \begin{cases} |(M_n)_{I+1/2}| & \text{if } |(M_n)_{I+1/2}| > \delta, \\ \frac{(M_n)_{I+1/2}^2 + \delta^2}{2\delta} & \text{if } |(M_n)_{I+1/2}| \leq \delta, \end{cases} \quad (4.77)$$

where δ is a small value ($0 < \delta \leq 0.5$). Hence, there will always be a sufficient amount of numerical dissipation. In order to retain the accuracy of AUSM for boundary layers, the parameter δ could be reduced in the wall normal direction using the same idea as given for the central scheme by Eq. (4.57).

Further improvements of the basic AUSM, with respect to better behavior in the vicinity of shocks, was presented in Refs. [42, 57] as AUSM⁺. The modifications consist of new Mach and pressure splittings, which replace the relations (4.63), (4.64) and (4.73), (4.74), respectively.

CUSP scheme

The concept of the convective upwind split pressure (CUSP) scheme is quite similar to that of AUSM. However, the CUSP approach has the advantage to be formulated as an average of fluxes (but without weighting like within AUSM) minus a dissipation term. This feature is crucial for the implementation in an explicit, hybrid multistage scheme. Furthermore, because of the different scaling factors as compared to AUSM, the CUSP scheme behaves more favorably in the case of flow alignment. The CUSP scheme was introduced by Jameson [43, 60, 61], and subsequently modified by Tatsumi et al. [44, 62]. It can be implemented either within the cell-centered or the (cell-vertex) dual control-volume type of spatial discretization.

The convective fluxes (Eq. (2.21)) through a face of the control volume are approximated using the arithmetic average of fluxes according to Eqs. (4.15), (4.20), (4.35), or (4.40), respectively. The dissipation term is then subtracted from the central fluxes for stabilization. Thus, the total convective fluxes at the face ($I + 1/2$) read

$$(\vec{F}_c)_{I+1/2} = \frac{1}{2} [\vec{F}_c(\vec{W}_R) + \vec{F}_c(\vec{W}_L)] - \vec{D}_{I+1/2}. \quad (4.78)$$

In the case of the dual control-volume discretization, ($i + 1/2$) would apply instead. The dissipation term, which is composed of a linear combination of the differences of the state and the flux vector, can be expressed as

$$\vec{D}_{I+1/2} = \frac{1}{2}(\alpha^* c)_{I+1/2} \left\{ \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho \phi \end{bmatrix}_R - \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho \phi \end{bmatrix}_L \right\} + \frac{1}{2} \beta_{I+1/2} \left\{ \begin{bmatrix} \rho V \\ \rho uV + n_x p \\ \rho vV + n_y p \\ \rho wV + n_z p \\ \rho HV \end{bmatrix}_R - \begin{bmatrix} \rho V \\ \rho uV + n_x p \\ \rho vV + n_y p \\ \rho wV + n_z p \\ \rho HV \end{bmatrix}_L \right\}. \quad (4.79)$$

There are two choices for the term ϕ in Eq. (4.79). Either ϕ is set equal to the total energy E , or it is set to be the total enthalpy H . In the first case we speak of the E-CUSP scheme [63], the second choice is consequently called the H-CUSP scheme. The formulation with $\phi = H$ preserves the total enthalpy [44] and is therefore suitable for inviscid flows. The left (L) and right (R) state is evaluated similarly to the MUSCL approach [30], using the limited interpolation (4.118)–(4.121). The two factors $\alpha^* c$ and β in Eq. (4.79) are defined as

$$\alpha^* c = \begin{cases} |V| & \text{if } \beta = 0, \\ -(1 + \beta)\Lambda^- & \text{if } \beta > 0 \text{ and } 0 < M_n < 1, \\ +(1 - \beta)\Lambda^+ & \text{if } \beta < 0 \text{ and } -1 < M_n < 0 \\ 0 & \text{if } |M_n| \geq 1, \end{cases} \quad (4.80)$$

and

$$\beta = \begin{cases} +\max\left(0, \frac{V + \Lambda^-}{V - \Lambda^-}\right) & \text{if } 0 \leq M_n < 1, \\ -\max\left(0, \frac{V + \Lambda^+}{V - \Lambda^+}\right) & \text{if } -1 < M_n < 0, \\ \text{sign}(M_n) & \text{if } |M_n| \geq 1 \end{cases} \quad (4.81)$$

with $M_n = V/c$. The contravariant velocity V in Eqs. (4.80) and (4.81) is computed from the arithmetic mean of the velocity components, that is,

$$V_{I+1/2} = \frac{1}{2} [(u_I + u_{I+1}) n_x + (v_I + v_{I+1}) n_y + (w_I + w_{I+1}) n_z]. \quad (4.82)$$

The speed of sound c in the evaluation of M_n is also obtained from arithmetically averaged quantities.

The positive and negative eigenvalues Λ^+ and Λ^- in Eqs. (4.80), (4.81) are those of the *Roe matrix* [64], which will be discussed in Section 4.3.3. The eigenvalues are given in Ref. [61]

$$\Lambda^{\pm} = \frac{\gamma + 1}{2\gamma} \tilde{V} \pm \sqrt{\left(\frac{\gamma - 1}{2\gamma} \tilde{V}\right)^2 + \frac{\tilde{c}^2}{\gamma}}, \quad (4.83)$$

where \tilde{V} denotes the contravariant velocity, γ is the ratio of specific heat coefficients and \tilde{c} stands for the speed of sound. The flow variables in Eq. (4.83), which have to be evaluated at the faces of the control volumes, are obtained using the *Roe averages* [64]

$$\begin{aligned} \tilde{u}_{I+1/2} &= \frac{u_L \sqrt{\rho_L} + u_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \tilde{v}_{I+1/2} &= \frac{v_L \sqrt{\rho_L} + v_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \tilde{w}_{I+1/2} &= \frac{w_L \sqrt{\rho_L} + w_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \tilde{H}_{I+1/2} &= \frac{H_L \sqrt{\rho_L} + H_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \tilde{c}_{I+1/2} &= \sqrt{(\gamma - 1) \left(\tilde{H} - \frac{\tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2}{2} \right)_{I+1/2}}, \\ \tilde{V}_{I+1/2} &= \tilde{u}_{I+1/2} n_x + \tilde{v}_{I+1/2} n_y + \tilde{w}_{I+1/2} n_z. \end{aligned} \quad (4.84)$$

The factors $\alpha^* c$ and β are defined such that full upwinding of the convective fluxes results for supersonic flow, that is, $\alpha^* c = 0$ and $\beta = \text{sign}(M_n)$. On the other hand, in subsonic flow (when $\beta = 0$) the dissipation is scaled by $|V|$. This is a desirable property for the computation of viscous layers. In cases of large aspect ratio cells, explicit time-stepping schemes usually require increased numerical dissipation in the direction of the longer cell side in order to stay robust. This can be accomplished by employing ratios of the spectral radii similar to Eq. (4.57). More details can be found in Ref. [38], which also contains comparisons between the CUSP scheme and the scalar as well as the matrix artificial dissipation (Section 4.3.1) scheme.

The implementation of the CUSP scheme can be considerably simplified by replacing the Roe averages from Eq. (4.84) by arithmetic averages and the eigenvalues from Eq. (4.83) by $\Lambda^{\pm} = V \pm c$. Introducing the definition

$$\alpha^* c = \alpha c - \beta V, \quad (4.85)$$

the factors of the modified scheme read [44, 61]

$$\alpha = \begin{cases} |M_n| & \text{if } |M_n| \geq \delta, \\ \frac{M_n^2 + \delta^2}{2\delta} & \text{if } |M_n| < \delta, \end{cases} \quad (4.86)$$

and

$$\beta = \begin{cases} \max(0, 2M_n - 1) & \text{if } 0 \leq M_n < 1, \\ \min(0, 2M_n + 1) & \text{if } -1 < M_n < 0, \\ \text{sign}(M_n) & \text{if } |M_n| \geq 1. \end{cases} \quad (4.87)$$

The contravariant velocity in $M_n = V/c$ is still computed as indicated in Eq. (4.82). The parameter δ in Eq. (4.86) is intended to prevent the dissipation from disappearing at stagnation points, but this does not seem to be always necessary. The above simplifications lead to a computationally very efficient scheme, however the shock resolution is slightly reduced as compared to the original formulation with Roe averages.

4.3.3 Flux-difference splitting schemes

The flux-difference splitting schemes evaluate the convective fluxes at a face of the control volume from the (in general discontinuous) left and right states by solving the Riemann (shock tube) problem. The idea was first introduced by Godunov [65]. In contrast to the flux-vector splitting schemes, the flux-difference splitting considers not only the direction of wave (information) propagation, but also the waves themselves. In order to reduce the computational effort of Godunov's scheme for the exact solution of the Riemann problem, approximate Riemann solvers were developed, for example, by Osher et al. [66] and by Roe [64]. In particular, Roe's method is applied quite often because of its high accuracy in boundary layer flows and good resolution of shocks. Therefore, the Roe solver shall be presented in more detail in the following section.

Roe upwind scheme

Roe's approximate Riemann solver can be implemented either in the framework of the cell-centered scheme or the dual control-volume scheme. It is based on the decomposition of the flux difference over a face of the control volume into a sum of wave contributions, while ensuring the conservation properties of the Euler equations. On the face $(I + 1/2)$ or $(i + 1/2)$, respectively, the difference is expressed as [64]

$$(\vec{F}_c)_R - (\vec{F}_c)_L = (\bar{A}_{\text{Roe}})_{I+1/2} (\vec{W}_R - \vec{W}_L). \quad (4.88)$$

where \bar{A}_{Roe} denotes the *Roe matrix*, and L or R the left and right states (see Fig. 4.8), respectively. The Roe matrix is identical to the convective flux Jacobian \bar{A}_c (see Section A.9), where the flow variables are replaced by the *Roe-averaged* variables. The flux difference in Eq. (4.88) is exact, if the Roe's averages are computed from the left and the right state by the following formulas [64, 67]

$$\begin{aligned}
\tilde{\rho} &= \sqrt{\rho_L \rho_R}, \\
\tilde{u} &= \frac{u_L \sqrt{\rho_L} + u_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\
\tilde{v} &= \frac{v_L \sqrt{\rho_L} + v_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\
\tilde{w} &= \frac{w_L \sqrt{\rho_L} + w_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\
\tilde{H} &= \frac{H_L \sqrt{\rho_L} + H_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\
\tilde{c} &= \sqrt{(\gamma - 1) (\tilde{H} - \tilde{q}^2/2)}, \\
\tilde{V} &= \tilde{u} n_x + \tilde{v} n_y + \tilde{w} n_z, \\
\tilde{q}^2 &= \tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2.
\end{aligned} \tag{4.89}$$

We can make the decomposition into waves in Roe's scheme clearer when we insert the diagonalization of the Roe matrix, that is, $\bar{A}_{Roe} = \bar{T} \bar{\Lambda}_c \bar{T}^{-1}$, into Eq. (4.88)

$$(\vec{F}_c)_R - (\vec{F}_c)_L = \bar{T} \bar{\Lambda}_c (\vec{C}_R - \vec{C}_L). \tag{4.90}$$

The matrix of left (\bar{T}^{-1}) and right (\bar{T}) eigenvectors, as well as the diagonal matrix of eigenvalues ($\bar{\Lambda}_c$) are evaluated using Roe's averaging (4.89). In the above equation (4.90), the characteristic variables \vec{C} represent the wave amplitudes, the eigenvalues Λ_c are the associated wave speeds of the approximate Riemann problem, and finally the right eigenvectors are the waves themselves.

Following from the previous discussion, the convective fluxes are evaluated at the faces of a control volume faces according to the formula [64]

$$(\vec{F}_c)_{I+1/2} = \frac{1}{2} [\vec{F}_c(\vec{W}_R) + \vec{F}_c(\vec{W}_L) - |\bar{A}_{Roe}|_{I+1/2} (\vec{W}_R - \vec{W}_L)]. \tag{4.91}$$

The product of $|\bar{A}_{Roe}|$ and the difference of the left and right states can be efficiently evaluated as follows

$$|\bar{A}_{Roe}|(\vec{W}_R - \vec{W}_L) = |\Delta \vec{F}_1| + |\Delta \vec{F}_{2,3,4}| + |\Delta \vec{F}_5|, \tag{4.92}$$

where

$$|\Delta \vec{F}_1| = |\tilde{V} - \tilde{c}| \left(\frac{\Delta p - \tilde{\rho} \tilde{c} \Delta V}{2 \tilde{c}^2} \right) \begin{bmatrix} 1 \\ \tilde{u} - \tilde{c} n_x \\ \tilde{v} - \tilde{c} n_y \\ \tilde{w} - \tilde{c} n_z \\ \tilde{H} - \tilde{c} \tilde{V} \end{bmatrix}, \tag{4.93}$$

$$|\Delta \vec{F}_{2,3,4}| = |\tilde{V}| \left\{ \left(\Delta \rho - \frac{\Delta p}{\tilde{c}^2} \right) \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \tilde{q}^2/2 \end{bmatrix}, \right. \\ \left. + \tilde{\rho} \begin{bmatrix} 0 \\ \Delta u - \Delta V n_x \\ \Delta v - \Delta V n_y \\ \Delta w - \Delta V n_z \\ \tilde{u} \Delta u + \tilde{v} \Delta v + \tilde{w} \Delta w - \tilde{V} \Delta V \end{bmatrix} \right\}, \quad (4.94)$$

$$|\Delta \vec{F}_5| = |\tilde{V} + \tilde{c}| \left(\frac{\Delta p + \tilde{\rho} \tilde{c} \Delta V}{2 \tilde{c}^2} \right) \begin{bmatrix} 1 \\ \tilde{u} + \tilde{c} n_x \\ \tilde{v} + \tilde{c} n_y \\ \tilde{w} + \tilde{c} n_z \\ \tilde{H} + \tilde{c} \tilde{V} \end{bmatrix}. \quad (4.95)$$

The jump condition is defined as $\Delta(\bullet) = (\bullet)_R - (\bullet)_L$ and the Roe-averaged variables are given in Eq. (4.89), respectively.

The left and the right state are determined using the MUSCL scheme [30], which is given in Eq. (4.46). All higher-order schemes ($\hat{\kappa} = -1$, $\hat{\kappa} = 0$, and $\hat{\kappa} = 1/3$) have to be supplemented by limiters (Section 4.3.5), if the flow field contains any discontinuities.

Because of the formulation in Eq. (4.88), Roe's approximate Riemann solver will produce an unphysical expansion shock in the case of stationary expansion, for which $(\vec{F}_c)_L = (\vec{F}_c)_R$ but $\vec{W}_L \neq \vec{W}_R$. Furthermore, the “carbuncle phenomenon” may occur, where a perturbation grows ahead of a strong bow shock along the stagnation line [68, 69]. See also the discussion in Ref. [70]. The underlying difficulty is that the original scheme does not recognize the sonic point. In order to solve this problem, the modulus of the eigenvalues $|\Lambda_c| = |\tilde{V} \pm \tilde{c}|$ is modified using Harten's *entropy correction* [71, 72]

$$|\Lambda_c| = \begin{cases} |\Lambda_c| & \text{if } |\Lambda_c| > \delta \\ \frac{\Lambda_c^2 + \delta^2}{2\delta} & \text{if } |\Lambda_c| \leq \delta, \end{cases} \quad (4.96)$$

where δ is a small value, which can be conveniently set equal to some fraction (e.g., 1/10 or 1/20) of the local speed of sound. In order to prevent the linear waves $|\Delta \vec{F}_{2,3,4}|$ from disappearing for $\tilde{V} \rightarrow 0$ (e.g., at stagnation points or for grid-aligned flow), the above modification can also be applied to $|\tilde{V}|$.

A clear disadvantage of the Roe solver as compared to the central scheme or to the flux-vector splitting schemes shows up for a real gas simulation. Namely, the Roe matrix and averaging have to be changed correspondingly, which may become quite

complicated. The reader may find examples of formulations for equilibrium as well as non-equilibrium real gas flows in Refs. [73–76] and in the references cited therein. An implementation of Roe's scheme for arbitrary compressible and incompressible fluids was recently described in Ref. [77]. Note also that Section 9.5 contains a formulation of Roe scheme preconditioned for low Mach numbers (see Eq. (9.82)).

4.3.4 Total variation diminishing schemes

The idea of total variation diminishing (TVD) schemes was first pursued by Harten [78]. The TVD schemes are based on a concept aimed at preventing the generation of new extrema in the flow solution. The principal condition for a TVD scheme is that the total variation of the solution, defined as

$$\text{TV} \equiv \sum_I |U_{I+1} - U_I| \quad (4.97)$$

for a scalar conservation equation, decreases in time. This implies that maxima in the solution must be non-increasing and minima non-decreasing. Hence no new local extrema may be created during the time evolution. Thus, a discretization methodology with TVD properties allows it to resolve strong shock waves accurately, without any spurious oscillations of the solution, as they are for example generated by the central scheme with scalar or matrix artificial dissipation ([Section 4.3.1](#)).

The TVD schemes are implemented as an average of the convective fluxes combined with an additional dissipation term (flux-limited dissipation), which complies with the TVD conditions [78, 79]. If the dissipation term depends on the sign of the characteristic speeds, we speak of a *symmetric* TVD scheme [80, 81], otherwise of an *upwind* TVD scheme [82–86]. Experience shows that the upwind TVD scheme offers higher accuracy than the symmetric TVD scheme [87]. The upwind TVD scheme is particularly suitable for the simulation of supersonic and hypersonic flow fields [88]. It is also capable of accurate resolution of boundary layers [54], especially if the modification described in Ref. [89] is applied.

Upwind TVD scheme

In this framework, the convective fluxes through the face $(I + 1/2)$ of the control volume (see [Fig. 4.8](#)) can be expressed as

$$(\vec{F}_c)_{I+1/2} = \frac{1}{2} [(\vec{F}_c)_{I+1} + (\vec{F}_c)_I] + \frac{1}{2} \bar{T}_{I+1/2} \vec{\Theta}_{I+1/2}. \quad (4.98)$$

In the case of the cell-vertex scheme with dual control volumes ([Section 4.2.3](#)), the indexes would read $(i + 1/2)$, $(i + 1)$, etc. The matrix \bar{T} contains the right eigenvectors of the Jacobian $\bar{A}_c = \partial \vec{F}_c / \partial \vec{W}$. The entries of the matrix can be found in Section A.11. In Eq. (4.98), the term $\vec{\Theta}$ takes account of the direction of the characteristic speeds.

It controls the upwind direction of the difference operator. The l th component of the vector $\vec{\Theta}$ is defined as (cf. [85])

$$\Theta_{I+1/2}^l = \frac{1}{2} \psi \left(\Lambda_{I+1/2}^l \right) \left(\Psi_{I+1}^l + \Psi_I^l \right) - \psi \left(\Lambda_{I+1/2}^l + \chi_{I+1/2}^l \right) \Delta C_{I+1/2}^l, \quad (4.99)$$

where Λ^l represents the individual eigenvalues of the diagonal matrix $\bar{\Lambda}_c$ (see Section A.11), and Ψ the limiter function (Eq. (4.122)), respectively. Furthermore,

$$\chi_{I+1/2}^l = \frac{1}{2} \psi \left(\Lambda_{I+1/2}^l \right) \cdot \begin{cases} \frac{(\Psi_{I+1}^l - \Psi_I^l)}{\Delta C_{I+1/2}^l} & \text{if } \Delta C_{I+1/2}^l \neq 0, \\ 0 & \text{if } \Delta C_{I+1/2}^l = 0, \end{cases} \quad (4.100)$$

and finally ΔC^l are the elements of the difference of characteristic variables, that is,

$$\Delta \vec{C}_{I+1/2} = \bar{T}_{I+1/2}^{-1} (\vec{W}_{I+1} - \vec{W}_I) \quad (4.101)$$

with \bar{T}^{-1} being the matrix of left eigenvectors. The *entropy correction* of Harten [71, 72], that is,

$$\psi(z) = \begin{cases} |z| & \text{if } |z| > \delta_1, \\ \frac{z^2 + \delta_1^2}{2\delta_1} & \text{if } |z| \leq \delta_1 \end{cases} \quad (4.102)$$

prevents the value $\psi(z)$ from vanishing for $|z| \rightarrow 0$. The parameter δ_1 is best formulated as function of the velocity components and the speed of sound [85]

$$(\delta_1)_{I+1/2} = \delta \left(|u_{I+1/2}| + |v_{I+1/2}| + |w_{I+1/2}| + c_{I+1/2} \right), \quad (4.103)$$

where $0.05 \leq \delta \leq 0.5$. Values of the primitive variables at the face ($I + 1/2$) are obtained either from Roe's (4.89) or from simple arithmetic averaging of the states at I and $(I + 1)$. The limiter function Ψ , which prevents the generation of spurious solutions near strong gradients, will be presented in the following section. It should be stressed that the above upwind TVD scheme does not employ the MUSCL approach to achieve higher order accuracy.

One can show that the upwind TVD method is precisely of first-order in space when the limiter function Ψ in Eqs. (4.99) and (4.100) is set equal to zero [90], which happens at discontinuities. Otherwise, the upwind TVD scheme, as presented above, is second-order accurate in smooth flow regions.

4.3.5 Limiter functions

Second- and higher-order upwind spatial discretizations require the use of *limiters* or *limiter functions* in order to prevent the generation of oscillations and spurious solutions in regions with large gradients (e.g., at shocks). Hence, what we are looking for is at least a *monotonicity preserving* scheme. This means that maxima in the flow field must

be non-increasing, minima non-decreasing, and no new local extrema may be created during the time evolution. Or in other words, if the initial data is monotone then the solution has to remain monotone. The rather stringent conditions for monotonicity preserving schemes (or the more rigorous ones for TVD schemes) are often given up in favor of the *local extremum diminishing* (LED) conditions [61]. Here, a local extremum contained only **within** the stencil has to decrease.

However, due to Godunov's theorem there is no possibility for a higher-order linear scheme (such as the MUSCL approach) to be monotonicity preserving [91]. It is therefore necessary to employ non-linear limiter functions in order to construct a monotonicity preserving or a TVD discretization. This is demonstrated in Fig. 4.10, where the upwind TVD scheme of Eq. (4.98) was used with and without a limiter to compute 2-D transonic flow past the NACA 0012 airfoil. It can be clearly seen that without limiter, the solution exhibits large oscillations in the neighborhood of the shocks on the upper and the lower side of the airfoil. On the other hand, the limited and the unlimited solutions become nearly identical away from the shocks.

The purpose of a limiter is to reduce the slopes (i.e., $(U_{I+1} - U_I)/\Delta x$) used to interpolate a flow variable to the face of a control volume in order to constrain the

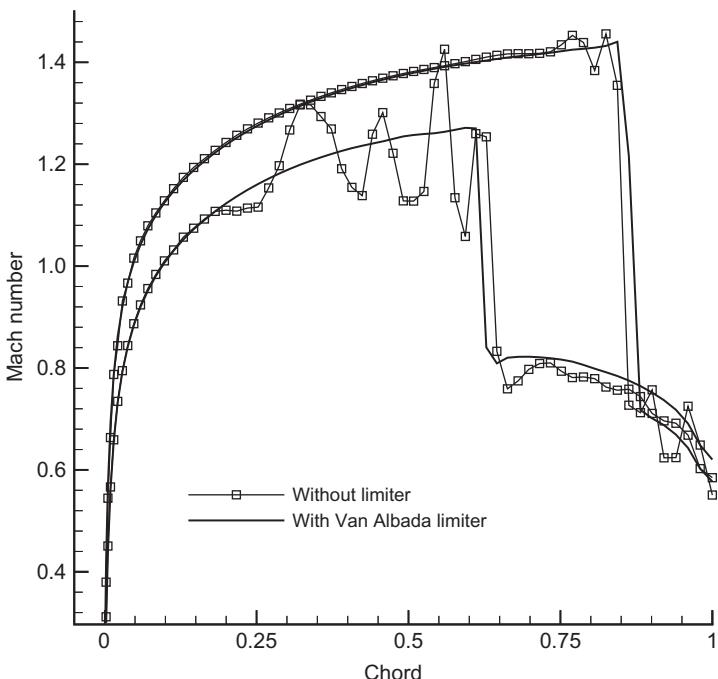


Figure 4.10 Comparison of inviscid transonic flow computation with and without limiter. NACA 0012 airfoil, $M_\infty = 0.85$, $\alpha = 1^\circ$.

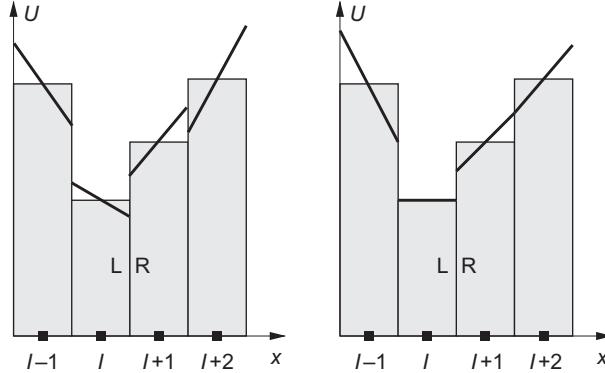


Figure 4.11 Comparison of direct (left) and limited (right) interpolation to the cell faces. Thick lines denote slopes $\Delta U / \Delta x$, bars represent values at cell centers.

solution variations. At strong discontinuities, the limiter has to reduce the slopes to zero to prevent the generation of new extrema. This implies for the MUSCL approach as well as for the TVD schemes that the (monotone) first-order upwind scheme ($\epsilon = 0$ in Eq. (4.46)) is recovered in the immediate vicinity of large gradients. The last requirement to be imposed on a limiter is quite obvious—the original unlimited discretization has to be obtained in smooth flow regions, in order to keep the amount of numerical dissipation as low as possible. The effect of a limiter on the interpolation of the left and right states is sketched in Fig. 4.11. The example shows the slope reduction at the local minimum at I and the change of the slope at the cells $(I + 1)$, $(I + 2)$ to achieve a monotone solution. It is important to realize that a difference between the left and right states at a face may (and generally will) still be present.

In the following, we shall describe four different limiter functions, which are well-established and proven in practice. We shall consider limiters for the second-order MUSCL, for the CUSP and for the upwind TVD scheme.

Limiter functions for MUSCL interpolation

Van Leer's MUSCL approach [30] is turned into a monotonicity preserving scheme by employing a limiter function to reduce the differences $\Delta_+ U_I$ and $\Delta_- U_I$ in Eq. (4.47) when necessary. Introducing *slope limiters* Φ^\pm , the MUSCL interpolation formulas in Eq. (4.46) are modified as follows (see also Fig. 4.8)

$$\begin{aligned} U_R &= U_{I+1} - \frac{1}{4} \left[(1 + \hat{\kappa}) \Phi_{I+1/2}^+ \Delta_- + (1 - \hat{\kappa}) \Phi_{I+3/2}^- \Delta_+ \right] U_{I+1}, \\ U_L &= U_I + \frac{1}{4} \left[(1 + \hat{\kappa}) \Phi_{I+1/2}^- \Delta_+ + (1 - \hat{\kappa}) \Phi_{I-1/2}^+ \Delta_- \right] U_I. \end{aligned} \quad (4.104)$$

The parameter ϵ in Eq. (4.46) was set equal to unity. The slope limiters are functions of the ratios of the consecutive solution variations, that is, $\Phi_{I+1/2}^{\pm} = \Phi(r_{I+1/2}^{\pm})$, with the definitions [1]

$$\begin{aligned} r_{I+1/2}^{+} &= \frac{U_{I+2} - U_{I+1}}{U_{I+1} - U_I}, \\ r_{I+1/2}^{-} &= \frac{U_I - U_{I-1}}{U_{I+1} - U_I}, \text{ etc.} \end{aligned} \quad (4.105)$$

If we substitute now r_L for $r_{I-1/2}^{+}$ and r_R for $r_{I+3/2}^{-}$, thus

$$\begin{aligned} r_R &= \frac{U_{I+1} - U_I}{U_{I+2} - U_{I+1}} = \frac{\Delta_-}{\Delta_+} U_{I+1}, \\ r_L &= \frac{U_{I+1} - U_I}{U_I - U_{I-1}} = \frac{\Delta_+}{\Delta_-} U_I, \end{aligned} \quad (4.106)$$

we can write Eq. (4.104) in the form

$$\begin{aligned} U_R &= U_{I+1} - \frac{1}{4} [(1 + \hat{\kappa})r_R \Phi(1/r_R) + (1 - \hat{\kappa})\Phi(r_R)] (U_{I+2} - U_{I+1}), \\ U_L &= U_I + \frac{1}{4} [(1 + \hat{\kappa})r_L \Phi(1/r_L) + (1 - \hat{\kappa})\Phi(r_L)] (U_I - U_{I-1}). \end{aligned} \quad (4.107)$$

The above relationships Eq. (4.107) can be simplified if we consider only slope limiters with the symmetry property

$$\Phi(r) = \Phi(1/r). \quad (4.108)$$

With this definition, the limited MUSCL interpolation equation (4.104) becomes [92]

$$\begin{aligned} U_R &= U_{I+1} - \frac{1}{2} \Psi_R (U_{I+2} - U_{I+1}), \\ U_L &= U_I + \frac{1}{2} \Psi_L (U_I - U_{I-1}) \end{aligned} \quad (4.109)$$

with the **limiter function** defined as

$$\Psi_{L/R} = \frac{1}{2} [(1 + \hat{\kappa}) r_{L/R} + (1 - \hat{\kappa})] \Phi_{L/R}. \quad (4.110)$$

Different formulations of the slope limiter Φ in Eq. (4.110) are now possible, which can be tailored to specific values of $\hat{\kappa}$ to give the most accurate but stable and monotonicity preserving MUSCL scheme.

MUSCL scheme with $\hat{\kappa} = 0$

One particularly suitable combination for the second-order, upwind-biased scheme with $\hat{\kappa} = 0$ is [93]

$$\Phi(r) = \frac{2r}{r^2 + 1}. \quad (4.111)$$

In this case, the function $\Psi(r)$ corresponds to the Van Albada limiter [94]

$$\Psi(r) = \frac{r^2 + r}{1 + r^2}, \quad (4.112)$$

and we obtain with Eq. (4.109) the following expressions for the left and right states

$$\begin{aligned} U_R &= U_{I+1} - \frac{1}{2}\delta_R, \\ U_L &= U_I + \frac{1}{2}\delta_L. \end{aligned} \quad (4.113)$$

The function δ is formally identical for both states. It reads

$$\delta = \frac{a(b^2 + \epsilon) + b(a^2 + \epsilon)}{a^2 + b^2 + 2\epsilon}. \quad (4.114)$$

The coefficients a and b are defined for the left and right states as

$$\begin{aligned} a_R &= \Delta_+ U_{I+1}, & b_R &= \Delta_- U_{I+1}, \\ a_L &= \Delta_+ U_I, & b_L &= \Delta_- U_I \end{aligned} \quad (4.115)$$

and the difference operators Δ_{\pm} are given by Eq. (4.47). The additional parameter ϵ in Eq. (4.114) prevents the activation of the limiter in smooth flow regions due to small-scale oscillations [93]. This is sometimes necessary in order to achieve a fully converged steady-state solution. The parameter ϵ is conveniently set proportional to the local grid scale, in 3D for example to $\Omega^{1/3}$ [93, 95]. Additional scaling of the parameter ϵ is required if the particular state variable U is given in physical units. It can be shown that the relations in Eq. (4.113) are identical to the original (unlimited) MUSCL scheme (4.46) with $\hat{\kappa} = 0$ for smoothly varying flow. Thus, the accuracy of the solution is not influenced. On the other hand, the function δ becomes zero at local extrema, reducing the accuracy to first order as desired.

MUSCL scheme with $\hat{\kappa} = 1/3$

Another limiter function was devised for the three-point, second-order accurate upwind-biased MUSCL scheme with $\hat{\kappa} = 1/3$. Here, the slope limiter is given by

$$\Phi(r) = \frac{3r}{2r^2 - r + 2}. \quad (4.116)$$

In this case, the function $\Psi(r)$ corresponds to the limiter of Hemker and Koren [96]. Following the same way as in the previous case, we obtain formulas for the left

and right states at the face ($I + 1/2$) which are identical to Eq. (4.113), but now with [93]

$$\delta = \frac{(2a^2 + \epsilon)b + (b^2 + 2\epsilon)a}{2a^2 + 2b^2 - ab + 3\epsilon}. \quad (4.117)$$

The definitions of the coefficients a , b , and of the parameter ϵ are retained.

Limiter for CUSP scheme

In the framework of the CUSP scheme (Section 4.3.2), the left (L) and right (R) states are evaluated to second-order accuracy according to [44]

$$\begin{aligned} U_R &= U_{I+1} - \frac{1}{2}L(\Delta U_{I+3/2}, \Delta U_{I-1/2}), \\ U_L &= U_I + \frac{1}{2}L(\Delta U_{I+3/2}, \Delta U_{I-1/2}), \end{aligned} \quad (4.118)$$

where

$$\begin{aligned} \Delta U_{I-1/2} &= U_I - U_{I-1}, \\ \Delta U_{I+3/2} &= U_{I+2} - U_{I+1}. \end{aligned} \quad (4.119)$$

In the above equations (4.118) and (4.119), U represents a dependent variable and $L()$ the limited average

$$L(\Delta_1, \Delta_2) = \frac{1}{2}\Psi(\Delta_1, \Delta_2)(\Delta_1 + \Delta_2), \quad (4.120)$$

respectively. The limiter itself is defined as

$$\Psi(\Delta_1, \Delta_2) = 1 - \left| \frac{\Delta_1 - \Delta_2}{|\Delta_1| + |\Delta_2| + \epsilon} \right|^{\sigma}, \quad (4.121)$$

where σ is a positive coefficient which is usually set equal to two. The constant ϵ is required to prevent division by zero (e.g., $\epsilon = 10^{-20}$). If Δ_1 and Δ_2 happen to have opposite sign but the same magnitude, the limiter becomes $\Psi = 0$. This means that we obtain only a first-order accurate approximation for the left and the right state.

It should be mentioned that it is also possible to employ the MUSCL scheme with $\hat{k} = 0$ and the Van Albada limiter from Eqs. (4.113)–(4.115) instead of the above relations.

Limiter for TVD scheme

In comparison to the previous cases, the limiter here acts not on the conservative or the primitive variables, but on the characteristic variables \vec{C} . One particularly suitable limiter function is given in Ref. [85]

$$\Psi_I^l = \frac{\Delta C_{I-1/2}^l \Delta C_{I+1/2}^l + |\Delta C_{I-1/2}^l \Delta C_{I+1/2}^l|}{\Delta C_{I-1/2}^l + \Delta C_{I+1/2}^l + \epsilon}, \quad (4.122)$$

where the $\Delta C_{I+1/2}^l$ represents the difference of the characteristic variables at face $(I + 1/2)$ of the control volume (Eq. (4.101)). The positive constant $\epsilon \approx 10^{-20}$ in the denominator prevents division by zero. In regions with high gradients, the limiter function becomes zero, which leads with Eqs. (4.99) and (4.98) to first-order accurate upwind scheme. The upwind TVD scheme in Eq. (4.98) retains second-order accuracy in areas with smoothly varying flow variables, where $\Psi_I^l = C_I^l - C_{I-1}^l$.

4.4 DISCRETIZATION OF THE VISCOUS FLUXES

The control volume for the viscous fluxes is generally chosen to be the same as for the convective fluxes in order to obtain a consistent spatial discretization. An exception is made only in the case of the cell-vertex scheme with overlapping control volumes ([Section 4.2.2](#)), where the dual control volume ([Section 4.2.3](#)) is employed instead, primarily due to stability reasons [97–99]. The viscous fluxes \vec{F}_v in the discretized governing equations (4.2) are, similar to Eqs. (4.17), (4.22), (4.37), (4.42), evaluated from variables averaged at the faces of the control volume. This is in line with the elliptic nature of the viscous fluxes. Thus, values of the velocity components (u, v, w) , the dynamic viscosity μ , and of the heat conduction coefficient k , which are required for the computation of the viscous terms (2.23), (2.24) and of the stresses (2.15), are simply averaged at a face. In the case of the cell-centered scheme ([Figs. 4.3](#) and [4.8](#)), the values at the face $(I + 1/2)$ of the control volume result from

$$U_{I+1/2} = \frac{1}{2} (U_I + U_{I+1}), \quad (4.123)$$

where U is any of the above flow variables. The same holds in the case of both cell-vertex schemes for the face $(i + 1/2)$ —see [Figs. 4.5](#) and [4.8](#), respectively.

The remaining task is the evaluation of the first derivatives (gradients) of the velocity components in Eq. (2.15) and of the temperature in Eq. (2.24). This can be accomplished in one of two ways, that is, by using

- finite differences, or
- Green's theorem.

The first approach applies a local transformation from Cartesian coordinates (x, y, z) to the curvilinear coordinates (ξ, η, ζ) , for example,

$$\frac{\partial U}{\partial x} = \frac{\partial U}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial U}{\partial \eta} \frac{\partial \eta}{\partial x} + \frac{\partial U}{\partial \zeta} \frac{\partial \zeta}{\partial x}, \text{ etc.} \quad (4.124)$$

The derivatives U_ξ , U_η , and U_ζ are obtained from finite-difference approximations. More details can be found in Refs. [97–99]. See Section A.1 for the derivatives of the coordinates and for the Jacobian of the transformation.

Here, we prefer the second approach, which is more in line with the finite-volume methodology treated in this book. However, it requires the construction of an additional control volume for the computation of the derivatives. This will be discussed below for the cell-centered and the cell-vertex scheme.

Once we obtained the values of the flow variables and of the first derivatives at the faces of the control volume, we can sum up the contributions due to the viscous fluxes according to Eq. (4.2). By adding the sum of the contributions to the inviscid fluxes, we completed the spatial discretization, and we can thus integrate the approximated governing equations in time.

4.4.1 Cell-centered scheme

In order to apply Green's theorem, which relates the volume integral of the first derivative to the surface integral of U , we have to define a suitable control volume first. Since we need the derivatives at the midpoints of the faces for the summation in Eq. (4.2), we construct an auxiliary control volume centered at the face by connecting the midpoints of the edges defining adjacent grid cells [20, 32, 99] as shown in Fig. 4.12a.

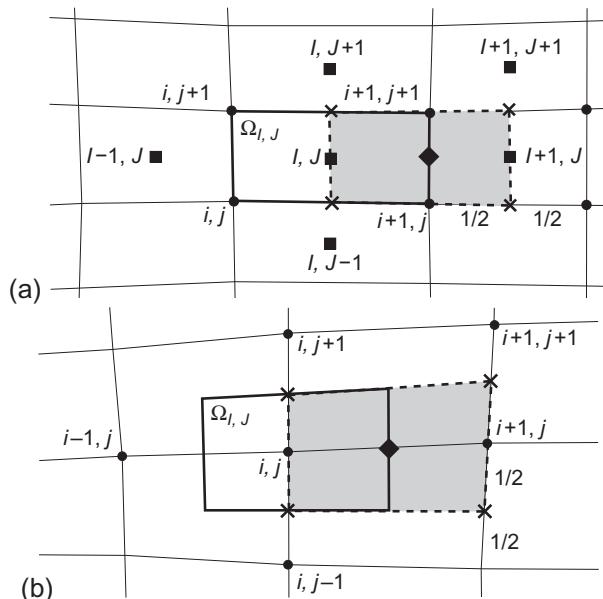


Figure 4.12 Auxiliary control volume Ω' (filled) for evaluation of first derivatives in 2D: (a) cell-centered scheme; (b) cell-vertex scheme. The diamond symbol denotes location where first derivatives are to be evaluated.

In order to evaluate the first derivative at the face ($I + 1/2$)—marked by a diamond symbol in Fig. 4.12a—we have to integrate the corresponding flow variable U over the boundary of the auxiliary control volume (denoted by the superscript ' in the following). Thus, for example, for the derivative in the x -direction

$$\frac{\partial U}{\partial x} = \frac{1}{\Omega'} \int_{\partial\Omega'} U dS'_x \approx \frac{1}{\Omega'} \sum_{m=1}^{N_F} U_m S'_{x,m}, \quad (4.125)$$

where N_F stands for the number of faces ($N_F = 4$ in 2D and $N_F = 6$ in 3D). The volume Ω' and the components of the face vector $\vec{S}'_m = [S'_{x,m}, S'_{y,m}, S'_{z,m}]^T$, respectively, are computed as already presented in Section 4.1. The face values U_m are obtained either directly as cell-centered values (i.e., $U_{i,j}$ and $U_{i+1,j}$ on the left and the right face), or by averaging like on the upper and the lower face, for example, at $J+1/2$

$$U_{m_{I,J+1/2}} = \frac{1}{4} (U_{I,J} + U_{I+1,J} + U_{I,J+1} + U_{I+1,J+1}), \text{ etc.} \quad (4.126)$$

We can apply the same approach in 3D, where again four cell-centered values can be utilized for the averaging. Hence,

$$\begin{aligned} U_{m_{I,J+1/2,K}} &= \frac{1}{4} (U_{I,J,K} + U_{I+1,J,K} + U_{I,J+1,K} + U_{I+1,J+1,K}), \\ U_{m_{I,J+1/2,K+1/2}} &= \frac{1}{4} (U_{I,J,K} + U_{I,J,K+1} + U_{I,J+1,K} + U_{I,J+1,K+1}), \\ &\dots \end{aligned} \quad (4.127)$$

The above scheme is quite compact, with the computational stencil extending over only nine cells in 2D and over 15 in 3D. It should be noted that this approach for computing the first derivatives cannot suppress the generation of two types of spurious modes (decoupled solutions at neighboring cell centers) [20, 100]: the checkerboard mode, arising from the form of the integral around the control volume, and a pair of corrugated or washboard modes, arising from the averaging of values in neighboring cells. However, there are generally no difficulties with this in practice. A more serious problem would occur, if the gradients would be first evaluated for each cell (similar to the convective fluxes) and then averaged at the cell faces. Although this approach may appear more attractive than the current methodology, it is not recommended since it leads to strong odd-even decoupling.

A disadvantage of above scheme is a loss of accuracy if the grid is not uniform [99, 100]. Namely, for arbitrarily stretched grids the approximation of the derivatives becomes inconsistent (the centroid of the auxiliary control volume does no longer correspond to the face center). Thus, the viscous fluxes are discretized with second-order accuracy only for moderately and smoothly stretched grids.

Finally, it should be noted that the TSL approximation of the Navier-Stokes equations (Section 2.4.3) can be realized easily by omitting the appropriate contributions when computing the gradients. For example, if the boundary layer would be oriented along the I -direction in Fig. 4.12a, contributions from the left (I, J) and the right side ($I + 1, J$) of the auxiliary control volume would be dropped.

4.4.2 Cell-vertex scheme

As already mentioned, both types of cell-vertex schemes resort to the dual control volume (Section 4.2.3) for the discretization of the viscous fluxes. Hence, the question is how to evaluate the first derivatives at the faces of this control volume. Considering Fig. 4.12b, one possible alternative is to calculate the gradients at the cell centers first by integrating over the grid cells, which yields first-order accuracy on arbitrarily stretched grids. In a next step, the cell-based gradients are averaged at the faces of the control volume Ω like in Refs. [32, 101]. However, this approach cannot prevent an odd-even decoupling of the solution.

Another possibility, similar to the cell-centered scheme, is to construct an auxiliary control volume around the face by connecting the midpoints of the edges defining adjacent grid cells [102, 103]. This is depicted in Fig. 4.12b. The evaluation of the first differences proceeds along the same lines as discussed for the cell-centered scheme, with averaged quantities where necessary. It should be noted that this approach is formally identical to the finite-difference approximation [97–99]. This scheme leads to first-order accurate discretization of the viscous fluxes on arbitrarily stretched grids and to second-order accuracy on smooth grids [97, 99]. Another positive feature is that the computational stencil is confined to only nine nodes in 2D and to 15 nodes in 3D.

Finally, one further approach should be mentioned, where a more complex integration path was chosen, with averaging incorporating all neighboring nodes [21]. A serious disadvantage of this scheme is that it encompasses a 25-point stencil even in 2D, which in general adds more numerical diffusion than compact stencils. Furthermore, if an implicit scheme would be envisioned for the time integration, the bandwidth of the flux Jacobian would become prohibitively large. A detailed discussion of various methodologies for the gradient evaluation can also be found in Ref. [104].

REFERENCES

- [1] Hirsch C. Numerical computation of internal and external flows, vols. 1 and 2. Chichester, UK: John Wiley and Sons; 1988.
- [2] Mohanraj R, Neumeier Y, Zinn BT. Characteristic-based treatment of source terms in Euler equations for Roe scheme. *AIAA J* 1999;37:417–24.
- [3] Jameson A, Baker TJ. Solution of the Euler equations for complex configurations. *AIAA Paper 83-1929*; 1983.
- [4] Rossow C-C. Berechnung von Strömungsfeldern durch Lösung der Euler-Gleichungen mit einer erweiterten Finite-Volumen Diskretisierungsmethode [Calculation of flow fields by the solution of

- Euler equations using an extended finite volume discretization scheme]. DLR Research Report No. 89-38; 1989.
- [5] Bruner, CWS. Geometric properties of arbitrary polyhedra in terms of face geometry. AIAA J 1995;33:1350.
 - [6] www.ma.ic.ac.uk/~rn/centroid.pdf
 - [7] Jameson A, Schmidt W, Turkel E. Numerical solutions of the euler equations by finite volume methods using Runge-Kutta time-stepping schemes. AIAA Paper 81-1259; 1981.
 - [8] Ni RH. Multiple grid scheme for solving the Euler equations. AIAA Paper 81-1025; 1981.
 - [9] Hall MG. Cell-vertex multigrid scheme for solution of the Euler equations. In: Proc. int. conf. on numerical methods for fluid dynamics. Reading, UK: Springer Verlag; 1985.
 - [10] Radespiel R. A cell-vertex multigrid method for the Navier-Stokes equations. NASA TM-101557; 1989.
 - [11] Radespiel R, Rossow C-C, Swanson RC. An efficient cell-vertex multigrid scheme for the three-dimensional Navier-Stokes equations. AIAA Paper 89-1953; 1989 [also in AIAA J 1990;28:1464-72].
 - [12] Rossow C-C. Flux balance splitting—a new approach for a cell-vertex upwind scheme. In: Proc. 12th int. conf. on numerical methods in fluid dynamics. Oxford, UK: Springer Verlag; 1990.
 - [13] Rossow C-C. Accurate solution of the 2D Euler equations with an efficient cell-vertex upwind scheme. AIAA Paper 93-0071; 1993.
 - [14] Usab WJ. Embedded mesh solutions of the Euler equations using a multiple-grid method. Ph.D. Thesis, MIT, Cambridge, MA, USA; 1983.
 - [15] Sidilkover D. A genuinely multidimensional upwind scheme and efficient multigrid solver for the compressible Euler equations. ICASE Report No. 94-84; 1994.
 - [16] Paillère H, Deconinck H, Roe PL. Conservative upwind residual-distribution schemes based on the steady characteristics of the Euler equations. AIAA Paper 95-1700; 1995.
 - [17] Issman E, Degrez G, Deconinck H. Implicit upwind residual-distribution Euler and Navier-Stokes solver on unstructured meshes. AIAA J 1996;34:2021-8.
 - [18] Van der Weide E, Deconinck H. Compact residual-distribution scheme applied to viscous flow simulations. VKI Lecture Series 1998-03; 1998.
 - [19] Dick E. A flux-difference splitting method for steady Euler equations. J Comput Phys 1988;76:19-32.
 - [20] Hall M. A vertex-centroid scheme for improved finite-volume solution of the Navier-Stokes equations. AIAA Paper 91-1540; 1991.
 - [21] Crumpton PI, Shaw GJ. A vertex-centred finite volume method with shock detection. Int J Numer Meth Fluids 1994;18:605-25.
 - [22] Roe PL. Error estimates for cell-vertex solutions of the compressible Euler equation. ICASE Report No. 87-6; 1987.
 - [23] Hoffman JD. Relationship between the truncation errors of centered finite-difference approximations on uniform and nonuniform meshes. J Comput Phys 1982;46:464-74.
 - [24] Arts T. On the consistency of four different control surfaces used for finite area blade-to-blade calculations. Int J Numer Meth Fluids 1984;4:1083-96.
 - [25] Turkel E. Accuracy of schemes with non-uniform meshes for compressible fluid flows. ICASE Report No. 85-43; 1985.
 - [26] Turkel E, Yaniv S, Landau U. Accuracy of schemes for the Euler equations with non-uniform meshes. ICASE Report No. 85-59; 1985.
 - [27] Rossow C-C. Comparison of cell-centered and cell-vertex finite volume schemes. In: Proc. 7th GAMM conference, notes on numerical fluid mechanics. Wiesbaden: Vieweg Publishing; 1987.
 - [28] Venkatakrishnan V. Implicit schemes and parallel computing in unstructured grid CFD. ICASE Report No. 95-28; 1995.
 - [29] Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. J. Comput Phys 1996;127:380-97.
 - [30] Van Leer B. Towards the ultimate conservative difference scheme. V. A second order sequel to Godunov's method. J Comput Phys 1979;32:101-36.

- [31] Leonard BP. Comparison of truncation error of finite-difference and finite-volume formulations of convection terms. NASA TM-105861; 1992.
- [32] Martinelli L. Calculation of viscous flows with a multigrid method. Ph.D. Thesis, Dept. of Mech. and Aerospace Eng., Princeton University; 1987.
- [33] Martinelli L, Jameson A. Validation of a multigrid method for Reynolds averaged equations. AIAA Paper 88-0414; 1988.
- [34] Yoon S, Kwak D. Artificial dissipation models for hypersonic external flow. AIAA Paper 88-3708; 1988.
- [35] Turkel E, Swanson RC, Vatsa VN, White JA. Multigrid for hypersonic viscous two- and three-dimensional flows. AIAA Paper 91-1572; 1991.
- [36] Swanson RC, Turkel E. On central difference and upwind schemes. J Comput Phys 1992;101:292-306.
- [37] Jameson A. Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence in transonic and hypersonic flow. AIAA Paper 93-3559; 1993.
- [38] Swanson RC, Radespiel R, Turkel E. Comparison of several dissipation algorithms for central difference schemes. ICASE Report No. 97-40; 1997 [also AIAA Paper 97-1945, 1997].
- [39] Pulliam TH. Artificial dissipation models for the Euler equations. AIAA J 1986;24:1931-40.
- [40] Van Leer B. Flux-vector splitting for the Euler equations. In: Proc. 8th Int. conf. on numerical methods in fluid dynamics, Springer Verlag; 1982. p. 507-12 [also ICASE Report 82-30; 1982].
- [41] Liou M-S, Steffen Jr CJ. A new flux splitting scheme. NASA TM-104404, 1991; also J Comput Phys 1993;107:23-39.
- [42] Liou M-S. A sequel to AUSM: $AUSM^+$. J Comput Phys 1996;129:364-82.
- [43] Jameson A. Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence in transonic and hypersonic flow. AIAA Paper 93-3559; 1993.
- [44] Tatsumi S, Martinelli L, Jameson A. A new high resolution scheme for compressible viscous flow with shocks. AIAA Paper 95-0466; 1995.
- [45] Edwards JR. A low-diffusion flux-splitting scheme for Navier-Stokes calculations. Comput Fluids 1997;26:653-9.
- [46] Rossow C-C. A simple flux splitting scheme for compressible flows. In: Proc. 11th DGLR-Fach-symposium, Berlin, Germany; November 10-12, 1998.
- [47] Rossow C-C. A flux splitting scheme for compressible and incompressible flows. AIAA Paper 99-3346; 1999.
- [48] Thomas JL, van Leer B, Walters RW. Implicit flux-split schemes for the Euler equations. AIAA Paper 85-1680; 1985.
- [49] Anderson WK, Thomas JL, van Leer B. A comparison of finite volume flux vector splittings for the Euler equations. AIAA J 1986;24:1453-60.
- [50] Hänel D, Schwane R, Seider G. On the accuracy of upwind schemes for the solution of the Navier-Stokes equations. AIAA Paper 87-1105; 1987.
- [51] Van Leer B, Thomas JL, Roe PL, Newsome RW. A comparison of numerical flux formulas for the Euler and Navier-Stokes equations. AIAA Paper 87-1104; 1987.
- [52] Hänel D, Schwane R. An implicit flux-vector scheme for the computation of viscous hypersonic flows. AIAA Paper 89-0274; 1989.
- [53] Van Leer B. Flux vector splitting for the 1990's. In: Invited Lecture for the CFD Symposium on Aeropropulsion, Cleveland, OH; 1990.
- [54] Seider G, Hänel D. Numerical influence of upwind TVD schemes on transonic airfoil drag prediction. AIAA Paper 91-0184; 1991.
- [55] Liou MS. On a new class of flux splittings. In: Proc. 13th int. conf. on numerical methods in fluid dynamics, Rome, Italy; 1992.
- [56] Wada Y, Liou M-S. A flux splitting scheme with high-resolution and robustness for discontinuities. AIAA Paper 94-0083; 1994.
- [57] Liou M-S. Progress towards an improved CFD method – $AUSM^+$. AIAA Paper 95-1701; 1995.
- [58] Radespiel R, Kroll N. Accurate flux vector splitting for shocks and shear layers. J Comput Phys 1995;121:66-78.

- [59] Edwards JR. Numerical implementation of a modified Liou-Steffen upwind scheme. AIAA J 1994;32:2120-22.
- [60] Jameson A. Positive schemes and shock modelling for compressible flow. Int J Numer Meth Fluids 1995;20:743-76.
- [61] Jameson A. Analysis and design of numerical schemes for gas dynamics. II: Artificial diffusion and discrete shock structure. Int J Comput Fluid Dynam 1995;5:1-38.
- [62] Tatsumi S, Martinelli L, Jameson A. A design, implementation and validation of flux limited schemes for the solution of the compressible Navier-Stokes equations. AIAA Paper 94-0647; 1994.
- [63] Nemec M, Zingg DW. Aerodynamic computations using the convective upstream split pressure scheme with local preconditioning. AIAA Paper 98-2444; 1998.
- [64] Roe PL. Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 1981;43:357-72.
- [65] Godunov SK. A difference scheme for numerical computation discontinuous solution of hydrodynamic equations. Math Sbornik [in Russian] 1959;47:271-306 [translated US Joint Publ. Res. Service, JPRS 7226, 1969].
- [66] Osher S, Solomon F. Upwind difference schemes for hyperbolic systems of conservation laws. Math Comput 1982;38:339-74.
- [67] Roe PL, Pike J. Efficient construction and utilisation of approximate Riemann solutions. In: Glowinski R, Lions JL, editors. Computing methods in applied sciences and engineering. The Netherlands: North Holland Publishing; 1984.
- [68] Peery KM, Imlay ST. Blunt-body flow simulations. AIAA Paper 88-2904; 1988.
- [69] Lin HC. Dissipation additions to flux-difference splitting. AIAA Paper 91-1544; 1991.
- [70] Quirk JJ. A contribution to the great Riemann solver debate. ICASE Report No. 92-64; 1992.
- [71] Harten A, Lax PD, Van Leer B. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. Soc Indust Appl Math Rev 1983;25(1):35-61.
- [72] Harten A, Hyman JM. Self adjusting grid methods for one-dimensional hyperbolic conservation laws. J Comput Phys 1983;50:235-69.
- [73] Vinokur M. Flux Jacobian matrices and generalized Roe average for an equilibrium real gas. NASA CR-177512; 1988.
- [74] Vinokur M, Montagné J-L. Generalized flux-vector splitting and Roe average for an equilibrium real gas. J Comput Phys 1990;89:276-300.
- [75] Grossman B, Cinnella P. Flux-split algorithms for flows with non-equilibrium chemistry and vibrational relaxation. J Comput Phys 1990;88:131-68.
- [76] Shuen J-S, Liou M-S, Van Leer B. Inviscid flux-splitting algorithms for real gases with non-equilibrium chemistry. J Comput Phys 1990;90:371-95.
- [77] Mulas M, Chibbaro S, Delussu G, Di Piazza I, Talice M. Efficient parallel computations of flows of arbitrary fluids for all regimes of Reynolds, Mach and Grashof Numbers. Int J Numer Meth Heat Fluid Flow 2002;12:637-57.
- [78] Harten A. High resolution schemes for hyperbolic conservation laws. J Comput Phys 1983;49:357-93.
- [79] Jameson A, Lax PD. Conditions for the construction of multi-point total variation diminishing difference schemes. MAE Report 1650, Dept. of Mechanical and Aerospace Engineering, Princeton University; 1984.
- [80] Davis SF. TVD finite difference schemes and artificial viscosity. ICASE Report No. 84-20; 1984.
- [81] Yee HC. Construction of implicit and explicit symmetric TVD schemes and their applications. J Comput Phys 1987;68:151-79.
- [82] Yee HC, Kutler P. Application of second-order accurate total variation diminishing (TVD) schemes to the Euler equations in general geometries. NASA TM-85845; 1983.
- [83] Yee HC. Upwind and symmetric shock-capturing schemes. NASA TM-89464; 1987.
- [84] Yee HC, Harten A. Implicit TVD schemes for hyperbolic conservation laws in curvilinear coordinates. AIAA J 1987;25:266-74.
- [85] Yee HC, Klopfer GH, Montagné J-L. High-resolution shock-capturing schemes for inviscid and viscous hypersonic flows. NASA TM-100097; 1988.

- [86] Yee HC. A class of high-resolution explicit and implicit shock-capturing methods. VKI Lecture Series 1989-04; 1989 [also NASA TM-101088; 1989].
- [87] Kroll N, Gaitonde D, Aftosmis M. A systematic comparative study of several high resolution schemes for complex problems in high speed flows. In: 29th AIAA Aerospace Sci. Meeting and Exhibit, Reno, USA; 1991.
- [88] Kroll N, Rossow C-C. A high resolution cell vertex tvd scheme for the solution of the two- and three-dimensional Euler equations. 12th International Conf. on Numerical Methods in Fluid Dynamics, Oxford, UK; 1990.
- [89] Müller B. Simple improvements of an upwind TVD scheme for hypersonic flow. AIAA Paper 89-1977; 1989.
- [90] Blazek J. Methods to accelerate the solutions of the Euler- and Navier-Stokes equations for steady-state super- and hypersonic flows. Translation of DLR Research Report 94-35, ESA-TT-1331; 1995.
- [91] LeVeque RJ. Numerical methods for conservation laws. Basel, Switzerland: Birkhäuser Verlag; 1992.
- [92] Spekreijse SP. Multigrid solution of the steady Euler equations. Ph.D. Dissertation, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands; 1987.
- [93] Venkatakrishnan V. Preconditioned conjugate gradient methods for the compressible Navier-Stokes equations. AIAA J 1991;29:1092-110.
- [94] Van Albada GD, Van Leer B, Roberts WW. A comparative study of computational methods in cosmic gas dynamics. Astron Astrophys 1982;108:76-84.
- [95] Venkatakrishnan V. On the accuracy of limiters and convergence to steady state solutions. AIAA Paper 93-0880; 1993.
- [96] Hemker PW, Koren B. Multigrid, defect correction and upwind schemes for the steady Navier-Stokes equations. In: Synopsis, HERMES Hypersonic Research Program Meeting, Stuttgart, Germany; 1987.
- [97] Radespiel R, Swanson RC. An investigation of cell centred and cell vertex multigrid schemes for the Navier-Stokes equations. AIAA Paper 89-0548; 1989.
- [98] Radespiel R. A cell-vertex multigrid method for the Navier-Stokes equations. NASA TM-101557; 1989.
- [99] Swanson RC, Radespiel R. Cell centered and cell vertex multigrid schemes for the Navier-Stokes equations. AIAA J 1991;29:697-703.
- [100] Morton KW, Paisley MF. On the cell-centre and cell-vertex approaches to the steady Euler equations and the use of shock fitting. In: Proc. int. conf. num. meth. fluid dynamics 10, Beijing, China; 1986.
- [101] Dimitriadiis KP, Leschziner MA. Multilevel convergence acceleration for viscous and turbulent transonic flows computed with cell-vertex method. In: Proc. 4th Copper Mountain conference on multigrid methods. Colorado: SIAM; 1989. p. 130-48.
- [102] Dick E. A flux-vector splitting method for steady Navier-Stokes equations. Int J Numer Meth Fluids 1988;8:317-26.
- [103] Dick E. A multigrid method for steady incompressible Navier-Stokes equations based on partial flux splitting. Int J Numer Meth Fluids 1989;9:113-20.
- [104] Crumpton PI, Mackenzie JA, Morton KW. Cell vertex algorithms for the compressible Navier-Stokes equations. J Comput Phys 1993;109:1-15.

CHAPTER 5

Unstructured Finite-Volume Schemes

Contents

5.1	Geometrical Quantities of a Control Volume	126
5.1.1	Two-dimensional case	126
<i>Triangular element</i>		126
<i>Quadrilateral element</i>		126
<i>Element center</i>		127
5.1.2	Three-dimensional case	128
<i>Triangular face</i>		128
<i>Quadrilateral face</i>		128
<i>Volume</i>		129
<i>Cell centroid</i>		130
5.2	General Discretization Methodologies	130
5.2.1	Cell-centered scheme	132
5.2.2	Median-dual cell-vertex scheme	134
5.2.3	Cell-centered versus median-dual scheme	137
<i>Accuracy</i>		137
<i>Computational work</i>		139
<i>Memory requirements</i>		139
<i>Grid generation/adaptation</i>		139
5.3	Discretization of the Convective Fluxes	139
5.3.1	Central scheme with artificial dissipation	140
5.3.2	Upwind schemes	144
5.3.3	Solution reconstruction	144
<i>Reconstruction based on MUSCL approach</i>		145
<i>Piecewise linear reconstruction</i>		146
<i>Linear reconstruction based on nodal weighting procedure</i>		147
<i>Piecewise quadratic reconstruction</i>		148
5.3.4	Evaluation of the gradients	149
<i>Green-Gauss approach</i>		150
<i>Least-squares approach</i>		151
5.3.5	Limiter functions	155
<i>Limiter of Barth and Jespersen</i>		155
<i>Venkatakrishnan's limiter</i>		156
5.4	Discretization of the Viscous Fluxes	157
5.4.1	Element-based gradients	159
<i>Face-centered control volume</i>		159
<i>Approximate Galerkin finite-element approach</i>		159
<i>Average of nodal values</i>		161
5.4.2	Average of gradients	161
	References	162

As we already noted in the introduction to Chapter 3, the vast majority of numerical schemes for the solution of the Euler- and the Navier-Stokes equations employs the *method of lines*, that is, a separate discretization in space and in time. The main advantage of this approach is that it allows us to select numerical approximations of different accuracy for the spatial and temporal derivatives. This offers a significantly larger flexibility as compared to methods based on *coupled* space and time discretization, like the Lax-Wendroff family of schemes (e.g., explicit MacCormack predictor-corrector scheme, implicit Lerat's scheme, etc.—details may be found, for example, in Ref. [1]). Because of the popularity of the method of lines, we shall follow this approach here.

The finite-volume schemes, which are discussed in this chapter, are based on the conservation laws, as they are represented by the Navier-Stokes (2.19) or by the Euler equations (2.44). In a pre-processing step, the physical domain is first subdivided into a number of elements (grid cells). In 2D, the elements are triangles, sometimes combined with quadrilaterals. In 3D, tetrahedra are most often employed [2–7]. However, an increasing number of flow solvers uses a mix of tetrahedra, prisms, pyramids, and in some cases also hexahedra (Fig. 5.1) for the simulation of high Reynolds number viscous flows [8–16]. Unstructured grids composed of various cell types are referred to as *mixed grids*. Examples are provided in Figs. 3.3 and 5.2. The designation “mixed grids” should not be confused with the term *hybrid grids*, which means combined structured-unstructured grids (e.g., [17–19]).

The grid generation has to be done in a way that preserves the conservation properties of the governing equations, namely:

- the physical domain has to be completely covered by the grid,
- there may be no free space left between the elements,
- the elements must not overlap.

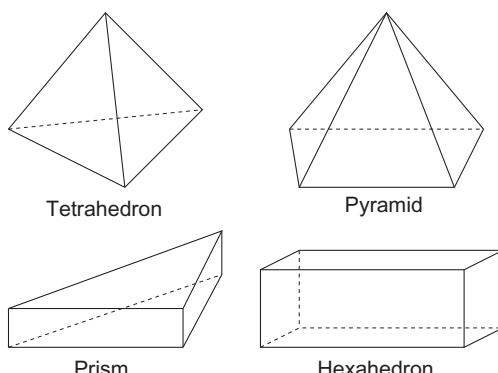


Figure 5.1 Elements used for the generation of 3-D unstructured grids.

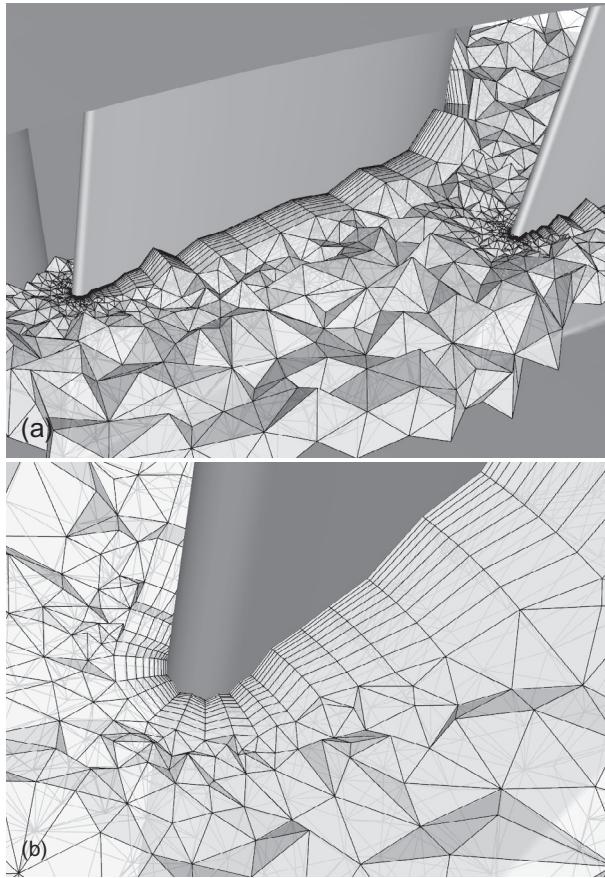


Figure 5.2 Rough cut through a 3-D unstructured mixed grid around a turbine guide vane. Grid was generated using CENTAUR™ [20, 21]. Note the layers of quadrilateral faces around the surface which is due to the prisms.

In addition to fulfilling the above requirements, the grid should be smooth, that is, there should be no large differences in the volumes or in the stretching ratio of adjacent grid cells. Furthermore, the elements should be as regular as possible, it is especially important to avoid degenerated grid elements such as slivers (see Fig. 11.24). Otherwise, the numerical errors could spoil the solution accuracy completely [22, 23]. Also, such elements will cause serious convergence problems.

Based on the grid, suitable control volumes are defined in order to evaluate the integrals of the convective and viscous fluxes as well as of the source term. For simplicity, let us assume that a particular control volume does not change in time (otherwise see Section A.5). Then, the time derivative of the conservative variables \vec{W} can be cast in the form

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega = \Omega \frac{\partial \vec{W}}{\partial t}.$$

Herewith, Eq. (2.19) becomes

$$\frac{\partial \vec{W}}{\partial t} = -\frac{1}{\Omega} \left[\oint_{\partial\Omega} (\vec{F}_c - \vec{F}_v) dS - \int_{\Omega} \vec{Q} d\Omega \right]. \quad (5.1)$$

The surface integral on the right-hand side of Eq. (5.1) is approximated by a sum of the fluxes crossing the faces of the control volume. This approximation is called *spatial discretization*. It is usually supposed that the flux is constant along the individual face and that it is evaluated at the midpoint of the face. This treatment is sufficient for a second-order accurate scheme. The source term is generally assumed to be constant inside the control volume. However, in cases where the source term becomes dominant, it is advisable to evaluate \vec{Q} as the weighted sum of values from the neighboring control volumes (see [24] and the references cited therein). If we consider a particular volume Ω_I , we obtain from Eq. (5.1)

$$\frac{d \vec{W}_I}{dt} = -\frac{1}{\Omega_I} \left[\sum_{m=1}^{N_F} (\vec{F}_c - \vec{F}_v)_m \Delta S_m - (\vec{Q}\Omega)_I \right]. \quad (5.2)$$

In the above expression, the index I in capital letters refers to the control volume, since it does not necessarily coincide with the grid, as we shall see later. Furthermore, N_F denotes the number of the faces of the control volume Ω_I , and the variable ΔS_m stands for the area of the face m , respectively. The number of faces N_F depends of course on the cell-type but also on the type of the control volume. In general, the number of faces changes between the control volumes as well, which is one of the main differences in comparison to structured grids. However, numerical procedures and data structures were developed which avoid the a priori knowledge of N_F . We shall return to this point in later sections.

The term in square brackets on the right-hand side of Eq. (5.2) is usually denoted as the *residual*. Thus, we may abbreviate Eq. (5.2) as

$$\frac{d \vec{W}_I}{dt} = -\frac{1}{\Omega_I} \vec{R}_I. \quad (5.3)$$

Writing down the relationship in Eq. (5.3) for all control volumes Ω_I , we obtain a system of ordinary differential equations of first order. The equations are hyperbolic in time, that means we have to advance them in time starting from a known initial solution. We have also to provide appropriate boundary conditions for the viscous and the inviscid fluxes, as they are described in Chapter 8.

When numerically solving the system of discretized governing equations (5.3), the first question is how to define the control volumes and where to locate the flow variables

with respect to the grid points. In the framework of finite-volume schemes, three basic strategies can be pursued:

- *Cell-centered* scheme [2, 16, 25, 26] —control volumes are identical with the grid cells and the flow variables are associated with their centroids (Fig. 5.6).
- *Cell-vertex* scheme with *overlapping* control volumes [27, 28] —flow quantities are assigned to the grid vertex and the control volumes are defined as the union of all grid cells having the respective node in common. This means that the control volumes associated with two neighboring vertexes overlap each other.
- Cell-vertex scheme with *median-dual* control volumes [13, 15, 29–33] —flow variables are again stored at the grid vertexes, but the control volumes are now created by connecting the centroids of the surrounding elements, face-centroids and edge-midpoints (Figs. 5.7 and 5.8). In this way, the grid points are encapsulated by their corresponding control volumes—representing a *dual grid*—which do not overlap.

Because the cell-vertex scheme with overlapping control volumes is no longer used, we shall concentrate here on the cell-centered and on the median-dual scheme. Both methodologies will be discussed in detail in [Section 5.2](#).

It is important to notice that in our case **all** flow variables, that is, the conservative variables (ρ , ρu , ρv , ρw , and ρE) and the dependent variables (p , T , c , etc.), are associated with the **same** location—with the cell center or with the grid point. This approach is known as the *co-located* grid scheme. By contrast, many older (structured) pressure-based methods (cf. [Section 3.1](#)) use the so-called *staggered* grid scheme, where the pressure and the velocity components are stored at different locations in order to suppress oscillations of the solution which arise from central spatial differences.

Many choices exist with respect to the evaluation of the convective fluxes. The basic problem is that we have to know their values at all N_F faces of a control volume, but the flow variables are not directly available there. This means, we have to interpolate either the fluxes or the flow variables to the faces of the control volume. The interpolation of flow variables is known as *reconstruction* of the solution from values inside the control volumes (see [Section 5.3.3](#)). In principle, the interpolation can be conducted in one of the following two ways:

- by arithmetic averaging like in *central* discretization schemes;
- by some biased interpolation like in *upwind* discretization schemes, which take care of the characteristics of the flow equations.

Besides the description, we shall treat aspects such as accuracy, range of applicability and numerical effort of the most widely used discretization schemes for the convective fluxes in [Section 5.3](#).

A common methodology applied for the evaluation of the viscous fluxes at a face of the control volume is based on arithmetic averaging of the flow quantities. Calculation of the velocity and the temperature gradients in Eqs. (2.15) and (2.24) is more difficult,

particularly in the case of mixed-element grids. A complete procedure will be presented in [Section 5.4](#).

5.1 GEOMETRICAL QUANTITIES OF A CONTROL VOLUME

Before we start to discuss the discretization methodologies applied to the convective and viscous fluxes, it is important to consider the evaluation of geometrical quantities of the control volume Ω_I —its volume, the unit normal vector \vec{n}_m (defined as outward facing) and the area ΔS_m of a face m , as well as the centroid of an element. The normal vector and the face area are also denoted as the *metrics* of the control volume. In the following sections, we shall consider the 2-D and the 3-D case separately.

5.1.1 Two-dimensional case

Generally, we think of the flow in a plane as being a special case of a 3-D problem, where the solution is symmetric with respect to one coordinate direction (e.g., to the z -direction). Because of the symmetry and in order to obtain correct physical units for volume, pressure, etc., we set the depth of all grid cells and control volumes equal to a constant value b . The volume of a control volume results then in 2D from the product of its area with the depth b . Since the depth b is arbitrary, we may set $b = 1$ for convenience. In the following discussion, we restrict ourselves to triangular and quadrilateral elements. Even though the control volume of a median-dual scheme can have a rather complex shape, it can always be decomposed into triangles and/or quadrilaterals.

Triangular element

The area of a general triangle can be the most conveniently and exactly calculated by the formula of Gauss. Thus, using a node numbering in accordance with [Fig. 5.3a](#), the volume results from

$$\begin{aligned}\Omega = \frac{b}{2} & [(x_1 - x_2)(y_1 + y_2) \\ & + (x_2 - x_3)(y_2 + y_3) \\ & + (x_3 - x_1)(y_3 + y_1)].\end{aligned}\tag{5.4}$$

The nodes have to be numbered in the anti-clockwise direction in order to obtain a positive value for the volume.

Quadrilateral element

The area of a general quadrilateral can be exactly calculated by Gauss' formula, which leads, after some algebra, to the expression

$$\Omega = \frac{b}{2} [(x_1 - x_3)(y_2 - y_4) + (x_4 - x_2)(y_1 - y_3)],\tag{5.5}$$

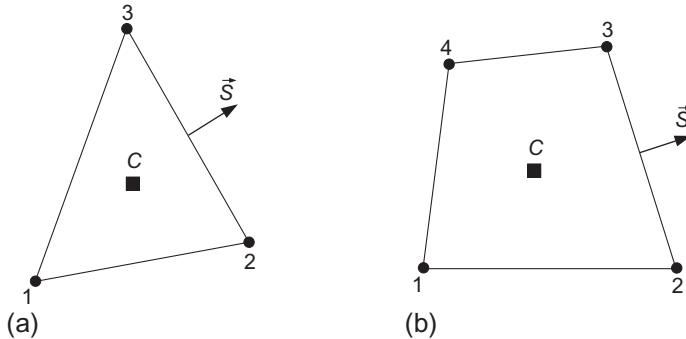


Figure 5.3 Numbering of nodes and face vector of: (a) triangular element, (b) quadrilateral element. C denotes the center of the element.

where the nodes are numbered according to Fig. 5.3b in the anti-clockwise direction. In the above, we assumed that the control volume is located in the x - y -plane and that the z -coordinate represents the symmetry axis.

The edges of a control volume are given by straight lines in 2D, therefore the unit normal vector is constant along them. When we integrate the fluxes according to the approximation of Eq. (5.2), we have to evaluate the product of the area of a face ΔS and the corresponding unit normal vector \vec{n} which is the *face vector* \vec{S} . Considering Fig. 5.3, the outward-pointing face vector, for example, at the side 2-3 is given by

$$\vec{S}_{23} = \vec{n}_{23} \Delta S_{23} = b \begin{bmatrix} y_3 - y_2 \\ x_2 - x_3 \end{bmatrix}. \quad (5.6)$$

Because of the symmetry, the z -component of the face vectors (and of the unit normal vector) is zero. It is therefore omitted in Eq. (5.6). The unit normal vector can be obtained from Eq. (5.6) with

$$\Delta S = | \vec{S} | = \sqrt{S_x^2 + S_y^2}, \quad (5.7)$$

where S_x and S_y denote the Cartesian components of the face vector.

Element center

The center of the triangle from Fig. 5.3a is defined as

$$\vec{r}_c = \frac{1}{3} (\vec{r}_1 + \vec{r}_2 + \vec{r}_3) \quad (5.8)$$

with $\vec{r}_{1/2/3}$ representing the Cartesian coordinates of the nodes. The center of a quadrilateral element can be computed by the formula given in Ref. [34]. For this purpose, the quadrilateral is decomposed into two triangles that share two points. With the node numbering according to Fig. 5.3b, and 1 and 3 being the common nodes, the relation reads

$$\vec{r}_c = \Omega_{123} \vec{r}_{c,123} + \Omega_{134} \vec{r}_{c,134} \Omega_{123} + \Omega_{134}. \quad (5.9)$$

The volumes of the two triangles Ω_{123} and Ω_{134} are evaluated using Eq. (5.4), and their centroids \vec{r}_c are obtained from Eq. (5.8). Formula for a general polygon can be found in Ref. [35].

5.1.2 Three-dimensional case

As opposed to the previous 2-D case, the computation of face vectors and volumes poses in 3D some problems for elements or control volumes with quadrilateral faces. The main reason for this is that, in general, the four vertexes of a quadrilateral face of a control volume may not lie in a plane. Then, the normal vector is no longer constant on such face (see Fig. 4.2). In order to overcome this difficulty, we could decompose each quadrilateral face into two or even more triangles. However, the gain in accuracy is hardly noticeable for a second-order scheme on a smooth grid. The additional effort can only be justified—and in fact it becomes necessary—for a third- and higher order spatial discretizations. Therefore, we shall apply a simplified treatment of the quadrilateral faces in the following considerations, which is based on an averaged normal vector.

Triangular face

The face vector \vec{S} can be exactly computed for a triangular face using Gauss' formula. Defining the nodes according to Fig. 5.4a, we obtain for the edge differences of the triangle 1-2-3

$$\begin{aligned}\Delta xy_A &= (x_1 - x_2)(y_1 + y_2), \\ \Delta yz_A &= (y_1 - y_2)(z_1 + z_2), \\ \Delta xy_B &= (x_2 - x_3)(y_2 + y_3), \\ \Delta yz_B &= (y_2 - y_3)(z_2 + z_3), \\ \Delta xy_C &= (x_3 - x_1)(y_3 + y_1), \\ \Delta yz_C &= (y_3 - y_1)(z_3 + z_1), \\ \Delta zx_A &= (z_1 - z_2)(x_1 + x_2), \\ \Delta zx_B &= (z_2 - z_3)(x_2 + x_3), \\ \Delta zx_C &= (z_3 - z_1)(x_3 + x_1).\end{aligned} \quad (5.10)$$

The outward pointing face vector $\vec{S} = \vec{n} \Delta S$ results then from

$$\vec{S} = \frac{1}{2} \begin{bmatrix} \Delta yz_A + \Delta yz_B + \Delta yz_C \\ \Delta zx_A + \Delta zx_B + \Delta zx_C \\ \Delta xy_A + \Delta xy_B + \Delta xy_C \end{bmatrix}. \quad (5.11)$$

Quadrilateral face

The averaged face vector \vec{S} of a quadrilateral face, like that rendered in Fig. 5.4b, is most conveniently computed using the same Gauss' formula as employed in 2-D for the area

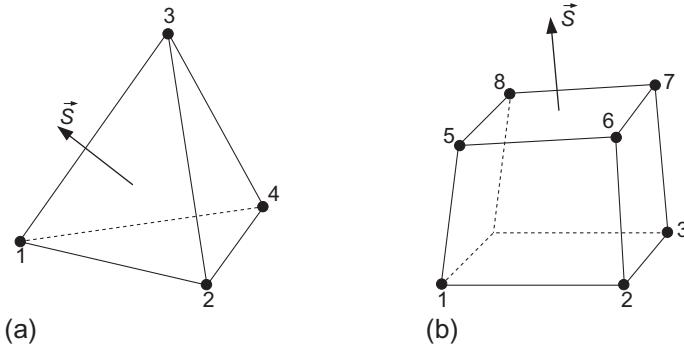


Figure 5.4 Numbering of nodes and face vector of: (a) tetrahedral element and (b) hexahedral element.

of a quadrilateral. Thus, for the face given by the nodes 5, 6, 7 and 8 in Fig. 5.4b, we first define the differences

$$\begin{aligned}\Delta x_A &= x_8 - x_6, & \Delta x_B &= x_7 - x_5, \\ \Delta y_A &= y_8 - y_6, & \Delta y_B &= y_7 - y_5, \\ \Delta z_A &= z_8 - z_6, & \Delta z_B &= z_7 - z_5.\end{aligned}\quad (5.12)$$

Then, we obtain the outward pointing face vector $\vec{S} = \vec{n}\Delta S$ from the relation

$$\vec{S} = \frac{1}{2} \begin{bmatrix} \Delta z_A \Delta y_B - \Delta y_A \Delta z_B \\ \Delta x_A \Delta z_B - \Delta z_A \Delta x_B \\ \Delta y_A \Delta x_B - \Delta x_A \Delta y_B \end{bmatrix}. \quad (5.13)$$

The approximation becomes exact when the face approaches a parallelogram, that is, when the vertexes of the face lie all in one plane.

The unit normal vector is obtained in both cases from $\vec{n} = \vec{S}/\Delta S$ with

$$\Delta S = \sqrt{S_x^2 + S_y^2 + S_z^2}, \quad (5.14)$$

where S_x , S_y , and S_z denote the Cartesian components of the face vector given by Eq. (5.11) or (5.13), respectively.

Volume

As we already stated in the case of 3-D structured finite-volume schemes, a very convenient approach for the computation of volumes is based on the divergence theorem [36]. The discussion in Section 4.1.2 led finally to the expression

$$\Omega = \frac{1}{3} \sum_{m=1}^{N_F} (\vec{r}_c \cdot \vec{S})_m \quad (5.15)$$

for the volume, where N_F denotes the number of the faces of the control volume, $(\vec{r}_c)_m$ the center of the face m of the control volume, and \vec{S}_m the face vector (outward directed) of the face m , respectively. The formula (5.15) is directly applicable on unstructured grids. It is exact for a volume with triangular faces, or a volume with planar quadrilateral faces.

Cell centroid

The previously mentioned median-dual type of control volume requires for its construction the knowledge of the centroid of the grid cell. The centroid of a general volume is defined as

$$\vec{r}_c = \frac{1}{\Omega} \int_{\Omega} \vec{r} d\Omega. \quad (5.16)$$

According to the derivation in Ref. [34], the relation in Eq. (5.16) can be discretized as

$$\vec{r}_c = \frac{3 \sum_{m=1}^{N_F} (\vec{r}_c \cdot \vec{n})_m (\vec{r}_c)_m \Delta S_m}{4 \sum_{m=1}^{N_F} (\vec{r}_c \cdot \vec{n})_m \Delta S_m}, \quad (5.17)$$

where the center of a face m , that is, $(\vec{r}_c)_m$ is obtained from Eq. (5.8) for a triangular face, or from Eq. (5.9) for a quadrilateral face. As we can note, the denominator in Eq. (5.17) is identical to 12Ω from Eq. (5.15). A similar formula for polyhedrons with triangular faces is provided in Ref. [35].

5.2 GENERAL DISCRETIZATION METHODOLOGIES

We already mentioned at the beginning of this chapter that there are two most popular approaches for the definition of the control volume and for the location of the flow variables. These are the cell-centered scheme and the median-dual scheme. We shall present both in more detail in this section.

However, before we start, let us briefly discuss about the basic data structure which is needed for an unstructured flow solver. In fact, a flexible but in terms of memory and operation count efficient data structure is the crucial point of any unstructured scheme. One could say that the structure which is missing in the grid has to be provided inside the flow solver. At least the following data is required:

- coordinates of the grid nodes (vertexes),
- pointers from elements to grid nodes,
- pointers from faces of elements located on a boundary to grid nodes.

Further data structures, which are required by the discretization schemes, can be generated from this information. In order to illustrate how the above data could possibly be stored, let us consider, for example, the tetrahedron in Fig. 5.4a. If we further assume that the face 1-2-4 is on a boundary (wall, inlet, far-field, etc.), we could employ the format:

```

# nodes (x, y, z):
P1.x  P1.y  P1.z
P2.x  P2.y  P2.z
P3.x  P3.y  P3.z
P4.x  P4.y  P4.z
...
# tetrahedra:
...
P1  P2  P3  P4
...
# boundaries:
...
type  P1  P4  P2
...

```

A similar format is also utilized by the 2-D unstructured flow solver provided with the accompanying source codes.

It is important to bear in mind the following two points related to the boundaries of the computational domain. First, it is more convenient to store boundary faces than just the nodes. This can be understood by considering the situation depicted in Fig. 5.5. The problem is that node P_1 is shared by three, nodes P_2 and P_3 by two boundaries of possibly physically different types. Therefore, it can become very cumbersome to apply the correct boundary conditions. On the contrary, a face can belong to only **one** boundary, like $P_1-P_2-P_4$ to boundary 1.

The second point concerns the numbering of the nodes of the boundary faces. This has to be done in a consistent way—for example, anti-clockwise when viewed from outside the flow domain—in order to have all face vectors (Eqs. (5.6) and (5.11) or (5.13)) either pointing outward or inward. This is important not only for flux integration, but also for grid generation.

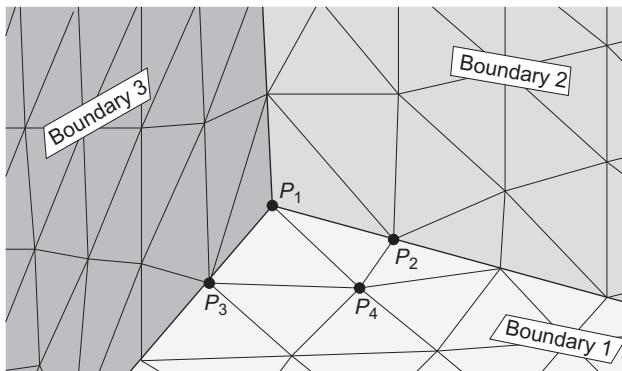


Figure 5.5 Three boundaries which meet at one corner—ambiguity of grid points with respect to boundary type.

5.2.1 Cell-centered scheme

We speak of a cell-centered scheme if the control volumes are identical with the grid cells and if the flow variables are associated with their centroids, as it is sketched in Fig. 5.6. When we evaluate the discretized flow equations (5.2), we have to supply the convective and the viscous fluxes at the midpoints of the faces of the control volume, which is sufficient for a second-order accurate discretization on smooth grids (averaged normal vector is employed for quadrilateral faces). The fluxes can be approximated in one of the following three ways:

1. by the *average of fluxes* computed from values at the centroids of the grid cells to the left and to the right of the cell face, but using the same unit normal vector (generally applied only to the convective fluxes);
2. by using an *average of variables* associated with the centroids of the grid cells adjacent to the left and to the right side of the cell face;
3. by computing the fluxes from flow quantities *reconstructed* separately on both sides of the cell face from values in the surrounding cells (employed only for the convective fluxes).

Thus, considering for example the cell face with the unit normal vector \vec{n}_{01} in Fig. 5.6, the first approach—average of fluxes—reads in 2D

$$(\vec{F}_c \Delta S)_{01} \approx \frac{1}{2} [\vec{F}_c(\vec{W}_0, \vec{n}_{01}) + \vec{F}_c(\vec{W}_1, \vec{n}_{01})] \Delta S_{01} \quad (5.18)$$

with the face area ΔS_{01} computed from Eqs. (5.6) and (5.7).

The second approach—average of variables—can be formulated as follows

$$(\vec{F} \Delta S)_{01} \approx \vec{F}(\vec{W}_{01}, \vec{n}_{01}) \Delta S_{01}, \quad (5.19)$$

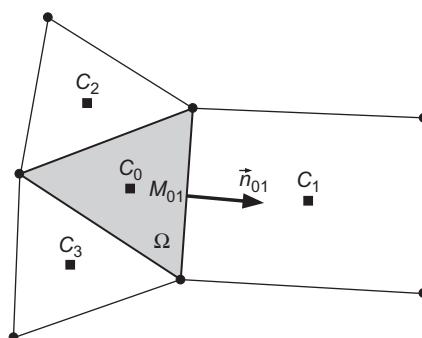


Figure 5.6 Control volume of a cell-centered scheme (in 2D). Grid nodes are represented by circles, cell centers by rectangles (C).

where the conservative/dependent variables at the face with the unit normal vector \vec{n}_{01} are defined as the arithmetic average of values at the two adjacent cells. Thus,

$$\vec{W}_{01} = \frac{1}{2} (\vec{W}_0 + \vec{W}_1). \quad (5.20)$$

The flux vector \vec{F} in Eq. (5.19) represents either the convective or the viscous fluxes.

The third methodology starts with an interpolation of flow quantities (usually velocity components, pressure, density and total enthalpy) separately to both sides of the cell face. The reconstructed quantities—termed the *left* and the *right state* (see Section 5.3.3)—differ in general. The fluxes through the cell face are then evaluated from the difference of the left and right state using an appropriate non-linear function. Hence,

$$(\vec{F}_c \Delta S)_{01} \approx f_{\text{Flux}} (\vec{U}_L, \vec{U}_R, \Delta S_{01}), \quad (5.21)$$

where

$$\begin{aligned} \vec{U}_L &= f_{\text{Rec}} (\dots, \vec{U}_2, \vec{U}_0, \dots) \\ \vec{U}_R &= f_{\text{Rec}} (\dots, \vec{U}_1, \vec{U}_0, \dots) \end{aligned} \quad (5.22)$$

represent the reconstructed states.

Of course, similar relations hold for the other control volume faces as well. The above approximations can be employed in the same way in 3D. The face vector \vec{S} is then evaluated using the formulas (5.11) or (5.13), respectively.

As we already stated in the introduction to this section, the basic data structure which describes the elements has to be extended in an appropriate way to support the discretization methodology. It is obvious from the previous discussion that numerical operations are carried out using mainly the faces of the elements (of the control volumes) together with values at the centers of the adjacent cells. It is therefore quite natural to employ a face-based data structure for the spatial discretization. Such data structure stores for each particular face in the grid (see Fig. 5.6):

- pointers to the two cells which share the respective face—this allows us to access the flow variables associated with the two cells (C_0, C_1);
- the face vector ($\vec{S}_{01} = \vec{n}_{01} \Delta S_{01}$)—must point consistently either outwards or inwards;
- two vectors from the centroid each cell to the midpoint of the face M_{01} , that is, $(C_0-M_{01}), (C_1-M_{01})$ —required for an accurate interpolation of flow variables to the face. This is not necessary for purely tetrahedral grids, where a simple extrapolation formula can be used [2, 26] (cf. Eq. (5.44)).

Hence, integration of the fluxes (e.g., according to Eq. (5.19)) could be implemented as a loop over **all** (i.e., internal and boundary) faces contained in the grid:

```

DO face = 1, nfaces
  I = pointer_to_left_cell(face )
  J = pointer_to_right_cell(face )
  ( $\vec{F} \Delta S$ )IJ ≈  $\vec{F}(\vec{W}_{IJ}, \vec{n}_I) \Delta S_{IJ}$ 
   $\vec{R}_I = \vec{R}_I + (\vec{F} \Delta S)_I$ 
   $\vec{R}_J = \vec{R}_J - (\vec{F} \Delta S)_J$ 
ENDDO

```

After the loop is completed and the source term $\vec{Q}_I \Omega_I$ is added, we obtain the final residuals (\vec{R}) in all cells. A less efficient approach would be to loop over elements because the face vectors would have to be stored twice and the fluxes would be computed twice (with the exception of the boundaries). Furthermore, because we use exactly the same face vector \vec{S}_I in order to evaluate the partial fluxes into the volumes Ω_I and Ω_J , the conservation properties of the governing equations are secured automatically.

5.2.2 Median-dual cell-vertex scheme

Within the cell-vertex scheme, the flow variables are associated with the grid nodes (vertices). Median-dual control volumes are formed by connecting the centroids, face- and edge-midpoints of all cells sharing the particular node. This is depicted in Fig. 5.7a for a tetrahedron and in Fig. 5.7b for a hexahedron. The definition of a median-dual control volume results in a polyhedral hull around each grid node, as it is sketched in Fig. 5.8 for a 2-D mixed grid. This polyhedron can be viewed as a *dual* grid—hence the name of the scheme. It is interesting to note that the median-dual finite-volume discretization is equivalent to the Galerkin finite-element scheme with linear elements (see, e.g., [37]).

In order to evaluate the discretized flow equation (5.2), we have to integrate the convective and viscous fluxes over the surface of the control volume. Hence,

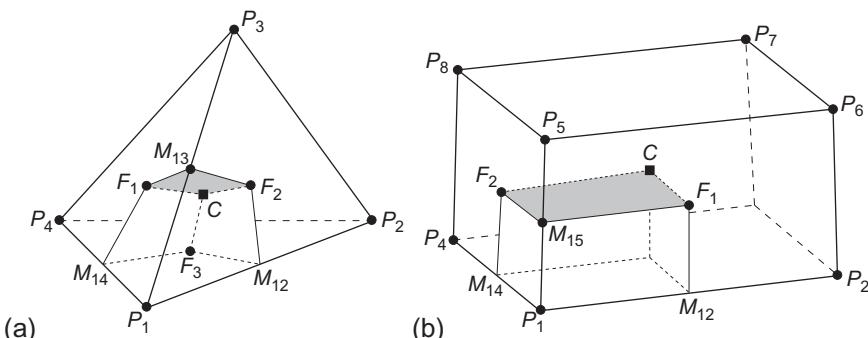


Figure 5.7 Partial control volume and faces (shaded) of a median-dual scheme for tetrahedron (a) and hexahedron (b). P denotes grid nodes, C cell-centroids (Eq. (5.17)), F face-centroids (Eq. (5.8) or (5.9)), and M stands for edge-midpoints. Shaded area represents one part of the control volume face assigned to edge P_1-P_3 or P_1-P_5 , respectively.

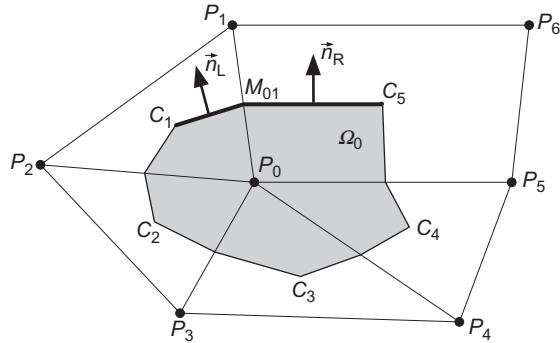


Figure 5.8 Control volume of a median-dual scheme (in 2D). C_1, C_2 , etc. denote cell centers; P_1, P_2 , etc. represent grid nodes. Face area associated with the edge P_0-P_1 is rendered by a bold line.

we would have to compute the fluxes for each partial face (e.g., $F_1-M_{13}-F_2-C$ in Fig. 5.7a) separately. However, this is only required for a third- or higher-order accurate discretizations [38, 39]. In the case of a second-order scheme, which is most frequently employed, we may assume the flow variables to be constant for all faces grouped around a particular edge. The fluxes are then evaluated at the midpoint of the edge using the variables and the gradients from both nodes. This approach allows us to define a mean unit normal vector and a total face area associated with each edge. Thus referring to Fig. 5.8, the mean normal vector, for example, for the edge P_0-P_1 becomes

$$\vec{n}_{01} = \vec{n}_L + \vec{n}_R, \quad (5.23)$$

and the total face area is given by: $\Delta S_{01} = \Delta S_L + \Delta S_R$. The same applies also in 3D, where the mean normal vector results from a sum over all partial faces having the particular edge-midpoint in common, as it is rendered in Fig. 5.9. The face vector ($\vec{S} = \vec{n} \Delta S$) is computed in 2D from Eq. (5.6). In 3D, where the partial faces are always quadrilaterals, we can either divide them into triangles and use Eq. (5.11), or we can employ a simplified treatment due to Eq. (5.13), which is sufficient for smooth grids. The element centroids and the face centers are obtained by the formulas (5.17), (5.8), or (5.9), respectively.

The fluxes can then be evaluated according to one of the three following methodologies:

1. by the *average of fluxes* computed from values at both nodes of an edge, but using the same mean unit normal vector (generally applied only to the convective fluxes);
2. by using an *average of variables* stored at the two nodes of an edge;
3. by computing the fluxes from flow quantities *reconstructed* separately on both sides of the face of the control volume from values at the surrounding nodes (employed only for the convective fluxes).

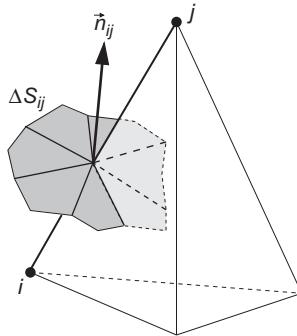


Figure 5.9 Total face area and mean unit normal vector associated with the edge ij of the 3-D median-dual cell-vertex scheme.

The computation of fluxes follows formally the same approaches as for the cell-centered scheme. Thus, the formulas (5.18)–(5.22) are applicable also in the case of the median-dual scheme. If we utilize the above approach which associates each edge with a mean unit normal, the most efficient methodology is to employ an edge-based data structure for the spatial discretization. The edge-based data structure stores for each particular edge in the grid (cf. Fig. 5.9):

- pointers to the two nodes which define the edge—this allows us to access the flow variables associated with the two control volumes Ω_i and Ω_j ;
- the face vector ($\vec{S}_{ij} = \vec{n}_{ij} \Delta S_{ij}$)—must point consistently either outwards or inwards;
- the edge vector from node i to node j —required for the interpolation of flow variables to the face (solution reconstruction). Alternatively, the edge vector can be computed on the fly from coordinates of the nodes. This is not required for the standard central scheme with artificial dissipation (Section 5.3.1).

With this, the integration of the fluxes (e.g., according to Eq. (5.19)) would be implemented as a loop over **all** edges in the grid:

```

DO edge = 1, nedges
    i = pointer_to_left_node( edge )
    j = pointer_to_right_node( edge )
    ( $\vec{F} \Delta S$ )ij  $\approx \vec{F}(\vec{W}_{ij}, \vec{n}_{ij}) \Delta S_{ij}$ 
     $\vec{R}_i = \vec{R}_i + (\vec{F} \Delta S)_{ij}$ 
     $\vec{R}_j = \vec{R}_j - (\vec{F} \Delta S)_{ij}$ 
ENDDO

```

After the loop is completed and the source term $\vec{Q}_i \Omega_i$ is added, we obtain the final residuals (\vec{R}) in all nodes. This approach is significantly more efficient than summing up the fluxes over each control volume separately, because we store each mean face vector only once and we also visit each edge only once instead of twice. Furthermore, since

we use exactly the same mean face vector \vec{S}_{ij} in order to evaluate the partial fluxes into the volumes Ω_i and Ω_j , the mass, momentum and energy remain exactly conserved.

5.2.3 Cell-centered versus median-dual scheme

The relative advantages and disadvantages of the cell-centered and the median-dual schemes are the subject of controversial debates. The main reason is the lack of fair comparisons of the two methodologies with respect to accuracy, computational time, and memory for realistic configurations. Our intention here is to collect the most important arguments for and against each of the approaches regarding:

- accuracy,
- computational work,
- memory requirements,
- flexibility.

This should lead to a greater understanding of the problems inherent to each scheme, and might be of help in selecting the most suitable scheme for the intended range of applications.

Accuracy

A cell-centered scheme on a triangular/tetrahedral grid leads to about twice/six times as many control volumes and hence degrees of freedom as a median-dual scheme [37]. On typical mixed grids, which consist of tetrahedra and prisms, a cell-centered scheme gives roughly three times more unknowns than a median-dual scheme. This suggests that cell-centered schemes are more accurate than cell-vertex discretizations on an identical grid. However, the residual of a cell-centered scheme results from a much smaller number of fluxes as compared to a median-dual scheme (three versus approximately seven on a tetrahedral grid), which may impair the accuracy. In this respect, there is no clear evidence as to which scheme might be superior.

The median-dual scheme suffers from a particular problem on stretched triangular and tetrahedral grids. Consider, for example, Fig. 5.10, which shows a tessellation composed of right triangles, as it is often employed near solid walls for viscous flows. We

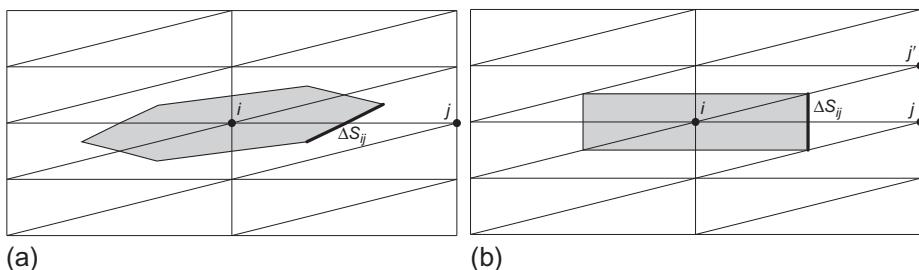


Figure 5.10 Comparison of median-dual (a) and containment-dual (b) control volumes for a stretched right-angle triangulation.

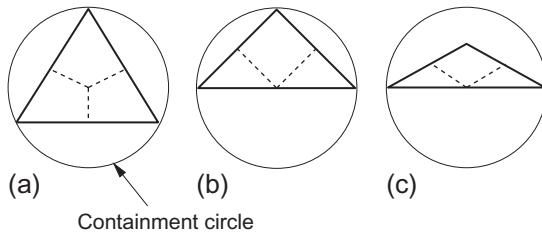


Figure 5.11 Part of containment dual (dashed line) in the case of acute (a) and obtuse (c) triangles [41]. The containment circle is the smallest circle which contains the triangle. For obtuse triangles, it is centered on the longest edge.

can see in Fig. 5.10a that the face ΔS_{ij} becomes highly skewed with respect to the edge ij . However, spatial discretization schemes mostly assume fluxes to be orthogonal to a face (especially Riemann solvers). Thus, an error is introduced which is particularly significant for a first-order scheme [39]. The situation can be improved using the so-called *containment-dual* control volume [40]. As depicted in Fig. 5.11, the containment-dual approach employs the centers of the minimum spanning circles/spheres instead of the cell centroids to define the faces. This leads to control volumes identical to those on quadrilateral grids (Fig. 5.10b). Notice that there is no face area associated with diagonal edges like ij' . An additional effort is required for pre-processing, but the solution accuracy can be improved noticeably [41]. Of course, another possibility is to directly employ quadrilateral or hexahedral cells within the boundary layers. Further discussion of grid-induced errors can be found in Ref. [22, 23].

Another problem inherent to the median-dual scheme is the discretization at boundaries of the physical domain. What happens is that there is only about one half of the control volume left at the boundary (cf. Fig. 4.6). The integration of fluxes around the faces results in a residual located **inside**—ideally at the **center**—of the control volume. However, the residual is associated with the **node** which resides directly on the boundary. This mismatch leads to increased discretization error in comparison to the cell-centered scheme, which is particularly undesirable on solid walls. The definition of the dual control volume causes also problems at sharp corners (like trailing edges), which show up as unphysical peaks in pressure or density. Further complications arise at periodic boundaries (see Section 8.8), where the fluxes from both parts of the control volume have to be summed up correctly.

The mismatch between the centroid of the control volume and the node where the residual is stored has also a further implication for the median-dual scheme. It arises as the mass matrix in the case of unsteady flows. We discussed this point already at the beginning of Section 3.2. The advantage of the cell-centered scheme is that the mass matrix can be eliminated from the equations, without compromising the solution accuracy. By contrast, the median-dual scheme requires a special treatment of the mass matrix [42, 43].

Computational work

In order to judge the computational effort required for both schemes, we have to consider primarily the integration of the fluxes. We know from the previous discussion that the cell-centered scheme uses a loop over cell faces whereas the median-dual scheme loops over the edges. Since the evaluation of the fluxes at an interface is quite similar for both schemes, the ratio of the number of cell faces to the number of edges gives the ratio of the computational work. Thus, on a tetrahedral grid, where the number of cell faces (if counted only once for every two cells) is approximately two times larger than the number of edges, the cell-centered scheme is computationally twice as much expensive as the median-dual scheme on an identical grid [37]. The cell-centered approach becomes, however, more competitive on mixed grids containing prismatic elements. Apart from boundary treatment, both methods are computationally equivalent on hexahedral grids, where the number of faces equals the number of edges.

Memory requirements

Considering the memory requirements, the cell-centered scheme has to store about six times more flow variables on tetrahedral and about three times more variables on standard mixed grids as compared to the median-dual scheme. Furthermore, as we could see, both schemes require the storage of two integers and three reals (pointers and face vector) per cell face or edge, respectively. Additionally, the cell-centered scheme has to keep two vectors to the face-midpoint—6 reals—per cell face in memory. On the contrary, the median-dual scheme can work with the node coordinates only, which leads to considerably fewer values. Thus, in summary, the cell-centered scheme needs, on average, more than twice as much computer memory as the median-dual method.

Grid generation/adaptation

Significant advantage of the cell-centered scheme appears in the case of non-conforming cell interfaces, like those at the letter “F” in Fig. 3.4. In contrast to the median-dual methodology, no special and expensive procedure is required for the computation of the fluxes at the interface. This allows for an increased flexibility during grid generation, as well as in cases of grid adaptation and motion. This advantage of the cell-centered scheme is also utilized in flow solvers which employ nested Cartesian grids.

5.3 DISCRETIZATION OF THE CONVECTIVE FLUXES

In the previous sections, we considered general issues of possible spatial discretization methodologies including the necessary data structures. In what follows, we shall learn more about the details, how the evaluation of the convective fluxes can be implemented.

As we could already see in Section 3.1.5, within the framework of the finite-volume approach, we have basically the choice between:

- central,
- flux–vector splitting,
- flux–difference splitting,
- total variation diminishing (TVD),
- fluctuation-splitting

schemes. First of all, we shall present the central discretization on unstructured grids at some length, since it differs considerably from that on structured grids. On the contrary, the basics of the upwind schemes are identical on structured and unstructured grids. Hence, the details can be found in Sections 4.3.2–4.3.4. However, what is new on unstructured grids is the solution *reconstruction*, which is required in order to obtain the values of the flow variables at a face of the control volume. Therefore, we shall discuss the common approaches in some detail in [Section 5.3.3](#). Because of space limitations, we will not treat the fluctuation-splitting approach here. The reader is referred to Section 3.1.5 for the bibliography related to fluctuation-splitting schemes.

5.3.1 Central scheme with artificial dissipation

The basic idea of the central scheme is to compute the convective fluxes at a face of the control volume from the arithmetic average of the conservative variables on both sides of the face according to Eq. (5.20). Since this would lead to odd–even decoupling of the solution (generation of two independent solutions of the discretized equations) and to wiggles at shocks, artificial dissipation has to be added for stability. The artificial dissipation is based on a blend of second- and fourth-order differences. The scheme was first implemented for the Euler equations on structured grids by Jameson et al. [44]. Because of the names of the authors, it is also abbreviated as the JST scheme.

The implementation of the JST scheme on unstructured grids utilizes the Laplacian operator for the second-order differences and the Laplacian of Laplacian for the fourth-order differences [27, 45]. In order to reduce the computational cost, pseudo-Laplacians are employed instead of true Laplacians. Using this simplification, a 2-D formulation was proposed first in [46] and then improved in [47]. Later on, the scheme was extended to 3D in [2]. It makes use of a distance-weighting procedure. In this way, the scheme leads to a vanishing pseudo-Laplacian for a linearly varying function on any grid. Applied to a general scalar quantity U in cell I , the pseudo-Laplacian assumes the form

$$L(U_I) = \sum_{J=1}^{N_A} \theta_{IJ} (U_J - U_I), \quad (5.24)$$

where N_A stands for the number of adjacent control volumes. The cell indexes have to be substituted by node indexes (i,j) in the case of the median-dual scheme. The sum in Eq. (5.24) is best evaluated using either a loop over faces (cell-centered scheme) or a

loop over edges (median-dual scheme) similar to the flux computation. The geometrical weights θ are defined as

$$\theta_{IJ} = 1 + \Delta\theta_{IJ} \quad (5.25)$$

and result from the solution of an optimization problem [2]. The optimization problem is solved by means of Lagrange multipliers. Herewith, the geometrical weights are obtained from the expression

$$\Delta\theta_{IJ} = \lambda_{x,I}(x_J - x_I) + \lambda_{y,I}(y_J - y_I) + \lambda_{z,I}(z_J - z_I), \quad (5.26)$$

where x, y, z are the Cartesian coordinates of the cell centroids (nodes in the case of the median-dual scheme). The Lagrange multipliers λ are computed for each cell (node) and follow from [2]

$$\begin{aligned} \lambda_x &= \frac{R_x a_{11} + R_y a_{12} + R_z a_{13}}{d}, \\ \lambda_y &= \frac{R_x a_{21} + R_y a_{22} + R_z a_{23}}{d}, \\ \lambda_z &= \frac{R_x a_{31} + R_y a_{32} + R_z a_{33}}{d} \end{aligned} \quad (5.27)$$

with the coefficients

$$\begin{aligned} a_{11} &= I_{yy}I_{zz} - I_{yz}^2, \\ a_{12} &= I_{xz}I_{yz} - I_{xy}I_{zz}, \\ a_{13} &= I_{xy}I_{yz} - I_{xz}I_{yy}, \\ a_{21} &= I_{xz}I_{yz} - I_{xy}I_{zz}, \\ a_{22} &= I_{xx}I_{zz} - I_{xz}^2, \\ a_{23} &= I_{xy}I_{xz} - I_{xx}I_{yz}, \\ a_{31} &= I_{xy}I_{yz} - I_{xz}I_{yy}, \\ a_{32} &= I_{xz}I_{xy} - I_{xx}I_{yz}, \\ a_{33} &= I_{xx}I_{yy} - I_{xy}^2, \\ d &= I_{xx}I_{yy}I_{zz} - I_{xx}I_{yz}^2 - I_{yy}I_{xz}^2 - I_{zz}I_{xy}^2 + 2I_{xy}I_{xz}I_{yz}. \end{aligned} \quad (5.28)$$

Written for a cell I , the first-order moments read

$$\begin{aligned} R_{x,I} &= \sum_{J=1}^{N_A} (x_J - x_I), \\ R_{y,I} &= \sum_{J=1}^{N_A} (y_J - y_I), \\ R_{z,I} &= \sum_{J=1}^{N_A} (z_J - z_I). \end{aligned} \quad (5.29)$$

Furthermore, the second-order moments are given by

$$\begin{aligned}
 I_{xx,I} &= \sum_{J=1}^{N_A} (x_J - x_I)^2, \\
 I_{yy,I} &= \sum_{J=1}^{N_A} (y_J - y_I)^2, \\
 I_{zz,I} &= \sum_{J=1}^{N_A} (z_J - z_I)^2, \\
 I_{xy,I} &= \sum_{J=1}^{N_A} (x_J - x_I)(y_J - y_I), \\
 I_{xz,I} &= \sum_{J=1}^{N_A} (x_J - x_I)(z_J - z_I), \\
 I_{yz,I} &= \sum_{J=1}^{N_A} (y_J - y_I)(z_J - z_I).
 \end{aligned} \tag{5.30}$$

The geometrical weights (5.25) can lead to a non-positive approximation of the Laplacian and hence to a loss of stability on severely distorted grids. Therefore, a clipping of the weights to the range $[0, 2]$ was suggested in [46]. However, this measure impairs the accuracy of the discretization. See also the discussion in Ref. [12] for further details.

The fourth-order differences are evaluated as the Laplacian of the Laplacian, that is, $L(U)$ is substituted for U in Eq. (5.24). Hence, the final form of the artificial dissipation term for a cell I is

$$\begin{aligned}
 \vec{D}_I &= \sum_{J=1}^{N_A} (\hat{\Lambda}_c)_{IJ} \epsilon_{IJ}^{(2)} \theta_{IJ} (\vec{W}_J - \vec{W}_I) \\
 &\quad - \sum_{J=1}^{N_A} (\hat{\Lambda}_c)_{IJ} \epsilon_{IJ}^{(4)} \theta_{IJ} [L(\vec{W}_J) - L(\vec{W}_I)].
 \end{aligned} \tag{5.31}$$

With the artificial dissipation term added, the system of equations in Eq. (5.2) becomes

$$\Omega_I \frac{d\vec{W}_I}{dt} = - \left[\sum_{m=1}^{N_F} (\vec{F}_c - \vec{F}_v)_m \Delta S_m \right] + \vec{D}_I + \vec{Q}_I \Omega_I, \tag{5.32}$$

where N_F denotes the number of the faces of the control volume (which may differ from the number of adjacent control volumes, for example, if a quadrilateral face is divided into two triangles).

The second- and the fourth-order terms in Eq. (5.31) are scaled by the spectral radius of the convective flux Jacobian. According to Ref. [28], the spectral radius for cell I can be evaluated as

$$(\hat{\Lambda}_c)_I = \sum_{m=1}^{N_F} (|V_m| + c_m) \Delta S_m, \quad (5.33)$$

where V_m represents the contravariant velocity (2.22) and c_m the speed of sound, respectively. Both quantities are computed from flow variables averaged at the face. The spectral radius at the face of the control volume is obtained from

$$(\hat{\Lambda}_c)_{IJ} = \frac{1}{2} \left[(\hat{\Lambda}_c)_I + (\hat{\Lambda}_c)_J \right]. \quad (5.34)$$

A pressure-based sensor is used to switch off the fourth-order differences at shocks and the second-order differences in smooth portions of the flow field. Herewith, the coefficients $\epsilon_{IJ}^{(2)}$ and $\epsilon_{IJ}^{(4)}$ in Eq. (5.31) are defined as

$$\begin{aligned} \epsilon_{IJ}^{(2)} &= k^{(2)} \max(\Upsilon_I, \Upsilon_J), \\ \epsilon_{IJ}^{(4)} &= \max \left[0, (k^{(4)} - \epsilon_{IJ}^{(2)}) \right] \end{aligned} \quad (5.35)$$

with the pressure sensor given by

$$\Upsilon_I = \frac{\left| \sum_{J=1}^{N_A} \theta_{IJ} (p_J - p_I) \right|}{\sum_{J=1}^{N_A} (p_J + p_I)}. \quad (5.36)$$

Typical values of the parameters are $k^{(2)} = 1/2$ and $1/128 \leq k^{(4)} \leq 1/64$.

As we already discussed in Section 4.3.1, the accuracy of the above scheme can be improved when we substitute a matrix for the spectral radius $(\hat{\Lambda}_c)_{IJ}$ in Eq. (5.31). The implementation of this so-called *matrix dissipation* scheme [48] on unstructured grids proceeds in the same way as on structured grids, with the scaling matrix defined as in Eq. (4.59). Application of the matrix dissipation scheme to 3-D mixed grids is discussed, for example, in Ref. [49].

It is important to note that for elements other than triangles/tetrahedra, the popular explicit Runge-Kutta type of temporal discretization experiences severe stability problems when it is coupled to the central scheme [50]. The reason is the representation of the fourth-order differences by the Laplacian of the Laplacian. A remedy is to employ a difference of the left and the right state (cf. Section 4.3) for the approximation of the fourth-order differences [50], that is,

$$\vec{D}_I = \sum_{J=1}^{N_A} (\hat{\Lambda}_c)_{IJ} \epsilon_{IJ}^{(2)} \theta_{IJ} (\vec{W}_J - \vec{W}_I) - \sum_{J=1}^{N_A} 4 (\hat{\Lambda}_c)_{IJ} \epsilon_{IJ}^{(4)} (\vec{W}_L - \vec{W}_R). \quad (5.37)$$

This approach leads on quadrilateral/hexahedral grids to the same stencil as the corresponding structured scheme. The left and right state are computed using, for example, the linear reconstruction described in [Section 5.3.3](#).

5.3.2 Upwind schemes

Upwind schemes seem to have gained, at least for the moment, much more popularity on unstructured grids than the above central scheme. In fact, the flux-difference splitting scheme of Roe [51] is the most widely employed approach on unstructured grids. It is the considerably more accurate resolution of boundary layers and the lower sensitivity to grid distortions in comparison to the central scheme, which explains the attractiveness of Roe's scheme. However, the price to be paid for the improved performance is the higher computational effort, which becomes quite significant if a limiter has to be used to suppress oscillations of the solution ([Section 5.3.5](#)).

Any of the upwind schemes presented in Section 4.3 for structured grids are applicable to unstructured grids without modifications to the basic methodology. Only the computation of the left and right state (Eq. (5.22)), which is denoted as solution reconstruction, as well as the evaluation of the limiting function require new formulations. For this reason, only the solution reconstruction and the limiters are discussed here. For details on the various upwind methods, the reader is referred to Sections 4.3.2–4.3.4. An example for the implementation of Roe's scheme on unstructured grids using the median-dual approach can be found, for example, in Ref. [33].

5.3.3 Solution reconstruction

As we saw in Sections 4.3.2–4.3.4, upwind schemes require flow states to be specified on the left and the right side of a control volume face. The same holds also for the modified artificial dissipation scheme in Eq. (5.37).

As a first approach, we can assume that the solution is constant inside each control volume. The left and right state are then simply the flow variables computed for the left and the right control volume. For example, in the case of the median-dual scheme ([Fig. 5.9](#)) we would set

$$\begin{aligned} U_L &= U_i, \\ U_R &= U_j \end{aligned} \tag{5.38}$$

with U representing some scalar flow variable. This leads to a spatial discretization which is only first-order accurate. For viscous flows, a first-order accurate solution is too diffusive and leads to an excessive growth of shear layers. Therefore, methods with higher spatial accuracy are a must for the computation of viscous flows.

We can achieve second- and higher-order spatial accuracy if we assume the solution changes within the control volume. For second-order accurate methods, which are the most commonly employed higher-order methods, the solution is assumed to vary in a

linear fashion over the control volume. In order to compute the left and right state, a reconstruction of the assumed solution variation becomes necessary. In what follows, we shall discuss the most popular approaches for the reconstruction of linear and quadratic variations. The interested reader is referred to [39] for a comparison of various linear reconstruction techniques.

Reconstruction based on MUSCL approach

One possibility to achieve second-order accuracy consists of the extension of the MUSCL approach [52] to unstructured grids. When applied to the median-dual scheme, the method generates for each edge ij two “phantom” nodes i' and j' [53–57]. These phantom nodes are located at the endpoints of the line obtained by extending the edge ij by its length in both directions as sketched in Fig. 5.12. After the solution is interpolated from the surrounding elements (gray colored in Fig. 5.12) to the phantom nodes, we can evaluate the left and right state using the MUSCL formulas Eq. (4.46). Hence,

$$\begin{aligned} U_R &= U_j - \frac{1}{4} [(1 + \hat{\kappa})\Delta_- + (1 - \hat{\kappa})\Delta_+] U_j, \\ U_L &= U_i + \frac{1}{4} [(1 + \hat{\kappa})\Delta_+ + (1 - \hat{\kappa})\Delta_-] U_i \end{aligned} \quad (5.39)$$

with forward (Δ_+) and the backward (Δ_-) difference operators defined as

$$\begin{aligned} \Delta_+ U_i &= U_j - U_i, & \Delta_- U_i &= U_i - U_{i'}, \\ \Delta_+ U_j &= U_{j'} - U_j, & \Delta_- U_j &= U_j - U_i. \end{aligned} \quad (5.40)$$

The MUSCL interpolation (5.39) has to be enhanced by a limiter function (according to Section 4.3.5) in the case of strong discontinuities.

A disadvantage of this methodology is the necessity to store for each edge the elements which contain the phantom nodes. A further conceptual disadvantage is that no unique gradient is reconstructed for a control volume. Furthermore, difficulties can arise at boundaries, where one of the phantom points lies outside the physical domain.

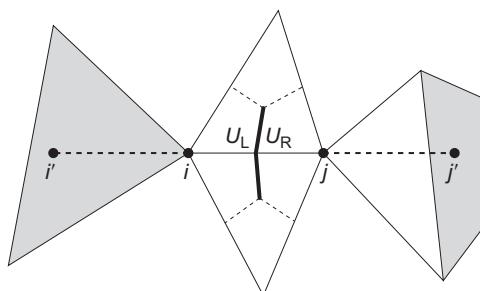


Figure 5.12 Evaluation of the left and right state based on interpolation from elements in the direction of an edge ij (median-dual scheme in 2D).

Piecewise linear reconstruction

Barth and Jespersen presented in [30] a reconstruction method, which is closely related to the finite-element schemes. Here, it is assumed that the solution is piecewise linearly distributed over the control volume. Then, we can find the left and right state for a cell-centered scheme from the relations

$$\begin{aligned} U_L &= U_I + \Psi_I (\nabla U_I \cdot \vec{r}_L), \\ U_R &= U_J + \Psi_J (\nabla U_J \cdot \vec{r}_R), \end{aligned} \quad (5.41)$$

where ∇U_I is the gradient of U ($= [\partial U / \partial x, \partial U / \partial y, \partial U / \partial z]^T$) at the cell center I and Ψ denotes a limiter function (cf. Section 5.3.5), respectively. The vectors \vec{r}_L and \vec{r}_R point from the cell-centroid to the face-midpoint, as indicated in Fig. 5.13a.

The same approach applies to the median-dual scheme [30], that is,

$$\begin{aligned} U_L &= U_i + \frac{1}{2} \Psi_i (\nabla U_i \cdot \vec{r}_{ij}), \\ U_R &= U_j - \frac{1}{2} \Psi_j (\nabla U_j \cdot \vec{r}_{ij}). \end{aligned} \quad (5.42)$$

According to Fig. 5.9 or 5.13b,

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i \quad (5.43)$$

represents the vector from node i to node j .

It can be easily seen that the method of Barth and Jespersen corresponds to a Taylor-series expansion around the neighboring centers/nodes of the face, where only the linear term is retained. The linear reconstruction is formally second-order accurate on regular grids [39]. The scheme reconstructs a linear function exactly on any grid, provided the gradient ∇U is evaluated without an error. The linear reconstruction is likely the most popular one among the reconstruction methods.

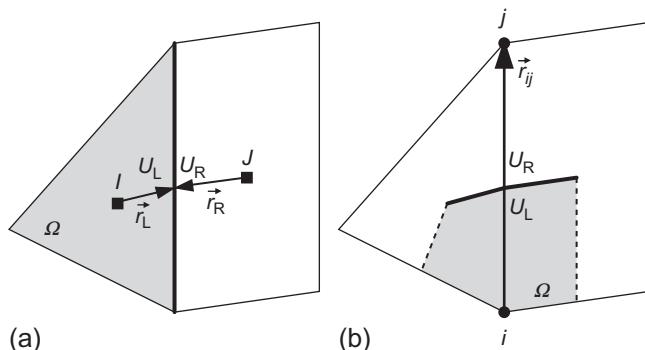


Figure 5.13 Linear reconstruction for the cell-centered (a) and the median-dual (b) scheme in 2D.

The above scheme requires the computation of gradients at cell centers or at nodes, respectively. This can be accomplished either by the Green-Gauss or the least-squares approach, which are presented below in [Section 5.3.4](#). Furthermore, the implementation of the limiter function on unstructured grids is described in detail in [Section 5.3.5](#).

Linear reconstruction based on nodal weighting procedure

It was demonstrated by Frink [26] that for the cell-centered scheme the linear reconstruction (5.41) does not require an explicit evaluation of the gradient on purely triangular or tetrahedral grids. The reason are two invariant geometric features of these elements. First, a line from a node through the cell-centroid will always intersect the midpoint of the opposing face. Second, the distance from the cell-centroid to the face-midpoint is one-fourth (one-third for a triangle) of the distance from the face-midpoint to the opposite node. Thus, the gradient at the cell center can be approximated by a simple finite difference [26]. For example, if we were to reconstruct the solution at the face-midpoint F_3 in [Fig. 5.7a](#), the formulas (5.41) would become

$$U_{L/R} = U_C + \frac{\Psi_C}{4} \left[\frac{1}{3}(U_1 + U_2 + U_4) - U_3 \right] \quad (5.44)$$

with U_C being the value at the cell-centroid, U_1 , U_2 , etc. denoting the nodal values, and finally Ψ standing for a limiter, respectively.

Two different ways were devised by Frink in order to determine the nodal values. The first approach is based on inverse distance weighting. Here, the contribution to a node from the surrounding cells is inverse proportional to the distance from the node to the cell-centroid [26, 58], that is,

$$U_i = \left(\sum_{J=1}^{N_A} \theta_{ij} U_J \right) \Bigg/ \left(\sum_{J=1}^{N_A} \theta_{ij} \right), \quad (5.45)$$

with the weights $\theta_{ij} = 1/r_{ij}$. The distance is computed from

$$r_{ij} = \sqrt{(x_J - x_i)^2 + (y_J - y_i)^2 + (z_J - z_i)^2}. \quad (5.46)$$

The subscripts J and i refer to the cell-centroid and to the node, respectively. The above methodology leads to a reconstruction which is less than second-order accurate. However, Frink pointed out that no limiter is needed at least for inviscid flows [26], which reduces the computational effort significantly.

The second approach is based on work of Holmes and Connell [46] and Rausch et al. [47] in 2D. It was later extended to 3D by Frink [2]. Here, the weights θ_{ij} in Eq. (5.45) are defined such that the nodal values are computed exactly if the variation is linear. This leads to the same constraints as for the computation of the pseudo Laplacian (5.24). Consequently, the weights are also the same and follow from Eqs. (5.25)–(5.30). The

coordinates x_I, y_I, z_I of the cell-centroids are just replaced by the node coordinates x_i, y_i, z_i . The scheme is formally second-order accurate because the nodal values are computed exactly for a linear function. In order to assure positivity on distorted grids, the weights have to be restricted to the range $[0, 2]$ as before [46]. Unfortunately, this reduces the accuracy of the reconstruction. Frink and Pirzadeh [4] also reported some anomalous behavior of the reconstruction for the Navier-Stokes equations.

Piecewise quadratic reconstruction

In order to achieve higher than second-order accuracy with a polynomial reconstruction, we have to keep further terms in the truncated Taylor-series expansion around the neighboring cell-centers/nodes of the face. Based on the work of Barth and Frederickson [59], Barth developed the concept of k -exact reconstruction scheme [60], that is, a reconstruction exact for a polynomial of degree k . The polynomial in Barth's method is defined in a way which guarantees the conservation of the mean, or in other words, the average of the reconstruction polynomial is equal to the mean solution in the control volume. This property assures the conservation of mass, momentum, and energy during the reconstruction. The method was implemented for $k = 3$ in a median-dual scheme. The coefficients of the polynomial were computed using a least-squares approach. Similar ideas were followed for the cell-centered scheme by Mitchell and Walters [61], and by Mitchell [62]. However, these methods require a prohibitively high numerical effort and a complex data structure which prevented their widespread use.

Delanaye and Essers [63] and Delanaye [64] developed a particular form of the quadratic reconstruction for cell-centered schemes, which is computationally more efficient than the method of Barth. The left and right state are approximated using Taylor series truncated after the quadratic term [63, 64]

$$\begin{aligned} U_L &= U_I + \Psi_{I,1} (\nabla U_I \cdot \vec{r}_L) + \frac{1}{2} \Psi_{I,2} (\vec{r}_L^T \bar{H}_I \vec{r}_L), \\ U_R &= U_J + \Psi_{J,1} (\nabla U_J \cdot \vec{r}_R) + \frac{1}{2} \Psi_{J,2} (\vec{r}_R^T \bar{H}_J \vec{r}_R). \end{aligned} \quad (5.47)$$

where \bar{H}_I denotes the Hessian matrix, that is,

$$\bar{H}_I = \begin{bmatrix} \partial_{xx}^2 U & \partial_{xy}^2 U & \partial_{xz}^2 U \\ \partial_{xy}^2 U & \partial_{yy}^2 U & \partial_{yz}^2 U \\ \partial_{xz}^2 U & \partial_{yz}^2 U & \partial_{zz}^2 U \end{bmatrix}_I, \quad (5.48)$$

evaluated at the cell-centroid I . The variables $\Psi_{I,1}$ and $\Psi_{I,2}$ represent two different limiter functions for the linear and the quadratic term [63], respectively. The quadratic reconstruction method is third-order accurate on regular grids and at least second-order accurate on arbitrary grids due to cancelation of error terms [64]. Necessary conditions for achieving these properties are, however, that the gradient ∇U in Eq. (5.47) is evaluated at least with second-order and the Hessian with first-order accuracy. This is accomplished by combining Green-Gauss gradient evaluation with least-squares based

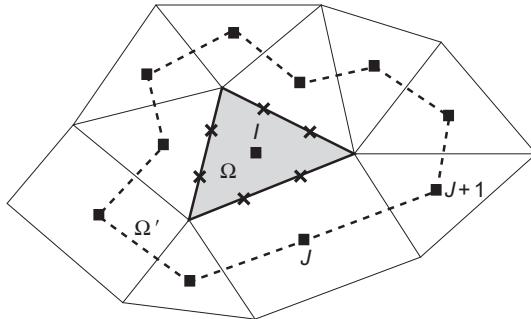


Figure 5.14 Stencil of the quadratic reconstruction method due to Delanaye [63, 64] in 2D (filled rectangles). Dashed line represents the integration path of the Green-Gauss gradient evaluation (control volume Ω'). Crosses denote the quadrature points for integration of the fluxes.

approximation of the second derivatives [63, 64], which leads to a numerically efficient scheme. But the memory and the CPU-time overheads are still quite significant in comparison to the linear reconstruction. The method utilizes a fixed stencil composed of face and node neighbors. The stencil is shown in Fig. 5.14 together with the integration path employed for the Green-Gauss gradient computation. In order to determine all coefficients of the quadratic polynomial, at least six (10 in 3D) values must be provided by the stencil. To maintain the accuracy provided by the quadratic reconstruction, it is necessary to consider a linear variation of the solution over the face instead of a constant value. This implies that the solution must be reconstructed at two points—the so-called *Gauss quadrature* points (cf. Fig. 5.14)—of a 2-D face (at three points of a triangular face), and that the fluxes have to be integrated in a piecewise manner over the face of the control volume [38].

Nowadays, the focus shifted to ENO (essentially non-oscillatory) and especially to WENO (weighted ENO) schemes for quadratic and higher-order reconstruction [65–85]. The idea is to employ varying discretization stencils (as to their orientation) in order to compute a number of different reconstruction polynomials for each control volume. The ENO scheme then selects the least oscillatory of the polynomials. The WENO methodology weights and combines the polynomials, the weights are inversely proportional to the degree of oscillations. In this way, the solution becomes spatially of high order in smooth regions, while true discontinuities are not smeared. A nice introduction to WENO schemes can be found in Ref. [73].

5.3.4 Evaluation of the gradients

An open point, which remains from the discussion of the piecewise linear and quadratic reconstruction, is the determination of the gradient. Gradients of the velocity components and the temperature are also required for the evaluation of the viscous fluxes (Section 5.4). Two approaches will be presented in the following: the first is based on the Green-Gauss theorem, and the second utilizes the least-squares method.

Green-Gauss approach

This method approximates the gradient of some scalar function U as the surface integral of the product of U with an outward-pointing unit normal vector over some control volume Ω' , that is,

$$\nabla U \approx \frac{1}{\Omega'} \int_{\partial\Omega'} U \vec{n} dS. \quad (5.49)$$

Median-dual scheme

Barth and Jespersen [30] derived a particular discretization of the Green-Gauss approach from the Galerkin finite-element method. Later on, the discretization was extended to 3D by Barth [86]. Barth and Jespersen applied equation (5.49) to the region formed by the union of the elements meeting at a node. They proved that the approach can be formulated such that it becomes compatible with the edge-based data structure. However, this works only for the median-dual scheme on triangular/tetrahedral grids. The resulting formula reads

$$\nabla U_i \approx \frac{1}{\Omega} \sum_{j=1}^{N_F} \frac{1}{2} (U_i + U_j) \vec{n}_{ij} \Delta S_{ij}. \quad (5.50)$$

Here, Ω' in Eq. (5.49) equals to the volume of the median-dual control volume Ω . The summation extends over all N_F edges incident to node i . Furthermore, \vec{n}_{ij} denotes the average unit normal vector according to Eq. (5.23), and ΔS_{ij} is the total face area, respectively. The same formula (5.50) is applicable in 2D or 3D. It is important to mention that the summation has to be changed at boundaries in order to obtain a consistent approximation [87] (see also Section 8.10).

Cell-centered scheme

We can use the Green-Gauss method in the cell-centered scheme as well. Hence, the gradient at some cell-centroid I can be obtained from

$$\nabla U_I \approx \frac{1}{\Omega} \sum_{J=1}^{N_F} \frac{1}{2} (U_I + U_J) \vec{n}_{IJ} \Delta S_{IJ}, \quad (5.51)$$

where the summation extends over all faces of the cell with the volume Ω . In Eq. (5.51), \vec{n}_{IJ} denotes the unit normal vector and ΔS_{IJ} the face area, respectively.

Mixed grids

The main attractiveness of the Green-Gauss gradient evaluation by Eq. (5.50) or (5.51) is its similarity to the computation of the fluxes (e.g., Eq. (5.19)). This means that no additional data structures are needed for the reconstruction of gradients. The main disadvantage is that the approximation in Eq. (5.50) or (5.51), respectively, fails on

mixed grids. It was demonstrated in [50] that the gradient can become highly inaccurate, particularly where different element types meet. We can solve the problem in the case of the median-dual scheme when we keep the volume Ω' in Eq. (5.49) identical to the union of all cells incident to node i . Referring to the situation sketched in Fig. 5.8, the gradient results then in 2D from

$$\nabla U_i \approx \frac{1}{\Omega'} \sum_{j=1}^{N_O} \frac{1}{2} (U_j + U_{j+1}) \vec{n}_j \Delta S_j \quad (5.52)$$

with $i = 0$, the number of outer faces $N_O = 6$ (given by the points P_1-P_6), and $j+1 = 1$ for $j = 6$. Furthermore, \vec{n}_j and ΔS_j stand for the unit normal vector and the area of the outer cell faces. In 3D, we can use

$$\nabla U_i \approx \frac{1}{\Omega'} \sum_{j=1}^{N_O} \frac{1}{3} (U_{j,1} + U_{j,2} + U_{j,3}) \vec{n}_j \Delta S_j, \quad (5.53)$$

when we assume all faces are triangles—either naturally or by decomposition. The same remedy can be also employed for the cell-centered scheme. The surface of the control volume Ω' is then defined by the centroids of the distant-one and distant-two neighboring cells [63, 64], as it is rendered in Fig. 5.14. The gradient is computed correspondingly to Eq. (5.52) or (5.53) with cell instead of node indexes.

The clear disadvantage of such a cell-based approach is the necessity of an additional data structure, which provides a link between the central node/centroid and the outer faces of Ω' . Thus, the approach is no longer grid-transparent (i.e., independent of the cell information) and an efficient gather-scatter loop is no longer possible. This renders the least-squares technique, which is described below, more attractive on mixed grids.

Using the edge-/face-based implementation (5.50) or (5.51) on triangular or tetrahedral grids, and the cell-based methodology (5.52) or (5.53) on mixed grids, the Green-Gauss approach is at least first-order accurate [64]. It is also consistent, that is, the gradient of a linear function is computed to roundoff error. First-order accuracy is sufficient for the linear reconstruction. Second-order accuracy on arbitrary grids, which is required for the quadratic reconstruction, can be achieved by subtracting an estimate of the truncation error from the first-order approximation of the gradient [64].

Least-squares approach

The evaluation of gradients by the least-squares approach was first introduced by Barth [37, 86]. In order to illustrate the method, let us consider the median-dual scheme. Herewith, the least-squares approach is based upon the use of a first-order Taylor series

approximation for each edge which is incident to the central node i . The change of the solution along an edge ij can be computed from

$$(\nabla U_i) \cdot \vec{r}_{ij} = U_j - U_i, \quad (5.54)$$

where \vec{r}_{ij} is given by Eq. (5.43) and represents the vector from node i to node j (see Fig. 5.9 or 5.13b). When we apply the relation (5.54) to all edges incident to node i , we obtain the following over-constrained system of linear equations

$$\begin{bmatrix} \Delta x_{i1} & \Delta y_{i1} & \Delta z_{i1} \\ \Delta x_{i2} & \Delta y_{i2} & \Delta z_{i2} \\ \vdots & \vdots & \vdots \\ \Delta x_{ij} & \Delta y_{ij} & \Delta z_{ij} \\ \vdots & \vdots & \vdots \\ \Delta x_{iN_A} & \Delta y_{iN_A} & \Delta z_{iN_A} \end{bmatrix} \begin{bmatrix} \partial_x U \\ \partial_y U \\ \partial_z U \end{bmatrix}_i = \begin{bmatrix} \theta_1 (U_1 - U_i) \\ \theta_2 (U_2 - U_i) \\ \vdots \\ \theta_j (U_j - U_i) \\ \vdots \\ \theta_{N_A} (U_{N_A} - U_i) \end{bmatrix} \quad (5.55)$$

with $\Delta(\cdot)_{ij} = (\cdot)_j - (\cdot)_i$ and $\partial_m(\cdot) = \partial(\cdot)/\partial m$. Further, N_A denotes the number of adjacent nodes j connected to i by an edge and θ_j stands for some weighting coefficient. The weights can depend on the geometry and/or on the solution (see, e.g., [41]). However, in practice θ_j is usually set to unity. For convenience, we abbreviate the above system (5.55) as

$$\bar{A} \vec{x} = \vec{b}. \quad (5.56)$$

Solving Eq. (5.56) for the gradient vector \vec{x} requires the inversion of the matrix \bar{A} . To prevent problems with ill-conditioning (particularly on stretched grids), Anderson and Bonhaus suggested to decompose \bar{A} into the product of an orthogonal matrix \bar{Q} and an upper triangular matrix \bar{R} using the Gram-Schmidt process [88]. Their approach was recently extended to 3D in [50]. Hence, the solution to Eq. (5.56) immediately follows from

$$\vec{x} = \bar{R}^{-1} \bar{Q}^T \vec{b}. \quad (5.57)$$

Using a lower case letter with double subscripts to denote a matrix element, we may write the Gram-Schmidt orthogonalization of the matrix $\bar{A} = [\vec{a}_1, \vec{a}_2, \vec{a}_3]$ as $\bar{Q} = [\vec{q}_1, \vec{q}_2, \vec{q}_3]$, where

$$\begin{aligned} \vec{q}_1 &= \frac{1}{r_{11}} \vec{a}_1, \\ \vec{q}_2 &= \frac{1}{r_{22}} \left(\vec{a}_2 - \frac{r_{12}}{r_{11}} \vec{a}_1 \right), \\ \vec{q}_3 &= \frac{1}{r_{33}} \left[\vec{a}_3 - \frac{r_{23}}{r_{22}} \vec{a}_2 - \left(\frac{r_{13}}{r_{11}} - \frac{r_{12} r_{23}}{r_{11} r_{22}} \right) \vec{a}_1 \right]. \end{aligned} \quad (5.58)$$

The entries in the upper triangular matrix \bar{R} are obtained from

$$\begin{aligned}
 r_{11} &= \sqrt{\sum_{j=1}^{N_A} (\Delta x_{ij})^2}, \\
 r_{12} &= \frac{1}{r_{11}} \sum_{j=1}^{N_A} \Delta x_{ij} \Delta \gamma_{ij}, \\
 r_{22} &= \sqrt{\sum_{j=1}^{N_A} (\Delta \gamma_{ij})^2 - r_{12}^2}, \\
 r_{13} &= \frac{1}{r_{11}} \sum_{j=1}^{N_A} \Delta x_{ij} \Delta z_{ij}, \\
 r_{23} &= \frac{1}{r_{22}} \left(\sum_{j=1}^{N_A} \Delta \gamma_{ij} \Delta z_{ij} - \frac{r_{12}}{r_{11}} \sum_{j=1}^{N_A} \Delta x_{ij} \Delta z_{ij} \right), \\
 r_{33} &= \sqrt{\sum_{j=1}^{N_A} (\Delta z_{ij})^2 - (r_{13}^2 + r_{23}^2)}.
 \end{aligned} \tag{5.59}$$

Using Eqs. (5.57)–(5.59), the gradient at node i follows from the weighted sum of the edge differences

$$\nabla U_i \equiv \vec{x} = \sum_{j=1}^{N_A} \vec{w}_{ij} \theta_j (U_j - U_i) \tag{5.60}$$

with the vector of weights \vec{w}_{ij} defined as

$$\vec{w}_{ij} = \begin{bmatrix} \alpha_{ij,1} - \frac{r_{12}}{r_{11}} \alpha_{ij,2} + \beta \alpha_{ij,3}, \\ \alpha_{ij,2} - \frac{r_{23}}{r_{22}} \alpha_{ij,3}, \\ \alpha_{ij,3} \end{bmatrix}. \tag{5.61}$$

The terms in Eq. (5.61) are given by

$$\begin{aligned}
 \alpha_{ij,1} &= \frac{\Delta x_{ij}}{r_{11}^2}, \\
 \alpha_{ij,2} &= \frac{1}{r_{22}^2} (\Delta \gamma_{ij} - r_{12} r_{11} \Delta x_{ij}), \\
 \alpha_{ij,3} &= \frac{1}{r_{33}^2} (\Delta z_{ij} - r_{23} r_{22} \Delta \gamma_{ij} + \beta \Delta x_{ij}),
 \end{aligned} \tag{5.62}$$

where

$$\beta = \frac{r_{12}r_{23} - r_{13}r_{22}}{r_{11}r_{22}}. \quad (5.63)$$

The formulation of the least-squares approach remains formally the same for a cell-centered scheme, only the nodes have to be substituted by cell-centroids. An example may be found in Ref. [16].

The least-squares method is first-order accurate on general grids [64]. It is also consistent, that is, the gradient of a linear function is computed to roundoff error, regardless of the type of the elements. Therefore, the method is particularly suited to mixed grids. The computational costs are comparable to those of the Green-Gauss approach, since only a vector-scalar multiplication (Eq. (5.60)) is needed within a single loop over faces/edges. However, we have to pre-compute and store the six entries (Eq. (5.59)) of the upper triangular matrix \bar{R} at each node. A detailed investigation in Ref. [89] revealed that the weighting coefficient θ_j in Eqs. (5.55) and (5.60) has to be set equal to the inverse of the distance between the nodes i and j (similar to θ_{ij} in Eq. (5.45)), in order to obtain an accurate approximation for the gradient of a **non-linear** function on a highly stretched and additionally curved grid. However, this does not help in the case of the cell-centered scheme on triangular (tetrahedral) grids [89].

Experience also shows that the least-squares approach requires some attention in the case of the median-dual scheme, if prismatic or hexahedral cells are employed on a viscous wall. Consider Fig. 5.15 and assume that we want to compute gradients of the velocity components at node i . It may become obvious that only the contribution from the edge ij is useful, since at other nodes connected to i by an edge $u = v = w = 0$. To increase the support of the stencil, we can insert the *virtual edges* [50], as they are rendered by the dashed lines in Fig. 5.15. The virtual edges help to improve the accuracy and the robustness of the discretization scheme considerably. It should be stressed that they are employed only for the gradient reconstruction but not for the flux computation.

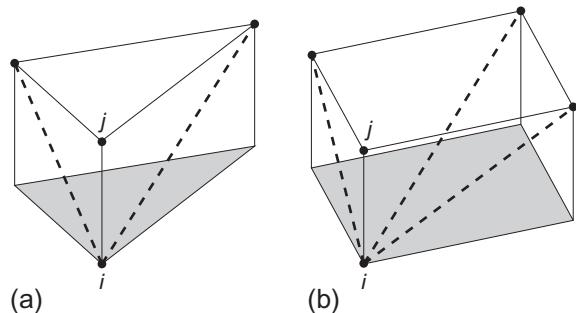


Figure 5.15 Virtual edges (dashed lines) used for the computation of gradients at node i [50]. Shown for prismatic (a) and hexahedral cell (b) on boundary (shaded).

5.3.5 Limiter functions

Second- and higher-order upwind spatial discretizations require the use of so-called *limiters* or *limiter functions* in order to prevent the generation of oscillations and spurious solutions in regions with large gradients (e.g., at shocks). Hence, what we want to achieve is at least a *monotonicity preserving* scheme. This means that maxima in the flow field must be non-increasing, minima non-decreasing, and no new local extrema may be created during the time evolution. We discussed this point in Section 4.3.5 for the case of structured upwind discretization schemes.

On unstructured grids, the purpose of a limiter is to reduce the gradients used to reconstruct the left and right state at the face of the control volume. The limiter function must be zero at strong discontinuities, in order to obtain a first-order upwind scheme which guarantees monotonicity. Setting the limiter to zero leads to the constant reconstruction of Eq. (5.38). Of course, the original unlimited reconstruction has to be retained in smooth flow regions, in order to keep the amount of numerical dissipation as low as possible. In the following, we shall describe two widely used limiter functions—namely the limiters of Barth and Jespersen [30], and of Venkatakrishnan [90, 91]. For a recently presented concept of multi-dimensional limiters, the reader is referred to [92] and [93].

Limiter of Barth and Jespersen

The first implementation of a limiter function on unstructured grids was presented in Ref. [30]. In the case of the median-dual scheme, it is defined at the node i as

$$\Psi_i = \min_j \begin{cases} \min \left(1, \frac{U_{\max} - U_i}{\Delta_2} \right) & \text{if } \Delta_2 > 0, \\ \min \left(1, \frac{U_{\min} - U_i}{\Delta_2} \right) & \text{if } \Delta_2 < 0, \\ 1 & \text{if } \Delta_2 = 0 \end{cases} \quad (5.64)$$

with the abbreviations

$$\begin{aligned} \Delta_2 &= \frac{1}{2} (\nabla U_i \cdot \vec{r}_{ij}), \\ U_{\max} &= \max(U_i, \max_j U_j), \\ U_{\min} &= \min(U_i, \min_j U_j). \end{aligned} \quad (5.65)$$

In Eqs. (5.64) and (5.65), \min_j or \max_j means the minimum or maximum value of all direct neighbors j of node i (i.e., all nodes connected to i by an edge). Furthermore, the edge vector \vec{r}_{ij} , which is shown in Fig. 5.9 or in Fig. 5.13b, is defined according to Eq. (5.43). Finally, U_j denotes a scalar quantity at some neighboring node j . Similar formulas to those above hold for the cell-centered scheme with cell instead of node indexes and with

$$\Delta_2 = \nabla U_I \cdot \vec{r}_L, \quad (5.66)$$

where \vec{r}_L denotes the vector from the cell-centroid to the midpoint of the corresponding cell face. In order to avoid division by a very small value of Δ_2 in Eq. (5.64), it is better to modify Δ_2 as $\text{Sign}(\Delta_2)(|\Delta_2| + \omega)$, where ω is approximately the machine accuracy [90].

Barth's limiter enforces a monotone solution. However, it is rather dissipative and it tends to smear discontinuities. A further problem presents the activation of the limiter due to numerical noise in smooth flow regions. This usually prevents the full convergence to steady state [39, 90]. Therefore, the limiter function due to Venkatakrishnan became more popular.

Venkatakrishnan's limiter

Venkatakrishnan's limiter [90, 91] is widely used because of its superior convergence properties. The limiter reduces the reconstructed gradient ∇U at the vertex i by the factor

$$\Psi_i = \min_j \begin{cases} \frac{1}{\Delta_2} \left[\frac{(\Delta_{1,\max}^2 + \epsilon^2)\Delta_2 + 2\Delta_2^2\Delta_{1,\max}}{\Delta_{1,\max}^2 + 2\Delta_2^2 + \Delta_{1,\max}\Delta_2 + \epsilon^2} \right] & \text{if } \Delta_2 > 0, \\ \frac{1}{\Delta_2} \left[\frac{(\Delta_{1,\min}^2 + \epsilon^2)\Delta_2 + 2\Delta_2^2\Delta_{1,\min}}{\Delta_{1,\min}^2 + 2\Delta_2^2 + \Delta_{1,\min}\Delta_2 + \epsilon^2} \right] & \text{if } \Delta_2 < 0, \\ 1 & \text{if } \Delta_2 = 0, \end{cases} \quad (5.67)$$

where

$$\begin{aligned} \Delta_{1,\max} &= U_{\max} - U_i, \\ \Delta_{1,\min} &= U_{\min} - U_i. \end{aligned} \quad (5.68)$$

where U_{\max} and U_{\min} stand for the minimum/maximum values of all surrounding nodes j and including the node i itself. Definitions of U_{\max} , U_{\min} , and Δ_2 are given in Eq. (5.65). The parameter ϵ^2 is intended to control the amount of limiting. Setting ϵ^2 to zero results in full limiting, but this may stall the convergence. Contrary to that, if ϵ^2 is set to a large value, the limiter function will return a value of about unity. Hence, there will be no limiting at all and wiggles could occur in the solution. In practice, it was found that ϵ^2 should be proportional to a local length scale, that is,

$$\epsilon^2 = (K \Delta h)^3, \quad (5.69)$$

where K is a constant of $\mathcal{O}(1)$ and Δh is, for example, the cube-root of the volume (square-root of the area in 2D) of the control volume. It is important to notice that the limiter function (5.67) must be defined with non-dimensional quantities. The influence of the coefficient K in Eq. (5.69) on the resolution of a shock is demonstrated in Fig. 5.16.

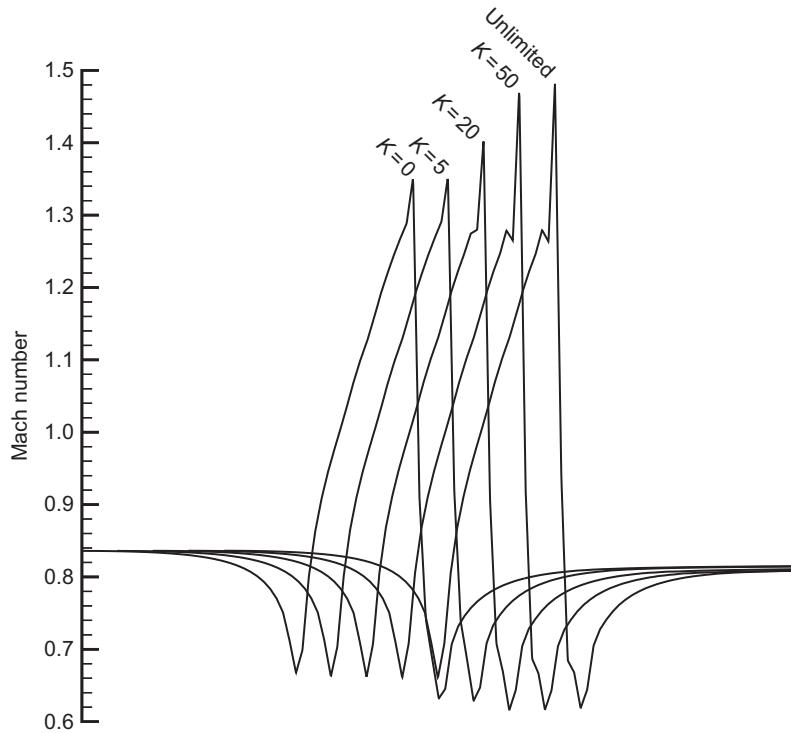


Figure 5.16 Effect of the constant K in Venkatakrishnan's limiter, given by Eq. (5.67), on the solution for an inviscid flow past a circular arc.

It can be seen that the fully limited ($K = 0$) and the solution for $K = 5$ are identical. However, the explicit time-stepping scheme converged only about three orders of magnitude for $K = 0$, whereas for $K = 5$ it converged to machine zero (Fig. 5.17). Figure 5.16 also shows that the solution becomes gradually unlimited with increasing values of K . This manifests itself as an increasing overshoot at the shock.

The computational effort for the evaluation of one of the above limiter functions is relatively high. Two loops over edges (faces in the case of the cell-centered scheme) and one loop over nodes (cells) are necessary in order to compute U_{\max} , U_{\min} as well as the limiter Ψ itself. Furthermore, U_{\max} , U_{\min} , and Ψ have to be stored node-(cell-)wise, separately for each flow variable.

5.4 DISCRETIZATION OF THE VISCOUS FLUXES

In order to evaluate the diffusive fluxes \vec{F}_v in Eq. (5.2), flow quantities and their first derivatives have to be known at the faces of the control volumes. The control volume for the viscous fluxes is conveniently chosen to be the same as for the convective fluxes

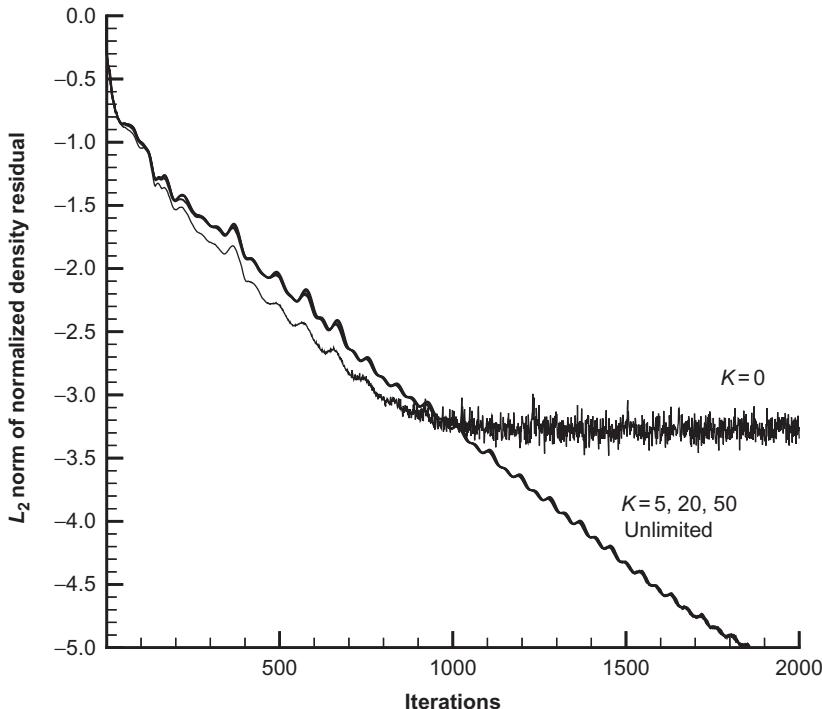


Figure 5.17 Effect of the constant K in Venkatakrishnan's limiter on the convergence for an inviscid flow past a circular arc.

in order to obtain a consistent spatial discretization and to simplify the data structure. Because of the elliptic nature of the viscous fluxes, values of the velocity components (u, v, w), the dynamic viscosity μ , and of the heat conduction coefficient k , which are required for the computation of the viscous terms (2.23), (2.24) and of the stresses (2.15), are simply averaged at a face. Thus, in the case of the cell-centered scheme (Fig. 5.13a), the values at the face IJ of the control volume result from

$$U_{IJ} = \frac{1}{2} (U_I + U_J), \quad (5.70)$$

where U is any of the above flow variables. A similar expression holds in the case of the median-dual scheme for the face ij —see Fig. 5.13b.

The remaining task is the evaluation of the first derivatives (gradients) of the velocity components in Eq. (2.15) and of temperature in Eq. (2.24). This can be accomplished in one of two ways, that is, by using

- element-based gradients, or
- average of gradients.

In the following, we shall learn more about both approaches.

5.4.1 Element-based gradients

A common feature of this type of gradient computation is the necessity to store either information about the grid elements or some coefficients related to the geometry of the elements. Hence, we have to extend the data structure beyond the face-/edge-based formulation presented earlier for the convective fluxes. Below, we discuss three well-established methods for the cell-centered and the median-dual discretization.

Face-centered control volume

One possible way of evaluating the gradients at a face of the control volume is to define an auxiliary control volume centered at the face and to employ the Green-Gauss theorem. We already discussed this approach in Section 4.4 in the framework of the structured finite-volume discretization. For example, in the case of the median-dual scheme we can compute the gradient at the edge-midpoint as the volume average of gradients for all elements which share the edge [94]. The element-based gradients are evaluated according to Eq. (5.49) by looping over all grid cells and accumulating the gradients at the edges. The values of U at the cell-faces are obtained by averaging the nodal values, in a manner similar to Eq. (5.53). This approach is relatively costly in terms of memory and the number of operations. However, it can be implemented for any mix of grid elements.

Approximate Galerkin finite-element approach

Another methodology, which is applicable to the median-dual scheme, was derived from the Galerkin finite-element method [31]. Basically speaking, the approach transforms the integration of gradients over the surface of the control volume into an evaluation of the Hessian matrix (second derivatives) at the central node. The viscous terms then follow the differential form of the Navier-Stokes equations in Cartesian coordinates (Eq. (A.4) with $\xi = x$, $\eta = y$, $\zeta = z$ and $J^{-1} = 1$), which contains terms such as

$$\partial_x(\mu \partial_x u), \text{ etc.}$$

with $\partial_m(\cdot) = \partial(\cdot)/\partial m$. Hence, no further integration of the viscous fluxes over the faces of the control volume is required.

The original scheme was formulated for purely triangular/tetrahedral grids. It employs a union of all elements that contain the particular node. In order to simplify the implementation, the dynamic viscosity coefficient is averaged from the nodal values, which is a difference to the Galerkin method. Then, the second derivatives can be evaluated at node i as follows [95]

$$\begin{aligned} & \begin{bmatrix} \partial_x(\mu \partial_x U) & \partial_y(\mu \partial_x U) & \partial_z(\mu \partial_x U) \\ \partial_x(\mu \partial_y U) & \partial_y(\mu \partial_y U) & \partial_z(\mu \partial_y U) \\ \partial_x(\mu \partial_z U) & \partial_y(\mu \partial_z U) & \partial_z(\mu \partial_z U) \end{bmatrix}_i \\ &= \frac{1}{\Omega'} \sum_{j=1}^{N_A} \left\{ \begin{bmatrix} \alpha_{xx} & \alpha_{xy} & \alpha_{xz} \\ \alpha_{yx} & \alpha_{yy} & \alpha_{yz} \\ \alpha_{zx} & \alpha_{zy} & \alpha_{zz} \end{bmatrix}_{ij} \frac{\mu_i + \mu_j}{2} (U_i - U_j) \right\}. \end{aligned} \quad (5.71)$$

The volume Ω' contains all tetrahedra which share the node i . The coefficient matrix $\bar{\alpha}$ in Eq. (5.71) is symmetric about the diagonal [95], that is, $\alpha_{xy} = \alpha_{yx}$, $\alpha_{xz} = \alpha_{zx}$, and $\alpha_{yz} = \alpha_{zy}$, respectively. Thus, it is necessary to store only six coefficients for each edge. The coefficients are given by [95]

$$\alpha_{nk} = \sum_e \frac{(S_e^j)_n (S_e^i)_k}{\Omega_e}, \quad (5.72)$$

where n, k denote the x, y, z subscripts, and $(S_e^i)_k, (S_e^j)_n$ represent components of the outer face vectors \vec{S}_e^i, \vec{S}_e^j displayed in Fig. 5.18. The summation is carried out over all tetrahedra (with particular volumes Ω_e) which share the edge ij . If the grid is stationary, the coefficients can be computed in a pre-processing step. A desirable feature is that the same edge-based data structure can be employed as for the convective fluxes.

The disadvantage of this approach is however that the full viscous terms are retained only on triangular or tetrahedral grids. For elements like prisms or hexahedra, this technique simplifies to a TSL-like approximation of the Navier-Stokes equations in all three coordinate directions [8]. An extension to non-simplex elements which conserves the full viscous terms was presented in [96]. But the efficient edge-based data structure can then no longer be used since the extended stencil involves nodes not connected by an edge to the point i .

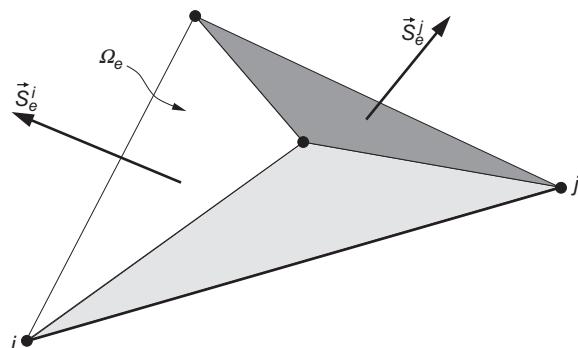


Figure 5.18 Viscous terms at node i : tetrahedron with volume Ω_e and the triangular faces involved in the computation of coefficients associated with the edge ij [95].

Average of nodal values

This scheme is intended for the cell-centered type of control volume and purely tetrahedral grids. It employs a modified version [4] of the stencil introduced in [62] to evaluate the gradients at the cell faces. The approach is based on an average of the values at the three nodes which define the cell face, combined with the quantities at the cell centroids. The first derivatives at a cell face result from the solution of the linear system of equations [4]

$$\begin{bmatrix} x_J - x_I & y_J - y_I & z_J - z_I \\ \frac{1}{2}(x_2 + x_3) - x_1 & \frac{1}{2}(y_2 + y_3) - y_1 & \frac{1}{2}(z_2 + z_3) - z_1 \\ \frac{1}{2}(x_1 + x_3) - x_2 & \frac{1}{2}(y_1 + y_3) - y_2 & \frac{1}{2}(z_1 + z_3) - z_2 \end{bmatrix} \begin{bmatrix} \partial_x U \\ \partial_y U \\ \partial_z U \end{bmatrix} = \begin{bmatrix} U_J - U_I \\ \frac{1}{2}(U_2 + U_3) - U_1 \\ \frac{1}{2}(U_1 + U_3) - U_2 \end{bmatrix}, \quad (5.73)$$

Referring to Fig. 5.19, the subscripts I, J denote the cell-centroids, and the subscripts $1, 2, 3$ stand for the nodes P_1, P_2 and P_3 , respectively. Flow variables at the nodes can be determined either from the inverse-distance weighting (Eq. (5.45)) or by the pseudo-Laplacian weighting [2] (similar to Eq. (5.24)).

5.4.2 Average of gradients

Since we already computed the gradients inside each control volume (e.g., using the piecewise linear reconstruction, Eq. (5.41) or (5.42)), it would be tempting to evaluate the gradient at the face-midpoint by a simple average [97]

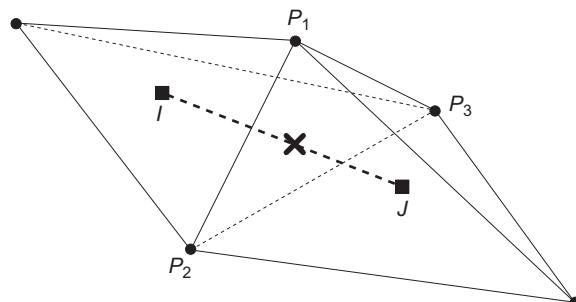


Figure 5.19 Cell-centered scheme: stencil for the computation of gradients on tetrahedral grids [4]. Cross denotes location where the gradients are evaluated in order to compute the viscous fluxes (face-midpoint $P_1P_2P_3$).

$$\overline{\nabla U}_{IJ} = \frac{1}{2} [\nabla U_I + \nabla U_J]. \quad (5.74)$$

This approach is particularly attractive, because it requires only the basic face- or edge-based data structure and no additional storage. However, as it was pointed out, for example, in Ref. [95], it leads to a wide stencil with an unfavorable distribution of the weights [50]. Furthermore, it was demonstrated in [50] that the stencil allows for the decoupling of the solution on quadrilateral or hexahedral grids.

The properties of the method can be improved, and particularly the decoupling can be prevented, by using the directional derivative along the connection between the cell-centroids (in the case of the cell-centered scheme), that is,

$$\left(\frac{\partial U}{\partial \ell} \right)_{IJ} \approx U_J - U_I \ell_{IJ}, \quad (5.75)$$

where ℓ_{IJ} represents the distance between the cell-centroids I and J (dashed line in Fig. 5.19). A similar expression holds also for the median-dual scheme with \vec{r}_{ij} according to Eq. (5.43). With the definition of the unit vector \vec{t}_{IJ} along the line connecting I and J ,

$$\vec{t}_{IJ} = \frac{\vec{r}_{IJ}}{\ell_{IJ}}, \quad (5.76)$$

the modified average can be written as [98, 99]

$$\nabla U_{IJ} = \overline{\nabla U}_{IJ} - \left[\overline{\nabla U}_{IJ} \cdot \vec{t}_{IJ} - \left(\frac{\partial U}{\partial \ell} \right)_{IJ} \right] \vec{t}_{IJ}, \quad (5.77)$$

where $\overline{\nabla U}_{IJ}$ is given by Eq. (5.74). The modification leads to strongly coupled stencils on tetrahedral as well as on prismatic or hexahedral grids [50]. This approach is also still compatible with the face-/edge-based data structure and requires no additional storage. It is therefore more attractive than the element-based methodology, provided the gradients inside control volumes are utilized for the convective fluxes anyway.

REFERENCES

- [1] Hirsch C. Numerical computation of internal and external flows, vols. 1 and 2. Chichester, UK: John Wiley and Sons; 1988.
- [2] Frink NT. Recent progress toward a three-dimensional Navier-Stokes solver. AIAA Paper 94-0061; 1994.
- [3] Hassan O, Probert EJ, Weatherill NP, Marchant MJ, Morgan K, Marcum DL. The numerical simulation of viscous transonic flows using unstructured grids. AIAA Paper 94-2346; 1994.
- [4] Frink NT, Pirzadeh SZ. Tetrahedral finite-volume solutions to the Navier-Stokes equations on complex configurations. In: 10th int. conf. on finite elements in fluids. Tucson, USA; 1998.
- [5] Luo H, Baum JD, Löhner R. Computation of compressible flows using a two-equation turbulence model on unstructured grids. AIAA Paper 97-0430; 1997.

- [6] Wang Q, Massey SJ, Abdol-Hamid KS, Frink NT. Solving Navier-Stokes equations with advanced turbulence models on three-dimensional unstructured grids. AIAA Paper 99-0156; 1999.
- [7] Luo H, Baum JD, Löhner R. A fast, matrix-free implicit method for compressible flows on unstructured grids. AIAA Paper 99-0936; 1999.
- [8] Mavriplis DJ, Venkatakrishnan V. A unified multigrid solver for the Navier-Stokes equations on mixed element meshes. ICASE Report No. 95-53; 1995.
- [9] Khawaja A, Kallinderis Y, Parthasarathy V. Implementation of adaptive hybrid grids for 3-D turbulent flows. AIAA Paper 96-0026; 1996.
- [10] Coirier WJ, Jorgenson PCE. A mixed-volume approach for the Euler and Navier-Stokes equations, AIAA Paper 96-0762; 1996.
- [11] Mavriplis DJ. Adaptive meshing technique for viscous flow calculations on mixed-element unstructured meshes. AIAA Paper 97-0857; 1997.
- [12] Haselbacher AC, McGuirk JJ, Page GJ. Finite volume discretization aspects for viscous flows on mixed unstructured grids. AIAA Paper 97-1946; 1997 [also AIAA J 1999;37:177-84].
- [13] Kano S, Nakahashi K. Navier-Stokes computations of HSCT off-design aerodynamics using unstructured hybrid meshes. AIAA Paper 98-0232; 1998.
- [14] Sharov D, Nakahashi K. Hybrid prismatic/tetrahedral grid generation for viscous flow applications. AIAA J 1998;36:157-62.
- [15] Blazek J, Irmisch S, Haselbacher A. Unstructured mixed-grid Navier-Stokes solver for turbomachinery applications. AIAA Paper 99-0664; 1999.
- [16] Strang WZ, Tomaro RF, Grismer MJ. The defining methods of Cobalt₆₀: a parallel, implicit, unstructured Euler/Navier-Stokes flow solver. AIAA Paper 99-0786; 1999.
- [17] Nakahashi K. FDM-FEM zonal approach for computations of compressible viscous flows. Lecture Notes in Physics, vol. 264. Springer Verlag; 1986. p. 494-8.
- [18] Nakahashi K. A finite-element method on prismatic elements for the three-dimensional Navier-Stokes equations. Lecture Notes in Physics, vol. 323. New York: Springer Verlag; 1989. p. 434-8.
- [19] Soetrino M, Imlay ST, Roberts DW, Taflin DE. Computations of viscous flows for multi-element wings using hybrid structured-unstructured grids. AIAA Paper 97-0623; 1997.
- [20] Kallinderis Y, Khawaja A, McMorris H. Hybrid prismatic/tetrahedral grid generation for viscous flows around complex geometries. AIAA J 1996;34:291-8.
- [21] Kallinderis Y, Khawaja A, McMorris H, Irmisch S, Walker D. Hybrid prismatic/tetrahedral grids for turbomachinery applications. In: Proc. 6th int. meshing roundtable. Park City, Utah, USA; 1997. p. 21-31.
- [22] Baker TJ. Discretization of the Navier-Stokes equations and mesh induced errors. In: Proc. 5th int. conf. on numerical grid generation in CFD. Mississippi State University, Mississippi; April 1996.
- [23] Baker TJ. Irregular meshes and the propagation of solution errors. In: Proc. 15th int. conf. on numerical methods in fluid dynamics. Monterey, CA; June 1996.
- [24] Mohanraj R, Neumeier Y, Zinn BT. Characteristic-based treatment of source terms in Euler equations for Roe scheme. AIAA J 1999;37:417-24.
- [25] Jameson A, Mavriplis D. Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh. AIAA J 1986;24:611-18.
- [26] Frink NT, Parikh P, Pirzadeh S. A fast upwind solver for the Euler equations on three-dimensional unstructured meshes. AIAA Paper 91-0102; 1991.
- [27] Mavriplis DJ. Multigrid solution of the two-dimensional Euler equations on unstructured triangular meshes. AIAA J 1988;26:824-31.
- [28] Mavriplis DJ, Jameson A, Martinelli L. Multigrid solution of the Navier-Stokes equations on triangular meshes. ICASE Report No. 89-11; 1989.
- [29] Whitaker DL, Grossman B, Löhner R. Two-dimensional euler computations on a triangular mesh using an upwind, finite-volume scheme. AIAA Paper 89-0470; 1989.
- [30] Barth TJ, Jespersen DC. The design and application of upwind schemes on unstructured meshes. AIAA Paper 89-0366; 1989.
- [31] Barth TJ. Numerical aspects of computing high-Reynolds number flows on unstructured meshes. AIAA Paper 91-0721; 1991.

- [32] Hwang CJ, Wu SJ. Adaptive finite volume upwind approach on mixed quadrilateral-triangular meshes. AIAA J 1993;31:61-7.
- [33] Whitaker DL. Three-dimensional unstructured grid Euler computations using a fully-implicit, upwind method. AIAA Paper 93-3337; 1993.
- [34] Wang ZJ. Improved formulation for geometric properties of arbitrary polyhedra. AIAA J 1999;37:1326-7.
- [35] www.ma.ic.ac.uk/~rn/centroid.pdf
- [36] Bruner CWS. Geometric properties of arbitrary polyhedra in terms of face geometry. AIAA J 1995;33:1350.
- [37] Barth TJ. Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations. AGARD R-787, Special course on unstructured grid methods for advection dominated flows. Brussels, Belgium; 18-22 May, 1992. p. 6.1-61.
- [38] Essers JA, Delanaye M, Rogiest P. An upwind-biased finite-volume technique for solving compressible Navier-Stokes equations on irregular meshes. Applications to supersonic blunt-body flows and shock-boundary layer interactions. AIAA Paper 93-3377; 1993.
- [39] Aftosmis M, Gaitonde D, Tavares TS. Behavior of linear reconstruction techniques on unstructured meshes. AIAA J 1995;33:2038-49.
- [40] Xia X, Nicolaides R. Covolume techniques for anisotropic media. Numer Math 1992;61:215-34.
- [41] Barth TJ, Linton SW. An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation. AIAA Paper 95-0221; 1995.
- [42] Venkatakrishnan V. Implicit schemes and parallel computing in unstructured grid CFD. ICASE Report No. 95-28; 1995.
- [43] Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. J Comput Phys 1996;127:380-97.
- [44] Jameson A, Schmidt W, Turkel E. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. AIAA Paper 81-1259; 1981.
- [45] Jameson A, Baker TJ, Weatherill NP. Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper 86-0103; 1986.
- [46] Holmes, D.G. and Connell SD. Solution of the 2D Navier-Stokes equations on unstructured adaptive meshes. AIAA Paper, No. 89-1932; 1989.
- [47] Rausch RD, Batina JT, Yang, HTY. Spatial adaption procedures on unstructured meshes for accurate unsteady aerodynamic flow computations. AIAA Paper 91-1106; 1991.
- [48] Swanson RC, Turkel E. On central difference and upwind schemes. J Comput Phys 1992;101:292-306.
- [49] Mavriplis DJ. Directional agglomeration multigrid techniques for high Reynolds number viscous flow solvers. AIAA Paper 98-0612; 1998.
- [50] Haselbacher A, Blazek J. On the accurate and efficient discretization of the Navier-Stokes equations on mixed grids. AIAA Paper 99-3363, 1999 [also AIAA J 2000;38:2094-102].
- [51] Roe PL. Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 1981;43:357-72.
- [52] Van Leer B. Towards the ultimate conservative difference scheme. V. A second order sequel to Godunov's method. J Comput Phys 1979;32:101-36.
- [53] Desideri JA, Dervieux A. Compressible flow solvers using unstructured grids. VKI Lecture Series 1988-05. Rhode-St-Genèse, Belgium: Von Karman Institute; 1988. p. 1-115.
- [54] Fezoui L, Stoufflet B. A class of implicit upwind schemes for Euler simulations with unstructured meshes. J Comput Phys 1989;84:174-206.
- [55] Whitaker DL, Slack DC, Walters RW. Solution algorithms for the two-dimensional Euler equations on unstructured meshes. AIAA Paper 90-0697; 1990.
- [56] Arminjon P, Dervieux A. Construction of TVD-like artificial viscosities on two-dimensional arbitrary FEM grids. J Comput Phys 1993;106:176-98.
- [57] Jameson A. Positive schemes and shock modelling for compressible flows. Int J Numer Meth Fluids 1995;20:743-76.
- [58] Frink NT. Upwind scheme for solving the Euler equations on unstructured tetrahedral meshes. AIAA J 1992;30:70-77.

- [59] Barth TJ, Frederickson PO. Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction. AIAA Paper 90-0013; 1990.
- [60] Barth TJ. Recent developments in high order k-exact reconstruction on unstructured meshes. AIAA Paper 93-0668; 1993.
- [61] Mitchell CR, Walters RW. k-Exact reconstruction for the Navier-Stokes equations on arbitrary grids. AIAA Paper 93-0536; 1993.
- [62] Mitchell CR. Improved reconstruction schemes for the Navier-Stokes equations on unstructured meshes. AIAA Paper 94-0642; 1994.
- [63] Delanaye M, Essers JA. Finite volume scheme with quadratic reconstruction on unstructured adaptive meshes applied to turbomachinery flows. ASME IGTI gas turbine conference, Houston, USA; 1995.
- [64] Delanaye M. Polynomial reconstruction finite volume schemes for the compressible Euler and Navier-Stokes equations on unstructured adaptive grids. PhD Thesis, The University of Liège, Belgium; 1996.
- [65] Harten A, Engquist B, Osher S, Chakravarthy S. Uniformly high order accurate essentially non-oscillatory schemes III. J Comput Phys 1987;71:231-303 [also ICASE Report No. 86-22; 1986].
- [66] Casper J, Atkins HL. A finite-volume high-order ENO scheme for two-dimensional hyperbolic systems. J Comput Phys 1993;106:62-76.
- [67] Godfrey AG, Mitchell CR, Walters RW. Practical aspects of spatially high-order accurate methods. AIAA J 1993;31:1634-42.
- [68] Abgrall R, Lafon FC. ENO schemes on unstructured meshes. VKI Lecture Series 1993-04. Rhode-St-Genèse, Belgium: Von Karman Institute; 1993.
- [69] Abgrall R. On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation. J Comput Phys 1994;114:45-58.
- [70] Jiang G-S, Shu C-W. Efficient implementation of weighted ENO schemes. J Comput Phys 1996;126:202-28.
- [71] Ollivier-Gooch CF. High-Order ENO schemes for unstructured meshes based on least-squares reconstruction. AIAA Paper 97-0540; 1997.
- [72] Stanescu D, Habashi WG. Essentially nonoscillatory Euler solutions on unstructured meshes using extrapolation. AIAA J 1998;36:1413-16.
- [73] Friedrich O. Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids. J Comput Phys 1998;144:194-212.
- [74] Shi J, Hu C, Shu C-W. A technique of treating negative weights in WENO schemes. J Comput Phys 2002;175:108-27.
- [75] Martín MP, Taylor EM, Wu M, Weirs VG. A bandwidth-optimized WENO scheme for the effective direct numerical simulation of compressible turbulence. J Comput Phys 2006;220:270-89.
- [76] Dumbser M, Käser M, Titarev VA, Toro EF. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. J Comput Phys 2007;226:204-43.
- [77] Borges R, Carmona M, Costa B, Don WS. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. J Comput Phys 2008;227:3191-211.
- [78] Dumbser M, Enaux C, Toro EF. Finite volume schemes of very high order of accuracy for stiff hyperbolic balance laws. J Comput Phys 2008;227:3971-4001.
- [79] Castro M, Costa B, Don WS. High order weighted essentially non-oscillatory WENO-Z schemes for hyperbolic conservation laws. J Comput Phys 2011;230:1766-92.
- [80] Tsoutsanis P, Antoniadis AF, Drikakis D. WENO schemes on arbitrary unstructured meshes for laminar, transitional and turbulent flows. J Comput Phys 2014;256:254-76.
- [81] Ivan L, Groth CPT. High-order solution-adaptive central essentially non-oscillatory (CENO) method for viscous flows. J Comput Phys 2014;257:830-62.
- [82] Xu L, Weng P. High order accurate and low dissipation method for unsteady compressible viscous flow computation on helicopter rotor in forward flight. J Comput Phys 2014;258:470-88.
- [83] Huang C-S, Arbogast T, Hung C-H. A re-averaged WENO reconstruction and a third order CWENO scheme for hyperbolic conservation laws. J Comput Phys 2014;262:291-312.
- [84] Boscheri W, Balsara DS, Dumbser M. Lagrangian ADER-WENO finite volume schemes on unstructured triangular meshes based on genuinely multidimensional HLL Riemann solvers. J Comput Phys 2014;267:112-38.

- [85] Fan P, Shen Y, Tian B, Yang C. A new smoothness indicator for improving the weighted essentially non-oscillatory scheme. *J Comput Phys* 2014;269:329–54.
- [86] Barth TJ. A 3-D upwind euler solver for unstructured meshes. *AIAA Paper* 91-1548; 1991.
- [87] Luo H, Baum JD, Löhner R. An improved finite volume scheme for compressible flows on unstructured grids. *AIAA Paper* 95-0348; 1995.
- [88] Anderson WK, Bonhaus DL. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Comput Fluids* 1994;23:1–21.
- [89] Mavriplis DJ. Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes. *AIAA Paper* 2003-3986; 2003.
- [90] Venkatakrishnan V. On the accuracy of limiters and convergence to steady state solutions. *AIAA Paper* 93-0880; 1993.
- [91] Venkatakrishnan V. Convergence to steady-state solutions of the Euler equations on unstructured grids with limiters. *J Comput Phys* 1995;118:120–30.
- [92] Li W, Ren Y-X, Lei G, Luo H. The multi-dimensional limiters for solving hyperbolic conservation laws on unstructured grids. *J Comput Phys* 2011;230:7775–95.
- [93] Li W, Ren Y-X. The multi-dimensional limiters for solving hyperbolic conservation laws on unstructured grids II: Extension to high order finite volume schemes. *J Comput Phys* 2012;231:4053–77.
- [94] Sharov D, Nakahashi K. Low speed preconditioning and LU-SGS scheme for 3-D viscous flow computations on unstructured grids. *AIAA Paper* 98-0614; 1998.
- [95] Mavriplis DJ. Three-dimensional multigrid Reynolds-averaged Navier-Stokes solver for unstructured meshes. *AIAA Paper* 94-1878, 1994; also *AIAA J* 1995;33:445–53.
- [96] Braaten ME, Connell SD. Three dimensional unstructured adaptive multigrid scheme for the Navier-Stokes equations. *AIAA J* 1995;34:281–90.
- [97] Luo H, Baum JD, Löhner R, Cabello J. Adaptive edge-based finite element schemes for the Euler and Navier-Stokes equations on unstructured grids. *AIAA Paper* 93-0336; 1993.
- [98] Crumpton PI, Moiner P, Giles MB. An unstructured algorithm for high Reynolds number flows on highly-stretched grids. In: 10th int. conf. num. meth. for laminar and turbulent flows. Swansea, England; July 21–25, 1997.
- [99] Weiss JM, Maruszewski JP, Smith WA. Implicit solution of preconditioned Navier-Stokes equations using algebraic multigrid. *AIAA J* 1999;37:29–36.

CHAPTER 6

Temporal Discretization

Contents

6.1	Explicit Time-Stepping Schemes	168
6.1.1	Multistage schemes (Runge-Kutta)	169
6.1.2	Hybrid multistage schemes	170
6.1.3	Treatment of the source term	172
6.1.4	Determination of the maximum time step	173
	<i>Time step on structured grids</i>	174
	<i>Time step on unstructured grids</i>	175
6.2	Implicit Time-Stepping Schemes	176
6.2.1	Matrix form of the implicit operator	178
	<i>Implicit operator on structured grids</i>	179
	<i>Implicit operator on unstructured grids</i>	181
6.2.2	Evaluation of the flux Jacobian	182
	<i>Central scheme</i>	183
	<i>Flux-vector splitting scheme</i>	183
	<i>Flux-difference splitting scheme</i>	185
	<i>Viscous flows</i>	185
6.2.3	Alternating direction implicit scheme	186
6.2.4	Lower-upper symmetric Gauss-Seidel scheme	188
	<i>LU-SGS on structured grids</i>	189
	<i>LU-SGS on unstructured grids</i>	192
6.2.5	Newton-Krylov method	193
	<i>GMRES method</i>	194
	<i>Computation of the flux Jacobian</i>	195
	<i>Preconditioning</i>	195
	<i>Start-up problem</i>	196
6.2.6	Implicit Runge-Kutta schemes	197
6.3	Methodologies for Unsteady Flows	202
6.3.1	Dual time-stepping for explicit multistage schemes	203
6.3.2	Dual time-stepping for implicit schemes	205
	References	206

The application of the *method of lines*, that is, the discretization of the governing equations (2.19) separately for spatial and temporal, leads, written down for each control volume, to a system of coupled ordinary differential equations in time

$$\frac{d(\Omega \vec{M} \vec{W})_I}{dt} = -\vec{R}_I. \quad (6.1)$$

In Eq. (6.1), Ω represents the volume, \vec{R} the residual, \bar{M} the mass matrix, and the index I denotes the particular control volume. The system (6.1) has to be integrated in time—either to obtain a steady-state solution ($\vec{R}_I = 0$), or to reproduce the time history of an unsteady flow.

We briefly discussed the aspects of the solution of the 6.1 in Section 3.2. We saw that the various *explicit* and *implicit* methods can be derived from a basic non-linear scheme. It reads for a stationary grid

$$\frac{(\Omega \bar{M})_I}{\Delta t_I} \Delta \vec{W}_I^n = -\frac{\beta}{1 + \omega} \vec{R}_I^{n+1} - \frac{1 - \beta}{1 + \omega} \vec{R}_I^n + \frac{\omega}{1 + \omega} \frac{(\Omega \bar{M})_I}{\Delta t_I} \Delta \vec{W}_I^{n-1}, \quad (6.2)$$

where

$$\Delta \vec{W}_I^n = \vec{W}_I^{n+1} - \vec{W}_I^n \quad (6.3)$$

stands for the *update* (correction) of the solution. The superscripts n and $(n+1)$ denote the time levels. Hence, \vec{W}^n means the flow solution at the present time t . Consequently, \vec{W}^{n+1} represents the solution at the time $(t + \Delta t)$. The parameters β and ω determine the discretization type (explicit or implicit) and also the temporal accuracy. For example, the condition expressed by Eq. (3.6) must be fulfilled to achieve second-order temporal accuracy.

In the following sections, we shall consider the most popular explicit and implicit time-stepping methods in some detail. We shall also present how the maximum allowable time step can be evaluated for a particular scheme. Furthermore, we shall discuss the issues of the appropriate implementations on structured as well as on unstructured grids. The last section will be devoted to time-accurate solutions of unsteady flow problems.

6.1 EXPLICIT TIME-STEPPING SCHEMES

An explicit scheme starts from a known solution \vec{W}^n and employs the corresponding residual \vec{R}^n in order to obtain a new solution at time $(t + \Delta t)$. In other words, the new solution \vec{W}^{n+1} depends solely on values already known. This fact makes the explicit schemes very simple and easy to implement.

As we discussed it in Section 3.2.1, a basic explicit scheme can be derived from Eq. (6.2) by setting $\beta = 0$ and $\omega = 0$. This results in

$$\bar{M}_I \Delta \vec{W}_I^n = -\frac{\Delta t_I}{\Omega_I} \vec{R}_I^n, \quad (6.4)$$

which is termed the *forward Euler* approximation. The mass matrix \bar{M} can be lumped (i.e., substituted by the identity matrix) for steady problems or for the cell-centered discretization.

The most popular and widespread explicit method is the *multistage* (Runge–Kutta) time-stepping scheme and its variant the *hybrid* multistage scheme. Therefore, we shall describe both methods in the following sections.

6.1.1 Multistage schemes (Runge–Kutta)

The concept of explicit multistage schemes was first presented by Jameson et al. [1]. The multistage scheme advances the solution in a number of steps—so-called *stages*—which can be viewed as a sequence of updates according to Eq. (6.4). Applied to the discretized governing equations (6.1), where the mass matrix was lumped, an m -stage scheme reads

$$\begin{aligned}\vec{W}_I^{(0)} &= \vec{W}_I^n, \\ \vec{W}_I^{(1)} &= \vec{W}_I^{(0)} - \alpha_1 \frac{\Delta t_I}{\Omega_I} \vec{R}_I^{(0)}, \\ \vec{W}_I^{(2)} &= \vec{W}_I^{(0)} - \alpha_2 \frac{\Delta t_I}{\Omega_I} \vec{R}_I^{(1)}, \\ &\vdots \\ \vec{W}_I^{n+1} &= \vec{W}_I^{(m)} = \vec{W}_I^{(0)} - \alpha_m \frac{\Delta t_I}{\Omega_I} \vec{R}_I^{(m-1)}.\end{aligned}\tag{6.5}$$

where $\alpha_1 \dots \alpha_m$ represent the stage coefficients. Furthermore, the denotation $\vec{R}_I^{(k)}$ means that the residual is evaluated with the solution $\vec{W}_I^{(k)}$ from the k th stage.

Unlike the classical Runge–Kutta schemes, only the zeroth solution and the newest residual are stored here in order to reduce the memory requirements. The stage coefficients can be tuned to increase the maximum time step and to improve the stability for a particular spatial discretization [2–4]. For consistency, it is only required that $\alpha_m = 1$. A consequence of the modification to the Runge–Kutta scheme is that second-order time accuracy can be realized only if $\alpha_{m-1} = 1/2$. Otherwise, the multistage scheme is first-order accurate in time.

The above multistage approach (6.5) is particularly suitable for upwind spatial discretization on structured as well as unstructured grids. Central discretization schemes perform more efficiently with the hybrid multistage methodology, which will be described next. Sets of optimized stage coefficients for first- and second-order upwind schemes are presented in Table 6.1 for three- to five-stage schemes [2]. Practical experience shows that the coefficients for the first-order scheme should be preferred in cases, where the flow field contains strong shocks, regardless of the order of the spatial discretization. This can be explained by the fact that every higher-order scheme switches to first order at shocks to prevent oscillations of the solution. However, the residuals at strong shocks influence the convergence to steady state the most significantly.

Table 6.1 Multistage scheme: optimized stage coefficients (α) and Courant-Friedrichs-Lowy (CFL) numbers (σ) for first- and second-order upwind spatial discretizations

Stages	First-order scheme			Second-order scheme		
	3	4	5	3	4	5
σ	1.5	2.0	2.5	0.69	0.92	1.15
α_1	0.1481	0.0833	0.0533	0.1918	0.1084	0.0695
α_2	0.4000	0.2069	0.1263	0.4929	0.2602	0.1602
α_3	1.0000	0.4265	0.2375	1.0000	0.5052	0.2898
α_4		1.0000	0.4414		1.0000	0.5060
α_5			1.0000			1.0000

The main disadvantage of every explicit scheme is that the time step (Δt) is severely restricted by the characteristics of the governing equations as well as by the grid geometry. We shall discuss the computation of the maximum allowable time step in [Section 6.1.4](#). Theoretical aspects of the determination of the time step and the *CFL number* will be considered in [Section 10.3](#) on stability analysis.

6.1.2 Hybrid multistage schemes

The computational work of an explicit multistage scheme [\(6.5\)](#), applied to the system [\(6.1\)](#), can be substantially reduced if the viscous fluxes and the dissipation are not re-evaluated at each stage. Additionally, the dissipation terms from different stages can be blended to increase the stability of the scheme. Methods of this type were devised by Martinelli [\[5\]](#) and by Mavriplis and Jameson [\[6\]](#). They are known as hybrid multistage schemes. Provided the stage coefficients are carefully optimized, the hybrid schemes are as robust as the basic multistage schemes.

Let us for illustration consider a popular 5-stage hybrid scheme, where the dissipative terms are evaluated at odd stages—generally denoted as the (5,3)-scheme. First, we split the spatial discretization into two parts, that is,

$$\vec{R}_I = (\vec{R}_c)_I - (\vec{R}_d)_I. \quad (6.6)$$

The first part, \vec{R}_c , contains the central discretization of the convective fluxes, which can be either the average of variables or the average of fluxes. It also includes the source term. The second part, \vec{R}_d , is composed of the viscous fluxes and the numerical dissipation. For example, in the case of the central scheme with artificial dissipation ([Sections 4.3.1](#) or [5.3.1](#)) we would set

$$(\vec{R}_c)_I = \sum_{k=1}^{N_F} [\vec{F}_c(\vec{W}_{av}) \Delta S]_k - (\vec{Q}\Omega)_I,$$

$$(\vec{R}_d)_I = \sum_{k=1}^{N_F} [\vec{F}_v \Delta S + \vec{D}]_k,$$

where \vec{W}_{av} represents the arithmetic average of flow variables from the left and the right side of face k .

With the residual split according to Eq. (6.6), the (5,3)-scheme can be formulated as

$$\begin{aligned}\vec{W}_I^{(0)} &= \vec{W}_I^n, \\ \vec{W}_I^{(1)} &= \vec{W}_I^{(0)} - \alpha_1 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(0)} - \vec{R}_d^{(0)} \right]_I, \\ \vec{W}_I^{(2)} &= \vec{W}_I^{(0)} - \alpha_2 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(1)} - \vec{R}_d^{(0)} \right]_I, \\ \vec{W}_I^{(3)} &= \vec{W}_I^{(0)} - \alpha_3 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(2)} - \vec{R}_d^{(2,0)} \right]_I, \\ \vec{W}_I^{(4)} &= \vec{W}_I^{(0)} - \alpha_4 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(3)} - \vec{R}_d^{(2,0)} \right]_I, \\ \vec{W}_I^{n+1} &= \vec{W}_I^{(0)} - \alpha_5 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(4)} - \vec{R}_d^{(4,2)} \right]_I,\end{aligned}\quad (6.7)$$

where

$$\begin{aligned}\vec{R}_d^{(2,0)} &= \beta_3 \vec{R}_d^{(2)} + (1 - \beta_3) \vec{R}_d^{(0)}, \\ \vec{R}_d^{(4,2)} &= \beta_5 \vec{R}_d^{(4)} + (1 - \beta_5) \vec{R}_d^{(2,0)}.\end{aligned}\quad (6.8)$$

The stage coefficients α_m and the blending coefficients β_m in the above relations (6.7) and (6.8) are given in Table 6.2 for central and upwind schemes. Both sets of coefficients are particularly optimized for the multigrid method (Section 9.4). We shall discuss the properties of the above hybrid multistage scheme later in Section 10.3.

It should be mentioned that it is also popular to evaluate the dissipation term \vec{R}_d in the first two stages only, without any blending. A well-known (5,2)-scheme, which

Table 6.2 Hybrid 5-stage scheme: optimized stage (α) and blending (β) coefficients, as well as CFL numbers (σ) for central and upwind spatial discretizations. Note the identical coefficients for the first- and the second-order upwind scheme but the different CFL numbers

Stage	Central scheme		First-order upwind		Second-order upwind	
	$\sigma = 3.6$	$\sigma = 2.0$	$\sigma = 1.0$	$\sigma = 1.0$		
1	0.2500	1.00	0.2742	1.00	0.2742	1.00
2	0.1667	0.00	0.2067	0.00	0.2067	0.00
3	0.3750	0.56	0.5020	0.56	0.5020	0.56
4	0.5000	0.00	0.5142	0.00	0.5142	0.00
5	1.0000	0.44	1.0000	0.44	1.0000	0.44

is often employed with the central spatial discretization, uses the stage coefficients of [Table 6.2](#). However, the (5,2)-scheme is less suitable for viscous flows and multigrid than the (5,3)-scheme.

6.1.3 Treatment of the source term

There are certain cases in which the source term \vec{Q} in Eq. (4.2) or (5.2) becomes dominant. Such situation is often encountered when chemistry or turbulence models are employed. The problem is that a large source term changes the flow variables rapidly in space and in time. The changes due to a strong source term happen at much smaller time scales than those of the flow equations. This increases the *stiffness* of the governing equations significantly. The stiffness is defined as the ratio of the largest to the smallest eigenvalue of the Jacobian matrix $\partial \vec{R} / \partial \vec{W}$. The stiffness can also be viewed as the ratio of the largest to the smallest time scale.

When we apply one of the above explicit multistage schemes (or any other purely explicit scheme) to a stiff system of equations, we will have to reduce the time step considerably in order to stabilize the time integration. Hence, the convergence to the steady state will become very slow. More seriously, an explicit scheme can even fail to find the correct solution [7]. A remedy suggested by Curtiss and Hirschfelder [8] is to treat the source term in an implicit way. In order to demonstrate the approach, we rewrite the basic explicit scheme in Eq. (6.4) as follows (cf. Eq. (4.2) or (5.2))

$$\begin{aligned} \frac{\Omega_I}{\Delta t_I} \Delta \vec{W}_I^n &= - \left[\sum_{k=1}^{N_F} (\vec{F}_c^n - \vec{F}_v^n)_k \Delta S_k - \Omega_I \vec{Q}_I^{n+1} \right] \\ &= -(\vec{R}_Q)_I^n + \Omega_I \vec{Q}_I^{n+1}, \end{aligned} \quad (6.9)$$

where the source term is now evaluated at the new time level ($n+1$). For simplicity, the mass matrix was omitted from Eq. (6.9). Since the value of the source term at the time ($n+1$) is unknown, we have to approximate it. For this purpose, we linearize the source term about the current time level n , resulting in

$$\vec{Q}^{n+1} \approx \vec{Q}^n + \frac{\partial \vec{Q}}{\partial \vec{W}} \Delta \vec{W}^n. \quad (6.10)$$

If we insert Eq. (6.10) into Eq. (6.9) and rearrange the terms, we obtain the following relation [9, 10]

$$\left[\frac{1}{\Delta t_I} \bar{I} - \left(\frac{\partial \vec{Q}}{\partial \vec{W}} \right)_I \right] \Delta \vec{W}_I^n = -\frac{1}{\Omega_I} [(\vec{R}_Q)_I^n - \Omega_I \vec{Q}_I^n], \quad (6.11)$$

where \bar{I} represents the identity matrix. The formulation (6.11) is called *point implicit* because the term in square brackets on the left-hand side—the implicit

operator—depends only on values in the control volume Ω_I itself. A comparison with Eq. (6.9) reveals that now the scalar time step Δt changed to a matrix. Thus, each flow equation becomes scaled by an individual parameter, corresponding to the associated eigenvalue. In this way, the disparity between the time scales is offset and the time step restriction due to the source term is alleviated.

When we apply the above point-implicit approach to the multistage scheme in Eq. (6.5), we obtain for the k th stage

$$\vec{W}_I^{(k)} = \vec{W}_I^{(0)} - \left[(\vec{R}_Q)_I^{(k-1)} - \Omega_I \vec{Q}_I^{(0)} \right] \left[\frac{\Omega_I}{\alpha_k \Delta t_I} \bar{I} - \left(\frac{\partial \vec{Q}}{\partial \vec{W}} \right)_I \right]^{-1} \quad (6.12)$$

with \vec{R}_Q defined in Eq. (6.9). A similar expression holds also for the hybrid multistage scheme in Eq. (6.7). The interested reader may find a detailed investigation of the influence of the source term on stability in Refs. [11–13].

A more elaborate approach is to treat the stiff source term by means of an *implicit Runge-Kutta* scheme, described first by Rosenbrock [14] and Butcher [15]. Schemes of this type are numerically more stable than the approach in Eq. (6.12), however they require the inversion of a large system of linear equations at each stage. Details of implicit Runge-Kutta schemes and applications are presented in Section 6.2.6.

6.1.4 Determination of the maximum time step

Every explicit time-stepping scheme remains stable only up to a certain value of the time step Δt . To be stable, a time-stepping scheme has to fulfill the so-called CFL condition [16]. It states that the domain of dependence of the numerical method has to include the domain of dependence of the partial differential equation. The CFL condition means for the basic explicit scheme (6.4) that the time step should be equal to or smaller than the time required to transport information across the stencil of the spatial discretization scheme. Hence, in 1D the condition for the time step would read for the linear convection equation

$$\Delta t = \sigma \frac{\Delta x}{|\Lambda_c|}, \quad (6.13)$$

where $\Delta x/|\Lambda_c|$ represents the time necessary to propagate information over the cell size Δx with the velocity Λ_c . The velocity Λ_c corresponds to the maximum eigenvalue of the convective flux Jacobian. The positive coefficient σ denotes the *CFL number*. The magnitude of the CFL number depends on the type and the parameters of the time-stepping scheme, as well as on the form of the spatial discretization scheme. We shall investigate the dependency of σ in Section 10.3 for two model problems. Tables 6.1 and 6.2 list the CFL numbers for various multistage schemes and discretizations.

The maximum time step can be determined for linear model equations with the aid of Von Neumann stability analysis (Section 10.3). However, the maximum time step can be calculated only approximately in multiple dimensions and for non-linear governing equations. In the following, we will present relations for the estimation of the time step on structured and unstructured grids, and for inviscid as well as for viscous flows.

Time step on structured grids

Euler equations

On a structured grid, the time step Δt can be determined for a control volume Ω_I from the approximate relation [17–19]

$$\Delta t_I = \sigma \frac{\Omega_I}{(\hat{\Lambda}_c^I + \hat{\Lambda}_c^J + \hat{\Lambda}_c^K)_I}. \quad (6.14)$$

The CFL number σ is given for multistage schemes in [Table 6.1](#) and for hybrid schemes in [Table 6.2](#). The spectral radii of the convective flux Jacobians (A.9) read for the three grid directions

$$\begin{aligned} \hat{\Lambda}_c^I &= (|\vec{v} \cdot \vec{n}^I| + c) \Delta S^I, \\ \hat{\Lambda}_c^J &= (|\vec{v} \cdot \vec{n}^J| + c) \Delta S^J, \\ \hat{\Lambda}_c^K &= (|\vec{v} \cdot \vec{n}^K| + c) \Delta S^K. \end{aligned} \quad (6.15)$$

The normal vectors and face areas in Eq. (6.15) are obtained by averaging the corresponding values from the two opposite sides of the control volume in the respective direction. For example, if a dual control volume (Section 4.2.3) would be oriented as sketched in Fig. 4.1b, we would use in the I -direction

$$\vec{n}_{i,j,k}^I = \frac{1}{2}(\vec{n}_1 - \vec{n}_2), \quad \Delta S_{i,j,k}^I = \frac{1}{2}(\Delta S_1 + \Delta S_2). \quad (6.16)$$

Similar expressions hold for the J - and K -direction.

With Eq. (6.14), we obtain a *local* time step, which is valid for one control volume only. If we are interested in a steady state solution, we may use the local time step to accelerate the convergence (cf. Section 9.1). However, if time accuracy is important, we have to employ one *global* time step for all volumes, that is,

$$\Delta t = \min_I (\Delta t_I), \quad (6.17)$$

where the minimum is taken over all control volumes.

Navier-Stokes equations

For viscous flows, the spectral radii of the viscous flux Jacobians (A.10) have to be included in the computation of Δt . They can severely limit the maximum time step in boundary layers. The time step can be evaluated from [5, 18, 19]

$$\Delta t_I = \sigma \frac{\Omega_I}{(\hat{\Lambda}_c^I + \hat{\Lambda}_v^I + \hat{\Lambda}_v^K)_I + C(\hat{\Lambda}_v^I + \hat{\Lambda}_v^K + \hat{\Lambda}_v^K)_I}. \quad (6.18)$$

The constant which multiplies the viscous spectral radii is usually set as $C = 4$ for central spatial discretizations, $C = 2$ for first-order upwind and $C = 1$ for second-order upwind discretizations. If we assume that an eddy-viscosity turbulence model is employed, the viscous spectral radii are given by [18, 19]

$$\hat{\Lambda}_v^I = \max \left(\frac{4}{3\rho}, \frac{\gamma}{\rho} \right) \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T} \right) \frac{(\Delta S^I)^2}{\Omega} \quad (6.19)$$

and similarly for the other directions. In Eq. (6.19), μ_L denotes the laminar and μ_T the turbulent dynamic viscosity coefficient, respectively. Furthermore, Pr_L and Pr_T are the laminar and the turbulent Prandtl numbers. The CFL numbers in Tables 6.1 and 6.2 apply also for viscous flows. Particularly efficient for viscous flows is the (5,3) hybrid scheme from Eq. (6.7).

Time step on unstructured grids

Several approaches were suggested for the estimation of the maximum time step on unstructured grids. We shall present two different approaches below.

Method 1

One proven method, which closely follows the implementation on structured grids, reads [6]

$$\Delta t_I = \sigma \frac{\Omega_I}{(\hat{\Lambda}_c + C\hat{\Lambda}_v)_I}, \quad (6.20)$$

where $\hat{\Lambda}_c$ and $\hat{\Lambda}_v$ represent a sum of the convective and viscous spectral radii over all faces of the control volume. As on structured grids, $1 \leq C \leq 4$ is usually used. In the case of the cell-centered scheme, the spectral radii are defined as [6]

$$\begin{aligned} (\hat{\Lambda}_c)_I &= \sum_{J=1}^{N_F} (|\vec{v}_{IJ} \cdot \vec{n}_{IJ}| + c_{IJ}) \Delta S_{IJ}, \\ (\hat{\Lambda}_v)_I &= \frac{1}{\Omega_I} \sum_{J=1}^{N_F} \left[\max \left(\frac{4}{3\rho_{IJ}}, \frac{\gamma_{IJ}}{\rho_{IJ}} \right) \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T} \right)_{IJ} (\Delta S_{IJ})^2 \right]. \end{aligned} \quad (6.21)$$

The values of the flow variables at the faces of the control volume are obtained by arithmetic averaging.

Method 2

The spectral radii predicted by Eq. (6.21) are too large, particularly on mixed element grids. This leads to a smaller time step than necessary for stability. The implementation in Ref. [20] offers a more accurate estimation of the time step, namely

$$\Delta t_I = \sigma \frac{\Omega_I}{(\hat{\Lambda}_c^x + \hat{\Lambda}_c^y + \hat{\Lambda}_c^z)_I + C(\hat{\Lambda}_v^x + \hat{\Lambda}_v^y + \hat{\Lambda}_v^z)_I} \quad (6.22)$$

with the convective spectral radii

$$\begin{aligned}\hat{\Lambda}_c^x &= (|u| + c) \Delta \hat{S}^x, \\ \hat{\Lambda}_c^y &= (|v| + c) \Delta \hat{S}^y, \\ \hat{\Lambda}_c^z &= (|w| + c) \Delta \hat{S}^z\end{aligned} \quad (6.23)$$

and with the viscous spectral radii (eddy-viscosity turbulence model assumed)

$$\hat{\Lambda}_v^x = \max \left(\frac{4}{3\rho}, \frac{\gamma}{\rho} \right) \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T} \right) \frac{(\Delta \hat{S}^x)^2}{\Omega}, \text{ etc.} \quad (6.24)$$

The variables $\Delta \hat{S}^x$, $\Delta \hat{S}^y$, and $\Delta \hat{S}^z$, respectively, represent projections of the control volume on the y - z -, x - z -, and the x - y -plane. They are given by the formulas

$$\begin{aligned}\Delta \hat{S}^x &= \frac{1}{2} \sum_{J=1}^{N_F} |S_x|_J, \\ \Delta \hat{S}^y &= \frac{1}{2} \sum_{J=1}^{N_F} |S_y|_J, \\ \Delta \hat{S}^z &= \frac{1}{2} \sum_{J=1}^{N_F} |S_z|_J,\end{aligned} \quad (6.25)$$

where S_x , S_y , and S_z denote the x -, y -, and the z -component of the face vector $\vec{S} = \vec{n} \cdot \Delta S$.

The CFL numbers remain in general the same as on structured grids. Thus, the values collected in Tables 6.1 and 6.2 still apply. Furthermore, the convergence to steady state can also be accelerated by local time stepping, in the same way as on the structured grids. A global time step, necessary for simulating unsteady flows, can be obtained from Eq. (6.17) as before.

6.2 IMPLICIT TIME-STEPPING SCHEMES

Various implicit time integration schemes can be obtained by setting $\beta \neq 0$ in Eq. (6.2). An implicit scheme with $\omega = 0$ was found to be best suited for the solution of stationary flow problems (for unsteady flows see Section 6.3). Herewith, Eq. (6.2) simplifies to

$$\frac{(\Omega \bar{M})_I}{\Delta t_I} \Delta \vec{W}_I^n = -\beta \vec{R}_I^{n+1} - (1 - \beta) \vec{R}_I^n. \quad (6.26)$$

As we can see, the implicit formulation leads to a set of non-linear equations for the unknown flow variables at the time $(t + \Delta t)$. The solution of Eq. (6.26) requires the evaluation of the residual at the new time level, that is, \vec{R}^{n+1} . Since we do not know \vec{W}^{n+1} , this cannot be done directly. However, we can linearize the residual \vec{R}^{n+1} in Eq. (6.26) about the current time level, that is,

$$\vec{R}_I^{n+1} \approx \vec{R}_I^n + \left(\frac{\partial \vec{R}}{\partial \vec{W}} \right)_I \Delta \vec{W}^n, \quad (6.27)$$

where the term $\partial \vec{R} / \partial \vec{W}$ is referred to as the *flux Jacobian*. We should mention that the flux Jacobian is often derived from a rather crude approximation to the spatial discretization represented by \vec{R}^n . For example, in the case of higher-order upwind discretizations, it is quite common to base the flux Jacobian solely on a first-order upwind scheme. However, for best efficiency and robustness, the flux Jacobian should still reflect the most important features of the spatial discretization.

If we substitute now the linearization in Eq. (6.27) for \vec{R}^{n+1} into Eq. (6.26), we obtain the following implicit scheme

$$\left[\frac{(\Omega \bar{M})_I}{\Delta t_I} + \beta \left(\frac{\partial \vec{R}}{\partial \vec{W}} \right)_I \right] \Delta \vec{W}^n = -\vec{R}_I^n. \quad (6.28)$$

The term in square brackets on the left-hand side of Eq. (6.28) is referred to as the *implicit operator* or the *system matrix*. Consequently, the right-hand side of Eq. (6.28) is called the *explicit operator*. It is only the explicit operator that determines the spatial accuracy of the solution.

The implicit operator constitutes a large, sparse, and non-symmetric block matrix with dimensions equal to the total number of cells (cell-centered scheme) or grid points (cell-vertex scheme). Below we will discuss further the form of the implicit operator for structured as well as for unstructured grids. As we already saw in Section 3.2, the mass matrix \bar{M} can be replaced by the identity matrix, without influencing the steady state solution. The parameter β in Eq. (6.28) is generally set to 1, which results in a first-order accurate temporal discretization. A second-order time accurate scheme is obtained for $\beta = 1/2$. However, this is not recommended since the scheme with $\beta = 1$ is much more robust, and the time accuracy is of no importance for steady problems.

In the case of stiff governing equations, the source term has to be included in the implicit operator. This happens quite naturally with the linearization of the residual in Eq. (6.27), which leads to a formulation identical to Eq. (6.10). As demonstrated in Ref. [12], the above implicit scheme (6.28) remains stable for any time step if the eigenvalues of $\partial \vec{Q} / \partial \vec{W}$ are all negative or zero.

The solution of the linear equation system (6.28) requires the inversion of the implicit operator, that is, the inversion of a very large matrix. In principle, this can be done in two ways. The first one consists of a direct matrix inversion, using either the Gaussian elimination or some direct sparse matrix method [21, 22]. However, because of the excessive amount of memory and the very high computational effort, this approach is not suited for practical problems [23].

The second possibility of inverting the implicit operator represent iterative methods. We mentioned the most widely used ones in Section 3.2.2. Iterative methods can be divided roughly into two groups. The first one consists of approaches which decompose the implicit operator into several parts—a process called *factorization*. The factors are constructed in such a way that they can be more easily inverted than the original implicit operator. To the second group belong schemes, which employ a Krylov-subspace method for the inversion of the implicit operator. In this case, the implicit time-stepping scheme (6.28) is usually turned into Newton's method by setting $\Delta t \rightarrow \infty$. The scheme is then named *Newton-Krylov* method.

In the following, we shall first discuss the matrix structure of the implicit operator. Then, we shall investigate the possibilities of computing the flux Jacobian $\partial \vec{R} / \partial \vec{W}$ in Eq. (6.28). Further, we shall present the three most popular iterative methods in detail. We shall close with a discussion of implicit Runge-Kutta schemes.

6.2.1 Matrix form of the implicit operator

Referring to Eq. (6.27), we can write the linearization of the residual \vec{R}^{n+1} in the form

$$\vec{R}^{n+1} \approx \vec{R}^n + \sum_{m=1}^{N_F} \left\{ \frac{\partial}{\partial \vec{W}} [(\vec{F}_c - \vec{F}_v)_m \Delta S_m] \Delta \vec{W}^n \right\} - \frac{\partial (\Omega \vec{Q})}{\partial \vec{W}} \Delta \vec{W}^n \quad (6.29)$$

with N_F being the number of faces of the control volume Ω (cf. Eq. (4.2) or (5.2)). Thus, the flux Jacobian reads

$$\frac{\partial \vec{R}}{\partial \vec{W}} = \sum_{m=1}^{N_F} \frac{\partial (\vec{F}_c)_m}{\partial \vec{W}} \Delta S_m - \sum_{m=1}^{N_F} \frac{\partial (\vec{F}_v)_m}{\partial \vec{W}} \Delta S_m - \frac{\partial (\Omega \vec{Q})}{\partial \vec{W}}. \quad (6.30)$$

It should be stressed that the flux Jacobian has to be conceived as an operator which acts on the update $\Delta \vec{W}$. As stated above, the convective and viscous fluxes in Eq. (6.30) do not necessarily have to be identical to the fluxes in the explicit operator.

Because of the significant differences between the implicit operators on structured and unstructured grids, we shall treat each case separately.

Implicit operator on structured grids

In order to derive the form of the system matrix in Eq. (6.28), let us consider the 1-D grid in Fig. 6.1. Let us further assume that the cell-vertex scheme with dual control volumes (Section 4.2.3) and a simple average of fluxes are used for the spatial discretization (cf. Fig. 4.8). In the absence of viscous fluxes, the residual is given by

$$\vec{R}_i^n = (\vec{F}_c)_{i+1/2} \Delta S_{i+1/2} + (\vec{F}_c)_{i-1/2} \Delta S_{i-1/2} - \Omega_i \vec{Q}_i. \quad (6.31)$$

Furthermore, the derivative of the convective fluxes at the face $m = i + 1/2$ in Eq. (6.30) can be expressed as follows

$$\begin{aligned} \frac{\partial(\vec{F}_c)_{i+1/2}}{\partial \vec{W}} &= \frac{\partial}{\partial \vec{W}} \left\{ \frac{1}{2} [\vec{F}_c(\vec{W}_{i+1}^n) + \vec{F}_c(\vec{W}_i^n)] \right\} \\ &= \frac{1}{2} [(\bar{A}_c)_{i+1} + (\bar{A}_c)_i], \end{aligned} \quad (6.32)$$

where \bar{A}_c denotes the convective flux Jacobian (Section A.9). Hence, according to Eqs. (6.29), (6.31), and (6.32), the residual at $(t + \Delta t)$ is approximated as

$$\begin{aligned} \vec{R}_i^{n+1} &\approx \vec{R}_i^n + \frac{1}{2} [(\bar{A}_c)_{i+1} \Delta \vec{W}_{i+1}^n + (\bar{A}_c)_i \Delta \vec{W}_i^n] \Delta S_{i+1/2} \\ &\quad + \frac{1}{2} [(\bar{A}_c)_{i-1} \Delta \vec{W}_{i-1}^n + (\bar{A}_c)_i \Delta \vec{W}_i^n] \Delta S_{i-1/2} - \frac{\partial(\Omega_i \vec{Q}_i)}{\partial \vec{W}}. \end{aligned} \quad (6.33)$$

Finally, for $\beta = 1$ and a lumped mass matrix we can derive from Eq. (6.26) the implicit scheme

$$\begin{aligned} &\left\{ \frac{\Omega_i}{\Delta t_i} \bar{I} + \frac{1}{2} [(\bar{A}_c)_i \Delta S_{i+1/2} + (\bar{A}_c)_i \Delta S_{i-1/2}] - \frac{\partial(\Omega_i \vec{Q}_i)}{\partial \vec{W}} \right. \\ &\quad \left. + \frac{1}{2} [(\bar{A}_c)_{i+1} \Delta S_{i+1/2}] + \frac{1}{2} [(\bar{A}_c)_{i-1} \Delta S_{i-1/2}] \right\} \Delta \vec{W}^n = -\vec{R}_i^n, \end{aligned} \quad (6.34)$$

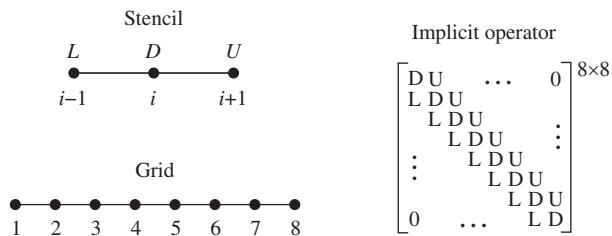


Figure 6.1 1-D structured grid and the associated implicit operator matrix for a 3-point stencil.

where \bar{I} stands for the identity matrix. It should be noted that the unit normal vector in each matrix \bar{A}_c is evaluated at the same side of the control volume as the associated area ΔS . As we can see, the implicit operator involves the same 3-point stencil (nodes $i - 1$, i , and $i + 1$) as the spatial discretization. In order to visualize the system matrix, we denote all terms in the implicit operator associated with the central node i as \mathbf{D} , that is,

$$\mathbf{D} \equiv \frac{\Omega_i}{\Delta t_i} \bar{I} + \frac{1}{2} [(\bar{A}_c)_i \Delta S_{i+1/2} + (\bar{A}_c)_i \Delta S_{i-1/2}] - \frac{\partial(\Omega_i \vec{Q}_i)}{\partial \vec{W}},$$

with the downwind node ($i + 1$) as \mathbf{U} , that is,

$$\mathbf{U} \equiv \frac{1}{2} (\bar{A}_c)_{i+1} \Delta S_{i+1/2},$$

and with the upwind node ($i - 1$) as \mathbf{L} , that is,

$$\mathbf{L} \equiv \frac{1}{2} (\bar{A}_c)_{i-1} \Delta S_{i-1/2},$$

respectively. Writing down Eq. (6.34) for all eight nodes of the grid in Fig. 6.1, we obtain the 8×8 block-tridiagonal matrix displayed on the right side of Fig. 6.1. Each of the blocks \mathbf{L} , \mathbf{D} , and \mathbf{U} represents a 3×3 matrix in 1D (because of the three conservation equations).

The same ideas carry over to multiple dimensions. For example, if the spatial discretization in 2D would involve the 5-point stencil sketched in Fig. 6.2, we would

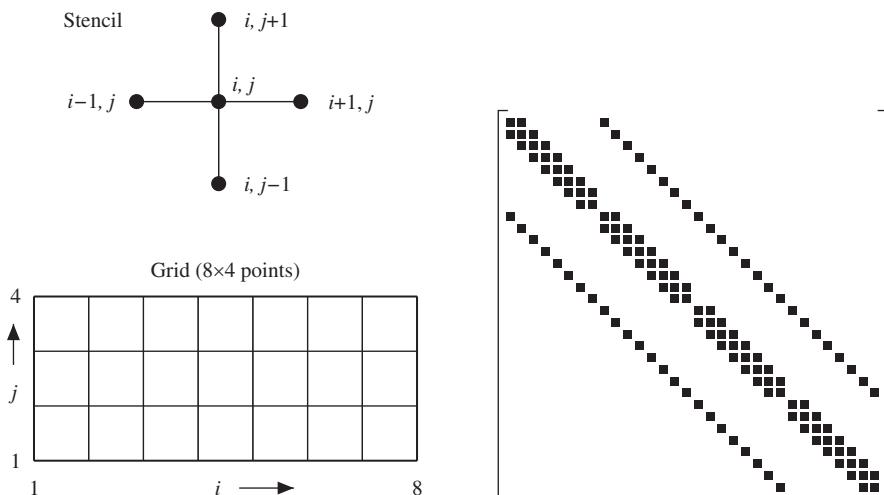


Figure 6.2 2-D structured grid (left) and the associated implicit operator matrix for a 5-point stencil (right). Nonzero block matrices displayed as filled rectangles.

obtain a block-pentadiagonal matrix. This is shown on the right side of Fig. 6.2. The nodes were ordered such that the i -index runs faster than the j -index (corresponds to $\text{mat}(i,j)$ in FORTRAN). It should be noted that the second off-diagonal is at the distance of eight elements from the main diagonal (the total number of nodes in i -direction). Finally, the system matrix would become block-septadiagonal in 3D, if we would employ the 7-point stencil of Fig. 4.9b for the spatial discretization.

In summary, we can state that the system matrix always possesses a regular, sparse and banded form for structured grids. It should be further mentioned that the term

$$\frac{(\Omega \bar{M})_I}{\Delta t_I} \quad (6.35)$$

in Eq. (6.28), which is always located on the main diagonal, can cause difficulties. Namely, if the time step becomes large, some iterative matrix inversion schemes (e.g., Gauss-Seidel) may fail due to reduced diagonal dominance of the implicit operator.

Implicit operator on unstructured grids

The appearance of the system matrix changes completely when we proceed from structured to unstructured grids. This can be demonstrated with the aid of a small unstructured grid sketched in Fig. 6.3. We want assume that the spatial discretization is given by the cell-centered scheme presented in Section 5.2.1, which uses flow values only from the nearest neighbors (e.g., like a first-order upwind scheme—see Sections 5.3.2 and 5.3.3). Thus, for example, the stencil for cell 2 includes the cells 18, 10, and 13. The resulting system matrix is displayed on the right side of Fig. 6.3. It is obvious that

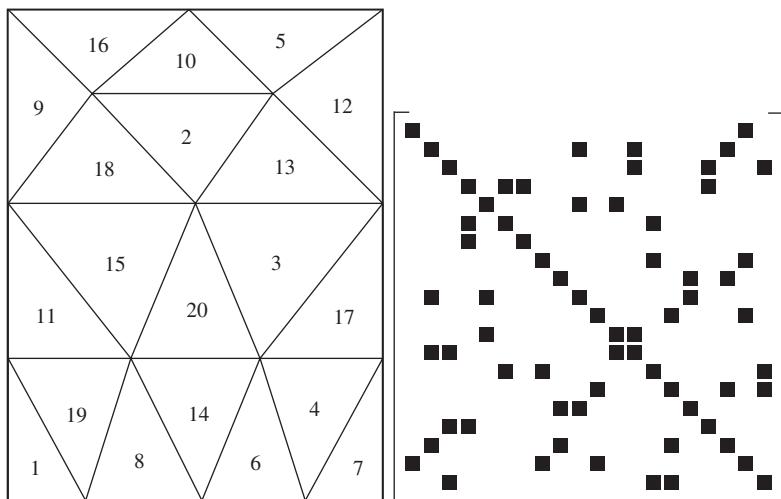


Figure 6.3 2-D unstructured grid (left) and the associated matrix of the implicit operator for a nearest neighbor stencil (right). Nonzero block matrices are displayed as filled rectangles.

since the grid cells (nodes in the case of a median-dual scheme) are in general numbered in an arbitrary order, no regular pattern can be expected for the matrix. Only the main diagonal, which contains at least the expression (6.35) and possibly the derivative of the source term, is always present.

The quasi-random distribution of nonzero elements in the system matrix is undesirable. It slows down the convergence of iterative inversion methods like Gauss-Seidel. Furthermore, preconditioning techniques for Krylov-subspace methods like incomplete lower-upper (ILU) factorization scheme cannot be used effectively. Therefore, strategies were developed where the cells (nodes) are renumbered such that the bandwidth of the system matrix is considerably reduced, that is, the nonzero elements are clustered close to the main diagonal. The best-known renumbering strategy is the *Reverse-Cuthill-McKee* (RCM) algorithm [24, 25]. Figure 6.4 shows the resulting cell numbering and the system matrix when the RCM algorithm is applied to the example grid of Fig. 6.3. As we can see, the bandwidth of the matrix is reduced significantly and the matrix obtains a more regular structure.

Several other renumbering strategies were developed in order to minimize cache misses or to allow for vectorization of the numerical scheme. An overview can be found, for example, in Ref. [26].

6.2.2 Evaluation of the flux Jacobian

Depending on the type of the underlying spatial discretization scheme, an analytical evaluation of the flux Jacobian $\partial \vec{R} / \partial \vec{W}$ in Eq. (6.28) may become very complex if not

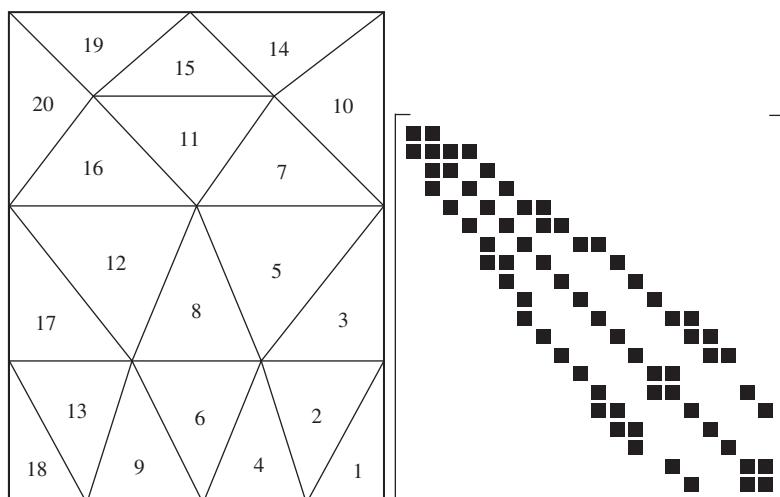


Figure 6.4 Reduced bandwidth (from 18 to 5) of the implicit operator from Fig. 6.3 with reverse-Cuthill-McKee ordering. Nonzero block matrices are displayed as filled rectangles.

impossible. In order to make the concepts more clear, we shall derive the flux Jacobian for inviscid flows and then discuss the extension to the Navier-Stokes equations.

Central scheme

The flux Jacobian is most easily formulated in the case of the central spatial discretization. As we already saw for the example in Fig. 6.1, $\partial \vec{R} / \partial \vec{W}$ consists of the convective flux Jacobians (cf. Eq. (6.34)), which can be derived analytically (see Section A.9). Artificial viscosity is usually included in a simplified form, without the non-linear pressure sensor (Eq. (4.55)). We shall return to this point in Section 6.2.3.

Flux-vector splitting scheme

The evaluation of the flux Jacobian becomes more involved when one of the flux-vector splitting schemes (Section 4.3.2) is used as the basis for its derivation. Let us, for illustration, consider the scheme due to Steger and Warming [27]. Previous investigations [28] revealed that the Steger-Warming splitting is preferable over, for example, van Leer's flux-vector splitting scheme (Eq. (4.60)) for various upwind discretizations of the implicit operator.

The basic idea of the Steger-Warming flux-vector splitting scheme is to divide the convective fluxes into a positive and a negative part, that is,

$$\vec{F}_c = \vec{F}_c^+ + \vec{F}_c^- \quad (6.36)$$

with the fluxes defined as

$$\vec{F}_c^\pm = \bar{A}_{SW}^\pm \vec{W} = (\bar{T} \bar{\Lambda}^\pm \bar{T}^{-1}) \vec{W}. \quad (6.37)$$

where \bar{A}_{SW}^\pm denotes the positive/negative Steger-Warming flux-splitting Jacobian. Furthermore, \bar{T} represents the matrix of right eigenvectors, \bar{T}^{-1} the matrix of left eigenvectors, and $\bar{\Lambda}^\pm$ stands for the diagonal matrix of positive/negative eigenvalues, respectively (cf. Section A.11). The eigenvalue matrices are defined as [27]

$$\bar{\Lambda}^\pm = \frac{1}{2}(\bar{\Lambda}_c \pm |\bar{\Lambda}_c|), \quad (6.38)$$

where $\bar{\Lambda}_c$ is given by Eq. (A.84).

Using the splitting defined in Eq. (6.36), we obtain for the product of the flux Jacobian with the update $\Delta \vec{W}^n$ in Eq. (6.29)

$$\frac{\partial \vec{R}_I}{\partial \vec{W}} \Delta \vec{W}^n = \sum_{m=1}^{N_F} \left[\frac{\partial (\vec{F}_c^+ \Delta S)_m}{\partial \vec{W}_{L,m}} \Delta \vec{W}_{L,m}^n + \frac{\partial (\vec{F}_c^- \Delta S)_m}{\partial \vec{W}_{R,m}} \Delta \vec{W}_{R,m}^n \right]. \quad (6.39)$$

where $\Delta \vec{W}_{L,m}^n$ and $\Delta \vec{W}_{R,m}^n$ denote the updates of the left and right state at the face m , respectively. On structured grids, the left and right state can be evaluated by the MUSCL approach (Eq. (4.46)). On unstructured grids, the reconstruction methods discussed

in Section 5.3.3 can be applied. However, the stencil becomes wider with increasing accuracy, which leads to larger bandwidth of the system matrix. Therefore, and in order to reduce the numerical complexity, only first-order accurate approximation is usually employed in Eq. (6.39). As a compromise, we could reconstruct the left and right state with higher accuracy but retain the stencil of the first-order scheme for the evaluation of the derivatives [29].

To proceed with the discussion of the evaluation of the derivatives $\partial \vec{F}_c^\pm / \partial \vec{W}$ in Eq. (6.39), let us consider, for example, the positive flux at face m

$$\frac{\partial (\vec{F}_c^+ \Delta S)_m}{\partial \vec{W}_{L,m}} = \frac{\partial}{\partial \vec{W}_{L,m}} [(\bar{A}_{SW}^+ \vec{W})_{L,m} \Delta S_m]$$

which becomes with Eqs. (6.37) and (6.38)

$$\begin{aligned} \frac{\partial (\vec{F}_c^+ \Delta S)_m}{\partial \vec{W}_{L,m}} &= \frac{\Delta S_m}{2} [(\bar{A}_c)_{L,m} + |(\bar{A}_c)_{L,m}|] \\ &\quad + \frac{\Delta S_m}{2} \left[\frac{\partial (\bar{A}_c)_{L,m}}{\partial \vec{W}_{L,m}} + \frac{\partial |(\bar{A}_c)_{L,m}|}{\partial \vec{W}_{L,m}} \right] \vec{W}_{L,m}^n. \end{aligned} \quad (6.40)$$

An expression similar to Eq. (6.40) can also be found for the negative flux. As we can see, the first term in Eq. (6.40) consists of convective flux Jacobians (see Section A.9) and thus presents no difficulty. However, the second term involves derivatives of matrix elements. Although it is possible to obtain the derivatives analytically either by hand calculation or by using a symbolic algebra package, this will produce a large, computationally inefficient code [30]. Alternatively, it is possible to assume the matrix \bar{A}_c is locally constant so that the second term in Eq. (6.40) can be neglected. However, depending on the type of the implicit scheme, this may severely restrict the CFL number [31].

Other approaches that we could use to compute the derivatives $\partial \vec{F}_c^\pm / \partial \vec{W}$ in Eq. (6.39) would be automatic differentiation of the source code (e.g., using ADIFOR [32], ADIC [33], OpenAD [34], or Adept [35]), or the finite-difference method (see, e.g., [23, 30]). Herewith, the derivative of the i th component of a vector \vec{F} with respect to the j th component of a dependent variable \vec{X} can be approximated as

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\vec{X} + h_j \vec{e}^j) - f_i(\vec{X})}{h_j}, \quad (6.41)$$

where \vec{e}^j denotes the j th standard basis vector. Dennis and Schnabel [36] suggested a step size h_j of the form

$$h_j = \sqrt{\epsilon} \max \{|x_j|, \text{typ } x_j\} \text{ sign}(x_j) \quad (6.42)$$

with ϵ being the machine accuracy and $\text{typ } x_j$ a typical size of x_j .

The reader is also referred to Refs. [37, 38] for hints on efficient numerical evaluation of Jacobian matrices.

Flux-difference splitting scheme

In the case of the flux-difference splitting scheme due to Roe (Section 4.3.3, Eq. (4.91)), we can write the product of the flux Jacobian with the update in Eq. (6.29) as

$$\begin{aligned} \frac{\partial \vec{R}_I}{\partial \vec{W}} \Delta \vec{W}^n = & \sum_{m=1}^{N_F} \frac{\Delta S_m}{2} \left\{ (\bar{A}_c)_{L,m} \Delta \vec{W}_{L,m}^n + (\bar{A}_c)_{R,m} \Delta \vec{W}_{R,m}^n \right. \\ & - \frac{\partial}{\partial \vec{W}_{L,m}} [|\bar{A}_{Roe}|_m (\vec{W}_{R,m}^n - \vec{W}_{L,m}^n)] \Delta \vec{W}_{L,m}^n \\ & \left. - \frac{\partial}{\partial \vec{W}_{R,m}} [|\bar{A}_{Roe}|_m (\vec{W}_{R,m}^n - \vec{W}_{L,m}^n)] \Delta \vec{W}_{R,m}^n \right\}. \end{aligned} \quad (6.43)$$

Similar to flux-vector splitting, the expression (6.43) contains convective flux Jacobians as well as derivatives of the Roe matrix \bar{A}_{Roe} . The derivatives were presented in [31]. Since the corresponding formulas are very complex (see also Ref. [30]), it is a better idea to evaluate the term $\partial \vec{F}_c / \partial \vec{W}$ numerically, as discussed above. However, we can also simplify Eq. (6.43) by assuming locally constant Roe matrices [39]

$$\begin{aligned} \frac{\partial \vec{R}_I}{\partial \vec{W}} \Delta \vec{W}^n \approx & \sum_{m=1}^{N_F} \frac{\Delta S_m}{2} \left\{ (\bar{A}_c)_{L,m} \Delta \vec{W}_{L,m}^n + (\bar{A}_c)_{R,m} \Delta \vec{W}_{R,m}^n \right. \\ & \left. - |\bar{A}_{Roe}|_m (\Delta \vec{W}_{R,m}^n - \Delta \vec{W}_{L,m}^n) \right\}. \end{aligned} \quad (6.44)$$

In contrast to the Steger-Warming flux-vector splitting scheme, the above approximate linearization (6.44) degrades the performance of the implicit scheme only slightly [31].

Viscous flows

For the Navier-Stokes equations, we have to account also for the viscous fluxes in the implicit operator. The derivative $\partial \vec{F}_v / \partial \vec{W}$, that is, the viscous flux Jacobian in Eq. (6.30) is in general not straightforward to obtain. Additional complexity arises due to the fact that the viscous flux vector contains derivatives of flow variables. For this reason, we have either to evaluate the viscous flux Jacobian by finite differences (Eq. (6.41)), or we have to use a simplified formulation.

In the case of the TSL approximation of the Navier-Stokes equations (cf. Section 2.4.3 and Section A.6), it is possible to find the viscous flux Jacobian analytically by assuming locally constant dynamic viscosity and thermal conductivity coefficients. Then, according to the discussion in Section A.10, the term related to the viscous fluxes in Eq. (6.29) becomes

$$\frac{\partial (\vec{F}_v \Delta S)_m}{\partial \vec{W}} \Delta \vec{W}^n \approx [(\bar{A}_v^*)_{R,m} \Delta \vec{W}_{R,m}^n - (\bar{A}_v^*)_{L,m} \Delta \vec{W}_{L,m}^n] \Delta S_m. \quad (6.45)$$

where \bar{A}_v^* stands for the viscous flux Jacobian given by Eq. (A.71) or Eq. (A.75) but without the spatial operators $\partial_\psi()$ (cf. Eq. (A.74) and (A.79)). The Jacobians are evaluated using either the left or the right state for all variables except for the dynamic viscosity, which is determined from an arithmetic average. It should be mentioned that Eq. (6.45) leads to a second-order central difference approximation in the implicit operator, supposed the left and right state are computed with first-order accuracy.

6.2.3 Alternating direction implicit scheme

The *alternating direction implicit* (ADI) scheme was one of the first iterative implicit methods [40]. The ADI scheme can be implemented on structured grids only. It is based on an approximate splitting (or, in other words, factorization) of the implicit operator in Eq. (6.28) into two (in 2D) or three (in 3D) factors. Each factor contains the linearization of the convective and viscous fluxes for one particular direction in the computational space. In 3D, this leads to the formulation [40, 41]

$$\left\{ \begin{array}{l} \frac{\Omega}{\Delta t} \bar{I} + \frac{\partial [(\vec{F}_c^I - \vec{F}_v^I) \Delta S^I]_{i+1/2}}{\partial \vec{W}} + \frac{\partial [(\vec{F}_c^I - \vec{F}_v^I) \Delta S^I]_{i-1/2}}{\partial \vec{W}} \\ \frac{\Omega}{\Delta t} \bar{I} + \frac{\partial [(\vec{F}_c^J - \vec{F}_v^J) \Delta S^J]_{j+1/2}}{\partial \vec{W}} + \frac{\partial [(\vec{F}_c^J - \vec{F}_v^J) \Delta S^J]_{j-1/2}}{\partial \vec{W}} \\ \frac{\Omega}{\Delta t} \bar{I} + \frac{\partial [(\vec{F}_c^K - \vec{F}_v^K) \Delta S^K]_{k+1/2}}{\partial \vec{W}} + \frac{\partial [(\vec{F}_c^K - \vec{F}_v^K) \Delta S^K]_{k-1/2}}{\partial \vec{W}} \\ - \frac{\partial (\Omega \vec{Q})}{\partial \vec{W}} \end{array} \right\} \Delta \vec{W}^n = -\vec{R}^n. \quad (6.46)$$

For clarity, the node indexes i, j, k were omitted from Eq. (6.46) where not required. Furthermore, the superscripts I, J, K in Eq. (6.46) mark the flux vector or of the face area associated with certain coordinate in the computational space. For example, $\Delta S^I_{i+1/2,j,k}$ corresponds to ΔS_2 in Fig. 4.1b. By replacing the node indexes by cell indexes, the scheme in Eq. (6.46) can be applied to a cell-centered discretization.

The derivatives of the convective and viscous fluxes in Eq. (6.46) can be evaluated as discussed in the previous subsection. The ADI method is traditionally coupled to the central scheme with artificial dissipation. In such a case, formulation similar to that in Eq. (6.34) holds for each factor (of course, the linearization of the source term is included in one factor only). In order to obtain a robust and efficient scheme, it is necessary to include a linearization of the artificial dissipation terms in the implicit operator [42–44]. The linearization is in general simplified by treating the spectral radii and the dissipation coefficients as independent of \vec{W} . Hence, according to Eq. (4.48) the factor, for example, in the I -direction becomes

$$\left\{ \frac{\Omega}{\Delta t} \vec{I} + \frac{\partial [(\vec{F}_c^I - \vec{F}_v^I) \Delta S^I]_{i+1/2}}{\partial \vec{W}} + \frac{\partial [(\vec{F}_c^I - \vec{F}_v^I) \Delta S^I]_{i-1/2}}{\partial \vec{W}} \right. \\ \left. - \frac{\partial (\vec{D}_{IM}^I \Delta S^I)_{i+1/2}}{\partial \vec{W}} - \frac{\partial (\vec{D}_{IM}^I \Delta S^I)_{i-1/2}}{\partial \vec{W}} \right\}$$

with the implicit artificial dissipation term JST scheme (cf. Eq. (4.50))

$$\frac{\partial (\vec{D}_{IM}^I)_{i+1/2}}{\partial \vec{W}} \Delta \vec{W}^n \approx \hat{\Lambda}_{i+1/2}^S \left[(\epsilon_{IM}^{(2)})_{i+1/2} (\Delta \vec{W}_{i+1}^n - \Delta \vec{W}_i^n) \right. \\ \left. - (\epsilon_{IM}^{(4)})_{i+1/2} (\Delta \vec{W}_{i+2}^n - 3\Delta \vec{W}_{i+1}^n + 3\Delta \vec{W}_i^n - \Delta \vec{W}_{i-1}^n) \right]. \quad (6.47)$$

The implicit dissipation terms in other directions are defined in a similar way. It is also possible to retain only the second-order differences in the implicit operator [43, 44]. This reduces the matrix for each of the factors from block-pentadiagonal to block-tridiagonal form (as sketched in Fig. 6.1). However, as pointed out in Ref. [44], this restricts the stability of the scheme.

The inversion of the implicit operator in Eq. (6.46) proceeds in three steps (two steps in 2D), that is,

$$\begin{aligned} (\mathbf{D}^I + \mathbf{L}^I + \mathbf{U}^I) \Delta \vec{W}^{(1)} &= -\vec{R}^n, \\ (\mathbf{D}^J + \mathbf{L}^J + \mathbf{U}^J) \Delta \vec{W}^{(2)} &= \Delta \vec{W}^{(1)}, \\ (\mathbf{D}^K + \mathbf{L}^K + \mathbf{U}^K) \Delta \vec{W}^n &= \Delta \vec{W}^{(2)}, \end{aligned} \quad (6.48)$$

where **D** represents the diagonal, **L** the lower-diagonal, and **U** the upper-diagonal terms, respectively. Each step requires the inversion of a block-tridiagonal or a block-penta-diagonal matrix (if $\epsilon_{IM}^{(4)} > 0$ in Eq. (6.47)). This is done by a direct solution method. In order to reduce the numerical effort, Pulliam and Chaussee [45] suggested a diagonalized form of the ADI scheme. Herewith, the block matrices (composed of convective and viscous flux Jacobians) are transformed into diagonal matrices. Hence, only non-block tri- or pentadiagonal matrices have to be inverted, which results in significant savings of computational work and memory [43]. Although the diagonalization is valid strictly for the Euler equations only, it can be employed for viscous flows as well [43]. The linearization of the viscous fluxes (e.g., like in Eq. (6.45)) is then either dropped from the implicit operator, or it is approximated by the viscous eigenvalue (Eq. (6.19)).

The splitting of the implicit operator introduces what is called the *factorization error*. It is the difference between the implicit operator of the base scheme in Eq. (6.28) and the factorized operator. In the case of the ADI scheme, this error term is scaled by the factor $(\Delta t)^N$, where N denotes the number of space dimensions. This term causes the ADI scheme to loose its unconditional stability in 3D [46]. However, the stability is improved if the fourth-order differences of the artificial viscosity are included in the implicit operator [44]. In fact, the ADI scheme was successfully used for the solution

of various 3-D problems [43, 47]. An interesting implementation of the ADI scheme on multiblock grids, which treats the block boundaries implicitly, was presented by Rosenfeld and Yassour [48].

The time step Δt can be computed in the same way as presented in [Section 6.1.4](#), using Eq. (6.14). The optimal CFL number varies between 20 and 50, depending on the physics of the particular flow case.

6.2.4 Lower-upper symmetric Gauss-Seidel scheme

The implicit *lower-upper symmetric Gauss-Seidel* (LU-SGS) scheme, which is also called the *lower-upper symmetric successive overrelaxation* (LU-SSOR) scheme, became widely used because of its low numerical complexity and modest memory requirements, which are both comparable to an explicit multistage scheme. Furthermore, the LU-SGS scheme can be implemented easily on vector and parallel computers. It can also be used on structured as well as on unstructured grids.

The LU-SGS scheme has its origins in the work of Jameson and Turkel [49], who considered decompositions of the implicit operator into lower and upper diagonally dominant factors. The LU-SGS method itself was introduced by Yoon and Jameson [50–52] as a relaxation method for solving the unfactored implicit scheme in Eq. (6.28). It was further developed and applied to 3-D viscous flow fields by Rieger and Jameson [53]. Since then, various researchers applied the LU-SGS scheme to viscous flows on structured [31, 54, 56–63] and on unstructured grids [64–71]. The LU-SGS approach is often used for the simulation of chemically reacting flows [72–78].

The LU-SGS scheme employs a simplification of the first-order accurate flux-vector splitting approach due to Steger and Warming (see [Section 6.2.2](#)) for the linearization of the convective fluxes in Eq. (6.29). The linearization is always kept the same regardless of the discretization of the explicit operator. The LU-SGS scheme is further based on the factorization of the implicit operator in Eq. (6.28) into the following three parts

$$(\mathbf{D} + \mathbf{L}) \mathbf{D}^{-1} (\mathbf{D} + \mathbf{U}) \Delta \vec{W}^n = -\vec{R}_f^n. \quad (6.49)$$

The factors are constructed such that \mathbf{L} consists only of terms in the strictly lower triangular matrix, \mathbf{U} of terms in the strictly upper triangular matrix, and \mathbf{D} of diagonal terms. It is important to remark that the number of factors remains always the same independent of the number of space dimensions.

The system matrix of the LU-SGS scheme in Eq. (6.49) can be inverted in two steps—a forward and a backward sweep, that is,

$$\begin{aligned} (\mathbf{D} + \mathbf{L}) \Delta \vec{W}^{(1)} &= -\vec{R}_I^n, \\ (\mathbf{D} + \mathbf{U}) \Delta \vec{W}^n &= \mathbf{D} \Delta \vec{W}_I^{(1)} \end{aligned} \quad (6.50)$$

with $\vec{W}^{n+1} = \vec{W}^n + \Delta \vec{W}^n$. The operators \mathbf{L} , \mathbf{D} , and \mathbf{U} and also the inversion procedure differ between structured and unstructured grids in some aspects. Therefore, we shall discuss below each case separately.

LU-SGS on structured grids

On structured grids, the operators are defined as (see Refs. [50–52, 55, 73])

$$\begin{aligned}\mathbf{L} &= (\bar{A}^+ + \bar{A}_v)_{i-1} \Delta S_{i-1/2}^I + (\bar{A}^+ + \bar{A}_v)_{j-1} \Delta S_{j-1/2}^J \\ &\quad + (\bar{A}^+ + \bar{A}_v)_{k-1} \Delta S_{k-1/2}^K, \\ \mathbf{U} &= (\bar{A}^- - \bar{A}_v)_{i+1} \Delta S_{i+1/2}^I + (\bar{A}^- - \bar{A}_v)_{j+1} \Delta S_{j+1/2}^J \\ &\quad + (\bar{A}^- - \bar{A}_v)_{k+1} \Delta S_{k+1/2}^K, \\ \mathbf{D} &= \frac{\Omega}{\Delta t} \bar{I} + (\bar{A}^- - \bar{A}_v) \Delta S_{i-1/2}^I + (\bar{A}^- - \bar{A}_v) \Delta S_{j-1/2}^J \\ &\quad + (\bar{A}^- - \bar{A}_v) \Delta S_{k-1/2}^K + (\bar{A}^+ + \bar{A}_v) \Delta S_{i+1/2}^I \\ &\quad + (\bar{A}^+ + \bar{A}_v) \Delta S_{j+1/2}^J + (\bar{A}^+ + \bar{A}_v) \Delta S_{k+1/2}^K - \frac{\partial(\Omega \vec{Q})}{\partial \vec{W}}.\end{aligned}\tag{6.51}$$

For better readability, only those node indexes (or cell indexes in the case of a cell-centered scheme) are shown in Eq. (6.51), which differ from i, j, k . The superscripts i, j, k at ΔS indicate the direction in the computational space. The unit normal vectors in the positive/negative flux Jacobians \bar{A}^\pm and in the viscous flux Jacobians \bar{A}_v are evaluated at the same side of the control volume like the associated face areas ΔS . Note that the unit normal vectors are assumed to point outwards of the control volume. In contrast, in various references it is supposed that the unit normal vectors from opposite sides of the control volume point in the same direction.

The viscous flux Jacobians in Eq. (6.51) are computed either numerically, or are replaced by their TSL approximation, corresponding to Eq. (6.45). It is possible to apply the TSL approximation in all computational coordinates, regardless of the actual orientation of the boundary layer(s). A further simplification consists of substituting the viscous flux Jacobians by the viscous spectral radii (Eq. (6.19)), that is, $\bar{A}_v \Delta S \approx \hat{\Lambda}_v$, as suggested in [65].

The split convective flux Jacobians \bar{A}^\pm are constructed in such a way that the eigenvalues of the (+) matrices are all non-negative, and of the (-) matrices are all non-positive. In general, the matrices are defined as [49]

$$\bar{A}^\pm \Delta S = \frac{1}{2} (\bar{A}_c \Delta S \pm r_A \bar{I}), \quad r_A = \omega \hat{\Lambda}_c,\tag{6.52}$$

where \bar{A}_c stands for the convective flux Jacobian (Section A.9) and $\hat{\Lambda}_c$ represents the spectral radius of the convective flux Jacobian (given by Eq. (4.53) or (6.15)), respectively. Note the similarity between the above approximation (6.52) and Eq. (6.40), when the derivatives of \bar{A}_c are neglected. The factor ω in Eq. (6.52) represents an overrelaxation parameter. It also determines the amount of implicit dissipation and hence it influences the convergence properties of the scheme. The factor can be chosen in the range $1 < \omega \leq 2$. Higher values of ω increase the stability of the LU-SGS scheme, but may slow down the convergence to steady state. The definition of the Jacobians \bar{A}^\pm in Eq. (6.52) ensures a diagonally dominant system matrix, which is very important for the efficiency and robustness of the iterative inversion procedure (6.50).

The splitting according to Eq. (6.52) allows together with averaged face vectors for a simplified evaluation of the diagonal operator \mathbf{D}

$$\begin{aligned}\mathbf{D} = & \left[\frac{\Omega}{\Delta t} + \omega \left(\hat{\Lambda}_c^I + \hat{\Lambda}_c^J + \hat{\Lambda}_c^K \right) \right] \bar{I} \\ & + 2 \left(\bar{A}_v^I \Delta S^I + \bar{A}_v^J \Delta S^J + \bar{A}_v^K \Delta S^K \right) - \frac{\partial (\Omega \vec{Q})}{\partial \vec{W}}.\end{aligned}\quad (6.53)$$

The spectral radii of the convective flux Jacobians $\hat{\Lambda}_c$ are given in Eq. (6.15). The face areas and normal vectors are averaged in the respective I -, J -, or K -direction according to Eq. (6.16). As we shall see immediately, this approximation helps to reduce the operation count and the memory requirements significantly.

A distinguishing feature of the LU-SGS method is how the forward and the backward sweep in Eq. (6.50) are carried out. In 2D, the sweeps are accomplished along diagonal lines $(i + j) = \text{const.}$ in computational space. This is depicted in Fig. 6.5 for the forward sweep (first line of Eq. (6.50)). In this way, the off-diagonal terms involved in the \mathbf{L} and the \mathbf{U} operator become known from the previous part of a sweep (denoted by crosses in Fig. 6.5). In 3D, the implicit operator is inverted on $i + j + k = \text{const.}$ planes, as sketched in Fig. 6.6. Hence, the LU-SGS scheme can be written as

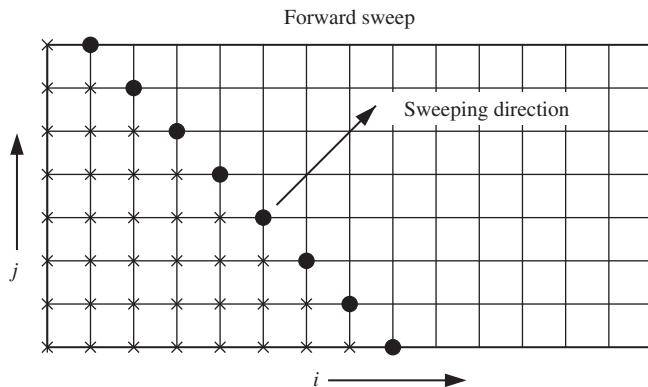


Figure 6.5 Sweeping direction of the LU-SGS scheme in computational space: • denotes where the operator \mathbf{D} is currently inverted (line $i + j = \text{const.}$); × denotes the already updated values of \mathbf{L} .

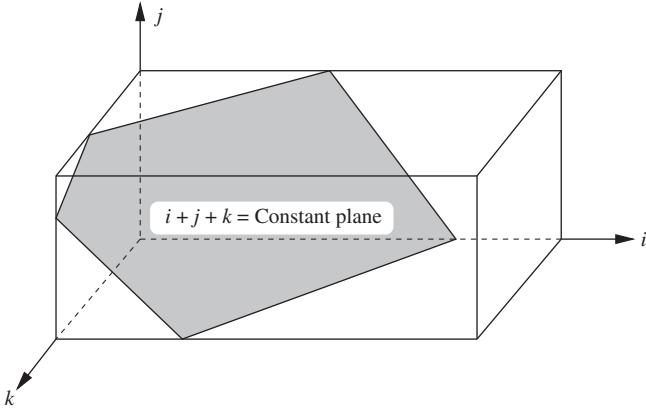


Figure 6.6 Diagonal plane of sweep in computational space for the implicit LU-SGS scheme in 3D.

$$\begin{aligned} \mathbf{D} \Delta \vec{W}_{i,j,k}^{(1)} &= -\vec{R}_{i,j,k}^n - \mathbf{L} \Delta \vec{W}^{(1)}, \\ \mathbf{D} \Delta \vec{W}_{i,j,k}^n &= \mathbf{D} \Delta \vec{W}_{i,j,k}^{(1)} - \mathbf{U} \Delta \vec{W}^n. \end{aligned} \quad (6.54)$$

As we can see from Eq. (6.54), the only term which needs to be inverted is the diagonal term \mathbf{D} . Thus, the LU-SGS methodology transforms the inversion of a sparse banded matrix into the inversion of a block-diagonal matrix. Furthermore, if the viscous flux Jacobians in Eq. (6.53) are approximated by the viscous spectral radii, the operator \mathbf{D} becomes a diagonal matrix (except for the source term). Hence, the LU-SGS scheme requires a very small computational effort as compared to other implicit schemes (e.g., the ADI scheme discussed previously). Furthermore, the inversion of the diagonal operator can be carried out independently for each node (cell) of the diagonal plane, which makes the scheme easy to vectorize. The indexes of the nodes/cells on the diagonal planes can be obtained with the following pseudo-code [79]:

```

DO plane = 1, nplanes
  DO k = 1, kmax
    DO j = 1, jmax
      DO i = 1, imax
        IF (i+j+k = plane+2) store indexes
      ENDDO
    ENDDO
  ENDDO
ENDDO

```

The number of diagonal planes is: $nplanes = imax + jmax + kmax - 2$. Obviously, the above code can be optimized for higher computational efficiency.

In order to avoid explicit evaluation and storage of the convective flux Jacobians in \mathbf{L} and \mathbf{U} , the products $\bar{A}^{\pm} \Delta \vec{W}^n$ can be substituted by Taylor series expansion of the fluxes [53]. Using Eq. (6.52), we can write

$$(\bar{A}^{\pm} \Delta S) \Delta \vec{W}^n \approx \frac{1}{2} (\Delta \vec{F}_c \Delta S \pm r_A \bar{I} \Delta \vec{W}^n) \quad (6.55)$$

with the update of the convective fluxes

$$\Delta \vec{F}_c = \vec{F}_c^{n+1} - \vec{F}_c^n. \quad (6.56)$$

The simplification given by Eq. (6.55) is possible due to the sweeping along diagonal planes, since \vec{F}_c^{n+1} is then known. This leads to a further significant decrease of the numerical effort of the LU-SGS scheme.

The time step Δt can be computed in the same way as presented in Section 6.1.4, using Eq. (6.14). However, it should be noted that the implicit LU-SGS scheme in Eq. (6.49) represents an approximate Newton iteration in the case of $\Delta t \rightarrow \infty$ as stated by Rieger and Jameson [53]. Thus, in general, CFL numbers of the order of 10^4 to 10^6 are used in practice for stationary flows. The convergence is then controlled by the overrelaxation parameter ω . For the simulation of unsteady flows, we may employ the formulation presented below in Section 6.3. Another possibility is to use the modified version of the LU-SGS scheme described in Ref. [80].

LU-SGS on unstructured grids

Here, for a median-dual scheme the operators read [65–67]

$$\begin{aligned} \mathbf{L} &= \sum_{j \in L(i)} \left[\bar{A}_j^+ + (\bar{A}_v)_j \right] \Delta S_{ij}, \\ \mathbf{U} &= \sum_{j \in U(i)} \left[\bar{A}_j^- - (\bar{A}_v)_j \right] \Delta S_{ij}, \\ \mathbf{D} &= \frac{\Omega_i}{\Delta t_i} \bar{I} + \frac{\omega}{2} (\hat{\Lambda}_c)_i + \sum_{j=1}^{N_F} (\bar{A}_v)_i \Delta S_{ij} - \frac{\partial(\Omega_i \vec{Q}_i)}{\partial \vec{W}}. \end{aligned} \quad (6.57)$$

where $L(i)$, and $U(i)$ denote the nearest neighbors of node i which belong to the lower (upper) matrix, ΔS_{ij} represents the face area associated with the edge ij (see Fig. 5.9), and N_F stands for the number of faces of the control volume Ω_i , respectively. The spectral radius of the convective fluxes $(\hat{\Lambda}_c)_i$ is computed by Eq. (6.21). The viscous flux Jacobian \bar{A}_v can be again approximated by its spectral radius [65]. In this case, the diagonal operator becomes

$$\mathbf{D} = \frac{\Omega_i}{\Delta t_i} \bar{I} + \frac{\omega}{2} (\hat{\Lambda}_c)_i + (\hat{\Lambda}_v)_i - \frac{\partial(\Omega_i \vec{Q}_i)}{\partial \vec{W}}, \quad (6.58)$$

where $(\hat{\Lambda}_v)_i$ is evaluated according to Eq. (6.21). Formulas similar to Eq. (6.57) and (6.58) can be obtained in the case of the cell-centered scheme. The major difference

is that the summation in the \mathbf{L} and the \mathbf{U} operator is conducted over faces of the cell instead of incident edges.

The sets $L(i)$ and $U(i)$ in Eq. (6.57) should fulfill the same function as the diagonal planes on structured grids. For this reason, it is necessary to arrange the nodes (cells) into layers such that [65]:

- nodes i (cells I) of a current layer have connections to layers with previously updated flow variables—otherwise the LU-SGS scheme degenerates to a Jacobi iteration,
- nodes (cells) in a layer are not connected to each other—otherwise the scheme could not be vectorized.

The layers can be generated for the median-dual scheme with a procedure described in Ref. [65]. Approaches for the cell-centered scheme were suggested in Refs. [81, 82].

An appropriate definition of the sets $L(i)$ and $U(i)$ allows for the following two-step inversion procedure [65–67]

$$\begin{aligned}\mathbf{D} \Delta \vec{W}_i^{(1)} &= -\vec{R}_i^n - \sum_{j \in L(i)} \frac{1}{2} \left[(\Delta F_c^{(1)})_j \Delta S_{ij} + (r_A^*)_j \bar{I} \Delta \vec{W}_j^{(1)} \right], \\ \mathbf{D} \Delta \vec{W}_i^n &= \mathbf{D} \Delta \vec{W}_i^{(1)} - \sum_{j \in U(i)} \frac{1}{2} \left[(\Delta F_c^n)_j \Delta S_{ij} - (r_A^*)_j \bar{I} \Delta \vec{W}_j^n \right],\end{aligned}\quad (6.59)$$

where the viscous Jacobians were approximated by their spectral radii and where the positive/negative Jacobians were linearized according to Eq. (6.55). Furthermore, the factor $(r_A^*)_j$ is defined as

$$\begin{aligned}(r_A^*)_j &= \omega (|\vec{v}_j \cdot \vec{n}_{ij}| + c_j) \Delta S_{ij} \\ &\quad + \frac{\Delta S_{i,j}}{\|\vec{r}_j - \vec{r}_i\|_2} \left[\max \left(\frac{4}{3\rho_j}, \frac{\gamma_j}{\rho_j} \right) \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T} \right)_j \right]\end{aligned}\quad (6.60)$$

with $\|\vec{r}_j - \vec{r}_i\|_2$ being the length of the edge ij .

The time step can be computed in the same way as for the explicit scheme (Eq. (6.20)). However, the viscous eigenvalue should be omitted, since the viscous terms are already contained in the implicit operator and hence do not reduce the time step as in the case of an explicit scheme. The CFL number can be chosen in the range from 10^4 to 10^6 for steady flows. The convergence speed and the robustness of the LU-SGS scheme can then be tuned using the overrelaxation parameter ω .

6.2.5 Newton-Krylov method

First of all, let us rewrite the implicit scheme given by Eq. (6.28) as

$$\bar{J} \Delta \vec{W}^n = -\vec{R}^n,\quad (6.61)$$

where \bar{J} represents the implicit operator (system matrix). As we saw already, \bar{J} constitutes a large, sparse, and generally non-symmetric matrix. In the previous sections, we

discussed two methods that decompose \bar{J} into several factors which can be each inverted more easily than \bar{J} itself. However, due to the factorization error (and approximate linearization of \vec{R}^{n+1}), only a linear convergence to steady state can be achieved. In order to obtain the quadratic convergence of Newton's method for the solution of non-linear equations, four conditions must be fulfilled:

- the linearization of the residual must be exact,
- \bar{J} must be accurately inverted,
- the time step has to be $\Delta t \rightarrow \infty$,
- initial solution must be, in some sense, close to the final solution.

Obviously, the main obstacles that have to be overcome are the linearization and the inversion of the full system matrix.

A particularly suitable class of iterative techniques for the solution of large linear equation systems are the so-called *Krylov-subspace* methods. Several were proposed for the inversion of matrices which arise in CFD. Examples are the *conjugate gradient squared* (CGS) method [83], the *bi-conjugate gradient stabilized* (Bi-CGSTAB) scheme [84], or the *transpose-free quasi-minimum residual* (TFQMR) approach [85]. However, the most successful Krylov subspace method became the *generalized minimal residual* (GMRES) technique, which was originally suggested by Saad and Schulz [86, 87]. Since then, the GMRES method was improved and augmented by several researchers [88–91]. Because of its popularity, we shall focus on the GMRES approach in the following. Nevertheless, most of what we shall discuss also applies to the other Krylov subspace methods.

GMRES method

As we mentioned in Section 3.2.2 (see also Section A.12), the GMRES method minimizes the norm of the global residual, that is, $\|\bar{J} \Delta \vec{W}^n + \vec{R}^n\|$ over a set of m orthonormal vectors (search directions), which span the Krylov subspace \mathcal{K}_m given by Eq. (3.10). The GMRES algorithm can be summarized as follows:

1. guess a starting solution \vec{W}_0^n and evaluate the initial residual vector $\vec{r}_0 = \bar{J} \Delta \vec{W}_0^n + \vec{R}^n$,
2. generate the m search directions (by Gram-Schmidt orthogonalization),
3. solve the minimization problem,
4. form an approximate solution of Eq. (6.61) as $\Delta \vec{W}^n = \Delta \vec{W}_0^n + \vec{\gamma}_m$.

Since the memory requirements increase linearly with the number of search directions, m is restricted to values between 10 and 40 in practice. This might not be sufficient for a converged solution $\Delta \vec{W}^n$. Thus, the GMRES method has to be restarted, that is, we set $\Delta \vec{W}_0^n = \Delta \vec{W}^n$, compute \vec{r}_0 and proceed with step 2. As pointed out in Ref. [92], instead of working with a constant number of search directions, m should be reduced if the norm of the global residual drops below a specified tolerance. In this way, a large number of operations can be saved in later stages of the Newton iteration.

Computation of the flux Jacobian

GMRES and other Krylov subspace methods allow us to circumvent an explicit computation and storage of the flux Jacobian $\partial \vec{R} / \partial \vec{W}$. The idea is based on the observation that the methods rely only on matrix–vector products of the form $\bar{J} \Delta \vec{W}^n$ and do not need the matrix \bar{J} explicitly. The product of the flux Jacobian with the solution update can be approximated by a simple finite difference as

$$\frac{\partial \vec{R}}{\partial \vec{W}} \Delta \vec{W}^n \approx \frac{\vec{R}(\vec{W}^n + h \Delta \vec{W}^n) - \vec{R}(\vec{W}^n)}{h} \quad (6.62)$$

which requires only two evaluations of the residual. The step size h has to be chosen with some care, in order to minimize the numerical error [36]. One particularly suitable formulation reads [93]

$$h = \frac{\sqrt{\epsilon}}{\|\Delta \vec{W}^n\|_2^2} \max \{ |d|, \text{typ} [\vec{W}^n \cdot |\Delta \vec{W}^n|] \} \text{ sign}(d), \quad (6.63)$$

where ϵ denotes the machine accuracy, d the scalar product $\vec{W}^n \cdot \Delta \vec{W}^n$, $|\Delta \vec{W}^n|$ is the vector $\Delta \vec{W}^n$ with all elements set to their absolute values, and finally $\text{typ } U$ represents a typical size of U . Apart from saving memory and operations, there is an even more important advantage of the finite-difference approximation. Namely, numerically accurate linearization of a high-order residual \vec{R}^n (including boundary conditions, limiters, source terms, etc.) can be easily achieved. Thus, the quadratic convergence of Newton's scheme can be realized at moderate costs. For this reason, we speak of such a scheme as of *Newton-Krylov* approach [93–100].

Preconditioning

The efficiency of Krylov-subspace methods depends strongly on a good preconditioner. Its purpose is to cluster the eigenvalues of the system matrix \bar{J} around unity. Thus, instead of Eq. (6.61), the left- or right-preconditioned system according to Eq. (3.11) is solved. Using Eqs. (6.61) and (6.62) together with the condition $\Delta t \rightarrow \infty$, the Newton-Krylov method becomes

$$\bar{P}_L \frac{\vec{R}(\vec{W}^n + h \Delta \vec{W}^n) - \vec{R}(\vec{W}^n)}{h} = -\bar{P}_L \vec{R}^n \quad (6.64)$$

with left preconditioning, and

$$\begin{aligned} \frac{\vec{R}(\vec{W}^n + h \bar{P}_R \Delta \vec{W}^*) - \vec{R}(\vec{W}^n)}{h} &= -\vec{R}^n, \\ \bar{P}_R^{-1} \Delta \vec{W}^n &= \Delta \vec{W}^* \end{aligned} \quad (6.65)$$

in the case of right preconditioning, respectively. The main difference between the two preconditioning methodologies is that left preconditioning scales the residual \vec{R}^n whereas

right preconditioning does not. This has to be kept in mind when the convergence of the Krylov method is monitored.

Obviously, the preconditioner should be as close as possible to the inverse of the system matrix ($\vec{P}_{L,R} \approx \vec{J}^{-1}$). But on the other hand, it should be invertible with low numerical effort. Therefore, we have to find an optimal trade-off between the convergence speed of the Krylov method and the time spent for inverting the preconditioning matrix. One of the most successful preconditioners is the *incomplete lower upper* factorization method [101, 102] with varying level of fill-in (mostly with zero, designated as ILU(0)). The ILU preconditioner is especially efficient in the case of viscous, turbulent flows, that is, for stiff equations [103]. In order to obtain a good performance on unstructured grids, it is necessary to reorder the elements of the system matrix such that the bandwidth is reduced. We already discussed the RCM renumbering strategy [24, 25] in Section 6.2.1.

A serious disadvantage of the ILU preconditioning scheme is that elements of the matrix \vec{J} have to be computed (see Section 6.2.2) and stored. Therefore, some authors suggested to employ the LU-SGS scheme as a preconditioner [104, 105]. Hence, when Eq. (6.62) is applied, the formation and storage of \vec{J} is avoided completely. However, it was demonstrated in Ref. [103] on behalf of several 2-D cases that the GMRES method with LU-SGS is inferior to GMRES combined with ILU(0) in terms of the CPU-time. Nevertheless, the LU-SGS scheme (best when coupled with multigrid) still represents an attractive alternative, particularly in 3D.

Start-up problem

The time step of the implicit Newton-Krylov method is infinitely large. However, it is advisable to use small time steps at the beginning of the Newton iteration process. The reason is that the flow solution is in general far from the steady state at the beginning of the solution process, that is, the root of the nonlinear equation

$$\vec{R}(\vec{W}) = 0,$$

and this may cause a breakdown of the Newton iteration. One possible remedy is the so-called *switched evolution relaxation* technique [106]. Here, the term $\Omega/\Delta t$ is retained in Eq. (6.61). The time step is evaluated in the same way as presented for the explicit scheme (Eq. (6.14) or Eq. (6.20) without the viscous eigenvalue). The CFL number σ is increased starting from a small initial value correspondingly to the reduction of the 2-norm of the residual, that is,

$$\sigma^{n+1} = \sigma^n \frac{\|\vec{R}^{n-1}\|_2}{\|\vec{R}^n\|_2}. \quad (6.66)$$

Hence, the convergence of the iteration procedure (6.61) will be at first linear, but it approaches the quadratic convergence of Newton's method for large CFL numbers. A further effect of the time term is the increased diagonal dominance of \bar{J} (inversely proportional to $\Omega/\Delta t$), which will help to stabilize the iterative process. In Ref. [95], it was suggested to clip σ^{n+1} in Eq. (6.66) such that it increases by maximum factor of two and decreases less than factor of ten.

A further approach which can be used to overcome the start-up problems of Newton's method consist of grid sequencing, where the initial solution is obtained on a sequence of coarser grids and interpolated onto finer grids. Another possibility is to use a numerically cheap but robust iteration scheme for the initial guess. For example, we could start with a multigrid scheme driven by the LU-SGS method and then switch to GMRES with LU-SGS as a preconditioner. This might be particularly interesting for viscous turbulent flows, where the convergence of a multigrid scheme usually slows down after the initial phase. However, the global flow solution is then already close to the steady state.

6.2.6 Implicit Runge-Kutta schemes

A general m -stage scheme for the solution of the system of ordinary differential equations in time (6.1) can be written as

$$\vec{W}_I^{n+1} = \vec{W}_I^n - \frac{\Delta t}{(\Omega \bar{M})_I} \sum_{k=1}^m b_k \vec{R}_I^{(k)}, \quad (6.67)$$

where each stage residual $\vec{R}_I^{(k)}$ is a function of the time level and of a weighted sum of other intermediate residuals, that is,

$$\begin{aligned} \vec{R}_I^{(k)} &= \vec{R}_I \left(t^n + c_k \Delta t, \vec{W}_I^n - \frac{\Delta t}{(\Omega \bar{M})_I} \sum_{l=1}^m a_{kl} \vec{R}_I^{(l)} \right) \\ &= \vec{R}_I \left(t^n + c_k \Delta t, \vec{W}_I^{(k)} \right). \end{aligned} \quad (6.68)$$

The coefficients a_{kl} , b_k and c_k can be conveniently displayed in a form known as Butcher tableau [15]

$$\begin{array}{c|ccccc} c_1 & a_{11} & a_{12} & \cdots & a_{1m} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_m & a_{m1} & a_{m2} & \cdots & a_{mm} \\ \hline & b_1 & b_2 & \cdots & b_m \end{array} \quad (6.69)$$

The coefficients c_k define the intermediate time levels, a_{kl} represent weights for the k th stage, and b_k are the weights of the final stage, respectively. The coefficients determine the accuracy, stability and also the efficiency of the particular Runge-Kutta scheme. Depending on their values, various Runge-Kutta schemes can be composed. For example, setting $a_{kl} = 0$ for all $l \geq k$ leads to purely explicit methods, like the classical 4th-order Runge-Kutta scheme given by the tableau

$$\begin{array}{c|cccc} & 0 & 0 & 0 & 0 \\ \hline 0 & & & & \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \quad (6.70)$$

On the other hand, *implicit Runge-Kutta* schemes are obtained if $a_{kl} \neq 0$ for at least some $l \geq k$, that is, along the diagonal or in the upper triangular matrix of Table (6.69). In such a case, the intermediate residual at stage k , that is, $\vec{R}_I^{(k)}$ in Eq. (6.68) depends implicitly also on the solution at the same stage, and possibly on solutions at future stages. Implicit Runge-Kutta schemes employed for CFD applications normally avoid coupling to solutions at later than the current stage. Such schemes, where a_{kl} are nonzero only in the lower triangular matrix, are termed *diagonally implicit Runge-Kutta* (DIRK) schemes. If additionally all diagonal coefficients a_{kk} are identical, the scheme is called *singly diagonally implicit Runge-Kutta* (SDIRK) scheme.

The coefficients have to meet certain conditions. First of all, $c_k = \sum_l a_{kl}$ for a physically sound scheme. Further, there are the following order conditions

$$\sum_{k=1}^m b_k = 1 \quad (6.71)$$

for first-order accuracy,

$$\sum_{k=1}^m b_k c_k = \frac{1}{2} \quad (6.72)$$

for second-order accuracy,

$$\begin{aligned} \sum_{k=1}^m b_k c_k^2 &= \frac{1}{3} \\ \sum_{k=1}^m \sum_{l=1}^m b_k a_{kl} c_l &= \frac{1}{6} \end{aligned} \quad (6.73)$$

for third-order accuracy, and

$$\begin{aligned}
\sum_{k=1}^m b_k c_k^3 &= \frac{1}{4} \\
\sum_{k=1}^m \sum_{l=1}^m b_k c_k a_{kl} c_l &= \frac{1}{8} \\
\sum_{k=1}^m \sum_{l=1}^m b_k a_{kl} c_l^2 &= \frac{1}{12} \\
\sum_{k=1}^m \sum_{l=1}^m \sum_{n=1}^m b_k a_{kl} a_{ln} c_n &= \frac{1}{24}
\end{aligned} \tag{6.74}$$

for fourth-order accuracy. Meeting the above conditions, the coefficients can be optimized to achieve the desired accuracy, stability and damping properties.

Within implicit Runge-Kutta schemes, the intermediate residual $\vec{R}_I^{(k)}$ in Eq. (6.68) depends implicitly on residual(s) yet to be computed (i.e., on $\vec{R}_I^{(l)}$ for $l \geq k$). Hence, each stage requires the solution of a set of nonlinear equations. This is usually accomplished by the means of Newton's method. Let us for illustration consider a diagonally implicit Runge-Kutta scheme. The intermediate solution at stage k can then be written as

$$\vec{W}_I^{(k)} = \vec{W}_I^n - \frac{\Delta t}{(\Omega \bar{M})_I} \left[\sum_{l=1}^{k-1} a_{kl} \vec{R}_I^{(l)} + a_{kk} \vec{R}_I^{(k)} \right]. \tag{6.75}$$

The implicit term $\vec{R}_I^{(k)}$ in Eq. (6.75), which is a function of $\vec{W}_I^{(k)}$ and of $(t^n + c_k \Delta t)$, can be linearized using Eq. (6.27) as follows

$$\vec{R}_I^{(k)} \approx \vec{R}_I^n + \bar{J}_I \Delta \vec{W}^{(k)} \tag{6.76}$$

with $\Delta \vec{W}^{(k)} = \vec{W}^{(k)} - \vec{W}^n$ and $\bar{J} = \partial \vec{R} / \partial \vec{W}$ being the Jacobian matrix. Inserted into Eq. (6.75), we obtain for the solution at stage k the expression

$$\left[\frac{(\Omega \bar{M})_I}{\Delta t} + a_{kk} \bar{J}_I \right] \Delta \vec{W}^{(k)} = - \sum_{l=1}^{k-1} a_{kl} \vec{R}_I^{(l)} - a_{kk} \vec{R}_I^n. \tag{6.77}$$

This is formally equivalent to Eq. (6.28) and can be solved by any of the implicit methods from the previous subsections.

Implicit Runge-Kutta schemes are employed in cases, where the differential equations (6.1) are stiff and have to be advanced in time accurately. The stiffness can be caused, for example, by turbulence modeling, highly stretched grid cells, or by real-gas

effects. An idea developed by Cooper and Sayfy [107, 108] consists of splitting the equations into stiff and nonstiff terms. The stiff terms are then treated implicitly, whereas at the same time the nonstiff terms are integrated explicitly. Such multistage methods are known as *additive Runge-Kutta* schemes. Depending on the case, such splitting can result in a much simpler Jacobian \bar{J} in Eq. (6.77) and hence in a considerable reduction of the numerical effort, as compared to implicit treatment of all terms. Kennedy and Carpenter devised in [109] optimized Runge-Kutta approaches for this purpose. The coefficients for the coupled explicit-implicit schemes are given in Tables 6.3 and in 6.4. Both methods have six stages and are 4th-order accurate in time. Note that both schemes have identical coefficients b_k and c_k in order to facilitate the coupling. The second set of coefficients \hat{b}_k (final stage weights) defines in both cases a third-order scheme—this is called *embedded scheme*—which is used to estimate the error of the integration and with this to adjust the time step. The temporal error is computed as

$$\epsilon_I = \frac{\Delta t}{(\Omega \bar{M})_I} \sum_{k=1}^m (b_k - \hat{b}_k) \vec{R}_I^{(k)}. \quad (6.78)$$

Coefficients of other diagonally implicit Runge-Kutta schemes, which might be of particular interest, are gathered in the Tables 6.5–6.7. For examples of implementations and results obtained, the reader is referred to [113–121].

Table 6.3 Coefficients of explicit, six-stage, fourth- and third-order accurate (\hat{b}_k) Runge-Kutta scheme [109]

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0
$\frac{83}{250}$	$\frac{13861}{62500}$	$\frac{6889}{62500}$	0	0	0	0
$\frac{31}{50}$	$-\frac{116923316275}{2393684061468}$	$-\frac{2731218467317}{15368042101831}$	$\frac{9408046702089}{11113171139209}$	0	0	0
$\frac{17}{20}$	$-\frac{451086348788}{2902428689909}$	$-\frac{2682348792572}{7519795681897}$	$\frac{1266286775082}{11960479115383}$	$\frac{3355817975965}{11060851509271}$	0	0
1	$\frac{647845179188}{3216320057751}$	$\frac{73281519250}{8382639484533}$	$\frac{552539513391}{3454668386233}$	$\frac{3354512671639}{8306763924573}$	$\frac{4040}{17871}$	0
b_k	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
\hat{b}_k	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$-\frac{3700637}{11593932}$	$\frac{61727}{225920}$

Table 6.4 Coefficients of singly diagonally implicit, six-stage, fourth- and third-order accurate (\hat{b}_k) Runge-Kutta scheme [109]

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
$\frac{83}{250}$	$\frac{8611}{62500}$	$-\frac{1743}{31250}$	$\frac{1}{4}$	0	0	0
$\frac{31}{50}$	$\frac{5012029}{34652500}$	$-\frac{654441}{2922500}$	$\frac{174375}{388108}$	$\frac{1}{4}$	0	0
$\frac{17}{20}$	$\frac{15267082809}{155376265600}$	$-\frac{71443401}{120774400}$	$\frac{730878875}{902184768}$	$\frac{2285395}{8070912}$	$\frac{1}{4}$	0
1	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
b_k	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
\hat{b}_k	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$-\frac{3700637}{11593932}$	$\frac{61727}{225920}$

Table 6.5 Coefficients of singly diagonally implicit, three-stage, 4th-order accurate Runge-Kutta scheme [110]. Parameter $a = (2/\sqrt{3}) \cos(\pi/18)$. The scheme has good damping properties, also at large time steps

$(1+a)/2$	$(1+a)/2$	0	0
$1/2$	$-a/2$	$(1+a)/2$	0
$(1-a)/2$	$1+a$	$-(1+2a)$	$(1+a)/2$
	$1/(6a^2)$	$1 - 1/(3a^2)$	$1/(6a^2)$

Table 6.6 Coefficients of diagonally implicit, three-stage, 4th-order accurate Runge-Kutta scheme [111]. It is optimized to achieve low dispersion and dissipation errors even at large time steps. However, the scheme is physically inconsistent because of $c_k \neq \sum_I a_{ki}$

0.257820901066211	0.377847764031163	0.0	0.0
0.434296446908075	0.385232756462588	0.461548399939329	0.0
0.758519768667167	0.675724855841358	-0.061710969841169	0.241480233100410
	0.750869573741408	-0.362218781852651	0.611349208111243

Table 6.7 Optimized coefficients of diagonally implicit, three-stage, third-order accurate Runge-Kutta scheme [112]

1.559910257360900	1.559910257360900	0.0	0.0
14.78349948371908	1.656464402419667	13.12703508129941	0.0
14.37813978673026	1.658169529369301	1.672660289271657	11.04730996808931
	1.169960021844266	2.759712723699827	-2.929672745544100

6.3 METHODOLOGIES FOR UNSTEADY FLOWS

The simulation of unsteady flow phenomena is nowadays very important in many engineering disciplines. Examples are the interaction between stationary and rotating parts in turbomachinery, piston engines, fluid-structure interaction, helicopter aerodynamics, aeroacoustics, DNS or LES of turbulent flows, detonations, etc. Clearly, the simulation has to be conducted efficiently and with accuracy adequate for the problem being solved.

Explicit schemes represent the best choice for certain unsteady applications when the time scales are comparable to the spatial scales divided by the eigenvalues, that is, when the *CFL* number dictated by the physics is of the order of unity. This is for example the case in aeroacoustics, DNS, and LES. For such applications, explicit Runge-Kutta schemes are quite popular. Since the global physical phenomena evolve much slower than the solution changes locally in these applications, it is necessary to integrate over a long period of physical time. In order to do this accurately, the temporal resolution of the explicit scheme has to be of third or higher order. This requires the use of Runge-Kutta methods different from those we presented in [Section 6.1](#) (see, e.g., [122, 123]).

In other cases, where the physical time scales are large in comparison to the spatial scales divided by the eigenvalues (e.g., flutter and rotor-stator interaction), the *CFL* number can be chosen in the order of several hundreds or even thousands without impairing the accuracy of the simulation. Obviously, in such cases an implicit scheme is more appropriate. In the following, we shall discuss a particular technique known as the *dual time-stepping* approach, which is quite often employed for unsteady flows. Comparisons of this methodology with implicit Runge-Kutta schemes of [Section 6.2.6](#) can be found in Refs. [124–126].

The dual time-stepping approach is based on the second-order time accurate version of the basic non-linear scheme in Eq. (6.2). For this purpose we set $\beta = 1$ and $\omega = 1/2$ in Eq. (6.2). Hence, we obtain

$$\frac{3(\Omega \tilde{M})_I^{n+1} \vec{W}_I^{n+1} - 4(\Omega \tilde{M})_I^n \vec{W}_I^n + (\Omega \tilde{M})_I^{n-1} \vec{W}_I^{n-1}}{2 \Delta t} = -\vec{R}_I^{n+1}, \quad (6.79)$$

where Δt denotes the **global** physical time step and \tilde{M} the mass matrix, respectively. Equation (6.79) constitutes a 3-point backward-difference (backward Euler) approximation of the time derivative in Eq. (6.1). In order to solve the system of

non-linear equations given by Eq. (6.79), we can use either Newton's method or a time-stepping methodology. The latter can be written as

$$\frac{\partial}{\partial t^*} \left(\Omega_I^{n+1} \vec{W}_I^* \right) = -\vec{R}_I^*(\vec{W}^*), \quad (6.80)$$

where \vec{W}^* is the approximation of \vec{W}^{n+1} and t^* denotes a pseudo-time variable. Note that there is no mass matrix in the time derivative. The **unsteady** residual is defined as

$$\vec{R}_I^*(\vec{W}^*) = \vec{R}_I(\vec{W}^*) + \frac{3}{2 \Delta t} (\Omega \bar{M})_I^{n+1} \vec{W}_I^* - \vec{Q}_I^*. \quad (6.81)$$

All terms which are constant during the time-stepping in Eq. (6.80) are gathered in a source term, that is,

$$\vec{Q}_I^* = \frac{2}{\Delta t} (\Omega \bar{M})_I^n \vec{W}_I^n - \frac{1}{2 \Delta t} (\Omega \bar{M})_I^{n-1} \vec{W}_I^{n-1}. \quad (6.82)$$

In the case of moving and/or deforming grids, the new size of the control volume, that is, Ω^{n+1} in Eq. (6.80) has to satisfy the Geometric Conservation Law (see Section A.5 for details and references).

The stationary solution of Eq. (6.80) corresponds to the flow variables at the new time level, that is, $\vec{W}^* = \vec{W}^{n+1}$. Since $\vec{R}_I^* = 0$ at steady state in pseudo time, Eq. (6.79) is fulfilled. Any of the previously presented explicit or implicit time-marching schemes can be employed for the solution of the system of equations (6.80) in the pseudo time. In the following, we shall discuss the implementation of the dual time-stepping approach for explicit multistage and implicit schemes.

6.3.1 Dual time-stepping for explicit multistage schemes

Jameson [127] first implemented the dual-time methodology using an explicit multistage scheme accelerated by local time-stepping and multigrid. The significant advantage of this approach is that the physical time step is not restricted as usual in explicit methods. It can be chosen based solely on the flow physics. On the other hand, additional storage is needed only for the source term \vec{Q}^* , which makes the approach very attractive. An m -stage explicit scheme for the solution of the pseudo-time problem (6.80) reads

$$\begin{aligned} \vec{W}_I^{(0)} &= (\vec{W}_I^*)^l, \\ \vec{W}_I^{(1)} &= \vec{W}_I^{(0)} - \frac{\alpha_1 \Delta t_I^*}{\Omega_I^{n+1}} \vec{R}_I^*(\vec{W}^{(0)}), \\ \vec{W}_I^{(2)} &= \vec{W}_I^{(0)} - \frac{\alpha_2 \Delta t_I^*}{\Omega_I^{n+1}} \vec{R}_I^*(\vec{W}^{(1)}), \\ &\vdots \\ (\vec{W}_I^*)^{l+1} &= \vec{W}_I^{(0)} - \frac{\alpha_m \Delta t_I^*}{\Omega_I^{n+1}} \vec{R}_I^*(\vec{W}^{(m-1)}), \end{aligned} \quad (6.83)$$

where l denotes the actual and $(l+1)$ the new pseudo-time level, respectively. The time-marching process is started either with $(\vec{W}_I^*)^l = \vec{W}_I^n$ or with a value extrapolated from previous physical time steps, for example, Ref. [128]

$$(\vec{W}_I^*)^l = \vec{W}_I^n + \frac{3\vec{W}_I^n - 4\vec{W}_I^{n-1} + \vec{W}_I^{n-2}}{2}. \quad (6.84)$$

It is continued until $(\vec{W}_I^*)^{l+1}$ approximates \vec{W}_I^{n+1} with sufficient accuracy (usually when the residual \vec{R}_I^* was reduced by two or three orders of magnitude). After that, the next **physical** time step is conducted. The pseudo time step Δt^* is computed in the same way that we saw in Section 6.1.4.

Arnone et al. [128] pointed out that the multistage scheme (6.83) becomes unstable when the physical time step Δt is of the order of the pseudo time step Δt^* or smaller. Melson et al. [129] demonstrated that the instability is caused by the term

$$\frac{3}{2\Delta t}(\Omega\bar{M})_I^{n+1}\vec{W}_I^*$$

in Eq. (6.81), which becomes significant for small Δt . They suggested an implicit treatment of this term. Thus, we have to modify the multistage scheme in Eq. (6.83) such that the k th stage becomes [129]

$$\begin{aligned} \vec{W}_I^{(k)} &= \vec{W}_I^{(0)} - \frac{\alpha_k \Delta t_I^*}{\Omega_I^{n+1}} \left[\bar{I} + \frac{3}{2\Delta t} \alpha_k \Delta t_I^* \bar{M}^{n+1} \right]^{-1} \\ &\cdot \left[\vec{R}_I(\vec{W}_I^{(k-1)}) + \frac{3}{2\Delta t} (\Omega\bar{M})_I^{n+1} \vec{W}_I^{(0)} - \vec{Q}_I^* \right]. \end{aligned} \quad (6.85)$$

The same methodology can also be applied to a hybrid multistage scheme (see Section 6.1.2). The above formulation (6.85) is stable for any physical time step Δt [129].

In the case of cell-centered schemes, the mass matrix \bar{M}^{n+1} in Eq. (6.85) can be lumped (substituted by the identity matrix) without reducing the solution accuracy. In this way, the term

$$\left[\bar{I} + \frac{3}{2\Delta t} \alpha_k \Delta t_I^* \bar{M}^{n+1} \right]^{-1}$$

in Eq. (6.85) is turned into a scalar value. However, we have to account for the mass matrix in the case of a cell-vertex spatial discretization scheme. Otherwise, the multistage scheme will be unstable for small physical time steps. In order to circumvent the expensive inversion of \bar{M}^{n+1} , Venkatakrishnan [130] and Venkatakrishnan and Mavriplis [131] suggested the following modification to Eq. (6.85)

$$\vec{W}_I^{(k)} = \vec{W}_I^{(0)} - \frac{\alpha_k \Delta t_I^*}{\Omega_I^{n+1}} \left[1 + \frac{3}{2 \Delta t} \alpha_k \Delta t_I^* \beta \right]^{-1} \cdot \left[\vec{R}_I^*(\vec{W}^{(k-1)}) - \frac{3}{2 \Delta t} \Omega_I^{n+1} \beta \vec{W}_I^{(k-1)} \right]. \quad (6.86)$$

The parameter β can now be utilized to stabilize the time-stepping scheme. In practice, choosing $\beta = 2$ was found sufficient [130, 131].

The dual time-stepping approach, where the solution in pseudo time is obtained by an explicit multistage scheme, is widely used. The highest computational efficiency results when the multistage scheme is accelerated by local time-stepping (in t^*) and multigrid. The reason is that the multigrid scheme converges quickly (in few cycles) to the stationary solution of Eq. (6.80). Examples of applications on structured grids can be found in Refs. [127–129] as well as in Refs. [132–134]. Implementations of the methodology on unstructured grids were described, for example, in Refs. [130, 131, 135].

6.3.2 Dual time-stepping for implicit schemes

The implementation of an implicit scheme for the solution of Eq. (6.80) in pseudo time t^* proceeds in the same way as outlined in Section 6.2. First of all, we formulate Eq. (6.80) as an nonlinear implicit scheme, that is,

$$\frac{\partial}{\partial t^*} (\Omega_I^{n+1} \vec{W}_I^*) = -(\vec{R}_I^*)^{l+1} \quad (6.87)$$

with $(l+1)$ being the new pseudo-time level. Note again the absence of \bar{M} in the time derivative. The unsteady residual, which is defined in Eq. (6.81), can be linearized in pseudo time as follows

$$(\vec{R}^*)^{l+1} \approx (\vec{R}^*)^l + \frac{\partial \vec{R}^*}{\partial \vec{W}^*} \Delta \vec{W}^*, \quad (6.88)$$

where $\Delta \vec{W}^* = (\vec{W}^*)^{l+1} - (\vec{W}^*)^l$ and the flux Jacobian is defined as

$$\frac{\partial \vec{R}^*}{\partial \vec{W}^*} = \frac{\partial \vec{R}}{\partial \vec{W}} + \frac{3}{2 \Delta t} (\Omega \bar{M})^{n+1}. \quad (6.89)$$

If we insert the above linearization into Eq. (6.87), we obtain the unfactored implicit scheme [136]

$$\left[\left(\frac{1}{\Delta t_I^*} + \frac{3}{2 \Delta t} \right) (\Omega \bar{M})_I^{n+1} + \left(\frac{\partial \vec{R}}{\partial \vec{W}} \right)_I \right] \Delta \vec{W}^* = -(\vec{R}_I^*)^l. \quad (6.90)$$

Any of the methodologies presented in [Section 6.2](#) can be employed for the solution of the system [\(6.90\)](#). A detailed discussion of time-accurate implicit methods can be found in [\[137\]](#). For examples of implementations, the reader is referred to, for example, [\[136, 138, 139\]](#).

REFERENCES

- [1] Jameson A, Schmidt W, Turkel E. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. *AIAA Paper 81-1259*; 1981.
- [2] Van Leer B, Tai C-H, Powell KG. design of optimally smoothing multi-stage schemes for the Euler equations. *AIAA Paper 89-1933*, 1989.
- [3] Tai C-H, Sheu J-H, van Leer B. Optimal multistage schemes for Euler equations with residual smoothing. *AIAA J* 1995;33:1008-16.
- [4] Tai C-H, Sheu J-H, Tzeng P-Y. Improvement of explicit multistage schemes for central spatial discretization. *AIAA J* 1996;34:185-8.
- [5] Martinelli L. Calculations of viscous flows with a multigrid method. PhD Thesis. Dept. of Mechanical and Aerospace Engineering, Princeton University; 1987.
- [6] Mavriplis DJ, Jameson A. Multigrid solution of the Navier-Stokes equations on triangular meshes. *AIAA J* 1990;28:1415-25.
- [7] Lafon A, Yee HC. On the numerical treatment of nonlinear source terms in reaction-convection equations. *AIAA Paper 92-0419*; 1992.
- [8] Curtiss CF, Hirschfelder JO. Integration of stiff equations. *Proc Natl Acad Sci USA* 1952;38.
- [9] Bussing TRA. A finite volume method for the Navier-Stokes equations with finite rate chemistry. PhD Thesis. Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology; 1985.
- [10] Bussing TRA, Murman EM. Finite-volume method for the calculation of compressible chemically reacting flows. *AIAA J* 1988;26:1070-8.
- [11] Kunz RF, Lakshminarayana B. Stability of explicit Navier-Stokes procedures using $k - \varepsilon$ and $k - \varepsilon$ /algebraic Reynolds stress turbulence models. *J Comput Phys* 1992;103:141-59.
- [12] Jonas S, Fröhlauf HH, Knab O. Fully coupled approach to the calculation of nonequilibrium hypersonic flows using a Godunov-type method. In: 1st European computational fluid dynamics conf. Brussels, Belgium; September 7-11, 1992.
- [13] Merci B, Steelant J, Vierendeels J, Riemsdagh K, Dick E. Computational treatment of source terms in two-equation turbulence models. *AIAA J* 2000;38:2085-93.
- [14] Rosenbrock HH. Some general implicit processes for the numerical solution of differential equations. *Comput J* 1963;5:329-30.
- [15] Butcher JC. Implicit Runge-Kutta processes. *Math Comput* 1964;18:50-64.
- [16] Courant R, Friedrichs KO, Lewy H. Über die partiellen Differenzengleichungen der mathematischen Physik. *Math Ann* 1928;100:32-74. [Transl.: On the Partial Difference Equations of Mathematical Physics. *IBM J* 1967;11:215-34].
- [17] Rizzi A, Inouye M. A time-split finite volume technique for three-dimensional blunt-body flow. *AIAA Paper 73-0133*; 1973.
- [18] Müller B, Rizzi A. Runge-Kutta finite-volume simulation of laminar transonic flow over a NACA0012 airfoil using the Navier-Stokes equations. *FFA TN* 1986-60; 1986.
- [19] Swanson RC, Turkel E, White JA. An effective multigrid method for high-speed flows. In: Proc. 5th Copper Mountain conf. on multigrid methods; 1991.
- [20] Vijayan P, Kallinderis Y. A 3D finite-volume scheme for the Euler equations on adaptive tetrahedral grids. *J Comput Phys* 1994;113:249-67.
- [21] George A, Liu JW. Computer solution of large sparse positive definite systems. Prentice Hall Series in Comput Math. Englewood Cliffs, NJ: Prentice Hall; 1981.
- [22] Pothen A, Simon HD, Liou KP. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J Matrix Anal Appl* 1990;11:430-52.

- [23] Vanden KJ, Whitfield DL. Direct and iterative algorithms for the three-dimensional Euler equations. AIAA Paper 93-3378, 1993 [also AIAA J 1995;33:851-8].
- [24] Cuthill E, McKee J. Reducing the bandwidth of sparse symmetric matrices. In: Proc. ACM 24th national conference; 1969. p. 157-61.
- [25] Gibbs NE, Poole WG, Stockmeyer PK. An algorithm for reducing the bandwidth and profile of a sparse matrix. SIAM J Numer Anal 1976;13:236-50.
- [26] Löhner R. Some useful renumbering strategies for unstructured grids. Int J Numer Meth Eng 1993;36:3259-70.
- [27] Steger JL, Warming RF. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods. J Comput Phys 1981;40:263-93.
- [28] Liou MS, Van Leer B. Choice of implicit and explicit operators for the upwind differencing method. AIAA Paper 88-0624; 1988.
- [29] Barth TJ, Linton SW. An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation. AIAA Paper 95-0221; 1995.
- [30] Orkwis PD, Vanden KJ. On the accuracy of numerical versus analytical Jacobians. AIAA Paper 94-0176; 1994 [also AIAA J 1996;34:1125-9].
- [31] Barth TJ. Analysis of implicit local linearization techniques for upwind and TVD algorithms. AIAA Paper 87-0595; 1987.
- [32] Bischof C, Khademi P, Mauer A, Carle A. Adifor 2.0: automatic differentiation of Fortran 77 programs. IEEE Comput Sci Eng 1996;Fall:18-32.
- [33] Narayanan SHK, Norris B, Winnicka B. ADIC2: Development of a component source transformation system for differentiating C and C++. In: Int. conf. on computational science, ICCS; 2010.
- [34] <http://www.mcs.anl.gov/OpenAD>.
- [35] Hogan RJ. Fast reverse-mode automatic differentiation using expression templates in C++. ACM Trans Math Software 2014;40:26:1-16 [see also: <http://www.met.rdg.ac.uk/clouds/adept>].
- [36] Dennis JE, Schnabel RB. Numerical methods for unconstrained optimization and nonlinear equations. Englewood Cliffs, NJ: Prentice-Hall; 1983.
- [37] Xu X, Richards BE. Simplified procedure for numerically approximate jacobian matrix generation in Newton's method for solving the Navier-Stokes equations. GU Aero Report 9320, Dept. Aerospace Eng., University of Glasgow; 1993.
- [38] Curtis AR, Powell MD, Reid JK. On the estimation of sparse Jacobian matrices. J Inst Maths Appl 1974;13:117-19.
- [39] Venkatakrishnan V. Preconditioned conjugate gradient methods for the compressible Navier-Stokes equations. AIAA Paper 90-0586; 1990 [also AIAA J 1991;29:1092-100].
- [40] Briley WR, McDonald H. Solution of the multi-dimensional compressible Navier-Stokes equations by a generalized implicit method. J Comput Phys 1977;24:372-97.
- [41] Beam R, Warming RF. An implicit factored scheme for the compressible Navier-Stokes equations. AIAA J 1978;16:393-402.
- [42] Steger JL. Implicit finite difference simulation of flow about arbitrary geometries with application to airfoils. AIAA J 1978;16:679.
- [43] Pulliam TH, Steger JL. Recent improvements in efficiency, accuracy and convergence for implicit approximate factorization scheme. AIAA Paper 85-0360; 1985.
- [44] Pulliam TH. Artificial dissipation models for the Euler equations. AIAA J 1986;24:1931-40.
- [45] Pulliam TH, Chaussee DS. A diagonal form of an implicit approximate factorization algorithm. J Comput Phys 1987;39:347-63.
- [46] Lomax H. Some notes on finite difference methods. Lecture Notes. Stanford University; 1986.
- [47] Pulliam TH. Implicit methods in CFD. Numerical methods for fluid dynamics III. Oxford University Press; 1988.
- [48] Rosenfeld M, Yassour Y. The alternating direction multi-zone implicit method. J Comput Phys 1994;110:212-20.
- [49] Jameson A, Turkel E. Implicit scheme and LU-decompositions. Math Comput 1981;37:385-97.
- [50] Yoon S, Jameson A. A multigrid LU-SSOR scheme for approximate Newton-iteration applied to the Euler equations. NASA CR-17954; 1986.

- [51] Yoon S, Jameson A. Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations. AIAA Paper 87-0600, 1987 [AIAA J 1988;26:1025-26].
- [52] Yoon S, Jameson A. LU implicit schemes with multiple grids for the Euler equations. AIAA Paper 86-0105; 1986 [also AIAA J 1987;7:929-35].
- [53] Rieger H, Jameson A. Solution of steady 3-D compressible Euler and Navier-Stokes equations by an implicit LU scheme. AIAA Paper 88-0619; 1988.
- [54] Yoon S, Kwak D. 3-D incompressible Navier-Stokes solver using lower-upper symmetric-Gauss-Seidel algorithm. AIAA J 1991;29.
- [55] Blazek J. Investigations of the implicit LU-SSOR scheme. DLR Research Report No. 93-51; 1993.
- [56] Pahlke K, Blazek J, Kirchner A. Time-accurate Euler computations for rotor flows. European forum: recent developments and applications in aeronautical CFD, Paper No. 15. Bristol, UK; 1993.
- [57] Yoon S, Kwak D. Multigrid convergence of an implicit symmetric relaxation scheme. AIAA Paper 93-3357; 1993.
- [58] Blazek J. A multigrid LU-SSOR scheme for the solution of hypersonic flow problems. AIAA Paper 94-0062; 1994.
- [59] Candler GV, Wright MJ, McDonald JD. A data-parallel LU-SGS method for reacting flows. AIAA Paper 94-0410; 1994.
- [60] Stoll P, Gerlinger P, Brüggemann D. Domain decomposition for an implicit LU-SGS scheme using overlapping grids. AIAA Paper 97-1896; 1997.
- [61] Lee B-S, Lee DH. Data parallel symmetric Gauss-Seidel algorithm for efficient distributed computing using massively parallel supercomputers. AIAA Paper 97-2138; 1997.
- [62] Otero E, Eliasson P. Parameter investigation with line-implicit lower-upper symmetric Gauss-Seidel on 3D stretched grids. AIAA Paper 2014-2094; 2014.
- [63] Xu L, Weng P. High order accurate and low dissipation method for unsteady compressible viscous flow computation on helicopter rotor in forward flight. J Comput Phys 2014;258:470-88.
- [64] Tomaro RF, Strang WZ, Sankar LN. An implicit algorithm for solving time dependent flows on unstructured grids. AIAA Paper 97-0333; 1997.
- [65] Sharov D, Nakahashi K. Reordering of 3-D hybrid unstructured grids for vectorized LU-SGS Navier-Stokes calculations. AIAA Paper 97-2102; 1997.
- [66] Kano S, Kazuhiro N. Navier-Stokes computations of HSCT off-design aerodynamics using unstructured hybrid grids. AIAA Paper 98-0232; 1998.
- [67] Sharov D, Nakahashi K. Low speed preconditioning and LU-SGS scheme for 3-D viscous flow computations on unstructured grids. AIAA Paper 98-0614; 1998.
- [68] Strang WZ, Tomaro RF, Grismer MJ. The defining methods of Cobalt₆₀: a parallel, implicit, unstructured Euler/Navier-Stokes flow solver. AIAA Paper 99-0786; 1999.
- [69] Parsani M, Van den Abeele K, Lacor C, Turkel E. Implicit LU-SGS algorithm for high-order methods on unstructured grid with p-multigrid strategy for solving the steady Navier-Stokes equations. J Comput Phys 2010;229:828-50.
- [70] Parsani M, Ghorbaniasl G, Lacor C. Analysis of the implicit LU-SGS algorithm for 3rd- and 4th-order spectral volume scheme for solving the steady Navier-Stokes equations. J Comput Phys 2011;230:7073-85.
- [71] Haga T, Kuzuu K, Takaki R, Shima E. Assessment of an unstructured CFD solver for RANS simulation on body-fitted Cartesian grids. AIAA Paper 2014-0241; 2014.
- [72] Shuen S, Yoon S. Numerical study of chemically reacting flows using a lower-upper symmetric successive overrelaxation scheme. AIAA J 1989;27:1752-60.
- [73] Shuen JS. Upwind differencing and LU factorization for chemical non-equilibrium Navier-Stokes equations. J Comput Phys 1992;99:233-50.
- [74] Shuen JS, Chen KH, Choi Y. A coupled implicit method for chemical non-equilibrium flows at all speeds. J Comput Phys 1993;106:306-18.
- [75] Palmer G, Venkapaty E. Comparison of nonequilibrium solution algorithms applied to chemically stiff hypersonic flows. AIAA J 1995;33:1211-19.
- [76] Gerlinger P, Stoll P, Brüggemann D. An implicit multigrid method for the simulation of chemically reacting flows. J Comput Phys 1998;146:322-45.

- [77] Yoon S, Jost G, Chang S. Parallelization of lower-upper symmetric Gauss-Seidel method for chemically reacting flow. AIAA Paper 2005-4627; 2005.
- [78] Chen B, Xu X, Cai G. A multi-code CFD solver for the efficient simulation of Ramjet/Scramjet inlet-engine coupled flowfields. AIAA Paper 2007-5414; 2007.
- [79] Janus JM. A matrix study of $Ax=b$ for implicit factored methods. AIAA Paper 98-0112; 1998.
- [80] Yuan X, Daiguji H. A new LU-type implicit scheme for three-dimensional compressible Navier-Stokes equations. In: Hafez M, Oshima K, editors. 6th Int. symposium on CFD, vol. III. Lake Tahoe; 1995 [Pergamon Press; 1998. p. 1473-8].
- [81] Soetrino M, Imlay ST, Roberts DW. A zonal implicit procedure for hybrid structured-unstructured grids. AIAA Paper 94-0645; 1994.
- [82] Soetrino M, Imlay ST, Roberts DW, Taflin DE. Development of a zonal implicit procedure for hybrid structured-unstructured grids. AIAA Paper 96-0167; 1996.
- [83] Sonneveld P. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. SIAM J Sci Stat Comput 1989;10:36-52.
- [84] Van der Vorst HA. BiCGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J Sci Stat Comput 1992;13:631-44.
- [85] Freund RW. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. SIAM J Sci Comput 1993;14:470-82.
- [86] Saad Y, Schulz MH. GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 1986;7:856-69.
- [87] Saad Y. Krylov subspace techniques, conjugate gradients, preconditioning and sparse matrix solvers. VKI Lecture Series; 1994-05; 1994.
- [88] Walker HF. Implementation of the GMRES method using householder transformations. SIAM J Sci Comput 1988;9:152.
- [89] Morgan RB. A restarted GMRES method augmented with eigenvectors. SIAM J Matrix Anal Appl 1995;16:1154-71.
- [90] Chapman A, Saad Y. Deflated and augmented Krylov subspace techniques. Technical Report UMSI 95/181. Minnesota Supercomputer Institute; 1995.
- [91] Saad Y. Enhanced acceleration and reconditioning techniques. In: Hafez M, Oshima K, editors. CFD Review 1998. Singapore: World Scientific Publishing Co.; 1998. p. 478-87.
- [92] Ajmani K, Ng, W-F, Liou M-S. Preconditioned conjugate gradient methods for low speed flow calculations. AIAA Paper 93-0881; 1993.
- [93] Brown PN, Saad Y. Hybrid Krylov methods for nonlinear systems of equations. SIAM J Sci Stat Comput 1990;11:450-81.
- [94] Zingg D, Pueyo A. An efficient Newton-GMRES solver for aerodynamic computations. AIAA Paper 97-1955; 1997.
- [95] Gropp WD, Keyes DE, McInnes LC, Tidriri MD. Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD. ICASE Report No. 98-24; 1998.
- [96] Knoll DA, Keyes DE. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. J Comput Phys 2004;193:357-97.
- [97] Nejat A, Ollivier-Gooch C. A high-order accurate unstructured finite volume Newton-Krylov algorithm for inviscid compressible flows. J Comput Phys 2008;227:2582-609.
- [98] Chisholm TT, Zingg DW. A Jacobian-free Newton-Krylov algorithm for compressible turbulent fluid flows. J Comput Phys 2009;228:3490-507.
- [99] Lucas P, van Zuijlen AH, Bijl H. Fast unsteady flow computations with a Jacobian-free Newton-Krylov algorithm. J Comput Phys 2010;229:9201-15.
- [100] Cai X-C, Gropp WD, Keyes DE, Tidriri MD. Newton-Krylov-Schwarz methods in CFD. In: Hebecker F-K, Rannacher R, Wittum G, editors. Proceedings of the International Workshop on the Navier-Stokes Equations; 2013. p. 17-30.
- [101] Meijerink JA, Van der Vorst HA. Guidelines for usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. J Comput Phys 1981;44:134-55.
- [102] Hackbusch W. Iterative solution of large sparse systems of equations. New York: Springer Verlag; 1994.

- [103] Venkatakrishnan V, Mavriplis DJ. Implicit solvers for unstructured meshes. *J Comput Phys* 1993;105:83–91.
- [104] Ajmani K, Liou M-S. Implicit conjugate gradient solvers on distributed-memory architectures. AIAA Paper 95-1695; 1995.
- [105] Luo H, Baum JD, Löhner R. A fast, Matrix-free implicit method for compressible flows on unstructured grids. AIAA Paper 99-0936; 1999.
- [106] Mulder W, Van Leer B. Experiments with implicit upwind methods for the Euler equations. *J Comput Phys* 1985;59:232–46.
- [107] Cooper GJ, Sayfy A. Additive methods for the numerical solution of ordinary differential equations. *Math Comput* 1980;35:1159–72.
- [108] Cooper GJ, Sayfy A. Additive Runge-Kutta methods for stiff ordinary differential equations. *Math Comput* 1983;40:207–18.
- [109] Kennedy CA, Carpenter MH. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Appl Numer Math* 2003;44:139–81.
- [110] Alexander R. Diagonally implicit Runge-Kutta methods for stiff ODE's. *SIAM J Numer Anal* 1977;14:1006–21.
- [111] Najafi-Yazdi A, Mongeau L. A low-dispersion and low-dissipation implicit Runge-Kutta scheme. *J Comput Phys* 2013;233:315–23.
- [112] Nazari F, Mohammadian A, Charron M, Zadra A. Optimal high-order diagonally-implicit Runge-Kutta schemes for nonlinear diffusive systems on atmospheric boundary layer. *J Comput Phys* 2014;271:118–30.
- [113] Zhong X. Additive semi-implicit Runge-Kutta schemes for computing high-speed nonequilibrium reactive flows. *J Comput Phys* 1996;128:19–31.
- [114] Yoh JJ-I, Zhong X. Semi-implicit Runge-Kutta schemes for stiff multi-dimensional reacting flows. AIAA Paper 97-0803; 1997.
- [115] Yoh JJ-I, Zhong X. Low-Storage semi-implicit Runge-Kutta methods for reactive flow computations. AIAA Paper 98-0130; 1998.
- [116] Calvo MP, de Frutos J, Novo J. Linearly implicit Runge-Kutta methods for advection-reaction-diffusion equations. *Appl Numer Math* 2001;37:535–49.
- [117] Dong H, Zhong X. Time-accurate simulations of hypersonic boundary layer stability and transition over blunt bodies using implicit parallel algorithms. AIAA Paper 2002-0156; 2002.
- [118] Calvo MP, Gerisch A. Linearly implicit Runge-Kutta methods and approximate matrix factorization. *Appl Numer Math* 2005;53:183–200.
- [119] Kanevsky A, Carpenter MH, Gottlieb D, Hesthaven JS. Application of implicit-explicit high order Runge-Kutta methods to discontinuous-Galerkin schemes. *J Comput Phys* 2007;225:1753–81.
- [120] Kazemi-Kamyab V, van Zuijlen AH, Bijl H. Analysis and application of high order implicit Runge-Kutta schemes for unsteady conjugate heat transfer: a strongly-coupled approach. *J Comput Phys* 2014;272:471–86.
- [121] Langer S. Agglomeration multigrid methods with implicit Runge-Kutta smoothers applied to aerodynamic simulations on unstructured grids. *J Comput Phys* 2014;277:72–100.
- [122] Tam CKW. Computational aeroacoustics: issues and methods. AIAA Paper 95-0677; 1995.
- [123] Tam CKW. Applied aero-acoustics: prediction methods. VKI Lecture Series 1996-04; 1996.
- [124] Bijl H, Carpenter MH, Vatsa VN, Kennedy CA. Implicit time integration schemes for the unsteady compressible Navier-Stokes equations: laminar flow. *J Comput Phys* 2002;179:313–29.
- [125] Carpenter MH, Viken SA, Nielsen EJ. The efficiency of high order temporal schemes. AIAA Paper 2003-0086; 2003.
- [126] Jothiprasad G, Mavriplis DJ, Caughey DA. Higher-order time integration schemes for the unsteady Navier-Stokes equations on unstructured meshes. *J Comput Phys* 2003;191:542–66.
- [127] Jameson A. Time-dependent calculations using multigrid with applications to unsteady flows past airfoils and wings. AIAA Paper 91-1596; 1991.
- [128] Arnone A, Liou MS, Povinelli LA. Multigrid time-accurate integration of Navier-Stokes equations. AIAA Paper 93-3361; 1993.

- [129] Melson ND, Sanetrik MD, Atkins HL. Time-accurate Navier-Stokes calculations with multigrid acceleration. In: Proc. 6th Copper Mountain conf. on multigrid methods; 1993. p. 423-39.
- [130] Venkatakrishnan V. Implicit schemes and parallel computing in unstructured grid CFD. ICASE Report, No. 95-28; 1995.
- [131] Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. J Comput Phys 1996;127:380-97.
- [132] Alonso JJ, Jameson A. Fully-implicit time-marching aeroelastic solution. AIAA Paper 94-0056; 1994.
- [133] Belov A, Martinelli L, Jameson A. A new implicit algorithm with multigrid for unsteady incompressible flow calculations. AIAA Paper 95-0049; 1995.
- [134] Pierce NA, Alonso JJ. A preconditioned implicit multigrid algorithm for parallel computation of unsteady aeroelastic compressible flows. AIAA Paper 97-0444; 1997.
- [135] Singh KP, Newman JC, Baysal O. Dynamic unstructured method for flows past multiple objects in relative motion. AIAA J 1995;33:641-9.
- [136] Dubuc L, Cantariti F, Woodgate M, Gribben B, Badcock KJ, Richards BE. Solution of the unsteady euler equations using an implicit dual-time method. AIAA J 1998;36:1417-24.
- [137] Pulliam TH. Time accuracy and the use of implicit methods. AIAA paper 93-3360; 1993.
- [138] Bartels RE. An elasticity-based mesh scheme applied to the computation of unsteady three-dimensional spoiler and aeroelastic problems. AIAA Paper 99-3301; 1999.
- [139] Chassaing JC, Gerolymos GA, Vallet I. Reynolds-stress model dual-time-stepping computation of unsteady three-dimensional flows. AIAA J 2003;41:1882-94.

CHAPTER 7

Turbulence Modeling

Contents

7.1 Basic Equations of Turbulence	215
7.1.1 Reynolds averaging	217
7.1.2 Favre (mass) averaging	218
7.1.3 Reynolds-averaged Navier-Stokes equations	219
7.1.4 Favre- and Reynolds-averaged Navier-Stokes equations	220
7.1.5 Eddy-viscosity hypothesis	221
7.1.6 Non-linear eddy viscosity	223
7.1.7 Reynolds-stress transport equation	224
7.2 First-Order Closures	225
7.2.1 Spalart-Allmaras one-equation model	225
<i>Differential form</i>	226
<i>Integral form</i>	227
<i>Initial and boundary conditions</i>	228
7.2.2 $K - \epsilon$ two-equation model	228
<i>Differential form</i>	229
<i>Integral form</i>	230
<i>Initial and boundary conditions</i>	231
<i>Wall functions</i>	232
7.2.3 SST two-equation model of Menter	232
<i>Differential form</i>	233
<i>Boundary conditions</i>	234
7.3 Large-Eddy Simulation	235
7.3.1 Spatial filtering	236
7.3.2 Filtered governing equations	237
<i>Incompressible Navier-Stokes equations</i>	237
<i>Compressible Navier-Stokes equations</i>	238
7.3.3 Subgrid-scale modeling	240
<i>Eddy-viscosity models</i>	240
<i>Smagorinsky SGS model</i>	241
<i>Dynamic SGS models</i>	241
7.3.4 Wall models	242
7.3.5 Detached eddy simulation	243
References	244

The outstanding feature of a turbulent flow, in the opposite to a laminar flow, is that the molecules move in a chaotic fashion along complex irregular paths. The strong chaotic motion causes the various layers of the fluid to mix together intensely. Because

of the increased momentum and energy exchange between the molecules and solid walls, turbulent flow leads at the same conditions to higher skin friction and heat transfer as compared to laminar flow.

Although the chaotic fluctuations of the flow variables are of deterministic nature, the simulation of turbulent flows still continues to present a significant problem. Despite the performance of modern supercomputers, a direct simulation of turbulence by the time-dependent Navier-Stokes equations (2.19)—known as the *direct numerical simulation* (DNS) [1–16]—is applicable only to relatively simple flow problems at low Reynolds numbers (Re) in the order of 10^4 – 10^5 . A more widespread utilization of the DNS is prevented by the fact that the number of grid points needed for sufficient spatial resolution scales as $Re^{9/4}$ and the CPU-time as Re^3 . Therefore, we are forced to account for the effects of turbulence in an approximate manner. For this purpose, a large variety of *turbulence models* was developed and the research still goes on. There are five principal classes of turbulence models:

- algebraic,
- one-equation,
- multiple-equation,
- *second-order* closures (Reynolds-stress models),
- *large-eddy simulation* (LES).

The first three models belong to the so-called *first-order* closures. They are based mostly on the *eddy-viscosity* hypothesis of Boussinesq [17, 18], but for certain applications also on *non-linear eddy-viscosity* formulations. An overview of the classes of turbulence models, which are sorted according to their decreasing level of complexity, is displayed in Fig. 7.1.

One should be aware of the fact that there is no single turbulence model, which can predict reliably all kinds of turbulent flows. Each of the models has its strengths and weaknesses. For example, if a particular model works perfectly in the case of attached boundary layers, it may fail completely for separated flows. Thus, it is important always to ask whether the model includes all the significant features of the flow being investigated. Another point which should be taken into consideration is the computational effort versus the accuracy required by the particular application. We mean by this that in many cases a numerically inexpensive turbulence model can predict some global measures with the same accuracy as a more complex model.

In the following, we first introduce the basic equations of turbulence as they result from time and mass averaging of the governing equations. Then, we present the Boussinesq's and the non-linear eddy-viscosity approaches. After that, we briefly discuss the Reynolds-stress transport equation, which forms the basis of the algebraic and differential Reynolds-stress models. In Section 7.2, we present few wide-spread one- and two-equation first-order closures. Finally, we discuss LES and related approaches in some detail because of the growing number of their engineering applications.

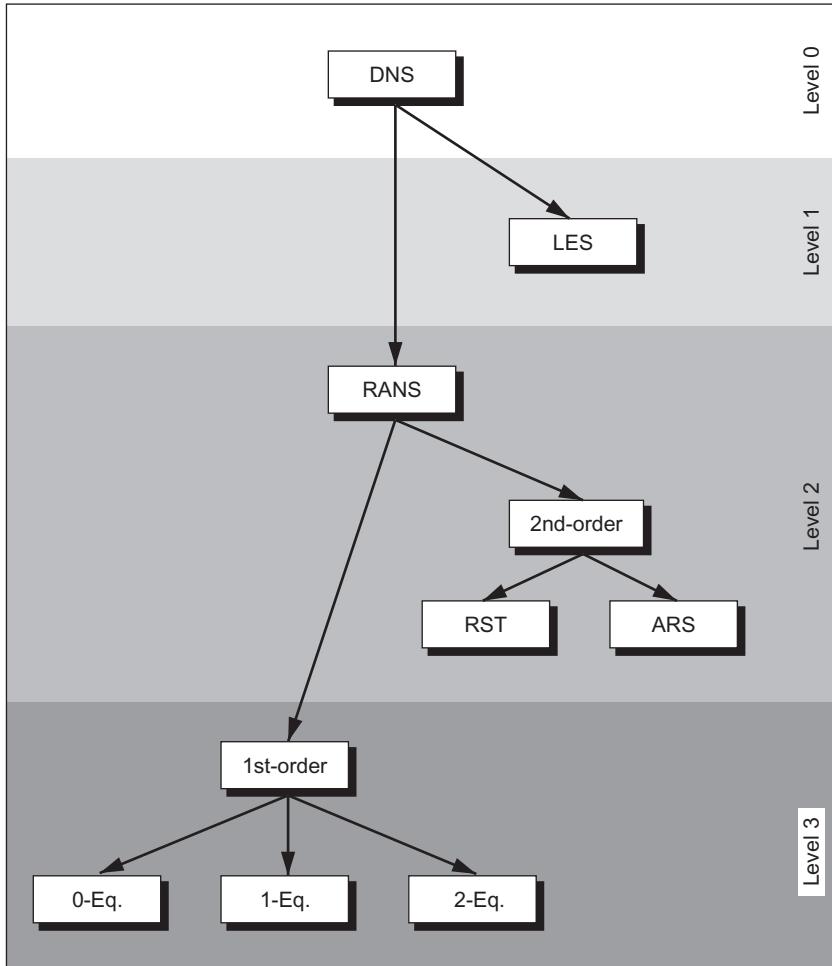


Figure 7.1 Hierarchy of turbulence models. Abbreviations: DNS, direct numerical simulation; LES, large-eddy simulation; RANS, Reynolds-averaged Navier-Stokes equations; 1st-order, first-order closures; 2nd-order, second-order closures; RST, Reynolds-stress transport models; ARS, algebraic Reynolds-stress models; 0-, 1-, 2-Eq., zero- (algebraic), one-, two-equations models.

7.1 BASIC EQUATIONS OF TURBULENCE

First of all, let us rewrite the governing equations (2.19) in differential form (see Section A.1), since this is used very often in literature on turbulence modeling. Furthermore, it allows for a compact and clear notation. However, we will also provide examples of turbulence equations in integral form.

In the case of a compressible Newtonian fluid, the Navier-Stokes equations read in the absence of source terms in coordinate invariant formulation as

$$\begin{aligned}\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho v_i) &= 0, \\ \frac{\partial}{\partial t}(\rho v_i) + \frac{\partial}{\partial x_j}(\rho v_j v_i) &= -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j}, \\ \frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho v_j H) &= \frac{\partial}{\partial x_j}(v_i \tau_{ij}) + \frac{\partial}{\partial x_j}\left(k \frac{\partial T}{\partial x_j}\right).\end{aligned}\quad (7.1)$$

where v_i denotes a velocity component ($\vec{v} = [v_1, v_2, v_3]^T$), and x_i stands for a coordinate direction, respectively. An explanation of the compact tensor notation can be found in Section A.13.

The components of the *viscous stress tensor* τ_{ij} in Eq. (7.1) are defined as

$$\tau_{ij} = 2\mu S_{ij} + \lambda \frac{\partial v_k}{\partial x_k} \delta_{ij} = 2\mu S_{ij} - \left(\frac{2\mu}{3}\right) \frac{\partial v_k}{\partial x_k} \delta_{ij}, \quad (7.2)$$

where we utilized the Stokes's hypothesis (Eq. (2.17)). In Cartesian coordinates, Eq. (7.2) is equivalent to Eq. (2.15). The second term in Eq. (7.2), that is, $\partial v_k / \partial x_k$, which corresponds to the divergence of the velocity, disappears for incompressible flows. The components of the *strain-rate tensor* are given by

$$S_{ij} = \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right). \quad (7.3)$$

In this connection, let us also define the *rotation-rate tensor* (antisymmetric part of the velocity gradient tensor) with the following components

$$\Omega_{ij} = \frac{1}{2} \left(\partial v_i \partial x_j - \frac{\partial v_j}{\partial x_i} \right). \quad (7.4)$$

The total energy E and the total enthalpy H in Eq. (7.1) are readily obtained from the formulas

$$E = e + \frac{1}{2} v_i v_i, \quad H = h + \frac{1}{2} v_i v_i \quad (7.5)$$

which, in Cartesian coordinate system, correspond to Eq. (2.6) and Eq. (2.12), respectively.

For incompressible flows, we can reduce Eq. (7.1) to the form

$$\begin{aligned}\frac{\partial v_i}{\partial x_i} &= 0, \\ \frac{\partial v_i}{\partial t} + v_j \frac{\partial v_i}{\partial x_j} &= -\frac{1}{\rho} \frac{\partial p}{\partial x_i} + \nu \nabla^2 v_i, \\ \frac{\partial T}{\partial t} + v_j \frac{\partial T}{\partial x_j} &= k \nabla^2 T\end{aligned}\quad (7.6)$$

with $\nu = \mu/\rho$ being the kinematic viscosity coefficient and ∇^2 denoting the Laplace operator. In the absence of buoyancy effects, the equation for the temperature T becomes decoupled from the mass conservation and the momentum equations.

7.1.1 Reynolds averaging

The first approach for the approximate treatment of turbulent flows was presented by Reynolds in 1895. The methodology is based on the decomposition of the flow variables into a mean and a fluctuating part. The governing equations (7.1) are then solved for the mean values, which are the most interesting for engineering applications. Thus, considering first incompressible flows, the velocity components and the pressure in Eq. (7.1) are substituted by [19]

$$v_i = \bar{v}_i + v'_i, \quad p = \bar{p} + p', \quad (7.7)$$

where the mean value is denoted by a bar and the turbulent fluctuations by a prime. The mean values are obtained by an averaging procedure. There are three different forms of the *Reynolds averaging*:

1. *Time averaging*—appropriate for stationary turbulence (statistically steady turbulence)

$$\bar{v}_i = \lim_{T \rightarrow \infty} \frac{1}{T} \int_t^{t+T} v_i dt. \quad (7.8)$$

As a consequence, the mean value \bar{v}_i does not vary in time, but only in space. The situation is sketched in Fig. 7.2. In practice, $T \rightarrow \infty$ means that the time interval T should be large as compared to the typical time scale of the turbulent fluctuations.

2. *Spatial averaging*—appropriate for homogeneous turbulence

$$\bar{v}_i = \lim_{\Omega \rightarrow \infty} \frac{1}{\Omega} \int_{\Omega} v_i d\Omega \quad (7.9)$$

with Ω being a control volume. In this case, \bar{v}_i is uniform in space, but it is allowed to variate in time.

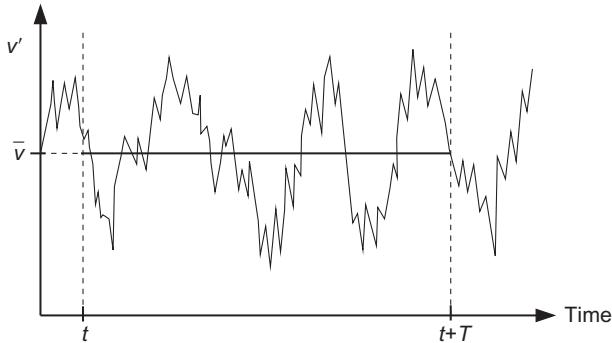


Figure 7.2 Reynolds averaging—illustration of turbulent velocity fluctuations v' and statistical mean value \bar{v} .

3. Ensemble averaging—appropriate for general turbulence

$$\bar{v}_i = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{m=1}^N v_i. \quad (7.10)$$

Here, the mean value \bar{v}_i still remains a function of time and of space coordinates. For all three approaches, the average of the fluctuating part is zero, that is, $\bar{v}'_i = 0$. However, it can be easily seen that $\bar{v}'_i \bar{v}'_i \neq 0$. The same is true for $\bar{v}'_i \bar{v}'_j$, if both turbulent velocity components are correlated.

In cases where the turbulent flow is both stationary and homogeneous, all three averaging forms are equivalent. This is called the *ergodic hypothesis*.

7.1.2 Favre (mass) averaging

In cases where the density is not constant, it is advisable to apply the *density (mass) weighted* or Favre decomposition [20, 21] to certain quantities in Eq. (7.1) instead of Reynolds averaging. Otherwise, the averaged governing equations would become considerably more complicated due to additional correlations involving density fluctuations. The most convenient way is to employ Reynolds averaging for density and pressure, and Favre averaging for other variables such as velocity, internal energy, enthalpy, and temperature. Favre averaged quantities, for example, the velocity components, are obtained from the relation [20, 21]

$$\tilde{v}_i = \frac{1}{\bar{\rho}} \lim_{T \rightarrow \infty} \frac{1}{T} \int_t^{t+T} \rho v_i dt, \quad (7.11)$$

where $\bar{\rho}$ denotes the Reynolds-averaged density. Hence, the Favre decomposition reads

$$v_i = \tilde{v}_i + v''_i, \quad (7.12)$$

where \tilde{v}_i represents the mean value and v_i'' the fluctuating part of the velocity v_i . Again, the average of the fluctuating part is zero, that is, $\overline{v_i''} = 0$. Furthermore, the average of the product of two fluctuating quantities is not zero, if the quantities are correlated. Hence, for example, $\overline{v_i'' v_i''} \neq 0$ and in general $\overline{v_i'' v_j''} \neq 0$.

The following relationships can be derived for a mix between Favre and Reynolds averaging

$$\widetilde{\rho v_i} = \overline{\rho} \tilde{v}_i, \quad \overline{\rho v_i''} = 0, \quad \text{but } \overline{v_i''} \neq 0. \quad (7.13)$$

These relations will be utilized in the following sections.

7.1.3 Reynolds-averaged Navier-Stokes equations

If we apply either the time averaging Eq. (7.8) or the ensemble averaging Eq. (7.10) to the incompressible Navier-Stokes equations (7.6), we obtain the following relations for the mass and momentum conservation

$$\begin{aligned} \frac{\partial \overline{v}_i}{\partial x_i} &= 0, \\ \rho \frac{\partial \overline{v}_i}{\partial t} + \rho \overline{v}_j \frac{\partial \overline{v}_i}{\partial x_j} &= -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j} \left(\overline{\tau}_{ij} - \rho \overline{v'_i v'_j} \right). \end{aligned} \quad (7.14)$$

These are known as the *Reynolds-averaged Navier-Stokes equations* (RANS). Equation (7.14) is formally identical to the Navier-Stokes equations (7.6) with the exception of the additional term

$$\overline{\tau}_{ij}^R = -\rho \overline{v'_i v'_j} = -\rho (\overline{v_i v_j} - \overline{v_i} \overline{v_j}), \quad (7.15)$$

which constitutes the so-called *Reynolds-stress tensor*. It represents the transfer of momentum due to turbulent fluctuations. The laminar viscous stresses are evaluated according to Eqs. (7.2) and (7.3) using Reynolds-averaged velocity components, that is,

$$\overline{\tau}_{ij} = 2\mu \overline{S}_{ij} = \mu \left(\frac{\partial \overline{v}_i}{\partial x_j} + \frac{\partial \overline{v}_j}{\partial x_i} \right). \quad (7.16)$$

The Reynolds-stress tensor consists in 3D of the nine components

$$\rho \overline{v'_i v'_j} = \begin{bmatrix} \rho \overline{(v'_1)^2} & \rho \overline{v'_1 v'_2} & \rho \overline{v'_1 v'_3} \\ \rho \overline{v'_2 v'_1} & \rho \overline{(v'_2)^2} & \rho \overline{v'_2 v'_3} \\ \rho \overline{v'_3 v'_1} & \rho \overline{v'_3 v'_2} & \rho \overline{(v'_3)^2} \end{bmatrix}. \quad (7.17)$$

However, since v'_i and v'_j in the correlations can be interchanged, the Reynolds-stress tensor contains only six independent components. The sum of the normal stresses divided by density defines the *turbulent kinetic energy*, that is,

$$K = \frac{1}{2} \overline{v'_i v'_i} = \frac{1}{2} \left[\overline{(v'_1)^2} + \overline{(v'_2)^2} + \overline{(v'_3)^2} \right]. \quad (7.18)$$

As we can see here, the fundamental problem of turbulence modeling based on the Reynolds-averaged Navier-Stokes equations is to find six additional relations in order to close Eq. (7.14). We shall introduce the basic methodologies in [Sections 7.1.5–7.1.7](#).

7.1.4 Favre- and Reynolds-Averaged Navier-Stokes Equations

In turbulence modeling, it is quite common to assume that *Morkovin's hypothesis* [22] is valid. It states that the turbulent structure of a boundary layer is not notably influenced by density fluctuations if $\rho' \ll \bar{\rho}$. This is generally true for wall-bounded flows up to a Mach number of about five. However, in the case of hypersonic flows or for compressible free shear layers, density fluctuations have to be accounted for. The same holds also for flows with combustion or with significant heat transfer.

Application of the Reynolds averaging (Eq. (7.8) or (7.10)) to density and pressure, and of the Favre averaging Eq. (7.11) to the remaining flow variables in the compressible Navier-Stokes equations (7.1) yields [23]

$$\begin{aligned} \frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_i} (\bar{\rho} \tilde{v}_i) &= 0 \\ \frac{\partial}{\partial t} (\bar{\rho} \tilde{v}_i) + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{v}_j \tilde{v}_i) &= -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j} \left(\tilde{\tau}_{ij} - \bar{\rho} \widetilde{v'_i v'_j} \right) \\ \frac{\partial}{\partial t} (\bar{\rho} \tilde{E}) + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{v}_j \tilde{H}) &= \frac{\partial}{\partial x_j} \left(k \frac{\partial \tilde{T}}{\partial x_j} - \bar{\rho} \widetilde{v'_j h''} + \widetilde{\tau_{ij} v'_i} - \bar{\rho} \widetilde{v'_j K} \right) \\ &\quad + \frac{\partial}{\partial x_j} \left[\tilde{v}_i \left(\tilde{\tau}_{ij} - \bar{\rho} \widetilde{v'_i v'_j} \right) \right]. \end{aligned} \quad (7.19)$$

These are the *Favre- and Reynolds-averaged Navier-Stokes* equations. In an analogy to the Reynolds averaging, the viscous stress tensor in the momentum (and energy) equation becomes extended by the *Favre-averaged Reynolds-stress* tensor, that is,

$$\tau_{ij}^F = -\bar{\rho} \widetilde{v'_i v'_j}. \quad (7.20)$$

Its form is similar to Eq. (7.17) with Favre instead of Reynolds averaging. The components of the laminar (molecular) viscous stress tensor $\tilde{\tau}_{ij}$ are evaluated by Eq. (7.2) using Favre-averaged velocity components.

If we employ the definition of the Favre-averaged turbulent kinetic energy, that is,

$$\bar{\rho} \tilde{K} = \frac{1}{2} \bar{\rho} \widetilde{v'_i v'_i}, \quad (7.21)$$

we can express the total energy in Eq. (7.19) as

$$\bar{\rho} \tilde{E} = \bar{\rho} \tilde{e} + \frac{1}{2} \bar{\rho} \tilde{v}_i \tilde{v}_i + \frac{1}{2} \bar{\rho} \widetilde{v'_i v'_i} = \bar{\rho} \tilde{e} + \frac{1}{2} \bar{\rho} \tilde{v}_i \tilde{v}_i + \bar{\rho} \tilde{K}. \quad (7.22)$$

The total enthalpy is defined as

$$\bar{\rho}\tilde{H} = \bar{\rho}\tilde{h} + \frac{1}{2}\bar{\rho}\tilde{v}_i\tilde{v}_i + \frac{1}{2}\bar{\rho}\tilde{v}_i''\tilde{v}_i'' = \bar{\rho}\tilde{h} + \frac{1}{2}\bar{\rho}\tilde{v}_i\tilde{v}_i + \bar{\rho}\tilde{K}. \quad (7.23)$$

The individual parts of the Favre- and Reynolds-averaged Navier-Stokes equations (7.19) have the following physical meaning [23]:

$$\begin{aligned} \frac{\partial}{\partial x_j} \left(k \frac{\partial \tilde{T}}{\partial x_j} \right) & - \text{molecular diffusion of heat}, \\ \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{v}_j'' h'') & - \text{turbulent transport of heat}, \\ \frac{\partial}{\partial x_j} (\tilde{\tau}_{ij} \tilde{v}_i'') & - \text{molecular diffusion of } \tilde{K}, \\ \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{v}_j'' \tilde{K}) & - \text{turbulent transport of } \tilde{K}, \\ \frac{\partial}{\partial x_j} (\tilde{v}_i \tilde{\tau}_{ij}) & - \text{work done by the molecular stresses}, \\ \frac{\partial}{\partial x_j} (\tilde{v}_i \tau_{ij}^F) & - \text{work done by the Favre-averaged Reynolds stresses}. \end{aligned}$$

The molecular diffusion and turbulent transport of \tilde{K} are very often neglected. This is a valid approximation for transonic and supersonic flows. In order to close the Favre- and Reynolds-averaged equations (7.19), we also have to supply six components of the Favre-averaged Reynolds-stress tensor (Eq. (7.20)) and three components of the turbulent heat-flux vector. We shall discuss the three basic approaches in the following sections.

7.1.5 Eddy-viscosity hypothesis

One of the most significant contributions to turbulence modeling was presented in 1877 by Boussinesq [17, 18]. His idea is based on the observation that the momentum transfer in a turbulent flow is dominated by the mixing caused by large energetic turbulent eddies. The Boussinesq hypothesis assumes that the turbulent shear stress depends linearly on the mean rate of strain, as in a laminar flow. The proportionality factor is the *eddy viscosity*. The Boussinesq hypothesis for Reynolds averaged incompressible flow (Eq. (7.14)) can be written as

$$\tau_{ij}^R = -\rho \overline{v'_i v'_j} = 2\mu_T \bar{S}_{ij} - \frac{2}{3}\rho K \delta_{ij}, \quad (7.24)$$

where \bar{S}_{ij} denotes the Reynolds-averaged strain-rate tensor (Eq. (7.3), cf. also Eq. (7.16)), K is the turbulent kinetic energy ($K = (1/2)\overline{v'_i v'_i}$), and μ_T stands for the eddy

viscosity. Unlike the molecular viscosity μ , the eddy viscosity μ_T represents no physical characteristic of the fluid, but it is a function of the local flow conditions. Additionally, μ_T is also strongly affected by flow history effects.

In the case of the compressible Favre- and Reynolds-averaged Navier-Stokes equations (7.19), the Boussinesq eddy-viscosity hypothesis reads

$$\tau_{ij}^F = -\bar{\rho} \widetilde{v_i'' v_j''} = 2\mu_T \tilde{S}_{ij} - \left(\frac{2\mu_T}{3} \right) \frac{\partial \tilde{v}_k}{\partial x_k} \delta_{ij} - \frac{2}{3} \bar{\rho} \tilde{K} \delta_{ij}, \quad (7.25)$$

where \tilde{S}_{ij} and \tilde{K} are the Favre-averaged strain rate and turbulent kinetic energy, respectively. Note the similarity to Eq. (7.2). The term $(2/3)\rho K \delta_{ij}$ in Eqs. (7.24) and (7.25) is required in order to obtain the proper trace of τ_{ij}^R or τ_{ij}^F . This means that we must have

$$\tau_{ii}^R = -2\rho K \quad \text{or} \quad \tau_{ii}^F = -2\bar{\rho} \tilde{K}$$

in the case of $\bar{S}_{ii} = 0$ (continuity equation) or $\tilde{S}_{ii} = 0$, in order to fulfill the relations Eq. (7.18) or (7.21) for the turbulent kinetic energy. However, the term $(2/3)\rho K \delta_{ij}$ is often neglected, particularly in connection with simpler turbulence models (like the algebraic ones).

The approximation, which is commonly used for the modeling of the turbulent heat-flux vector, is based on the classical Reynolds analogy [24]. Hence, we may write

$$\bar{\rho} \widetilde{v_j'' h''} = -k_T \frac{\partial \tilde{T}}{\partial x_j} \quad (7.26)$$

with the *turbulent thermal conductivity coefficient* k_T being defined as

$$k_T = c_p \frac{\mu_T}{Pr_T}. \quad (7.27)$$

In Eq. (7.27), c_p denotes the specific heat coefficient at constant pressure and Pr_T is the turbulent Prandtl number. In general, it is assumed that the turbulent Prandtl number is constant within the flow field ($Pr_T = 0.9$ for air).

By applying the eddy-viscosity approach to the Reynolds- (and Favre-) averaged form of the governing equations (2.19) or Eq. (7.1), the dynamic viscosity coefficient μ in the viscous stress tensor Eq. (2.15) or Eq. (7.2) is simply replaced by the sum of a laminar and a turbulent component, that is,

$$\mu = \mu_L + \mu_T. \quad (7.28)$$

The laminar viscosity μ_L is computed, for example, with the aid of the Sutherland formula (2.30). Furthermore, according to the Reynolds analogy given by Eq. (7.26), the thermal conductivity coefficient k in Eq. (2.24) or Eq. (7.2) is evaluated as

$$k = k_L + k_T = c_p \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T} \right). \quad (7.29)$$

The eddy-viscosity concept of Boussinesq is, at least from the engineering point of view, very attractive since it requires “only” the determination of μ_T (the turbulent kinetic energy K needed for the term $(2/3)\rho K \delta_{ij}$ in Eq. (7.24) or (7.25) is obtained either as a by-product of the turbulence model or is simply omitted). Once we know the eddy viscosity μ_T , we can easily extend the Navier-Stokes equations (2.19) to (7.1) to the simulation of turbulent flows by introducing averaged flow variables and by adding μ_T to the laminar viscosity. Therefore, Boussinesq’s approach became the basis for a large variety of first-order turbulence closures. However, there are applications for which the Boussinesq hypothesis is no longer valid (see, e.g., [23, p. 214] or [25, p. 111]):

- flows with sudden change of mean strain rate,
- flows with significant streamline curvature,
- flows with rotation and stratification,
- secondary flows in ducts and in turbomachinery,
- flows with boundary layer separation and reattachment.

The limitations of the eddy-viscosity approach are caused by the assumption of equilibrium between the turbulence and the mean strain field, as well as by the independence on system rotation. The results can be notably improved by using appropriate correction terms in the turbulence models [26, 27]. Further increased accuracy of predictions can be achieved through the application of non-linear eddy-viscosity models which are described in the following section.

7.1.6 Non-linear eddy viscosity

In order to remove the restrictions imposed by the assumption of equilibrium between the turbulence and the mean strain rate, Lumley [28, 29] proposed to extend the linear Boussinesq approach by higher-order products of strain and rotation tensors. This can be viewed as a Taylor series expansion. Following the idea of Lumley, numerous non-linear eddy-viscosity models were proposed, see, for example, Refs. [30–37].

In the following, we shall present one recent approach proposed by Shih et al. [31]. It includes up to third-order terms in the general eddy-viscosity formulation and is particularly suited to swirling flows. As already pointed out in [25, p. 194], cubic terms are essential for high accuracy. The Reynolds stresses τ_{ij}^R can be expressed as [31, 32] (cf. Eq. (7.24))

$$\begin{aligned} \rho \overline{\nu'_i \nu'_j} = & \frac{2}{3} \rho K \delta_{ij} - C_1 \frac{\rho K^2}{\varepsilon} 2 S_{ij}^* - C_3 \frac{\rho K^3}{\varepsilon^2} [\bar{S}_{ik} \bar{\Omega}_{kj} - \bar{\Omega}_{ik} \bar{S}_{kj}] \\ & - C_4 \frac{\rho K^4}{\varepsilon^3} [(\bar{S}_{ik})^2 \bar{\Omega}_{kj} - \bar{\Omega}_{ik} (\bar{S}_{kj})^2] \\ & + C_5 \frac{\rho K^4}{\varepsilon^3} \left[\bar{\Omega}_{ik} \bar{S}_{km} \bar{\Omega}_{mj} - \frac{1}{3} \bar{\Omega}_{kl} \bar{S}_{lm} \bar{\Omega}_{mk} \delta_{ij} + I_S S_{ij}^* \right] \end{aligned} \quad (7.30)$$

with \bar{S}_{ij} , $\bar{\Omega}_{ij}$ according to Eqs. (7.3) and (7.4). Furthermore,

$$\begin{aligned} I_s &= \frac{1}{2} [\bar{S}_{kk}\bar{S}_{ll} - (\bar{S}_{kk})^2] S_{ij}^*, \\ S_{ij}^* &= \bar{S}_{ij} - \frac{1}{3}\bar{S}_{kk}\delta_{ij}. \end{aligned} \quad (7.31)$$

The values of the turbulent kinetic energy K and the *dissipation rate* ε are obtained from low-Reynolds $K - \varepsilon$ turbulence model (cf. Section 7.2.2). The factors C_1 to C_5 in Eq. (7.30) are provided in Refs. [31, 32].

In comparison to the linear eddy-viscosity approach, the non-linear models are computationally only slightly more expensive, but they offer a substantially improved prediction capabilities for certain complex turbulent flows.

7.1.7 Reynolds-stress transport equation

It is possible to derive **exact** equations for the Reynolds stresses by taking the time average (second-order moment)

$$\overline{\nu'_i \mathcal{N}(\nu_j) + \nu'_j \mathcal{N}(\nu_i)} = 0, \quad (7.32)$$

where $\mathcal{N}(\nu_i)$ denotes the Navier-Stokes operator, that is,

$$\mathcal{N}(\nu_i) = \rho \frac{\partial \nu_i}{\partial t} + \rho \nu_j \frac{\partial \nu_i}{\partial x_j} + \frac{\partial p}{\partial x_i} - \mu \nabla^2 \nu_i. \quad (7.33)$$

Using the average Eq. (7.32) together with Eq. (7.33), we obtain the following *Reynolds-stress transport equation* [38]

$$\frac{\partial \tau_{ij}^R}{\partial t} + \bar{\nu}_k \frac{\partial \tau_{ij}^R}{\partial x_k} = P_{ij} + \Pi_{ij} - \varepsilon_{ij} - \frac{\partial C_{ijk}}{\partial x_k} + \mu \nabla^2 \tau_{ij}^R \quad (7.34)$$

for incompressible flow. The formulation for compressible flows can be found in Ref. [23, p. 179], or in Refs. [39, 40]. The production of the turbulent kinetic energy P_{ij} , the pressure-strain term Π_{ij} , the dissipation-rate term ε_{ij} , and the third-order diffusion term C_{ijk} in Eq. (7.34) are defined as

$$\begin{aligned} P_{ij} &= -\tau_{ik}^R \frac{\partial \bar{\nu}_j}{\partial x_k} - \tau_{jk}^R \frac{\partial \bar{\nu}_i}{\partial x_k}, \\ \Pi_{ij} &= \overline{p' \left(\frac{\partial \nu'_i}{\partial x_j} + \frac{\partial \nu'_j}{\partial x_i} \right)} = 2 \overline{p' S'_{ij}}, \\ \varepsilon_{ij} &= 2\mu \overline{\frac{\partial \nu'_i}{\partial x_k} \frac{\partial \nu'_j}{\partial x_k}}, \\ C_{ijk} &= \rho \overline{\nu'_i \nu'_j \nu'_k} + \overline{p' \nu'_i} \delta_{jk} + \overline{p' \nu'_j} \delta_{ik}, \end{aligned} \quad (7.35)$$

where S'_{ij} denotes the fluctuating part of the strain-rate tensor. The first part of C_{ijk} , the triple velocity term, represents transport driven by fluctuating convection, the two other parts are the pressure transport terms (pressure–velocity correlations), respectively.

As we can see, the exact Reynolds-stress equation contains new unknown higher-order correlations (e.g., $\overline{v'_i v'_j v'_k}$). Therefore, Eq. (7.34) can be closed only by using **empirical** models. This is caused by the non-linear nature of the Navier-Stokes equations. The second-order closures—the Reynolds-stress models—provide the necessary framework for solving Eq. (7.34). Examples of implementations can be found, for example, in Refs. [41–45].

7.2 FIRST-ORDER CLOSURES

The first-order closures represent the easiest way to approximate the Reynolds stresses in the Reynolds-/Favre-averaged Navier-Stokes equations. They are based on Boussinesq or non-linear eddy-viscosity models, which we discussed in Sections 7.1.5 and 7.1.6, respectively. Consequently, the task of an associated turbulence model is to compute the eddy viscosity μ_T .

From the large variety of first-order closure models, we selected three widely-used approaches which represent the current state-of-the-art. All three models can be implemented easily on structured as well as on unstructured grids. First, we shall discuss the one-equation model due to Spalart and Allmaras. Second, we shall present the well-known $K - \varepsilon$ two-equation model. Finally, we shall consider the $K - \omega$ shear-stress transport (SST) two-equation model proposed by Menter. A detailed comparison of this turbulence models for various cases can be found in Ref. [46].

In the following section, the density and the velocity components should be understood as Reynolds-/Favre-averaged, although the corresponding notation is omitted for convenience.

7.2.1 Spalart-Allmaras one-equation model

The Spalart-Allmaras one-equation turbulence model [47] employs transport equation for an eddy-viscosity variable $\tilde{\nu}$. It was developed based on empiricism, dimensional analysis, and Galilean invariance. It was calibrated using results for 2-D mixing layers, wakes, and flat-plate boundary layers. The Spalart-Allmaras model also allows for reasonably accurate predictions of turbulent flows with adverse pressure gradients. Furthermore, it is capable of smooth transition from laminar to turbulent flow at user specified locations. The Spalart-Allmaras model has several favorable numerical features. It is “local,” which means that the equation at one point does not depend on the solution at other points. Therefore, it can be readily implemented on structured multi-block or on unstructured grids. It is also robust, converges fast to steady-state, and requires only moderate grid resolution in the near-wall region.

Differential form

The Spalart-Allmaras turbulence model can be written in tensor notation as follows [47]

$$\begin{aligned} \frac{\partial \tilde{v}}{\partial t} + \frac{\partial}{\partial x_j} (\tilde{v} v_j) &= C_{b1}(1 - f_{t2}) \tilde{S} \tilde{v} \\ &+ \frac{1}{\sigma} \left\{ \frac{\partial}{\partial x_j} \left[(v_L + \tilde{v}) \frac{\partial \tilde{v}}{\partial x_j} \right] + C_{b2} \frac{\partial \tilde{v}}{\partial x_j} \frac{\partial \tilde{v}}{\partial x_j} \right\} \\ &- \left[C_w f_w - \frac{C_{b1}}{\kappa^2} f_{t2} \right] \left(\frac{\tilde{v}}{d} \right)^2 + f_{t1} \|\Delta \vec{v}\|_2^2. \end{aligned} \quad (7.36)$$

The terms on the right-hand side represent eddy-viscosity production, conservative diffusion, non-conservative diffusion, near-wall turbulence destruction, transition damping of production, and transition source of turbulence. Furthermore, $v_L = \mu_L / \rho$ denotes the laminar kinematic viscosity and d is the distance to the closest wall (for the computation of wall distances see Ref. [48]). The turbulent eddy viscosity in Eqs. (7.28) and (7.29) is obtained from

$$\mu_T = f_{v1} \rho \tilde{v}. \quad (7.37)$$

The production term is evaluated with the following formulas

$$\begin{aligned} \tilde{S} &= f_{v3} S + \frac{\tilde{v}}{\kappa^2 d^2} f_{v2}, \\ f_{v1} &= \frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad f_{v2} = \left(1 + \frac{\chi}{C_{v2}} \right)^{-3}, \\ f_{v3} &= \frac{(1 + \chi f_{v1})(1 - f_{v2})}{\max(\chi, 0.001)}, \quad \chi = \frac{\tilde{v}}{v_L}. \end{aligned} \quad (7.38)$$

where S stands for the magnitude of the mean rotation rate, that is,

$$S = \sqrt{2 \Omega_{ij} \Omega_{ij}},$$

where Ω_{ij} is given by Eq. (7.4). Note that \tilde{S} differs from its original definition in Ref. [47]. The modification was suggested by Spalart in order to prevent \tilde{S} from reaching zero (cf. Ref. [49], p. 155).

The terms controlling the destruction of the eddy viscosity read

$$\begin{aligned} f_w &= g \left(\frac{1 + C_{w3}^6}{g^6 + C_{w3}^6} \right)^{1/6}, \\ g &= r + C_{w2}(r^6 - r), \quad r = \frac{\tilde{v}}{\tilde{S} \kappa^2 d^2}. \end{aligned} \quad (7.39)$$

Functions used for modeling the laminar-turbulent transition are given by

$$\begin{aligned} f_{t1} &= g_t C_{t1} \exp \left(-C_{t2} \frac{\omega_t^2}{\Delta U^2} (d^2 + g_t^2 d_t^2) \right), \\ f_{t2} &= C_{t3} \exp (-C_{t4} \chi^2), \quad g_t = \min [0.1, \|\Delta \vec{v}\|_2 / (\omega_t \Delta x_t)], \end{aligned} \quad (7.40)$$

where ω_t represents the vorticity at the wall at the trip point (position has to be specified by the user), $\|\Delta \vec{v}\|_2$ denotes the 2-norm of the difference between the velocity at the trip point and the current field point, d_t is the distance to the nearest trip point, and Δx_t stands for the spacing along the wall at the trip point.

Finally, the various constants in Eqs. (7.36)–(7.40) are defined as

$$\begin{aligned} C_{b1} &= 0.1355, \quad C_{b2} = 0.622, \\ C_{\nu 1} &= 7.1, \quad C_{\nu 2} = 5, \quad \sigma = 2/3, \quad \kappa = 0.41, \\ C_{w1} &= C_{b1}/\kappa^2 + (1 + C_{b2})/\sigma, \quad C_{w2} = 0.3, \quad c_{w3} = 2, \\ C_{t1} &= 1, \quad C_{t2} = 2, \quad C_{t3} = 1.3, \quad C_{t4} = 0.5. \end{aligned} \quad (7.41)$$

In order to account for non-equilibrium effects on the production term, it was proposed to express C_{b1} as a function of the strain rate [50].

As pointed out in Ref. [47], it is convenient to substitute the non-conservative diffusion term in Eq. (7.36), that is,

$$\frac{1}{\sigma} \left\{ \frac{\partial}{\partial x_j} \left[(\nu_L + \tilde{\nu}) \frac{\partial \tilde{\nu}}{\partial x_j} \right] + C_{b2} \frac{\partial \tilde{\nu}}{\partial x_j} \frac{\partial \tilde{\nu}}{\partial x_j} \right\}$$

by the following expression

$$\frac{1 + C_{b2}}{\sigma} \frac{\partial}{\partial x_j} \left[(\nu_L + \tilde{\nu}) \frac{\partial \tilde{\nu}}{\partial x_j} \right] - \frac{C_{b2}}{\sigma} (\nu_L + \tilde{\nu}) \nabla^2 \tilde{\nu}. \quad (7.42)$$

In this way, difficulties with the discretization of the term $(\partial \tilde{\nu} / \partial x_j)^2$ are circumvented.

Integral form

The Spalart–Allmaras turbulence model (Eq. (7.36)) reads after the transformation into the finite-volume framework as follows

$$\frac{\partial}{\partial t} \int_{\Omega} \tilde{\nu} \, d\Omega + \oint_{\partial\Omega} (F_{c,T} - F_{v,T}) \, dS = \int_{\Omega} Q_T \, d\Omega, \quad (7.43)$$

where Ω represents the control volume, $\partial\Omega$ its surface, and dS is a surface element of Ω . The convective flux is defined as

$$F_{c,T} = \tilde{\nu} V \quad (7.44)$$

with V being the contravariant velocity (see Eq. (2.22)). The convective flux is in general discretized using a first-order upwind scheme. The viscous flux is given by

$$F_{v,T} = n_x \tau_{xx}^T + n_y \tau_{yy}^T + n_z \tau_{zz}^T, \quad (7.45)$$

where n_x , n_y , and n_z are the components of the unit normal vector. The normal viscous stresses read

$$\begin{aligned} \tau_{xx}^T &= \frac{1}{\sigma} (v_L + \tilde{v}) \frac{\partial \tilde{v}}{\partial x}, & \tau_{yy}^T &= \frac{1}{\sigma} (v_L + \tilde{v}) \frac{\partial \tilde{v}}{\partial y}, \\ \tau_{zz}^T &= \frac{1}{\sigma} (v_L + \tilde{v}) \frac{\partial \tilde{v}}{\partial z}. \end{aligned} \quad (7.46)$$

Finally, the source term in Eq. (7.43) becomes

$$\begin{aligned} Q_T &= C_{b1}(1 - f_{t2}) \tilde{S} \tilde{v} + \frac{C_{b2}}{\sigma} \left[\left(\frac{\partial \tilde{v}}{\partial x} \right)^2 + \left(\frac{\partial \tilde{v}}{\partial y} \right)^2 + \left(\frac{\partial \tilde{v}}{\partial z} \right)^2 \right] \\ &\quad - \left[C_{w1} f_w - \frac{C_{b1}}{\kappa^2} f_{t2} \right] \left(\frac{\tilde{v}}{d} \right)^2 + f_{t1} \|\Delta \tilde{v}\|_2^2. \end{aligned} \quad (7.47)$$

Alternatively, the non-conservative diffusion can be formulated as suggested by Eq. (7.42). The model constants are provided in Eq. (7.41).

Initial and boundary conditions

The initial value of \tilde{v} is usually taken as $\tilde{v} = 0.1 v_L$. The same value is also specified at inflow boundaries. At outflow boundaries, \tilde{v} is simply extrapolated from the interior of the computational domain. At solid walls, it is appropriate to set $\tilde{v} = 0$ and hence $\mu_T = 0$.

7.2.2 $K - \varepsilon$ two-equation model

The $K - \varepsilon$ turbulence model is likely the most widely employed two-equation eddy-viscosity model. It is based on the solution of equations for the turbulent kinetic energy K and the turbulent dissipation rate ε . The historic roots of the $K - \varepsilon$ model reach to the work of Chou [51]. During the 1970s, various formulations of the model were proposed. The most important contributions were due to Jones and Launder [52, 53], Launder and Sharma [54] as well as due to Launder and Spalding [55].

The $K - \varepsilon$ turbulence model requires addition of the so-called *damping functions* in order to stay valid through the viscous sublayer to the wall. The aim of the damping functions is to assure proper limiting behavior of K and ε at the wall, that is,

$$K \sim \gamma^2 \quad \text{and} \quad \frac{\varepsilon}{K} \sim \frac{2\nu}{\gamma^2} \quad \text{for } \gamma \rightarrow 0, \quad (7.48)$$

where γ represents the coordinate normal to the wall. Further, it can be shown that the Reynolds shear stress behaves like (see, e.g., [23, p. 138, 139])

$$\tau_{ij}^R \sim \gamma^3 \quad \text{for } \gamma \rightarrow 0, \quad i \neq j. \quad (7.49)$$

The $K - \varepsilon$ models with damping functions are also denoted as *low Reynolds number* models. The most widely used formulations of the damping functions were proposed by Jones and Launder [52], Launder and Sharma [54], Lam and Bremhorst [56], and by Chien [57]. The reader may find a comparison of seven different low Reynolds number $K - \varepsilon$ models in Ref. [58].

The $K - \varepsilon$ turbulence model is more difficult to solve numerically than the previously discussed Spalart-Allmaras model (Section 7.2.1). Particularly, the damping functions lead to turbulence equations with stiff source terms. This, and the necessary high grid resolution near walls (in order to resolve the viscous sublayer), require the utilization of at least point-implicit, or better, full-implicit time-stepping schemes. Reference [59] contains useful hints on the explicit time discretization of the $K - \varepsilon$ equations. Examples of implementations of the $K - \varepsilon$ model on structured as well as on unstructured grids can be found, for example, in Refs. [60–74]. Finally, it is important to note that the accuracy of the $K - \varepsilon$ model degrades for flows with an adverse pressure gradient [23, 58]. An interesting study of how the model parameters influence the accuracy for various pressure gradients was published in Ref. [75].

Differential form

A low Reynolds number $K - \varepsilon$ model can be written as

$$\begin{aligned} \frac{\partial \rho K}{\partial t} + \frac{\partial}{\partial x_j} (\rho v_j K) &= \frac{\partial}{\partial x_j} \left[\left(\mu_L + \frac{\mu_T}{\sigma_K} \right) \frac{\partial K}{\partial x_j} \right] + \tau_{ij}^F S_{ij} - \rho \varepsilon \\ \frac{\partial \rho \varepsilon^*}{\partial t} + \frac{\partial}{\partial x_j} (\rho v_j \varepsilon^*) &= \frac{\partial}{\partial x_j} \left[\left(\mu_L + \frac{\mu_T}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon^*}{\partial x_j} \right] + C_{\varepsilon 1} f_{\varepsilon 1} \frac{\varepsilon^*}{K} \tau_{ij}^F S_{ij} \\ &\quad - C_{\varepsilon 2} f_{\varepsilon 2} \rho \frac{(\varepsilon^*)^2}{K} + \phi_\varepsilon. \end{aligned} \quad (7.50)$$

The terms on the right-hand side represent conservative diffusion, eddy-viscosity production, and dissipation, respectively. Furthermore, ϕ_ε denotes the so-called explicit wall term. The Favre-averaged turbulent stresses τ_{ij}^F are given by Eq. (7.25) and the strain-rate tensor S_{ij} follows from Eq. (7.3). The turbulent eddy viscosity in Eqs. (7.28) and (7.29) results from

$$\mu_T = C_{\mu} f_\mu \rho \frac{K^2}{\varepsilon^*}. \quad (7.51)$$

The turbulent kinetic energy is also employed for the evaluation of the eddy viscosity according to Eq. (7.24) or Eq. (7.25). The quantity ε^* is related to the turbulent dissipation rate ε by

$$\varepsilon = \varepsilon_w + \varepsilon^*. \quad (7.52)$$

The term ε_w is the value of the dissipation rate at the wall. The definition in Eq. (7.52) greatly simplifies the application of wall-boundary conditions (see further below).

The constants, the near-wall damping functions as well as the wall term differ between the various $K - \varepsilon$ models. Here, we chose the Launder-Sharma model because it gives good results for a wide range of applications [58]. For the Launder-Sharma model, the constants and the turbulent Prandtl number are given by [54]

$$\begin{aligned} C_\mu &= 0.09, & C_{\varepsilon 1} &= 1.44, & C_{\varepsilon 2} &= 1.92, \\ \sigma_K &= 1.0, & \sigma_\varepsilon &= 1.3, & Pr_T &= 0.9. \end{aligned} \quad (7.53)$$

Furthermore, the near-wall damping functions read

$$\begin{aligned} f_\mu &= \exp\left(\frac{-3.4}{(1 + 0.02 Re_T)^2}\right) \\ f_{\varepsilon 1} &= 1 \\ f_{\varepsilon 2} &= 1 - 0.3 \exp(Re_T^2) \end{aligned} \quad (7.54)$$

with $Re_T = \rho K^2 / (\varepsilon^* \mu_L)$ being the turbulent Reynolds number. Finally, the explicit wall term ϕ_ε and the value ε_w are defined as

$$\phi_\varepsilon = 2\mu_T \frac{\mu_L}{\rho} \left(\frac{\partial^2 v_s}{\partial \gamma_n^2} \right)^2 \quad \text{and} \quad \varepsilon_w = \frac{2\mu_L}{\rho} \left(\frac{\partial \sqrt{K}}{\partial \gamma_n} \right)^2, \quad (7.55)$$

where v_s stands for the velocity parallel to the wall, and γ_n represents the coordinate normal to the wall. In order to avoid an explicit knowledge of the wall distance and orientation, it is common to compute the wall term and ε_w from the following Cartesian tensor form [66, 76]

$$\phi_\varepsilon = 2\mu_T \frac{\mu_L}{\rho} \left(\frac{\partial^2 v_i}{\partial x_j \partial x_k} \right)^2 \quad \text{and} \quad \varepsilon_w = \frac{2\mu_L}{\rho} \left(\frac{\partial \sqrt{K}}{\partial x_j} \right)^2 \quad (7.56)$$

instead of Eq. (7.55).

Integral form

Written in time-dependent integral form for a control volume Ω with a surface element dS , the low Reynolds number $K - \varepsilon$ turbulence model reads

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W}_T d\Omega + \oint_{\partial\Omega} (\vec{F}_{c,T} - \vec{F}_{v,T}) dS = \int_{\Omega} \vec{Q}_T d\Omega. \quad (7.57)$$

The vector of the conservative variables takes the form

$$\vec{W}_T = \begin{bmatrix} \rho K \\ \rho \varepsilon^* \end{bmatrix}. \quad (7.58)$$

The vector of the convective fluxes is defined

$$\vec{F}_{c,T} = \begin{bmatrix} \rho KV \\ \rho \varepsilon^* V \end{bmatrix}, \quad (7.59)$$

where V denotes the contravariant velocity (see Eq. (2.22)). The vector of the viscous fluxes is given by

$$\vec{F}_{v,T} = \begin{bmatrix} n_x \tau_{xx}^K + n_y \tau_{yy}^K + n_z \tau_{zz}^K \\ n_x \tau_{xx}^\varepsilon + n_y \tau_{yy}^\varepsilon + n_z \tau_{zz}^\varepsilon \end{bmatrix} \quad (7.60)$$

with the normal turbulent viscous stresses

$$\begin{aligned} \tau_{xx}^K &= \left(\mu_L + \frac{\mu_T}{\sigma_K} \right) \frac{\partial K}{\partial x}, \dots \\ \tau_{xx}^\varepsilon &= \left(\mu_L + \frac{\mu_T}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon^*}{\partial x}, \dots \end{aligned} \quad (7.61)$$

In Eq. (7.60), n_x , n_y , n_z represent the components of the outward-facing unit normal vector of the surface $\partial\Omega$. The source term is evaluated from

$$\vec{Q}_T = \begin{bmatrix} P - \rho \varepsilon \\ (C_{\varepsilon 1} f_{\varepsilon 1} P - C_{\varepsilon 2} f_{\varepsilon 2} \rho \varepsilon^*) \frac{\varepsilon^*}{K} + \phi_\varepsilon \end{bmatrix}, \quad (7.62)$$

where P denotes the production term of the turbulent kinetic energy. It is defined as

$$\begin{aligned} P = & \tau_{xx}^F \frac{\partial u}{\partial x} + \tau_{yy}^F \frac{\partial v}{\partial y} + \tau_{zz}^F \frac{\partial w}{\partial z} \\ & + \tau_{xy}^F \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) + \tau_{xz}^F \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) + \tau_{yz}^F \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \end{aligned} \quad (7.63)$$

with the Favre-averaged turbulent stresses τ_{ij}^F given by Eq. (7.25). The constants, the near-wall damping functions as well as the wall term follow for the Launder-Sharma model from the definitions in Eqs. (7.52)–(7.56). The turbulent eddy viscosity μ_T is obtained from Eq. (7.51).

Initial and boundary conditions

The simplest approach is to initialize K and ε^* with their free-stream values. A better alternative consists of prescribing profiles for K and ε^* near solid walls. The profiles can be obtained from analogy to turbulent flat-plate boundary layer [60]. However, this requires the knowledge of wall distances which may not be readily available like it is the case on unstructured grids.

The proper boundary conditions at solid walls are $K = 0$ and $\varepsilon^* = 0$, provided the transformation in Eq. (7.52) is utilized. This also implies $\mu_T = 0$ at walls. At inflow boundaries, K and ε^* can be computed from relations for the turbulent intensity and length scale, that is,

$$(T_u)_\infty = \sqrt{\frac{2}{3} K_\infty}, \quad (l_T)_\infty = \frac{C_\mu K_\infty^{3/2}}{\varepsilon_\infty^*}, \quad (7.64)$$

where we assumed $\varepsilon_\infty^* = \varepsilon_\infty$. In turbomachinery, $(l_T)_\infty$ is chosen between 10^{-3} and 10^{-2} times the mean radial blade spacing [77]. The values of K and ε^* are extrapolated from the interior at outflow boundaries.

Wall functions

As we already noted, the low Reynolds number models require very fine grids at walls. The standard condition is that the first node (or cell centroid) should be located at the distance $y^+ \leq 1$ from the wall. In order to reduce the stiffness of the turbulence equations and to save a number of grid points/cells, coarser grids with $10 \leq y^+ \leq 100$ are often employed. In such a case, the $K - \varepsilon$ model Eq. (7.50) or (7.57) is applied without the damping functions ($f_\mu = f_{\varepsilon 1} = f_{\varepsilon 2} = 1$; $\varepsilon_w = 0$) and the wall term ($\phi_\varepsilon = 0$). We speak here of a *high Reynolds number* turbulence model. Apparently, the distance between the first node (cell centroid) and the wall has to be bridged by the *wall functions*. The wall functions deliver the values of K and ε^* at the node (cell centroid) adjacent to the wall. The turbulence equations are not solved at the wall itself and at the first layer of nodes (cells). Various formulation of the wall functions are used, in general, based on the logarithmic wall-law. One example is the function of Spalding [78], which models the viscous sublayer, the transition region as well as the logarithmic layer. Implementations of high Reynolds number models were described, for example, in Refs. [79–85] or in Ref. [67].

The application of the wall functions leads (provided the grid is not too coarse) to reasonably accurate results for attached boundary layers. It also allows the utilization of purely explicit time-stepping schemes. However, the use of wall functions becomes highly questionable for separated flows.

7.2.3 SST two-equation model of Menter

The $K - \omega$ SST turbulence model of Menter [86, 87] merges the $K - \omega$ model of Wilcox [23, 88] with a high Reynolds number $K - \varepsilon$ model (transformed into the $K - \omega$ formulation). The SST model seeks to combine the positive features of both models. Therefore, the $K - \omega$ approach is employed in the sublayer of the boundary layer. The reason is that the $K - \omega$ model needs no damping function. This leads, for similar accuracy, to significantly higher numerical stability in comparison to the $K - \varepsilon$ model. Furthermore, the $K - \omega$ model is also utilized in logarithmic part of the boundary layer, where it is superior to the $K - \varepsilon$ approach in adverse pressure flows and in compressible flows. On the other hand, the $K - \varepsilon$ model is employed in the wake region of the boundary layer because the $K - \omega$ model is strongly sensitive to the free-stream value of ω [89]. The $K - \varepsilon$ approach is also used in free shear layers since it represents a fair compromise in accuracy for wakes, jets, and mixing layers.

One distinct feature of the SST turbulence model is the modified turbulent eddy-viscosity function. The purpose is to improve the accuracy of prediction of flows with strong adverse pressure gradients and of pressure-induced boundary layer separation. The modification accounts for the transport of the turbulent shear stress. It is based on the observation of Bradshaw that the principal shear stress is proportional to the turbulent kinetic energy.

A certain disadvantage of the SST model is that distances to the nearest wall have to be known explicitly. This requires special provisions on multiblock structured or on unstructured grids. See, for example, Ref. [48] for the computation of wall distances. Examples of applications of the SST turbulence model can be found in Refs. [90–97].

Differential form

The transport equations for the turbulent kinetic energy and the specific dissipation of turbulence read in differential form [86]

$$\begin{aligned} \frac{\partial \rho K}{\partial t} + \frac{\partial}{\partial x_j} (\rho v_j K) &= \frac{\partial}{\partial x_j} \left[(\mu_L + \sigma_K \mu_T) \frac{\partial K}{\partial x_j} \right] + \tau_{ij}^F S_{ij} - \beta^* \rho \omega K, \\ \frac{\partial \rho \omega}{\partial t} + \frac{\partial}{\partial x_j} (\rho v_j \omega) &= \frac{\partial}{\partial x_j} \left[(\mu_L + \sigma_\omega \mu_T) \frac{\partial \omega}{\partial x_j} \right] + \frac{C_\omega \rho}{\mu_T} \tau_{ij}^F S_{ij} \\ &\quad - \beta \rho \omega^2 + 2(1 - f_1) \frac{\rho \sigma_\omega^2}{\omega} \frac{\partial K}{\partial x_j} \frac{\partial \omega}{\partial x_j}. \end{aligned} \quad (7.65)$$

The terms on the right-hand side of Eq. (7.65) represent conservative diffusion, eddy-viscosity production, and dissipation, respectively. Furthermore, the last term in the ω -equation describes the cross diffusion. The Favre-averaged turbulent stresses τ_{ij}^F are given by Eq. (7.25) and the strain-rate tensor S_{ij} follows from Eq. (7.3). The turbulent eddy viscosity in Eqs. (7.28) and (7.29) is obtained from Ref. [86]

$$\mu_T = \frac{a_1 \rho K}{\max(a_1 \omega, f_2 \|\operatorname{curl} \vec{v}\|_2)}. \quad (7.66)$$

This definition of the turbulent viscosity guarantees that in an adverse pressure gradient boundary layer, where the production of K is larger than its dissipation ω (hence $a_1 \omega < \|\operatorname{curl} \vec{v}\|_2$), Bradshaw's assumption, that is, $\tau = a_1 \rho K$ (shear stress proportional to turbulent kinetic energy) is satisfied.

The function f_1 in Eq. (7.65), which blends the model coefficients of the $K - \omega$ model in boundary layers with the transformed $K - \varepsilon$ model in free-shear layers and free-stream zones, is defined as

$$\begin{aligned} f_1 &= \tan h(\operatorname{arg}_1^4), \\ \operatorname{arg}_1 &= \min \left[\max \left(\frac{\sqrt{K}}{0.09 \omega d}, \frac{500 \mu_L}{\rho \omega d^2} \right), \frac{4 \rho \sigma_\omega^2 K}{CD_{K\omega} d^2} \right], \end{aligned} \quad (7.67)$$

where d stands for the distance to the nearest wall and $CD_{K\omega}$ is the positive part of the cross-diffusion term in Eq. (7.65), that is,

$$CD_{K\omega} = \max \left(2 \frac{\rho \sigma_{\omega 2}}{\omega} \frac{\partial K}{\partial x_j} \frac{\partial \omega}{\partial x_j}, 10^{-20} \right). \quad (7.68)$$

The auxiliary function f_2 in Eq. (7.66) is given by

$$\begin{aligned} f_2 &= \tanh(\arg_2^2), \\ \arg_2 &= \max \left(\frac{2\sqrt{K}}{0.09\omega d}, \frac{500\mu_L}{\rho\omega d^2} \right). \end{aligned} \quad (7.69)$$

The model constants are as follows

$$a_1 = 0.31, \quad \beta^* = 0.09, \quad \kappa = 0.41. \quad (7.70)$$

Finally, the coefficients of the SST turbulence model β , C_ω , σ_K , and σ_ω are obtained by blending the coefficients of the $K - \omega$ model, denoted as ϕ_1 , with those of the transformed $K - \varepsilon$ model (ϕ_2). The corresponding relation reads

$$\phi = f_1 \phi_1 + (1 - f_1) \phi_2. \quad (7.71)$$

The coefficients of the inner model ($K - \omega$) are given by

$$\begin{aligned} \sigma_{K1} &= 0.85, \quad \sigma_{\omega 1} = 0.5, \quad \beta_1 = 0.075, \\ C_{\omega 1} &= \beta_1 / \beta^* - \sigma_{\omega 1} \kappa^2 / \sqrt{\beta^*} = 0.533. \end{aligned} \quad (7.72)$$

The coefficients of the outer model ($K - \varepsilon$) are defined as

$$\begin{aligned} \sigma_{K2} &= 1.0, \quad \sigma_{\omega 2} = 0.856, \quad \beta_2 = 0.0828, \\ C_{\omega 2} &= \beta_2 / \beta^* - \sigma_{\omega 2} \kappa^2 / \sqrt{\beta^*} = 0.440. \end{aligned} \quad (7.73)$$

The integral formulation of the SST turbulence model corresponds, in principle, to that of the $K - \varepsilon$ model from Section 7.2.2. It is therefore not repeated here.

Boundary conditions

The boundary conditions for the kinetic turbulent energy and the specific dissipation at solid walls are

$$K = 0 \quad \text{and} \quad \omega = 10 \frac{6\mu_L}{\rho\beta_1(d_1)^2} \quad (7.74)$$

with d_1 being the distance of the first node (cell centroid) from the wall. The grid has to be refined such that $y^+ < 3$ is satisfied.

For the inflow boundaries, the following free-stream values are recommended

$$\omega_\infty = C_1 \frac{\|\vec{v}_\infty\|_2}{L}, \quad (\mu_T)_\infty = (\mu_L)_\infty 10^{-C_2}, \quad K_\infty = \frac{(\mu_T)_\infty}{\rho_\infty} \omega_\infty, \quad (7.75)$$

where L denotes the length of the computational domain, $1 \leq C_1 \leq 10$ and $2 \leq C_2 \leq 5$, respectively. The values of K and ω are extrapolated from the interior at outflow boundaries.

7.3 LARGE-EDDY SIMULATION

The so-called LES methodology was employed already in 1963 by Smagorinsky in meteorology [98] (circulation of the atmosphere). The first engineering application of LES (turbulent channel flow) was presented by Deardorff in 1970 [99]. His method was later extended and improved by Schumann [100]. During 1980s, the research focus in the simulation of turbulence shifted from LES to DNS. However, some important work was still conducted, for example, by Bardina et al. [101], Moin and Kim [102]. The interest in LES returned back at the beginning of 1990s [103–109]. Nowadays, LES is increasingly employed for physically and geometrically complex flows of engineering relevance like, for example, in combustion chambers. Examples can be found in Refs. [110, 110–128]. Certainly, this trend is supported by the availability of low-cost, highly powerful compute servers. Additionally, today's engineers are also often faced with flow problems, for which the standard turbulence models fail. Furthermore, in certain cases the mean flow frequencies are in the same order as the turbulent fluctuations. Hence, the time averaging loses its validity and we have to resort to either LES or to DNS.

LES is based on the observation that the small turbulent structures are more universal in character than the large eddies. Therefore, the idea is to compute the contributions of the large, energy-carrying structures to momentum and energy transfer and to model the effects of the small structures, which are not resolved by the numerical scheme. Due to the more homogeneous and universal character of the small scales, we may expect that the so called *subgrid-scale* models can be kept much simpler than the turbulence models for the RANS equations.

LES represents a 3-D, time-dependent solution of the governing equations. In comparison to turbulence modeling based on the RANS equations, LES requires high grid resolution also in the stream-wise ($50 \leq x^+ \leq 150$) and in the cross-flow direction ($15 \leq z^+ \leq 40$). However, LES is computationally considerably cheaper than DNS. The number of grid points (cells) required to resolve the outer layer is proportional to $Re^{0.4}$ [129]. The resolution has to be increased like $Re^{1.8}$ in the viscous sublayer. Thus, if compared to $Re^{9/4}$ required by DNS, LES can be applied at Reynolds numbers at least one order of magnitude higher. In order to further reduce the requirements on grid resolution, LES can be used in conjunction with approximate wall models (Section 7.3.4), or coupled with a RANS model (Section 7.3.5). Both approaches allow for LES of engineering problems at reasonable computational costs.

An accurate resolution of high wave-number turbulent fluctuations requires spatial discretization schemes with corresponding properties in the wave-number space

(cf., e.g., [130] or [131]). Therefore, spectral methods are often employed. However, spectral methods are applicable only to geometrically simple domains with (quasi-)periodic boundaries. This is the reason why finite-difference or finite-volume spatial discretizations are becoming increasingly popular. Central differencing schemes proved to be more suitable than upwind schemes. The reason is that upwind schemes (regardless of the order of accuracy) dissipate too much energy over a significant portion of the turbulent spectra due to the inherent numerical damping [132, 133]. A discussion of numerical errors of various discretization methods can be found in Refs. [134–136].

The implementation of LES methods on unstructured grids [137–147] represents a particular challenge. However, it allows for the treatment of highly complex geometries, moving boundaries or for dynamic grid adaptation. The research topic consists of the development of numerically efficient, high-order spatial discretization on mixed-element grids.

An introduction to LES can be found in Ref. [25, pp. 269–336], and in Refs. [148–152] or [113]. An overview of the state of LES research is provided in Refs. [153–155].

7.3.1 Spatial filtering

LES is based on a spatial filtering operation, which decomposes any flow variable U into a filtered (large-scale, resolved) part \bar{U} and into a sub-filter (unresolved) part U' , that is,

$$U = \bar{U} + U' \quad (7.76)$$

The filtered variable at the location \vec{r}_0 in space is defined as

$$\bar{U}(\vec{r}_0, t) = \int_D U(\vec{r}, t) G(\vec{r}_0, \vec{r}, \Delta) d\vec{r}, \quad (7.77)$$

where Ω denotes the entire flow domain, G represents the filter function, and \vec{r} is the position vector, respectively. The filter function determines the structure and size of the small scales. The filter function depends on the difference $\vec{r}_0 - \vec{r}$ and on the filter width $\Delta = (\Delta_1 \Delta_2 \Delta_3)^{1/3}$, with Δ_i being the filter width in the i th spatial coordinate. The following filter functions are the mostly used ones (see Fig. 7.3):

1. Tophat filter:

$$G = \begin{cases} 1/\Delta^3 & \text{if } |(x_0)_i - x_i| \leq \Delta_i/2 \\ 0 & \text{otherwise.} \end{cases} \quad (7.78)$$

2. Sharp Fourier cut-off filter:

$$G = \prod_{i=1}^3 \frac{\sin\left(\frac{\pi}{\Delta_i}[(x_0)_i - x_i]\right)}{\pi [(x_0)_i - x_i]}. \quad (7.79)$$

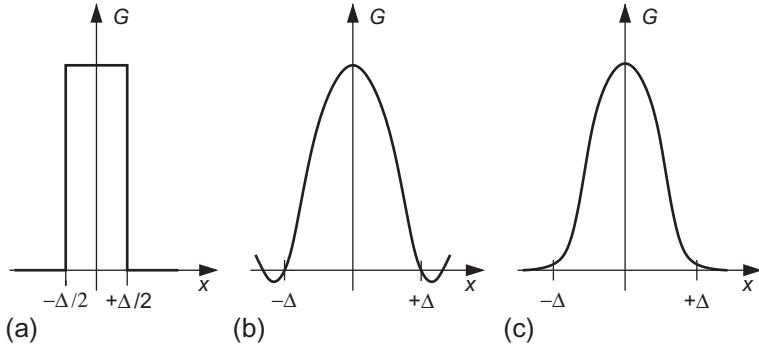


Figure 7.3 LES filter functions in physical space: tophat (a), cut-off (b), and Gaussian (c).

3. Gaussian filter:

$$G = \left(\frac{6}{\pi \Delta^2} \right)^{3/2} \exp \left(\frac{-6 \|\vec{r}_0 - \vec{r}\|_2^2}{\Delta^2} \right). \quad (7.80)$$

The tophat and the Gaussian filter smooth the large-scale fluctuations as well as the small scales below the filter width. The cut-off filter affects only the scales below the cut-off wave-number. In practice, the Gaussian filter is always employed in conjunction with a sharp Fourier cut-off. Filters suitable for grids with varying cell sizes were proposed in Refs. [156, 157].

7.3.2 Filtered governing equations

The spatial filtering, defined by Eqs. (7.76) and (7.77), has to be applied to the Navier-Stokes equations in order to remove the small turbulent scales. The filter width Δ as well as the filter function are considered as free parameters. In fact, the governing equations are usually not explicitly filtered. Instead, the grid as well as the discretization errors are assumed to define the filter G . For the discussion of explicit filtering see Refs. [158, 159].

Because of the differing treatment, we shall distinguish in the following between compressible (7.1) and incompressible (7.6) formulation of the Navier-Stokes equations.

Incompressible Navier-Stokes equations

For an incompressible flow of a Newtonian fluid, the filtered governing equation (7.6) take the form

$$\begin{aligned} \frac{\partial \bar{v}_i}{\partial x_i} &= 0, \\ \frac{\partial \bar{v}_i}{\partial t} + \frac{\partial}{\partial x_j} (\bar{v}_i \bar{v}_j) &= -\frac{1}{\rho} \frac{\partial \bar{p}}{\partial x_i} + \nu \nabla^2 \bar{v}_i - \frac{\partial \tau_{ij}^S}{\partial x_j}, \end{aligned} \quad (7.81)$$

where ν denotes the kinematic viscosity coefficient. The equations (7.81) describe the temporal and spatial evolution of the large, energy-carrying scales of motion. The non-linearity of the convective term leads to the appearance of the so-called *subgrid-scale stress* (SGS) tensor

$$\tau_{ij}^S = \bar{v}_i \bar{v}_j - \bar{v}_i \bar{v}_j, \quad (7.82)$$

which describes the effects of the unresolved scales. The SGS tensor has to be modeled (see [Section 7.3.3](#)) in order to close the equations.

The SGS tensor can be decomposed into three parts [160], namely

$$\tau_{ij}^S = L_{ij} + C_{ij} + \tau_{ij}^{SR}. \quad (7.83)$$

The individual parts have the following physical meaning:

$$L_{ij} = \overline{\bar{v}_i \bar{v}_j} - \bar{v}_i \bar{v}_j \quad (7.84)$$

is the so-called *Leonard stress* term and represents the interactions between large-scale eddies which produce small-scale turbulence. This term only can be evaluated explicitly from the filtered velocity field v_i . Further, the cross-stress term

$$C_{ij} = \overline{\bar{v}_i v'_j} + \overline{v'_i \bar{v}_j} \quad (7.85)$$

describes interactions between large- and small-scale eddies. Finally,

$$\tau_{ij}^{SR} = \overline{v'_i v'_j} \quad (7.86)$$

is the so-called *SGS Reynolds-stress* tensor. It reflects interactions between the small-scale structures. The above decomposition (7.83) is no longer used mainly because L_{ij} and C_{ij} are not invariant with respect to Galilean transformation.¹

Compressible Navier-Stokes equations

If LES is to be applied to compressible flows, we have to apply Favre averaging ([Section 7.1.2](#)) together with the spatial filtering to Eq. (7.1). Otherwise, the filtered Navier-Stokes equations would contain products between density and other variables like velocity or temperature. Thus, the velocity components, the energy and the temperature in Eq. (7.1) is decomposed as

$$U = \tilde{U} + U''. \quad (7.87)$$

The filtered variable at the location \vec{r}_0 in space is given by

$$\tilde{U}(\vec{r}_0, t) = \frac{\overline{\rho U}}{\overline{\rho}} = \frac{1}{\overline{\rho}} \int_D \rho(\vec{r}, t) U(\vec{r}, t) G(\vec{r}_0, \vec{r}, \Delta) d\vec{r}, \quad (7.88)$$

¹ Galilean invariance means that all frames of reference which are translating uniformly with respect to each other are equivalent.

where the bar denotes filtering according to the Eq. (7.77). The Favre-filtered Navier-Stokes equations (7.1) read [149, 153]

$$\begin{aligned} \frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{v}_j) &= 0, \\ \frac{\partial \bar{\rho} \tilde{v}_i}{\partial t} + \frac{\partial (\bar{\rho} \tilde{v}_j \tilde{v}_i)}{\partial x_j} + \frac{\partial \bar{p}}{\partial x_i} - \frac{\partial \hat{\sigma}_{ij}}{\partial x_j} &= -\frac{\partial \tau_{ij}^{SF}}{\partial x_j} + \frac{\partial}{\partial x_j} (\bar{\sigma}_{ij} - \hat{\sigma}_{ij}), \\ \frac{\partial \bar{\rho} \tilde{e}}{\partial t} + \frac{\partial (\bar{\rho} \tilde{v}_j \tilde{e})}{\partial x_j} + \frac{\partial \hat{q}}{\partial x_j} + \bar{p} \tilde{S}_{kk} - \hat{\sigma}_{ij} \tilde{S}_{ij} &= -\mathcal{A} - \mathcal{B} - \mathcal{C} + \mathcal{D} \end{aligned} \quad (7.89)$$

with the terms

$$\begin{aligned} \mathcal{A} &= \frac{\partial}{\partial x_j} [\bar{\rho}(\tilde{v}_j e - \tilde{v}_j \tilde{e})] && \text{-- divergence of subgrid-scale heat flux,} \\ \mathcal{B} &= \frac{\partial}{\partial x_j} [\bar{q}_j - \hat{q}_j] && \text{-- divergence of SGS heat diffusion,} \\ \mathcal{C} &= [\bar{p} \tilde{S}_{kk} - \bar{p} \tilde{S}_{kk}] && \text{-- SGS pressure-dilatation,} \\ \mathcal{D} &= [\bar{\sigma}_{ij} \tilde{S}_{ij} - \hat{\sigma}_{ij} \tilde{S}_{ij}] && \text{-- SGS viscous dissipation} \end{aligned}$$

and

$$\begin{aligned} \bar{\sigma}_{ij} &= \overline{2\mu \tilde{S}_{ij}} + \overline{\left(\mu_B - \frac{2\mu}{3} \right) \delta_{ij} \tilde{S}_{kk}}, \\ \hat{\sigma}_{ij} &= 2\tilde{\mu} \tilde{S}_{ij} + \left(\tilde{\mu}_B - \frac{2\tilde{\mu}}{3} \right) \delta_{ij} \tilde{S}_{kk}, \\ \tilde{S}_{ij} &= \frac{1}{2} \left(\frac{\partial \tilde{v}_i}{\partial x_j} + \frac{\partial \tilde{v}_j}{\partial x_i} \right), \\ \bar{q}_j &= -k \frac{\partial \bar{T}}{\partial x_j}, \quad \tilde{q}_j = -\tilde{k} \frac{\partial \tilde{T}}{\partial x_j}. \end{aligned} \quad (7.90)$$

In Eqs. (7.89)–(7.90), e denotes internal energy per unit mass, \tilde{S}_{ij} is the Favre-filtered strain-rate tensor, and $\tau_{ij}^{SF} = \bar{\rho}(\tilde{v}_i \tilde{v}_j - \tilde{v}_i \tilde{v}_j)$ represents the Favre-averaged subgrid-scale stress. Furthermore, μ , μ_B , and k stand for the molecular viscosity, the bulk viscosity, and for the thermal conductivity, respectively. Finally, $\tilde{\mu}$, $\tilde{\mu}_B$, and \tilde{k} are the corresponding values at the filtered temperature \tilde{T} .

The right-hand side of Eq. (7.89) contains terms which have to be modeled. In the momentum equation, the SGS stresses τ_{ij}^{SF} are approximated, but the second term, that is, $(\bar{\sigma}_{ij} - \hat{\sigma}_{ij})$ is usually neglected. In the energy equation, term \mathcal{A} can be expressed through the SGS stresses [161], term \mathcal{B} can be neglected, and terms \mathcal{C} and \mathcal{D} can be modeled as proposed in Ref. [162].

7.3.3 Subgrid-scale modeling

The main task of a subgrid-scale model is to simulate energy transfer between the large and the subgrid scales. On the average, the energy is transported from the large scales to the small ones (turbulent cascade process). Therefore, a subgrid-scale model has to provide means of adequate energy dissipation. However, in some instances the energy also flows from the small to the large scales—a process called *backscatter*. Clearly, the model should account for this effect as well. Backscatter models are discussed, for example, in Ref. [163].

Various subgrid-scale models were proposed in the past and the research still continues. The models can be divided into two basic classes. The first one consists of approaches which model the SGS tensor τ_{ij}^S explicitly. A necessary condition is then that the numerical dissipation caused by the spatial discretization scheme must be much lower than the subgrid-scale dissipation. The majority of explicit SGS models is based on the eddy-viscosity concept, which is explained next. Furthermore, we shall present the Smagorinsky model, which forms the basis of all subgrid-scale models. We shall also briefly discuss the basics of the so-called *dynamic subgrid-scale* models. A comparison of six different explicit subgrid-scale models was presented recently in Ref. [164].

The second class of subgrid-scale models consists of approaches where the SGS stresses are modeled implicitly by an appropriate discretization of the convective fluxes (thus τ_{ij}^S is omitted). These models are referred to as *monotonically integrated LES* (MILES). The methodology was first presented by Boris et al. [165] and advocated by Grinstein and Fureby [166, 167]. The condition in this case is that the numerical dissipation correctly models the subgrid-scale dissipation. This is not quite easy to achieve, as the investigations in Refs. [168, 169] prove. Nevertheless, MILES was applied with some success to a variety of flow problems [170–177].

Eddy-viscosity models

These explicit models are able to represent the global dissipative effects of the small scales, but they cannot reproduce the local details of the energy exchange. In the case of incompressible flows, the eddy-viscosity models relate the SGS stresses to the large-scale strain-rate \bar{S}_{ij} as follows

$$\tau_{ij}^S - \frac{\delta_{ij}}{3} \tau_{kk}^S = -2\nu_T \bar{S}_{ij}. \quad (7.91)$$

The strain-rate \bar{S}_{ij} is obtained from Eq. (7.3) by using filtered velocity components. The eddy viscosity ν_T is in general evaluated from algebraic relations in order to save numerical costs. The isotropic part of the SGS stresses (τ_{kk}^S) can either be added to the filtered pressure [178], modeled [179], or neglected.

Relation similar to Eq. (7.91) applies also in the case of compressible Navier-Stokes equations. The components of the Favre-averaged SGS stress tensor are approximated as

$$\tau_{ij}^{SF} - \frac{\delta_{ij}}{3} \tau_{kk}^{SF} = -2\bar{\rho} v_T \tilde{S}_{ij} + \left(\frac{2\bar{\rho} v_T}{3} \right) \frac{\partial \tilde{v}_k}{\partial x_k} \delta_{ij}. \quad (7.92)$$

The components of the strain-rate tensor \tilde{S}_{ij} are given in Eq. (7.90).

Smagorinsky SGS model

The Smagorinsky model [98] is based on the equilibrium hypothesis which implies that the small scales dissipate entirely and instantaneously all the energy they receive from the large scales. The algebraic model assumes the form

$$v_T = (C_s \Delta)^2 |\bar{S}|, \quad (7.93)$$

where $|\bar{S}| = (2\bar{S}_{ij}\bar{S}_{ij})^{1/2}$ is the magnitude of the strain-rate tensor and C_s denotes the *Smagorinsky constant*. The theoretical value found by Lilly [180] is $C_s = 0.18$. However, the Smagorinsky constant depends on the type of the flow. For example, in shear flows C_s has to be reduced to approximately 0.1. The filter width Δ in Eq. (7.93) is usually chosen to be twice the average grid size, that is, $\Delta = 2(\Delta x_1 \Delta x_2 \Delta x_3)^{1/3}$.

In order to account for the reduced growth of the small scales near walls, the value of the eddy viscosity v_T has to be reduced. Thus, the Smagorinsky model Eq. (7.93) is modified according to Van Driest damping as

$$v_T = \left[C_s \Delta (1 - e^{-y^+/25}) \right]^2 |\bar{S}|, \quad (7.94)$$

where y^+ represents the dimensionless wall distance (for the computation of wall distances see, for example, Ref. [48]).

The Smagorinsky model is numerically cheap and easy to implement. However, it has several serious disadvantages:

- it is too dissipative in laminar regions with mean shear;
- it requires special provisions near walls and at laminar-turbulent transition;
- the parameter C_s is not uniquely defined;
- the process of energy backscatter is not modeled.

Because of these shortcomings, various other approaches were proposed (see, e.g., Ref. [149]). Very popular are the following dynamic models.

Dynamic SGS models

The dynamic SGS models employ the same relation as the Smagorinsky model (Eq. (7.93)) for the evaluation of the eddy viscosity v_T in Eq. (7.91) or (7.92). The difference is that the Smagorinsky constant (adjusted a priori) is replaced by a parameter, which evolves dynamically in space and in time. Hence,

$$v_T = C_d(\vec{r}, t) \Delta^2 |\bar{S}|. \quad (7.95)$$

The parameter C_d is computed based on the energy content of the smallest scale of the turbulence. For this purpose, Germano et al. [181] proposed to employ a second filter—the so-called *test filter* $\hat{\Delta}$. The width of the test filter has to be larger than that of the filter Δ applied to the governing equations (usually $\hat{\Delta} = 2\Delta$). The application of the test filter to the filtered equations leads to the so-called *subtest-scale stresses* τ_{ij}^{ST}

$$\tau_{ij}^{ST} = \widehat{\bar{v}_i \bar{v}_j} - \widehat{\bar{v}_i} \widehat{\bar{v}_j}. \quad (7.96)$$

The subtest-scale stresses are related to the SGS stresses τ_{ij}^S (Eq. (7.83)) via the *Germano identity* [182]

$$\hat{L}_{ij} = \tau_{ij}^{ST} - \hat{\tau}_{ij}^S = \widehat{\bar{v}_i \bar{v}_j} - \widehat{\bar{v}_i} \widehat{\bar{v}_j}, \quad (7.97)$$

where \hat{L}_{ij} denotes the Leonard stresses associated with the test filter. It represents the contribution to the Reynolds stresses by the scales whose length is intermediate between the filter width Δ and the test filter width $\hat{\Delta}$.

If we express the subtest-scale and SGS stresses in Eq. (7.97) using the eddy-viscosity approach Eq. (7.91) together with Eq. (7.95), we obtain

$$\hat{L}_{ij} - \frac{\delta_{ij}}{3} L_{kk} = -2C_d M_{ij} \quad (7.98)$$

with

$$M_{ij} = \hat{\Delta}^2 |\hat{S}| \hat{S}_{ij} - [\Delta^2 |\bar{S}| \bar{S}_{ij}]^\wedge. \quad (7.99)$$

The notation $[\cdot]^\wedge$ means that the whole term enclosed in the square brackets is test-filtered. The parameter C_d can be derived from Eq. (7.98) by using the least-squares minimization of Lilly [183]. This leads to

$$C_d(\vec{r}, t) = -\frac{1}{2} \frac{L_{ij} M_{ij}}{M_{mn} M_{mn}}. \quad (7.100)$$

The above formulation (7.100) is mathematically inconsistent since the parameter C_d was taken outside the test filter in Eq. (7.98). In practice, the numerator and denominator in Eq. (7.100) are therefore ensemble-averaged in the homogeneous directions, that is,

$$C_d(\vec{r}, t) = -\frac{1}{2} \frac{\langle L_{ij} M_{ij} \rangle}{\langle M_{mn} M_{mn} \rangle}. \quad (7.101)$$

Improved dynamic SGS models were proposed, for example, by Ghosal et al. [184], Carati et al. [185], Piomelli and Liu [186], and Held [113].

7.3.4 Wall models

The computational costs of LES of wall-bounded flows at high Reynolds numbers ($Re > 10^6$) are still too high for engineering purposes. The reason is the excessively

large number of grid points (cells) required to resolve the wall layer appropriately. In order to reduce the costs, it is possible to model the wall layer by specifying a correlation between the velocity in the outer flow and the stress at the wall. This approach is quite similar to using wall functions in RANS simulations. The basic assumption is that there is only a weak interaction between the near-wall and the outer region, which is supported by the investigations in Refs. [187, 188].

Earlier implementations of the wall models were based on the assumption that the dynamics of the wall layer are universal and hence they can be approximated by a generalized law-of-the-wall. Basically, the models utilized the logarithmic law (see Refs. [100, 189–191]). Balaras et al. [192] proposed recently a new zonal approach. Within the two-layer model, the filtered Navier-Stokes equations (7.81) are solved up to the first grid point above the wall. From this point to the wall 2-D boundary layer equation are solved on a refined embedded grid. The solution on the embedded grid is then used to prescribe the wall shear stress as a boundary condition for the LES. The zonal approach of Balaras et al. [192] allows it to place the first point in a region $20 < \gamma^+ < 100$, which leads to significantly reduced grid size and hence computational time. The methodology was applied with success to turbulent flows in a plane channel, square duct and rotating channel. Later on, it was also employed for the LES of separated flows with encouraging results [193–196]. An overview and assessment of various wall models can be found in Ref. [197].

7.3.5 Detached eddy simulation

Even though the above wall models help to reduce the number of grid points (cells) considerably, LES still remains too costly for complex engineering configurations. For this reason, Spalart recently suggested another approach, the so-called *detached eddy simulation* (DES), which is aimed at the simulation of high Reynolds-number massively separated flows [198, 199]. The methodology represents a hybrid between the RANS and LES. The idea is to employ highly stretched grids together with a RANS turbulence model (mostly the Spalart-Allmaras model from Section 7.2.1 or Menter's SST model from Section 7.2.3) to resolve the attached boundary layer(s), and to use LES outside the wall region together with an isotropic grid to capture the detached 3-D eddies. Thus, DES tries to combine the strengths of both methods in a single framework.

The implementation of DES for the Spalart-Allmaras turbulence model consists of replacing the wall distance d in the formulas (7.36) and (7.38)–(7.40) by the DES length scale

$$l = \min(d, C_{\text{DES}} \Delta). \quad (7.102)$$

The length scale is dependent on the largest dimension of the control volume, that is, $\Delta = \max(\Delta x, \Delta y, \Delta z)$. The constant C_{DES} depends to some extent on the type of the flow. For a homogeneous turbulence, the value $C_{\text{DES}} = 0.65$ was found optimal [200].

On the other hand, $C_{DES} = 0.1$ was recommended for transonic and supersonic jets [201]. The definition of the length scale l in Eq. (7.102) makes sure that within the boundary layer, where $d < C_{DES} \Delta$ and hence $l = d$, the original RANS model is recovered. On the other hand, outside the boundary layer $l = C_{DES} \Delta$ and the Spalart–Allmaras model serves as a one-equation SGS model for the LES (cf. Eqs. (7.91) and (7.24)). When integrating the governing equations in time, the global time step has to be adjusted such as to resolve the time scales of the detached eddies. This usually means that the time step would by far exceed the stability margin of an explicit scheme for the boundary layer region. It is therefore more efficient to employ a time-accurate implicit scheme, such as the dual time-stepping approach described in Section 6.3. More details regarding the DES methodology and examples of simulations can be found in the above references or in Refs. [202–209].

REFERENCES

- [1] Liu C, Liu Z. Multigrid methods and high order finite difference for flow in transition. AIAA Paper 93-3354; 1993.
- [2] Olejniczak DJ, Weirs VG, Liu J, Candler GV. Hybrid finite-difference methods for DNS of compressible turbulent boundary layers. AIAA Paper 96-2086; 1996.
- [3] Cook AW, Riley JJ. Direct numerical simulation of a turbulent reactive plume on a parallel computer. *J Comput Phys* 1996;129:263-83.
- [4] Pinelli A, Vacca A, Quarteroni A. A spectral multidomain method for the numerical simulation of turbulent flows. *J Comput Phys* 1997;136:546-58.
- [5] Zhong X. High-order finite-difference schemes for numerical simulation of hypersonic boundary-layer transition. *J Comput Phys* 1998;144:662-709.
- [6] Lange M, Riedel U, Warnatz J. Parallel DNS of turbulent flames with detailed reaction schemes. AIAA Paper 98-2979; 1998.
- [7] Hernandez G, Brenner G. Boundary conditions for direct numerical simulations of free jets. AIAA Paper 99-0287; 1999.
- [8] Ladeinde F, Liu W, O'Brien EE. DNS evaluation of chemistry models for turbulent, reacting, and compressible mixing layers. AIAA Paper 99-0413; 1999.
- [9] Freund JB. Acoustic sources in a turbulent jet—a direct numerical simulation study. AIAA Paper 99-1858; 1999.
- [10] Rizzetta DP, Visbal MR. Application of a high-order compact difference scheme to large-eddy and direct numerical simulation. AIAA Paper 99-3714; 1999.
- [11] Luchini P, Quadrio M. A low-cost parallel implementation of direct numerical simulation of wall turbulence. *J Comput Phys* 2006;211:551-71.
- [12] Martín MP, Candler GV. A parallel implicit method for the direct numerical simulation of wall-bounded compressible turbulence. *J Comput Phys* 2006;215:153-71.
- [13] Martín MP, Taylor EM, Wu M, Weirs VG. A bandwidth-optimized WENO scheme for the effective direct numerical simulation of compressible turbulence. *J Comput Phys* 2006;220:270-89.
- [14] Salvadore F, Bernardini M, Botti M. GPU accelerated flow solver for direct numerical simulation of turbulent flows. *J Comput Phys* 2013;235:129-42.
- [15] Khajeh-Saeed A, Perot JB. Direct numerical simulation of turbulence using GPU accelerated supercomputers. *J Comput Phys* 2013;235:241-57.
- [16] Marxen Ö, Magin TE, Shaqfeh E, Iaccarino G. A method for the direct numerical simulation of hypersonic boundary-layer instability with finite-rate chemistry. *J Comput Phys* 2013;255:572-89.
- [17] Boussinesq J. Essai sur la théorie des eaux courantes. *Mem Pres Acad Sci* 1877;XXIII:46.

- [18] Boussinesq J. Theorie de l' écoulement tourbillonant et tumultueur des liquides dans les lits rectilignes. *Comptes Rendus Acad Sci* 1896;CXXII:1293.
- [19] Reynolds O. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philos Trans R Soc Lond A* 1895;186:123-64.
- [20] Favre A. Equations des gaz turbulents compressibles, part 1: formes générales. *J Mécan* 1965;4:361-90.
- [21] Favre A. Equations des gaz turbulents compressibles, part 2: méthode des vitesses moyennes; méthode des vitesses moyennes pondérées par la masse volumique. *J Mécan* 1965;4:391-21.
- [22] Morkovin MV. Effects of compressibility on turbulent flow. In: Favre A, editor. *The mechanics of turbulence*. New York: Gordon and Breach; 1964.
- [23] Wilcox DC. *Turbulence modeling for CFD*. La Canada, CA: DCW Industries, Inc.; 1993.
- [24] Reynolds O. On the extent and action of the heating surface for steam boilers. *Proc Manchester Lit Philos Soc* 1874;14:7-12.
- [25] Hallbäck M, Henningson DS, Johansson AV, Alfredsson PH, editors. *Turbulence and transition modelling*. ERCOFTAC Series, vol. 2. Dordrecht, The Netherlands: Kluwer Academic Publishers; 1996.
- [26] Spalart P, Shur M. On the sensitization of turbulence models to rotation and curvature. *Aerospace Sci Technol* 1997;5:297-302.
- [27] Shur M, Strelets M, Travlin A, Spalart P. Turbulence modeling in rotating and curved channels: assessment of the Spalart-Shur correction term. *AIAA Paper 98-0325*; 1998.
- [28] Lumley JL. Toward a turbulent constitutive equation. *J Fluid Mech* 1970;41:413-34.
- [29] Lumley JL. Computational modeling of turbulent flows. *Adv Appl Mech* 1978;18:123-76.
- [30] Craft TJ, Launder BE, Suga K. A non-linear eddy-viscosity model including sensitivity to stress anisotropy. In: Proc. 10th symp. turbulent shear flows, Paper 23:19. Pennsylvania State University; 1995.
- [31] Shih TH, Zhu J, Liou WW, Chen K-H, Liu N-S, Lumley J. Modeling of turbulent swirling flows. In: Proc. 11th symposium on turbulent shear flows. Grenoble, France; 1997 [also NASA TM-113112; 1997].
- [32] Chen K-H, Liu N-S. Evaluation of a non-linear turbulence model using mixed volume unstructured grids. *AIAA Paper 98-0233*, 1998.
- [33] Goldberg U, Peroomian O, Chakravarthy S. Application of the $k - \varepsilon - R$ turbulence model to wall-bounded compressive flows. *AIAA Paper 98-0323*; 1998.
- [34] Abdel Gawad AF, Abdel Latif OE, Ragab SA, Shabaka IM. Turbulent flow and heat transfer in rotating non-circular ducts with nonlinear $k - \varepsilon$ model. *AIAA Paper 98-0326*; 1998.
- [35] Yoshizawa A, Fujiwara H, Hamba F, Nisizima S, Kumagai Y. Nonequilibrium turbulent-viscosity model for supersonic free-shear layer/wall-bounded flows. *AIAA J* 2003;41:1029-36.
- [36] Tucker PG, Liu Y, Chung YM, Jouvray A. Computation of an unsteady complex geometry flow using novel non-linear turbulence models. *Int J Numer Methods Fluids* 2003;43:979-1001.
- [37] Yang X, Ma H. Computation of strongly swirling confined flows with cubic eddy-viscosity turbulence models. *Int J Numer Methods Fluids* 2003;43:1355-70.
- [38] Hinze JO. *Turbulence*. 2nd ed. New York: McGraw-Hill; 1975.
- [39] Bai XS. On the modeling of turbulent combustion at low Mach numbers. PhD Thesis. Stockholm, Sweden: Royal Inst. of Technology, Dept. of Mechanics/Applied CFD; 1994.
- [40] Adumitroaie V, Ristorcelli JR, Taulbee DB. Progress in Favre-Reynolds stress closures for compressible flows. ICASE Report No. 98-21; 1998.
- [41] Kobayashi MH, Marques NPC, Pereira JCF. Critical evaluation of near wall treatment for turbulent Reynolds stress models to predict aerodynamic stall. *AIAA Paper 97-1826*; 1997.
- [42] Rizzetta DP. Evaluation of algebraic Reynolds-stress models for separated high-speed flows. *AIAA Paper 97-2125*; 1997.
- [43] Chassaing JC, Gerolymos GA, Vallet I. Reynolds-stress model dual-time-stepping computation of unsteady three-dimensional flows. *AIAA J* 2003;41:1882-94.
- [44] Ben Nasr N, Gerolymos GA, Vallet I. Low-diffusion approximate Riemann solvers for Reynolds-stress transport. *J Comput Phys* 2014;268:186-35.

- [45] Mor-Yossef Y. Unconditionally stable time marching scheme for Reynolds stress models. *J Comput Phys* 2014;276:635–64.
- [46] Bardina JE, Huang PG, Coakley TJ. Turbulence modeling validation, testing, and development. NASA TM-110446; 1997.
- [47] Spalart SR, Allmaras SA. A one-equation turbulence model for aerodynamic flows. *AIAA Paper* 92-0439; 1992 [also in *La Recherche Aerospatiale* 1994;1:5-21].
- [48] Tucker PG. Differential equation based length scales to improve DES and RANS simulations. *AIAA Paper* 2003-3968; 2003.
- [49] Ashford GA. An unstructured grid generation and adaptive solution technique for high-Reynolds-number compressible flows. PhD Thesis. Ann Arbor, USA: The University of Michigan, Dept. of Aerospace Engineering; 1996.
- [50] Rung T, Bunge U, Schatz M, Thiele F. Restatement of the Spalart-Allmaras eddy-viscosity model in strain-adaptive formulation. *AIAA J* 2003;41:1396–9.
- [51] Chou PY. On velocity correlations and the solutions of the equations of turbulent fluctuations. *Quart Appl Math* 1945;3:38–54.
- [52] Jones WP, Launder BE. The prediction of laminarization with a two-equation model of turbulence. *Int J Heat Mass Transfer* 1972;15:301–14.
- [53] Jones WP, Launder BE. The prediction of low-Reynolds-number phenomena with a two-equation model of turbulence. *Int J Heat Mass Transfer* 1973;16:1119–30.
- [54] Launder BE, Sharma BI. Application of the energy dissipation model of turbulence to the calculation of flow near a spinning disc. *Lett Heat Mass Transfer* 1974;1:131–8.
- [55] Launder BE, Spalding B. The numerical computation of turbulent flows. *Comput Methods Appl Mech Eng* 1974;3:269–89.
- [56] Lam CKG, Bremhorst KA. Modified form of $K - \varepsilon$ model for predicting wall turbulence. *ASME J Fluid Eng* 1981;103:456–60.
- [57] Chien K-Y. Predictions of channel and boundary-layer flows with a low-Reynolds-number turbulence model. *AIAA J* 1982;20:33–38.
- [58] Patel VC, Rodi W, Scheuerer G. turbulence models for near-wall and low Reynolds number flows: a review. *AIAA J* 1985;23:1308–19.
- [59] Kunz RF, Lakshminarayana B. stability of explicit Navier-Stokes procedures using $K - \varepsilon$ and $K - \varepsilon$ /algebraic Reynolds stress turbulence models. *J Comput Phys* 1992;103:141–59.
- [60] Gerolymos GA. Implicit multiple-grid solution of the compressible Navier-Stokes equations using $K - \varepsilon$ turbulence closure. *AIAA J* 1990;28:1707–17.
- [61] Mavriplis DJ, Martinelli L. Multigrid solution of compressible turbulent flow on unstructured meshes using a two-equation model. *AIAA Paper* 91-0237; 1991.
- [62] Turner MG, Jennions IK. An investigation of turbulence modeling in transonic fans including a novel implementation of an implicit $K - \varepsilon$ turbulence model. *J Turbomach* 1993;115:249–60.
- [63] Grasso F, Falconi D. On high speed turbulence modeling of shock-wave boundary-layer interaction. *AIAA Paper* 93-0778; 1993.
- [64] Koobus B. An implicit method for turbulent boundary layers simulation. INRIA Report No. 2450, December 1994.
- [65] Sondak DL. Parallel implementation of the $K - \varepsilon$ turbulence model. *AIAA Paper* 94-0758; 1994.
- [66] Gerolymos GA, Vallet I. Implicit computation of three-dimensional compressible Navier-Stokes equations using $K - \varepsilon$ closure. *AIAA J* 1996;34:1321–30.
- [67] Luo H, Baum JD, Löhner R. Computation of compressible flows using a two-equation turbulence model on unstructured grids. *AIAA Paper* 97-0430; 1997.
- [68] Papp J, Ghia K. Study of turbulent compressible mixing layers using two-equation turbulence models including an RNG $K - \varepsilon$ model. *AIAA paper* 1998-320; 1998.
- [69] Wang Q, Massey SJ, Abdol-Hamid KS, Frink NT. Solving Navier-Stokes equations with advanced turbulence models on three-dimensional unstructured grids. *AIAA Paper* 99-0156; 1999.
- [70] Sinha K, Candler G, Martin M. Assessment of the $K - \varepsilon$ turbulence model for compressible flows using direct simulation data. *AIAA Paper* 2001-730; 2001.

- [71] Van Maele K, Merci B, Dick E. Comparative study of $K - \varepsilon$ turbulence models in inert and reacting swirling flow. AIAA Paper 2003-3744; 2003.
- [72] Ferreira VG, Mangiavacchi N, Tomé MF, Castelo A, Cuminato JA, McKee S. Numerical simulation of turbulent free surface flow with two-equation $K - \varepsilon$ eddy-viscosity models. Int J Numer Methods Fluids 2004;44:347-75.
- [73] Brinckman K, Rodebaugh G, Dash S. Towards a unified K-epsilon turbulence model for the practical analysis of aeropropulsive flows. AIAA Paper 2013-392; 2013.
- [74] Sinha K, Balasridhar SJ. Conservative formulation of the $K - \varepsilon$ turbulence model for shock-turbulence interaction. AIAA J 2013;51:1872-82.
- [75] Edeling WN, Cinnella P, Dwight RP, Bijl H. Bayesian estimates of parameter variability in the $K - \varepsilon$ turbulence model. J Comput Phys 2014;258:73-94.
- [76] Lakshminarayana B. Turbulence modeling for complex shear flows. AIAA J 1986;24:1900-17.
- [77] Kunz RF, Lakshminarayana B. Explicit Navier-Stokes computation of cascade flows using the $K - \varepsilon$ turbulence model. AIAA J 1992;30:13-22.
- [78] Spalding DB. A single formula for the “law of the wall”. ASME J Appl Mech 1961;28:455-58.
- [79] Viegas JR, Rubesin MW. Wall-function boundary conditions in the solution of the Navier-Stokes equations for complex compressible flows. AIAA Paper 83-1694; 1983.
- [80] Abdol-Hamid KS, Lashmanan B, Carlson JR. Application of Navier-Stokes code PAB3D with $k-\varepsilon$ turbulence models to attached and separated flows. NASA TR-3480; 1995.
- [81] Frink N. Assessment of an unstructured-grid method for predicting 3-D turbulent viscous flows. AIAA Paper 96-0292; 1996.
- [82] Mohammadi B, Pironneau O. Unsteady separated turbulent flows computation with wall-laws and $k - \varepsilon$ model. Comput Methods Appl Eng 1997;148:393-405.
- [83] Grotjans H, Menter FR. Wall functions for general application CFD codes. In: Proc. fourth european CFD conf, 7-11 Sept. Athens, Greece; 1998.
- [84] Kalitzin G, Medic G, Iaccarino G, Durbin P. Near-wall behavior of RANS turbulence models and implications for wall functions. J Comput Phys 2005;204:265-91.
- [85] Knopp T, Alrutz T, Schwamborn D. A grid and flow adaptive wall-function method for RANS turbulence modeling. J Comput Phys 2006;220:19-40.
- [86] Menter FR. Two-equation eddy-viscosity turbulence models for engineering applications. AIAA Paper 93-2906; 1993 [also AIAA J 1994;32:1598-605].
- [87] Menter FR, Rumsey LC. Assessment of two-equation turbulence models for transonic flows. AIAA Paper 94-2343; 1994.
- [88] Wilcox DC. Reassessment of the scale-determining equation for advanced turbulence models. AIAA J 1988;26:1299-1310.
- [89] Menter FR. Influence of freestream values on $K - \omega$ turbulence model predictions. AIAA J 1992;30:1651-9.
- [90] Hellsten A, Laine S. Extension of the $K - \omega$ -SST turbulence model for flows over rough surfaces. AIAA Paper 97-3577; 1997 [also AIAA J 1998;36:1728-29].
- [91] Hellsten A. Some improvements in Menter's $K - \omega$ SST turbulence model. AIAA Paper 98-2554; 1998.
- [92] Forsythe JR, Hoffmann KA, Suzen YB. Investigation of modified Menter's two-equation turbulence model for supersonic applications. AIAA Paper 99-0873; 1999.
- [93] Goodheart K, Dykas S, Schnerr G. Numerical insights into the solution of transonic flow test cases using the Wilcox $K - \omega$, EASM and SST models. AIAA Paper 2003-1141; 2003.
- [94] Steed R. High lift CFD simulations with an SST-based predictive laminar to turbulent transition model. AIAA paper 2011-864; 2011.
- [95] Murman S. Evaluating modified diffusion coefficients for the SST turbulence model using benchmark tests. AIAA Paper 2011-3571; 2011.
- [96] Geisbauer S. Numerical spoiler wake investigations at the borders of the flight envelope. AIAA Paper 2011-3811; 2011.
- [97] Schoenawa S, Hartmann R. Discontinuous Galerkin discretization of the Reynolds-averaged Navier-Stokes equations with the shear-stress transport model. J Comput Phys 2014;262:194-216.

- [98] Smagorinsky J. General circulation experiments with the primitive equations. *Mon Weather Rev* 1963;91:99–165.
- [99] Deardorff JW. A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *J Fluid Mech* 1970;41:453–80.
- [100] Schumann U. Subgrid-scale model for finite-difference simulations of turbulent flows in plane channels and annuli. *J Comput Phys* 1975;18:376–404.
- [101] Bardina J, Ferziger JH, Reynolds WC. Improved subgrid models for large eddy simulation. *AIAA Paper* 80-1357; 1980.
- [102] Moin P, Kim J. Numerical investigation of turbulent channel flow. *J Fluid Mech* 1982;118:341–77.
- [103] Piomelli U, Zang TA, Speziale CG, Hussaini MY. On the large-eddy simulation of transitional wall-bounded flows. ICASE Report No. 89-55; 1989.
- [104] Hussaini MY, Speziale CG, Zang TA. Discussion of “The potential and limitations of direct and large-eddy simulations.” ICASE Report No. 89-61; 1989.
- [105] Erlebacher G, Hussaini MY, Speziale CG, Zang TA. Toward the large-eddy simulation of compressible turbulent flows. ICASE Report No. 90-76; 1990.
- [106] Erlebacher G, Hussaini MY, Speziale CG, Zang TA. On the large-eddy simulation of compressible isotropic turbulence. In: 12th International conf. on numerical methods in fluid dynamics, Oxford, England; Lecture Notes in Physics, vol. 371. Berlin: Springer; 1990.
- [107] Schumann U. Direct and large-eddy simulation of turbulence—summary of the state of the art 1993. VKI Lecture Series 1993-02. Von Karman Institute: Rhode-St-Genèse, Belgium; 1993.
- [108] Jones WP, Wille M. Large eddy simulation of a jet in a cross-flow. In: 10th Symposium on turbulent shear flows, vol. 1. The Pennsylvania State University, PA, August 14–16; 1995.
- [109] Lund TS, Moin P. Large eddy simulation of a boundary layer on a concave surface. In: 10th symposium on turbulent shear flows, vol. 1. The Pennsylvania State University, PA, August 14–16; 1995.
- [110] Calhoon WH, Menon S. Subgrid modeling for reacting large eddy simulations. *AIAA Paper* 96-0516; 1996.
- [111] Cook AW, Riley JJ, Kosaly G. A laminar flamelet approach to subgrid-scale chemistry in turbulent flows. *Combust Flame* 1997;109:332–41.
- [112] Wu Y, Cao S, Yang J, Ge L, Guillaud M, Soula V. Turbulent flow calculation through a water turbine draft tube by using the Smagorinsky model. In: ASME fluids engineering division summer meeting (FEDSM), Paper No. FEDSM98-4864. Washington DC; 1998.
- [113] Held J. Large eddy simulations of separated compressible flows around 3-D configurations. PhD Thesis. Lund, Sweden: Department of heat and power engineering/fluid mechanics, Lund Institute of Technology; 1998.
- [114] Kim W-W, Menon S, Mongia HC. Large eddy simulations of reacting flows in a dump combustor. *AIAA Paper* 98-2432; 1998.
- [115] Ducros F, Ferrand V, Nicoud F, Weber C, Darraq D, Gachereau C, Poinsot T. Large-eddy simulation of the shock/turbulence interaction. *J Comput Phys* 1999;152:517–49.
- [116] Jaberi FA. Large eddy simulation of turbulent premixed flames via filtered mass density function. *AIAA Paper* 99-0199; 1999.
- [117] Han J, Lin Y-L, Arya SP, Proctor FH. Large eddy simulation of aircraft wake vortices in a homogeneous atmospheric turbulence—vortex decay and descent. *AIAA Paper* 99-0756; 1999.
- [118] Bui TT. A parallel, finite-volume algorithm for large-eddy simulation of turbulent flows. *AIAA Paper* 99-0789; 1999.
- [119] Kravchenko AG, Moin P, Shariff K. B-Spline method and zonal grids for simulations of complex turbulent flows. *J Comput Phys* 1999;151:757–89.
- [120] Cook AW. A consistent approach to large eddy simulation using adaptive mesh refinement. *J Comput Phys* 1999;154:117–33.
- [121] Pasarelli A, Piomelli U, Candler GV. Multi-block large-eddy simulations of turbulent boundary layers. *J Comput Phys* 2000;157:256–79.
- [122] Kurose R, Makino H. Large eddy simulation of a solid-fuel jet flame. *Combust Flame* 2003;135:1–16.
- [123] Di Mare F, Jones WP, Menzies KR. Large eddy simulation of a model gas turbine combustor. *Combust Flame* 2004;137:278–94.

- [124] Roux S, Lartigue G, Poinsot T, Meier U, Bérat C. Studies of mean and unsteady flow in a swirled combustor using experiments, acoustic analysis, and large eddy simulations. *Combust Flame* 2005;141:40–54.
- [125] Trouve A, Wang Y. Large eddy simulation of compartment fires. *Int J Comput Fluid Dynam* 2010;24:449–66.
- [126] Knopp T, Zhang X, Kessler R, Lube G. Enhancement of an industrial finite-volume code for large-eddy-type simulation of incompressible high Reynolds number flow using near-wall modelling. *Comput Methods Appl Mech Eng* 2010;199:890–902.
- [127] Granet V, Vermorel O, Lacour C, Enaux B, Dugué V, Poinsot T. Large-eddy simulation and experimental study of cycle-to-cycle variations of stable and unstable operating points in a spark ignition engine. *Combust Flame* 2012;159:1562–75.
- [128] Viazza S, Poncet S, Serre E, Randriamampianina A, Bontoux P. High-order large eddy simulations of confined rotor-stator flows. *Flow Turbul Combust* 2012;88:63–75.
- [129] Chapman DR. Computational aerodynamics development and outlook. *AIAA J* 1979;17: 1293–313.
- [130] Tam CKW, Webb JC. Dispersion-relation-preserving finite difference schemes for computational acoustics. *J Comput Phys* 1993;107:262–81.
- [131] Tam CKW. Applied aero-acoustics: prediction methods. VKI Lecture Series 1996–04. Rhode-St-Genèse, Belgium: Von Karman Institute; 1996.
- [132] Mital R, Moin P. Suitability of upwind-biased finite difference schemes for large-eddy simulation of turbulent flows. *AIAA J* 1997;35:1415–17.
- [133] Garnier E, Mossi M, Sagaut P, Comte P, Deville M. On the use of shock-capturing schemes for large-eddy simulation. *J Comput Phys* 1999;153:273–311.
- [134] Kravchenko AG, Moin P. On the effect of numerical errors in large eddy simulations of turbulent flow. *J Comput Phys* 1997;131:310–22.
- [135] Park N, Yoo JY, Choi H. Discretization errors in large eddy simulation: on the suitability of centered and upwind-biased compact difference schemes. *J Comput Phys* 2004;198:580–616.
- [136] Denaro FM. What does finite volume-based implicit filtering really resolve in Large-eddy simulations? *J Comput Phys* 2011;230:3849–83.
- [137] Miet P, Laurence D, Nitrosso B. Large eddy simulation with unstructured grids and finite elements. In: 10th Symposium on turbulent shear flows, vol. 2. The Pennsylvania State University, PA, August 14–16, 1995.
- [138] Jansen K. Large-eddy simulation of flow around a NACA 4412 Airfoil using unstructured grids. Center for Turbulence Research, NASA Ames/Stanford University, Annual Research Briefs; 1996. p. 225–32.
- [139] Chalot F, Marquez B, Ravachol M, Ducros F, Nicoud F, Poinsot T. A consistent finite element approach to large eddy simulation. *AIAA Paper 98-2652*; 1998.
- [140] Okong'o N, Knight D. Compressible large eddy simulation using unstructured grids: channel and boundary layer flows. *AIAA Paper 98-3315*; 1998.
- [141] Urbin G, Knight D, Zheltovodov AA. Compressible large eddy simulation using unstructured grid: supersonic turbulent boundary layer and compression corner. *AIAA Paper 99-0427*; 1999.
- [142] Marsden AL, Vasilyev OV, Moin P. Construction of commutative filters for LES on unstructured meshes. *J Comput Phys* 2002;175:584–603.
- [143] Camarri S, Salvetti MV, Koobus B, Dervieux A. A low-diffusion MUSCL scheme for LES on unstructured grids. *Comput Fluids* 2004;33:1101–29.
- [144] Selle L, Lartigue G, Poinsot T, Koch R, Schildmacher K-U, Krebs W, et al. Compressible large eddy simulation of turbulent combustion in complex geometry on unstructured meshes. *Combust Flame* 2004;137:489–505.
- [145] Koobus B, Farhat C. A variational multiscale method for the large eddy simulation of compressible turbulent flows on unstructured meshes—application to vortex shedding. *Comput Methods Appl Mech Eng* 2004;193:1367–83.
- [146] Ciardi M, Sagaut P, Klein M, Dawes WN. A dynamic finite volume scheme for large-eddy simulation on unstructured grids. *J Comput Phys* 2005;210:632–55.

- [147] Wang G, Duchaine F, Papadogiannis D, Duran I, Moreau S, Gicquel LYM. An overset grid method for large eddy simulation of turbomachinery stages. *J Comput Phys* 2014;274:333-55.
- [148] Ferziger JH. Direct and large eddy simulation of turbulence. In: Vincent A, editor. Numerical methods in fluid mechanics, vol. 16. Centre de Recherches Mathématiques Université de Montréal, Proceedings and Lecture Notes; 1998. p. 53-97.
- [149] Piomelli U. Large-eddy simulation of turbulent flows. VKI Lecture Series 1998-05. Rhode-St-Genèse, Belgium: Von Karman Institute; 1998.
- [150] Mossi M. Simulation of benchmark and industrial unsteady compressible turbulent fluid flows. PhD Thesis, No. 1958, École Polytechnique Fédérale de Lausanne, Département de Génie Mécanique, Switzerland; 1999.
- [151] Sagaut P. Large eddy simulation for incompressible flows: an introduction. 3rd ed. Berlin: Springer; 2005.
- [152] Garnier E, Adams N, Sagaut P. Large eddy simulation for compressible flows. Berlin: Springer; 2009.
- [153] Piomelli U. Large-eddy simulation: present state and future directions. AIAA Paper 98-0534; 1998.
- [154] Pitsch H. Large-eddy simulation of turbulent combustion. *Annu Rev Fluid Mech* 2006;38:453-482.
- [155] Piomelli U, Benocci C, van Beeck JP. Large eddy simulation and related techniques. VKI Lecture Series 2010-04. Rhode-St-Genèse, Belgium: Von Karman Institute; 2010.
- [156] Vasilyev OV, Lund TS. A General theory of discrete filtering for LES in complex geometry. Annual Research Briefs, Center for Turbulence Research, NASA Ames/Stanford Univ.; 1997. p. 67-82.
- [157] Vasilyev OV, Lund TS, Moin P. A general class of commutative filters for les in complex geometries. *J Comput Phys* 1998;146:82-104.
- [158] Lund TS, Kaltenbach H-J. Experiments with explicit filtering for les using a finite-difference method. Annual Research Briefs, Center for Turbulence Research, NASA Ames/Stanford Univ.; 1995. p. 91-105.
- [159] Lund TS. On the use of discrete filters for large eddy simulation. Annual Research Briefs, Center for Turbulence Research, NASA Ames/Stanford Univ.; 1997. p. 83-95.
- [160] Leonard A. Energy cascade in large-eddy simulations of turbulent fluid flows. *Adv Geophys* 1974;18:237-48.
- [161] Moin P, Squires KD, Cabot WH, Lee S. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Phys Fluids* 1991;3:2746-57.
- [162] Vreman B, Geurts B, Kuerten H, Broeze J, Wasistho B, Streng M. Dynamic subgrid-scale models for LES of transitional and turbulent compressible flow id 3-D shear layers. In: 10th symposium on turbulent shear flows, vol. 1. The Pennsylvania State University, PA, August 14-16; 1995.
- [163] Domaradzki JA, Saiki EM. Backscatter models for large-eddy simulations. *Theor Comput Fluid Dyn* 1997;9:75-83.
- [164] Lenormand E, Sagaut P, Phuoc LT, Comte P. Subgrid-scale models for large-eddy simulations of compressible wall bounded flows. *AIAA J* 2000;38:1340-50.
- [165] Boris JP, Grinstein FF, Oran ES, Kolbe RJ. New insights into large eddy simulations. *Fluid Dyn Res* 1992;10:199-228.
- [166] Fureby C, Grinstein FF. Monotonically integrated large eddy simulation of free shear flows. *AIAA J* 1999;37:544-56.
- [167] Grinstein FF, Fureby C. Recent progress on MILES for high Reynolds number flows. *J Fluids Eng* 2002;124:848-61.
- [168] Garnier E, Mossi M, Sagaut P, Comte P, Deville M. On the use of shock-capturing schemes for large-eddy simulation. *J Comput Phys* 1999;153:273-311.
- [169] Domaradzki JA, Radhakrishnan S, Xiao Z, Smolarkiewicz PK. Effective eddy viscosities in implicit modeling of high Reynolds number flows. AIAA Paper 2003-4103; 2003.
- [170] Fureby C, Grinstein FF. Monotonically integrated large eddy simulation of free shear flows. *AIAA J* 1999;37:544-56.
- [171] Fureby C. Large eddy simulation of rearward-facing step flow. *AIAA J* 1999;37:1401-10.
- [172] Okong'o N, Knight D, Zhou G. Large eddy simulations using an unstructured grid compressible Navier-Stokes algorithm. *Int J Comput Fluid Dynam* 2000;13:303-26.

- [173] Yan H, Knight D, Zheltovodov AA. Large-eddy simulation of supersonic flat-plate boundary layers using the monotonically integrated large-eddy simulation (MILES) technique. *J Fluids Eng* 2002;124:868-75.
- [174] Fureby C, Grinstein FF. Large eddy simulation of high-Reynolds-number free and wall-bounded flows. *J Comput Phys* 2002;181:68-97.
- [175] Raverdy B, Mary I, Sagaut P. High-resolution large-eddy simulation of flow around low-pressure turbine blade. *AIAA J* 2003;41:390-7.
- [176] Grinstein FF, Fureby C. Implicit large eddy simulation of high-re flows with flux-limiting schemes. *AIAA Paper* 2003-4100; 2003.
- [177] Hicken S, Adams NA, Domaradzki JA. An adaptive local deconvolution method for implicit LES. *J Comput Phys* 2006;213:413-36.
- [178] Rogallo RS, Moin P. Numerical simulation of turbulent flows. *Annu Rev Fluid Mech* 1984;16:99-137.
- [179] Vreman AW. Direct and large-eddy simulation of the compressible turbulent mixing layer. PhD Thesis. The Netherlands: Dept. of Applied Mathematics, University of Twente Enschede; 1995.
- [180] Lilly DK. The representation of small-scale turbulence in numerical simulation experiments. In: Proc. IBM sci. comp. symp. on environmental sciences, New York; November 14-16, 1967.
- [181] Germano M, Piomelli U, Moin P, Cabot WH. A dynamic subgrid-scale eddy viscosity model. *Phys Fluids* 1991;3:1760-5.
- [182] Germano M. Turbulence: the filtering approach. *J Fluid Mech* 1992;238:325-36.
- [183] Lilly DK. A proposed modification of the Germano subgrid-scale closure method. *Phys Fluids* 1992;4:633-5.
- [184] Ghosal S, Lund TS, Moin P, Akselvoll K. A dynamic localization model for large-eddy simulation of turbulent flows. *J Fluid Mech* 1995;286:229-55.
- [185] Carati D, Ghosal S, Moin P. On the representation of backscatter in dynamic localization models. *Phys Fluids* 1995;7:606.
- [186] Piomelli U, Liu J. Large-eddy simulation of rotating channel flows using a localized dynamic model. *Phys Fluids* 1995;7:839-48.
- [187] Chapman DR, Kuhn GD. The limiting behavior of turbulence near a wall. *J Fluid Mech* 1986;70:265-92.
- [188] Brooke JW, Hanratty TJ. Origin of turbulence-producing eddies in a channel flow. *Phys Fluids* 1993;5:1011-22.
- [189] Grötzsch G. Direct numerical and large eddy simulation of turbulent channel flows. In: Cheremisinoff NP, editor. Encyclopedia of fluid mechanics. West Orange: Gulf Publishing; 1987, p. 1337.
- [190] Mason PJ, Callen NS. On the magnitude of the subgrid-scale eddy coefficient in large-eddy simulations of turbulent channel flow. *J Fluid Mech* 1986;162:439-62.
- [191] Piomelli U, Ferziger J, Moin P, Kim J. New approximate boundary conditions for large eddy simulations of wall-bounded flows. *Phys Fluids* 1989;1:1061-68.
- [192] Balaras E, Benocci C, Piomelli U. Two-layer approximate boundary condition for large-eddy simulations. *AIAA J* 1996;34:1111-19.
- [193] Cabot W. Large-eddy simulations with wall models. Annual Research Briefs, Center for Turbulence Research, NASA Ames/Stanford Univ.; 1995. p. 41-50.
- [194] Cabot W. Near-wall models in large eddy simulations of flow behind a backward-facing step. Annual Research Briefs, Center for Turbulence Research, NASA Ames/Stanford Univ.; 1996. p. 199-210.
- [195] Cabot W, Moin P. Approximate wall boundary conditions in the large-eddy simulation of high Reynolds number flow. *Flow Turbulence Combust* 1999;63:269-91.
- [196] Wang M, Moin P. Wall modelling in LES of trailing-edge flow. In: Proc. 2nd int. symposium on turbulence and shear flow phenomena, vol. 2. Stockholm, Sweden; 2001. p. 165-70.
- [197] Piomelli U, Balaras E. Wall-layer models for large-eddy simulations. *Prog Aerosp Sci* 2008;44:437-46.
- [198] Spalart PR. Strategies for turbulence modelling and simulations. *Int J Heat Fluid Flow* 2000;21:252-63.
- [199] Spalart PR. Trends in turbulence treatments. *AIAA Paper* 2000-2306; 2000.

- [200] Strelets M. Detached eddy simulation of massively separated flows. AIAA Paper 2001-0879; 2001.
- [201] Mani M. Hybrid turbulence models for unsteady flow simulation. *J Aircraft* 2004;41:110-18.
- [202] Forsythe J, Squires K, Wurtzler K, Spalart PR. Detached eddy simulation of fighter aircraft at high alpha. AIAA Paper 2002-0591; 2002.
- [203] Forsythe J, Squires K, Wurtzler K, Spalart PR. Prescribed spin of the F-15E using detached-eddy simulation. AIAA Paper 2003-839; 2003.
- [204] Schlüter JU. Consistent boundary conditions for integrated RANS/LES simulations: LES inflow conditions. AIAA Paper 2003-3971; 2003.
- [205] Hamba F. A hybrid RANS/LES simulation of turbulent channel flow. *Theor Comput Fluid Dyn* 2003;16:387-403.
- [206] Georgiadis NJ, Alexander JID, Reshotko E. Hybrid Reynolds-averaged Navier-Stokes/large-eddy simulations of supersonic turbulent mixing. *AIAA J* 2003;41:218-29.
- [207] Schlüter JU, Pitsch H. Antialiasing filters for coupled Reynolds-averaged/large-eddy simulations. *AIAA J* 2005;43:608-15.
- [208] Kawai S, Fujii K. Analysis and prediction of thin-airfoil stall phenomena with hybrid turbulence methodology. *AIAA J* 2005;43:953-61.
- [209] Gopalan H, Heinz S, Stöllinger MK. A unified RANS-LES model: computational development, accuracy and cost. *J Comput Phys* 2013;249:249-74.

CHAPTER 8

Boundary Conditions

Contents

8.1	Concept of Dummy Cells	254
8.2	Solid Wall	255
8.2.1	Inviscid flow	255
	<i>Structured cell-centered scheme</i>	256
	<i>Structured cell-vertex scheme</i>	257
	<i>Unstructured cell-centered scheme</i>	258
	<i>Unstructured median-dual scheme</i>	259
8.2.2	Viscous flow	260
	<i>Cell-centered scheme</i>	261
	<i>Cell-vertex scheme</i>	261
8.3	Far-Field	262
8.3.1	Concept of characteristic variables	262
	<i>Supersonic inflow</i>	263
	<i>Supersonic outflow</i>	263
	<i>Subsonic inflow</i>	264
	<i>Subsonic outflow</i>	264
8.3.2	Modifications for lifting bodies	264
	<i>Vortex correction in 2D</i>	265
	<i>Vortex correction in 3D</i>	266
8.4	Inlet/Outlet Boundary	268
	<i>Subsonic inlet</i>	268
	<i>Subsonic outlet</i>	269
	<i>Supersonic inlet and outlet</i>	269
8.5	Injection Boundary	270
8.6	Symmetry Plane	271
	<i>Cell-centered scheme</i>	271
	<i>Cell-vertex scheme (dual control volume)</i>	271
8.7	Coordinate Cut	272
8.8	Periodic Boundaries	273
	<i>Cell-centered scheme</i>	273
	<i>Cell-vertex scheme (dual control volume)</i>	274
	<i>Rotational periodicity</i>	274
8.9	Interface Between Grid Blocks	275
8.10	Flow Gradients at Boundaries of Unstructured Grids	278
	References	279

Any numerical simulation can consider only a part of the real physical domain or of the system. The truncation of the domain leads to artificial boundaries, where we have to prescribe values for certain physical quantities. Furthermore, walls that are exposed to the flow represent natural boundaries of the physical domain. The numerical treatment of the boundary conditions requires a particular care. An improper implementation can result in an inaccurate simulation of the real system. Additionally, the stability and the convergence speed of the solution scheme can be negatively influenced.

The following types of boundary conditions are in general encountered in the numerical solution of the Euler and the Navier-Stokes equations:

- solid wall,
- far-field in external flows,
- inflow and outflow in internal flows,
- injection boundary,
- symmetry,
- coordinate cut and periodic boundary,
- boundary between blocks.

The numerical treatment of these boundary conditions is described in detail in the following sections. For literature on further boundary conditions, like heat radiation on walls or free surfaces, the reader is referred to Section 3.4.

8.1 CONCEPT OF DUMMY CELLS

Before we proceed with the discussion of the boundary conditions, we should mention the concept of *dummy cells* or *dummy points*. This approach is very popular on structured grids. However, dummy cells offer some advantages on unstructured grids as well. The dummy cells are additional layers of grid cells or points outside the simulated physical domain. This is sketched in Fig. 8.1 for the case of a 2-D structured grid. As we can see, the computational domain is surrounded on all sides by two layers of dummy cells (marked by dashed lines). The dummy cells (points) are usually not generated like the grid inside the domain (except on interfaces between grid blocks). Rather, the cells are only virtual, although geometrical quantities like volumes or face vectors are associated with them.

The purpose of the dummy cells is to simplify the computation of the fluxes, gradients, dissipation, etc., along the boundaries. This is achieved by the possibility of extending the stencil of the spatial discretization scheme beyond the physical boundaries. As we can see in Fig. 8.1, the same discretization scheme can be employed at the boundaries as inside the physical domain. Thus, we can solve the governing equations in the same way for all “physical” grid points. This makes the discretization schemes much easier to implement. Furthermore, all grid points of a structured grid can be accessed

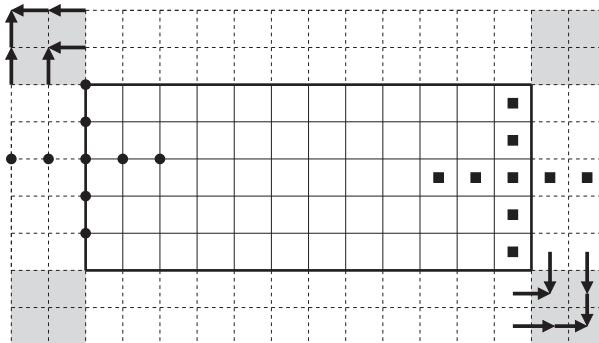


Figure 8.1 Two layers of dummy cells (dashed line) around the computational domain (thick line) in 2D. Filled circles represent the standard stencil of a second-order cell-vertex (dual) scheme, filled rectangles outline the stencil of a second-order cell-centered scheme (see Section 4.3).

in a single loop, which is of significant advantage in the case of loop parallelization or vectorization. The condition is of course that the dummy cells (points) contain appropriate values of the conservative variables as well as of the geometrical quantities. Clearly, the number of dummy-cell layers must be such that the part of the stencil outside the physical domain is completely covered. The conservative variables in the dummy cells (points) are obtained from boundary conditions. The geometrical quantities are usually taken from the corresponding control volume at the boundary. In the case of boundaries between multiple grid blocks (like in Fig. 3.4), all flow variables and the geometry are transferred from the neighboring block.

The gray-shaded dummy cells (and the associated points) in Fig. 8.1 represent a certain problem, since it is not quite clear how to set their values if there is no adjacent grid block. Their values are not required by the standard cross-type discretization stencil. However, the cells (points) are important for the computation of gradients (viscous fluxes—see Section 4.4), or for the transfer operators within multigrid (Section 9.4). The simplest way to solve the problem is to average the values from the adjacent “regular” dummy cells, as indicated in Fig. 8.1 by arrows. However, this does not work satisfactorily for wall or symmetry boundaries. In these cases, it is better to extend the physical boundary into the dummy-cell layers (see the provided structured 2-D code).

8.2 SOLID WALL

8.2.1 Inviscid flow

In the case of an inviscid flow, the fluid slips over the surface. Since there is no friction force, the velocity vector must be tangent to the surface. This is equivalent to the condition that there is no flow normal to the surface, that is,

$$\vec{v} \cdot \vec{n} = 0 \quad \text{at the surface,} \quad (8.1)$$

where \vec{n} denotes the unit normal vector at the surface. Hence, the contravariant velocity V (Eq. (2.22)) is zero at the wall. Consequently, the vector of convective fluxes (Eq. (2.21)) reduces to the pressure term alone, i.e.,

$$(\vec{F}_c)_w = \begin{bmatrix} 0 \\ n_x p_w \\ n_y p_w \\ n_z p_w \\ 0 \end{bmatrix} \quad (8.2)$$

with p_w being the wall pressure.

Structured cell-centered scheme

Within the cell-centered scheme, the pressure is evaluated at the centroid of the cell. However, p_w in Eq. (8.2) is required at the face of the boundary cell. We can obtain the wall pressure most easily by extrapolation from the interior of the domain. Considering Fig. 8.2, we could simply set $p_w = p_2$. Higher accuracy is achieved by using either a two-point

$$p_w = \frac{1}{2}(3p_2 - p_3), \quad (8.3)$$

or a three-point extrapolation formula

$$p_w = \frac{1}{8}(15p_2 - 10p_3 + 3p_4). \quad (8.4)$$

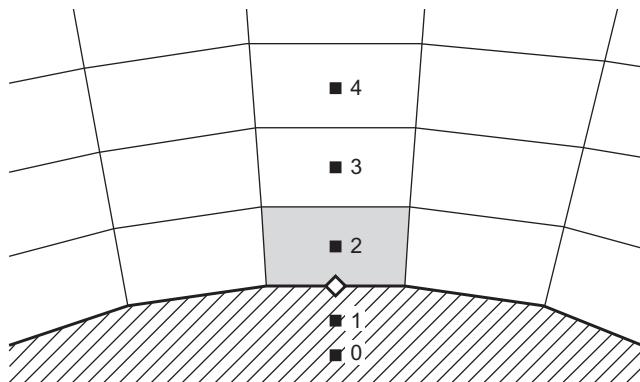


Figure 8.2 Solid wall boundary condition for the cell-centered scheme. Dummy cells are denoted as 0 and 1. Location, where the convective fluxes (Eq. (8.2)) are evaluated, is marked by a diamond.

In order to account for grid stretching, distances to the wall could be employed instead of the constant coefficients [1].

The above extrapolation formulas (8.3) and (8.4) do not account for the grid and the surface geometry. An alternative approach—the *normal-momentum relation*—was developed by Rizzi [2]. It is based on the fact that the wall represents a streamline in inviscid flow. Differentiation of the zero-normal-flow condition in Eq. (8.1) along the surface streamline, and the substitution of the result into the momentum equation yield

$$\rho \vec{v} \cdot (\vec{v} \cdot \vec{\nabla}) \vec{n} = \vec{n} \cdot \vec{\nabla} p. \quad (8.5)$$

Equation (8.5) relates the density, the velocity, and the wall geometry to the normal derivative of the pressure. It was demonstrated that the normal-momentum relation gives very accurate results [1]. However, problems can arise with the numerical solution of the normal-momentum relation in the case of complex geometries. Detailed description of the implementation and accuracy comparisons can be found in Ref. [1].

The values of the conservative variables in the dummy cells can be obtained by linear extrapolation from the interior, that is,

$$\begin{aligned} \vec{W}_1 &= 2\vec{W}_2 - \vec{W}_3, \\ \vec{W}_0 &= 3\vec{W}_2 - 2\vec{W}_3. \end{aligned} \quad (8.6)$$

The indexes in Eq. (8.6) correspond to Fig. 8.2. If the dummy cells are to be utilized within the spatial discretization scheme (e.g., for the evaluation of the dissipation operator), it is important that the calculation of the convective fluxes is compatible to Eq. (8.2).

Structured cell-vertex scheme

The implementation of the boundary condition (Eq. (8.1)) is straightforward for the cell-vertex discretization scheme with overlapping control volumes (Section 4.2.2). The convective fluxes at the wall faces are computed according to Eq. (8.2). The wall pressure p_w is obtained by averaging the nodal values as indicated in Eq. (4.26) for a 2-D, or in Eq. (4.27) for a 3-D case. The distribution formula (Eq. (4.30) in 2D) accounts now for only two (four in 3D) cells (compare Fig. 4.4 to Fig. 8.3).

Several different ways can be followed in the case of the cell-vertex scheme with dual control volumes (Section 4.2.3). One approach is to apply the condition in Eq. (8.2) separately for each face of the control volume which is on the wall. Thus, according to Fig. 8.3, we can write

$$(\vec{F}_{c,w})_{i,2} = \begin{bmatrix} 0 \\ (n_x)_{i-1,2} (p_w)_{i-1/4,2} \\ (n_y)_{i-1,2} (p_w)_{i-1/4,2} \\ (n_z)_{i-1,2} (p_w)_{i-1/4,2} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ (n_x)_{i,2} (p_w)_{i+1/4,2} \\ (n_y)_{i,2} (p_w)_{i+1/4,2} \\ (n_z)_{i,2} (p_w)_{i+1/4,2} \\ 0 \end{bmatrix}. \quad (8.7)$$

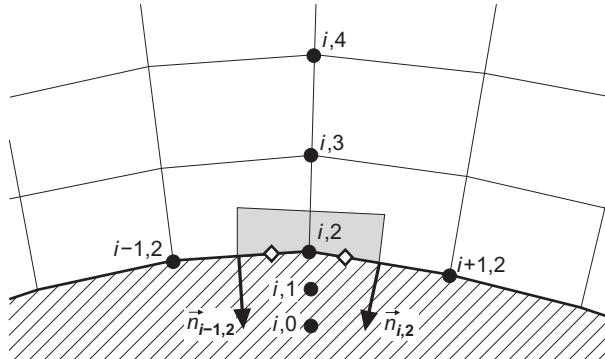


Figure 8.3 Solid wall boundary condition for the structured, cell-vertex (dual control volume) scheme. Dummy points are denoted as $(i, 0)$ and $(i, 1)$. Locations, where the convective fluxes (Eq. (8.2)) are evaluated, are marked by diamonds. Compare also to the sketch in Fig. 4.6.

The pressures $(p_w)_{i-1/4,2}$ and $(p_w)_{i+1/4,2}$ in Eq. (8.7) can be obtained by linear interpolation, for example,

$$(p_w)_{i+1/4,2} = \frac{1}{4} [3(p_w)_{i,2} + (p_w)_{i+1,2}]. \quad (8.8)$$

The corresponding 3-D formula will be presented in Unstructured median-dual scheme section (Eq. (8.11)).

Another possible implementation employs the condition in Eq. (8.2) directly in the respective wall node (i.e., node $(i, 2)$ in Fig. 8.3). The wall pressure p_w is simply set equal to $p_{i,2}$. The unit normal vector is computed as the average of the normal vectors of all wall facets which share the node $(i, 2)$. This approach requires a correction of the velocity vector. After the solution update by the time-stepping scheme, the velocity vector at the wall is projected onto the tangential plane [3, 4], that is,

$$(\vec{v}_{i,2})_{\text{corr}} = \vec{v}_{i,2} - [\vec{v}_{i,2} \cdot (\vec{n}_{\text{av}})_{i,2}] \cdot (\vec{n}_{\text{av}})_{i,2} \quad (8.9)$$

with \vec{n}_{av} being the averaged unit normal vector. In this way, the flow will become tangential to the wall.

In order to assign values to the dummy points, it is sufficient to extrapolate the conservative variables (Eq. (2.20)) from the interior field by using a relation similar to Eq. (8.6).

Unstructured cell-centered scheme

The wall boundary condition in Eq. (8.1) can be implemented for a cell-centered unstructured scheme in a way similar to that on structured grids. If the boundary cell is

a quadrilateral, hexahedron, or a prism (with triangular face on the wall), the pressure can be extrapolated to the wall by using Eq. (8.3). The neighboring cell (number 3 in Fig. 8.2) is known from the face-based data structure described in Section 5.2.1. For the case of a triangular or tetrahedral cell, it was suggested in Refs. [5, 6] to employ one layer of dummy cells. The velocity components in the dummy cells were obtained by reflecting the velocity vectors in the boundary cells at the wall. For example, in the dummy cell 1 in Fig. 8.2, the velocity would become

$$\vec{v}_1 = \vec{v}_2 - 2 V_2 \vec{n}, \quad (8.10)$$

where $V_2 = u_2 n_x + v_2 n_y + w_2 n_z$ is the contravariant velocity and $\vec{n} = [n_x, n_y, n_z]^T$ stands for the wall unit-normal vector. The pressure and density in the dummy cells were set equal to the values in the corresponding boundary cell (this implies $p_w = p_2$).

Unstructured median-dual scheme

The boundary condition (Eq. (8.1)) requires more attention in the case of the median-dual unstructured discretization scheme. The situation is shown in Fig. 8.4 for the 2-D and in Fig. 8.5 for the 3-D case. The convective fluxes in Eq. (8.2) are computed separately at each face of the control volume which is located on the wall. This is identical to the first approach discussed above for the structured cell-vertex scheme with dual control volumes. For quadrilateral elements (like 1-3-4-5 in Fig. 8.4), the pressure is interpolated correspondingly to Eq. (8.8). In the case of hexahedra, prisms or pyramids, where the face of the control volume is a quadrilateral (like face 1-4-5-6 in Fig. 8.5), the interpolation formula reads

$$p_w = \frac{1}{16}(9p_1 + 3p_4 + 3p_6 + p_5). \quad (8.11)$$

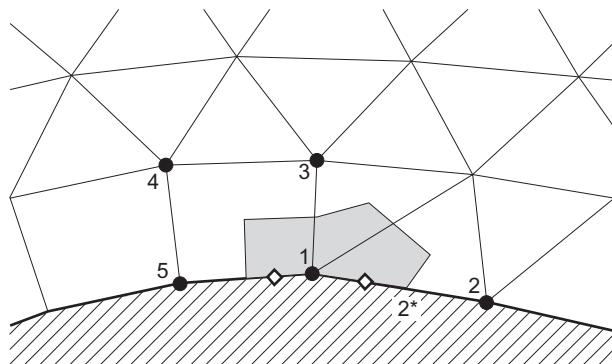


Figure 8.4 Solid wall boundary condition for the 2-D unstructured, dual-control volume mixed-grid scheme. Locations, where the convective fluxes (Eq. (8.2)) are evaluated, are marked by diamonds.

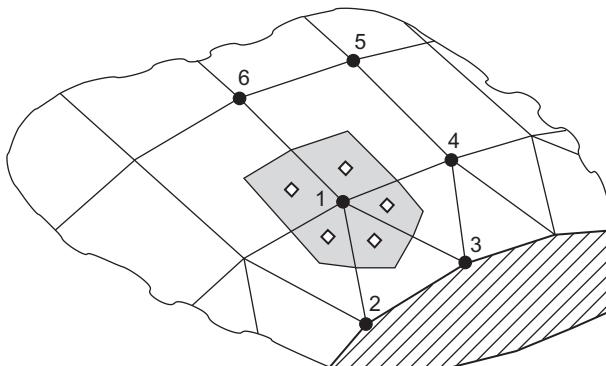


Figure 8.5 Solid wall boundary condition for the 3-D unstructured, mixed-grid scheme. Locations, where the convective fluxes (Eq. (8.2)) are evaluated, are marked by diamonds.

If the boundary elements are tetrahedra or prisms (or triangles in 2-D), the pressure at the wall face should be evaluated like in the finite-element method [7]. At the wall segment 1-2 in Fig. 8.4, for example, the pressure at the face $1-2^*$ would be computed as

$$p_w = \frac{1}{6}(5p_1 + p_2). \quad (8.12)$$

In the case of a tetrahedra with, for example, the wall face 1-2-3 in Fig. 8.5, the pressure is evaluated as

$$p_w = \frac{1}{8}(6p_1 + p_2 + p_3). \quad (8.13)$$

It should be stressed that this finite-element type of interpolation is important for an accurate flow solution [7].

8.2.2 Viscous flow

For a viscous fluid which passes a solid wall, the relative velocity between the surface and the fluid directly at the surface is assumed to be zero. Therefore, we speak of *noslip* boundary condition. In the case of a stationary wall surface, the Cartesian velocity components become

$$u = v = w = 0 \quad \text{at the surface.} \quad (8.14)$$

There are two basic consequences of the noslip condition. First, we do not need to solve the momentum equations on the wall. This fact is utilized in the cell-vertex scheme. Second, the convective fluxes through the noslip wall are given again by Eq. (8.2), and the terms in Eq. (2.24) simplify to $\vec{\Theta} = k\vec{\nabla}T$. Hence, the wall pressure in the convective fluxes is obtained in the same way as described above for the inviscid flow. However, the dummy cells (points) are treated differently.

Cell-centered scheme

The implementation of the noslip boundary condition in Eq. (8.14) can be simplified by the utilization of dummy cells. In the case of an adiabatic wall (no heat flux through the wall), we can set (see Fig. 8.2)

$$\begin{aligned}\rho_1 &= \rho_2, & E_1 &= E_2 \\ u_1 &= -u_2, & v_1 &= -v_2, & w_1 &= -w_2\end{aligned}\tag{8.15}$$

and likewise for the cells 0 and 3. The approach is applicable to both, structured and unstructured schemes (cf. Ref. [6]).

If the wall temperature is given, the velocity components are still reversed as in Eq. (8.15). The temperature in the dummy cells is linearly extrapolated from the interior field by using the specified wall temperature. Since the pressure gradient normal to the wall is zero, the pressure in the boundary element is prescribed also in the dummy cells (i.e., $p_0 = p_1 = p_2$). The density and the total energy in the dummy cells are evaluated from the interpolated values.

Cell-vertex scheme

Since the momentum equations need not to be solved, there is no contribution from the convective fluxes (Eq. (8.2)) at the wall. The viscous fluxes in Eq. (2.23) contribute only the temperature gradient normal to the wall to the energy equation. For an adiabatic wall, $\vec{\nabla} T_w \cdot \vec{n}$ is zero. Hence, we do not have to compute any convective or viscous fluxes through the wall. The residuals of the momentum equations should be set to zero, in order to prevent the generation of nonzero velocity components at the wall nodes.

In the case of a prescribed wall temperature, we can directly set the total energy at the wall (e.g., node $(i, 2)$ in Fig. 8.3) using (perfect gas assumed)

$$(\rho E)_{i,2} = \frac{c_p}{\gamma} \rho_{i,2} T_w,\tag{8.16}$$

where T_w denotes the given wall temperature. The residuals of the momentum and of the energy equation have to be zeroed out. The same strategy is applicable also to unstructured schemes.

Another approach for non-adiabatic walls, which seems to be more robust for some applications, does not solve the governing equations at the wall at all. Both, the density and the energy are specified directly

$$\rho_{i,2} = \frac{p_{i,3}}{T_w R} \quad \text{and} \quad (\rho E)_{i,2} = \frac{p_{i,3}}{\gamma}.\tag{8.17}$$

The relations in Eq. (8.17) assume that there is no pressure gradient normal to the wall (therefore $p_{i,2} = p_{i,3}$). Since all conservative variables are prescribed, the residuals of all

equations should be set to zero at $(i, 2)$. This technique can be utilized on unstructured grids as well. However, the extrapolation of the pressure requires additional operations on triangular or tetrahedral grids.

If the wall is adiabatic, the values in the dummy points are obtained as follows

$$\begin{aligned}\rho_{i,1} &= \rho_{i,3}, & E_{i,1} &= E_{i,3} \\ u_{i,1} &= -u_{i,3}, & v_{i,1} &= -v_{i,3}, & w_{i,1} &= -w_{i,3}.\end{aligned}\quad (8.18)$$

The same applies to the nodes 0 and 4. If the wall temperature is given, the temperature in the dummy points is extrapolated from the interior, that is,

$$T_{i,1} = 2T_w - T_{i,3} \quad \text{and} \quad T_{i,0} = 3T_w - 2T_{i,3} \quad (8.19)$$

with the indexes according to Fig. 8.3. The velocity components are again reversed as in Eq. (8.18). The density and energy are computed with the interpolated temperature value and with the pressure $p_{i,3}$.

8.3 FAR-FIELD

The numerical simulation of external flows past airfoils, wings, cars, and other configurations has to be conducted within a bounded domain. For this reason, artificial far-field boundary conditions become necessary. The numerical implementation of the far-field boundary conditions has to fulfill two basic requirements. First, the truncation of the domain should have no noticeable effects on the flow solution as compared to the infinite domain. Second, any outgoing disturbances must not be reflected back into the flow field [9]. Due to their elliptic nature, sub- and transonic flow problems are particularly sensitive to the far-field boundary conditions. An inadequate implementation can lead to a significant slowdown of convergence to the steady state. Furthermore, the accuracy of the solution is likely to be negatively influenced. Various methodologies were developed, which are capable of absorbing the outgoing waves at the artificial boundaries [10–15]. A review of different non-reflecting boundary conditions can be found in Ref. [16].

In the following two sections, we shall discuss the concept of characteristic variables as it was described by Whitfield and Janus [12]. We shall also present an extension of the far-field boundary conditions for lifting bodies.

8.3.1 Concept of characteristic variables

Depending on the sign of the eigenvalues of the convective flux Jacobians (Section A.11, Eq. (A.84) or (A.88)), the information is transported out of, or into the computational domain along the characteristics. For example, in the case of subsonic inflow there are four incoming characteristics (in 3D) and one outgoing (Λ_5 in Eq. (A.88)). The situation reverses for subsonic outflow. According to the 1-D theory of Kreiss [17], the number of conditions to be imposed from outside at the boundary should be equal to the number

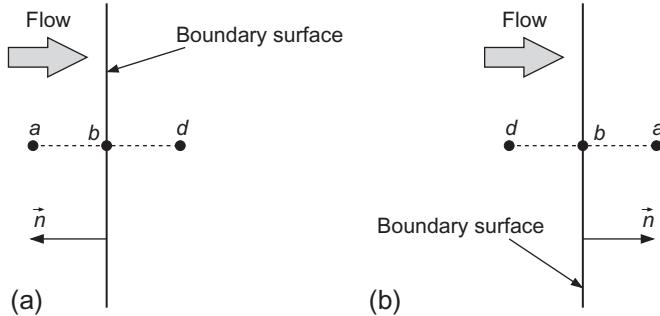


Figure 8.6 Far-field boundary: inflow (a) and outflow (b) situation. Position a is outside, b on the boundary, and position d is inside the physical domain. The unit normal vector $\vec{n} = [n_x, n_y, n_z]^T$ points out of the domain.

of *incoming* characteristics. The remaining conditions should be determined from the solution inside the domain.

The approach of Whitfield and Janus [12] is based on the characteristic form of the Euler equations (2.44) normal to the boundary. The methodology was found to perform very well on structured and unstructured grids in a large variety of flow cases. It can be applied not only to far-field boundaries but also to inviscid solid walls (Section 8.2.1).

The two basic flow situations at the far-field boundary are sketched in Fig. 8.6. The flow can either enter or it can leave the domain. Therefore, depending on the local Mach number, four different types of far-field boundary conditions have to be treated:

- supersonic inflow,
- supersonic outflow,
- subsonic inflow,
- subsonic outflow.

Supersonic inflow

For supersonic inflow, all eigenvalues have the same sign. Since the flow is entering the physical domain, the conservative variables on the boundary (point b in Fig. 8.6) are determined by free-stream values only. Thus,

$$\vec{W}_b = \vec{W}_a. \quad (8.20)$$

The values \vec{W}_a are specified based on the given Mach number M_∞ and on two flow angles (angle of attack, side-slip angle).

Supersonic outflow

In this case, all eigenvalues have also the same sign. However, the flow leaves now the physical domain and all conservative variables at the boundary must be determined from the solution inside the domain. This can be accomplished by simply setting

$$\vec{W}_b = \vec{W}_d, \quad (8.21)$$

which corresponds to constant extrapolation. Linear extrapolation is also possible, but it may lead to a less stable solution scheme.

Subsonic inflow

Here, four characteristics enter and one leaves the physical domain. Therefore, four characteristic variables are prescribed based on the free-stream values. One characteristic variable is extrapolated from the interior of the physical domain. This leads to the following set of boundary conditions [12]

$$\begin{aligned} p_b &= \frac{1}{2} \left\{ p_a + p_d - \rho_0 c_0 \left[n_x(u_a - u_d) + n_y(v_a - v_d) + n_z(w_a - w_d) \right] \right\} \\ \rho_b &= \rho_a + (p_b - p_a)/c_0^2 \\ u_b &= u_a - n_x(p_a - p_b)/(\rho_0 c_0) \\ v_b &= v_a - n_y(p_a - p_b)/(\rho_0 c_0) \\ w_b &= w_a - n_z(p_a - p_b)/(\rho_0 c_0), \end{aligned} \quad (8.22)$$

where ρ_0 and c_0 represent a reference state. The reference state is normally set equal to the state at the interior point (point d in Fig. 8.6). The values in point a are determined from the free-stream state.

Subsonic outflow

In the case of subsonic outflow, four flow variables (density and the three velocity components) have to be extrapolated from the interior of the physical domain. The remaining fifth variable (pressure) must be specified externally. The primitive variables at the far-field boundary are obtained from Ref. [12]

$$\begin{aligned} p_b &= p_a \\ \rho_b &= \rho_d + (p_b - p_d)/c_0^2 \\ u_b &= u_d + n_x(p_d - p_b)/(\rho_0 c_0) \\ v_b &= v_d + n_y(p_d - p_b)/(\rho_0 c_0) \\ w_b &= w_d + n_z(p_d - p_b)/(\rho_0 c_0) \end{aligned} \quad (8.23)$$

with p_a being the prescribed static pressure.

Physical properties in the dummy cells/nodes can be obtained by linear extrapolation from the states b and d .

8.3.2 Modifications for lifting bodies

The above characteristic far-field boundary conditions assume zero circulation, which is not correct for a lifting body in sub- or transonic flow. For this reason, the far-field boundary has to be located very far away from the body. Otherwise, the flow solution will

be inaccurate. The distance to the far-field can be significantly shortened (by one order of magnitude), if the free-stream flow includes the effect of a single vortex (horse-shoe vortex in 3D). The vortex is assumed to be centered at the lifting body. The strength of the vortex is proportional to the lift produced by the body. In the following section, we shall present implementations of the vortex correction in 2D and in 3D.

Vortex correction in 2D

The approach, which we want to describe here, was suggested by Usab and Murman [18]. The components of the corrected free-stream velocity are given by the expressions (compressible flow assumed)

$$\begin{aligned} u_{\infty}^* &= u_{\infty} + \left(\frac{\Gamma \sqrt{1 - M_{\infty}^2}}{2\pi d} \right) \frac{1}{1 - M_{\infty}^2 \sin^2(\theta - \alpha)} \sin \theta, \\ v_{\infty}^* &= v_{\infty} - \left(\frac{\Gamma \sqrt{1 - M_{\infty}^2}}{2\pi d} \right) \frac{1}{1 - M_{\infty}^2 \sin^2(\theta - \alpha)} \cos \theta \end{aligned} \quad (8.24)$$

with Γ being the circulation, (d, θ) the polar coordinates of the far-field point, α the angle of attack, and M_{∞} denoting the free-stream Mach number, respectively. The circulation is obtained from

$$\Gamma = \frac{1}{2} \|\vec{v}_{\infty}\|_2 a C_L \quad (8.25)$$

by using the theorem of Kutta-Joukowsky. In Eq. (8.25), a represents the chord length of the airfoil and C_L is the lift coefficient evaluated by the integration of the surface pressure. The polar coordinates in Eq. (8.24) are computed as

$$\begin{aligned} d &= \sqrt{(x - x_{\text{ref}})^2 + (\gamma - \gamma_{\text{ref}})^2} \\ \theta &= \tan \left(\frac{\gamma - \gamma_{\text{ref}}}{x - x_{\text{ref}}} \right), \end{aligned} \quad (8.26)$$

where x_{ref} and γ_{ref} are the coordinates of the reference point (location of the vortex—for example, at 1/4 chord).

The modified free-stream pressure p_{∞}^* is given by

$$p_{\infty}^* = \left[p_{\infty}^{(\gamma-1)/\gamma} + \left(\frac{\gamma - 1}{\gamma} \right) \frac{\rho_{\infty} (\|\vec{v}_{\infty}\|_2^2 - \|\vec{v}_{\infty}^*\|_2^2)}{2 p_{\infty}^{1/\gamma}} \right]^{\gamma/(\gamma-1)} \quad (8.27)$$

with $\|\vec{v}_{\infty}^*\|_2^2 = (u_{\infty}^*)^2 + (v_{\infty}^*)^2$. The corrected free-stream density is obtained from the equation of the state

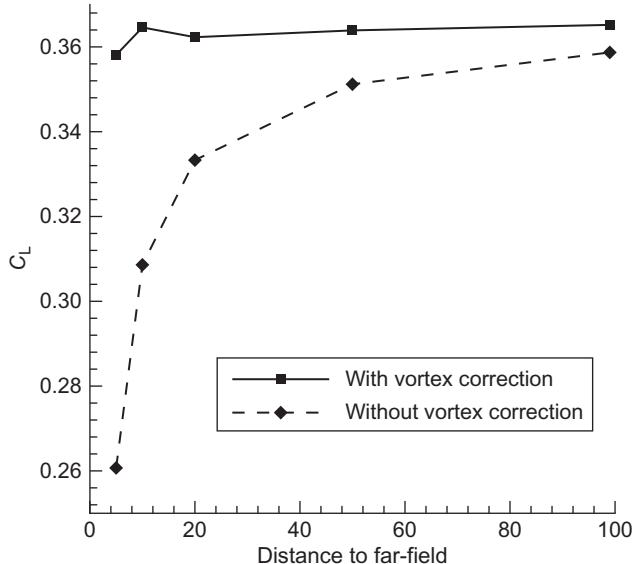


Figure 8.7 Effects of distance to the far-field boundary and of single vortex on the lift coefficient. NACA 0012 airfoil, $M_\infty = 0.8$, $\alpha = 1.25^\circ$.

$$\rho_\infty^* = \rho_\infty \left(\frac{p_\infty^*}{p_\infty} \right)^{1/\gamma}. \quad (8.28)$$

The corrected quantities u_∞^* , v_∞^* , p_∞^* , and ρ_∞^* are inserted into Eq. (8.22) or (8.23) instead of u_a , v_a , p_a , and ρ_a .

The above vortex correction (Eqs. (8.24)–(8.28)) is strictly valid for subsonic flow only. However, the modification of the free-stream conditions proved to be helpful in transonic flow as well. This is demonstrated in Fig. 8.7, where the dependence of the lift coefficient on the distance to the far-field boundary was investigated. The far-field radius was set to 5, 20, 50, and 99 chord lengths. As we can see, simulations without the vortex correction experiences a strong dependence on the far-field distance. On the contrary, simulations with the vortex remain sufficiently accurate up to a distance of about 20 chords. This leads to a significant reduction of the number of grid cells/points. It was demonstrated in Ref. [19] that by using higher-order terms in the vortex correction, the far-field boundary can be placed only about five chords away without any noticeable loss of accuracy.

Vortex correction in 3D

The effect of a wing on the far-field boundary can be approximated by a horse-shoe vortex. In the case of compressible flow, the modified free-stream velocity components can be obtained from the relations [20, 21]

$$\begin{aligned} u_{\infty}^* &= u_{\infty} + \frac{\Gamma \beta^2}{2\pi} \mathcal{A}, \\ v_{\infty}^* &= v_{\infty} - \frac{\Gamma}{2\pi} \left[\frac{z+l}{(z+l)^2 + \gamma^2} \mathcal{B} - \frac{z-l}{(z-l)^2 + \gamma^2} \mathcal{C} + \frac{x\beta^2}{x^2 + \gamma^2 \beta^2} \mathcal{A} \right], \\ w_{\infty}^* &= w_{\infty} + \frac{\Gamma}{2\pi} \left[\frac{\gamma}{(z+l)^2 + \gamma^2} \mathcal{B} - \frac{\gamma}{(z-l)^2 + \gamma^2} \mathcal{C} \right], \end{aligned} \quad (8.29)$$

where Γ denotes the circulation, (x, y, z) the Cartesian coordinates of the far-field point, and l stands for the half span, respectively. Furthermore, in Eq. (8.29) it was assumed that the flow is oriented in the positive x -direction, with the wing being positioned along the z -axis. The terms \mathcal{A} , \mathcal{B} , and \mathcal{C} in Eq. (8.29) read

$$\begin{aligned} \mathcal{A} &= \frac{z+l}{\sqrt{\psi_+}} - \frac{z-l}{\sqrt{\psi_-}}, \\ \mathcal{B} &= 1 + \frac{x}{\sqrt{\psi_+}}, \\ \mathcal{C} &= 1 + \frac{x}{\sqrt{\psi_-}}. \end{aligned} \quad (8.30)$$

The abbreviations are given by

$$\begin{aligned} \psi_+ &= x^2 + \beta^2(z+l)^2 + \gamma^2\beta^2, \\ \psi_- &= x^2 + \beta^2(z-l)^2 + \gamma^2\beta^2, \\ \beta &= \sqrt{1 - M_{\infty}^2} \end{aligned} \quad (8.31)$$

with M_{∞} being the free-stream Mach number. The circulation Γ is calculated using Eq. (8.25), where a represents the mean chord. The corrected values of pressure (p_{∞}^*) and of density (ρ_{∞}^*) are obtained from the formulas (8.27) and (8.28), respectively. The quantities u_a , v_a , w_a , p_a , and ρ_a in Eq. (8.22) or Eq. (8.23) are replaced by their corrected values u_{∞}^* , v_{∞}^* , w_{∞}^* , p_{∞}^* , and ρ_{∞}^* .

The expressions for the corrected velocity components v_{∞}^* and w_{∞}^* in Eq. (8.29) become infinite at locations, where the vortex lines cross the outflow boundary. These are the points

$$z = +l, \quad \gamma = 0,$$

$$z = -l, \quad \gamma = 0,$$

and $x = x_{\text{farf}}$. In order to avoid the numerical singularity, it was suggested in Ref. [21] to constrain the values of

$$\begin{aligned} (z+l)^2 + \gamma^2, \\ (z-l)^2 + \gamma^2 \end{aligned}$$

in Eq. (8.29) to $1/4$ wingspan, that is, to $l/2$. This simple measure reduces the corrections to the velocities v_∞ and w_∞ within the distance of $l/2$ around the vortex lines $z = l$ and $z = -l$.

The numerical results presented in Ref. [21] indicate a reduced sensitivity of the lift and drag coefficient with respect to the far-field distance, if the vortex correction in Eq. (8.29) is applied. It was found that a distance of $7 \cdot l$ to the far-field boundary is sufficient for accurate results.

8.4 INLET/OUTLET BOUNDARY

Various approaches were devised for the implementation of numerical inlet, and in particular, of outlet (also named open) boundary conditions for the Navier-Stokes equations [22–26]. Here, we will concentrate on methodologies, which were developed for turbomachinery applications. Suitable non-reflecting inlet and outlet boundary conditions were described, for example, in Refs. [27–30]. Giles [31] and Hirsch and Verhoff [32] suggested non-reflecting boundary conditions for the Euler equations, which are intended for domains with a short distance between the body and the inlet or the outlet plane.

In certain cases, the inlet and outlet boundary are in addition periodic with respect to the velocity as well as the pressure and temperature gradient. This type of flow is encountered, for example, in the simulation of heat exchangers [33]. The implementation of periodic inlet and outlet boundary conditions was presented in Refs. [34–36] for LES in channels.

Subsonic inlet

A common procedure consists of the specification of the total pressure, total temperature, and of two flow angles. One characteristic variable has to be interpolated from the interior of the flow domain. One possibility is to employ the outgoing Riemann invariant [30], which is defined as

$$\mathcal{R}^- = \vec{v}_d \cdot \vec{n} - \frac{2 c_d}{\gamma - 1}, \quad (8.32)$$

where the index d denotes the state inside the domain (cf. Fig. 8.6a). The Riemann invariant is used to determine either the absolute velocity or the speed of sound at the boundary. In practice, it was found that selecting the speed of sound leads to a more stable scheme, particularly for low Mach-number flows. Therefore, we set

$$c_b = \frac{-\mathcal{R}^-(\gamma - 1)}{(\gamma - 1)\cos^2\theta + 2} \left\{ 1 + \cos\theta \sqrt{\frac{[(\gamma - 1)\cos^2\theta + 2]c_0^2}{(\gamma - 1)(\mathcal{R}^-)^2} - \frac{\gamma - 1}{2}} \right\} \quad (8.33)$$

with θ being the flow angle relative to the boundary, and c_0 denoting the stagnation speed of sound. Hence,

$$\cos \theta = -\frac{\vec{v}_d \cdot \vec{n}}{\|\vec{v}_d\|_2} \quad (8.34)$$

and

$$c_0^2 = c_d^2 + \frac{\gamma - 1}{2} \|\vec{v}_d\|_2^2, \quad (8.35)$$

where $\|\vec{v}_d\|_2$ denotes the total velocity at the interior point d (Fig. 8.6a). The unit normal vector \vec{n} in Eq. (8.34) was assumed to point out of the domain.

Quantities like the static temperature, pressure, density, or the absolute velocity at the boundary are evaluated as follows:

$$\begin{aligned} T_b &= T_0 \left(\frac{c_b^2}{c_0^2} \right), \\ p_b &= p_0 \left(\frac{T_b}{T_0} \right)^{\gamma/(\gamma-1)}, \\ \rho_b &= \frac{p_b}{RT_b}, \\ \|\vec{v}_b\|_2 &= \sqrt{2 c_p (T_0 - T_b)}, \end{aligned} \quad (8.36)$$

where T_0 and p_0 are the given values of total temperature and pressure, R and c_p represent the specific gas constant and the heat coefficient at constant pressure, respectively. The velocity components at the inlet are obtained by decomposing $\|\vec{v}_b\|_2$ according to the two (one in 2D) prescribed flow angles.

Subsonic outlet

In turbomachinery, static pressure is usually prescribed at the outlet. The subsonic outlet boundary can be treated in a way quite similar to the outflow condition in Eq. (8.23). Only the ambient pressure p_a is replaced here by the given static exit pressure.

Flow variables in the dummy cells (points) can be obtained by linearly extrapolating the states at the boundary and at the interior point d .

Supersonic inlet and outlet

In these cases, all characteristics are either incoming or outgoing. Therefore, all conservative variables have to be prescribed at the inlet like in Eq. (8.20), and extrapolated at the outlet according to Eq. (8.21).

8.5 INJECTION BOUNDARY

The injection velocity and other flow variables are computed from the given mass flow rate \dot{m} and the injection temperature T_{inj} . We will assume here that the mass is injected perpendicular to the boundary. The convective fluxes are evaluated using Eq. (2.21) with the contravariant velocity V set equal to the injection velocity, that is,

$$V = V_{\text{inj}} = \frac{\dot{m}}{\rho_{\text{inj}}}. \quad (8.37)$$

Velocity components are evaluated using the face normal vector as

$$\vec{v}_{\text{inj}} = -\vec{n} V_{\text{inj}} \quad (8.38)$$

(see Fig. 8.8). The density ρ_{inj} in Eq. (8.37) is a function of T_{inj} and of the pressure at the boundary p_b . Hence, in the case of a perfect gas we have

$$\rho_{\text{inj}} = \frac{p_b}{R T_{\text{inj}}}. \quad (8.39)$$

For the cell-centered scheme sketched in Fig. 8.8a, p_b can be set identical to the pressure in the boundary cell, that is, $p_b = p_2$. The median-dual cell-vertex scheme requires an interpolation of p_b from the points defining the particular face area (i.e., from $(i, 2)$ and $(i+1, 2)$ in Fig. 8.8b). Formulas like (8.8), (8.11)–(8.13) can be utilized for this purpose. Values in the dummy cells (points) are obtained either by constant or by linear extrapolation from the interior.

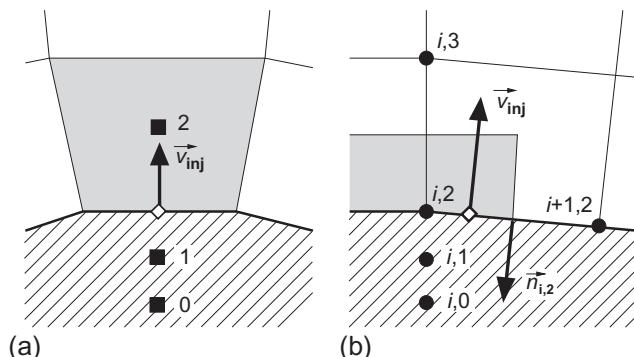


Figure 8.8 Injection boundary condition for the structured, cell-centered (a) and the cell-vertex dual control-volume scheme (b). Dummy cells (points) are denoted as 1 and 0. Locations, where the convective fluxes are evaluated, are marked by diamonds.

8.6 SYMMETRY PLANE

If the flow is to be symmetrical with respect to a line or a plane, the first condition which must be met is that there is no flux across the boundary. This is equivalent to the requirement that the velocity normal to the symmetry boundary is zero. Furthermore, the following gradients have to vanish:

- gradient of a scalar quantity normal to the boundary,
- gradient of the tangential velocity normal to the boundary,
- gradient of the normal velocity along the boundary (since $\vec{v} \cdot \vec{n} = 0$).

We can write these conditions as

$$\begin{aligned}\vec{n} \cdot \vec{\nabla} U &= 0, \\ \vec{n} \cdot \vec{\nabla}(\vec{v} \cdot \vec{t}) &= 0, \\ \vec{t} \cdot \vec{\nabla}(\vec{v} \cdot \vec{n}) &= 0,\end{aligned}\tag{8.40}$$

where U stands for a scalar variable and \vec{t} denotes a vector tangential to the symmetry boundary.

Cell-centered scheme

The implementation of the symmetry boundary condition can be largely simplified by employing dummy cells. The flow variables in the dummy cells are obtained using the concept of *reflected cells*. This means that scalar quantities like density or pressure in the dummy cells are set equal to the values in the opposite interior cells, that is,

$$U_1 = U_2 \quad \text{and} \quad U_0 = U_3.\tag{8.41}$$

The notation corresponds to that in Fig. 8.2. The velocity components are reflected with respect to the boundary as indicated in Eq. (8.10). The normal gradient of the normal velocity in the dummy cell equals to that in the opposite interior cell, but it has a reversed sign.

Cell-vertex scheme (dual control volume)

Two different approaches can be followed. One possibility is to construct the missing half of the control volume by mirroring the grid on the boundary. The fluxes and the gradients are then evaluated like in the interior using reflected flow variables (see above). The second methodology computes the fluxes for the halved control volume (but not across the boundary). The components of the residual normal to the symmetry plane are then zeroed out. It is also necessary to correct normal vectors of those faces

of the control volume, which touch the boundary (like at point 2* in Fig. 8.4). The modification consists of removing all components of the face vector, which are normal to the symmetry plane. The gradients also have to be corrected according to Eq. (8.40).

8.7 COORDINATE CUT

This type of boundary condition is encountered only in the case of structured grids. The coordinate cut represents an artificial, not a physical, boundary. It is a line (plane in 3D) composed of grid points with different computational coordinate(s) but the same physical location. This means that the grid is folded such that it touches itself. As we shall see in Section 11.1.1, the coordinate cut appears for the C- (Fig. 11.5) or O-grid topology (Fig. 11.9). The flow variables and their gradients have to stay continuous across the cut.

The best way to implement coordinate cuts is to employ dummy cells (points). The situation is sketched in Fig. 8.9. As we can see, the dummy layers here are not virtual, but they coincide with the grid on the opposite side of the cut. Hence, the values of physical quantities in the dummy cells (cell-centered scheme), or in the dummy points (cell-vertex scheme), are obtained directly from the opposite cells (points). In the case of the cell-centered scheme, the fluxes across the faces of the boundary cell (shaded in Fig. 8.9a) are evaluated exactly like in the interior field.

The cut boundary can be treated in two different ways for the cell-vertex scheme. One possibility is to generate a complete control volume at the cut (the second part is denoted by a dashed line in Fig. 8.9b). Using the dummy points, the fluxes can be calculated in the same way as inside the domain. If the implementation is done correctly, the flow quantities at the points 2 (upper grid part) and 5 (lower part) will be equal. The second approach is to integrate the fluxes separately for each half of the control volume.

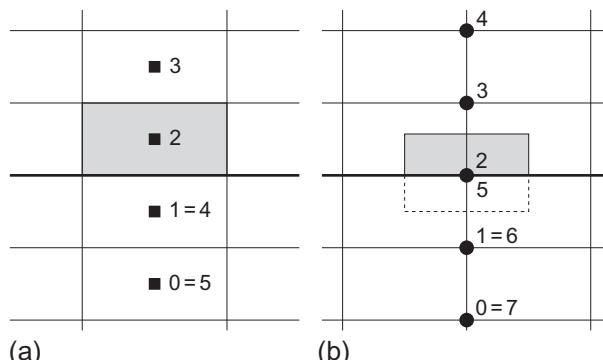


Figure 8.9 Coordinate cut (thick line): cell-centered scheme (a), dual control-volume scheme (b). Dummy cells (points) are numbered as 0 and 1.

The residuals at the points 2 and 5 in Fig. 8.9b are then added. It is important that the partial control volumes at the points 2 and 5 are summed up as well.

8.8 PERIODIC BOUNDARIES

There are certain practical applications where the flow field is periodic with respect to one or multiple coordinate directions. In such a case, it is sufficient to simulate the flow only within one of the repeating regions. The correct interaction with the remaining physical domain is enforced via periodic boundary conditions.

We can distinguish between two basic types of periodic boundaries. The first one covers *translational* periodicity. This means that one periodic boundary can be transformed into the other boundary by pure coordinate translation. The second type represents periodic boundaries, which were generated by coordinate rotation. Thus, we speak of *rotational* periodicity.

In the following section, we shall describe the implementation of the periodic boundary conditions for the cell-centered and the cell-vertex schemes. We shall also consider the case of rotational periodicity. Further details of the treatment of periodic boundaries can be found in Refs. [37, 38].

Cell-centered scheme

The utilization of the dummy-cell concept enables a simple implementation of the periodic boundary condition. Let us consider the example from turbomachinery in Fig. 8.10. The configuration is periodic in the vertical direction. The shaded cells 1 and 2 are located on the lower and the upper periodic boundaries, respectively. Due to the periodicity condition, the first dummy-cell layer corresponds to the boundary cells at

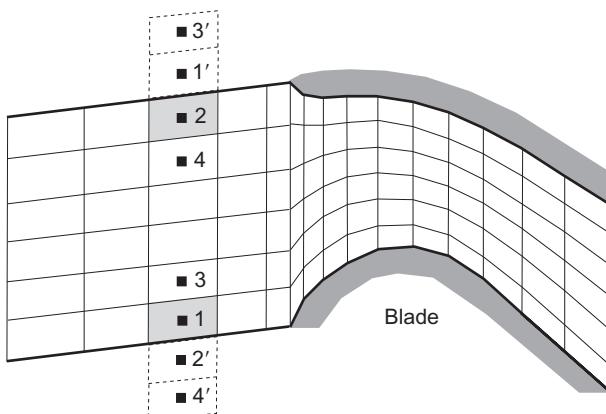


Figure 8.10 Periodic boundaries (thick lines) in the case of 2-D unstructured, cell-centered scheme. Dummy cells (dashed line) are denoted by the (primed) numbers of the corresponding physical cells.

the opposite periodic boundary. The second dummy-cell layer communicates with the second layer of the physical cells and so on. Hence, all scalar quantities (density, pressure, etc.) in the dummy cells are obtained directly from the corresponding physical cells, i.e,

$$U_{1'} = U_1 \quad \text{and} \quad U_{2'} = U_2. \quad (8.42)$$

The same relations hold also for the vector quantities (velocity, gradients) in the case of translational periodicity. Rotational-periodic boundaries require a correction of the vector variables. This will be discussed further in the following section.

Cell-vertex scheme (dual control volume)

This situation is sketched in Fig. 8.11. One approach for the treatment of periodic boundaries consists of the integration of the fluxes around the faces of the shaded control volumes. The residuals at the points 1 and 2 in Fig. 8.11 are then summed up in order to obtain the complete flux. Thus,

$$\vec{R}_{1,\text{sum}} = \vec{R}_1 + \vec{R}_{2'} \quad \text{and} \quad \vec{R}_{2,\text{sum}} = \vec{R}_2 + \vec{R}_{1'}. \quad (8.43)$$

The partial control volumes at the points 1 and 2 (shaded in Fig. 8.11) have to be added up as well. In the case of translational periodicity, the residuals from the opposite boundary remain unchanged, that is, $\vec{R}_{1'} = \vec{R}_1$ and $\vec{R}_{2'} = \vec{R}_2$. This results in $\vec{R}_{1,\text{sum}} = \vec{R}_{2,\text{sum}}$. Rotationally periodic boundaries require a transformation of the momentum equations before Eq. (8.43) can be applied.

Rotational periodicity

The rotational periodicity condition is based on a rotation of the coordinate system. Therefore, all vector quantities like velocity or gradients of scalars have to be transformed

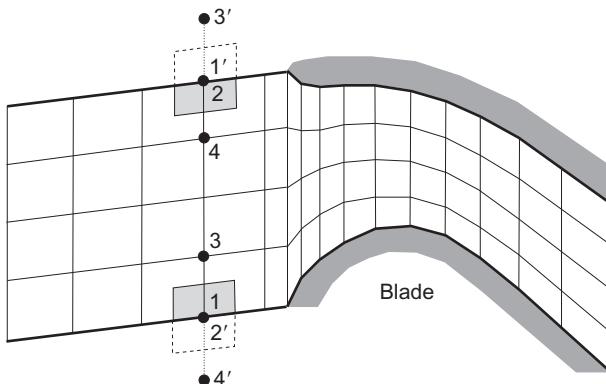


Figure 8.11 Periodic boundaries (thick lines) in the case of 2-D unstructured, cell-vertex scheme with dual control volumes. The “dummy” parts of the control volumes (dashed line) are denoted by the (primed) numbers of the corresponding control volumes at the opposite boundary. The same holds also for the dummy points 3' and 4'.

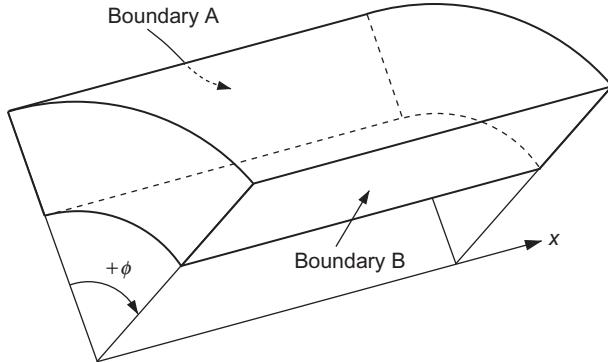


Figure 8.12 Rotationally periodic boundaries (A and B). The rotational axis is assumed to coincide with the x -axis.

accordingly. Scalar quantities like pressure or density, which are invariant with respect to coordinate rotation, remain unchanged. If we assume the rotational axis is parallel to the x -axis (see Fig. 8.12), the rotation matrix becomes

$$\bar{\mathcal{R}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix}, \quad (8.44)$$

where the angle ϕ between the periodic boundaries A and B is positive in the clockwise direction. Hence, for example, the velocity vector transformed from boundary A to B (cells 1', 2' in Fig. 8.10 and points 1'-4' in Fig. 8.11) reads

$$\vec{v}_B = \bar{\mathcal{R}} \vec{v}_A. \quad (8.45)$$

It is easy to show that the x -component of \vec{v}_A (i.e., u_A) is not changed by the rotation. Thus, $u_B = u_A$. The gradients of all flow quantities are transformed in a similar way.

As stated above, in the case of the cell-vertex scheme the residuals of the momentum equations must be corrected before the summation in Eq. (8.43) can take place. The application of the rotation matrix (Eq. (8.44)) leads to

$$\vec{R}_{B, \text{sum}}^{u,v,w} = \vec{R}_B^{u,v,w} + \bar{\mathcal{R}} \vec{R}_A^{u,v,w}. \quad (8.46)$$

The superscript u, v, w in Eq. (8.46) denotes the three momentum equations.

8.9 INTERFACE BETWEEN GRID BLOCKS

During the discussion of the spatial discretization with structured grids in Section 3.1, it became evident that it is usually not possible to generate a single grid inside a

geometrically complex domain (see Fig. 3.4). We mentioned two possible methodologies how to solve the problem. The first one was the multiblock approach and the second one was the Chimera technique. In the following section, we shall describe the basic implementation issues of the multiblock approach. For more throughout discussion, the reader is referred to Refs. [39–45]. A very helpful introduction to the multiblock methodology was presented in Ref. [46]. Details of the Chimera technique, which is not treated here, can be found in Refs. [47–53].

Within the multiblock technique, the physical domain is split into a certain number of virtual parts. Consequently, the computational domain becomes also divided into the same number of blocks. In a general case, the physical solution in a particular block will depend on the flow in one or multiple neighboring blocks. Therefore, we have to provide a data structure which allows for an efficient exchange of information between the blocks. This structure is also required for inter-process communication, if different processors are used to solve the governing equations in the grid blocks.

The first part of the data structure consists of the numbering of the block boundaries. One particular numbering scheme is displayed in Fig. 8.13.

The numbering strategy in Fig. 8.13 can be summarized as follows:

- boundary 1 : $i = IBEG$
- boundary 2 : $i = IEND$
- boundary 3 : $j = JBEG$
- boundary 4 : $j = JEND$
- boundary 5 : $k = KBEG$
- boundary 6 : $k = KEND$.

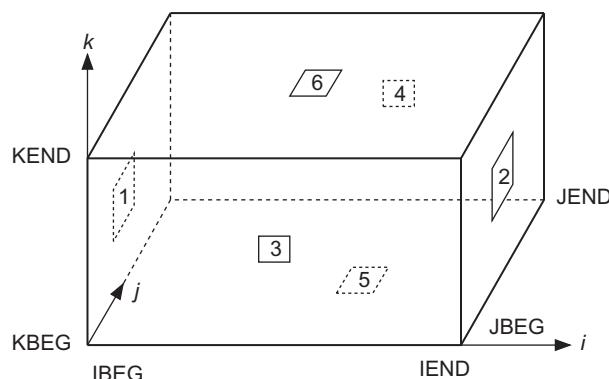


Figure 8.13 Numbering of the sides of the computational space and of the block boundaries.

It is important that all blocks employ the same numbering scheme. The indexes i, j, k of the grid points in the computational space are defined in the ranges

$$\begin{aligned} IBEG \leq i \leq IEND \\ JBEG \leq j \leq JEND \\ KBEG \leq k \leq KEND. \end{aligned}$$

The cell indexes I, J, K , which are required by the cell-centered scheme are defined in a similar way. Since the multiblock approach is usually implemented using dummy cells/points, the physical cells/points will have a certain offset from the start or the end of each range (see Fig. 8.1).

The boundary of each block is divided into a number of non-overlapping patches. This allows the specification of different boundary conditions on the same block boundary. The situation is depicted in Fig. 8.14. For a unique identification of each patch it is necessary to store the number of the corresponding block and the number of the block boundary. Furthermore, the origin, the height, and the width of the patch must be stored. For this purpose, the coordinates $L1BEG$, $L1END$, $L2BEG$, and $L2END$ are used in Fig. 8.14. It is suggested to orient the coordinate system of the patch according to the *cyclic directions*. This means, that if we consider the i -coordinate, j and k will be the first and the second cyclic direction. In the case of the j -coordinate, the cyclic directions will become k and i , respectively. Therefore, since the patch in Fig. 8.14 is on the $j = JBEG$ boundary, the l_1 -coordinate is oriented in the k -direction and l_2 in the i -direction. The application of the cyclic directions allows for a unique definition of the orientation of each patch.

The remaining part of the data structure makes sure that data can be exchanged between those patches, which represent interfaces between the blocks (we assume here that the blocks communicate only across their faces). For this purpose, it is required to

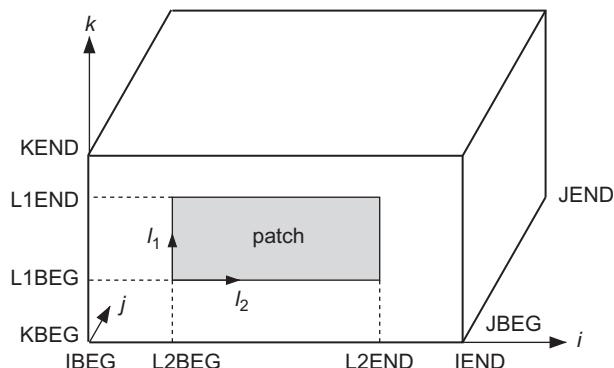


Figure 8.14 Coordinates of a boundary patch in computational space. The patch has its own local coordinate system l_1, l_2 .

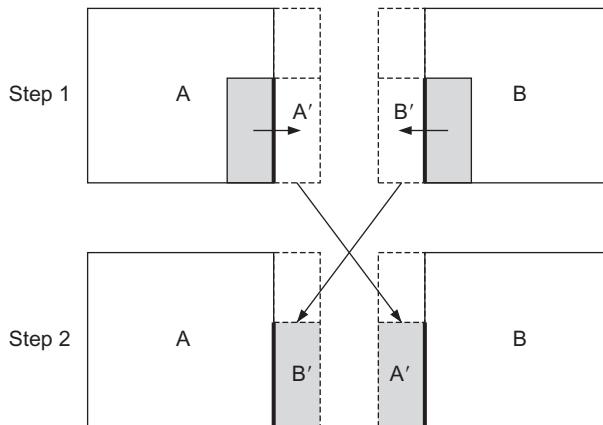


Figure 8.15 Exchange of flow variables (in shaded regions A' , B') between two blocks A and B. Dummy layers are denoted by a dashed line.

extend the above patch data structure by the numbers of the adjacent block and patch. It is furthermore necessary to code the orientation of the communicating patches with respect to each other.

The exchange of flow quantities between two blocks is sketched in Fig. 8.15. The procedure consists of two steps. In the first step, variables from the part of the domain, which is overlapped by the dummy layers of the adjacent patch are written to the own dummy cells/points or to a temporary storage (A' and B' in Fig. 8.15). This is done for all blocks. In the second step, the data in A' and B' is exchanged between both blocks. This means that A' is written to the dummy layers of block B and B' to the dummy layers of block A. If the two patches have a different orientation, the data must be transformed accordingly. In cases where the grid lines do not match at the block interface, further operations are required as described, for example, in Refs. [54–56].

8.10 FLOW GRADIENTS AT BOUNDARIES OF UNSTRUCTURED GRIDS

We already stated in Section 5.3.4 that the evaluation of the flow gradients requires some care in the case of the median-dual scheme. If the gradients are calculated on triangular or tetrahedral grids using the Green-Gauss approach in Eq. (5.50), the contributions from the boundaries of the domain (except at symmetry or periodic boundaries) must be evaluated similarly to Eq. (8.12) or (8.13), instead of arithmetic average. Otherwise, the gradient will not be accurate. Considering the notation in Fig. 8.4, the contribution to the boundary node 1 reads

$$\frac{1}{6}(5U_1 + U_2)\vec{n}_{12}\frac{\Delta S_{12}}{2},$$

where ΔS_{12} is the length of the boundary face between node 1 and 2 (therefore halved). Corresponding to Eq. (8.13), the contribution of the triangular face 1-2-3 to node 1 in Fig. 8.5 becomes

$$\frac{1}{8}(6U_1 + U_2 + U_3) \vec{n}_{123} \frac{\Delta S_{123}}{3}$$

with $\Delta S_{123}/3$ being the gray area in the triangle 1-2-3. On mixed grids, it is more appropriate to employ the least squares approach with virtual edges [8] (see Fig. 5.15).

The cell-centered scheme requires no special provisions at symmetry or periodic boundaries. The implementation is identical to that discussed for the fluxes in Section 8.6 or Section 8.8. This holds also for the median-dual scheme, if the gradients are evaluated using the least-squares approach. The only additional work required is to set certain gradients to zero as described previously in Section 8.6 (cf. Eq. (8.40)).

If the Green-Gauss approach is employed within the median-dual scheme (i.e., if Eq. (5.50) is applied), it is necessary to correct normal vectors of those faces of the control volume, which touch a symmetry boundary (like at point 2* in Fig. 8.4). This is done by setting all components of the face vector to zero, which are normal to the symmetry plane. Finally, the gradients are corrected as discussed in Section 8.6. At periodic boundaries, the gradients and the volumes from both sides of the boundary have to be summed up as presented in Section 8.8 for the fluxes (Eq. (8.43)). In the case of rotational periodicity, the gradients need to be transformed by applying the rotation matrix in Eq. (8.44).

REFERENCES

- [1] Kroll N, Jain RK. Solution of two-dimensional Euler equations—experience with a finite volume code. DFVLR-FB 87-41; 1987.
- [2] Rizzi A. Numerical implementation of solid-body boundary conditions for the Euler equations. ZAMM 1978;58:301-4.
- [3] Hall MG. Cell vertex multigrid scheme for solution of the Euler equations. Proc. Conf. on Numerical Methods for Fluid Dynamics. Reading, UK; 1985.
- [4] Koeck C. Computation of three-dimensional flow using the Euler equations and a multiple-grid scheme. Int J Numer Meth Fluids 1985;5:483-500.
- [5] Frink NT, Parikh P, Pirzadeh S. A fast upwind solver for the Euler equations on three-dimensional unstructured meshes. AIAA Paper 91-0102; 1991.
- [6] Frink NT. Recent progress toward a three-dimensional Navier-Stokes solver. AIAA Paper 94-0061; 1994.
- [7] Luo H, Baum JD, Löhner R. An improved finite volume scheme for compressible flows on unstructured grids. AIAA Paper 95-0348; 1995.
- [8] Haselbacher A, Blazek J. On the accurate and efficient discretisation of the Navier-Stokes equations on mixed grids. AIAA Paper 99-3363; 1999 [also AIAA J 2000;38:2094-102].
- [9] Mazaheri K, Roe PL. Numerical wave propagation and steady-state solutions: soft wall and outer boundary conditions. AIAA J 1997;35:965-75.
- [10] Engquist B, Majda A. Absorbing boundary conditions for numerical simulation of waves. Math Comput 1977;31:629-51.

- [11] Bayliss A, Turkel E. Far field boundary conditions for compressible flow. *J Comput Phys* 1982;48:182-99.
- [12] Whitfield DL, Janus JM. Three-dimensional unsteady Euler equations solution using flux vector splitting. *AIAA Paper 84-1552*; 1984.
- [13] Gustafsson B. Far field boundary conditions for time-dependent hyperbolic systems. Center for Large Scale Sci. Comput., CLaSSiC-87-16. Stanford University; 1987.
- [14] Thompson KW. Time dependent boundary conditions for hyperbolic systems. *J Comput Phys* 1987;68:1-24.
- [15] Hayder ME, Hu FQ, Hussaini MY. Towards perfectly absorbing boundary conditions for Euler equations. *ICASE Report No. 97-25*; 1997.
- [16] Givoli D. Non-reflecting boundary conditions. *J Comput Phys* 1991;94:1-29.
- [17] Kreiss HO. Initial boundary value problems for hyperbolic systems. *Commun Pure Appl Math* 1970;23:277-98.
- [18] Usab WJ, Murman EM. Embedded mesh solution of the Euler equation using a multiple-grid method. *AIAA Paper 83-1946*; 1983.
- [19] Giles MB, Drela M, Thompkins WT. Newton solution of direct and inverse transonic Euler equations. *AIAA Paper 85-1530*; 1985.
- [20] Klunker EB, Harder KC. Notes on linearized subsonic wing theory. Unpublished.
- [21] Radespiel R. A cell-vertex multigrid method for the Navier-Stokes equations. *NASA TM-101557*; 1989.
- [22] Papanastasiou TC, Malamataris N, Ellwood K. A new outflow boundary condition. *Int J Numer Meth Fluids* 1992;14:587-608.
- [23] Sani RL, Gresho PM. Résumé and remarks on the open boundary condition minisymposium. *Int J Numer Meth Fluids* 1994;18:983-1008.
- [24] Baum M, Poinsot T, Thevenin D. Accurate boundary conditions for multicomponent reactive flows. *J Comput Phys* 1994;116:247-61.
- [25] Griffiths DF. The ‘no boundary condition’ outflow boundary condition. *Int J Numer Meth Fluids* 1997;24:393-411.
- [26] Renardy M. Imposing ‘no’ boundary condition at outflow: why does it work? *Int J Numer Meth Fluids* 1997;24:413-17.
- [27] Rudy DH, Strikwerda JC. A nonreflective outflow boundary condition for subsonic Navier-Stokes calculations. *J Comput Phys* 1980;36:55-70.
- [28] Yokota JW, Caughey DA. An L-U implicit multigrid algorithm for the three-dimensional Euler equations. *AIAA Paper 87-0453*; 1987.
- [29] Yokota JW. Diagonally inverted lower-upper factored implicit multigrid scheme for the three-dimensional Navier-Stokes equations. *AIAA J* 1989;28:1642-9.
- [30] Holmes DG. Inviscid 2D solutions on unstructured, adaptive grids. *Numerical Methods for Flows in Turbomachinery*, Von Karman Institute, Rhode-St-Genèse, Belgium: VKI Lecture Series 1989-06, 1989.
- [31] Giles MB. Non-reflecting boundary conditions for Euler equation calculations. *AIAA Paper 89-1942*; 1989.
- [32] Hirsch Ch, Verhoff A. Far field numerical boundary conditions for internal and cascade flow computations. *AIAA Paper 89-1943*; 1989.
- [33] Patankar SV, Liu CH, Sparrow EM. Fully developed flow and heat transfer in ducts having streamwise-periodic variations of cross-sectional area. *ASME J Heat Transfer* 1977;99:180-86.
- [34] Deschamps V. Simulations numériques de la turbulence inhomogène incompressible dans un écoulement de canal plan. ONERA, note technique 1988-5; 1988.
- [35] Mossi M. Simulation of benchmark and industrial unsteady compressible turbulent fluid flows. PhD Thesis, No. 1958, École Polytechnique Fédérale de Lausanne, Département de Génie Mécanique, Switzerland; 1999. p. 136-7.
- [36] Lenormand E, Sagaut P, Phuoc LT, Comte P. Subgrid-scale models for large-eddy simulations of compressible wall bounded flows. *AIAA J* 2000;38:1340-50.

- [37] Chung H-T, Baek J-H. Influence of trailing-edge grid structure on Navier-Stokes computation of turbomachinery cascade flow. *Int J Numer Meth Fluids* 1992;15:883-94.
- [38] Segal G, Vuik K, Kassels K. On the implementation of symmetric and antisymmetric periodic boundary conditions for incompressible flow. *Int J Numer Meth Fluids* 1994;18:1153-1165.
- [39] Rossow C-C. Efficient computation of inviscid flow fields around complex configurations using a multiblock multigrid method. *Commun Appl Numer Meth* 1992;8:735-47.
- [40] Jacquotte OP. Generation, optimization and adaptation of multiblock grids around complex configurations in computational fluid dynamics. *Int J Numer Meth Eng* 1992;34:443-454.
- [41] Szmelter J, Marchant MJ, Evans A, Weatherill NP. Two-dimensional Navier-Stokes equations with adaptivity on structured meshes. *Comput Meth Appl Mech Eng* 1992;101:355-68.
- [42] Marchant MJ, Weatherill NP. The construction of nearly orthogonal multiblock grids for compressible flow simulation. *Commun Numer Meth Eng* 1993;9:567-78.
- [43] Jenssen CB. Implicit multiblock Euler and Navier-Stokes calculations. *AIAA J* 1994;32:1808-14.
- [44] De-Nicola C, Pinto G, Tognaccini R. On the numerical stability of block structured algorithms with applications to 1-D advection-diffusion problems. *Comput Fluids* 1995;24:41-54.
- [45] Enander R, Sterner E. Analysis of internal boundary conditions and communication strategies for multigrid multiblock methods. Report No. 191, Dept. Sci. Computing, Uppsala University, Sweden; 1997.
- [46] Rizzi A, Eliasson P, Lindblad I, Hirsch C, Lacor C, Haeuser J. The engineering of multiblock/multigrid software for Navier-Stokes flows on structured meshes. *Comput Fluids* 1993;22:341-67.
- [47] Benek JA, Buning PG, Steger JL. A 3-D chimera grid embedding technique. *AIAA Paper* 85-1523; 1985.
- [48] Buning PG, Chu IT, Obayashi S, Rizk YM, Steger JL. Numerical simulation of the integrated space shuttle vehicle in ascent. *AIAA Paper* 88-4359-CP; 1988.
- [49] Cheshire G, Henshaw WD. Composite overlapping meshes for the solution of partial differential equations. *J Comput Phys* 1990;90:1-64.
- [50] Pearce DG, Stanley SA, Martin EW, Gomez RJ, Le Beau GJ, Buning PG, et al. Development of a large scale chimera grid system for the space shuttle launch vehicle. *AIAA Paper* 93-0533; 1993.
- [51] Kao H-J, Liou M-S, Chow C-Y. Grid adaptation using chimera composite overlapping meshes. *AIAA J* 1994;32:942-9.
- [52] Henshaw WD, Schwendeman DW. An adaptive numerical scheme for high-speed reactive flow on overlapping grids. *J Comput Phys* 2003;191:420-47.
- [53] Xu L, Weng P. High order accurate and low dissipation method for unsteady compressible viscous flow computation on helicopter rotor in forward flight. *J Comput Phys* 2014;258:470-88.
- [54] Rai MM. A conservative treatment of zonal boundaries for Euler equation calculations. *J Comput Phys* 1986;62:472-503.
- [55] Kassies A, Tognaccini R. Boundary conditions for Euler equations at internal block faces of multi-block domains using local grid refinement. *AIAA Paper* 90-1590; 1990.
- [56] Peng Y-F, Mittal R, Sau A, Hwang RR. Nested Cartesian grid method in incompressible viscous fluid flow. *J Comput Phys* 2010;229:7072-101.

CHAPTER 9

Acceleration Techniques

Contents

9.1 Local Time-Stepping	284
9.2 Enthalpy Damping	284
9.3 Residual Smoothing	286
9.3.1 Central IRS on structured grids	287
9.3.2 Central IRS on unstructured grids	290
9.3.3 Upwind IRS on structured grids	290
9.4 Multigrid	293
9.4.1 Basic multigrid cycle	294
<i>Transfer of the solution and residuals to the coarser grid</i>	294
<i>Computation of a new solution on the coarse grid</i>	295
<i>Solution interpolation from the coarse to the fine grid</i>	295
9.4.2 Multigrid strategies	296
<i>Number of time steps</i>	297
<i>Starting grid</i>	297
<i>Accuracy of transfer operators</i>	297
9.4.3 Implementation on structured grids	298
<i>Transfer operators for the cell-centered scheme</i>	298
<i>Transfer operators for the cell-vertex scheme</i>	300
<i>Upwind prolongation (cell-vertex scheme)</i>	301
9.4.4 Implementation on unstructured grids	303
<i>Nonnested grids</i>	303
<i>Topological methods</i>	304
<i>Agglomeration multigrid method</i>	304
9.5 Preconditioning for Low Mach Numbers	307
9.5.1 Derivation of preconditioned equations	308
9.5.2 Implementation	310
9.5.3 Form of the matrices	312
<i>Transformation matrices</i>	313
<i>Preconditioning matrices</i>	314
<i>Eigenvalues of the preconditioned system</i>	317
<i>Eigenvectors of the preconditioned system</i>	318
9.6 Parallelization	320
9.6.1 MPI	324
9.6.2 OpenMP	326
9.6.3 CUDA	327
9.6.4 OpenCL	329
References	329

Various methodologies were developed in order to accelerate the solution of the governing equations (2.19) for stationary problems. The acceleration techniques are applicable also to the inner iteration of the dual-time-stepping scheme (Section 6.3). The following numerical methods will be discussed in this chapter:

1. local time-stepping,
2. enthalpy damping,
3. residual smoothing,
4. multigrid,
5. low Mach-number preconditioning.

The local time-stepping, enthalpy damping, and the preconditioning are based on a modification of the system of the ordinary differential equations (6.1), while the two remaining techniques are improvements of the solution process. With the exception of the residual smoothing, all methods can be applied to both the explicit (Section 6.1) and the implicit (Section 6.2) time-stepping schemes. Residual smoothing was developed especially for the explicit multistage schemes (Sections 6.1.1 and 6.1.2). An overview of several acceleration methodologies can be found in Refs. [1, 2].

One further technique presented in this chapter is parallelization. Although being concerned primarily with the design of the computer code, parallelization has an effect on the numerical schemes being used. Given the current trends in computer hardware, it is an important additional acceleration technique, and we shall discuss several possible approaches.

9.1 LOCAL TIME-STEPPING

In this case, the discretized governing equations (6.1) are integrated using the largest possible time step for each control volume. The local time step Δt_l is calculated according to one of the formulas (6.14), (6.18), (6.20), or (6.22). As a result, the convergence to the steady state is considerably accelerated; however, the transient solution is no longer temporally accurate. The efficiency of the local time-stepping is demonstrated in Fig. 9.1 for an inviscid subsonic flow past a wing. The flow is driven to the steady state by an explicit multistage scheme (Section 6.1.1). It is apparent that using a global time step (i.e., a time step identical for all control volumes) results in an unnecessarily slow convergence toward the steady state. Even larger savings in terms of the CPU time are usually achieved for stationary viscous flows.

9.2 ENTHALPY DAMPING

In certain cases, the total enthalpy H (Eq. (2.12)) is constant in the whole flow field. This situation occurs, for example, for external flows governed by the Euler equations in absence of heat sources and external forces. We can take advantage of this fact in order to reduce the computational effort.

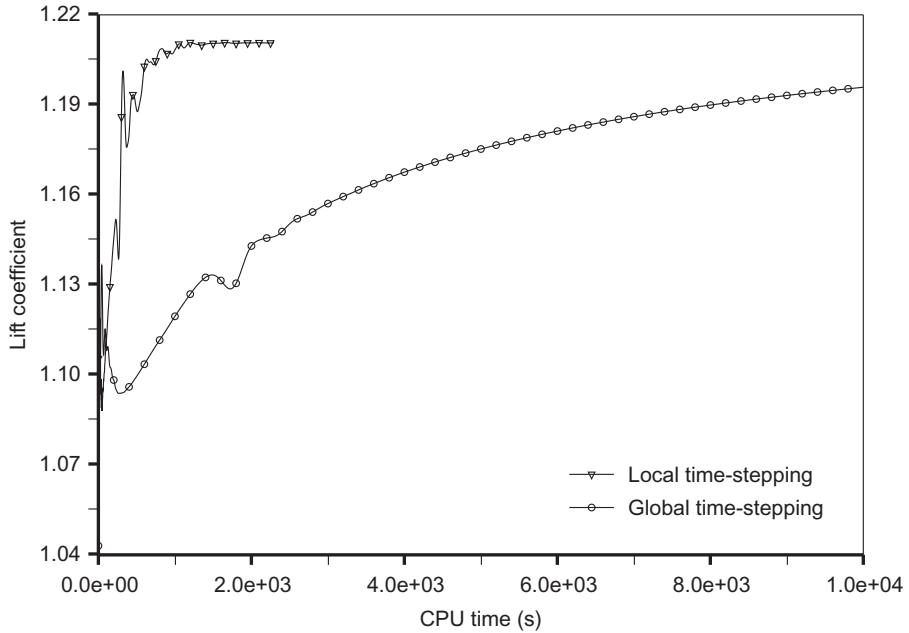


Figure 9.1 Comparison of convergence histories of the lift coefficient with and without local time-stepping for an inviscid subsonic flow on an unstructured grid.

The first possibility would be to prescribe the value of the total enthalpy in the flow domain. In consequence, the energy equation can be omitted from the Euler equations (2.44). This saves memory and CPU time.

A different methodology—the *enthalpy damping*—was suggested by Jameson for the solution of the potential flow equation [3]. It employs the difference between the total enthalpy H and its free-stream value H_∞ in order to define an additional forcing term. With this, the convergence to the steady state can be considerably accelerated. The application of the enthalpy damping to the Euler equations (2.44) results in the following modification [4]:

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega + \oint_{\partial\Omega} \vec{F}_c dS = \int_{\Omega} (\vec{Q} - \vec{Q}_{ED}) d\Omega, \quad (9.1)$$

where the forcing term \vec{Q}_{ED} is given by

$$\vec{Q}_{ED} = \vartheta \begin{bmatrix} \rho (H - H_\infty) \\ \rho u (H - H_\infty) \\ \rho v (H - H_\infty) \\ \rho w (H - H_\infty) \\ \rho (H - H_\infty) \end{bmatrix}. \quad (9.2)$$

The damping factor ϑ is a small constant, which has to be determined empirically. The spatial discretization scheme, and particularly the artificial dissipation has to be implemented in such a way that $H = H_\infty$ is a valid solution of the discretized equations. Then, the addition of the source term \vec{Q}_{ED} does not alter the final steady state.

The enthalpy damping is conducted as an additional step after each update of the flow solution. For example, in the case of the explicit m -stage time-stepping scheme (Section 6.1.1 or 6.1.2), the damping step reads

$$\vec{W}_I^{n+1} = \frac{1}{1 + \vartheta (H_I^{(m)} - H_\infty)} \vec{W}_I^{(m)} \quad (9.3)$$

with the exception of the energy equation which becomes

$$(\rho E)_I^{n+1} = \frac{1}{1 + \vartheta} \left[(\rho E)_I^{(m)} - \vartheta p_I^{(m)} \right]. \quad (9.4)$$

In Eq. (9.3), $\vec{W}_I^{(m)}$ denotes the final solution of the m -stage explicit time-stepping scheme. Numerical experiments in Ref. [1], which were conducted for a transonic flow past the NACA 0012 airfoil ($M_\infty = 0.8$, $\alpha = 1.25^\circ$), confirmed that the number of time steps to reach the steady state can be reduced by about factor of two.

9.3 RESIDUAL SMOOTHING

The maximum CFL number and the convergence properties of the explicit multi-stage time-stepping scheme (Sections 6.1.1 and 6.1.2) can be—to some extent— influenced by optimizing the stage coefficients [5–7]. Jameson and Baker [8] introduced the *residual smoothing* technique with the aim to lend the explicit scheme and implicit character and hence to increase the maximum allowable CFL number substantially. A further purpose of the residual smoothing is a better damping of the high-frequency error components of the residual. This is of particular importance for a successful application of the multigrid method.

The residual smoothing can be implemented in explicit, implicit, or in mixed manner [9, 10], respectively. The residual smoothing is usually applied in each stage of the explicit time-stepping scheme (Eqs. (6.5) and (6.7)). The previously computed residuals $\vec{R}^{(k)}$ are replaced by the smoothed residuals \vec{R}^* before the solution $\vec{W}^{(k)}$ is updated. In the following, we shall discuss the implementation of the popular implicit residual smoothing (IRS) on structured as well as on unstructured grids.

9.3.1 Central IRS on structured grids

The standard formulation of the IRS reads in 3D

$$\begin{aligned} -\epsilon^I \vec{R}_{I-1,J,K}^* + (1 + 2\epsilon^I) \vec{R}_{I,J,K}^* - \epsilon^I \vec{R}_{I+1,J,K}^* &= \vec{R}_{I,J,K}, \\ -\epsilon^J \vec{R}_{I,J-1,K}^{**} + (1 + 2\epsilon^J) \vec{R}_{I,J,K}^{**} - \epsilon^J \vec{R}_{I,J+1,K}^{**} &= \vec{R}_{I,J,K}^*, \\ -\epsilon^K \vec{R}_{I,J,K-1}^{***} + (1 + 2\epsilon^K) \vec{R}_{I,J,K}^{***} - \epsilon^K \vec{R}_{I,J,K+1}^{***} &= \vec{R}_{I,J,K}^{**}, \end{aligned} \quad (9.5)$$

where $\vec{R}_{I,J,K}^*$, $\vec{R}_{I,J,K}^{**}$, and $\vec{R}_{I,J,K}^{***}$ denote the smoothed residuals in I -, J -, and K -direction, respectively. The parameters ϵ^I , ϵ^J , and ϵ^K stand for the smoothing coefficients in the three computational coordinates. The implicit operator in Eq. (9.5) resembles second-order central difference. The term central implicit residual smoothing (CIRS) is therefore used. The implicit system in Eq. (9.5) is solved by the Thomas algorithm for the inversion of tridiagonal matrices.

The smoothing coefficients are usually defined as functions of spectral radii of the convective flux Jacobians [11]. The purpose is to apply only as much smoothing in each coordinate direction as it is necessary for stability and good error damping. A suitable formula for 2D was suggested in Ref. [12]

$$\begin{aligned} \epsilon^I &= \max \left\{ \frac{1}{4} \left[\left(\frac{\sigma^*}{\sigma} \frac{1}{1 + \Psi r} \right)^2 - 1 \right], 0 \right\}, \\ \epsilon^J &= \max \left\{ \frac{1}{4} \left[\left(\frac{\sigma^*}{\sigma} \frac{1}{1 + \Psi/r} \right)^2 - 1 \right], 0 \right\}, \end{aligned} \quad (9.6)$$

where σ^*/σ denotes the ratio of the CFL numbers of the smoothed and nonsmoothed scheme. The variable r stands for the ratio of the convective spectral radii (Eq. (4.53)), that is, $r = \hat{\Lambda}_c^J / \hat{\Lambda}_c^I$. The parameter $\Psi \approx 0.125$ ensures linear stability of the smoothing operation.

In 3D, the smoothing coefficients can be evaluated using an expression similar to Eq. (9.6) [12]

$$\begin{aligned} \epsilon^I &= \max \left\{ \frac{1}{4} \left[\left(\frac{\sigma^*}{\sigma} \frac{1}{1 + \Psi(r^H + r^{KI})} \right)^2 - 1 \right], 0 \right\} \\ \epsilon^J &= \max \left\{ \frac{1}{4} \left[\left(\frac{\sigma^*}{\sigma} \frac{1}{1 + \Psi(r^{KJ} + r^I)} \right)^2 - 1 \right], 0 \right\} \\ \epsilon^K &= \max \left\{ \frac{1}{4} \left[\left(\frac{\sigma^*}{\sigma} \frac{1}{1 + \Psi(r^{IK} + r^{JK})} \right)^2 - 1 \right], 0 \right\}, \end{aligned} \quad (9.7)$$

where

$$r^{II} = \hat{\Lambda}_c^J / \hat{\Lambda}_c^I, \quad r^{KI} = \hat{\Lambda}_c^K / \hat{\Lambda}_c^I,$$

etc. The typical value of the parameter Ψ is 0.0625.

The maximum of the ratio of the CFL numbers σ^*/σ depends on the value of the smoothing coefficient and on the type of the spatial discretization scheme. In the case of the central scheme (Section 4.3.1), the ratio is given by

$$\frac{\sigma^*}{\sigma} \leq \sqrt{1 + 4\epsilon}. \quad (9.8)$$

In practice, value of $\sigma^*/\sigma \approx 2$ can be reached ($\epsilon = 0.8$). Higher ratios reduce the damping of the time-stepping scheme. There is no such simple condition like (9.8) for the upwind spatial discretization since the maximum of σ^*/σ depends also on the stage coefficients. But normally, the CFL number (see Tables 6.1 and 6.2) can be raised by about factor two for inviscid and up to a factor of five for viscous flows (using the hybrid scheme from Table 6.2). Figures 9.2 and 9.3 demonstrate the effect of the central IRS on the convergence for an inviscid and viscous flow on structured grids. The comparison is done in terms of the CPU time, in order to account for the additional numerical effort due to the IRS. As we can see, the solution time can be significantly reduced especially

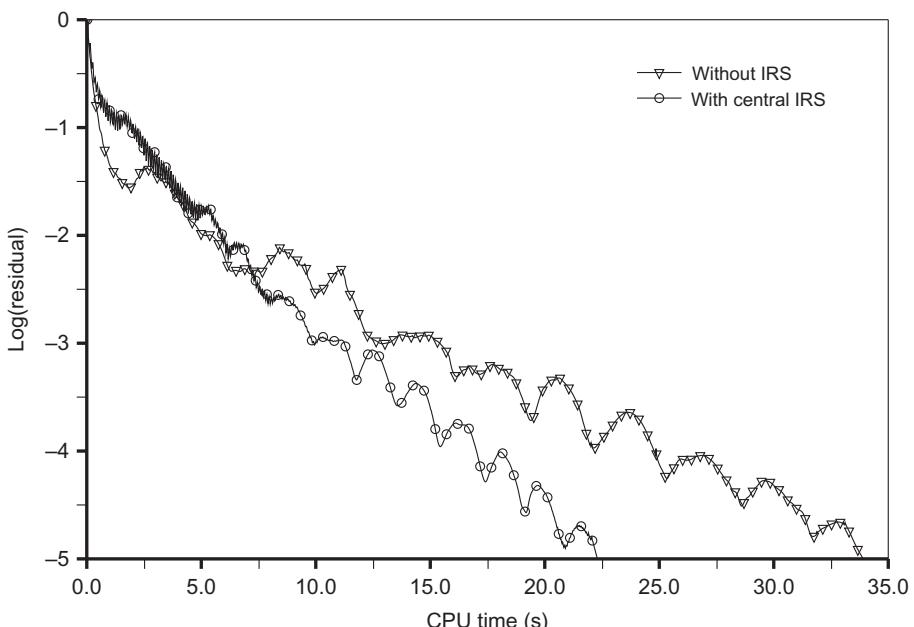


Figure 9.2 Comparison of convergence histories with and without IRS for an inviscid transonic flow on structured grid.

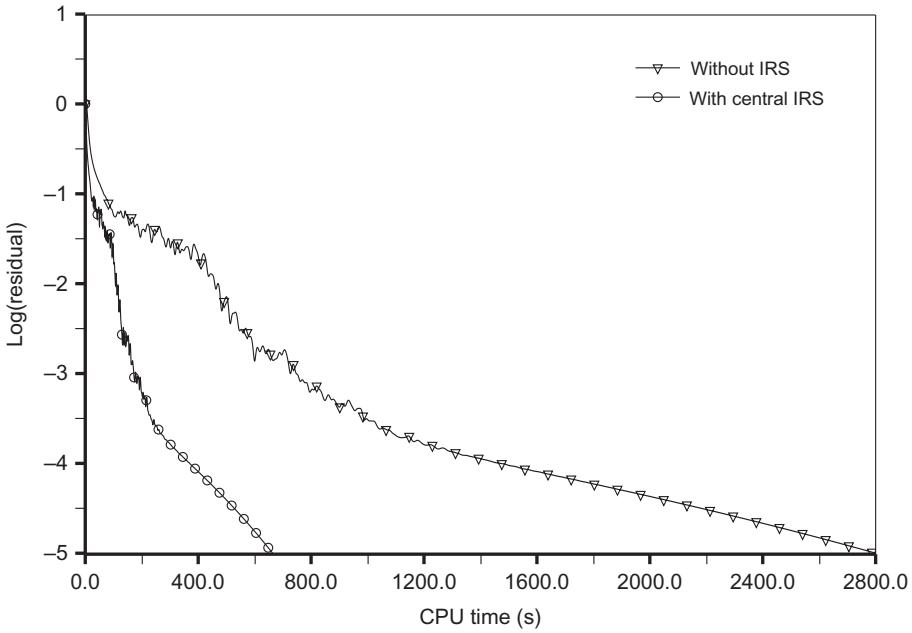


Figure 9.3 Comparison of convergence histories with and without IRS for a viscous subsonic flow on structured grid.

for a viscous flow. Even larger savings can be realized when the IRS is employed together with multigrid (Section 9.4).

The limitation of the time step due to the viscous spectral radius $\hat{\Lambda}_\nu$ in Eq. (6.18) can be offset with the aid of the IRS. The maximum time step is calculated according to the formula (6.14) without $\hat{\Lambda}_\nu$. In flow regions where the viscous spectral radius dominates, smoothing is carried out using higher coefficients ϵ_v^I , ϵ_v^J , ϵ_v^K . Smoothing coefficients based on the viscous spectral radii can be computed from the formulas [12, 13]:

$$\begin{aligned}\epsilon_v^I &= C \left(\frac{\sigma^*}{\sigma} \right) \frac{\hat{\Lambda}_\nu^I}{\hat{\Lambda}_c^I + \hat{\Lambda}_c^J + \hat{\Lambda}_c^K}, \\ \epsilon_v^J &= C \left(\frac{\sigma^*}{\sigma} \right) \frac{\hat{\Lambda}_\nu^J}{\hat{\Lambda}_c^I + \hat{\Lambda}_c^J + \hat{\Lambda}_c^K}, \\ \epsilon_v^K &= C \left(\frac{\sigma^*}{\sigma} \right) \frac{\hat{\Lambda}_\nu^K}{\hat{\Lambda}_c^I + \hat{\Lambda}_c^J + \hat{\Lambda}_c^K},\end{aligned}\quad (9.9)$$

where the constant $C \approx 5/4$. The maximum of the coefficients from Eqs. (9.7) to (9.9) is then taken as the resulting smoothing coefficient.

9.3.2 Central IRS on unstructured grids

CIRS is implemented on unstructured grids by applying the Laplacian operator (see Eq. (5.24)) to the residual. Thus, the smoothed residual \vec{R}_I^* in a control volume I is obtained from the implicit relation [14]

$$\vec{R}_I^* + \sum_{J=1}^{N_A} \epsilon \left(\vec{R}_I^* - \vec{R}_J^* \right) = \vec{R}_I. \quad (9.10)$$

The sum includes all N_A adjacent control volumes. The relation (9.10) is solved for \vec{R}_I^* using Jacobi iteration. Useful values of the smoothing coefficient are $0.3 \leq \epsilon \leq 0.8$. With these ϵ values it is possible to increase the CFL number (and hence the time step) two to five times. Due to the diagonal dominance of the matrix, the Jacobi iteration converges in just about two steps.

9.3.3 Upwind IRS on structured grids

The previously discussed CIRS method works satisfactorily for subsonic and transonic flows. It is also helpful in viscous dominated regions. However, CIRS exhibits poor error-damping characteristics in conjunction with upwind spatial discretization schemes. Furthermore, the robustness of a multistage scheme accelerated by CIRS suffers in the case of strong shocks. Therefore, the so-called upwind implicit residual smoothing (UIRS) method was developed [15, 16], which is particularly suited for high Mach-number flows.

By contrast to CIRS, the UIRS methodology takes the sign of the convective eigenvalues $\bar{\Lambda}_c$ into account. The idea is to smooth the residuals only in the direction of the characteristic of the Euler equations (i.e., $dx/dt = \text{const.} = \Lambda_c$). This approach prevents unphysical influences on the upstream residuals. The UIRS method requires transformation of the residuals into the characteristic variables (cf. Section A.11). In this way, each component of the residual vector can be smoothed independently according to the sign of the corresponding eigenvalue. For the l th component of the residual, the implicit operator is defined in 1D as [15, 16]

$$\begin{aligned} -\epsilon^I (R^*)_{I-1}^l + (1 + \epsilon^I) (R^*)_I^l &= (R^c)_I^l && \text{if } \Lambda_c^l \geq 0, \\ (1 + \epsilon^I) (R^*)_I^l - \epsilon^I (R^*)_{I+1}^l &= (R^c)_I^l && \text{if } \Lambda_c^l < 0, \end{aligned} \quad (9.11)$$

where \vec{R}^c denotes the residual transformed into characteristic variables, that is,

$$\vec{R}^c = \bar{T}^{-1} \vec{R}. \quad (9.12)$$

The smoothed residuals \vec{R}^* are obtained by the solution of a tridiagonal (bidiagonal if Λ_c^l does not change its sign) equation system using the Thomas algorithm. Afterward, the residuals \vec{R}^* are transformed back into the conservative variables and the solution can be updated.

The most desirable feature of UIRS is that it leads to very favorable damping properties of the multistage scheme, particularly in connection with an upwind spatial discretization. The ability to damp solution errors remains or even improves for high smoothing coefficients. It was demonstrated for 1-D Euler equations that parameter values like $\epsilon = 500$ and $\sigma^* = 1000$ result in a stable and very fast explicit multistage scheme [15, 16] (see also in analysis/mstage in the source codes). However, the problem is the implementation of UIRS in multiple dimensions. Since the convective flux Jacobians cannot be diagonalized simultaneously in all coordinate directions (see Section A.11), the transformation Eq. (9.12) and the smoothing Eq. (9.11) have to be carried out separately for each computational coordinate. The effect of the coordinate splitting is a reduced maximum smoothing coefficient to $2 \leq \epsilon \leq 6$. Despite this, the convergence to the steady state is strongly accelerated as compared to CIRS [16, 17]. The largest improvements in terms of convergence speed and robustness occur in combination with multigrid [16, 18].

In multiple dimensions, the smoothing coefficients are scaled by the eigenvalues. For example, in 2D the following formula can be employed

$$\begin{aligned}\epsilon^I &= \epsilon \cdot \min \left[\frac{\Lambda_c^I}{\Lambda_c^J}, 1 \right], \\ \epsilon^J &= \epsilon \cdot \min \left[\frac{\Lambda_c^J}{\Lambda_c^I}, 1 \right].\end{aligned}\quad (9.13)$$

The relation between the CFL number of the smoothed scheme and the coefficient ϵ reads

$$\frac{\sigma^*}{\sigma} \leq 1 + C\epsilon. \quad (9.14)$$

The constant C depends on the kind of the spatial discretization. In the case of the central scheme $C = 1$. For the first- or second-order upwind scheme the value is $C = 2$.

In order to circumvent the numerical effort of the transformation to the characteristic variables, a simplified version of the UIRS method was suggested [16–18]. Written in the I -direction it reads

$$\begin{aligned}-\epsilon^I \vec{R}_{I-1}^* + (1 + \epsilon^I) \vec{R}_I^* &= \vec{R}_I && \text{if } M^I > 1, \\ -\epsilon^I \vec{R}_{I-1}^* + (1 + 2\epsilon^I) \vec{R}_I^* - \epsilon^I \vec{R}_{I+1}^* &= \vec{R}_I && \text{if } |M^I| \leq 1, \\ (1 + \epsilon^I) \vec{R}_I^* - \epsilon^I \vec{R}_{I+1}^* &= \vec{R}_I && \text{if } M^I < -1.\end{aligned}\quad (9.15)$$

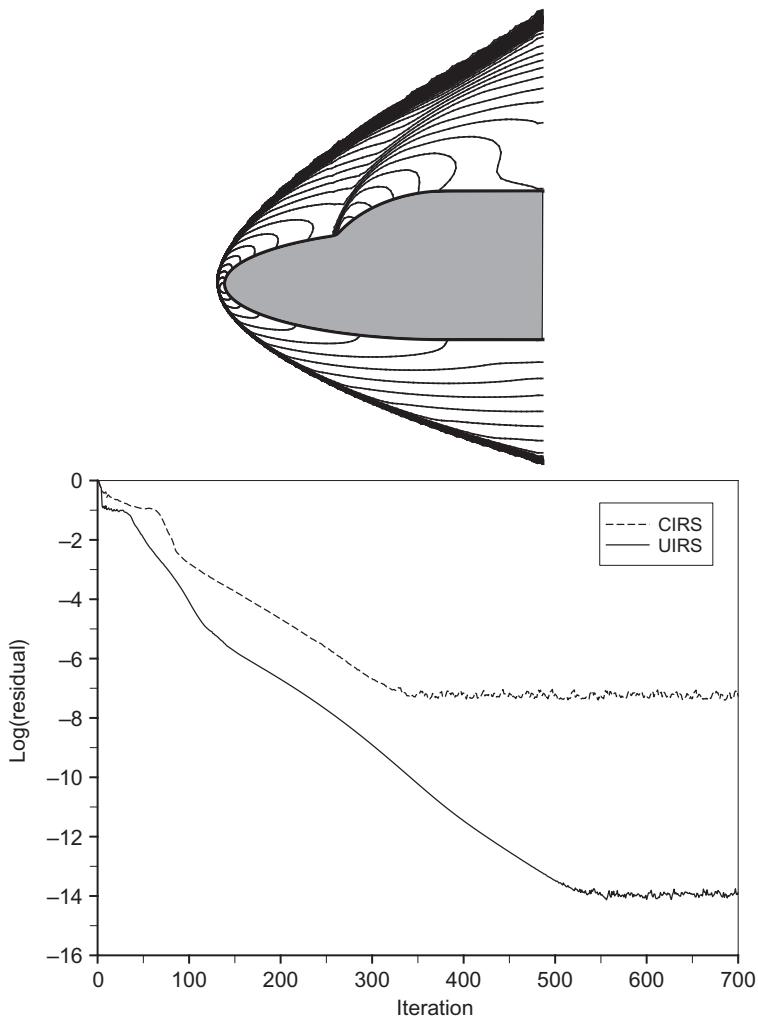


Figure 9.4 Comparison of multigrid convergence between CIRS and UIRS for hypersonic flow past a generic blunt body at $M_{\infty} = 20$, $\alpha = 10^\circ$.

Mach number M is based on velocity projected into the direction of the particular computational coordinate (here: I). Due to the low operation count, the simplified UIRS is suitable especially for 3-D flow problems. In Ref. [16], it was demonstrated that the CPU time needed to reach the steady state can be halved as compared to CIRS. Another example is displayed in Fig. 9.4. As it can be seen, UIRS helps to stabilize the multigrid scheme and also to accelerate its convergence. The scheme with CIRS fails to obtain fully converged solution, and without upwind prolongation it would not work at all.

9.4 MULTIGRID

The multigrid methodology is a very powerful acceleration technique (cf. Fig. 3.7). It is based on the solution of the governing equations on a series of successively coarser grids. The solution updates from the coarse grid are then combined and added to the solution on the finest grid. The technique was originally developed by Brandt [19] for elliptic partial differential equations and later applied to the Euler equations by Jameson [20–22]. After that, the multigrid scheme was employed to solve the Navier-Stokes equations [11–13, 23–32]. The multigrid method can be implemented for both the explicit and the implicit time-stepping schemes [33–39]. The goal of the current research is a significant improvement of the efficiency of multigrid methods for hyperbolic flow problems [40–43].

The basic idea of the multigrid scheme is to employ coarse grids in order to drive the solution on the finest grid faster to steady-state. Two effects are utilized for this purpose:

1. Larger time steps can be employed on the coarser grids (owing to larger control volumes) in conjunction with a reduced numerical effort. Since the work for determining a new solution is distributed mainly over the coarser grids, a more rapid convergence and a reduction of the computing time results.
2. The majority of the explicit and implicit time-stepping and iterative schemes reduces efficiently mainly the high-frequency components of the solution error (see Section 10.3). The low-frequency components are usually only hardly damped. This results in a slow convergence to the steady state, after the initial phase (where the largest errors are eliminated) is over. The multigrid scheme helps at this point—the low-frequency components on the finest grid become high-frequency components on the coarser grids and hence are successively damped. As a result, the error is quickly reduced over the entire spectrum, and the convergence is significantly accelerated.

Thus, as we can follow, the success of the multigrid scheme depends heavily on good damping of the high-frequency error components by the time-stepping or the iterative scheme.

An alternative to the geometrical (grid-based) multigrid is offered by the algebraic multigrid (AMG) method [44–53]. The AMG technique was developed specifically for implicit schemes, where it operates directly on the system matrix (the left-hand side operator). The basic idea of AMG is to apply a coarsening matrix in order to reduce the dimension of the implicit operator and hence the number of equations. The reduced system, which represents a coarse level, is then solved to obtain the correction of the fine-level solution. The coarsening matrix is constructed such that the equations with the strongest coupling (i.e., the largest off-diagonals in the system matrix) are added together. Thus, the generation of coarse levels is governed solely by the physics of

the flow problem and not by the grid. Therefore, the advantage of AMG is that no coarse grid topology has to be constructed or stored, which is particularly beneficial for complex unstructured grids.

9.4.1 Basic multigrid cycle

Before the geometric multigrid scheme can be applied, the coarser grids have to be generated. The standard way is to coarsen the grid evenly in all coordinate directions. However, Mulder [54] proposed an approach called *semi-coarsening*. Here, the grid is coarsened only in one direction, which is changed from one coarse level to another. The semi-coarsening methodology is especially suited for flow problems, where the governing equations are stiff in one spatial direction. An example is the direction normal to the wall in boundary layers. Applications of semi-coarsening to the Navier-Stokes equations were reported, for example, in Refs. [26, 55–57].

The discretized governing equations on the fine grid read in accordance with the relationship (6.1)

$$\frac{d}{dt} \vec{W}_h = -\frac{1}{\Omega_h} \vec{R}_h. \quad (9.16)$$

In the following sections, the finest grid will be denoted by subscript h in reference to the spacing of the grid lines (characteristic dimension of the control volume on unstructured grids). Starting from a known solution \vec{W}_h^n , a new solution \vec{W}_h^{n+1} is obtained after one time step with some suitable iterative scheme. A new residual \vec{R}_h^{n+1} is evaluated with this solution. In order to improve the solution \vec{W}_h^{n+1} using a coarse grid, the following three steps are carried out.

Transfer of the solution and residuals to the coarser grid

The solution is transferred to the coarse grid by means of the interpolation

$$\vec{W}_{2h}^{(0)} = \hat{I}_h^{2h} \vec{W}_h^{n+1}, \quad (9.17)$$

where the subscript $2h$ denotes the coarse grid¹ and \hat{I}_h^{2h} is the *interpolation operator*. The residuals have to be transferred to the coarse grid as well, so that their low-frequency error components can be smoothed. A conservative transfer operator is employed for this purpose. This means that when the control volume size increases, the value of the residual must increase by the same amount. The residuals of the fine grid are also required in order to retain the solution accuracy of the fine grid on the coarse grid. For this purpose,

¹ The notation $2h$ must not be understood in a strictly geometric sense. On unstructured grids, the ratio of the characteristic dimensions of the control volumes on the fine and the coarse grid will usually differ from two. The same holds also for semi-coarsening.

a source term, the so-called *forcing function* [19, 22], is formed as the difference between the residual transferred from the fine grid and the residual computed using the initial solution $\vec{W}_{2h}^{(0)}$ (Eq. (9.17)) on the coarse grid, i.e.,

$$(\vec{Q}_F)_{2h} = I_h^{2h} \vec{R}_h^{n+1} - \vec{R}_{2h}^{(0)}. \quad (9.18)$$

Here, I_h^{2h} represents the *restriction operator* which transfers residuals from the fine to the coarse grid. This type of multigrid scheme is known as the full approximation storage (FAS) method [19]. The FAS method is particularly suited for nonlinear equations because the nonlinearities in the system are carried down to the coarse levels through the re-discretization.

Computation of a new solution on the coarse grid

The solution is evaluated on the coarse grid in the same way as on the fine grid. The forcing function in Eq. (9.18) is added to the residual of the coarse grid, that is,

$$(\vec{R}_F)_{2h} = \vec{R}_{2h} + (\vec{Q}_F)_{2h}. \quad (9.19)$$

Hence, the time-stepping scheme in Eq. (6.1) can be written in the form

$$\frac{d}{dt} \vec{W}_{2h} = -\frac{1}{\Omega_{2h}} (\vec{R}_F)_{2h} = -\frac{1}{\Omega_{2h}} [\vec{R}_{2h} + (\vec{Q}_F)_{2h}]. \quad (9.20)$$

In the case of the explicit multistage scheme (Section 6.1.1), this results in

$$\vec{W}_{2h}^{(k)} = \vec{W}_{2h}^{(0)} - \alpha_k \frac{\Delta t_{2h}}{\Omega_{2h}} [\vec{R}_{2h}^{(k-1)} + (\vec{Q}_F)_{2h}], \quad k = 1, \dots, m \quad (9.21)$$

$$\vec{W}_{2h}^{n+1} = \vec{W}_{2h}^{(m)}$$

in accordance with Eq. (6.5). It has to be noted that during the first iteration (stage in Eq. (9.21)), $(\vec{R}_F)_{2h}$ is identical to the residual transferred from the fine grid (i.e., $(\vec{R}_F)_{2h} = I_h^{2h} \vec{R}_h^{n+1}$ from Eq. (9.18)). This guarantees that the solution on the coarse grid depends on the residual of the fine grid and thus retains the accuracy of the fine grid.

An important question is the accuracy of the spatial discretization scheme on coarse grids. Since the coarse grids do not influence the accuracy of the fine-grid solution, first-order schemes are fully sufficient. The advantages of first-order accurate discretization on coarse grids are the increased robustness, better damping properties, and lower numerical effort in comparison to higher-order schemes.

Solution interpolation from the coarse to the fine grid

After one or several time steps (iterations) were carried out on the coarse grid, the correction with respect to the initial—interpolated—solution (Eq. (9.17)) is computed. This so-called *coarse grid correction* is given by

$$\delta \vec{W}_{2h} = \vec{W}_{2h}^{n+1} - \vec{W}_{2h}^{(0)}. \quad (9.22)$$

The coarse-grid correction is interpolated to the fine grid in order to improve the solution there. Hence, the new solution on the fine grid reads

$$\vec{W}_h^+ = \vec{W}_h^{n+1} + I_{2h}^h \delta \vec{W}_{2h}, \quad (9.23)$$

where I_{2h}^h is denoted as the *prolongation operator*.

9.4.2 Multigrid strategies

The basic multigrid scheme described above consists of one coarse grid only. If multiple coarse grids are present, steps 1 and 2 are repeated until the coarsest grid is reached. It is important to realize that the forcing function on the coarse grids is formed from the restricted *corrected residual* of Eq. (9.19)). For example, on the coarse grid $4h$, the forcing function is obtained from

$$(\vec{Q}_F)_{4h} = I_{2h}^{4h} (\vec{R}_F)_{2h}^{n+1} - \vec{R}_{4h}^{(0)} = I_{2h}^{4h} \left[\vec{R}_{2h}^{n+1} + (\vec{Q}_F)_{2h} \right] - \vec{R}_{4h}^{(0)}. \quad (9.24)$$

In this way, the residual of the finest grid controls the accuracy of the solution on all coarse grids. After a given number of time steps on the coarsest grid, step 3 can be repeated successively until the finest grid is reached again. This procedure is known as a saw-tooth or V-cycle (see Fig. 9.5a). However, it is also possible to conduct more cycles on the coarse grids. This strategy, termed the W-cycle, is displayed in Fig. 9.5b. It is employed particularly frequently for transonic flows. In the case of supersonic and hypersonic flows, the V-cycle proved to be more efficient.

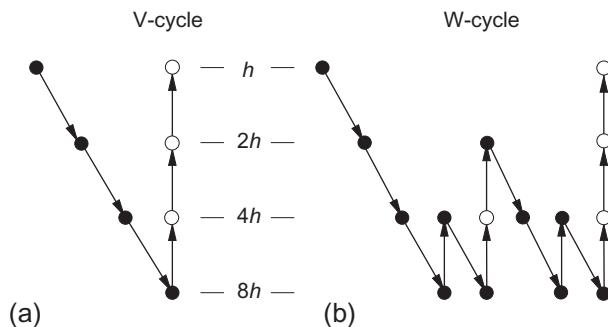


Figure 9.5 Types of multigrid cycles. • denotes time steps before restriction; ○ represents time steps after prolongation.

Number of time steps

The optimum number of time steps before the restriction and after the prolongation depends on the type of the time-stepping scheme. In the case of the explicit multistage scheme (Section 6.1.1 or 6.1.2), it is common to carry out only one time step before the restriction of residuals and no time step after the prolongation. However, the robustness of the multigrid scheme can be improved by smoothing the coarse grid corrections (Eq. (9.22)) before adding them to the fine grid solution \tilde{W}_h^{n+1} (Eq. (9.23)). The same central implicit smoothing (with constant coefficients) as described in Section 9.3 is utilized.

The other popular time-stepping method, the implicit LU-SGS scheme (see Section 6.2.4), requires two iterations before the restriction for the best multigrid efficiency [35]. The number of time steps after the prolongation depends on the spatial discretization. In the case of the central scheme (Section 4.3.1), no time step is necessary [35–37], but the solution correction can be smoothed. On the contrary, one time step should be carried out after the prolongation if an upwind spatial discretization is used. This (2,1)-strategy proved to be an optimum with respect to robustness and computing time for various flow conditions [36, 37].

Starting grid

It should be pointed out that in practice the multigrid scheme is not started directly from the finest grid. Instead, several multigrid cycles are executed from one of the coarse grids. The approximate solution is interpolated to the next finer grid (using the same operator as for the prolongation), few more cycles are performed, the solution is again interpolated to the next finer grid and so on, until the finest grid is reached. In this way, a good starting solution is obtained on the finest grid with only a moderate numerical effort. This very efficient procedure is termed the full multigrid (FMG) method [19].

Accuracy of transfer operators

The restriction (9.18) and the prolongation (9.23) operator must fulfill certain accuracy requirements, namely [58]

$$m_R + m_P > m_E, \quad (9.25)$$

where m_R and m_P denote the degree plus 1 of the polynomial, which is exactly interpolated by the restriction and the prolongation operator, respectively. For example, m_R or m_P are equal to two in the case of linear interpolation. Furthermore, m_E represents the order of the governing equations. Thus, $m_E = 1$ for the Euler equations, and $m_E = 2$ in the case of the Navier-Stokes equations. If the condition (9.25) is violated, the additional errors introduced by the restriction and/or prolongation will disturb the fine-grid solution. Hence, such multigrid scheme will converge only slowly or it will even diverge.

9.4.3 Implementation on structured grids

The implementation of multigrid on structured grids is straightforward since the coarse grids can be easily generated by deleting every second grid line in the respective coordinate direction. The spacing of the grid lines is therefore $2h$, $4h$, etc. This guarantees that the numerical effort on the coarse grids stays low as compared to the finest grid. Several representative examples are provided in Refs. [11–13, 20–22, 26, 33–37].

As we can conclude from Fig. 9.6, the operators for the solution interpolation, the restriction of residuals, and the prolongation of corrections have to be defined differently for cell-centered and node-centered (cell-vertex) schemes. For example, two successive grids have every second grid point in common. On the contrary, the cell centers are always at different locations. Therefore, we shall discuss the standard forms of the transfer operators separately for the cell-centered and node-centered (identical to cell-vertex) finite-volume schemes.

Apart from the symmetrical, purely geometrically defined restriction and prolongation operators, which will be presented next, upwind-biased forms were suggested in Chapter 4 of Ref. [16]. Upwind restriction and prolongation, which account both for the characteristics of the flow equations, improve the robustness of the multigrid scheme for hypersonic flows. In order to save space, we shall discuss the implementation of an upwind prolongation operator for the node-centered spatial discretization only.

Transfer operators for the cell-centered scheme

The solution is transferred from the fine to the coarse grid by using a volume weighted interpolation. In 2D, Eq. (9.17) becomes (see Fig. 9.7a)

$$\begin{aligned} (\vec{W}_{2h}^{(0)})_{IJ} = & \frac{(\vec{W}_h^{n+1})_{IJ}\Omega_{IJ} + (\vec{W}_h^{n+1})_{I+1,J}\Omega_{I+1,J} + (\vec{W}_h^{n+1})_{I,J+1}\Omega_{I,J+1}}{\Omega_{IJ} + \Omega_{I+1,J} + \Omega_{I,J+1} + \Omega_{I+1,J+1}} \\ & + \frac{(\vec{W}_h^{n+1})_{I+1,J+1}\Omega_{I+1,J+1}}{\Omega_{IJ} + \Omega_{I+1,J} + \Omega_{I,J+1} + \Omega_{I+1,J+1}}. \end{aligned} \quad (9.26)$$

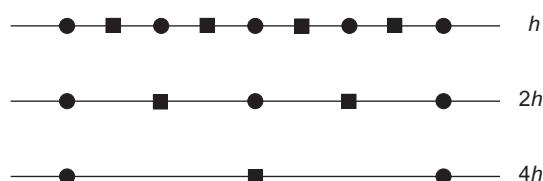


Figure 9.6 Representation of a fine (h) and of two coarse ($2h$, $4h$) 1-D grids. Circles denote grid points, rectangles represent cell centers.

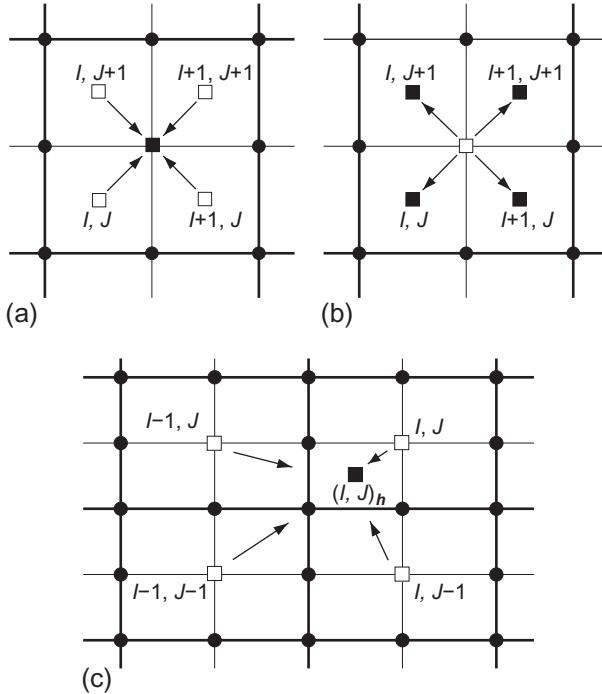


Figure 9.7 Solution interpolation and residual restriction (a), prolongation of coarse-grid correction (b and c) for a structured cell-centered scheme in 2D. Filled circle = grid point; filled rectangle = cell center to which it is interpolated; rectangle = cell center from which it is interpolated; thick line = coarse grid; thin line = fine grid.

Similar transfer operator is employed in 3D, where the summation is over the eight fine-grid control volumes, which form one coarse-grid cell.

The restriction operator is defined as a sum of the residuals from all cells which are contained in one coarse-grid control volume. Hence, in 2D we have (cf. Fig. 9.7a)

$$(I_h^{2h} \vec{R}_h^{n+1})_{IJ} = (\vec{R}_h^{n+1})_{IJ} + (\vec{R}_h^{n+1})_{I+1,J} + (\vec{R}_h^{n+1})_{I,J+1} + (\vec{R}_h^{n+1})_{I+1,J+1} \quad (9.27)$$

and likewise in 3D. Those residuals \vec{R}_h^{n+1} in Eq. (9.27), which are located outside the physical domain, have to be set to zero.

The prolongation of the coarse-grid correction Eq. (9.23) can be conducted in two different ways. First, a zeroth-order prolongation operator results, if $\delta \vec{W}_{2h}$ is equally distributed to all surrounding cell centers as indicated in Fig. 9.7b. Thus, for instance

$$(\vec{W}_h^+)_{IJ+1} = (\vec{W}_h^{n+1})_{IJ+1} + (\delta \vec{W}_{2h})_{IJ}, \quad \text{etc.} \quad (9.28)$$

The second possibility, which leads to a faster convergence of the multigrid scheme, consists of two steps. In first step, $\delta \vec{W}_{2h}$ is interpolated to the grid nodes like for the cell-vertex scheme (see below). In second step, the nodal values are averaged to obtain the value in the center of the fine-grid cell. Referring to Fig. 9.7c, the following final relationship can be derived

$$(I_{2h}^h \delta \vec{W}_{2h})_{IJ} = \frac{1}{16} [9(\delta \vec{W}_{2h})_{IJ} + 3(\delta \vec{W}_{2h})_{I-1,J} \\ + 3(\delta \vec{W}_{2h})_{I,J-1} + (\delta \vec{W}_{2h})_{I-1,J-1}] \quad (9.29)$$

The corresponding expression in 3D can be found by a similar procedure. It reads

$$(I_{2h}^h \delta \vec{W}_{2h})_{I,J,K} = \frac{1}{64} [27(\delta \vec{W}_{2h})_{I,J,K} + 9(\delta \vec{W}_{2h})_{I-1,J,K} \\ + 9(\delta \vec{W}_{2h})_{I,J-1,K} + 9(\delta \vec{W}_{2h})_{I,J,K-1} \\ + 3(\delta \vec{W}_{2h})_{I-1,J-1,K} + 3(\delta \vec{W}_{2h})_{I-1,J,K-1} \\ + 3(\delta \vec{W}_{2h})_{I,J-1,K-1} + (\delta \vec{W}_{2h})_{I-1,J-1,K-1}] \quad (9.30)$$

Transfer operators for the cell-vertex scheme

Because of the common nodes between the fine and coarse grid, the solution can be transferred simply by injection, that is,

$$(\vec{W}_{2h}^{(0)})_{i,j,k} = (\vec{W}_h^{n+1})_{i,j,k}. \quad (9.31)$$

The standard central restriction operator represents a linear interpolation from the nodes of all four (eight in 3D) fine-grid cells, which resemble one coarse-grid cell. According to Fig. 9.8, the restricted residual is computed in 2D as

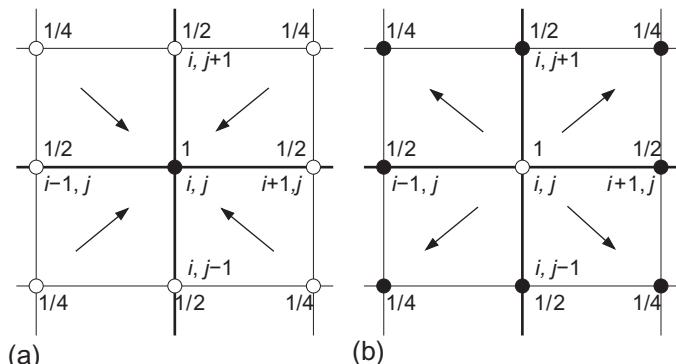


Figure 9.8 Interpolation factors in the case of the restriction (a) and the prolongation (b) for a structured cell-vertex scheme in 2D. Filled circle = point to which it is interpolated; circle = point from which it is interpolated; thick line = coarse grid; thin line = fine grid. Point (i, j) is common to both grids.

$$\begin{aligned}
(I_h^{2h} \vec{R}_h^{n+1})_{i,j} = & (\vec{R}_h^{n+1})_{i,j} + \frac{1}{2} \left[(\vec{R}_h^{n+1})_{i-1,j} + (\vec{R}_h^{n+1})_{i+1,j} \right. \\
& + (\vec{R}_h^{n+1})_{i,j-1} + (\vec{R}_h^{n+1})_{i,j+1} \Big] \\
& + \frac{1}{4} \left[(\vec{R}_h^{n+1})_{i-1,j-1} + (\vec{R}_h^{n+1})_{i+1,j-1} \right. \\
& \left. + (\vec{R}_h^{n+1})_{i-1,j+1} + (\vec{R}_h^{n+1})_{i+1,j+1} \right]. \tag{9.32}
\end{aligned}$$

In 3D, the fine-grid residuals are collected as follows

$$(I_h^{2h} \vec{R}_h^{n+1})_{i,j,k} = (\vec{R}_h^{n+1})_{i,j,k} + \frac{1}{2} \mathcal{A} + \frac{1}{4} \mathcal{B} + \frac{1}{8} \mathcal{C} \tag{9.33}$$

with the factors

$$\begin{aligned}
\mathcal{A} = & (\vec{R}_h^{n+1})_{i+1} + (\vec{R}_h^{n+1})_{i-1} + (\vec{R}_h^{n+1})_{j+1} + (\vec{R}_h^{n+1})_{j-1} + (\vec{R}_h^{n+1})_{k+1} + (\vec{R}_h^{n+1})_{k-1}, \\
\mathcal{B} = & (\vec{R}_h^{n+1})_{i+1,j+1} + (\vec{R}_h^{n+1})_{i-1,j+1} + (\vec{R}_h^{n+1})_{i+1,j-1} + (\vec{R}_h^{n+1})_{i-1,j-1} \\
& + (\vec{R}_h^{n+1})_{i+1,k+1} + (\vec{R}_h^{n+1})_{i-1,k+1} + (\vec{R}_h^{n+1})_{i+1,k-1} + (\vec{R}_h^{n+1})_{i-1,k-1} \\
& + (\vec{R}_h^{n+1})_{j+1,k+1} + (\vec{R}_h^{n+1})_{j-1,k+1} + (\vec{R}_h^{n+1})_{j+1,k-1} + (\vec{R}_h^{n+1})_{j-1,k-1}, \\
\mathcal{C} = & (\vec{R}_h^{n+1})_{i+1,j+1,k+1} + (\vec{R}_h^{n+1})_{i-1,j+1,k+1} + (\vec{R}_h^{n+1})_{i-1,j-1,k+1} + (\vec{R}_h^{n+1})_{i+1,j-1,k+1} \\
& + (\vec{R}_h^{n+1})_{i+1,j+1,k-1} + (\vec{R}_h^{n+1})_{i-1,j+1,k-1} + (\vec{R}_h^{n+1})_{i-1,j-1,k-1} + (\vec{R}_h^{n+1})_{i+1,j-1,k-1}. \tag{9.34}
\end{aligned}$$

In the above equation, only those indexes are shown which are different from i, j, k . It should be noted that the residuals \vec{R}_h^{n+1} must be set to zero at all physical boundary and dummy points *before* the restriction.

The prolongation of the coarse-grid correction can be implemented as a loop over the points of the coarse grid. Within the loop, the values $(\delta \vec{W}_{2h})_{i,j,k}$ are distributed to the fine-grid points using the same weights as for the restriction (cf. Fig. 9.8b). The particular contributions are summed up in order to obtain the complete transferred correction at each point of the fine grid.

Upwind prolongation (cell-vertex scheme)

In principle, upwind prolongation can be formulated either in characteristic or in conservative variables. A particularly efficient implementation in conservative variables was proposed in Ref. [16]. The methodology employs upwind-biased interpolation of the corrections in Eq. (9.23) according to the Mach number and the velocity direction.

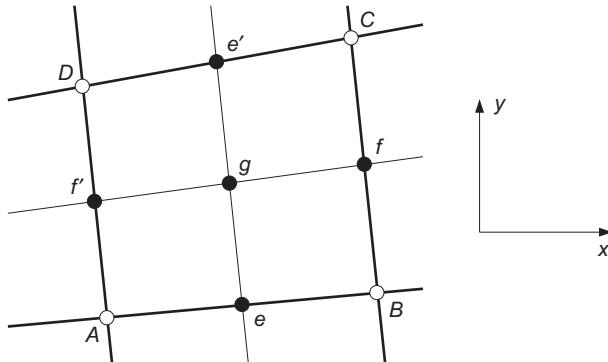


Figure 9.9 Upwind prolongation in 2D. Points A, B, C, D are common to the coarse (thick line) and the fine grid. Points e, f, e', f', g belong the fine grid (thin line) only.

The numerical effort is very low, nevertheless the robustness of the multigrid scheme can be improved significantly for high Mach-number flows [16].

The interpolation of the solution corrections $\delta \vec{W}_{2h}$ to the finer grid is accomplished in two steps. In the first step, the corrections at the points A, B, C , and D (see Fig. 9.9), which are common to both grids, are transferred directly to the finer grid. In the second step, the corrections are interpolated to the points e, f, e', f' , and g , which are contained only on the finer grid. The interpolation depends on the sign and on the absolute value of the Mach number at the corresponding point. The Mach number in this case is calculated using the contravariant velocity. Thus, for example, at the point e the formula

$$M_e = \frac{\vec{n} \cdot \vec{v}_e}{c} \quad (9.35)$$

is employed. The normal vector \vec{n} in Eq. (9.35) can be obtained either by averaging the face vectors of the control volume (in the direction $A-B$), or by normalizing the vector from point A to point B . Then, the correction is transferred to point e according to the rule

$$(I_{2h}^h \delta \vec{W}_{2h})_e = \begin{cases} (\delta \vec{W}_{2h})_A & \text{if } M_e > 1, \\ \frac{1}{2} [(\delta \vec{W}_{2h})_A + (\delta \vec{W}_{2h})_B] & \text{if } |M_e| \leq 1, \\ (\delta \vec{W}_{2h})_B & \text{if } M_e < -1. \end{cases} \quad (9.36)$$

The same procedure applies also to the points f to f' . The upwinding helps to match the information exchange between the grids better to the real physics. The interpolation to

the point g is more difficult. In Ref. [16], the values at the surrounding points A to D were simply averaged

$$(I_{2h}^h \delta \vec{W}_{2h})_g = \frac{1}{4} [(\delta \vec{W}_{2h})_A + (\delta \vec{W}_{2h})_B + (\delta \vec{W}_{2h})_C + (\delta \vec{W}_{2h})_D]. \quad (9.37)$$

However, some sort of upwind weighted interpolation would be more appropriate. The upwind prolongation can be implemented in similar way also in 3D. Despite the simplification, encouraging results were obtained for a number of test cases [16].

9.4.4 Implementation on unstructured grids

As compared to the structured grids, the construction of the coarse grids is much more involved in the case of unstructured grids. The problem is how to construct an uniformly coarsened grid from a set of elements (grid cells) which have no particular ordering. Additionally, the ratio of the cell volumes of the coarse to the fine grid has to stay within a certain margins (about 4 in 2D and 8 in 3D). One possibility how to solve this problem is to apply the AMG methodology [44–53], which we briefly discussed at the beginning of [Section 9.4](#). However, the geometric multigrid is still more widely used. Therefore, we shall concentrate here on this approach.

Three main methods for the generation of coarse grids can be identified:

- nonnested grids approach,
- topological methods,
- agglomeration of control volumes.

Reviews of the above methods were presented in Refs. [59, 60].

The standard restriction and prolongation operators are based on purely geometrically defined interpolation. Leclercq and Stoufflet [61] suggested upwind transfer operators, which are particularly promising for flows with strong shocks. Their upwind restriction/prolongation is based on the transformation of the residuals/corrections into the characteristic variables. After an upwind-biased interpolation, the restricted/prolongated values are transformed back into the physical variables. Numerically much less expensive upwind multigrid method was presented in Ref. [16] (see also [Section 9.4.3](#), Eq. (9.36)).

Nonnested grids

The most obvious idea is to generate a sequence of completely independent, increasingly coarser grids [62–67]. It is not necessary that the grids contain any common nodes. Therefore, we speak of *nonnested* grids. However, it is important that the main geometrical features (leading and trailing edge, fuselage nose, etc.) are retained on all coarse grids. This is not easy to accomplish, particularly in the case of a geometrically complex configuration. Multigrid based on nonnested grids is hardly used today.

Topological methods

One particular approach applies graph-based algorithms in order to remove certain nodes from the fine grid. The remaining nodes are then re-triangulated [68, 69]. On the contrary to the nonnested grids approach, the interpolation between the grids becomes easier, since the successive grids contain common nodes. However, the method inherits the drawback of the nonnested grids with respect to geometry conformance.

A further topological method employs *grid refinement* [29, 70, 71]. The technique starts from a coarse grid and generates finer grids by element division. The methodology can be applied either over the whole physical domain or only locally (e.g., at boundary layers). The disadvantage of this approach is that the quality of the finest grid strongly depends on the initial coarse grid and the refinement procedure. The problem can be partially cured by edge swapping [72, 73].

Another idea for the generation of coarse grids is based on *edge collapsing* [74]. It was initially developed for inviscid flows on tetrahedral grids. The edge-collapsing method was further extended to viscous flows on mixed-element grids in Ref. [75].

Agglomeration multigrid method

An efficient methodology for unstructured grids is the so-called *agglomeration multigrid*. It was first presented by Lallemand [76], Lallemand et al. [77], and Koobus et al. [78]. Later on, the agglomeration multigrid was adopted by various authors [31, 79–87]. The method generates a coarse grid by fusing the *control volumes* of the finer grid with their neighbors. The resulting coarse grids consist of successively larger, irregularly shaped polyhedral cells. This is depicted in Fig. 9.10. As we can see, the agglomeration technique retains the full discretization of the boundary surfaces. This represents a significant advantage over all previously discussed unstructured multigrid methods. However, it should be mentioned that up to now, implementations of the agglomeration multigrid were based mostly on the median-dual cell-vertex scheme (Section 5.2.2). The application of the agglomeration multigrid to a cell-centered scheme (Section 5.2.1) was described in Ref. [79].

Generation of coarse grids by volume agglomeration

The volume agglomeration for a node-centered scheme proceeds in the following steps:

1. Build a list of the so-called *seed points*. Seed points are grid points selected to agglomerate the surrounding control volumes. The list of seed points can contain either those points which form an approximate maximal independent set [81], or simply all points of the current grid level.
2. Loop over all seed points.

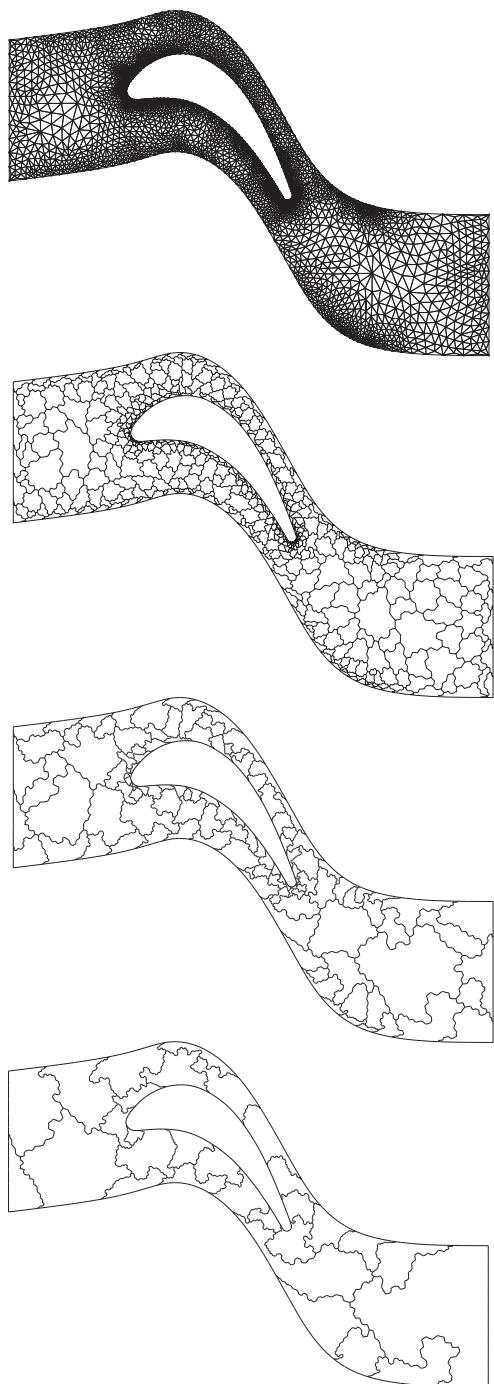


Figure 9.10 Generation of coarse grids by the agglomeration multigrid (median-dual scheme) in 2D. The sequence shows the finest grid and three coarse grids (top to bottom).

3. If the seed point is un-agglomerated, agglomerate all its nearest neighbors (connected by an edge), which were not already agglomerated.
4. Check the coarsening ratio (i.e., how many fine-grid control volumes are contained within a coarse-grid volume). If the ratio is less than four (eight in 3D), the neighbors of the already agglomerated nearest neighbors are added (if not associated with another seed point), until the optimum coarsening ratio is achieved. In Ref. [27], it was proposed to agglomerate those distance-two neighbors first, which are connected to at least two (three in 3D) agglomerated nearest neighbors.
5. If there are still seed points in the list, go to step 2.
6. Eliminate *singletons*. These are single control volumes which could not be agglomerated, because there were no un-agglomerated neighbors. A singleton can be eliminated by agglomeration with such neighboring control volume, which has the smallest coarsening ratio. This leads to coarse-grid levels with a more regular distribution of the control-volume sizes [31].

The above procedure is repeated until all coarse grids are generated.

The volume agglomeration has to start from the boundary in order to preserve grid isotropy. In 3D, user intervention may be required to prescribe the agglomeration direction depending on the shape of the boundary. To overcome this difficulty, Okamoto et al. [83] proposed another algorithm denoted as *global coarsening*. The method employs a global partitioning scheme, which is based on edge coloring. The partitioning scheme is used to generate an independent set of edges. In a second step, all control volumes, which share an edge of the independent set, are agglomerated. The procedure is repeated until the prescribed coarsening ratio is achieved. The method does not require the specification of an initial seed point or agglomeration direction. It can also treat any type of grid cells. One further alternative was developed by Moultsas and Karypis [88, 89]. It employs a multilevel paradigm in order to optimize a given coarse-grid quality measure.

Problems of agglomeration multigrid

The implementation of agglomeration multigrid presents little difficulty for inviscid flows. The Euler equations are in general discretized as fluxes over individual control-volume faces. In this respect, it does not matter how complex is the shape of the control volume (in fact, averaged face vectors are used). Furthermore, a first-order accurate spatial scheme requires the knowledge of flow quantities in the neighboring control volumes only. This information is readily available on the coarse grids.

In the case of viscous flows, the discretization of the diffusive fluxes on arbitrary shaped control volumes is no longer straightforward. The problem is the evaluation of gradients at face midpoints (see the discussion in Ref. [31]). Another, and even more

serious, difficulty is related to the required accuracy of the prolongation operator. The inequality in Eq. (9.25) suggests $m_p = 2$, since the standard restriction operator (sum of residuals) leads to $m_R = 1$ only. However, construction of linear interpolation is not easy on the coarse grids.

In order to circumvent the problem with a first-order accurate prolongation operator, Mavriplis [59] proposed to use constant prolongation (i.e., all points of the fine grid contained within an agglomerated coarse-grid volume get the same solution correction), and an additional scaling of the viscous fluxes (factor $0.5^{(\text{level}-1)}$; applied also to gradients). However, this approach does not lead to optimal multigrid efficiency.

Haselbacher [31] suggested to retain the fine-grid discretization of the viscous fluxes also on the coarse grids and to enforce the boundary conditions like on the finest grid. Moreover, he proposed a piecewise linear prolongation operator. For a scalar quantity U , it can be written as

$$I_{2h}^h \delta U_{2h} = (\delta U_{2h})_i + \Psi_i (\nabla \delta U_{2h})_i \cdot \vec{r}_{i,j}. \quad (9.38)$$

The gradient $(\nabla \delta U_{2h})_i$ in Eq. (9.38) is calculated by using the linear least-squares reconstruction described in Section 5.3.4, Eq. (5.55). The values of the limiter function Ψ_i are evaluated according to the Barth-Jespersen limiter function presented in Section 5.3.5.

9.5 PRECONDITIONING FOR LOW MACH NUMBERS

In the low subsonic Mach-number regime, when the magnitude of the flow velocity becomes small in comparison with the acoustic speed, the convective terms of the governing equations (2.19) become stiff. We can demonstrate this with the following example. In the 3-D case, we have the five eigenvalues

$$\begin{aligned} (\Lambda_c)_{1,2,3} &= V \quad (\text{convective modes}) \\ (\Lambda_c)_{4,5} &= V \pm c \quad (\text{acoustic modes}), \end{aligned} \quad (9.39)$$

where V denotes the contravariant velocity and c the speed of sound. The stiffness of the governing equations (when marching in time) is determined by the characteristic *condition number*. This number is defined as the ratio of the largest to the smallest eigenvalue

$$C_N = \frac{|(\Lambda_c)_{\max}|}{|(\Lambda_c)_{\min}|} = \frac{|V| + c}{|V|} = \frac{M + 1}{M}. \quad (9.40)$$

The allowable local time step is limited by the fastest wave, i.e., by $(\Lambda_c)_4$. During one time step, the slowest wave moves only over a fraction of the cell width: $\Lambda_{\min}\Delta t \approx (\Lambda_{\min}/\Lambda_{\max})h = h/C_N$. Thus, a large condition number C_N (i.e., for $M \rightarrow 0$) reduces the efficiency of wave propagation—it slows down the convergence to steady state [90]. Furthermore, it was demonstrated in Refs. [91, 92] that schemes for *compressible* flows have an amount of artificial dissipation which does not scale correctly for Mach numbers approaching zero. Thus, the accuracy of such spatial discretization suffers at low Mach numbers [93].

If the velocity in the entire flow field is low ($M < 0.2$), then the compressibility effects can be neglected and the incompressible equations can be utilized. The incompressible Navier-Stokes equations can be solved by the well-known pressure-based schemes [94–99]. The other possibility is the application of the artificial compressibility (or pseudo-compressibility) method [100–106]. However, there are flow cases like:

- High-speed flows with large embedded regions of low velocity. An example is the subsonic flow upstream of a strongly converging nozzle.
- Low-speed flows which are compressible due to density changes induced by heat sources. This occurs for surface heat transfer or volumetric heat addition (combustion simulation).
- Problems, where compressible and incompressible flow at varying Mach numbers occurs side by side—we speak of *all-speed flows*. Such situations arise, for instance, in propulsion, for high-lift configurations and in V/STOL maneuvering.

Such cases require the application of the compressible governing equations. In order to solve them efficiently and accurately at low Mach numbers, *preconditioning* can be employed. The advantage of preconditioning is that it enables a solution method, which is applicable at all Mach numbers. In the following, we shall derive the preconditioned governing equations.

9.5.1 Derivation of preconditioned equations

We demonstrate preconditioning with the aid of 1-D Euler equations. They can be written in differential form as

$$\frac{\partial \vec{W}}{\partial t} + \frac{\partial \vec{F}_c}{\partial x} = 0, \quad (9.41)$$

where $\vec{W} = [\rho, \rho u, \rho E]^T$ is the vector of conservative variables and \vec{F}_c denotes the convective fluxes. In order to see the effect of preconditioning on the spectral radii (important for the computation of the time step) and on the convective flux Jacobian (important for upwind dissipation), the Euler equations (9.41) are rewritten in the quasilinear form

$$\frac{\partial \vec{W}}{\partial t} + \bar{A}_c \frac{\partial \vec{W}}{\partial x} = 0 \quad (9.42)$$

with $\bar{A}_c = \partial \vec{F}_c / \partial \vec{W}$ being the convective flux Jacobian (cf. Section A.2).

The idea behind low Mach-number preconditioning is to transform the governing equations (9.41) such that the new equations have more favorable properties at low Mach numbers (i.e., below $M \approx 0.2$). First of all, we want to equalize the convective eigenvalues in order to bound the condition number (see Eq. (9.40)) and thus to remove the stiffness at $M \rightarrow 0$. We further want to change the scaling of the numerical dissipation in order to improve the accuracy. Finally, we want to couple the pressure and the velocity as it is done in the pressure-based schemes. The transformation replaces the conservative variables \vec{W} by a different set of flow variables. Various choices are possible [107], but the most often used are the pressure, the velocity components, and the temperature. Transforming the quasilinear form in Eq. (9.42) into the new variables \vec{W}_p , we obtain

$$\bar{P} \frac{\partial \vec{W}_p}{\partial t} + \bar{A}_c \bar{P} \frac{\partial \vec{W}_p}{\partial x} = 0. \quad (9.43)$$

where $\bar{P} = \partial \vec{W} / \partial \vec{W}_p$ represents the transformation matrix from the new variables \vec{W}_p into the conservative variables \vec{W} . Introducing a new flux Jacobian

$$\bar{A}_{c,p} = \bar{A}_c \bar{P} = \frac{\partial \vec{F}_c}{\partial \vec{W}_p}, \quad (9.44)$$

we can write Eq. (9.43) as

$$\bar{P} \frac{\partial \vec{W}_p}{\partial t} + \bar{A}_{c,p} \frac{\partial \vec{W}_p}{\partial x} = 0. \quad (9.45)$$

If we now replace \bar{P} in front of the time derivative by a suitable preconditioning matrix $\bar{\Gamma}$ (to be specified later), the preconditioned Eq. (9.45) reads

$$\bar{\Gamma} \frac{\partial \vec{W}_p}{\partial t} + \bar{A}_{c,p} \frac{\partial \vec{W}_p}{\partial x} = 0, \quad (9.46)$$

or equivalently

$$\frac{\partial \vec{W}_p}{\partial t} + \bar{\Gamma}^{-1} \bar{A}_{c,p} \frac{\partial \vec{W}_p}{\partial x} = 0. \quad (9.47)$$

It is now possible to solve Eq. (9.47) in the new variables \vec{W}_p using any standard spatial and temporal discretization scheme. Another possibility is to transform the preconditioned equations (9.47) back into the conservative variables \vec{W} . We have

$$\bar{P}^{-1} \frac{\partial \vec{W}}{\partial t} + \bar{\Gamma}^{-1} \bar{A}_{c,p} \bar{P}^{-1} \frac{\partial \vec{W}}{\partial x} = 0, \quad (9.48)$$

where $\bar{P}^{-1} = \partial \vec{W}_p / \partial \vec{W}$ is the transformation matrix from the conservative variables. Equation (9.48) can be for convenience written as

$$\frac{\partial \vec{W}}{\partial t} + \bar{P} \bar{\Gamma}^{-1} \bar{A}_{c,p} \bar{P}^{-1} \frac{\partial \vec{W}}{\partial x} = 0, \quad (9.49)$$

or

$$\frac{\partial \vec{W}}{\partial t} + \bar{P} \bar{\Gamma}^{-1} \bar{A}_c \frac{\partial \vec{W}}{\partial x} = 0. \quad (9.50)$$

The term $\bar{P} \bar{\Gamma}^{-1}$ in Eqs. (9.49) and (9.50) is called the *conservative variable preconditioning matrix*.

We can see now that the preconditioned equations in the conservative variables (9.49) and (9.50) have the following features:

- Spatial derivatives are multiplied by $\bar{P} \bar{\Gamma}^{-1}$.
- Unsteady equations are different from the original form in Eq. (9.42) and hence the solution is no longer time accurate.
- Stationary solution (i.e., $\partial \vec{W} / \partial t = 0$) remains unchanged.
- Eigenvalues and eigenvectors of the preconditioned system correspond to those of the matrix $\bar{P} \bar{\Gamma}^{-1} \bar{A}_c = \bar{P} \bar{\Gamma}^{-1} \bar{A}_{c,p} \bar{P}^{-1}$ and hence are different from those of the original system in Eq. (9.41).

The main task is now to find a suitable set of variables \vec{W}_p and the matrix $\bar{\Gamma}$ such that the convective eigenvalues of the preconditioned system (9.50) are equalized as close as possible. But it is also important that the matrices remain defined for $M \rightarrow 0$. Furthermore, it is desirable that the preconditioned system converts into the original system for higher Mach numbers.

Before we present the transformation and preconditioning matrices in Section 9.5.3, we shall discuss the implementation of the low Mach-number preconditioning in a flow solver.

9.5.2 Implementation

Since we solve the governing equations in their integral form, we have to write Eq. (9.50) as

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega + \bar{P} \bar{\Gamma}^{-1} \oint_{\partial\Omega} (\vec{F}_c - \vec{F}_v) dS = \bar{P} \bar{\Gamma}^{-1} \int_{\Omega} \vec{Q} d\Omega. \quad (9.51)$$

It is important to note that Eq. (9.51) is not conservative for unsteady flows. Therefore, the dual time-stepping approach (see Section 6.3) has to be employed in order to obtain a time-accurate solution.

The solution of the preconditioned governing equations (9.51) using an explicit multistage scheme (described in Section 6.1) proceeds according to the following steps:

1. Compute a new time step based on the spectral radii of the matrix $\bar{P}\bar{\Gamma}^{-1}\bar{A}_c$.
2. Evaluate artificial dissipation using the spectral radii (e.g., JST scheme) or the eigenvalues and eigenvectors of the preconditioned flux Jacobian (e.g., Roe upwind scheme). The dissipation can be formulated either in the conservative or in the primitive variables [107].
3. Compute the convective fluxes (either without change or based on the new variables \vec{W}_p).
4. Sum up the dissipative and convective fluxes. Depending on the formulation of the artificial dissipation, either the whole residual or just the convective terms are multiplied by the conservative variable preconditioning matrix, that is, by $\bar{P}\bar{\Gamma}^{-1}$.
5. Multiply the residual by $\alpha_k \Delta t / V$.
6. Carry out the IRS (optionally).
7. Subtract the residuals from the old conservative variables $\vec{W}^{(0)}$ in order to obtain the new conservative variables \vec{W}^{n+1} .
8. Update the boundary conditions (note that all boundary conditions which are based on the characteristic variables—like inflow, outflow, far-field—need to be changed according to $\bar{P}\bar{\Gamma}^{-1}\bar{A}_c$).

Similar procedure is followed for an implicit scheme, only the steps 5 and 6 are omitted and the conservative variables are updated in a different way (cf. Section 6.2). It should also be noted that the preconditioning has to be included in the implicit operator since the Jacobian is $\bar{P}\bar{\Gamma}^{-1}\bar{A}_c$.

The preconditioned scalar dissipation scheme (JST) takes the form (see Eq. (4.50))

$$\begin{aligned} \vec{D}_{I+1/2} = & \hat{\Lambda}_{I+1/2}^S (\bar{\Gamma}\bar{P}^{-1})_{I+1/2} \left[\epsilon_{I+1/2}^{(2)} (\vec{W}_{I+1} - \vec{W}_I) \right. \\ & \left. - \epsilon_{I+1/2}^{(4)} (\vec{W}_{I+2} - 3\vec{W}_{I+1} + 3\vec{W}_I - \vec{W}_{I-1}) \right]. \end{aligned} \quad (9.52)$$

Note that the term $\bar{\Gamma}\bar{P}^{-1}$ is required if the complete residual is later multiplied by $\bar{P}\bar{\Gamma}^{-1}$, since the spectral radius $\hat{\Lambda}^S$ already contains the preconditioning matrix (thus it is different from Eq. (4.53)). It is also possible to combine \bar{P}^{-1} and \vec{W} into W_p in Eq. (9.52). Then, we can formulate the preconditioned scalar dissipation scheme in primitive variables as

$$\vec{D}_{I+1/2} = \hat{\Lambda}_{I+1/2}^S \left[\epsilon_{I+1/2}^{(2)} (\vec{W}_{p,I+1} - \vec{W}_{p,I}) - \epsilon_{I+1/2}^{(4)} (\vec{W}_{p,I+2} - 3\vec{W}_{p,I+1} + 3\vec{W}_{p,I} - \vec{W}_{p,I-1}) \right]. \quad (9.53)$$

It is important to realize that in this case only the convective fluxes are multiplied by Γ^{-1} and the complete residual then by \bar{P} . The nonconservative form in Eq. (9.53) is somewhat simpler than the relation (9.52); however, there are problems with the accuracy for internal flows or for all-speed flows containing shocks.

The preconditioned Roe upwind scheme can be written as (cf. Eq. (4.91))

$$(\vec{F}_c)_{I+1/2} = \frac{1}{2} \left[\vec{F}_c(\vec{W}_R) + \vec{F}_c(\vec{W}_L) - (\bar{P}\bar{\Gamma}^{-1})_{I+1/2} |\bar{A}_{Roe}|_{I+1/2} (\vec{W}_R - \vec{W}_L) \right], \quad (9.54)$$

where $|\bar{A}_{Roe}| = \bar{T}_{c,p} |\bar{\Lambda}_{c,p}| \bar{T}_{c,p}^{-1}$. The left ($\bar{T}_{c,p}^{-1}$) and right ($\bar{T}_{c,p}$) eigenvectors, as well as the eigenvalues ($\bar{\Lambda}_{c,p}$) are those of the matrix $\bar{P}\bar{\Gamma}^{-1}\bar{A}_c = \bar{P}\bar{\Gamma}^{-1}\bar{A}_{c,p}\bar{P}^{-1}$. Values of the flow variables at $(I + 1/2)$ are obtained by Roe's averaging given in Eq. (4.89), and the whole residual is multiplied by $\bar{P}\bar{\Gamma}^{-1}$. Again, the preconditioned Roe scheme in Eq. (9.54) can be formulated in the primitive variables \vec{W}_p leading us to

$$(\vec{F}_c)_{I+1/2} = \frac{1}{2} \left[\vec{F}_c(\vec{W}_{p,R}) + \vec{F}_c(\vec{W}_{p,L}) - \bar{\Gamma}_{I+1/2} |\bar{A}_{Roe,p}|_{I+1/2} (\vec{W}_{p,R} - \vec{W}_{p,L}) \right]. \quad (9.55)$$

The eigenvalues and eigenvectors which compose $\bar{A}_{Roe,p}$ are now determined by the matrix $\Gamma^{-1}\bar{A}_{c,p}$ (cf. Eq. (9.47)). The eigenvalues of $\Gamma^{-1}\bar{A}_{c,p}$ are identical to the eigenvalues of $\bar{P}\bar{\Gamma}^{-1}\bar{A}_c$, that is, to $\bar{\Lambda}_{c,p}$, but the eigenvectors are different. Thus, the Roe matrix in Eq. (9.55) is composed as $|\bar{A}_{Roe,p}| = \bar{T}_p |\bar{\Lambda}_{c,p}| \bar{T}_p^{-1}$. It should be mentioned that also in this case, the residual has to be multiplied by $\bar{P}\bar{\Gamma}^{-1}$ in order to transform it into the conservative variables. For a discussion of upwind schemes at low Mach numbers see also Refs. [108, 109].

9.5.3 Form of the matrices

Among the various choices for the primitive variables \vec{W}_p , the form

$$\vec{W}_p = [p, u, v, w, T]^T \quad (9.56)$$

appears most often. Therefore, we shall restrict the further discussion to this particular form of \vec{W}_p . In the following, we will present the transformation and the preconditioning matrices together with the eigenvalues and the left and right eigenvectors for a general fluid, as well as a perfect gas.

Transformation matrices

For a general fluid, the transformation matrix from the conservative into the primitive variables $\bar{P}^{-1} = \partial \vec{W}_p / \partial \vec{W}$ is given by [110]

$$\bar{P}^{-1} = \begin{bmatrix} \frac{\rho h_T + \rho_T(H - q^2)}{a_1} & \frac{\rho_T u}{a_1} & \frac{\rho_T v}{a_1} & \frac{\rho_T w}{a_1} & -\frac{\rho_T}{a_1} \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \frac{1 - \rho_p(H - q^2) - \rho h_p}{a_1} & -\frac{\rho_p u}{a_1} & -\frac{\rho_p v}{a_1} & -\frac{\rho_p w}{a_1} & \frac{\rho_p}{a_1} \end{bmatrix} \quad (9.57)$$

with

$$\begin{aligned} q^2 &= \|\vec{v}\|_2^2 = u^2 + v^2 + w^2, \\ a_1 &= \rho \rho_p h_T + \rho_T(1 - \rho h_p). \end{aligned} \quad (9.58)$$

The transformation matrix from the primitive into the conservative variables, that is, $\bar{P} = \partial \vec{W} / \partial \vec{W}_p$ reads [110]

$$\bar{P} = \begin{bmatrix} \rho_p & 0 & 0 & 0 & \rho_T \\ \rho_p u & \rho & 0 & 0 & \rho_T u \\ \rho_p v & 0 & \rho & 0 & \rho_T v \\ \rho_p w & 0 & 0 & \rho & \rho_T w \\ \rho_p H - \rho h_p & \rho u & \rho v & \rho w & \rho_T H + \rho h_T \end{bmatrix}. \quad (9.59)$$

Derivatives of the density and of the enthalpy with respect to the pressure and the temperature in Eqs. (9.57)–(9.59) can be written in the form

$$\begin{aligned} \rho_p &= \rho \alpha_p, \\ \rho_T &= -\rho \alpha_T, \\ h_p &= \frac{1 - \alpha_T T}{\rho}, \\ h_T &= c_p, \end{aligned} \quad (9.60)$$

where α_p and α_T are the compressibility coefficients at constant pressure and temperature, respectively. The speed of sound can be computed from

$$c^2 = \frac{\rho h_T}{a_1}. \quad (9.61)$$

In the case of a perfect gas (see Section 2.4.1), the compressibility coefficients in Eq. (9.60) become $\alpha_p = 1/p$ and $\alpha_T = 1/T$. In this instance, the transformation matrix from the conservative into the primitive variables $\bar{P}^{-1} = \partial \vec{W}_p / \partial \vec{W}$ from Eq. (9.57) can be cast into

$$\bar{P}^{-1} = \begin{bmatrix} (\gamma - 1) \frac{q^2}{2} & (1 - \gamma)u & (1 - \gamma)v & (1 - \gamma)w & \gamma - 1 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \frac{1}{\rho} \left[\frac{\gamma q^2}{2c_p} - T \right] & -\frac{\gamma u}{c_p \rho} & -\frac{\gamma v}{c_p \rho} & -\frac{\gamma w}{c_p \rho} & \frac{\gamma}{c_p \rho} \end{bmatrix}. \quad (9.62)$$

The transformation matrix from the primitive into the conservative variables, that is, $\bar{P} = \partial \vec{W} / \partial \vec{W}_p$ reads for a perfect gas

$$\bar{P} = \begin{bmatrix} \frac{\rho}{p} & 0 & 0 & 0 & -\frac{\rho}{T} \\ \frac{\rho u}{p} & \rho & 0 & 0 & -\frac{\rho u}{T} \\ \frac{\rho v}{p} & 0 & \rho & 0 & -\frac{\rho v}{T} \\ \frac{\rho w}{p} & 0 & 0 & \rho & -\frac{\rho w}{T} \\ \frac{\rho E}{p} & \rho u & \rho v & \rho w & -\frac{\rho q^2}{2T} \end{bmatrix} \quad (9.63)$$

with q^2 as defined in Eq. (9.58).

Preconditioning matrices

The construction of a preconditioning matrix is relatively easy in the case of the Euler equations. The formulation proposed by Van Leer et al. [111] achieves the lowest attainable condition number. The methodology was discussed in detail in Ref. [91]. However, the preconditioning of the Navier-Stokes equations is more involved. The reason is that the viscous terms lead to complex wave speeds, which makes the preconditioned system difficult to analyze. The most recognized preconditioners for

viscous flows were proposed by Choi and Merkle [112, 113], Turkel [107, 114–116], Lee and van Leer [117, 118] and Lee [92], Jorgenson and Pletcher [119], and Weiss and Smith [48, 120], respectively. Examples of applications can be found in Refs. [110, 121–135].

Weiss and Smith preconditioner

The preconditioning matrix $\bar{\Gamma}$ due to Weiss and Smith [48, 120] has, in the case of a general fluid, a form identical to \bar{P} in Eq. (9.59) with ρ_p replaced by a suitable *preconditioning parameter* θ . The same holds also for the inverse of the preconditioning matrix, that is, $\bar{\Gamma}^{-1}$ which resembles \bar{P}^{-1} from Eq. (9.57). Consequently, the parameter a_1 from Eq. (9.58) is changed into

$$a_1^\Gamma = \rho \theta h_T + \rho_T(1 - \rho h_p). \quad (9.64)$$

In the case of a perfect gas, the preconditioning matrix $\bar{\Gamma}$ can be written as

$$\bar{\Gamma} = \begin{bmatrix} \theta & 0 & 0 & 0 & -\frac{\rho}{T} \\ \theta u & \rho & 0 & 0 & -\frac{\rho u}{T} \\ \theta v & 0 & \rho & 0 & -\frac{\rho v}{T} \\ \theta w & 0 & 0 & \rho & -\frac{\rho w}{T} \\ \theta H - 1 & \rho u & \rho v & \rho w & -\frac{\rho q^2}{2T} \end{bmatrix}, \quad (9.65)$$

where θ is again the preconditioning parameter, which will be defined later. The inverse of the preconditioning matrix is given by

$$\bar{\Gamma}^{-1} = \begin{bmatrix} a_2 \left[\frac{c^2}{\gamma - 1} - (H - q^2) \right] & -a_2 u & -a_2 v & -a_2 w & a_2 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ a_3 \left[1 - \theta(H - q^2) \right] & -\theta a_3 u & -\theta a_3 v & -\theta a_3 w & \theta a_3 \end{bmatrix} \quad (9.66)$$

with the abbreviations

$$\begin{aligned} a_2 &= (\gamma - 1)\phi, \\ a_3 &= \frac{(\gamma - 1)\phi T}{\rho}, \\ \phi &= \frac{1}{\theta c^2 - (\gamma - 1)}. \end{aligned} \quad (9.67)$$

Several choices exist for the preconditioning parameter θ in Eqs. (9.64)–(9.67). One definition, which is based on a reference velocity u_r was provided in Ref. [48]. It reads for a general fluid

$$\theta = \frac{1}{u_r^2} - \frac{\rho_T}{\rho h_T}, \quad (9.68)$$

or, correspondingly for a perfect gas,

$$\theta = \frac{1}{u_r^2} + (\gamma - 1) \frac{1}{c^2}. \quad (9.69)$$

The reference velocity u_r in Eq. (9.68) or Eq. (9.69) is obtained from the relation

$$u_r = \min \left[\max \left(||\vec{v}||_2, \frac{\nu}{\Delta h}, \frac{\kappa}{\Delta h}, \epsilon \sqrt{\frac{|\Delta p|}{\rho}} \right), c \right], \quad (9.70)$$

where Δh is a measure of the control volume size, the quantity Δp stands for the pressure difference between the adjacent control volumes and ϵ is a small number ($\approx 10^{-3}$). As we can see from the definition in Eq. (9.70), the reference velocity is bounded by the local transport velocity. The terms $\nu/\Delta h$ and $\kappa/\Delta h$ become important in boundary layers with dominant diffusion or heat conduction. The pressure term is intended to prevent u_r from vanishing at stagnation points. In the case that $u_r = c$, $\bar{\Gamma}$ becomes identical to \bar{P} and $\bar{\Gamma}^{-1}$ is converted into \bar{P}^{-1} . Hence, the preconditioning is turned off for a supersonic flow as intended.

Another possibility, which was devised for a perfect gas, is to set [123]

$$\theta = \frac{1}{\beta \gamma R T} = \frac{1}{\beta c^2}. \quad (9.71)$$

The parameter β in Eq. (9.71) is defined as

$$\beta = \frac{M_r^2}{1 + (\gamma - 1)M_r^2} \quad (9.72)$$

with the reference Mach number given by

$$M_r^2 = \max [\min(M^2, 1), M_{\min}^2], \quad (9.73)$$

and M being the local Mach number ($M^2 = ||\vec{v}||_2^2/c^2$). According to Ref. [123], parameter $M_{\min}^2 = KM_\infty^2$ and $K \approx 3$ (others choose $K = 1$ or even $K = 0.15$; the exact value seems to depend on the number of control volumes in the stagnation region or inside the boundary layer). As it can be easily verified, when $M \geq 1$ the parameter β equals to $1/\gamma$ and the matrix $\bar{\Gamma}$ becomes identical to \bar{P} . It should be mentioned that both definitions of θ in Eq. (9.69) and in Eq. (9.71) are equivalent. Hence, the reference Mach number could be determined as $M_r = u_r/c$ with the help of Eq. (9.70).

Eigenvalues of the preconditioned system

The matrix of the eigenvalues of the preconditioned system Eq. (9.49), i.e., $\bar{P}\bar{\Gamma}^{-1}\bar{A}_c = \bar{P}\bar{\Gamma}^{-1}\bar{A}_{c,p}\bar{P}^{-1}$ is given by

$$\bar{\Lambda}_{c,p} = \begin{bmatrix} V & 0 & 0 & 0 & 0 \\ 0 & V & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & \frac{(a_4 + 1)}{2} V + c' & 0 \\ 0 & 0 & 0 & 0 & \frac{(a_4 + 1)}{2} V - c' \end{bmatrix}, \quad (9.74)$$

where $V = \vec{v} \cdot \vec{n}$ represents the contravariant velocity, and

$$c' = \frac{1}{2} \sqrt{V^2(a_4 - 1)^2 + 4a_5} \quad (9.75)$$

denotes the modified speed of sound. The parameters a_4 and a_5 in Eqs. (9.74) and (9.75) read in the general case

$$\begin{aligned} a_4 &= \frac{a_1}{a_1^\Gamma}, \\ a_5 &= \frac{\rho h_T}{a_1^\Gamma}, \end{aligned} \quad (9.76)$$

where a_1 is defined in Eq. (9.58) and a_1^Γ in Eq. (9.64). In the case of a perfect gas and the definition of the preconditioning parameter θ according to Eq. (9.69), the parameters become $a_4 = \phi$ and $a_5 = \phi c^2$, with ϕ given by Eq. (9.67). When using the second

definition of θ from Eq. (9.71), the parameters in Eq. (9.76) simplify to $a_4 = M_r^2$ and $a_5 = M_r^2 c^2$, respectively. As we can see, $c' \approx (V/2)\sqrt{5}$ for $|M| \rightarrow 0$ and hence the eigenvalues become equalized as intended. In this way, the stiffness represented by the condition number Eq. (9.40) is reduced (the condition number is $C_N \approx 2.6$) and the convergence of the time-stepping or iterative solution process is dramatically enhanced. On the other hand, $a_4 = 1$ and $c' = c$ for $|M| \geq 1$, and thus the eigenvalues of \bar{A}_c are recovered.

Based on Eq. (9.74), the spectral radius of the preconditioned system becomes

$$\hat{\Lambda}_{c,p} = \left[\frac{(a_4 + 1)}{2} |V| + c' \right] \Delta S \quad (9.77)$$

with ΔS denoting the face area. This expression is employed to compute the time step in Eq. (6.14), (6.18), (6.20), or (6.22). It also replaces the spectral radius in Eq. (4.53) in the case of the central artificial dissipation (see also Eqs. (5.33), (9.52), and (9.53)).

Eigenvectors of the preconditioned system

The left and right eigenvectors of the convective flux Jacobian (cf. Section A.11) will be changed by the preconditioning. This is of importance for spatial discretizations based on characteristic variables like Roe's upwind scheme (Section 4.3.3), or the upwind TVD scheme presented in Section 4.3.4.

The matrix of the left eigenvectors of the preconditioned new flux Jacobian $\Gamma^{-1}\bar{A}_{c,p}$ in Eq. (9.47) can be written as [110]

$$\bar{T}_p^{-1} = \begin{bmatrix} -\frac{\rho_T n_x}{a_1^\Gamma} & 0 & \frac{a_6 n_z}{c'} & -\frac{a_6 n_y}{c'} & -\frac{a_6 n_x}{T} \\ -\frac{\rho_T n_y}{a_1^\Gamma} & -\frac{a_6 n_z}{c'} & 0 & \frac{a_6 n_x}{c'} & -\frac{a_6 n_y}{T} \\ -\frac{\rho_T n_z}{a_1^\Gamma} & \frac{a_6 n_y}{c'} & -\frac{a_6 n_x}{c'} & 0 & -\frac{a_6 n_z}{T} \\ \frac{1}{2} + a_7 & \frac{a_6 n_x}{2c'} & \frac{a_6 n_y}{2c'} & \frac{a_6 n_z}{2c'} & 0 \\ \frac{1}{2} - a_7 & -\frac{a_6 n_x}{2c'} & -\frac{a_6 n_y}{2c'} & -\frac{a_6 n_z}{2c'} & 0 \end{bmatrix}. \quad (9.78)$$

The matrix of the right eigenvectors of $\Gamma^{-1}\bar{A}_{c,p}$ multiplied by $\bar{\Gamma}$, as indicated in Eq. (9.55), is given by the formula [110]

$$\bar{\Gamma} \bar{T}_p = \frac{a_1^\Gamma}{\rho h_T} \begin{bmatrix} -a_8 n_x & -a_8 n_y \\ -a_8 u n_x & -a_8 u n_y - c' n_z \\ -a_8 v n_x + c' n_z & -a_8 v n_y \\ -a_8 w n_x - c' n_y & -a_8 w n_y + c' n_x \\ a_{11} - a_{10} n_x & a_{12} - a_{10} n_y \\ -a_8 n_z & 1 & 1 \\ -a_8 u n_z + c' n_y & u + (a_9 + c') n_x & u + (a_9 - c') n_x \\ -a_8 v n_z - c' n_x & v + (a_9 + c') n_y & v + (a_9 - c') n_y \\ -a_8 w n_z & w + (a_9 + c') n_z & w + (a_9 - c') n_z \\ a_{13} - a_{10} n_z & H + (a_9 + c') V & H + (a_9 - c') V \end{bmatrix}, \quad (9.79)$$

where a_1^Γ is defined in Eq. (9.64). Further parameters in the Eqs. (9.78) and (9.79) read

$$a_6 = \rho a_5$$

$$a_7 = \frac{V(a_4 - 1)}{4c'}$$

$$a_8 = \frac{\rho_T T}{\rho}$$

$$a_9 = -\frac{1}{2} V(a_4 - 1) \quad (9.80)$$

$$a_{10} = a_8 H + T h_T$$

$$a_{11} = c'(v n_z - w n_y)$$

$$a_{12} = c'(w n_x - u n_z)$$

$$a_{13} = c'(u n_y - v n_x)$$

with a_4, a_5 defined in Eq. (9.76) and c' in Eq. (9.75), respectively.

The eigenvectors $\bar{T}_{c,p}$ and $\bar{T}_{c,p}^{-1}$ of the matrix $\bar{P} \bar{\Gamma}^{-1} \bar{A}_c = \bar{P} \bar{\Gamma}^{-1} \bar{A}_{c,p} \bar{P}^{-1}$, which are required for the preconditioned Roe scheme formulated in the conservative variables (see Eq. (9.54)), can be obtained by a modification of the above eigenvectors Eqs. (9.78) and (9.79). It can be shown that the columns of the right-eigenvector matrix of $\bar{P} \bar{\Gamma}^{-1} \bar{A}_c$

are composed of $\bar{P}\vec{x}$, where \vec{x} are the right eigenvectors of $\Gamma^{-1}\bar{A}_{c,p}$. Hence, the matrices of the right and left eigenvectors of $\bar{P}\bar{\Gamma}^{-1}\bar{A}_c$ can be expressed as

$$\begin{aligned}\bar{T}_{c,p} &= \bar{P}\bar{T}_p, \\ \bar{T}_{c,p}^{-1} &= \bar{T}_p^{-1}\bar{P}^{-1}.\end{aligned}\tag{9.81}$$

Inserting the above relations (9.81) into the formula for the preconditioned Roe scheme in the conservative variables Eq. (9.54), we obtain

$$\begin{aligned}(\vec{F}_c)_{I+1/2} &= \frac{1}{2} \left[\vec{F}_c(\vec{W}_R) + \vec{F}_c(\vec{W}_L) \right. \\ &\quad \left. - (\bar{\Gamma}\bar{T}_p|\bar{\Lambda}_{c,p}|\bar{T}_p^{-1}\bar{P}^{-1})_{I+1/2}(\vec{W}_R - \vec{W}_L) \right]\end{aligned}\tag{9.82}$$

with $\bar{\Gamma}\bar{T}_p$ given by Eq. (9.79) and \bar{T}_p^{-1} by Eq. (9.78), respectively.

An example of the application of the preconditioned finite-volume scheme from Eq. (9.51) with $\bar{\Gamma}$ according to Eq. (9.65) to airfoil flow is presented in Figs. 9.11–9.13. As we can observe, both the central scheme as well as Roe’s upwind scheme fail to deliver the correct solution for an inflow Mach number of 0.01. The schemes without preconditioning cannot predict the pressure distribution and hence the lift coefficient ($C_L = 0.323$ and 0.324 versus the correct 0.352). As demonstrated in Fig. 9.13, preconditioning helps to obtain the correct solution ($C_L = 0.353$), and it also significantly accelerates the convergence.

Note that both 2-D solvers provided with the source codes feature low Mach-number preconditioning for the central, as well as for Roe’s upwind scheme.

9.6 PARALLELIZATION

Nowadays, virtually all computers—desktops, laptops, tablets—have their central processing unit (CPU) composed of multiple cores. In addition to that, modern graphics processing units (GPUs) offer hundreds or even thousands of generally programmable processors. Harnessing this dramatically increased computing performance however requires an appropriately designed software. Although parallelization is not directly connected to any particular numerical methodology, it may influence the choices of numerical schemes and it certainly does their implementation.

Several programming paradigms are being employed today to parallelize fluid dynamics codes. First of all it is the message-passing interface (MPI), which works on a program level. Also using message passing between parallel tasks is the *asynchronous agents library*.² However in comparison to MPI, its set of commands is much more

² Where applicable, names are registered trademarks of their respective owners.

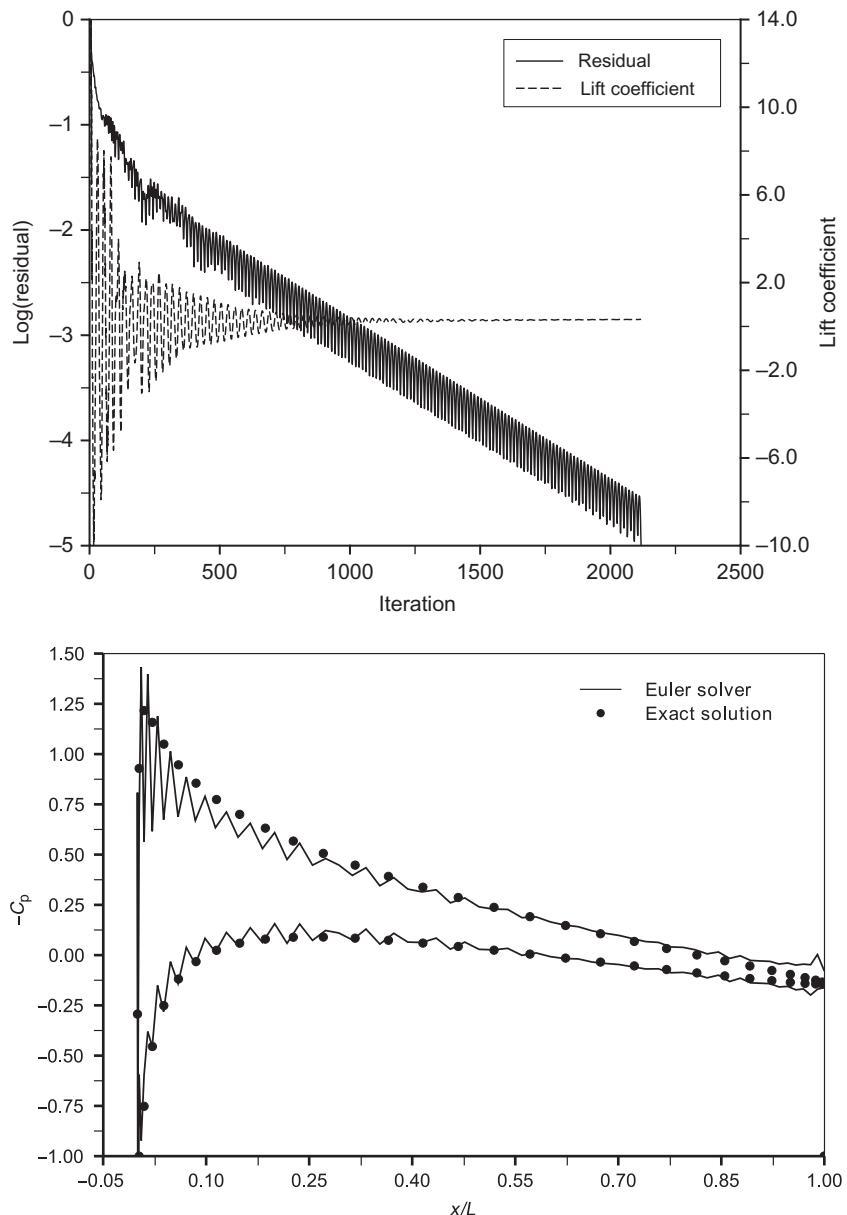


Figure 9.11 Inviscid 2-D flow around symmetric Joukowsky airfoil (10% thickness). Structured grid, $M_\infty = 10^{-2}$, $\alpha = 3^\circ$, central spatial discretization, explicit multistage time-stepping scheme, no preconditioning. Shown are the convergence history (top), and comparison of the pressure coefficient with the exact potential solution (bottom).

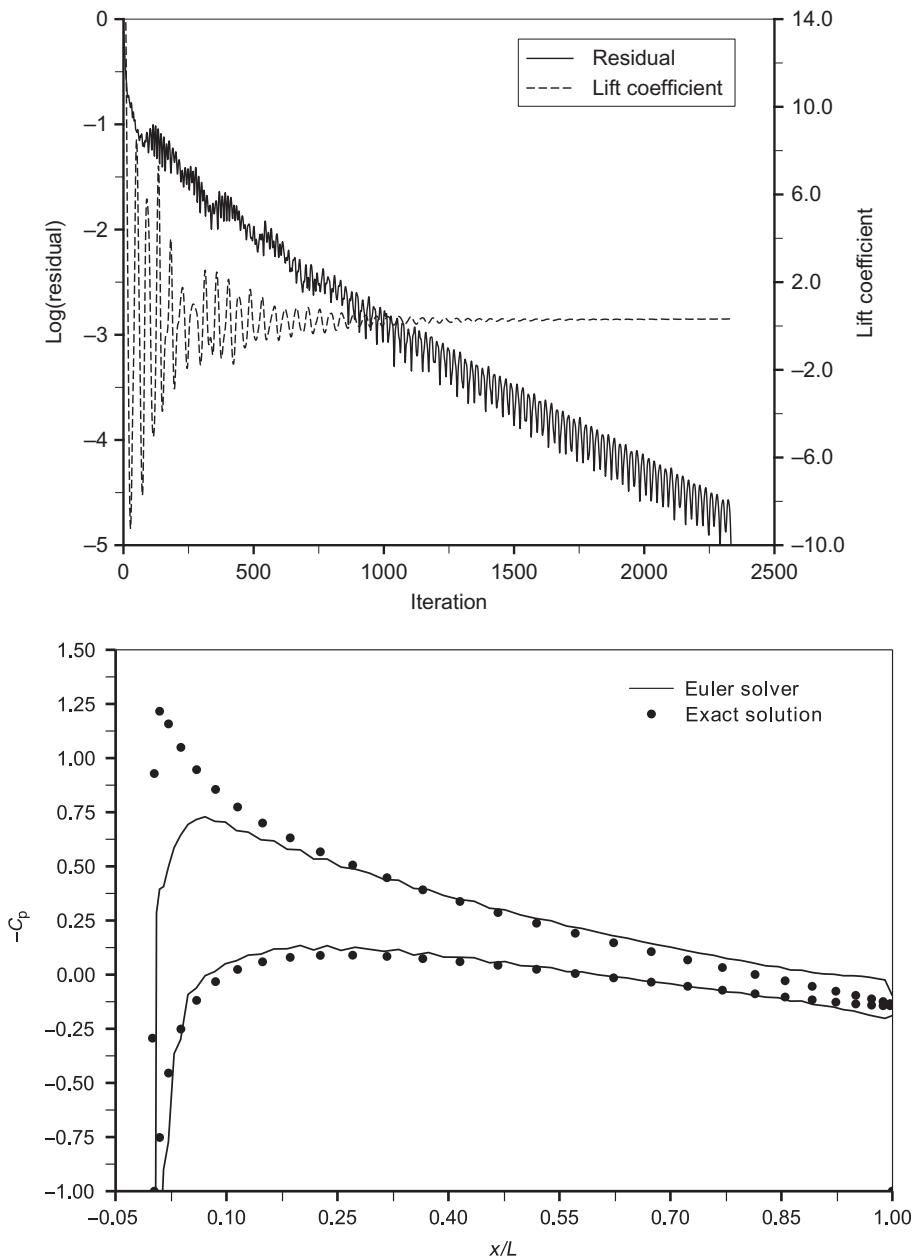


Figure 9.12 Inviscid 2-D flow around symmetric Joukowsky airfoil (10% thickness). Structured grid, $M_\infty = 10^{-2}$, $\alpha = 3^\circ$, second-order Roe's upwind discretization, explicit multistage time-stepping scheme, no preconditioning. Shown are the convergence history (top), and comparison of the pressure coefficient with the exact potential solution (bottom).

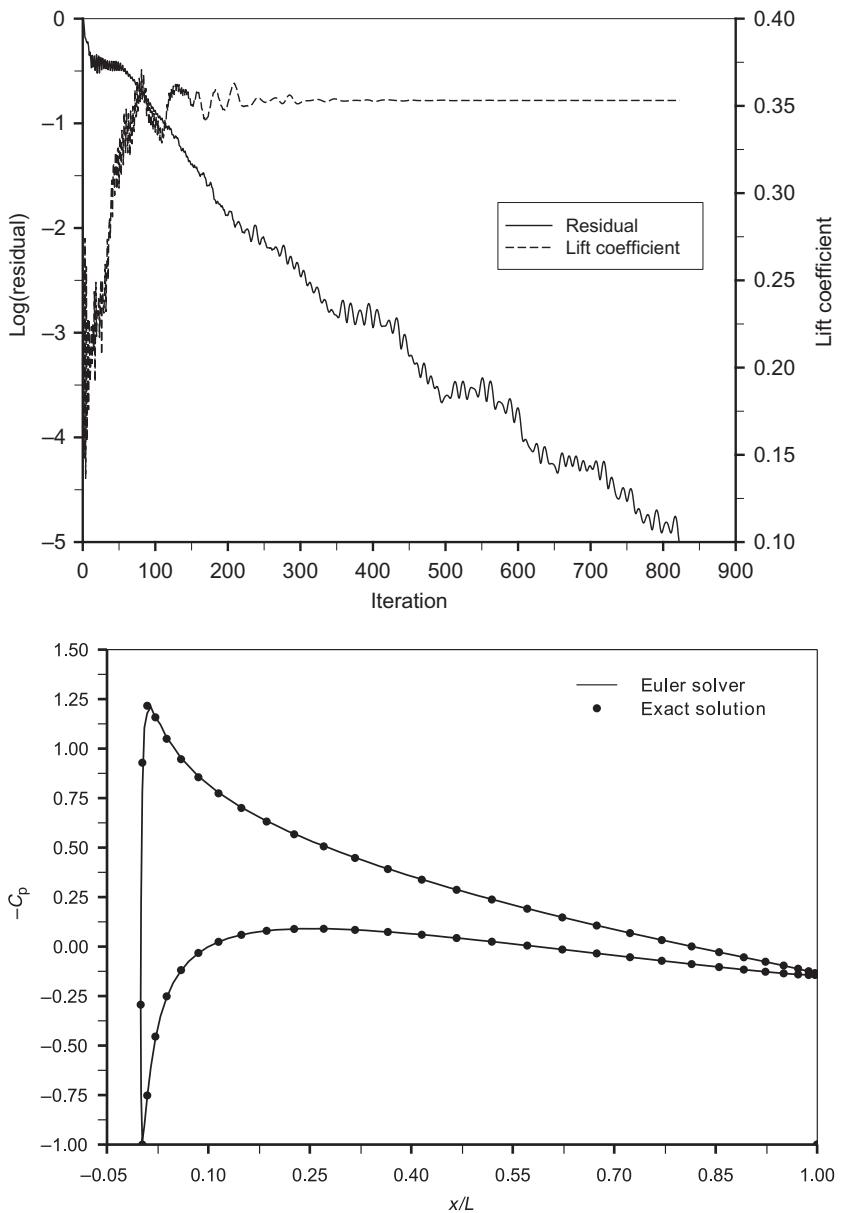


Figure 9.13 Inviscid 2-D flow around symmetric Joukowsky airfoil (10% thickness). Structured grid, $M_\infty = 10^{-2}$, $\alpha = 3^\circ$, central spatial discretization, explicit multistage time-stepping scheme, Weiss-Smith preconditioning. Shown are the convergence history (top), and comparison of the pressure coefficient with the exact potential solution (bottom).

restricted [136] Methods like open multi-processing (OpenMP), compute unified device architecture (CUDA), open computing language (OpenCL), or OpenACC [137] belong on the other hand to more fine-grained approaches, which work on function or loop level. The same holds also for the shortly (end of 2012) released *accelerated massive parallelism* (C++ AMP) programming model proposed by Microsoft [138]. It is possible to combine for example MPI with one of the other methodologies in order to achieve maximum speedup. Of course, this always depends on the capabilities of the available hardware.

In the following, four of the approaches will be discussed in more depth. In order to demonstrate practical use, we shall employ a simple program which solves 2-D Laplace equation with fixed boundary values by Jacobi iteration. To show a reduction operation, the 2-norm of the residual is computed as well. Source codes of the example programs are provided with the accompanying software in the directory `Parallelization`. The sources are written in C and in C++. More details regarding their organization and compilation can be found in Chapter 12 and Section 12.8.

9.6.1 MPI

The MPI is already the classical programming model for the parallelization of numerical codes written in C/C++ or Fortran. The basic idea is to split the computational grid into as many parts as there are processors (or CPU cores), assign each part to one of them, and then to compute the flow solution in parallel. A very simple program in C notation might look like this:

```
#include <mpi.h>
...
MPI_Init( &argc,&argv ); // initialize MPI
MPI_Comm_rank( MPI_COMM_WORLD,&myProc ); // individual process number
MPI_Comm_size( MPI_COMM_WORLD,&nProcs ); // total number of processes
...
err = ReadData( myProc );
...
err = Solve( myProc );
...
err = SaveSolution( myProc );
...
MPI_Finalize(); // finish all MPI operations
...
```

Since the flow in one grid part—called *partition*—usually depends on the flow in other partition(s), and because each process uses its own dedicated memory (distributed-memory model), it is necessary to exchange data between the processors. MPI offers for this purpose an explicit control over the interprocess communication.

Messages—flow variables, geometrical data, residuals, etc.—can be sent (`MPI_Send`, `MPI_Isend`) and received (`MPI_Recv`) by the partitions. Data can be gathered from all other processors (`MPI_Allreduce`—for example, the residual norm, forces and moments), or they can be broadcast from one partition to all the other (`MPI_Bcast` or `MPI_Scatter`). Another important possibility is the synchronization of the processes (`MPI_Barrier`). Description of the programming interface, libraries to download and examples can be found at Ref. [139]. A highly recommendable tutorial can be found in Ref. [140].

Although the concept of MPI is appealingly simple, the problems come with the details. The first question is how to best partition (split) a given grid into separate pieces. For a good parallel efficiency it is crucial that each of the processors is tasked with about the same amount of work. We speak in this respect of *load balancing*. In the case of a structured grid, computational work is proportional to the number of control volumes, that is, to the number of grid cells (cell-centered scheme) or grid nodes (node-centered scheme). If the grid consists of multiple grid blocks, those form the basis for the partitioning. It is advisable to divide a block along its largest dimension so that the number of faces (nodes) shared by neighboring partitions is minimized (to reduce the amount of data to communicate).

For an unstructured grid, the amount of work also depends on the number of edges (median-dual scheme) or on the number of cell faces (cell-centered scheme) within a partition. Computational effort for schemes with order of spatial accuracy >2 will further depend on the number of neighbors of the nearest neighbors. Given in addition the unordered numbering of nodes and cells, partitioning of an unstructured grid can become quite involved. Fortunately, there are a few libraries available for this purpose. The most convenient one (in author's view) is Metis [141]. Besides being fast and reliable, it offers a number of options useful for optimizing the load balancing.

The grid partitions have to exchange data at least along their common boundaries. Depending on the partitioning and also on the underlying discretization methodology, it might become necessary to communicate between partitions which touch each other just along cell edges or even at single nodes. Structured, multiblock flow solvers need the corresponding logic (see Section 8.9) already without any parallelization. However, solvers utilizing unstructured grids will require a fitting program design, or a substantial redesign of existing software, in order to accommodate MPI. This is a considerable difference to the other parallelization approaches, which can be incorporated successively into a code.

Which data have to be communicated between the partitions? Let us, for example, consider the unstructured, median-dual scheme of Section 5.2.2. First of all, control volumes computed separately in each partition will be left incomplete at the inter-process boundaries. It is therefore necessary to sum up the (partial) control volumes of all partitions which share a given boundary node. As stated previously, this might require

communication between partitions, which have just a single node in common. MPI does not provide any function for searching the related partitions, it must be provided by the programmer together with all necessary logic.

Furthermore, higher-order spatial discretizations and/or the solution of the Navier-Stokes equations require the computation of flow gradients and of limiter functions. In both cases, flow variables at the end nodes of all edges connected to a boundary node have to be known, and thus have to be communicated from the surrounding partitions (see the places where data are transferred between periodic boundaries in the example codes `Unstruct2D` provided with the sources). Once the residuals are computed, they too have to be summed up at the boundary nodes just like the control volumes above.

Explicit time-stepping schemes (see Section 6.1) do not require any communication between the partitions. This is however different in the case of IRS (Section 9.3.2) as well as for all implicit schemes. Those require at least an approximate form of coupling between neighboring partitions, otherwise the convergence and often the stability of the scheme will suffer considerably. Regarding the popular LU-SGS scheme (presented in Section 6.2.4), one possibility are additional smoothing sweeps along the boundaries [142]. In the case of the Newton-Krylov method discussed in Section 6.2.5, it is the preconditioner which requires modification, since the system matrix is now split between the processors. The most often used approach is the *additive Schwarz* method which employs overlaps between the partitions [143–146].

Helpful for the parallelization of a flow solver might be the PETSc library (portable, extensible toolkit for scientific computation) [147]. PETSc contains a suite of parallel linear, nonlinear equation solvers, and time integrators. The library uses MPI and can be invoked from MPI-based application codes written in C/C++ or in Fortran.

9.6.2 OpenMP

OpenMP is intended for shared-memory parallel programming in C/C++ and in Fortran [148]. It is based on splitting the computational task into multiple threads which are then executed in parallel, while sharing a common portion of application's memory. OpenMP provides the capability to implement both coarser-grain and fine-grain parallelism. It enables the programmer to incrementally parallelize a serial program, unlike the message-passing paradigm of MPI.

OpenMP offers a simple and limited set of compiler directives, so that significant parallelism can be achieved by using just a few lines of code. As an example may serve here the (simplified) main loop of the demonstration program `openmp` provided in the directory `Parallelization` of the source codes:

```
#include <omp.h>
...
#pragma omp parallel default(none) shared(phi,phiOld,Ni,Nj)
    private(i,j,ij)
{
    ...
#pragma omp for schedule(static) nowait
    for (j=1; j<(Nj-1); j++)
    {
        for (i=1; i<(Ni-1); i++)
        {
            ij      = i + j*Ni;
            phi[ij] = 0.25*(phiOld[ij+1]+phiOld[ij-1]+
                            phiOld[ij+Ni]+phiOld[ij-Ni]);
        }
    }
    ...
} // parallel region
```

Just two compiler directives (`#pragma omp`) are required to parallelize the *j*-loop. The first directive defines a parallel region, which encloses code to be executed in parallel. In this case, it also declares variables to be shared by all threads. The second directive sets up a parallel for-loop. The loop itself remains unchanged from the serial code. The directives are evaluated only if the corresponding compiler option is set (e.g., `-fopenmp` in the case of GNU compilers), otherwise the code is executed in serial.

The relative simplicity of OpenMP has certain downsides. There is, for example, no provision for parallel I/O like in MPI. Instead, the implementation is left up to the programmer. However, as already mentioned, OpenMPI and MPI can be combined together to achieve optimum parallel efficiency. Such combination can be in particular interesting for multi-physics simulation codes, as recently demonstrated for fluid flow with particles [149].

9.6.3 CUDA

The CUDA is a parallel programming and computing platform for C/C++ and Fortran [150]. Released publicly in 2007, CUDA offers code developers a direct access to the virtual instruction set and memory of the parallel computational elements in NVIDIA's GPUs, so that the GPUs can be utilized for general purpose processing. This approach is known as GPGPU. GPUs have a parallel throughput architecture optimized for executing many concurrent threads relatively slowly, rather than executing a single thread very quickly like CPUs. Thus, if a computational task can be split up into many parallel threads, it will be accelerated considerably.

The CUDA toolkit (available for free at Ref. [150]) includes a compiler, math libraries (e.g., CUBLAS for basic linear algebra functions), and tools for debugging and optimizing the performance of applications. Interesting for CFD programmers is also the CUDA-based template library named CUSP [151], which contains various solvers for sparse linear systems (e.g., GMRES). It can be downloaded for free at Ref. [152]. A detailed description of a CFD code accelerated using CUDA can be found in Ref. [153].

The CUDA platform is accessible through extensions to the standard programming languages C, C++, and Fortran. Consider, for example, the following excerpt from the demonstration code `cuda` located in the directory `Parallelization` of the sources:

```
#include <cuda_runtime.h>
#include <device_launch_parameters.h>

// Body of the kernel function
__global__ void Jacobi( int Ni, int Nj, float *phi, float *phiOld )
{
    int i = blockDim.x * blockIdx.x + threadIdx.x; // i, j indexes
    int j = blockDim.y * blockIdx.y + threadIdx.y; // in phi, phiOld
    ...
    int ij = i + j*Ni;
    phi[ij] = 0.25f*(phiOld[ij+1]+phiOld[ij-1]+
                    phiOld[ij+Ni]+phiOld[ij-Ni]);
}
...

// calling kernel function to execute on the GPU
Jacobi<<<blocksPerGrid,threadsPerBlock>>>( Ni,Nj,phiDev,phiDevOld );
...
```

CUDA enables the programmer to define functions called *kernels*, which are then executed N times in parallel by N different CUDA threads. Kernel function is defined by using the `__global__` qualifier, like in front of the `Jacobi` function declaration. The numbers of blocks and of threads that shall execute the particular kernel on the GPU is specified by the so-called *execution configuration*, which is an expression of the form `<<<...>>>` inserted between the function name and the parenthesized argument list. Each thread that executes the kernel function is given a unique index, which is accessible within the kernel through the built-in `threadIdx` variable. In order to simplify the treatment of 2-D and 3-D grids or of matrices, different block and thread numbers can be specified for each spatial direction using the structure `dim3` with the components x , y , and z . This is also employed in the above example which utilizes a 2-D grid in physical, as well as in computational space.

The computational efficiency of CUDA, the availability of optimized libraries like CUBLAS or CUSP, as well as the development and debugging tools are very appealing. An advantage of the toolkit is also the possibility of mixing native and CUDA code, both will get compiled together. A possible downside of CUDA is however its dependency on NVIDIA's graphics hardware. Thus, if your GPU comes from ATI, or if you intend to write a portable software, you will have to consider either C++ AMP or OpenCL, which is described next.

9.6.4 OpenCL

The development of OpenCL was initiated by Apple, Inc. It offers a uniform environment for parallel programming of heterogeneous systems that include CPUs and GPUs [154]. OpenCL views the computing system as a composition of compute devices, which might be CPUs or “accelerators” such as GPUs. It employs a C-like language (OpenCL C) for writing programs, called *kernels*, that execute on the compute devices. Each compute device usually consists of a number of individual processing elements (GPUs typically have hundreds or thousands of processors), and a kernel executes on the processing elements in parallel. The programming language used to write kernels is based on C99, but it is adapted to fit the device model in OpenCL. It omits function pointers, bit fields and variable-length arrays, and it also forbids recursion. The C standard library is replaced by a custom set of functions aimed at math programming.

Furthermore, OpenCL defines an application programming interface (API) which allows programs executing on the host to launch kernels on the compute devices and to manage the device memory. Devices may or may not share memory with the host CPU. The API provides handles on device memory buffers and functions to transfer data back and forth between the host and the compute devices. The OpenCL standard defines APIs for C and C++. However, third-party APIs exist for other programming languages such as Python or Java. OpenCL C is intended to be compiled at run-time, so that applications are portable between implementations for various devices. This means, however, an OpenCL application has to be shipped with the kernel codes in plain text format, which is a problem in the case of proprietary software.

REFERENCES

- [1] Kroll N, Jain RK. Solution of two-dimensional Euler equations—experience with a finite volume code. DFVLR-FB 87-41; 1987.
- [2] Mavriplis DJ. On convergence acceleration techniques for unstructured meshes. AIAA Paper 98-2966; 1998.
- [3] Jameson A. Iterative solution of transonic flow over airfoil and wings including at Mach 1. Commun Pure Appl Math 1974;27:283-309.
- [4] Jameson A, Schmidt W, Turkel E. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. AIAA Paper 81-1259; 1981.

- [5] Van Leer B, Tai CH, Powell KG. Design of optimally smoothing multi-stage schemes for the Euler equations. AIAA Paper 89-1933; 1989.
- [6] Tai C-H, Sheu J-H, van Leer B. Optimal multistage schemes for Euler equations with residual smoothing. AIAA J 1995;33:1008-16.
- [7] Tai C-H, Sheu J-H, Tzeng P-Y. Improvement of explicit multistage schemes for central spatial discretization. AIAA J 1996;34:185-8.
- [8] Jameson A, Baker TJ. Solution of the Euler equations for complex configurations. AIAA Paper 83-1929; 1983.
- [9] Enander R, Karlsson AR. Implicit explicit residual smoothing in multigrid cycle. AIAA Paper 95-0204; 1995.
- [10] Enander R, Sjögren B. Implicit explicit residual smoothing for upwind schemes. Uppsala University, Report No. 179; 1996.
- [11] Martinelli L. Calculation of viscous flows with a multigrid method. PhD Thesis, Dept. of Mechanical and Aerospace Eng. Princeton University; 1987.
- [12] Turkel E, Swanson RC, Vatsa VN, White JA. Multigrid for hypersonic viscous two- and three-dimensional flows. AIAA Paper 91-1572; 1991.
- [13] Radespiel R, Kroll N. Multigrid schemes with semicoarsening for accurate computations of hypersonic viscous flows. DLR Internal Report, No. 129-90/19; 1991.
- [14] Jameson A, Baker TJ, Weatherill NP. Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper 86-0103; 1986.
- [15] Blazek J, Kroll N, Radespiel R, Rossow C-C. Upwind implicit residual smoothing method for multi-stage schemes. AIAA Paper 91-1533; 1991.
- [16] Blazek J. Methods to accelerate the solutions of the Euler- and Navier-Stokes equations for steady-state super- and hypersonic flows. Translation of DLR Research Report 94-35, ESA-TT-1331; 1995.
- [17] Blazek J, Kroll N, Rossow C-C. A comparison of several implicit smoothing methods. In: ICFD conf. numerical methods for fluid dynamics. Reading, England; 1992.
- [18] Blazek J, Kroll N, Rossow C-C, Swanson RC. A comparison of several implicit residual smoothing methods in combination with multigrid. In: 13th int. conf. on numerical methods in fluid dynamics. Roma, Italy; 1992.
- [19] Brandt A. Guide to multigrid development. Multigrid methods I. Lecture Notes in Mathematics No. 960. New York: Springer Verlag; 1981.
- [20] Jameson A. Solution of the Euler equations for two-dimensional transonic flow by a multigrid method. MAE Report No. 1613, Dept. of Mechanical and Aerospace Engineering. Princeton University; 1983.
- [21] Jameson A. Solution of the Euler equations by a multigrid method. Appl Math Comput 1983;13:327-56.
- [22] Jameson A. Multigrid algorithms for compressible flow calculations. Multigrid Methods II, Lecture Notes in Mathematics No. 1228. New York: Springer Verlag; 1985.
- [23] Schröder W, Hänel D. An unfactored implicit scheme with multigrid acceleration for the solution of the Navier-Stokes equations. Comput Fluids 1987;15:313-20.
- [24] Koren B. Multigrid and defect correction for the steady Navier-Stokes equations. J Comput Phys 1990;87:25-46.
- [25] Thomas JL. An implicit multigrid scheme for hypersonic strong-interaction flowfields. In: 5th Copper Mountain conf. on multigrid methods. Denver, USA; 1991.
- [26] Radespiel R, Swanson R.C. Progress with multigrid schemes for hypersonic flow problems. ICASE Report, No. 91-89; 1991 (also J Comput Phys 1995;116:103-22).
- [27] Mavriplis DJ, Venkatakrishnan V. A unified multigrid solver for the Navier-Stokes equations on mixed element meshes. ICASE Report No. 95-53; 1995.
- [28] Mavriplis DJ, Venkatakrishnan V. A 3D agglomeration multigrid solver for the Reynolds-averaged Navier-Stokes equations on unstructured meshes. Int J Numer Meth Fluids 1996;23:527-44.
- [29] Braaten ME, Connell SD. Three-dimensional unstructured adaptive multigrid scheme for the Navier-Stokes equations. AIAA J 1996;34:281-90.

- [30] Mavriplis DJ. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *J Comput Phys* 1998;145:141-65.
- [31] Haselbacher AC. A grid-transparent numerical method for compressible viscous flows on mixed unstructured grids. PhD Thesis, Loughborough University, England; 1999.
- [32] Wasserman M, Mor-Yossef Y, Yavneh I, Greenberg JB. A robust implicit multigrid method for RANS equations with two-equation turbulence models. *J Comput Phys* 2010;229:5820-42.
- [33] Yoon S, Jameson A. A multigrid LU-SSOR scheme for approximate Newton-iteration applied to the Euler equations. NASA-CR-17954; 1986.
- [34] Yoon S, Jameson A. Lower-upper implicit schemes with multiple grids for the Euler equations. AIAA Paper 86-0105; 1986 (also AIAA J 1987;7:929-35).
- [35] Yoon S, Kwak D. Multigrid convergence of an implicit symmetric relaxation scheme. AIAA Paper 93-3357; 1993.
- [36] Blazek J. Investigations of the implicit LU-SSOR scheme. DLR Research Report No. 93-51; 1993.
- [37] Blazek J. A multigrid LU-SSOR scheme for the solution of hypersonic flow problems. AIAA Paper 94-0062; 1994.
- [38] Gerlinger P, Stoll P, Brüggemann D. An implicit multigrid method for the simulation of chemically reacting flows. *J Comput Phys* 1998;146:322-45.
- [39] Langer S. Agglomeration multigrid methods with implicit Runge-Kutta smoothers applied to aerodynamic simulations on unstructured grids. *J Comput Phys* 2014;277:72-100.
- [40] Roberts TW, Sidilkover D, Swanson RC. Textbook multigrid efficiency for the steady Euler equations. AIAA Paper 97-1949; 1997.
- [41] Thomas JL, Bonhaus DL, Anderson WK, Rumsey CL, Biedron RT. An $\mathcal{O}(Nm^2)$ plane solver for the compressible Navier-Stokes equations. AIAA Paper 99-0785; 1999.
- [42] van Leer B, Darmofal D. Steady Euler solutions in $O(N)$ operations. In: Proc. sixth European multigrid conf. Gent, Belgium; September 27-30, 1999 (Berlin: Springer; 2000).
- [43] Nishikawa H, van Leer B. Optimal multigrid convergence by elliptic/hyperbolic splitting. AIAA Paper 2002-2951; 2002 (also *J Comput Phys* 2003;190:52-63).
- [44] Ruge JW, Stüben K. Algebraic multigrid. Multigrid methods. In: McCormick SF, editor. SIAM frontiers in applied mathematics. Philadelphia (PA): SIAM; 1987. p. 73-131.
- [45] Lonsdale RD. An algebraic multigrid solver for the Navier-Stokes equations on unstructured meshes. *Int J Numer Meth Heat Fluid Flow* 1993;3:3-14.
- [46] Webster R. An algebraic multigrid solver for Navier-Stokes problems. *Int J Numer Meth Heat Fluid Flow* 1994;18:761-80.
- [47] Raw MJ. Robustness of coupled algebraic multigrid for the Navier-Stokes equations. AIAA Paper 96-0297; 1996.
- [48] Weiss JM, Maruszewski JP, Smith WA. Implicit solution of preconditioned Navier-Stokes equations using algebraic multigrid. *AIAA J* 1999;37:29-36.
- [49] Cleary AJ, Falgout RD, Van Henson E, Jones JE, Manteuffel TA, McCormick SF, et al. Robustness and scalability of algebraic multigrid. *SIAM J Sci Comput* 2000;21:1886-908.
- [50] Stüben K. A review of algebraic multigrid. *J Comput Applied Math* 2001;128:281-309.
- [51] Van Henson E, Yang UM. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Appl Numer Math* 2002;41:155-77.
- [52] Haase G, Kuhn M, Reitzinger S. Parallel algebraic multigrid methods on distributed memory computers. *SIAM J Sci Comput* 2002;24:410-27.
- [53] Chang Q, Huang Z. Efficient algebraic multigrid algorithms and their convergence. *SIAM J Sci Comput* 2002;24:597-618.
- [54] Mulder WA. A new multigrid approach to convection problems. *J Comput Phys* 1989;83:303-23.
- [55] Lynn JF, van Leer B. A semi-coarsened multigrid solver for the Euler and Navier-Stokes equations with local preconditioning. AIAA Paper 95-1667; 1995.
- [56] Mavriplis DJ. Directional agglomeration multigrid techniques for high Reynolds number viscous flow solvers. AIAA Paper 98-0612; 1998.
- [57] Mavriplis DJ. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *J Comput Phys* 1998;145:141-65.

- [58] Hackbusch W. Multigrid methods and applications. Berlin: Springer Verlag; 1985.
- [59] Mavriplis DJ. Multigrid techniques for unstructured meshes. ICASE Report No. 95-27; 1995.
- [60] Mavriplis DJ. Unstructured grid technique. Annu Rev Fluid Mech 1997;29:473-514.
- [61] Leclercq MP, Stoufflet B. Characteristic multigrid method application to solve the Euler equations with unstructured and unnested grids. J Comput Phys 1993;104:329-46.
- [62] Mavriplis DJ. Three-dimensional multigrid for the Euler equations. AIAA J 1992;30:1753-61.
- [63] Peraire J, Peiró J, Morgan K. A 3D finite-element multigrid solver for the Euler equations. AIAA Paper 92-0449; 1992.
- [64] Morano E, Dervieux A. Looking for O(N) Navier-Stokes solutions on non-structured meshes. In: Proc. 6th Copper Mountain conf. on multigrid methods; 1993. p. 449-64 (also ICASE Report 93-26; 1993).
- [65] Riemslagh K, Dick E. A multigrid method for steady Euler equations on unstructured adaptive grids. In: Proc. 6th Copper Mountain conf. on multigrid methods; 1993. p. 527-42.
- [66] Mavriplis DJ, Martinelli L. Multigrid solution of compressible turbulent flow on unstructured meshes using a two-equation model. Int J Numer Meth Fluids 1994;18:887-914.
- [67] Crumpton PI, Giles MB. Aircraft computations using multigrid and an unstructured parallel library. AIAA Paper 95-0210; 1995.
- [68] Guillard H. Node-nested multigrid with Delaunay coarsening. INRIA Report No. 1898, 1993.
- [69] Ollivier-Gooch CF. Multigrid acceleration of an upwind Euler solver on unstructured meshes. AIAA J 1995;33:1822-7.
- [70] Parthasarathy V, Kallinderis Y. New multigrid approach for three-dimensional unstructured, adaptive grids. AIAA J 1994;32:956-63.
- [71] Barth TJ. Randomized multigrid. AIAA Paper 95-0207; 1995.
- [72] Lawson CL. Software for C^1 surface interpolation. In: Rice JR, editor. Mathematical software III. New York: Academic Press; 1977.
- [73] Lawson CL. Properties of n-dimensional triangulations. Comput Aided Geomet Design 1986;3:231-46.
- [74] Crumpton PI, Giles MB. Implicit time accurate solutions on unstructured dynamic grids. AIAA Paper 95-1671; 1995.
- [75] Moinier P, Müller J-D, Giles MB. Edge-based multigrid for hybrid grids. AIAA Paper 99-3339; 1999.
- [76] Lallemand MH. Schémas décentrés multigrilles pour la résolution des équations d'Euler en éléments finis. PhD Thesis, Université de Provence, France; 1988.
- [77] Lallemand MH, Steve H, Dervieux A. Unstructured multigrid by volume agglomeration: current status. Comput Fluids 1992;21:397-433.
- [78] Koobus B, Lallemand MH, Dervieux A. Unstructured volume-agglomeration multigrid: solution of the Poisson equation. Int J Numer Meth Fluids 1994;18:27-42.
- [79] Smith WA. Multigrid solution of transonic flow on unstructured grids. In: Recent advances and applications in CFD, Proc. ASME winter annual meeting; December 1990. 145-52.
- [80] Venkatakrishnan V, Mavriplis DJ. Agglomeration multigrid for the three-dimensional Euler equations. AIAA J 1995;33:633-40.
- [81] Mavriplis DJ, Venkatakrishnan V. Agglomeration multigrid for viscous turbulent flows. ICASE Report No. 94-62, 1994 (also AIAA Paper 94-2332; 1994).
- [82] Elias SR, Stuble GD, Raithby GD. An adaptive agglomeration method for additive correction multigrid. Int J Numer Meth Eng 1997;40:887-903.
- [83] Okamoto N, Nakahashi K, Obayashi S. A coarse grid generation algorithm for agglomeration multigrid method on unstructured grids. AIAA Paper 98-0615; 1998.
- [84] Lassaline JV, Zingg DW. Development of an agglomeration multigrid algorithm with directional coarsening. AIAA Paper 99-3338; 1999.
- [85] Nishikawa H, Diskin B, Thomas JL. Critical study of agglomerated multigrid methods for diffusion. AIAA J 2010;48:839-47.
- [86] Thomas JL, Diskin B, Nishikawa H. A critical study of agglomerated multigrid methods for diffusion on highly-stretched grids. Comput Fluids 2011;41:82-93.

- [87] Nishikawa H, Diskin B, Thomas JL, Hammond DH. Recent advances in agglomerated multigrid. AIAA Paper 2013-863; 2013.
- [88] Moulitsas I, Karypis G. Multilevel algorithms for generating coarse grids for multigrid methods. In: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing. November 10-16, 2001, Denver, CO, USA. p. 45-55.
- [89] <http://www-users.cs.umn.edu/~moulitsa/software.html>
- [90] Volpe G. Performance of compressible flow codes at low Mach number. *AIAA J* 1993;31:49-56.
- [91] Lee WT. Local preconditioning of the Euler equations. PhD Thesis, University of Michigan; 1991.
- [92] Lee D. Local preconditioning of the Euler and Navier-Stokes equations. PhD Thesis, University of Michigan; 1996.
- [93] Guillard H, Viozat C. On the behavior of upwind schemes in the low Mach number limit. INRIA Report No. 3160; 1997.
- [94] Patankar SV. Numerical heat transfer and fluid flow. McGraw-Hill, New York; 1980.
- [95] Ho Y-H, Lakshminarayana B. Computation of unsteady viscous flow using a pressure-based algorithm. *AIAA J* 1993;31:2232-40.
- [96] Javadia K, Darbandia M, Taeibi-Rahni M. Three-dimensional compressible-incompressible turbulent flow simulation using a pressure-based algorithm. *Comput Fluids* 2008;37:747-66.
- [97] Darwish M, Sraj I, Moukalled F. A coupled finite volume solver for the solution of incompressible flows on unstructured grids. *J Comput Phys* 2009;228:180-201.
- [98] Shterev KS, Stefanov SK. Pressure based finite volume method for calculation of compressible viscous gas flows. *J Comput Phys* 2010;229:461-80.
- [99] Chen ZJ, Przekwas AJ. A coupled pressure-based computational method for incompressible/compressible flows. *J Comput Phys* 2010;229:9150-65.
- [100] Chorin AJ. A numerical method for solving incompressible viscous flow problems. *J Comput Phys* 1967;2:12-26.
- [101] Turkel E. Preconditioned methods for solving the incompressible and low speed compressible equations. *J Comput Phys* 1987;72:277-98.
- [102] Rogers SE, Kwak D. Upwind differencing scheme for the time-accurate incompressible Navier-Stokes equations. *AIAA J* 1990;28:253-62.
- [103] Cabuk H, Sung C-H, Modi V. Explicit Runge-Kutta method for three-dimensional internal incompressible flows. *AIAA J* 1992;30:2024-31.
- [104] Shi J, Toro EF. A Riemann-problem-based approach for steady incompressible flows. *Int J Numer Meth Heat Fluid Flow* 1996;6:81-93.
- [105] Liu C, Zheng X, Sung CH. Preconditioned multigrid methods for unsteady incompressible flows. *J Comput Phys* 1998;139:35-57.
- [106] Briley WR, Taylor LK, Whitfield DL. High-resolution viscous flow simulations at arbitrary Mach number. *J Comput Phys* 2003;184:79-105.
- [107] Turkel E, Radespiel R, Kroll N. Assessment of preconditioning methods for multidimensional aerodynamics. *Comput Fluids* 1997;26:613-34.
- [108] Rieper F. On the dissipation mechanism of upwind-schemes in the low Mach number regime: a comparison between Roe and HLL. *J Comput Phys* 2010;229:221-32.
- [109] Rieper F. A low-Mach number fix for Roe's approximate Riemann solver. *J Comput Phys* 2011;230:5263-87.
- [110] Mulas M, Chibbaro S, Delussu G, Di Piazza I, Talice M. Efficient parallel computations of flows of arbitrary fluids for all regimes of Reynolds, Mach and Grashof numbers. *Int J Numer Meth Heat Fluid Flow* 2002;12:637-57.
- [111] Van Leer B, Lee WT, Roe P. Characteristic time-stepping or local preconditioning of the Euler equations. AIAA Paper 91-1552; 1991.
- [112] Choi YH, Merkle CL. Time-derivative preconditioning for viscous flows. AIAA Paper 91-1652; 1991.
- [113] Choi YH, Merkle CL. The application of preconditioning in viscous flows. *J Comput Phys* 1993;105:207-23.

- [114] Turkel E. A review of preconditioning methods for fluid dynamics. *Appl Numer Math* 1993;12:257-84.
- [115] Turkel E, Vatsa VN, Radespiel R. Preconditioning methods for low-speed flows. *AIAA Paper* 96-2460; 1996.
- [116] Turkel E. Preconditioning techniques in computational fluid dynamics. *Annu Rev Fluid Mech* 1999;31:385-416.
- [117] Lee D, van Leer B. Progress in local preconditioning of the Euler and Navier-Stokes equations. *AIAA Paper* 93-3328; 1993.
- [118] Lee D, van Leer B, Lynn JF. A local Navier-Stokes preconditioner for all Mach and cell Reynolds numbers. *AIAA Paper* 97-2024; 1997.
- [119] Jorgenson PC, Pletcher RH. An implicit numerical scheme for the simulation of internal viscous flow on unstructured grids. *Comput Fluids* 1996;25:447-66.
- [120] Weiss J, Smith WA. Preconditioning applied to variable and constant density flows. *AIAA J* 1995;33:2050-7.
- [121] Godfrey AG, Walters RW, van Leer B. Preconditioning for the Navier-Stokes equations with finite-rate chemistry. *AIAA Paper* 93-0535; 1993.
- [122] Dailey LD, Pletcher RH. Evaluation of multigrid acceleration for preconditioned time-accurate Navier-Stokes algorithms. *Comput Fluids* 1996;25:791-811.
- [123] Jespersen D, Pulliam T, Buning P. Recent enhancements to OVERFLOW. *AIAA Paper* 97-0644; 1997.
- [124] Sharov D, Nakahashi K. Low speed preconditioning and LU-SGS scheme for 3-D viscous flow computations on unstructured grids. *AIAA Paper* 98-0614; 1998.
- [125] Nemec M, Zingg DW. Aerodynamic computations using the convective upstream split pressure scheme with local preconditioning. *AIAA Paper* 98-2444; 1998.
- [126] Merkle CL, Sullivan JY, Buelow PEO, Venkateswaran S. Computation of flows with arbitrary equations of state. *AIAA J* 1998;36:515-21.
- [127] Sockol PM. Multigrid solution of the Navier-Stokes equations at low speeds with large temperature variations. *J Comput Phys* 2003;192:570-92.
- [128] Puoti V. Preconditioning method for low-speed flows. *AIAA J* 2003;41:817-30.
- [129] Lee S-H. Cancellation problem of preconditioning method at low Mach numbers. *J Comput Phys* 2007;225:1199-210.
- [130] Lee S-H. Alleviation of cancellation problem of preconditioned Navier-Stokes equations. *J Comput Phys* 2009;228:4970-5.
- [131] Park H, Nourgaliev RR, Martineau RC, Knoll DA. On physics-based preconditioning of the Navier-Stokes equations. *J Comput Phys* 2009;228:9131-46.
- [132] De Medeiros FEL, De B. Alves LS. Stiffness, sensitiveness and robustness in low Mach preconditioned density-based methods. *AIAA Paper* 2012-1113; 2012.
- [133] Hejranfar K, Kamali-Moghadam R. Preconditioned characteristic boundary conditions for solution of the preconditioned Euler equations at low Mach number flows. *J Comput Phys* 2012;231:4384-402.
- [134] Bas O, Tuncer IH, Kaynak U. A Mach-uniform preconditioner for incompressible and subsonic flows. *Int J Numer Meth Fluids* 2014;74:100-12.
- [135] Falcão CEG, De Medeiros FEL, De B. Alves LS. Implicit Runge-Kutta physical-time marching in low mach preconditioned density-based methods. *AIAA Paper* 2014-3085; 2014.
- [136] <http://msdn.microsoft.com/en-us/library/dd492627.aspx>
- [137] <http://openacc.org/>
- [138] <http://blogs.msdn.com/b/nativeconcurrency>
- [139] <http://www.open-mpi.org>
- [140] <https://computing.llnl.gov/tutorials/mpi>
- [141] <http://www.cs.umn.edu/~metis>
- [142] Stoll P, Gerlinger P, Brüggemann D. Domain decomposition for an implicit LU-SGS scheme using overlapping grids. *AIAA Paper* 97-1896; 1997.

- [143] Gropp WD, Keyes DE, McInnes LC, Tidriri MD. Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD. ICASE Report No. 98-24; 1998.
- [144] Cai X-C, Sarkis M. A restricted additive Schwarz preconditioner for general sparse linear systems. SIAM J Sci Comput 1999;21:792-7.
- [145] Knoll DA, Keyes DE. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. J Comput Phys 2004; 193:357-97.
- [146] Chen R, Wu Y, Yan Z, Zhao Y, Cai X-C. A parallel domain decomposition method for 3D unsteady incompressible flows at high Reynolds number. SIAM J Sci Comput 2014;58:275-89.
- [147] <http://www.mcs.anl.gov/petsc>
- [148] <http://www.openmp.org>
- [149] Amritkar A, Deb S, Tafti D. Efficient parallel CFD-DEM simulations using OpenMP. J Comput Phys 2014;256:501-19.
- [150] http://www.nvidia.com/object/cuda_home_new.html
- [151] <https://developer.nvidia.com/cusp>
- [152] <https://github.com/cusplibrary/cusplibrary>
- [153] Salvadore F, Bernardini M, Botti M. GPU accelerated flow solver for direct numerical simulation of turbulent flows. J Comput Phys 2013; 235:129-42.
- [154] <http://www.khronos.org/opencl>

CHAPTER 10

Consistency, Accuracy, and Stability

Contents

10.1	Consistency Requirements	338
10.2	Accuracy of Discretization Scheme	339
10.3	Von Neumann Stability Analysis	340
10.3.1	Fourier symbol and amplification factor	340
10.3.2	Convection model equation	342
<i>Central scheme with artificial dissipation</i>		342
<i>Upwind scheme</i>		342
10.3.3	Convection-diffusion model equation	342
10.3.4	Explicit time-stepping	343
<i>Examples of Fourier symbols and amplification factors</i>		345
10.3.5	Implicit time-stepping	349
<i>Examples of amplification factors</i>		350
10.3.6	Derivation of the CFL condition	353
<i>CFL condition by von Neumann analysis</i>		354
	References	355

Within the finite-volume methodology, the surface integral in the Navier-Stokes equations (2.19) is approximated for each control volume by some appropriate method. We considered various possible spatial discretizations in Chapters 4 and 5. The approximation of the convective and viscous fluxes across the boundaries of the control volume means that the discretized equations differ from the governing equations, which we seek to solve. This difference causes a certain spatial *discretization error*. The error is influenced by the numerical scheme applied for the evaluation of the integrals and by the resolution of the discretization. Therefore, the important question is whether and how fast the solution of the discretized equations converges to the exact solution of the governing equations with increasingly finer grid. We shall discuss this question in the next two sections on the consistency and accuracy.

The solution of the time-dependent governing equations (2.19) requires also the discretization of the time derivative of the conservative variables. We described several temporal discretization methods in Chapter 6. We mentioned that each of the schemes has a specific order of accuracy and that there are certain limitations on the maximum size of the time step. These limitations can be assessed by means of the von Neumann stability analysis, which will be presented in [Section 10.3](#). In this section, we shall also investigate the ability of the explicit multistage time-stepping scheme and of a generic

implicit scheme to damp errors of the discrete solution in time. The damping properties decide about the convergence speed and the robustness of a particular scheme. They are also very important for the success or failure of the multigrid acceleration (presented in Section 9.4).

10.1 CONSISTENCY REQUIREMENTS

A discretization scheme is called *consistent*, if the discretized equations converge to the given differential equations for both the time step and grid size tending to zero. A consistent scheme gives us the security that we really solve the governing equations and nothing else. This is quoted in Ref. [1] as “solving the equations right” and it is called *verification*. Verification should not be confused with the term *validation*. Validation means: do we solve “the right equations”? Thus, verification tries to quantify the *numerical* errors and uncover programming bugs, whereas validation deals with the *modeling* errors. For an exhaustive discussion and recommendations see the guidelines presented in Ref. [2].

The consistency of a numerical scheme can be checked by expanding the function values into Taylor series. The developments are then inserted back into the discretized equations. If we subtract the differential equations, we obtain terms that represent the numerical error—the so-called *truncation error*. For a consistent scheme, the truncation error should go to zero with decreasing time step and grid size.

We will now illustrate this concept on a simple example. Let us consider the following 1-D scalar equation

$$\frac{\partial U}{\partial t} + \frac{\partial U}{\partial x} = 0. \quad (10.1)$$

A very simple but valid discretization scheme would be

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \frac{U_i^n - U_{i-1}^n}{\Delta x} = 0, \quad (10.2)$$

where n denotes the time level and i the node index, respectively. Expanding the solution U_i^{n+1} around the time level n gives

$$U_i^{n+1} = U_i^n + \Delta t \left(\frac{\partial U}{\partial t} \right)_i^n + \frac{(\Delta t)^2}{2} \left(\frac{\partial^2 U}{\partial t^2} \right)_i^n + \dots \quad (10.3)$$

where (\dots) represents the higher-order terms. The Taylor series for the solution U_{i-1}^n reads (cf. Eq. (3.1))

$$U_{i-1}^n = U_i^n - \Delta x \left(\frac{\partial U}{\partial x} \right)_i^n + \frac{(\Delta x)^2}{2} \left(\frac{\partial^2 U}{\partial x^2} \right)_i^n + \dots \quad (10.4)$$

If we substitute Eqs. (10.3) and (10.4) into the discretized equation (10.2), we obtain

$$\left(\frac{\partial U}{\partial t} + \frac{\partial U}{\partial x} \right)_i^n = -\frac{\Delta t}{2} \left(\frac{\partial^2 U}{\partial t^2} \right)_i^n + \frac{\Delta x}{2} \left(\frac{\partial^2 U}{\partial x^2} \right)_i^n + \dots \quad (10.5)$$

A comparison with the differential equation (10.1) shows that the terms on the right-hand side of Eq. (10.5) represent the truncation error, which is of the order $\mathcal{O}(\Delta t, \Delta x)$. The numerical scheme (10.2) is consistent, since the truncation error vanishes for $\Delta t \rightarrow 0, \Delta x \rightarrow 0$.

10.2 ACCURACY OF DISCRETIZATION SCHEME

The accuracy of a discretization scheme is connected to its truncation error. If, for instance, the leading term of the truncation error is proportional to Δx , we speak of *first-order* accurate spatial scheme. If the leading term behaves like $(\Delta x)^2$, the scheme is second-order accurate, and so on. Thus, the example scheme Eq. (10.2) is first-order accurate in space and in time (Eq. (10.5)). This leads us to the condition that the numerical scheme must be at least first-order accurate in order to be consistent. Otherwise, the truncation error cannot be reduced by decreasing the values of Δt and Δx .

The question is now, how can we assess the truncation error in practice. Certainly, the expansion into the Taylor series is not adequate for complex numerical schemes with nonlinear switches, limiters, etc. The following are the possible ways of error estimation [1]:

- additional solution(s) on different grid(s)—grid refinement or coarsening, unrelated grid(s);
- additional solution(s) on the same grid—higher- or lower-order accurate discretization;
- solution of an auxiliary partial-differential equation on the same grid;
- algebraic evaluations on the same grid.

The most common approach is to solve the governing equations on a series of grids with different cell sizes. If we happen to know the exact solution, we can easily quantify the error for each grid. The rate by which the truncation error decreases determines the accuracy of the discretization. For example, if we halve the grid size in all coordinate directions and the error drops by a factor of four, the scheme is second-order accurate. However, the exact physical solution is quite often not known. In such a case, the order of accuracy can be estimated using the formula [3]

$$p = \frac{\ln \left(\frac{f_3 - f_2}{f_2 - f_1} \right)}{\ln(r)}, \quad (10.6)$$

where p denotes the accuracy, r the refinement (coarsening) ratio, and f the numerical solution, respectively. Index 1 denotes the finest grid and index 3 the coarsest grid. If the coarsening ratio is not constant between the grids, a more general relation can be found in Ref. [1].

Besides the estimation of the truncation error, varying the grid resolution is also used to obtain what is called *grid converged solution*. This is achieved if the solution does not change (within a certain tolerance) with further grid refinement. In this connection, we speak of *grid convergence studies*. Although grid convergence studies can be very time consuming, it is recommended always to check if the solution is grid converged.

10.3 VON NEUMANN STABILITY ANALYSIS

Before a new discretization method is implemented, it is important to know, at least approximately, how the method will influence the stability and the convergence behavior of the numerical scheme. It was already frequently confirmed that the von Neumann method of stability analysis can deliver reliable assessment of the properties of a solution scheme. The methodology was developed at Los Alamos during the Second World War. However, it was briefly described first in 1947 by Cranck and Nicholson [4]. Later, it was also published in Ref. [5]. A very helpful introduction to the von Neumann stability analysis can be found in Ref. [6].

Von Neumann stability analysis is applicable to discretized *linear* partial-differential equations under the assumption of periodic boundary conditions. It is based on the decomposition of the solution into a Fourier series. On one hand, this allows the investigation of the stability of a solution scheme. On the other hand, the behavior of the solution across the frequency spectrum can be examined in detail. Precisely this is of fundamental importance for the estimation of the convergence properties and robustness of a scheme, since the individual components (Fourier modes) of the solution error have to be reduced (damped) as quickly as possible. Therefore, we speak of the *damping properties* of a solution scheme.

Because the von Neumann analysis is limited to linear problems, the Euler or the Navier-Stokes equations have to be substituted with a suitable model equation. Two often employed 1-D model problems will be presented further below. The first one describes pure convection of a disturbance, which models the behavior of the Euler equations. The second model equation contains additionally a diffusion term in order to simulate the behavior of the Navier-Stokes equations.

10.3.1 Fourier symbol and amplification factor

Before we proceed with the model problems, let us examine the von Neumann analysis for a general 1-D scalar linear equation. After the spatial discretization of the fluxes, we obtain a system of ordinary differential equations in time (cf. Eq. (4.3))

$$\Delta t \left(\frac{d}{dt} U_i \right) = -\frac{\Delta t}{\Delta x} R_i, \quad (10.7)$$

where U_i denotes a scalar variable at point i , R_i stands for the residual, and Δx represents the grid size. Assuming periodic boundary conditions, we can expand the solution U in Eq. (10.7) into a finite Fourier series [6–8]

$$U_i = \sum_{k=-N}^N \hat{U}_k e^{I p_k (i \Delta x)} \quad (10.8)$$

with the point index i running from 0 to N ($x_i = i \Delta x$), p_k being the wave number and I the imaginary unit. Due to the linearity of the model equation, it is sufficient to consider only an individual (l th) Fourier mode, i.e.,

$$\hat{U} e^{I p_l (i \Delta x)} \quad (10.9)$$

since the entire influence can be obtained by superposition. If we now insert the Fourier mode Eq. (10.9) into the discretized model equation (10.7), we get

$$\Delta t \frac{d}{dt} (\hat{U} e^{I p_l (i \Delta x)}) = -\frac{\Delta t}{\Delta x} R_i = -\frac{\Delta t}{\Delta x} z \hat{U} e^{I l \Phi} \quad (10.10)$$

with the definition of the *phase angle*: $\Phi = p_l \Delta x$. The complex function z in Eq. (10.10) represents the so-called *Fourier symbol* of the spatial operator. Its form depends on the discretization type (central, upwind) and on the order of accuracy.

An explicit or implicit time-stepping scheme used to solve Eq. (10.7) will be linearly stable, if the amplitude of any harmonic (\hat{U}) does not grow in time. This means that the amplitude of the new solution must be equal or smaller than the amplitude of the previous solution. Hence, the *amplification factor* is defined as

$$g = \frac{\hat{U}^{n+1}}{\hat{U}^n}. \quad (10.11)$$

If we introduce the *time-stepping operator* f , we can write the amplification factor in the general form as

$$g = 1 - fz. \quad (10.12)$$

The time-stepping scheme will be linearly stable if $|g| \leq 1$. In cases, where the spatial and temporal discretizations are separated (method of lines), the domain of stability depends on the time-stepping scheme (f), but not on the spatial discretization (z). The Fourier symbol of the spatial operator z must lie within the domain of stability for all phase angles Φ . We speak of good damping properties if $|g|$ is well below unity. This means that the perturbations of the numerical solution are rapidly damped in time. Consequently, the time-stepping scheme converges faster to the steady-state solution than for $|g| \rightarrow 1$.

10.3.2 Convection model equation

In this case, the scalar linear model equation reads

$$\frac{\partial U}{\partial t} + \Lambda \frac{\partial U}{\partial x} = 0. \quad (10.13)$$

The convection velocity Λ (also the eigenvalue) is assumed to be constant and positive. In the following, we shall consider different spatial discretization schemes applied to Eq. (10.7) and their corresponding Fourier symbols.

Central scheme with artificial dissipation

According to Eqs. (4.48)–(4.50), the residual R_i in Eq. (10.7) takes the form

$$R_i = \frac{\Lambda}{2}(U_{i+1} - U_{i-1}) + \Lambda \epsilon^{(4)}(U_{i+2} - 4U_{i+1} + 6U_i - 4U_{i-1} + U_{i-2}) \quad (10.14)$$

with $\epsilon^{(4)}$ denoting the dissipation coefficient. In order to simplify the analysis, only the fourth-order differences were retained in Eq. (10.14). The Fourier symbol of the spatial operator is obtained according to Eq. (10.10) by inserting the harmonic Eq. (10.9) into Eq. (10.14)

$$z = \Lambda \left[I \sin \Phi + 4\epsilon^{(4)}(1 - \cos \Phi)^2 \right]. \quad (10.15)$$

Upwind scheme

Using the first-order upwind scheme (see Eq. (4.46)), the residual R_i becomes

$$R_i = \Lambda(U_i - U_{i-1}) \quad (10.16)$$

and the associated Fourier symbol reads

$$z = \Lambda [I \sin \Phi + (1 - \cos \Phi)]. \quad (10.17)$$

In the case of the second-order upwind spatial discretization, the residual in Eq. (10.7) is given by

$$R_i = \frac{\Lambda}{2}(3U_i - 4U_{i-1} + U_{i-2}). \quad (10.18)$$

The corresponding Fourier symbols z of the spatial operator appear as

$$z = \Lambda \left[I \sin \Phi (2 - \cos \Phi) + (1 - \cos \Phi)^2 \right]. \quad (10.19)$$

10.3.3 Convection-diffusion model equation

The combined convection-diffusion model equation can be written in the form

$$\frac{\partial U}{\partial t} + \Lambda \frac{\partial U}{\partial x} = \nu \frac{\partial^2 U}{\partial x^2}, \quad (10.20)$$

where ν represent the viscosity coefficient. The diffusion term $\partial^2 U / \partial x^2$ is normally approximated by second-order central differences. Thus,

$$\nu \frac{\partial^2 U}{\partial x^2} \approx \frac{\Lambda_\nu}{\Delta x} (U_{i+1} - 2U_i + U_{i-1}) \quad (10.21)$$

with

$$\Lambda_\nu = \frac{\nu}{\Delta x} \quad (10.22)$$

being the viscous eigenvalue.

The Fourier symbol of the spatial operator is now composed of the convective and the diffusive part, i.e.,

$$z = z_c - z_\nu, \quad (10.23)$$

where

$$z_\nu = 2\Lambda_\nu(\cos \Phi - 1) \quad (10.24)$$

and z_c corresponds to one of the forms presented in Eq. (10.15), (10.17), or (10.19), respectively.

As we can conclude from Eq. (10.24), the diffusion term changes only the real part of the Fourier symbol. This is typical for any kind of diffusion or artificial dissipation (cf. Eq. (10.15)). Furthermore, we can see from Eq. (10.17) or (10.19) that the upwind discretization of the convective term also causes the Fourier symbol to have a real part ($1 - \cos \Phi$). Therefore, we speak of upwind dissipation. Only the central discretization of the convective term adds no numerical dissipation, since the Fourier symbol is given by ($I \sin \Phi$). However, the central scheme allows for the unwanted odd-even decoupling of the solution.

10.3.4 Explicit time-stepping

The application of an m -stage explicit time-stepping scheme to the discretized model problem Eq. (10.7) can be described in the following way (see Section 6.1.1)

$$\begin{aligned} U_i^{(0)} &= U_i^n, \\ U_i^{(k)} &= U_i^{(0)} - \alpha_k \frac{\Delta t}{\Delta x} R_i^{(k-1)}, \quad k = 1, \dots, m, \\ U_i^{n+1} &= U_i^{(m)}, \end{aligned} \quad (10.25)$$

where α_k denote the stage coefficients. If we substitute the variable U by its Fourier representation Eq. (10.8) and the residual by the Fourier symbol z , Eq. (10.25) transforms to

$$\begin{aligned}\hat{U}^{(0)} &= \hat{U}^n, \\ \hat{U}^{(k)} &= \hat{U}^{(0)} - \alpha_k \frac{\Delta t}{\Delta x} z \hat{U}^{(k-1)}, \quad k = 1, \dots, m, \\ \hat{U}^{n+1} &= \hat{U}^{(m)}.\end{aligned}\tag{10.26}$$

The amplification factor of the above m -stage explicit scheme is given by Eq. (10.12). Based on Eq. (10.26), it can be shown that the Fourier symbol of the time-stepping operator f resumes the form [9]

$$\begin{aligned}f &= \frac{\Delta t}{\Delta x} \left[\alpha_m - \alpha_{m-1} \alpha_m \left(\frac{\Delta t}{\Delta x} z \right) \right. \\ &\quad \left. + \dots - (-1)^m \alpha_1 \alpha_2 \dots \alpha_m \left(\frac{\Delta t}{\Delta x} z \right)^{m-1} \right],\end{aligned}\tag{10.27}$$

provided the convective and the dissipative part of z are evaluated at each stage (so-called (m, m) -scheme). The derivation of f becomes more involved for the hybrid multistage schemes (Section 6.1.2). In the case of the (5,3)-scheme (Eq. (6.7)), the Fourier symbol of the time-stepping operator reads [10]

$$\begin{aligned}f &= \frac{\Delta t}{\Delta x} \{ \alpha_5 - \alpha_5 \alpha_4 (z_I + \beta_5 z_R) (1 - \alpha_3 z_I) \\ &\quad - \alpha_5 \alpha_4 \alpha_2 (1 - \alpha_1 z_I) (z_I + \beta_5 z_R) [\alpha_3 (z_I + \beta_3 z_R) z_I - \beta_3 z_R] \\ &\quad - \alpha_5 \alpha_2 (1 - \beta_5) \beta_3 (1 - \alpha_1 z_I) z_R \} \end{aligned}\tag{10.28}$$

with

$$z_R = \frac{\Delta t}{\Delta x} \text{Real}(z) \quad \text{and} \quad z_I = \frac{\Delta t}{\Delta x} \text{Imag}(z)\tag{10.29}$$

representing the real and the imaginary part of z . The stage coefficients α_k and the blending coefficients β_k are shown in Table 6.2.

The time step Δt can be determined with the aid of the Courant-Friedrichs-Lowy (CFL) condition [11]. The following formula can be found for the convection model equation (10.13):

$$\Delta t = \sigma \frac{\Delta x}{|\Lambda|}.\tag{10.30}$$

The parameter σ in Eq. (10.30) denotes the CFL number. Its magnitude depends on the type and on the stage coefficients of the time-stepping scheme. The derivation of Eq. (10.30) is presented in Section 10.3.6. In the case of the convection-diffusion model equation (10.20), the relation

$$\Delta t = \sigma \frac{\Delta x}{|\Lambda| + C\Lambda_v} \quad (10.31)$$

holds for the time step. The factor C in Eq. (10.31) varies with the spatial discretization. For the central scheme Eq. (10.15), $C = 4$ results in good damping. In the case of the first-order upwind scheme Eq. (10.17), $C = 2$ guarantees that the Fourier symbol of $z = z_c - z_v$ (Eq. (10.23)) remains bounded by z_c . In fact, the Fourier symbol with the diffusion term is identical to the Fourier symbol of z_c for $\Phi = \pi$, regardless of the ratio $\Lambda_v/|\Lambda|$. The same situation is encountered for the second-order upwind scheme Eq. (10.19) if one sets $C = 1$.

Examples of Fourier symbols and amplification factors

In the following, we shall consider a few applications of the von Neumann stability analysis to the convection and the mixed convection-diffusion model problems. The LHS of Fig. 10.1 shows the locus of the Fourier symbol of the spatial operator z (thick line) together with the isolines of the magnitude of the amplification factor $|g|$. The boundary of the stability region is represented by $|g| = 1$. The behavior of $|g|$ with respect to the phase angle Φ is displayed on the right-hand side. This allows the assessment of the damping properties.

Fourier symbols and damping of upwind discretization schemes are displayed in Fig. 10.1 for the convection model equation. In both cases, a three-stage scheme is employed with optimized coefficients (Table 6.1). The behavior of the first-order upwind scheme Eq. (10.16) is shown in Fig. 10.1a and b. As we can see, the locus of the Fourier symbol passes all three minima of $|g|$ (small circular areas on the isoplot). This leads to very low magnitude of the amplification factor for a phase angle $\Phi \geq \pi/2$, which is particularly important for an efficient multigrid scheme ($\Phi \geq \pi/2$ on coarse grid corresponds to $\Phi < \pi/2$ on fine grid). We can also observe that the damping properties are poor for $\Phi < \pi/2$. This behavior is typical for the multistage schemes. It explains their low asymptotic convergence rate, which can be best improved by multigrid. It should be noted that the range $0 \leq \Phi \leq \pi$ represents the first half of the locus on the LHS.

Figure 10.1b demonstrates what happens, if the CFL number σ is increased too much. As expected, the Fourier symbol extends behind the stability boundary. Consequently, the magnitude of g becomes larger than unity (dashed line). This means that the solution errors are amplified for the corresponding phase angles (frequencies). Such a time-stepping scheme will clearly diverge. Figure 10.1c shows the properties of the second-order upwind scheme Eq. (10.18). In principle, the behavior is similar to that of the first-order upwind scheme.

The following plots in Fig. 10.2 display the loci of the Fourier symbols and the damping properties of the hybrid (5,3)-scheme (Section 6.1.2) applied to the convection

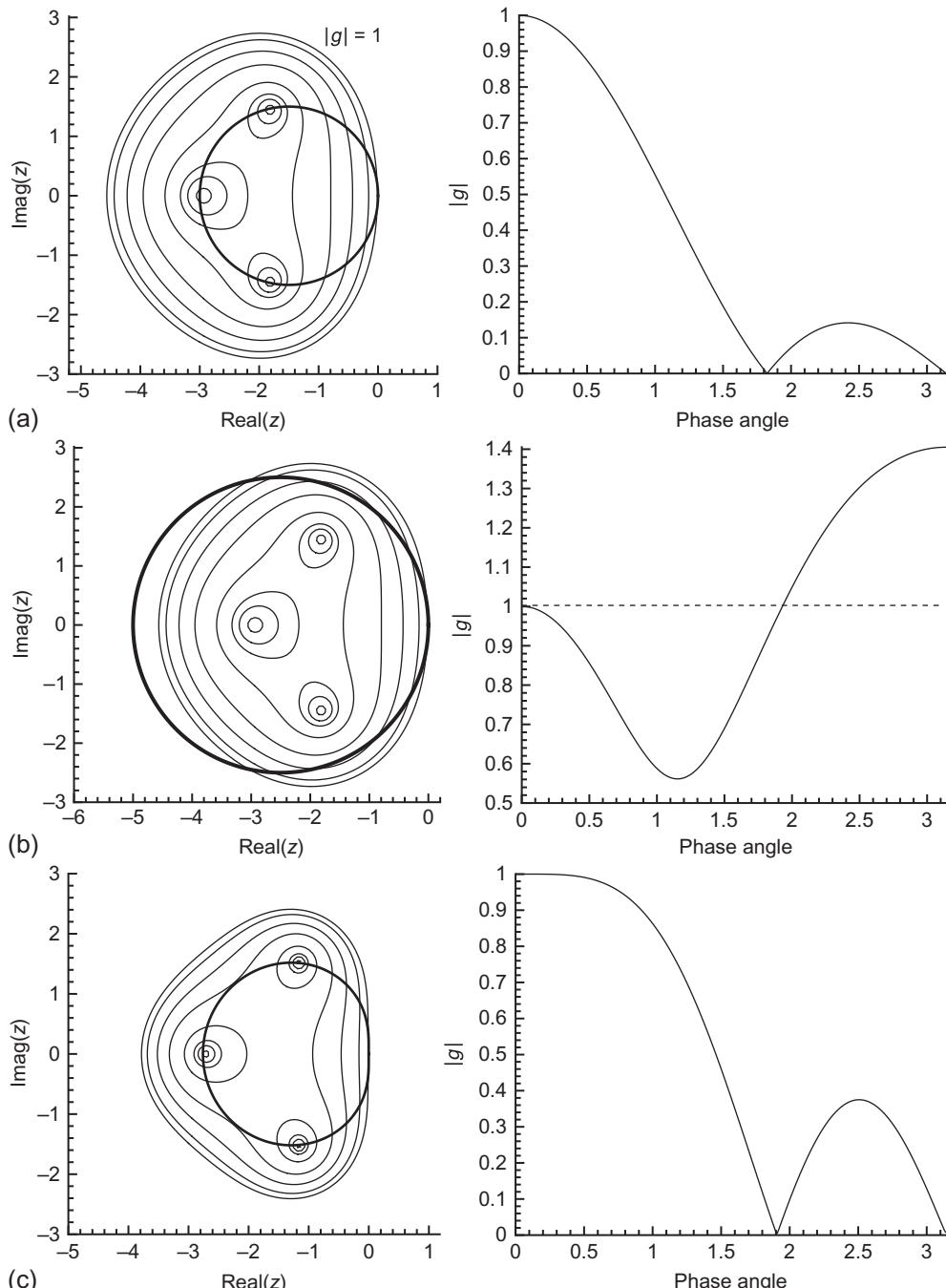


Figure 10.1 Convection model equation—Fourier symbol of the spatial operator (z) and magnitude of the amplification factor ($|g|$) in the case of the explicit (3,3)-scheme: (a) first-order upwind discretization; $\sigma = 1.5$; stage coefficients: 0.1481, 0.4, 1.0, (b) like above but $\sigma = 2.5$, (c) second-order upwind; $\sigma = 0.69$; stage coefficients: 0.1918, 0.4929, 1.0.

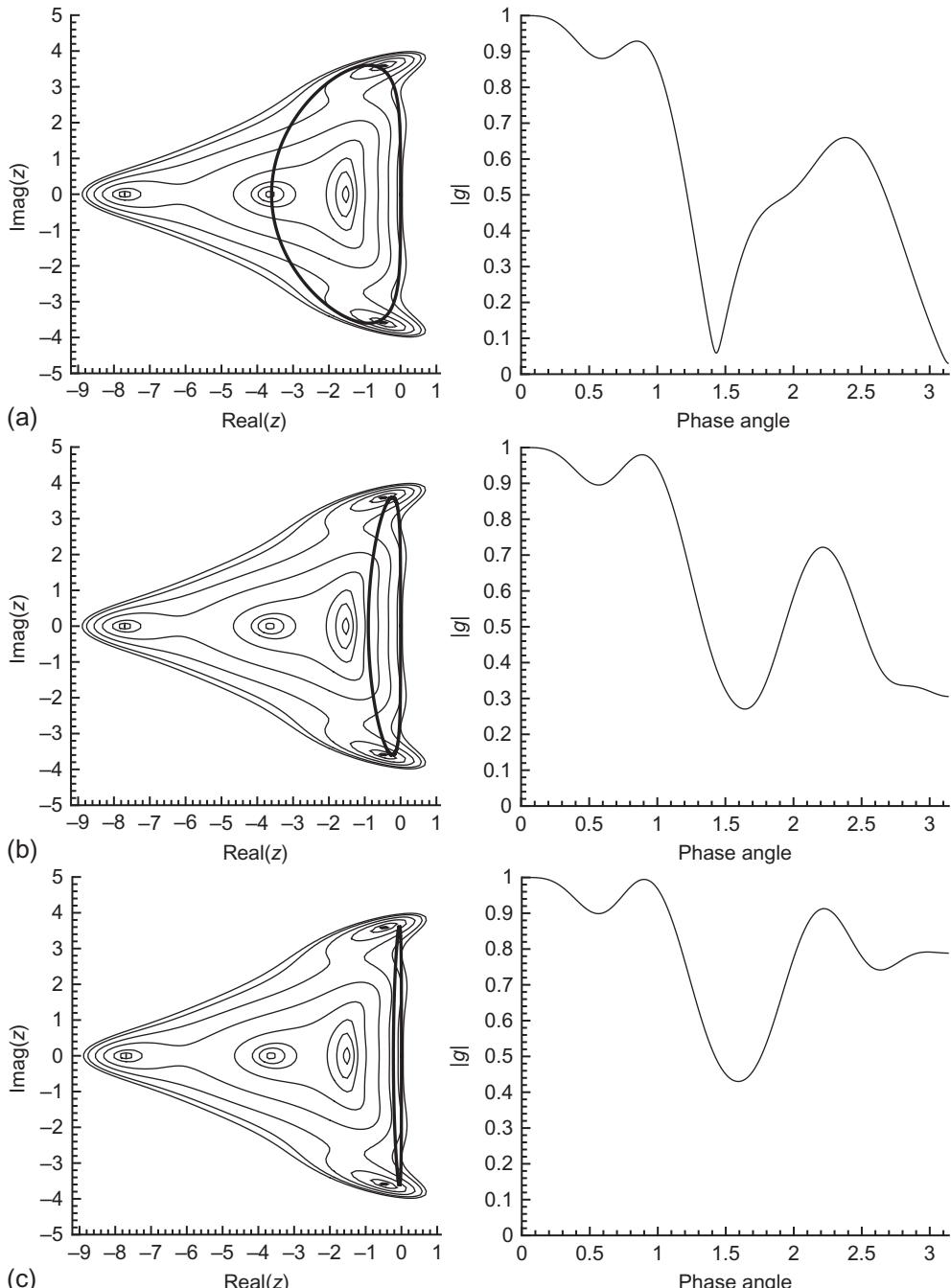


Figure 10.2 Convection model equation—Fourier symbol of the spatial operator (z) and magnitude of the amplification factor ($|g|$) in the case of the explicit (5,3)-scheme with central spatial discretization, $\sigma = 3.6$, stage coefficients from Table 6.2: (a) $\epsilon^{(4)} = 1/16$, (b) $\epsilon^{(4)} = 1/64$, (c) $\epsilon^{(4)} = 1/256$.

model equation (10.13). The spatial discretization utilizes the central scheme with artificial dissipation (Eq. (10.14)). In order to demonstrate the influence of the dissipation coefficient $\epsilon^{(4)}$, its value is varied from 1/16 (Fig. 10.2a) to 1/256 (Fig. 10.2c). We can conclude from the results that the locus is contracted along the real axis with decreasing amount of artificial dissipation. This effect is caused by down-sizing the term $(1 - \cos \Phi)^2$ in Eq. (10.15). More important is the fact that the damping properties deteriorate with reduced artificial dissipation. This means in practice that the convergence speed and the robustness of the scheme will degrade with lower dissipation level.

The properties of the hybrid (5,3)-scheme employed to solve the convection-diffusion equation (10.20) are investigated next. At first, the scheme is coupled to the first-order upwind spatial discretization Eq. (10.16). In Fig. 10.3, comparison is made between the cases $\Lambda_v = 0$ (pure convection) and $\Lambda_v/\Lambda = 2$ (denoted by dashed line). As we can see, the locus of the Fourier symbol is contracted along the imaginary axis due to the influence of the diffusion term z_v , which has only a real component. The extension of the Fourier symbol along the real axis is kept, if the time step is computed according to Eq. (10.31) with $C = 2$. The damping of the scheme remains on about the same favorable level as for pure convection. This is caused by the optimized, flat minimum of $|g|$ in the region surrounded by the locus of z_c (right-hand side of Fig. 10.3).

The behavior of the hybrid (5,3)-scheme with central discretization (Eq. (10.14)) is displayed in Fig. 10.4. The dissipation coefficient was set to $\epsilon^{(4)} = 1/64$ and the ratio of the eigenvalues to $\Lambda_v/\Lambda = 2$, respectively. We can observe that the locus of the Fourier symbol changes its form completely as compared to the convection equation. In the limit case $\Lambda_v \rightarrow \infty$, the locus would degrade to a line. Therefore, it is important that

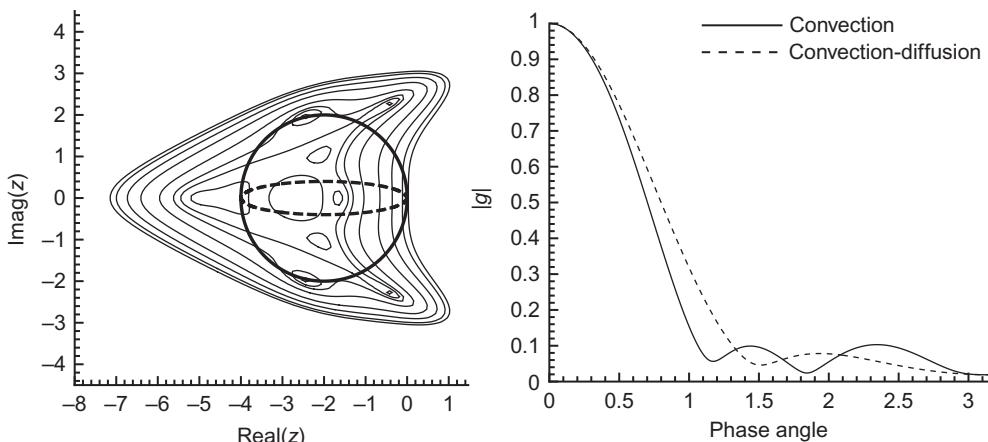


Figure 10.3 Convection-diffusion model equation—Fourier symbol of the spatial operator (z) and the magnitude of the amplification factor ($|g|$) in the case of the explicit (5,3)-scheme with first-order upwind spatial discretization, $\sigma = 2.0$, stage and blending coefficients from Table 6.2.

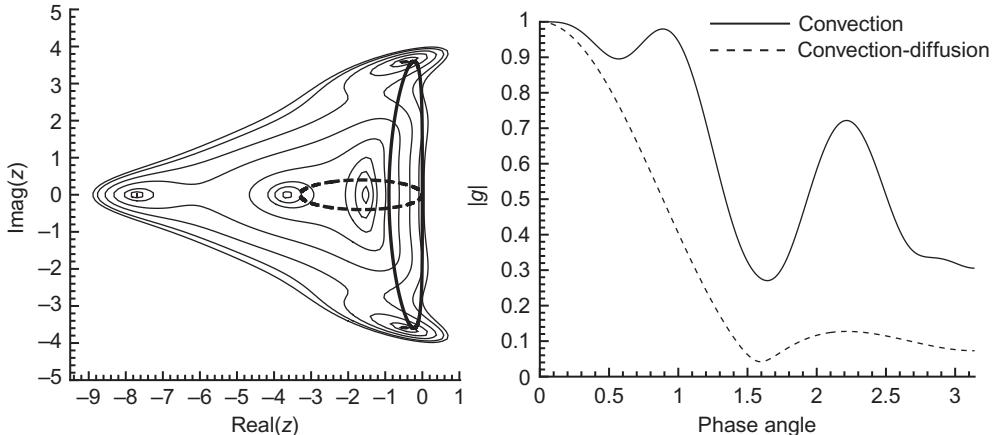


Figure 10.4 Convection-diffusion model equation—Fourier symbol of the spatial operator (z) and the magnitude of the amplification factor ($|g|$) in the case of the explicit (5,3)-scheme with central spatial discretization, $\sigma = 3.6$, $\epsilon^{(4)} = 1/64$, stage and blending coefficients from Table 6.2.

the time-stepping is optimized to have a shallow minimum of $|g|$ along the real axis. It should be mentioned that the time step was evaluated with Eq. (10.31) and $C = 4$.

Further examples related to the von Neumann stability analysis can be found in Refs. [9, 12–18]. A program for the stability analysis of explicit multistage schemes (with implicit residual smoothing) is also provided with the source codes.

10.3.5 Implicit time-stepping

According to Eq. (6.28) in Section 6.2, a general implicit scheme for the integration of the model equations can be formulated as

$$\left[\frac{\Delta x}{\Delta t} + \beta \left(\frac{\partial R}{\partial U} \right)_i \right] \Delta U^n = -R_i^n. \quad (10.32)$$

We can rewrite the flux Jacobian $\partial R / \partial U$ as a sum of two difference operators—one for the convection and one for the diffusion term in Eq. (10.20), i.e.,

$$\left\{ \frac{\Delta x}{\Delta t} + \beta \left[(D_x^I)_c - (D_x^I)_v \right]_i \right\} \Delta U^n = -R_i^n. \quad (10.33)$$

The convection difference operator can take various forms. For instance, in the case of the central scheme with artificial dissipation it becomes (cf. Eq. (6.47))

$$(D_x^I)_c \Delta U^n = \frac{\Lambda}{2} (\Delta U_{i+1}^n - \Delta U_{i-1}^n) + \Lambda \epsilon^I (\Delta U_{i+1}^n - 2\Delta U_i^n + \Delta U_{i-1}^n), \quad (10.34)$$

where ϵ^I denotes the implicit dissipation coefficient. We do not include here the 4th differences, since those are only seldom used in practice (because of high numerical effort). The first- or the second-order upwind scheme in the implicit operator leads to

$$(D_x^I)_c \Delta U^n = \Lambda(\Delta U_i^n - \Delta U_{i-1}^n), \quad (10.35)$$

or

$$(D_x^I)_c \Delta U^n = \frac{\Lambda}{2}(3\Delta U_i^n - 4\Delta U_{i-1}^n) + \Delta U_{i-2}^n, \quad (10.36)$$

respectively. The diffusion difference operator is usually of central type

$$(D_x^I)_v \Delta U^n = \frac{\Lambda_v}{2}(\Delta U_{i+1}^n - 2\Delta U_i^n + \Delta U_{i-1}^n). \quad (10.37)$$

The discretization of the explicit operator can be conducted accordingly to one of the relations (10.14), (10.16), or (10.18).

If we insert the Fourier mode Eq. (10.9) into the implicit scheme Eq. (10.33), we obtain

$$\left[\frac{\Delta x}{\Delta t} + \beta z^I \right] \Delta \hat{U}^n = -z^E \hat{U}^n, \quad (10.38)$$

where $z^I = z_c^I - z_v^I$ denotes the Fourier symbol of the flux Jacobian and $z^E = z_c^E - z_v^E$ represents the Fourier symbol of the explicit operator. The forms of the Fourier symbols correspond to those derived in Sections 10.3.2 and 10.3.3. The amplification factor results with Eq. (10.12) as

$$g = 1 - \frac{z^E}{\left[\frac{\Delta x}{\Delta t} + \beta z^I \right]}. \quad (10.39)$$

Thus, the Fourier symbol of the time-stepping operator is $f = 1/[\dots]$. A quick inspection of Eq. (10.39) reveals that for $\Delta t \rightarrow \infty$, $\beta = 1$ and $z^I = z^E$, the amplification factor will be zero for phase angles $\Phi > 0$. This situation occurs for the Newton scheme (see Section 6.2.5), if the flux Jacobian is exact. This explains the very fast convergence of the exact Newton's method.

Examples of amplification factors

In the following, we shall investigate the damping properties of a few implicit schemes with varying discretizations of the explicit and the implicit operator. For further discussion, the interested reader is referred to Ref. [19], which contains von Neumann analysis of the popular LU-SGS scheme in 2D (single grid and multigrid). A program for the analysis of implicit schemes is also provided with the accompanying source codes.

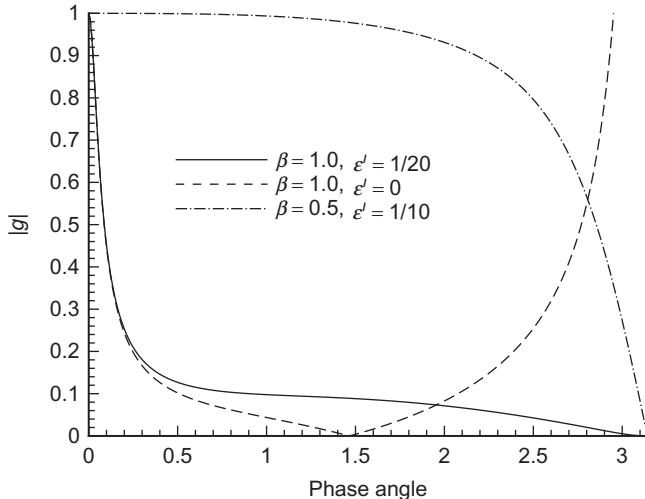


Figure 10.5 Convection model equation—magnitude of the amplification factor ($|g|$) in the case of an implicit scheme. Explicit and implicit operators discretized using central scheme, $\sigma = 20$, $\epsilon^{(4)} = 1/64$.

In the first example, we consider a scheme which applies central spatial discretization to the explicit (Eq. (10.14)) and the implicit operator (Eq. (10.34)). This is similar to the standard ADI scheme as presented in Section 6.2.3. We want to investigate the influence of parameter settings within the implicit operator on the amplification factor. Three exemplary results are compared in Fig. 10.5. The first curve (solid line) was generated with $\beta = 1$ and $\epsilon^I = 1/20$. The value of ϵ^I was chosen such that $|g| = 0$ at $\Phi = \pi$. The following formulas can be used to compute ϵ^I

$$\epsilon^I = \frac{1}{\beta} \left(4\epsilon^{(4)} - \frac{\Delta x}{4|\Lambda|\Delta t} \right). \quad (10.40)$$

The second curve (dashed line) demonstrates that the artificial dissipation has to be included in the implicit operator. For $\epsilon^I = 0$, the scheme becomes unstable at high frequencies ($\Phi \rightarrow \pi$). The last curve (dash-dotted) shows the magnitude of g for $\beta = 1/2$, i.e., for second-order accuracy in time. As we can see, the damping properties are much worse than for $\beta = 1$. Therefore, this value should be preferred. Unsteady flows are more efficiently solved by the dual-time stepping described in Section 6.3.2.

The effect of increasing CFL number is illustrated in Fig. 10.6. First-order upwind scheme (Eqs. (10.16) and (10.35)) is employed on both sides. The damping is very good due to the similarity between the explicit and the implicit operator. This confirms our remark with respect to Newton's scheme.

In Fig. 10.7, we compare the amplification factors for two different discretizations of the implicit operator (the left-hand side—LHS). The explicit operator is in both cases

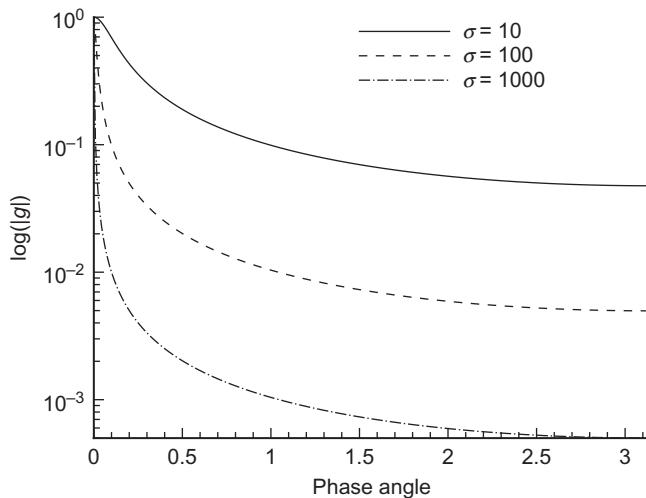


Figure 10.6 Convection model equation—magnitude of the amplification factor ($|g|$) in the case of an implicit scheme. Explicit and implicit operators discretized using first-order upwind scheme, $\beta = 1.0$. Note the logarithmic scaling of the y-axis.

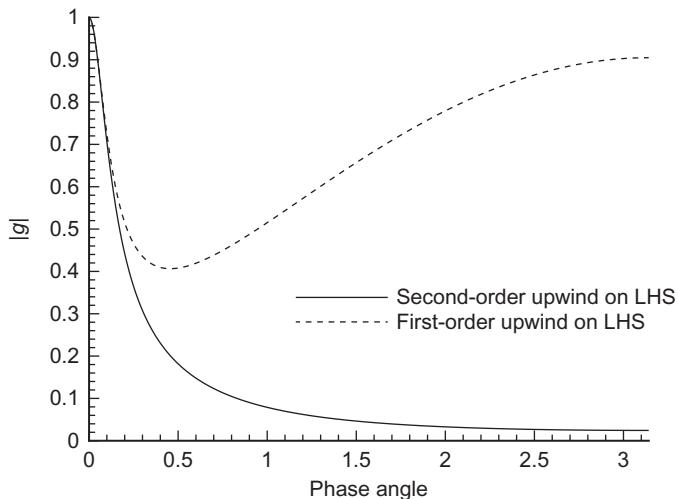


Figure 10.7 Convection model equation—magnitude of the amplification factor ($|g|$) in the case of an implicit scheme. Explicit operator discretized using second-order upwind scheme, $\sigma = 10$, $\beta = 1.0$.

discretized using second-order upwind scheme Eq. (10.18). It is evident that the exact representation of the explicit operator on the LHS leads to significantly better damping properties and thus to faster convergence. This observation was often confirmed in practice. The damping properties of a mixed first-/second-order discretization can be improved by increased over-relaxation as demonstrated for the LU-SGS scheme [19, 20].

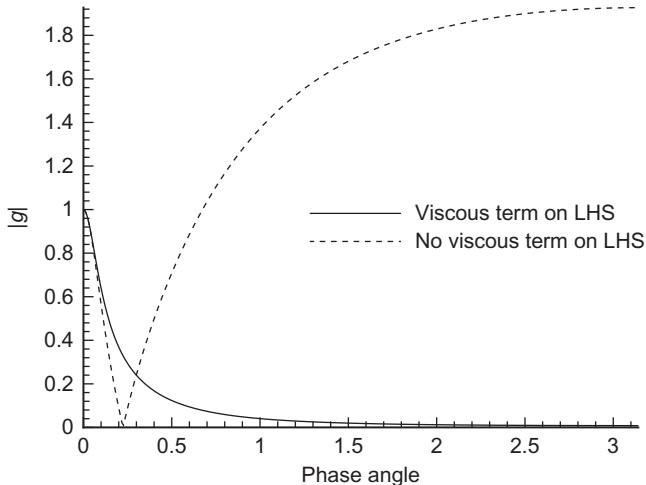


Figure 10.8 Convection-diffusion model equation—magnitude of the amplification factor ($|g|$) in the case of an implicit scheme. Explicit and implicit operators discretized using second-order upwind scheme, $\sigma = 10$, $\beta = 1.0$, $\Lambda_v/\Lambda = 2$.

The last example in Fig. 10.8 deals with the convection-diffusion model equation (10.20). We want to compare two cases. In the first one, the discretization of the diffusion term is also contained in the implicit operator (solid line). We can observe that the damping properties are very favorable—similar to those found for the convection model problem. However, if the diffusion term is removed from the implicit operator (dashed line in Fig. 10.8), the scheme becomes unstable, even at this low ratio of viscous to convective eigenvalue ($\Lambda_v/\Lambda = 2$). Therefore, the viscous fluxes should be always included in the approximation of the flux Jacobian to obtain a robust scheme.

In summary, we can state that the implicit operator should include at least the most important features of the spatial discretization and of the physical problem. Rather crude approximations of the flux Jacobian $\partial \vec{R} / \partial \vec{W}$ can easily lead to an unstable scheme. Therefore, it is very important to find a reasonable compromise between the numerical effort and the accuracy of the numerical flux Jacobian.

A comparison to the results of the explicit multistage scheme reveals that the damping properties of a properly designed implicit scheme are significantly better. This is especially true for the damping at low phase angles (frequencies). Therefore, we can expect faster asymptotic convergence rates from an implicit scheme as compared to a multistage scheme.

10.3.6 Derivation of the CFL condition

Every explicit time-stepping scheme remains stable only up to a certain value of the time step Δt . The necessary, but not sufficient, condition for stability of a time-stepping scheme was formulated by Courant et al. [11]. The so-called CFL condition states

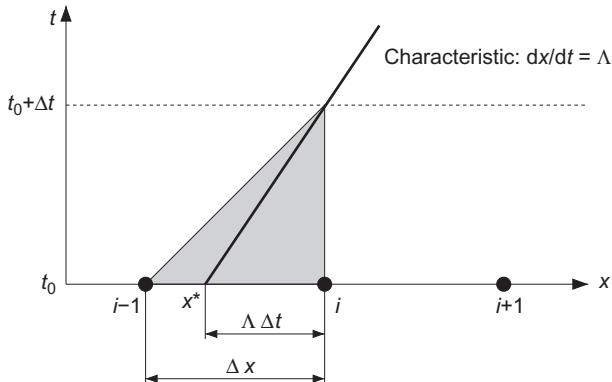


Figure 10.9 Domain of dependence of an explicit time-stepping scheme (shaded area) versus domain of dependence of the convective equation (thick line).

that the domain of dependence of the numerical scheme has to include the domain of dependence of the partial-differential equation. In order to clarify this statement, let us consider the $x - t$ diagram in Fig. 10.9. The domain of dependence of the convection model equation (10.13) is given by the characteristic $dx/dt = \Lambda$. This means that any information is carried with this speed across the domain. Consequently, the exact solution at the time $(t_0 + \Delta t)$ is equal to the solution at the time t_0 but at the space coordinate $x^* = x_i - \Lambda \Delta t$. In order to simulate the behavior of the exact solution correctly, the stencil of the spatial discretization must enclose the point x^* . Thus, at least the point x_{i-1} has to be included (if $\Lambda > 0$). The domain of dependence of such a numerical scheme is represented by the shaded area in Fig. 10.9. Hence, we can formulate the condition

$$\Lambda \Delta t \leq \Delta x \quad \text{or} \quad \Delta t \leq \frac{\Delta x}{\Lambda}. \quad (10.41)$$

This leads us to the CFL condition

$$\sigma = |\Lambda| \frac{\Delta t}{\Delta x} \leq 1. \quad (10.42)$$

Of course, the explicit multistage scheme allows for CFL numbers $\sigma > 1$, since the new solution at the time $(t_0 + \Delta t)$ is obtained in more than just one step.

Based on the above arguments, we can easily show that an implicit scheme like in Eq. (10.32) always fulfills the CFL condition, since the numerical domain of dependence extends over all grid points.

CFL condition by von Neumann analysis

The CFL condition in Eq. (10.42) can also be derived by the von Neumann stability analysis. Let us consider for this purpose the convection model equation (10.13)

discretized using the first-order upwind scheme Eq. (10.16) and a one-stage explicit scheme. The domain of dependence of this numerical scheme corresponds to the shaded region in Fig. 10.9. According to Eq. (10.12), we obtain the amplification factor g from the relationship

$$g = 1 - \frac{\Delta t}{\Delta x} z, \quad (10.43)$$

since the Fourier symbol of the time-stepping operator reduces to $f = \Delta t / \Delta x$. If we substitute Eq. (10.17) for z in Eq. (10.43), the amplification factor will read

$$g = 1 - \frac{\Delta t}{\Delta x} \Lambda [I \sin \Phi + (1 - \cos \Phi)]. \quad (10.44)$$

Thus, the amplitude of g is given by (we assume $\Lambda > 0$)

$$|g|^2 = 2 \frac{\Delta t}{\Delta x} \Lambda \left(\frac{\Delta t}{\Delta x} \Lambda - 1 \right) (1 - \cos \Phi) + 1. \quad (10.45)$$

It is easy to show that the maximum of $|g|^2$ occurs at a phase angle $\Phi = \pi$. If we set $\Phi = \pi$ in Eq. (10.45), we obtain

$$|g(\Phi = \pi)|^2 = 4 \frac{\Delta t}{\Delta x} \Lambda \left(\frac{\Delta t}{\Delta x} \Lambda - 1 \right) + 1. \quad (10.46)$$

In order for the time-stepping scheme to be stable, it must hold that $|g|^2 \leq 1$. This condition can only be fulfilled if

$$\frac{\Delta t}{\Delta x} \Lambda \leq 1 \quad (10.47)$$

and hence

$$\Delta t \leq \frac{\Delta x}{\Lambda}. \quad (10.48)$$

This corresponds exactly to Eq. (10.41).

It is important to note that the CFL condition (10.42) is *not* sufficient (however necessary) to guarantee stability of the numerical scheme. Therefore, the von Neumann analysis should be carried out as well.

REFERENCES

- [1] Roache PJ. Quantification of uncertainty in computational fluid dynamics. *Ann Rev Fluid Mech* 1997;29:123–60.
- [2] AIAA. Guide for the verification and validation of computational fluid dynamics simulations. AIAA G-077-1998(2002); 1998 and 2002.
- [3] De Vahl DG. Natural convection of air in a square cavity: a bench mark numerical solution. *Int J Num Method Fluids* 1983;3:249–64.
- [4] Cranck J, Nicholson P. A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type. *Proc Camb Philos Soc* 1947;43:50–67.

- [5] Charney JG, Fjortoft R, von Neumann J. Numerical integration of the barotropic vorticity equation. *Tellus* 1950;2:237–54.
- [6] Hirsch C. Numerical computation of internal and external flows, vol. 1. New York: John Wiley and Sons; 1988.
- [7] Roache PJ. Computational fluid dynamics. Albuquerque, USA: Hermosa Publishers; 1972.
- [8] Roache PJ. Fundamentals of computational fluid dynamics. Albuquerque, USA: Hermosa Publishers; 1998.
- [9] Kroll N, Jain RK. Solution of two-dimensional Euler equations—experience with a finite volume code. DLR Research Report, No. 87-41; 1987.
- [10] Radespiel R, Swanson RC. Progress with multigrid schemes for hypersonic flow problems. ICASE Report No. 91-89; 1991; also *J Comput Phys* 1995;116:103–22.
- [11] Courant R, Friedrichs KO, Lewy H. Über die partiellen differenzengleichungen der mathematischen Physik. *Math Ann* 1928;100:32–74. Translation: On the partial difference equations of mathematical physics. *IBM J* 1967;11:215–34.
- [12] Jameson A. Multigrid algorithms for compressible calculations. multigrid methods II, Lecture notes in mathematics, vol. 1228. New York: Springer Verlag; 1985.
- [13] Van Leer B, Tai C-H, Powell KG. Design of optimally smoothing multi-stage schemes for the Euler equations. AIAA Paper 89-1933; 1989.
- [14] Lötstedt P, Gustafsson B. Fourier analysis of multigrid methods for general systems of PDE. Report No. 129/1990. Sweden: Department of Scientific Computing, Uppsala University; 1990.
- [15] Blazek J, Kroll N, Radespiel R, Rossow, C-C. Upwind implicit residual smoothing method for multi-stage schemes. AIAA Paper 91-1533; 1991.
- [16] Blazek J. Methods to accelerate the solution of the Euler- and the Navier-Stokes equations for steady-state super- and hypersonic flows. Translation of DLR Research Report, No. 94-35, ESA-TT-1331; 1995.
- [17] Tai C-H, Sheu J-H, van Leer B. Optimal multistage schemes for Euler equations with residual smoothing. *AIAA J* 1995;33:1008–16.
- [18] Tai C-H, Sheu J-H, Tzeng P-Y. Improvement of explicit multistage schemes for central spatial discretization. *AIAA J* 1996;34:185–8.
- [19] Blazek J. Investigations of the implicit LU-SSOR scheme. DLR Research Report, No. 93-51; 1993.
- [20] Blazek J. A multigrid LU-SSOR scheme for the solution of hypersonic flow problems. AIAA Paper 94-0062; 1994.

CHAPTER 11

Principles of Grid Generation

Contents

11.1	Structured Grids	360
11.1.1	C-, H-, and O-grid topology	361
<i>C-grid topology</i>	362	
<i>H-grid topology</i>	362	
<i>O-grid topology</i>	362	
11.1.2	Algebraic grid generation	365
11.1.3	Elliptic grid generation	367
11.1.4	Hyperbolic grid generation	370
11.2	Unstructured Grids	373
11.2.1	Delaunay triangulation	374
<i>Watson's algorithm</i>	376	
<i>Constrained Delaunay triangulation</i>	377	
11.2.2	Advancing-front method	378
11.2.3	Generation of anisotropic grids	380
<i>Stretched Delaunay triangulation</i>	380	
<i>Advancing-layers method</i>	381	
<i>Combined advancing-normal/Delaunay method</i>	383	
<i>Stretched surface grids</i>	383	
11.2.4	Mixed-element/hybrid grids	384
<i>Mixed prismatic/tetrahedral grids</i>	385	
<i>Mixed prismatic/tetrahedral/Cartesian grids</i>	385	
11.2.5	Assessment and improvement of grid quality	386
<i>Grid smoothing</i>	388	
	References	388

Prior to the numerical solution of the governing equations, we have to discretize the surfaces of all boundaries and to generate a volume grid inside the flow domain. As we discussed at the beginning of Chapter 3 (in Section 3.1), we can choose basically between:

- *structured*, and
- *unstructured*

grids. An example of structured grid for a civil aircraft [1, 2] is presented in Figs. 11.1 and 11.2. For comparison, an unstructured surface and volume grid for a similar configuration [3–6] is displayed in Figs. 11.3 and 11.4.

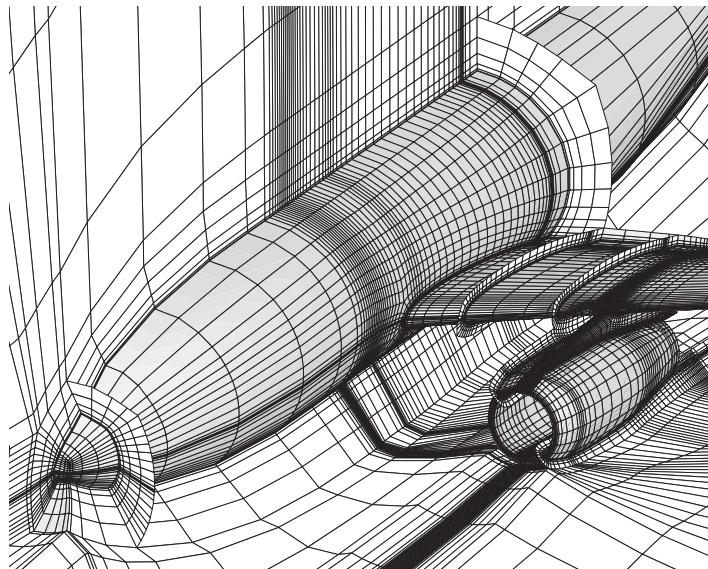


Figure 11.1 Structured surface and volume grid of a wing-body configuration. (Courtesy O. Brodersen, DLR, Germany.)

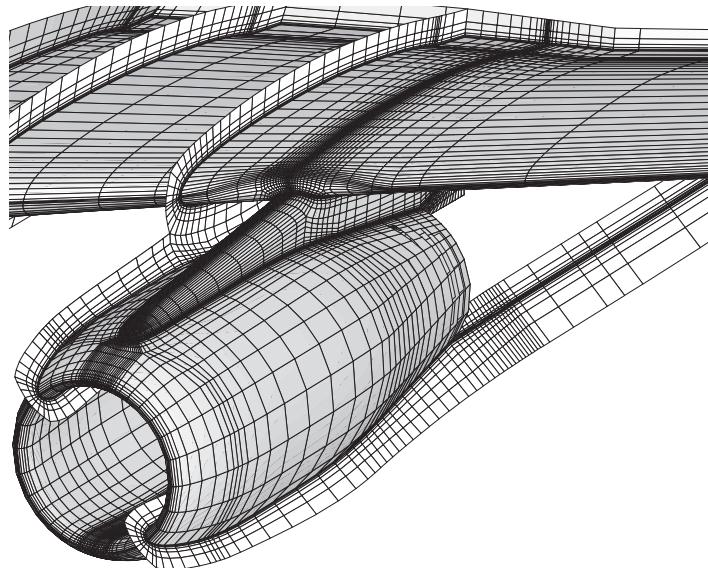


Figure 11.2 Structured surface and volume grid of a wing-body configuration—detail of the pylon and the engine nacelle. (Courtesy O. Brodersen, DLR, Germany.)

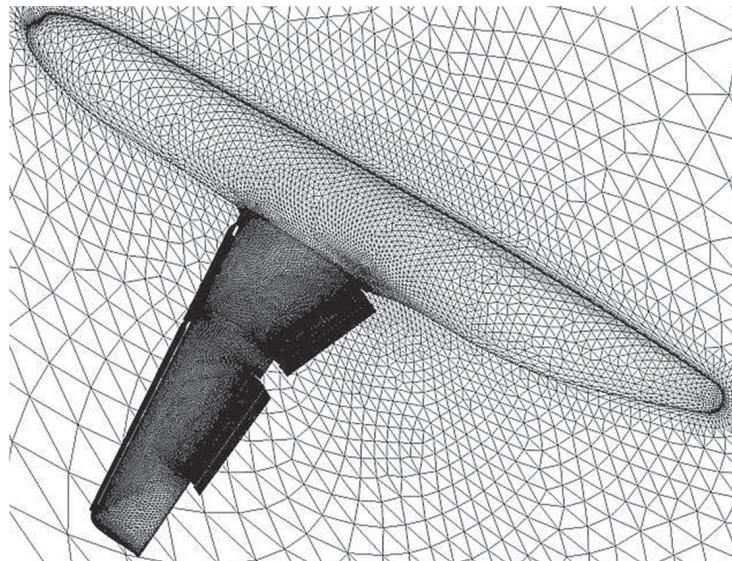


Figure 11.3 Unstructured surface grid of a wing-body configuration. (Courtesy D. Mavriplis, University of Wyoming, USA.)

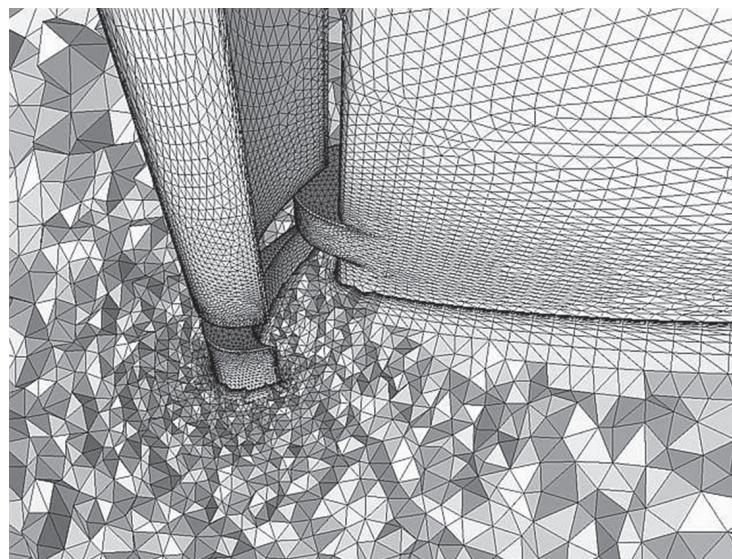


Figure 11.4 Unstructured grid of a wing-body configuration—detail of the slats and rough-cut through the volume grid. (Courtesy D. Mavriplis, University of Wyoming, USA.)

The structured as well as the unstructured grids have their specific advantages and shortcomings, which we mentioned in Section 3.1. However, regardless of the grid type, quite often the bottleneck is the quality of data being imported from a CAD (computer-aided design) system into the grid generation program. The analytical surface definition is usually transferred via one of the standard formats like IGES (Initial Graphics Exchange Specification—[7, 8]), STEP (STandard for the Exchange of Product model data—[9–11]), or DXF (Drawing eXchange Format—[12]). Another popular possibility is to transfer directly triangular surface meshes via the STL (STereoLithography) format [13–15], which is offered by all modern CAD programs. Direct utilization of native CAD data is rather rare [16]. Experience shows that the transfer process can impair the accuracy of the data. Furthermore, the surface representation in the CAD system itself is often imprecise. This leads mostly to gaps, overlaps, or to discontinuities between neighboring surface patches. Such defects have to be eliminated before the surfaces can be discretized, or before they can be employed to generate the volume grid. We speak in this respect of “CAD repair” [17, 18].

In the following, we shall present the basic methodologies applied in the generation of structured (Section 11.1) as well as of unstructured grids (Section 11.2). Due to the restricted space, we can provide here only a brief description. We refer the reader to Refs. [19, 20] for a deeper discussion of surface modeling, structured and unstructured surface, and volume grid generation. A review of the development of grid generation approaches can be found in Ref. [21].

11.1 STRUCTURED GRIDS

The distinguishing feature of structured grids is that the grid points in the physical space are mapped in an unique way onto a continuous set of three integers i, j, k (one for each coordinate direction). The set of integers defines what is called the *computational* space (see Fig. 3.2). The coordinates ξ, η, ζ in the computational space are related to i, j, k as follows:

$$\begin{aligned}\xi &= i/i_{\max}, & i &= 0, 1, 2, \dots, i_{\max}, \\ \eta &= j/j_{\max}, & j &= 0, 1, 2, \dots, j_{\max}, \\ \zeta &= k/k_{\max}, & k &= 0, 1, 2, \dots, k_{\max}.\end{aligned}\tag{11.1}$$

This mapping implies that $0 \leq \xi \leq 1$, $0 \leq \eta \leq 1$, and $0 \leq \zeta \leq 1$. Neighboring grid points can be connected to form cubes in the computational and hexahedra (quadrilaterals in 2D) in the physical space. Structured grid generation systems discretize the boundary surfaces of the flow domain using quadrilaterals—termed the *surface* grid—and fill the interior with hexahedra. The grid inside the domain is named the *volume* grid.

The generation of a structured grid starts by distributing grid points along boundary curves (boundaries of surface patches). The usual procedure is to place the nodes more

dense in regions with high curvature. Using the point distribution on boundary curves, the surface grid can be generated. Based on the surface grids which enclose the physical domain, we can finally construct the volume grid. Thus, the common problem is how to generate a grid inside the domain based on known boundary discretization. This can be solved by two different approaches:

- algebraic grid generation, or
- grid generation using *partial differential* equations (PDEs).

The application of PDEs requires a valid initial grid (surface or volume), which is mostly generated algebraically. Two different types of PDEs are common:

- *elliptic* equations, and
- *hyperbolic* equations.

The algebraic grid generation ([Section 11.1.2](#)) employs a direct functional description of the coordinate transformation between the computational and the physical space. The most widely used algebraic technique is the so-called *Transfinite Interpolation* (TFI). Given the point distribution on all boundaries, it generates the grid points inside the physical domain by interpolation. Particular formulations of the TFI method allow for angle and grid spacing control at the boundaries.

The methodology based on elliptic PDEs ([Section 11.1.3](#)) is the most popular one. It allows the user to prescribe the angle between a grid line and boundary and to control the grid spacing and the expansion ratio near surfaces. Elliptic grid generation also guarantees a smooth grid in the entire domain. Thus, high quality, boundary orthogonal grids can be generated. The downside of the elliptic method is a much longer computing time as compared to algebraic or hyperbolic grid generation approaches. The method also suffers from numerical difficulties.

Hyperbolic PDEs can also be utilized for the grid generation ([Section 11.1.4](#)). This technique generates the volume grid by marching between two surfaces in the direction of one particular computational coordinate. The marching procedure is explicit, i.e., based on a known surface point distribution a new layer is generated. A natural restriction of the hyperbolic grid generation is that the shape of the outer grid boundary cannot be fully controlled. However, this may represent no real problem like in the case of external flows. The hyperbolic grid generation can provide approximate grid orthogonality over the entire domain. It is also computationally inexpensive.

11.1.1 C-, H-, and O-grid topology

Before we can start to generate any grid, we have to think about its topology. This means, we have to decide how many grid blocks are necessary and how the blocks should be ordered with respect to each other (by the way, this work may take weeks to months in the case of a complex geometry). For each grid block, we have to assign

boundaries (or their parts) in the computational domain to particular boundaries in the physical space (e.g., solid wall, far-field, etc.). The appearance of the grid in the physical space will depend strongly on this assignment. In practice, three standard single-block grid topologies are established. They are named as the C-, H-, or the O-grid topology because in a plane view the grid lines resemble the corresponding capital letter. A grid in 3D can be described as a combination of two topologies. For example, the grid around a wing usually consists of a C-grid in the flow direction (cf. Fig. 11.2) and of an O-grid (or an H-grid) in the span-wise direction. In this case, we speak of a C-O-grid. In the following, we shall discuss all three topologies in more detail.

C-grid topology

In the case of the C-topology the aerodynamic body is enclosed by one family of grid lines, which also form the wake region (if present). The situation is sketched in Fig. 11.5. As we can see, the lines $\eta = \text{const.}$ start at the far-field ($\xi = 0$), follow the wake, pass the trailing edge (node b), wrap in clockwise direction round the body, and finally continue to the far-field again ($\xi = 1$). The other family of grid lines ($\xi = \text{const.}$) emanates in the normal direction from the body and the wake. The part (segment) $a - b$ of the grid line $\eta = 0$ represents a *coordinate cut*. This means that the segment $a - b$ in the physical space is mapped onto two segments in the computational space, namely $a \leq \xi \leq b$ and $b' \leq \xi \leq a'$. Hence, the nodes on the upper ($b' - a'$) and the lower part ($a - b$) of the cut are kept separately in the computer memory. The appropriate boundary condition was presented in Section 8.7 (Fig. 11.6).

H-grid topology

The H-grid topology is quite often employed in turbomachinery for grid generation in the bladed flow-path. The topology is displayed in Fig. 11.7. As one can observe, the surface of the aerodynamic body is described here by two different grid lines, i.e., $\eta = 0$ and $\eta = 1$. On the contrary to the C-grid, one family of grid lines ($\eta = \text{const.}$) closely follows the streamlines (inlet located at $\xi = 0$, outlet at $\xi = 1$).

At the first sight, there is no obvious coordinate cut. However, in turbomachinery the segments $a - b$ and $e - f$ are periodic (rotationally periodic in 3D) to each other. The same is true for the segments $c - d$ and $g - h$. This type of boundary condition is treated in Section 8.8. Figure 11.8 shows an example of a nonorthogonal H-grid between turbine blades.

O-grid topology

We can see from the rendering of the O-topology in Fig. 11.9 that one family of grid lines ($\eta = \text{const.}$) forms closed curves around the aerodynamic body. The second

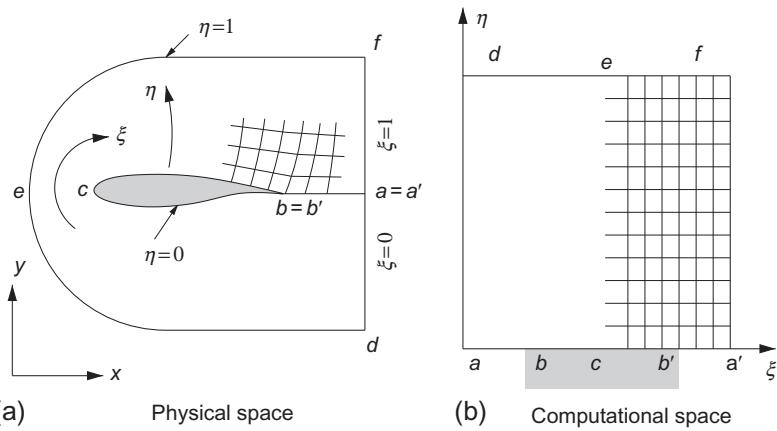


Figure 11.5 C-grid topology in 2D.

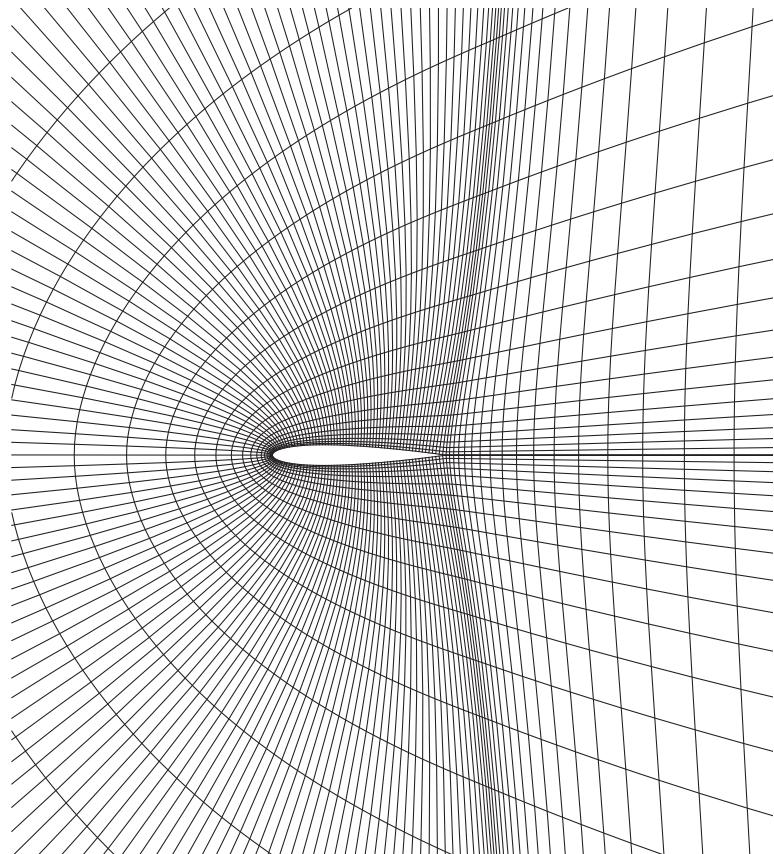


Figure 11.6 Partial view of a C-grid around NACA 0012 airfoil.

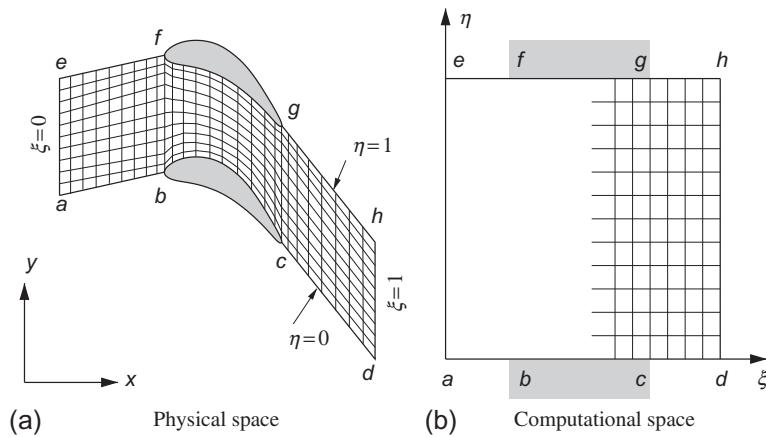


Figure 11.7 H-grid topology in 2D.

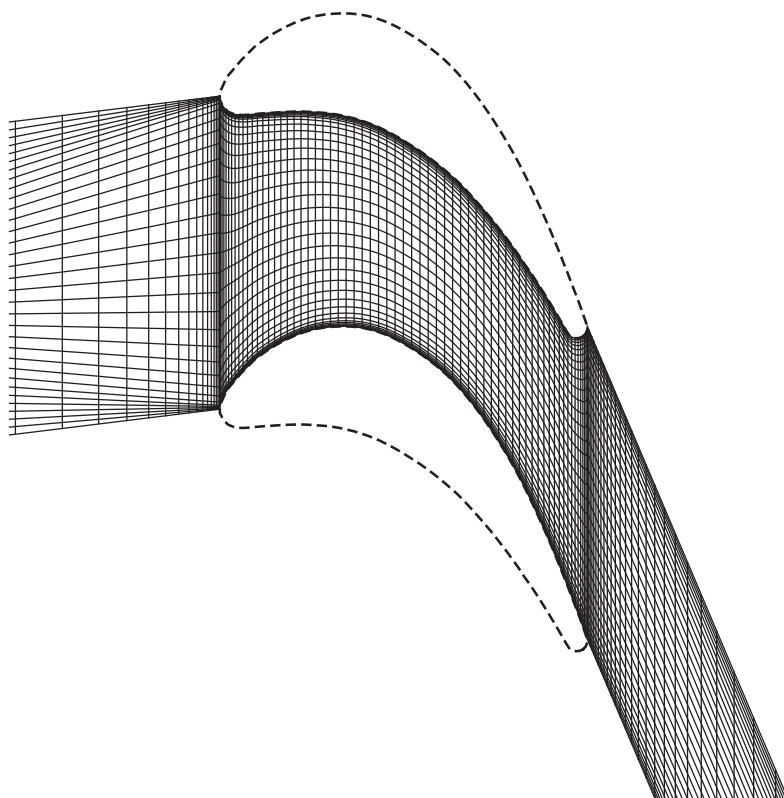


Figure 11.8 Partial view of an H-grid between turbine blades (dotted line).

family of grid lines ($\xi = \text{const.}$) is spanned in radial direction between the body and the outer boundary (far-field in this case). The complete boundary line $\eta = 0$ represents the contour of the body (from a to a'). The coordinate cut is defined by the boundaries $\xi = 0$ (nodes $a - c$) and $\xi = 1$ (nodes $a' - c'$) in the computational space. The example in Fig. 11.10 shows a standard O-grid used to simulate inviscid flow past an airfoil. A disadvantage of the O-topology is the poor grid quality at a sharp trailing edge.

11.1.2 Algebraic grid generation

The most widely used algebraic technique for surface or volume grid generation from prescribed boundary point distribution is the TFI method. It was first described by Gordon and Hall in 1973 [22]. The TFI scheme utilizes 1-D univariate interpolations in each of the coordinate directions in the computational space. The general form of the univariate interpolation functions reads

$$\begin{aligned}\vec{U} &= \sum_{i=1}^L \sum_{n=0}^P \alpha_i^n(\xi) \frac{\partial^n \vec{r}(\xi_i, \eta, \zeta)}{\partial \xi^n}, \\ \vec{V} &= \sum_{j=1}^M \sum_{m=0}^Q \beta_j^m(\eta) \frac{\partial^m \vec{r}(\xi, \eta_j, \zeta)}{\partial \eta^m}, \\ \vec{W} &= \sum_{k=1}^N \sum_{l=0}^R \gamma_k^l(\zeta) \frac{\partial^l \vec{r}(\xi, \eta, \zeta_k)}{\partial \zeta^l}.\end{aligned}\quad (11.2)$$

In Eq. (11.2), \vec{U} , \vec{V} , and \vec{W} denote the univariate interpolation functions in the ξ -, η -, and ζ -direction, respectively. Furthermore, $\alpha_i^n(\xi)$, $\beta_j^m(\eta)$, $\gamma_k^l(\zeta)$ are the blending functions, and \vec{r} stands for the position of a grid point in the physical space. In order to evaluate the interpolation functions, positions \vec{r} and derivatives $\partial^n \vec{r} / \partial \xi^n$, etc. have to be specified. Since we already discretized the boundary curves or surfaces, we can insert these values into Eq. (11.2). With \vec{U} , \vec{V} , \vec{W} known, we can generate the grid inside the domain by using the Boolean sum of the interpolation functions, i.e.,

$$\vec{r} = \vec{U} \oplus \vec{V} \oplus \vec{W} = \vec{U} + \vec{V} + \vec{W} - \vec{U}\vec{V} - \vec{U}\vec{W} - \vec{V}\vec{W} + \vec{U}\vec{V}\vec{W}. \quad (11.3)$$

The approach in Eq. (11.3) guarantees that in 2D all four boundary curves and in 3D all six boundary faces are matched. The tensor products in Eq. (11.3) are evaluated as follows:

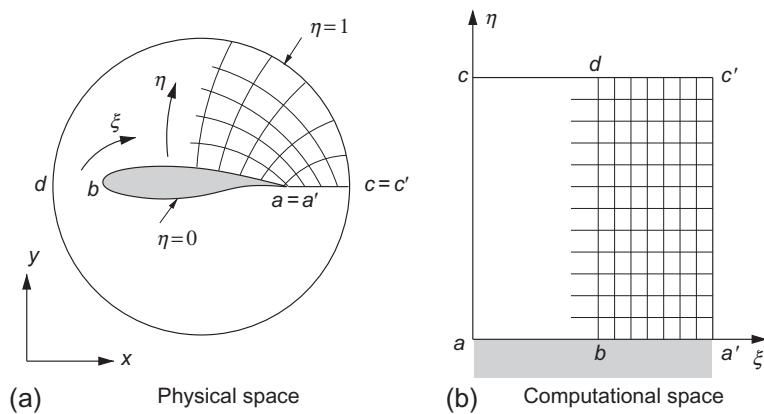


Figure 11.9 O-grid topology in 2D.

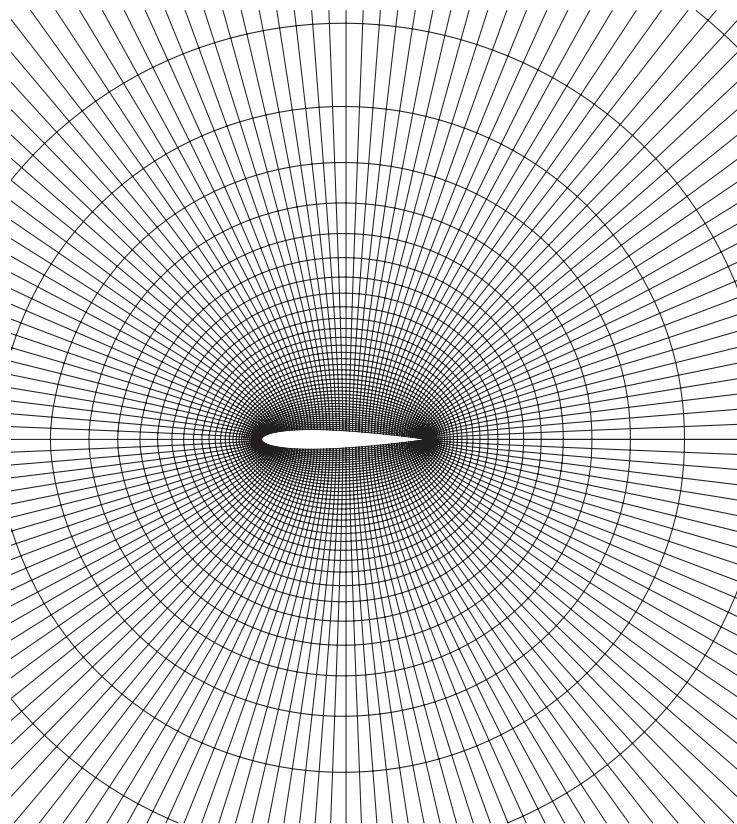


Figure 11.10 Partial view of an O-grid around NACA 0012 airfoil.

$$\begin{aligned}
\vec{U}\vec{V} &= \sum_{i=1}^L \sum_{j=1}^M \sum_{m=0}^Q \sum_{n=0}^P \alpha_i^n \beta_j^m \frac{\partial^{mn} \vec{r}(\xi_i, \eta_j, \zeta)}{\partial \eta^m \partial \xi^n}, \\
\vec{U}\vec{W} &= \sum_{i=1}^L \sum_{k=1}^N \sum_{l=0}^R \sum_{n=0}^P \alpha_i^n \gamma_k^l \frac{\partial^{ln} \vec{r}(\xi_i, \eta, \zeta_k)}{\partial \zeta^l \partial \xi^n}, \\
\vec{V}\vec{W} &= \sum_{j=1}^M \sum_{k=1}^N \sum_{l=0}^R \sum_{m=0}^Q \beta_j^m \gamma_k^l \frac{\partial^{lm} \vec{r}(\xi, \eta_j, \zeta_k)}{\partial \zeta^l \partial \eta^m}, \\
\vec{U}\vec{V}\vec{W} &= \sum_{i=1}^L \sum_{j=1}^M \sum_{k=1}^N \sum_{l=0}^R \sum_{m=0}^Q \sum_{n=0}^P \alpha_i^n \beta_j^m \gamma_k^l \frac{\partial^{lmn} \vec{r}(\xi_i, \eta_j, \zeta_k)}{\partial \zeta^l \partial \eta^m \partial \xi^n}.
\end{aligned} \tag{11.4}$$

More details on Boolean operators and tensor products related to grid generation can be found in Refs. [23, 24].

Various types of interpolation functions can be employed—linear, Lagrangian, Hermite, spline, etc. The most widespread method is the linear TFI. It is obtained by setting $L = M = N = 2$ and $P = Q = R = 0$ in Eqs. (11.2) and (11.4). With this, the volume grid can be generated based solely on the given point distribution on the six bounding surfaces (we set $\xi_1 = 0$, $\xi_2 = 1$, and similarly for η_j and ζ_k). The blending functions of the linear TFI read [25]

$$\begin{aligned}
\alpha_1^0(\xi) &= 1 - \xi, & \alpha_2^0(\xi) &= \xi, \\
\beta_1^0(\eta) &= 1 - \eta, & \beta_2^0(\eta) &= \eta, \\
\gamma_1^0(\zeta) &= 1 - \zeta, & \gamma_2^0(\zeta) &= \zeta.
\end{aligned} \tag{11.5}$$

The linear TFI is computationally very efficient. An example of algebraically generated grid using the linear TFI is shown in Fig. 11.8.

The Hermite TFI with cubic blending functions allows it to prescribe additionally the slopes of the grid lines at the boundaries. The Hermite TFI results if we use $L = M = N = 2$ and $P = Q = R = 1$ in Eqs. (11.2) and (11.4). However, it is also possible to mix linear and Hermite interpolations in different computational coordinates. A description of the Hermite TFI and further extensions to the TFI technique (e.g., grid spacing control) can be found in Ref. [20, Chapter 3].

11.1.3 Elliptic grid generation

Grid generation approaches based on elliptic PDEs are known to produce grids with smoothly varying cell sizes and slopes of the grid lines. Furthermore, elliptic grid generation methods offer the possibility to control the orthogonality and the spacing

near boundaries, which is particularly important for the simulation of viscous flows. Elliptic PDEs in grid generation were introduced first by Thompson et al. [26] in 1974. A detailed description of the numerical implementation in 2D was presented, e.g., in Refs. [27, 28].

In 3D, the system of Poisson equations for the unknown Cartesian coordinates $\vec{r} = [x, y, z]^T$ of the grid points can be written as

$$\begin{aligned} \alpha_{11} \frac{\partial^2 \vec{r}}{\partial \xi^2} + \alpha_{22} \frac{\partial^2 \vec{r}}{\partial \eta^2} + \alpha_{33} \frac{\partial^2 \vec{r}}{\partial \zeta^2} + 2 \left(\alpha_{12} \frac{\partial^2 \vec{r}}{\partial \xi \partial \eta} + \alpha_{13} \frac{\partial^2 \vec{r}}{\partial \xi \partial \zeta} + \alpha_{23} \frac{\partial^2 \vec{r}}{\partial \eta \partial \zeta} \right) \\ = -\frac{1}{J^2} \left(P \frac{\partial \vec{r}}{\partial \xi} + Q \frac{\partial \vec{r}}{\partial \eta} + R \frac{\partial \vec{r}}{\partial \zeta} \right), \end{aligned} \quad (11.6)$$

where P , Q , and R denote the *control functions*. The metric coefficients α are given by the relations

$$\begin{aligned} \alpha_{11} &= \left(\frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \eta} \right) \left(\frac{\partial \vec{r}}{\partial \zeta} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right) - \left(\frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right)^2, \\ \alpha_{22} &= \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \xi} \right) \left(\frac{\partial \vec{r}}{\partial \zeta} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right) - \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right)^2, \\ \alpha_{33} &= \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \xi} \right) \left(\frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \eta} \right) - \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \eta} \right)^2, \\ \alpha_{12} &= \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right) \left(\frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right) - \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \eta} \right) \left(\frac{\partial \vec{r}}{\partial \zeta} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right), \\ \alpha_{13} &= \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \eta} \right) \left(\frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right) - \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right) \left(\frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \eta} \right), \\ \alpha_{23} &= \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right) \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \eta} \right) - \left(\frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \zeta} \right) \left(\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \xi} \right). \end{aligned} \quad (11.7)$$

In Eq. (11.7), the terms in brackets represent scalar products. The inverse of the determinant of the coordinate transformation Jacobian (J^{-1}) is evaluated according to Eq. (A.19). It should be noted that by setting $P = Q = R = 0$, Eq. (11.6) reduces to the Laplace equation. The inherent smoothing properties of the Laplace equation can be utilized to improve the quality of an algebraically generated grid. However, the point distribution in the interior field cannot be controlled.

Elliptic equations for the generation of 2-D or surface grids are easily derived from Eq. (11.6) by omitting the derivatives with respect to the ζ -direction. We obtain

$$\begin{aligned}\alpha_{11} \frac{\partial^2 x}{\partial \xi^2} - 2\alpha_{12} \frac{\partial^2 x}{\partial \xi \partial \eta} + \alpha_{22} \frac{\partial^2 x}{\partial \eta^2} &= -\frac{1}{J^2} \left(P \frac{\partial x}{\partial \xi} + Q \frac{\partial x}{\partial \eta} \right), \\ \alpha_{11} \frac{\partial^2 y}{\partial \xi^2} - 2\alpha_{12} \frac{\partial^2 y}{\partial \xi \partial \eta} + \alpha_{22} \frac{\partial^2 y}{\partial \eta^2} &= -\frac{1}{J^2} \left(P \frac{\partial y}{\partial \xi} + Q \frac{\partial y}{\partial \eta} \right).\end{aligned}\quad (11.8)$$

The metric coefficients in Eq. (11.8) become

$$\begin{aligned}\alpha_{11} &= \left(\frac{\partial x}{\partial \eta} \right)^2 + \left(\frac{\partial y}{\partial \eta} \right)^2, \\ \alpha_{12} &= \frac{\partial x}{\partial \xi} \frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi} \frac{\partial y}{\partial \eta}, \\ \alpha_{22} &= \left(\frac{\partial x}{\partial \xi} \right)^2 + \left(\frac{\partial y}{\partial \xi} \right)^2,\end{aligned}\quad (11.9)$$

and the inverse of the determinant of the coordinate transformation Jacobian J^{-1} is defined by Eq. (A.25).

The generation of a boundary orthogonal grid requires the specification of appropriate boundary conditions during the solution of Eq. (11.6) or (11.8). Basically, we can apply either *Neumann* or *Dirichlet* conditions. Neumann boundary conditions allow it to prescribe directly the intersection angle between a grid line and the boundary. In this case, the control functions are not required ($P = Q = R = 0$). However, the point distribution on the boundary and the spacing cannot be controlled. In fact, the boundary nodes will be redistributed automatically to match the given grid line skewness. Hence, the approach is not suited for viscous wall boundaries. More details regarding the Neumann condition in elliptic grid generation are provided, e.g., in Ref. [20, Chapter 6].

Dirichlet boundary conditions are used in cases where the positions of the boundary points have to stay fixed. The control functions are then employed to achieve the desired intersection angles and spacing. The effects of the control functions on the grid are presented in Fig. 11.11 for the 2-D case. As we can see, negative values of P in Eq. (11.8) cause the lines $\xi = \text{const.}$ to rotate to the left since the boundary nodes remain fixed. On the other hand, $Q < 0$ shifts the lines $\eta = \text{const.}$ nearer to the boundary. The values of the control functions are computed first at the respective boundaries from the difference between the prescribed and the actual skewness and grid spacing. Then, the boundary values of P , Q , and R are interpolated in the interior of the domain and the system Eq. (11.6) or (11.8) is solved. The procedure is repeated until the required grid properties are achieved. This approach based on Dirichlet conditions was developed in 2D by Sorenson [27] and by Thomas and Middlecoff [29]. Later it was extended to 3D by Sorenson [30] and Thompson [31].

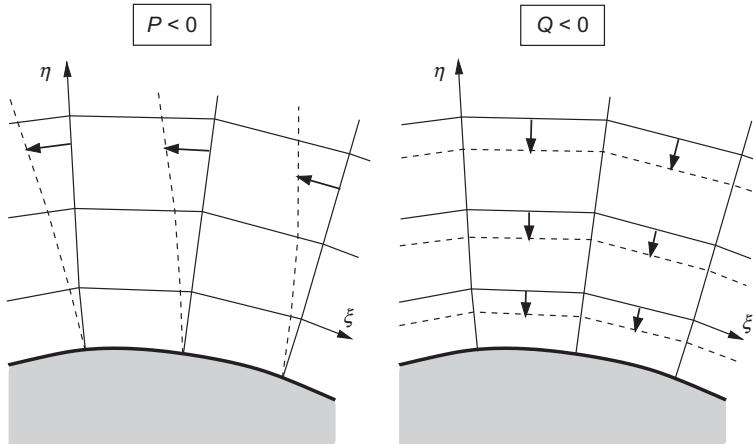


Figure 11.11 Effects of the control functions in 2-D elliptic grid generation at $\eta = \text{const.}$ boundary. P controls the skewness and Q the spacing of the grid.

The elliptic equations (11.6) or (11.8) are usually discretized using second-order central finite differences. The resulting set of linear algebraic equations can be solved by any standard technique, e.g., by the Gauss-Seidel relaxation scheme accelerated by multigrid. The values of the control functions are updated in an outer iteration. In order to increase the robustness of the procedure, it is advisable to carry out several iterations with the Laplace equations ($P = Q = R = 0$), in order to smooth the initial (usually algebraic) grid [32].

The effect of the elliptic equations is demonstrated in Fig. 11.12. The grid around an airfoil is generated first by using the TFI methodology from the previous subsection. The result is shown at the top of Fig. 11.12. The algebraic grid is then smoothed by the Laplace equation, i.e., Eq. (11.8) with $P = Q = 0$ is employed. As we can see, the Laplace smoothing improves the grid around the leading edge of the airfoil. However, the spacing of the grid normal to the surface cannot be controlled. This is only possible with the aid of the control functions $P(\xi, \eta)$ and $Q(\xi, \eta)$, as it is presented at the bottom of Fig. 11.12. The control functions also ensure that the grid lines are orthogonal to the surface of the airfoil. Further examples of elliptically generated grids are shown in Figs. 11.1, 11.2, and 11.6, respectively.

11.1.4 Hyperbolic grid generation

Hyperbolic PDEs are suitable for grid generation in cases where the shape of the outer boundary need not to be exactly controlled. In order to generate the grid, an initial point distribution has to be prescribed. Then, the grid is build by marching a given distance in the normal direction from a previous to a new layer of grid points. The application

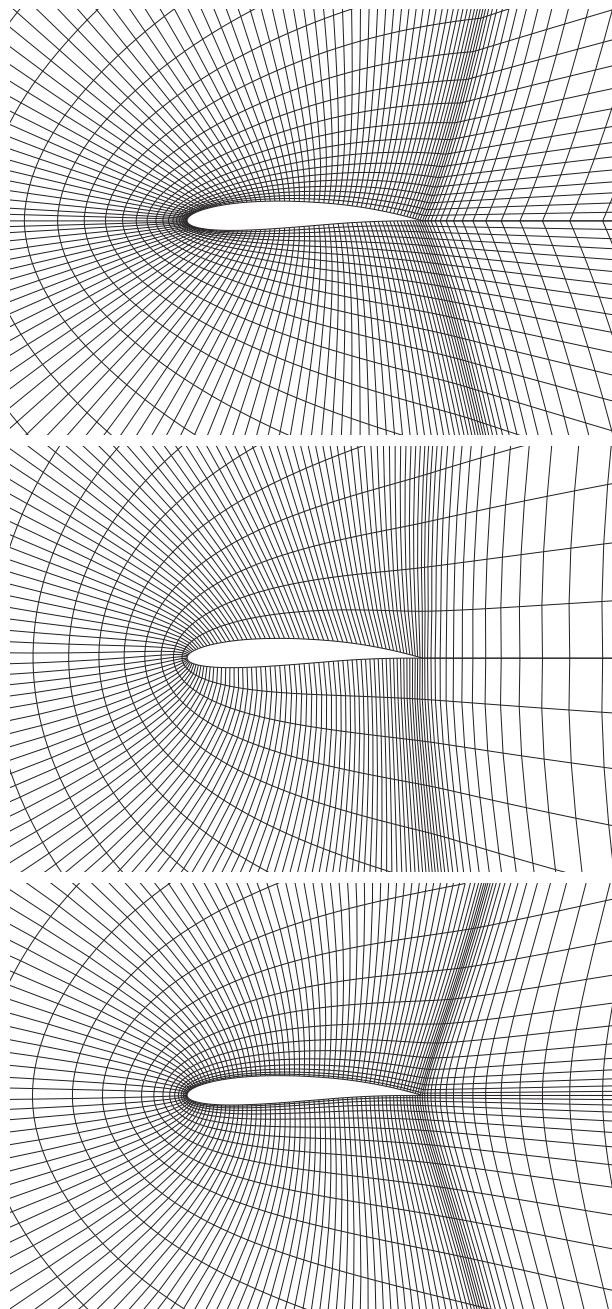


Figure 11.12 Sequence of grids generated for the same airfoil (top to bottom): algebraically by TFI, by the Laplace equation ($P = Q = 0$), and by the elliptic equation (11.8).

of hyperbolic PDEs for grid generation was proposed by Starius [33] and by Steger and Chaussee [34]. A more recent discussion of the hyperbolic grid generation methodology can be found in Ref. [20, Chapter 5]. Various extensions and improvements of the scheme were described in Refs. [35–42].

The hyperbolic equations for the generation of a volume grid read

$$\begin{aligned}\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \zeta} &= 0, \\ \frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \zeta} &= 0, \\ \frac{\partial \vec{r}}{\partial \zeta} \cdot \left(\frac{\partial \vec{r}}{\partial \xi} \times \frac{\partial \vec{r}}{\partial \eta} \right) &= \Omega,\end{aligned}\tag{11.10}$$

where $\vec{r} = [x, y, z]^T$ denotes the Cartesian coordinates of the grid points, ξ, η, ζ stand for the computational coordinates (Eq. (11.1)), and Ω is the user-specified cell volume. Furthermore, we assumed in Eq. (11.10) that the surface $\zeta = \text{const.}$ represents the initial state. The first two relations in Eq. (11.10) represent orthogonality conditions ($\xi = \text{const.}$ and $\eta = \text{const.}$ orthogonal to $\zeta = \text{const.}$ plane). The last relation in Eq. (11.10) guarantees that the cell volume becomes equal to Ω . This opens the possibility to control the spacing between the grid layers.

The 2-D formulation of the hyperbolic equations can be written as

$$\begin{aligned}\frac{\partial x}{\partial \xi} \frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi} \frac{\partial y}{\partial \eta} &= 0, \\ \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi} \frac{\partial x}{\partial \eta} &= \Omega,\end{aligned}\tag{11.11}$$

where Ω denotes now the prescribed cell area. The initial point distribution is given here on the curve $\eta = 0$.

The hyperbolic grid generation equations (11.10) or (11.11) are discretized with respect to the ξ and η coordinate (in 3D) or the ξ coordinate (in 2D) using second-order central differences. The marching in the ζ -direction (Eq. (11.10)) or in the η -direction (Eq. (11.11)) is carried out by a first-order implicit scheme (ADI scheme—see Section 6.2.3). In this way, the marching step can be selected based only on the desired grid spacing. The implicit operator is inverted using a standard tridiagonal solver. Artificial dissipation terms (second differences) have to be added to the discretized equations in order to stabilize the marching procedure. The solution of the hyperbolic equations (11.10) or (11.11) is faster and usually also easier than that of the elliptic system from the previous subsection.

11.2 UNSTRUCTURED GRIDS

Unstructured grids are typically composed of triangles in 2D and of tetrahedra in 3D. However, nowadays it becomes increasingly popular to build unstructured grids from various element types. For example, hexahedra or prisms are employed to discretize boundary layers. The rest of the flow domain is then filled with tetrahedra. Pyramids are used as transitional elements between the hexahedra or the prisms and the tetrahedra. Hence the name *mixed element* grids. The advantages of structured hexahedral grids are the preserved accuracy in the wall normal direction for highly stretched viscous grids as well as the reduced number of elements, edges, and faces as compared to a tetrahedral grid. On the other hand, the desirable feature of unstructured tetrahedral grids is the capability to discretize complex geometries (like in Figs. 11.3 and 11.4) fast and with a minimum user intervention. Mixed grids seek to combine the advantages of both approaches.

In the case of unstructured grids, nodes and grid cells are quasi-randomly ordered, i.e., neighboring cells or grid points cannot be directly identified by their indexes (cf. Fig. 3.3). This leads to tremendous geometric flexibility of unstructured grids, since the grid does not need to conform to any predetermined topology. Furthermore, adaptation of the grid to the physical solution—grid refinement or coarsening—is much easier to accomplish on unstructured than on structured grids.

Unstructured grid generation methodologies for CFD applications are mostly based on either an

- *Delaunay*, or
- *advancing-front*

method. Both approaches can also be combined together. Depending on the base methodology, we speak either of *advancing-front Delaunay* [43–45] or of *frontal Delaunay* schemes [46–48]. Besides the both standard techniques, there are also other methods like the so-called *bubble packing* algorithm [49–54]. Here, we shall describe only the basic features of the Delaunay and the advancing-front method. A survey of both approaches is contained, e.g., in Refs. [20, 55, 56].

The Delaunay approach refers primarily to a particular way of connecting grid points to form triangles in 2D and tetrahedra in 3D. The most important feature of the Delaunay triangulation is that the circumcircle (or the circumsphere in 3D) of any triangle (tetrahedron) contains no other grid point. The consequence of the empty circumcircle criterion is that in 2D the minimum angle is maximized for all triangles (*max-min triangulation*). Thus, a high grid regularity can be achieved.

The idea of the advancing-front method is to generate the grid sequentially element by element starting from a known boundary discretization (surface grid). The open surfaces of the elements constitute the front. New triangles (tetrahedra) are constructed by placing points ahead of the front. In this way, the front moves through the domain until

all cavities are filled. The point placement is controlled by the so-called *background grid*. The advancing-front approach offers the advantages of smooth point distribution and implicitly assured boundary integrity. However, it is slower than the Delaunay method.

11.2.1 Delaunay triangulation

The Delaunay triangulation is based on a methodology proposed by Dirichlet in 1850 for the unique subdivision of space into a set of packed convex regions [57]. Given a set of points, each region represents the space around the particular point, which is closer to that point than to any other. The regions form polygons (polyhedra in 3D) which are known as the *Dirichlet tessellation* or the *Voronoi diagram* [58]. If we connect point pairs which share some segment (face) of the Voronoi diagram by straight lines, we obtain the Delaunay triangulation [59]. The triangulation defines a set of triangles (tetrahedra in 3D), which cover the convex hull of the points. This is displayed in Fig. 11.13. The Delaunay triangulation is the dual of the Voronoi diagram. The nodes of the Voronoi polygons are in 2D the centers of circumcircles of the triangles. In 3D, the nodes represent the centers of circumspheres of the tetrahedra. This implies that the circumcircle of every triangle (circumsphere of every tetrahedron) contains no other point from the set in its interior.

As we already stated, the Delaunay method constitutes a particular way of connecting grid points. The positions of the points must be determined by some other technique. Therefore, one popular approach for the construction of a Delaunay grid is to insert sequentially nodes into an initial triangulation. The grid is then locally retriangulated in order to fulfill the empty circumcircle (circumsphere) criterion. The *incremental point-insertion* strategy can be described by the following steps:

1. Discretize the boundaries of the physical domain.
2. Generate an initial Delaunay grid which covers all boundary nodes. This can be either a triangulation of the boundary nodes itself (Fig. 11.14a) or a surrounding quadrilateral (hexahedron in 3D), which is decomposed into triangles (tetrahedra) as displayed in Fig. 11.14b.

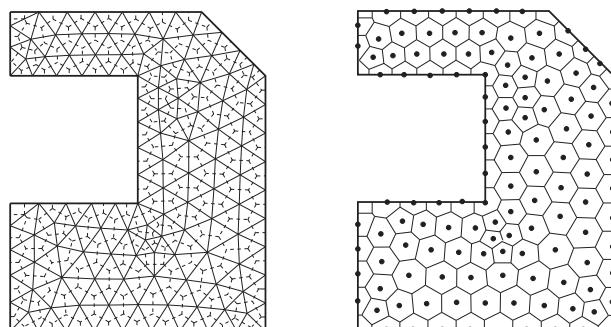


Figure 11.13 The Delaunay triangulation (left) and the Voronoi diagram (left as dashed line, right as solid line).

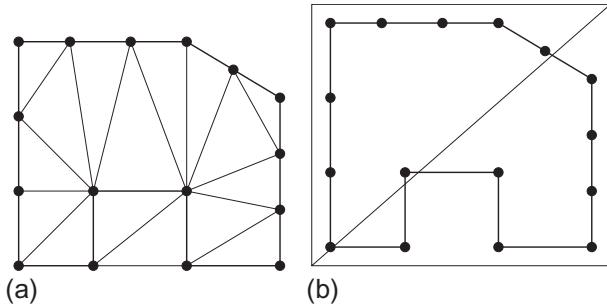


Figure 11.14 Possibilities for initial Delaunay grid in 2-D: triangulation of boundary nodes (a) and quadrilateral divided into two triangles (b).

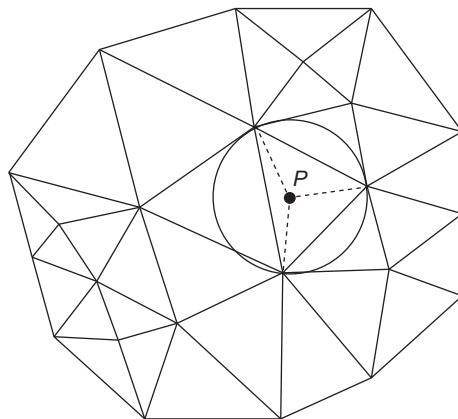


Figure 11.15 Insertion of new node P into valid Delaunay triangulation. The node is located at the center of the circumcircle.

3. If not already done, insert all boundary nodes into the initial triangulation using a Delaunay-conforming technique.
4. Build a list of all triangles (tetrahedra) in the grid which violate some size or quality measure. Order the list to start with the worst element.
5. Place a new point at the circumcenter [60–62] (see Fig. 11.15) or at the Voronoi segment [63, 64] of the first element in the list and locally retriangulate the grid. Check each new element and add it to the list if not in accordance with the size/quality measure.
6. If there are still elements in the list, go to step 5.
7. Delete elements outside of the domain and recover the boundaries.
8. Check the grid quality (Section 11.2.5). Smooth the grid and/or swap edges if necessary.

Alternatively, boundary recovery (step 7) can be carried out after the step 3.

As we can see, the core task of the Delaunay-grid generation is the insertion of a new point into a valid triangulation (steps 3 and 5). Several algorithms were proposed for this purpose. The most popular approaches are due to Green and Sibson [65], Bowyer [66], and Watson [67]. We shall describe Watson's algorithm further below.

In order to accomplish the steps 4 and 5, we have to assess the elements with respect to some appropriate measure. This can be either the quality of the element (minimum angle, aspect ratio—see Refs. [68, 69]), or the size (volume, edge length) of the element. The quality of an element can be assessed directly from its geometry. However, the size measure has to be formulated as a function of the spatial position within the domain. Several approaches are possible:

- Specification of an analytical function (e.g., size is proportional to distance from the body).
- Interpolation of size distribution from boundaries using the initial triangulation (see, e.g., Ref. [20, Chapter 1, pp. 17–20]).
- Interpolation on background grid based on quadtree (octree in 3D) structure [70–75].
- Specification of sources (point, line, etc.) inside the domain (see, e.g., Ref. [20, Chapter 1, pp. 20–22]).

Furthermore, the points can be placed with the aid of the advancing-front technique [45, 46].

Watson's algorithm

The point insertion and retriangulation method of Watson consists of the following steps [67]:

1. Locate the element which contains the inserted point P (Fig. 11.15).
2. Find all elements whose circumcircle (circumsphere in 3D) is intersected by point P . This situation is sketched on the left-hand side of Fig. 11.16.

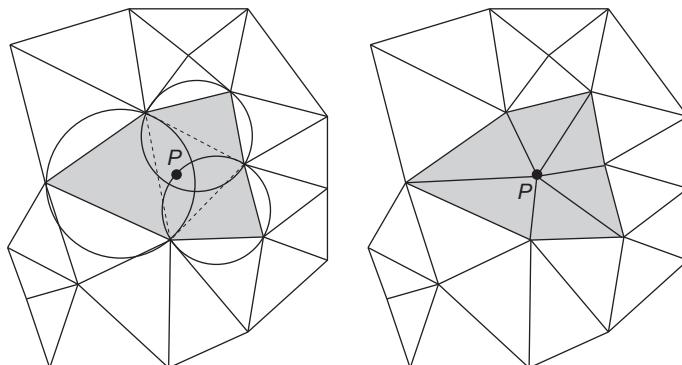


Figure 11.16 Watson algorithm: deletion of invalid triangles (left) and retriangulation of the convex cavity (right).

3. Delete all intersected elements from the triangulation.
4. Form new elements by connecting the points on the boundary of the convex cavity to the point P (right-hand side of Fig. 11.16).

Data structure, which is particularly suitable for the algorithm of Watson, results when we store for each element:

- indexes of the forming nodes;
- pointers to neighbors which share a common face with the element;
- circumcenter and radius of the circumcircle (circumsphere).

This data structure allows for an efficient search of intersected elements in the step 2. We start the search with the neighbors of the element which contains the inserted point. Then, we proceed to the neighbors of these neighbors and so on. We stop the search in a particular direction, if the circumcircle (circumsphere) of the element is not intersected. The geometrical properties of the Delaunay triangulation make sure that the neighbors of this element are not intersected. It is also guaranteed that all elements being involved are localized by this simple strategy [67, 76]. The numerical effort of Watson's algorithm is of the order of $N \log(N)$, where N is the number of grid points. This results in a very fast grid generation methodology.

Constrained Delaunay triangulation

The Delaunay triangulation does not automatically take care of prescribed edges and faces, like those on the boundaries of the physical domain. This is the purpose of the so-called *constrained Delaunay* triangulation [77]. The restoration of boundary edges in 2D is sketched in Fig. 11.17. Depending on the situation, either edge swapping or retriangulation is required. The constrained Delaunay triangulation leads in 2D always to a valid grid. However, this cannot be guaranteed in 3D. The recovery of the boundary discretization in 3D has to be conducted in two steps—first for boundary edges and second for boundary faces [56]. In some cases, additional points (the so-called *Steiner points*) have to be inserted on the boundaries [78–81]. Otherwise, the cavity cannot

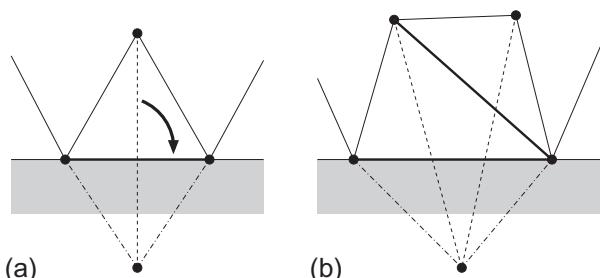


Figure 11.17 Insertion of missing boundary edge: by edge swapping (a) and by deleting intersected triangles and retriangulating the cavity (b). Swapped or deleted edges are represented by dashed lines, new edges by thick solid lines. The triangles outside of the domain (dash-dotted line) are removed.

be retetrahedralized. After the boundary edges (and faces) are recovered, the elements outside of the flow domain can be deleted.

11.2.2 Advancing-front method

The advancing-front methodology was introduced first by Peraire et al. [82, 83], and by Löhner and Parikh [84] by the end of the 1980s. The individual steps of this grid generation scheme can be summarized as follows:

1. Discretize the boundaries of the physical domain (generate surface grids).
2. Generate a list of edges (faces in 3D) which represent the front. Initially, these are the boundary edges (faces). Sort the list in the order of increasing edge (face) size [55]. This strategy helps to generate smoothly varying elements.
3. Select the first edge (face) of the list and place a new point P_0 in the normal direction above the center of the front edge (face). The situation is displayed in Fig. 11.18. The distance d into the domain is governed by the local values of the size-distribution function (see, e.g., Ref. [56]).
4. Define a circle (sphere) with radius r centered at the point P_0 . The radius r depends on the local grid size.
5. Determine all points which are located within the circle (sphere).
6. If there are no intersected points, generate a new element with the point P_0 . Otherwise, order the intersected points with respect to their distance to P_0 (i.e., P_1, P_2, P_3). Form elements with the points and accept the first one which does not intersect any other element and satisfies given quality measure(s).
7. Delete the current front edge (face) and add the newly formed edges (faces) to the list. Sort the list again.
8. Continue with step 3 until the list is empty.

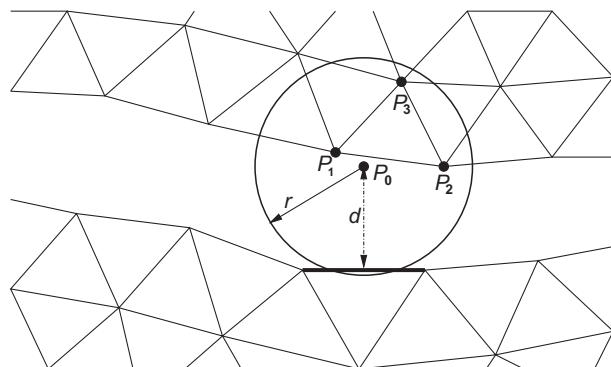


Figure 11.18 Insertion of a new node P_0 in the 2-D advancing-front method. The active edge is denoted by a thick line.

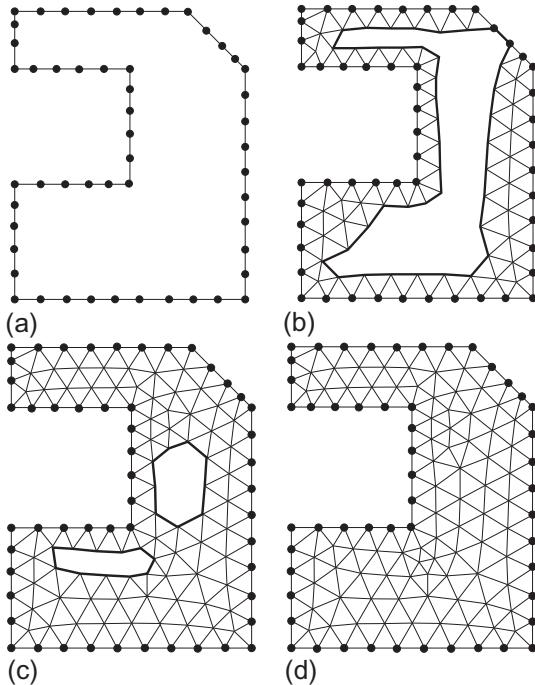


Figure 11.19 Different stages of grid generated using the advancing-front technique. The actual front is represented by a thick solid line.

9. Check the quality of the grid (see [Section 11.2.5](#)). Smooth the grid and/or swap edges if required.

Different stages of the advancing-front process are shown in [Fig. 11.19a–d](#) for an exemplary 2-D configuration.

The distribution of the element size in the domain is mostly governed by the point density on the boundaries. Additionally, the distance d in step 3 can be controlled by placing sources of various type—point, line, cylinder, etc.—inside the domain (see, e.g., Ref. [20, Chapter 1, pp. 20–22]). The local element size can be obtained from a background grid based on the quadtree (octree in 3D) data structure [70–75]. Another possibility is to interpolate the sizes by employing Delaunay triangulation of the boundary nodes [46–48, 85], or to use a Cartesian background grid [86]. The quadtree (octree) data structure can also be used to search efficiently for nearby points (step 5) as well as to check for possible intersections between the elements. A simple test for the intersection between fronts based on spring analogy was devised in Refs. [3, 87]. A detailed comparison of different search algorithms can be found in Ref. [88].

The advantages of the advancing-front method as compared to the Delaunay triangulation scheme are the implicitly retained boundary discretization and the better

control over element sizes and grid smoothness. Furthermore, the Delaunay approach is more sensitive to round-off errors than the advancing-front method. However, the point searching and intersection checking algorithms require significant numerical effort. A combination between the advancing-front and the Delaunay methodology is often employed in CFD, particularly in 3D [43–48, 89]. A quite detailed description of the advancing-front methodology can be found in Ref. [90].

11.2.3 Generation of anisotropic grids

In Sections 11.2.1 and 11.2.2, we described two methodologies for the generation of *isotropic* triangular or tetrahedral elements. This is appropriate for the simulation of inviscid flows. However, an accurate solution of the Reynolds-averaged Navier-Stokes equations requires high spatial resolution in the direction across boundary layers and wakes. Thus, in the case of high Reynolds-number flows, isotropic grid would result in an excessively large number of elements. Clearly, we have to employ *anisotropic*, high aspect-ratio elements in the viscous flow regions. Basically, we can utilize quadrilaterals (hexahedra or prisms in 3D), which allow quite naturally for stretching. This leads to mixed-element grids which are very popular today (see next subsection). Of course, it is possible to decompose the hexahedra or prisms in order to obtain a purely tetrahedral grid. On the other hand, we can generate directly stretched triangular (tetrahedral) grids. In any case, the grid generation scheme should prevent the creation of obtuse triangles (tetrahedra) of the form sketched in Fig. 11.20a. It was demonstrated in Ref. [91] that such elements lead to an exceedingly large truncation error of the spatial discretization. Therefore, it is advised to generate right-angle elements like those displayed in Fig. 11.20b and c. However, depending on the type of the control volume, right-angle elements may also induce discretization errors [92, 93] (cf. Section 5.2.3).

Stretched Delaunay triangulation

The generation of stretched triangular (tetrahedral) elements requires the definition of the magnitude and the direction of the stretching in the physical domain. Background grid has to be used for this purpose in the case of stretched Delaunay triangulation.

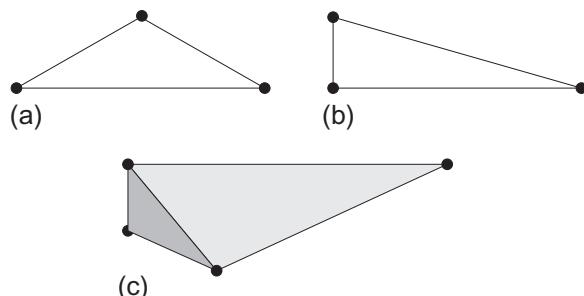


Figure 11.20 High aspect-ratio elements: obtuse triangle (a), right-angle triangle (b), and right-angle tetrahedron (c).

The same point insertion techniques can be employed as for the generation of isotropic Delaunay grids. However, the circumcircle (circumsphere) criterion is replaced by a condition of empty circumellipse (cirumellipsoid). The ellipse (ellipsoid) is oriented in the local direction of the stretching vector and the magnitude of stretching is reflected by the ratio of the axes [94–96]. An alternative proposal is the bubble packing algorithm with ellipsoids instead of spheres [51–53].

Advancing-layers method

The specification of the stretching vector in the anisotropic Delaunay triangulation is quite involved for geometrically complex domains. Therefore, the so-called *advancing-layers* method proposed by Pirzadeh [3, 4, 97, 98] and others [99, 100] is more widely utilized. The advancing-layers method is similar to the hyperbolic structured grid generation technique (Section 11.1.4). It can also be considered as a modified advancing-front technique.

The generation of stretched triangular (tetrahedral) grids by the advancing-layers approach proceeds according to the following steps (see Fig. 11.21):

1. Triangulate all boundary surfaces.
2. Compute approximate normal vectors at the boundary nodes.
3. Place grid points along the surface normals and form layers of quadrilateral (prismatic in 3D) elements of increasing thickness.
4. Decompose each quadrilateral (prism) into two triangles (three tetrahedra). The connectivity pattern requires particular attention in 3D.
5. Continue growing each stack of elements until the front intersects either itself or another front, or until the cell aspect ratio becomes close to unity.

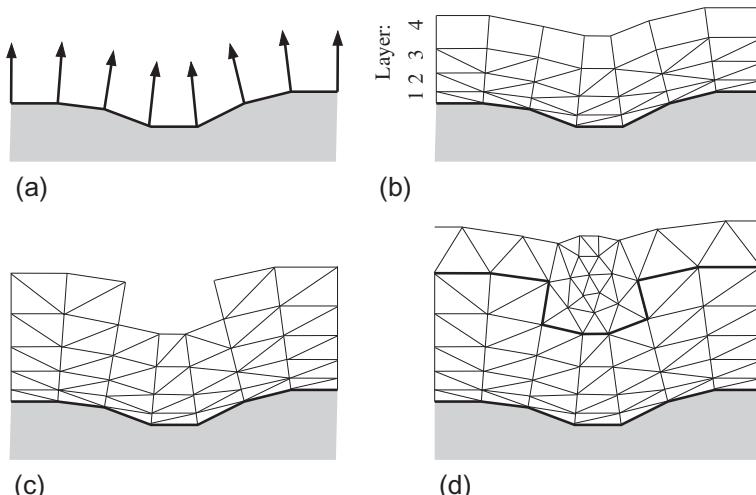


Figure 11.21 Steps of the advancing-layer method: computation of boundary normals (a), generation of a new layer and subdivision of the elements (b), finished stretched grid (c), and generation of isotropic elements in the rest of the domain (d).

The rest of the flow domain is then filled with isotropic tetrahedra. Usually the advancing-front methodology is employed, which starts from the surface represented by the last layer of boundary cells. Figures 5.2 and 11.4 show examples of grids generated using the above procedure.

In principle, two options exist for the evaluation of the normal vectors at the boundary nodes (step 2). First, if the bounding surfaces are described by analytical functions like NURBS (Nonuniform Rational B-Splines) [101], the normal derivatives are easily calculated. A grid generation method of this type was described in Ref. [102]. However, care has to be taken at patch interfaces (e.g., like at trailing edges).

The second, more widespread approach, is to use the surface triangulation. This can be done quite easily in 2D, where it is sufficient to average the normals of the boundary edges which join at the particular node. However, such simple averaging is not appropriate in 3D. The reason is that the boundary node can be shared by any number of triangular faces. Hence, the normal vector can become biased depending on the triangulation. The problem is particularly severe at sharp corners and bends. The necessary condition for an optimal normal vector is given by the so-called *visibility criterion* [103, 104]. It states that a point placed along the normal direction has to be equally visible from all triangular faces sharing the boundary node. Various procedures based on the visibility criterion were devised for the construction of the normal vectors. For details see, e.g., Refs. [3, 103–106]. The normal vectors are smoothed, in order to prevent abrupt changes of the marching direction and crossing of the grid lines. In general, weighted-Laplacian type of smoothing is employed [104, 105], i.e.,

$$\vec{n}_i = \omega \vec{n}_i^{(0)} + \frac{(1 - \omega)}{\sum_j 1/|\vec{r}_{ij}|} \sum_j \frac{\vec{n}_j}{|\vec{r}_{ij}|}. \quad (11.12)$$

In Eq. (11.12), \vec{n}_i and $\vec{n}_i^{(0)}$ denote the new and the initial node-normals, respectively. Furthermore, \vec{n}_j represents the normal vectors at the adjacent nodes and $|\vec{r}_{ij}|$ is the distance between the nodes i and j . The weighting factor ω depends on the surface curvature. It takes small values in concave regions and large values in convex ones. The smoothing is applied separately to each layer. A common procedure is to use the initial node-normals for the first few layers and then to increasingly smooth the normal vector. The reason is that a boundary orthogonal grid helps to reduce the discretization error of a spatial scheme.

The size of the marching step is calculated based on a user-specified thickness of the first layer and on stretching rate in the normal direction. Thus, for example, the spacing can be obtained from Ref. [3]

$$\Delta n_k = \Delta n_0 \left[1 + a(1 + b)^{k-1} \right]^{k-1}, \quad (11.13)$$

where Δn_k represents the spacing of the k th layer, Δn_0 is the given first-layer thickness, and $0.04 \leq a \leq 0.2$ and $0 \leq b \leq 0.07$ stand for the stretching parameters. The marching step Δn_k can be additionally modified at convex and concave corners [106]. It is also possible to increase Δn_k in the direction of growing boundary layer to keep γ^+ constant.

Each stage of the advancing-layers algorithm generates in 2D a layer of quadrilaterals and in 3D a layer of prisms (or hexahedra if the surface is discretized with quadrilaterals). Whereas the quadrilaterals of the layers can be easily divided into two triangles (along the shortest diagonal), the decomposition of prisms into three tetrahedra requires some care. The point is that faces between the prisms have to be divided in the same way. An integer-based approach for the consistent decomposition of prisms was suggested in Ref. [3]. A simpler scheme was presented in Ref. [107]. One further decomposition methodology was reported in Ref. [108].

Combined advancing-normal/Delaunay method

A further possibility for the generation of stretched triangular (tetrahedral) grids is offered by a modified advancing-front method (called here *advancing-normal point placement*) combined with the Delaunay triangulation. Corresponding approach was proposed by Müller [109] for 2-D grids. Marcum [110, 111] as well as Sharov and Nakahashi [106] developed 3-D versions of the algorithm. The procedure starts with the volume triangulation of the boundary nodes. The triangulation is employed as a background grid. It also serves as an efficient search structure. The boundaries can be recovered [106], but it is not necessary at this stage [110, 111]. Next, points are inserted in regions of the stretched grid using techniques similar to the advancing-layers method. The background triangulation is utilized for the determination of nearby fronts and for intersection checking. In the last stage, the inflated surface is employed as a new boundary for the generation of isotropic elements by any standard Delaunay technique.

Stretched surface grids

In regions, where the flow exhibits a strong directionality, anisotropic grids can substantially reduce the number of triangular faces and hence of volume elements, without impairing the solution accuracy (see Fig. 11.22). For example, a simulation with RANS does not require high grid resolution in the span-wise direction of a wing or a blade. The same holds for a fuselage in the axial direction. Savings in the number of surface triangles between factor 2 and 6 were reported in Refs. [112, 113]. However, as demonstrated in Fig. 11.22, it is important to orient the surface elements properly, particularly in areas of high curvature. Otherwise, the true surface contour becomes poorly represented and the flow solution will be falsified. Stretched surface grids can be generated by any of the above methodologies for anisotropic grids (for a general discussion of surface grid generation see, e.g., Refs. [16, 75, 114]). The stretching ratio and orientation is usually specified through a line source. Another possibility consists of

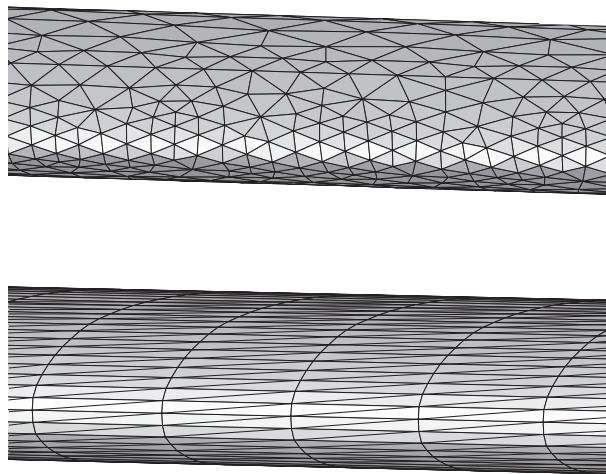


Figure 11.22 Unstructured surface grid at leading edge: isotropic with no specific orientation (top); stretched and oriented in span-wise direction (bottom).

generating first an isotropic grid in parametric coordinates. Then, the grid is transformed into the physical space by using stretching functions.

11.2.4 Mixed-element/hybrid grids

Nowadays, it became quite popular to discretize the flow domain by using grids, which consist of different elements (prisms, tetrahedra, pyramids, etc.). Such grids are referred to as *mixed* grids or *mixed-element* grids. Another idea is to compose the grid of structured and unstructured zones [115–122]. In this case we speak of *hybrid* grids. It should be noted that some authors use the term “hybrid grids,” but in reality they mean mixed grids. The motivation behind the use of either the mixed or the hybrid grids is to employ the best suitable elements or grid topology in each flow region. The goals are to increase the accuracy of the simulation and to reduce the computational costs. However, the constraint is that the generation of such grid has to proceed in a largely automatic way. The process has to be sufficiently fast as well.

Hybrid grids are relatively seldom employed in practice. The problem is that two different flow solvers are needed—one structured and one unstructured, which have to be coupled. The effort required not only to write, but primarily to update and to maintain two CFD codes is significant. It is therefore necessary to develop a special data and program structure. The classical application of hybrid grids consists of the discretization of the near-wall regions using structured grids (hexahedral or

possibly semi-structured prismatic grids), which allow for an easier implementation of higher-order spatial schemes, implicit methods [123], or multigrid [124, 125]. The rest of the domain is filled with an unstructured grid, which offers considerable advantages in geometrically complex areas. The idea of hybrid grids can also be utilized in the case of rotor-stator interaction, like encountered in turbomachinery. Here, the interface between the structured grids around the rotor and the stator consists of a slice of unstructured grid (similar to the idea in Ref. [126]). The unstructured grid is regenerated with each rotor movement. In this way, the accuracy and the conservation properties of the discretization scheme are retained across the interface.

Mixed grids may offer a larger flexibility than the hybrid grids, especially for complex geometries. Mixed grids can also be more easily generated and in particular refined. However, the challenge is to develop data structures and numerical schemes for the flow solver, which can handle varying element types in a seamless and efficient way. We discussed some of the associated problems in Chapter 5. In the following, we briefly present methodologies for the generation of mixed prismatic/tetrahedral and prismatic/tetrahedral/Cartesian grids.

Mixed prismatic/tetrahedral grids

A method for the generation of grids consisting of prismatic elements in the near-wall region and of tetrahedral elements everywhere else was initially developed by Nakahashi [127–129]. The methodology was further pursued by Kallinderis [130, 131], and by Connell and Braaten [107]. The generation of the prismatic cells is conducted using the same techniques as described previously for the advancing-layer or the advancing-normal approach, respectively. The tetrahedra are grown directly from the last layer of prisms. In cases where one of the quadrilateral faces of a prism remained exposed, pyramid is used as transition element to the tetrahedra.

Mixed prismatic/tetrahedral/Cartesian grids

Cartesian grids are composed of squares (cubes in 3D), which are aligned with the Cartesian coordinate axes. Cartesian grids can be generated easily and with low computational effort even for geometrically complex domains. However, the weakness of the Cartesian methods is the accuracy of the flow solution at solid boundaries, which are either curved or not oriented along the Cartesian coordinates. This problem becomes especially serious in the case of RANS simulations, where highly stretched and boundary orthogonal cells are required. Therefore, various authors, e.g., Melton et al. [132], Karman [133], Smith and Leschziner [134], Wang et al. [135], Delanaye et al. [136], and Hitzel et al. [137] proposed to insert body-conforming quadrilaterals (prisms) near wall surfaces. It is possible to create a continuous interface between the inner layers and the outer Cartesian grid by using pyramids and tetrahedra as the transitional elements. On the other hand, the grid generation procedure can be simplified by allowing for hanging nodes and lines. This situation is displayed in Fig. 11.23. If treated correctly,

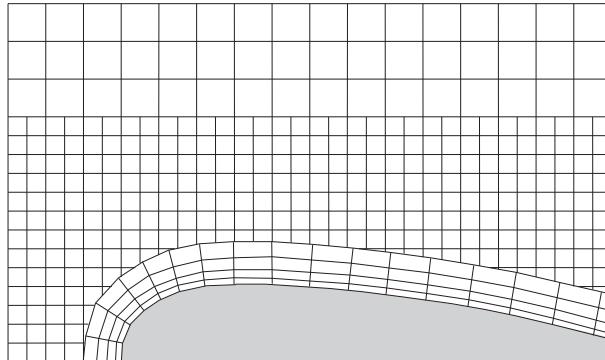


Figure 11.23 Example of mixed Cartesian/stretched quadrilateral grid near surface.

the interchange of fluxes at the interface can be kept conservative [136, 138]. A detailed description of a mixed Cartesian grid approach can be found in Ref. [139]. An interesting idea proposed by Karman [140] consists of the projection of a Cartesian volume grid onto the boundaries, which automatically generates the surface discretization. After that, viscous layers are injected along the solid boundaries.

11.2.5 Assessment and improvement of grid quality

The quality of the grid strongly influences the accuracy of the simulation. This is particularly true for unstructured grids. We mentioned this problem in various places of Chapter 5 (e.g., in Section 5.3.3 on solution reconstruction). Therefore, it is important that the resulting grid consists of elements which are as regular as possible. Furthermore, the cell sizes (and stretching in viscous regions) should vary smoothly over the domain. Of course, the grid should also be fine enough to resolve the relevant geometrical and flow features.

Figure 11.24 shows the shapes of tetrahedra, which should be avoided in the grid: obtuse elements (see also Fig. 11.20a), slivers (elements with four nearly co-planar points), needles, and wedges. The only exception are wedge-like stretched tetrahedra

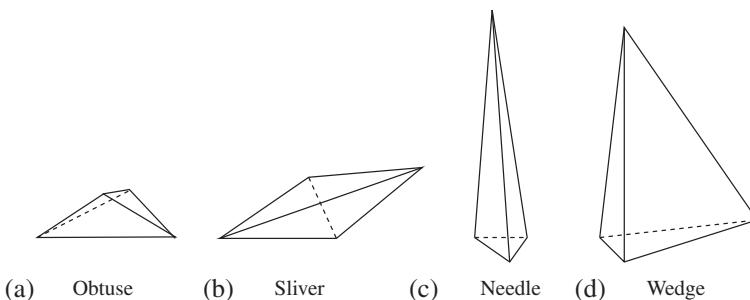


Figure 11.24 Undesirable shapes of tetrahedral elements.

in viscous regions (see Fig. 11.20). They can be avoided only by using prisms. However, since the largest gradient of the flow is in the wall-normal direction (short side of the wedge), the numerical error is not overly disturbing.

Beyond the element angles [111], the following parameters can be employed for tetrahedral grids as quality measures [20, 141] (the numbers are values for an equilateral tetrahedron):

1. radius of circumscribed sphere/radius of inscribed sphere = 3.0
2. maximum edge length/radius of inscribed sphere = 4.8990
3. radius of circumscribed sphere/maximum edge length = 0.6125
4. maximum edge length/minimum edge length = 1.0

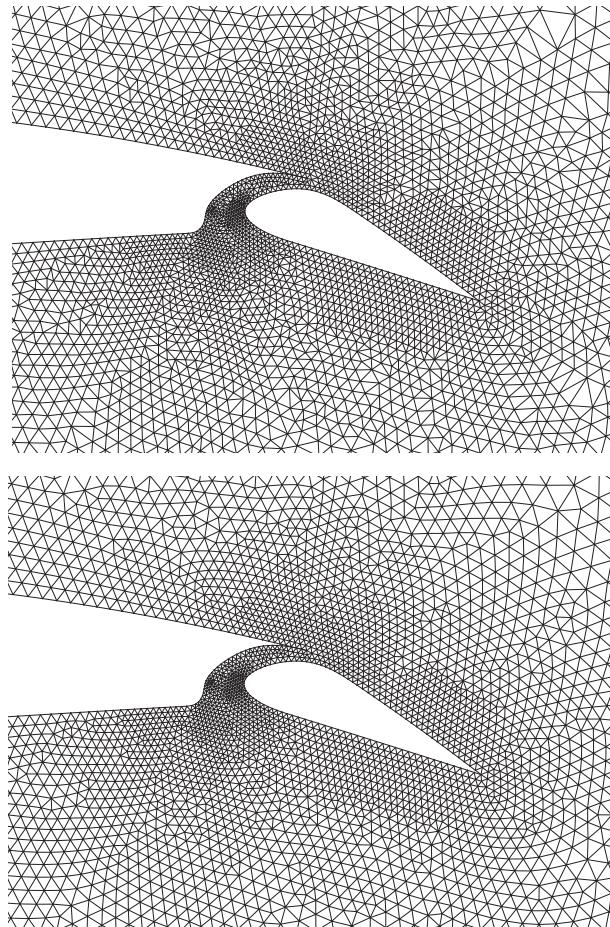


Figure 11.25 Unstructured triangular grid before (top) and after smoothing (bottom).

5. (average element edge length)³/volume = 8.4797
6. (volume)⁴/(sum of areas of all triangular faces)³ = 4.585 × 10⁻⁴.

In order to improve the grid quality, the edge swapping algorithm due to Lawson [142, 143] can be used. Edge swapping is particularly suitable for removing sliver elements from the grid [48].

Grid smoothing

Another technique, which is used quite often in order to improve the grid regularity, is smoothing. Here, the grid nodes are moved by using an approximate Laplacian operator

$$\vec{r}_i^{n+1} = \vec{r}_i^n + \frac{\omega}{N_A} \sum_{j=1}^{N_A} (\vec{r}_j - \vec{r}_i), \quad (11.14)$$

where N_A denotes the number of nodes adjacent to i and ω is the relaxation factor (0.5–1.0 in the interior, 0.25 at points adjacent to the boundaries, 0.0 at the boundary nodes—cf. Ref. [48]). The relation in Eq. (11.14) has to be solved iteratively. The point Gauss-Seidel scheme was preferred in Ref. [47] over the point Jacobi scheme because of the better suppression of negative volumes. An example of unstructured grid smoothed using the Laplacian operator (11.14) is displayed in Fig. 11.25.

REFERENCES

- [1] Brodersen O, Hepperle M, Ronzheimer A, Rossow CC, Schöning B. The parametric grid generation system MegaCads. Proceeding of the 5th international conference on numerical grid generation in computing field simulations. Mississippi: NSF Engineering Center; 1996. p. 353–62.
- [2] Brodersen O, Ronzheimer A, Ziegler R, Kunert T, Wild J, Hepperle M. Aerodynamic applications using MegaCads. Proceedings of the 6th international conference on numerical grid generation in computing field simulations. London, England: University of Greenwich; 1998. p. 793–802
- [3] Pirzadeh S. Three-dimensional unstructured viscous grids by the advancing-layers method. *AIAA J* 1996;34:43–9.
- [4] Pirzadeh S. Progress toward a user-oriented unstructured viscous grid generator. *AIAA Paper 96-0031*; 1996.
- [5] Mavriplis DJ, Pirzadeh S. Large-scale parallel unstructured mesh computations for 3D high-lift analysis. *AIAA Paper 99-0537*; 1999.
- [6] Pirzadeh S. Unstructured grid generation for complex 3D high-lift configurations. Paper No. 1999-01-5557. San Francisco: World Aviation Congress and Exposition, October 1999.
- [7] Digital Representation for Communication of Product Definition Data: IGES 5.2 (Initial Graphics Exchange Specification Version 5.2). US Product Data Association, November 1993, ISBN 978-1-88538900-8.
- [8] <http://www.wiz-worx.com/iges5x/>.
- [9] ISO 10303-21:2002 Industrial automation systems and integration—product data representation and exchange—part 21: implementation methods: clear text encoding of the exchange structure.
- [10] STEP application handbook: ISO 10303 (Version 3 ed.). North Charleston, SC: SCRA, 30 June 2006.
- [11] http://www.wikistep.org/index.php/Main_Page.
- [12] http://images.autodesk.com/adsk/files/autocad_2014_pdf_dxf_reference_enu.pdf.

- [13] Burns M. Automated fabrication. Prentice-Hall, New Jersey; 1993, ISBN 978-0-13-119462-5.
- [14] Chua CK, Leong KF, Lim CS. Rapid prototyping: principles and applications (2nd ed.). Singapore: World Scientific Publishing Co.; 2003, ISBN 981-238-117-1.
- [15] <http://www.ennex.com/~fabbers/StL.asp>.
- [16] Aftosmis MJ, Delanaye M, Haines R. Automatic generation of CFD-ready surface triangulations from CAD geometry. AIAA Paper 99-0776; 1999.
- [17] Bohn JW, Zozny, MJ. Automatic CAD-model repair: shell-closure. Proceedings symposium on freeform fabrication, Department of Mechanical Engineering, University of Texas at Austin; 1992.
- [18] Guéziec A, Taubin G, Lazarus F, Horn W. Cutting and stitching: efficient conversion of a non-manifold polygonal surface to a manifold. IBM-RC-20935, IBM Research Division, Yorktown Heights; 1997.
- [19] Carey GF. Computational grids: generation, adaption, and solution strategies. Washington, DC: Taylor & Francis; 1997.
- [20] Thompson JF, Soni BK, Weatherill NP, editors. Handbook of grid generation. Boca Raton: CRC Press; 1999.
- [21] Baker TJ. Three decades of meshing; a retrospective view. AIAA Paper 2003-3563; 2003.
- [22] Gordon WN, Hall CA. Construction of curvilinear coordinate systems and application to mesh generation. Int J Numer Methods Eng 1973;7:461-77.
- [23] Gordon WJ, Thiel LC. Transfinite mappings and their application to grid generation. In: Thompson JF, editor. Numerical grid generation, North-Holland, p. 171; 1982.
- [24] Thompson JF, Warsi ZUA, Mastin CW. Numerical grid generation: foundations and applications. Amsterdam, Netherlands: Elsevier Science; 1985.
- [25] Soni BK. Two- and three-dimensional grid generation for internal flow applications of computational fluid dynamics. AIAA Paper 85-1526; 1985.
- [26] Thompson JF, Thamess FC, Mastin CW. Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. J Comput Phys 1974;15:299-319.
- [27] Sorenson RL. A computer program to generate two-dimensional grids about airfoils and other shapes by the use of Poisson's equation. NASA TM-81198; 1980.
- [28] Hsu K, Lee SL. A numerical technique for two-dimensional grid generation with grid control at all of the boundaries. J Comput Phys 1991;96:451-69.
- [29] Thomas PD, Middlecoff JF. Direct control of the grid point distribution in meshes generated by elliptic equations. AIAA J 1980;18:652-6.
- [30] Sorenson RL. Three-dimensional elliptic grid generation about fighter aircraft for zonal finite difference computations. AIAA Paper 86-0429; 1986.
- [31] Thompson JF. A general three-dimensional elliptic grid generation system based on a composite block structure. Comp Method Appl Mech Eng 1987;64:377-411.
- [32] Sonar T. Grid generation using elliptic partial differential equations. DFVLR-FB 89-15; 1989.
- [33] Starius G. Constructing orthogonal curvilinear meshes by solving initial value problems. Numer Math 1977;28:25-48.
- [34] Steger JL, Chaussee DS. Generation of body-fitted coordinates using hyperbolic partial differential equations. SIAM J Sci Stat Comput 1980;1:431-7.
- [35] Steger JL, Rizh YM. Generation of three-dimensional body-fitted coordinates using hyperbolic partial differential equations. NASA TM-86753; 1985.
- [36] Steger JL. Generation of three-dimensional body-fitted grids by solving hyperbolic partial differential equations. NASA TM-101069; 1989.
- [37] Dwyer HA. A geometric interpretation of hyperbolic grid generation. AIAA Paper 94-0315; 1994.
- [38] Sethian JA. Curvature flow and entropy conditions applied to grid generation. J Comput Phys 1994;115:440-54.
- [39] Jeng YN, Shu YL. The grid combination method for the hyperbolic grid solver in regions with enclosed boundaries. AIAA Paper 95-0857; 1995.
- [40] Jeng YN, Liou Y-C. Hyperbolic equation method of grid generation for enclosed regions. AIAA J 1996;34:1293-5.

- [41] Tai CH, Yin SL, Soong CY. A novel hyperbolic grid generation procedure with inherent adaptive dissipation. *J Comput Phys* 1995;116:173-9.
- [42] Brakhage K-H, Müller S. Algebraic-hyperbolic grid generation with precise control of intersection of angles. *Int J Numer Method Fluids* 2000;33:89-123.
- [43] Mavriplis DJ. An advancing front Delaunay triangulation algorithm designed for robustness. ICASE Report No. 92-49, 1992; also AIAA Paper 93-0671; 1993; also *J Comput Phys* 1995;117:90-101.
- [44] Frey PJ, Borouchaki H, George P-L. Delaunay tetrahedralization using an advancing-front approach. Proceedings of the 5th international meshing roundtable; October 1996. p. 31-46.
- [45] Fleischmann P, Selberherr S. Three-dimensional Delaunay mesh generation using a modified advancing front approach. Proceedings of the 6th international meshing roundtable; October 1997. p. 31-46.
- [46] Müller JD, Roe PL, Deconinck H. A frontal approach for internal node generation in Delaunay triangulations. *Int J Numer Method Fluids* 1993;17:241-56.
- [47] Müller JD. On triangles and flow [PhD Thesis]. The University of Michigan; 1996.
- [48] Marcum DL, Weatherill NP. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA J* 1995;33:1619-25.
- [49] Bern M, Eppstein D. Quadrilateral meshing by circle packing. Proceedings of the 6th international meshing roundtable; October 1997. p. 7-19
- [50] Liu J, Tang R. Ball-packing method: a new approach for quality automatic triangulation of arbitrary domains. Proceedings of the 6th international meshing roundtable; October 1997. p. 85-96
- [51] Shimada K, Yamada A, Itoh T. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. Proceedings of the 6th international meshing roundtable; October 1997. p. 375-90.
- [52] Li X-Y, Teng S-H, Ungor A. Biting spheres in 3D. Proceedings of the 8th international meshing roundtable; October 1999. p. 85-95.
- [53] Soji Y, Shimada K. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. Proceedings of the 9th international meshing roundtable; October 2000. p. 263-73.
- [54] Benabou A, Borouchaki H, Laug P, Lu J. Sphere packing and applications to granular structure modeling. Proceedings of the 17th international meshing roundtable; October 2008. p. 1-18.
- [55] Mavriplis DJ. Unstructured mesh generation and adaptivity. ICASE Report No. 95-26; 1995.
- [56] Weatherill NP, Hassan O, Morgan K, Peraire J, Peiro J, Marchant MJ. Unstructured grid generation. COSMASE: short course on grid generation, automation and parallel utilisation, Lausanne, Switzerland, September 23-27; 1996.
- [57] Dirichlet GL. Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Z Reine Angew Math* 1850;40:209-27.
- [58] Voronoi G. Nouvelles applications de paramètres continus à la théorie des formes quadratiques. *Z Reine Angew Math* 1908;133:97-178.
- [59] Delaunay B. Sur la sphère vide. *Bull Acad Sci USSR, Class Sci Mat Nat* 1934;7:793-800.
- [60] Holmes DG, Snyder DD. The generation of unstructured meshes using Delaunay triangulation. Proceedings of the 2nd international conference on numerical grid generation in CFD, Pineridge, Swansea, Wales, UK; 1988. p. 643-52.
- [61] George PL, Hecht F, Saltel E. Fully automatic mesh generator for 3-D domains of any shape. *Imp Comput Sci Eng* 1990;2:187-218.
- [62] Weatherill NP, Hassan O, Marcum DL. Compressible flowfield solutions with unstructured grids generated by Delaunay triangulation. *AIAA J* 1995;33:1196-204.
- [63] Rebay S. Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm. *J Comput Phys* 1993;106:125-38.
- [64] Baker TJ. Triangulations, mesh generation and point placement strategies. Proc. conf. frontiers of computational fluid dynamics 1994, Ithaca, New York, November 1994.
- [65] Green PJ, Sibson R. Computing the Dirichlet tessellation in the plane. *Comput J* 1977;21:168-73.
- [66] Bowyer A. Computing Dirichlet tessellations. *Comput J* 1981;24:162-6.
- [67] Watson DF. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *Comput J* 1981;24:167-72.

- [68] Ruppert J. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J Algorithms* 1995;18:548–85.
- [69] Shewchuk JR. Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. Proceedings of the 1st workshop application computational geometry, Philadelphia, USA; May 1996. p. 124–33.
- [70] Finkel RA, Bentley JL. Quad trees, a data structure for retrieval of composite keys. *Acta Inform* 1974;4:1–9.
- [71] Samet H. The quadtree and related hierarchical data structures. *Comput Surv* 1984;16:188–260.
- [72] Samet H. The design and analysis of spatial data structures. Boston: Addison-Wesley; 1990.
- [73] Yerry MA, Shephard MS. A modified quadtree approach to finite element mesh generation. *IEEE Comput Graph Appl* 1983;3:39–46.
- [74] Yerry MA, Shephard MS. Automatic three-dimensional mesh generation by the modified-octree technique. *Int J Numer Methods Eng* 1984;20:1965–90.
- [75] Miranda ACO, Martha LF. Mesh generation on high-curvature surfaces based on a background quadtree structure. Proceedings of the 11th international meshing roundtable; September 2002. p. 333–42.
- [76] Baker TJ. Three dimensional mesh generation by triangulation of arbitrary point sets. AIAA Paper 87-1124; 1987.
- [77] Chew LP. Constrained Delaunay triangulations. *Algorithmica* 1989;4:97–108.
- [78] George PL, Hecht F, Saltel E. Automatic mesh generator with specified boundary. *Comput Methods Appl Mech Eng* 1991;33:975–95.
- [79] Weatherill NP, Hassan O, Marcum DL. Calculation of steady compressible flowfields with the finite-element method. AIAA Paper 93-0341; 1993.
- [80] Sharov D, Nakahashi K. A boundary recovery algorithm for Delaunay tetrahedral meshing. Proceedings of the 5th international conference on numerical grid generation in computing field simulations, Mississippi State University; 1996. p. 229–38.
- [81] Joe B. Construction of three-dimensional constrained triangulations. Technical Report ZCS2008-06. Zhou Computing Services, Inc.; 2008.
- [82] Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. *J Comput Phys* 1987;72:449–66.
- [83] Peraire J, Peiro J, Formaggia L, Morgan K, Zienkiewicz OC. Finite element Euler computations in three dimensions. *Int J Numer Method Eng* 1988;26:2135–59.
- [84] Löhner R, Parikh P. Three-dimensional grid generation by the advancing front method. *Int J Numer Method Fluids* 1988;8:1135–49; also AIAA Paper 88-0515; 1988.
- [85] Löhner R. Progress in grid generation via the advancing front technique. *Eng Comput* 1996;12:186–210.
- [86] Pirzadeh S. Structured background grids for generation of unstructured grids by advancing-front method. *AIAA J* 1993;31:257–65.
- [87] Pirzadeh S. Unstructured viscous grid generation by advancing-front method. NASA CR-191449; 1993.
- [88] Barth TJ. On unstructured grids and solvers. VKI lecture series 1990-03; 1990. p. 1–65.
- [89] Marcum DL. Advancing-front/local-reconnection (AFLR) unstructured grid generation. In: Hafez M, Oshima K, editors. CFD review 1998, Part I. Singapore: World Scientific Publishing Co.; 1998.
- [90] Ito Y, Shih AM, Soni BK. Reliable isotropic tetrahedral mesh generation based on an advancing front method. Proceedings of the 13th international meshing roundtable; September 2004. p. 95–106.
- [91] Babuška I, Aziz AK. On the angle condition in the finite-element method. *SIAM J Numer Anal* 1976;13:214–26.
- [92] Baker TJ. Discretization of the Navier-Stokes equations and mesh induced errors. Proceedings of the 5th international conference on numerical grid generation in CFD. Mississippi: Mississippi State University; April 1996.
- [93] Baker TJ. Irregular meshes and the propagation of solution errors. Proceedings of the 15th international conference on numerical methods in fluid dynamics, Monterey, CA; June 1996.
- [94] Mavriplis DJ. Adaptive mesh generation for viscous flows using Delaunay triangulation. *J Comput Phys* 1990;90:271–91.

- [95] Vallet MG, Hecht F, Mantel B. Anisotropic control of mesh generation based upon a Voronoi type method. In: Arcilla AS, et al., editors. Proceedings of the 3rd international conference on numerical grid generation in computing fluid dynamics and related fields, Barcelona, Spain. North-Holland; 1991. p. 93-103.
- [96] Irmisch S, Schwolow R. Erzeugung unstrukturierter Dreiecknetze mittels der Delaunay-triangulierung. Zeitschrift Wundheilung 1994;18:361-8.
- [97] Pirzadeh S. Unstructured viscous grid generation by the advancing-layers method. AIAA J 1994;32:1735-7; also AIAA Paper 93-3453; 1993.
- [98] Pirzadeh S. Viscous unstructured three-dimensional grids by the advancing-layers method. AIAA Paper 94-0417; 1994.
- [99] Hassan O, Probert EJ, Weatherill NP, Marchant MJ, Morgan K, Marcum DL. The numerical simulation of viscous transonic flow using unstructured grids. AIAA Paper 94-2346; 1994.
- [100] Marchant MJ, Weatherill NP. Unstructured grid generation for viscous flow simulations. In: Weatherill NP, et al. editors. Proceedings of the 4th international conference on numerical grid generation in computing fluid dynamics and related fields, Swansea, UK. Pineridge Press; 1994. p. 151-62.
- [101] Piegl L, Tiller W. The NURBS book. 2nd ed. Berlin: Springer Verlag; 1997.
- [102] Whitmire JB, Dollar TW. Prismatic grid generation for NURBS bounded triangulations. AIAA Paper 98-0219; 1998.
- [103] Kallinderis Y, Ward S. Prismatic grid generation with an efficient algebraic method for aircraft configurations. AIAA Paper 92-2721; 1992.
- [104] Kallinderis Y, Ward S. Prismatic grid generation for 3-D complex geometries. AIAA J 1993;31:1850-6.
- [105] Kallinderis Y, Khawaja A, McMorris H. Hybrid prismatic/tetrahedral grid generation for viscous flows around complex geometries. AIAA J 1996;34:291-8.
- [106] Sharov D, Nakahashi K. Hybrid prismatic/tetrahedral grid generation for viscous flow applications. AIAA Paper 96-2000; 1996; also AIAA J 1998;36:157-62.
- [107] Connell SD, Braaten ME. Semistructured mesh generation for three-dimensional Navier-Stokes calculations. AIAA J 1995;33:1017-24.
- [108] Dompierre J, Labb   P, Vallet M-G, Camarero R. How to subdivide pyramids, prisms and hexahedra into tetrahedra. Report CERCA R99-78; 1999; also 8th international meshing roundtable, Lake Tahoe, CA; 1999.
- [109] M  ller JD. Quality estimates and stretched meshes based on Delaunay triangulations. AIAA J 1994;32:2372-9.
- [110] Marcum DL. Generation of unstructured grids for viscous flow applications. AIAA Paper 95-0212; 1995.
- [111] Marcum DL, Gaither JA. Mixed element type unstructured grid generation for viscous flow applications. AIAA Paper 99-3252; 1999.
- [112] McMorris H, Kallinderis Y. A combined octree-advancing front method for tetrahedra and anisotropic surface meshes. AIAA Paper 96-2442; 1996.
- [113] McMorris H, Kallinderis Y. Octree-advancing front method for generation of unstructured surface and volume meshes. AIAA J 1997;35:976-84.
- [114] Chen H, Bishop J. Delaunay triangulation for curved surfaces. Proceedings of the 6th international meshing roundtable, Park City, Utah, USA, October 13-15; 1997. p. 115-27.
- [115] Nakahashi K. FDM-FEM zonal approach for computations of compressible viscous flows. Lecture notes in physics, vol. 264. Heidelberg: Springer Verlag; 1986. p. 494-8.
- [116] Nakahashi K, Obayashi S. FDM-FEM zonal approach for viscous flow computations over multiple bodies. AIAA Paper 87-0604; 1987.
- [117] Nakahashi K, Obayashi S. Viscous flow computations using a composite grid. AIAA Paper 87-1128; 1987.
- [118] Weatherill NP. Mixed structured-unstructured meshes for aerodynamic flow simulations. Aeronaut J 1990;94:111-23.
- [119] Shaw JA, Georgala JM, Peace AJ, Childs PN. The construction, application and interpretation of three-dimensional hybrid meshes. In: Arcilla AS, et al., editors. Proceedings of the 3rd international

- conference on numeric grid generation in computing fluid dynamics an related fields, Barcelona, Spain. Amsterdam: North-Holland; 1991.
- [120] Michal T, Johnson J. A hybrid structured/unstructured grid multi-block flow solver for distributed parallel processing. AIAA Paper 97-1895; 1997.
 - [121] Shaw JA, Peace AJ. Simulating three-dimensional aeronautical flows on mixed block-structured/semi-unstructured/unstructured grids. Technical Memorandum 428. Bedford: Aircraft Research Association; 1998.
 - [122] Blazek J. Comparison of two conservative coupling algorithms for structured-unstructured grid interfaces. AIAA Paper 2003-3536; 2003.
 - [123] Pandya SA, Hafez MM. A semi-implicit finite-volume scheme for the solution of Euler equations on 3-D prismatic grids. AIAA Paper 93-3431; 1993.
 - [124] Parthasarathy V, Kallinderis Y, Nakajima K. Hybrid adaption method and directional viscous multigrid with prismatic tetrahedral meshes. AIAA Paper 95-0670; 1995.
 - [125] Parthasarathy V, Kallinderis Y. Directional viscous multigrid using adaptive prismatic meshes. AIAA J 1995;33:69-78.
 - [126] Kao KH, Liou MS. Direct replacement of arbitrary grid-overlapping by nonstructured grid. AIAA Paper 95-0346; 1995.
 - [127] Nakahashi K. Computations of three-dimensional Navier-Stokes equations by using a prismatic mesh. Proceedings of the 6th NAL symposium on aircraft computational aerodynamics, Tokyo, Japan; June 1988.
 - [128] Nakahashi K. A finite-element method on prismatic elements for the three-dimensional Navier-Stokes equations. Lecture notes in physics, vol. 323. Heidelberg: Springer Verlag; 1989. p. 434-8.
 - [129] Nakahashi K. Marching grid generation for external viscous flow problems. Trans Jpn Soc Aerospace Sci 1992;35:88-102.
 - [130] Kallinderis Y. A new finite-volume Navier-Stokes scheme on three-dimensional semi-structured prismatic elements. AIAA Paper 92-2697; 1992.
 - [131] Kallinderis Y. Adaptive hybrid prismatic/tetrahedral grids. Int J Numer Method Fluids 1995;20:1023-37.
 - [132] Melton JE, Pandya SA, Steger JL. 3-D Euler flow solutions using unstructured Cartesian and prismatic grids. AIAA Paper 93-0331; 1993.
 - [133] Karman SL. SPLITFLOW: a 3-D unstructured Cartesian/prismatic grid CFD code for complex geometries. AIAA Paper 95-0343, 1995.
 - [134] Smith RJ, Leschziner MA. Automatic grid generation for complex geometries. Aeronaut J 1996;100:7-14.
 - [135] Wang ZJ, Hufford GS, Przekwas AJ. Adaptive Cartesian/adaptive prism (ACAP) grid generation for complex geometries. AIAA Paper 97-0860; 1997.
 - [136] Delanaye M, Aftosmis M, Berger MJ, Liu Y, Pulliam TH. Automatic hybrid-Cartesian grid generation for high-Reynolds number flows around complex geometries. AIAA Paper 99-0777; 1999.
 - [137] Hitzel SM, Tremel U, Deister F, Rieger H. Complex configuration meshing—an industrial view and approach. AIAA Paper 2003-4130; 2003.
 - [138] Rai MM. A conservative treatment of zonal boundaries for Euler equation calculations. J Comput Phys 1986;62:472-503.
 - [139] Shaw JA, Stokes S, Lucking MA. The rapid and robust generation of efficient hybrid grids for RANS simulations over complete aircraft. Int J Numer Method Fluids 2003;43:785-821.
 - [140] Karman SL. Hierarchical unstructured mesh generation. AIAA Paper 2004-0613; 2004.
 - [141] Parthasarathy VN, Graichen CM, Hathaway AF. A comparison of tetrahedron quality measures. Finite Elements Anal Des 1993;15:255-61.
 - [142] Lawson CL. Software for C^1 surface interpolation. In: Rice JR, editor. Mathematical software III. New York: Academic Press; 1977. p. 161-94.
 - [143] Lawson CL. Properties of n-dimensional triangulations. Comput Aided Geom Des 1986;3:231-46.

CHAPTER 12

Software Applications

Contents

12.1	Programs for Stability Analysis	397
12.2	Structured 1-D Grid Generator	397
12.3	Structured 2-D Grid Generators	398
12.4	Structured to Unstructured Grid Converter	399
12.5	Quasi 1-D Euler Solver	399
12.5.1	Example run	400
12.6	Structured 2-D Euler/Navier-Stokes Solver	400
12.6.1	Example run	403
12.7	Unstructured 2-D Euler/Navier-Stokes Solver	405
12.7.1	Example run	405
12.8	Parallelization	406
	References	407

The software package contains the source codes and executables of several 1-D and 2-D flow solvers and grid generators. Provided are also input datasets and grids for various 1-D and 2-D examples of flow cases. Furthermore, there are two programs for the von Neumann stability analysis of explicit and implicit time-stepping schemes. Finally, you will find examples for code parallelization using different approaches.

The aim of the software is to demonstrate how to translate the theoretical principles of the computational fluid dynamics, which were presented in the previous chapters, into a computer code. The programs should be conceived as a basis for further experimentation and enhancements. The source codes are provided under the terms of the GNU General Public License, which means that they may be freely copied and redistributed free of charge. See the file `LICENSE.txt` in one of the directories for more details.

Source codes of the flow solvers and grid generators are written in Fortran-77 and in Fortran-90. The source of the 2-D unstructured flow solver is also provided in modern C++. The programs do not contain any references to external libraries. The source codes are kept as simple as possible, but still flexible enough. No extensive attempts were made to optimize the software for speed or for memory consumption. With a few exceptions, the source codes are organized in such a way that there is just one subroutine or function per file. Sources of the 2-D flow solvers are fully documented using the **Doxygen**¹ tool.

All grid, solution, and convergence files are stored in a plain ASCII format. The convergence and the solution files are written out in a form suitable for visualization

¹ <http://www.doxygen.org>.

with the program **Vis2D**.² The format of the convergence and solution files is always the same—there is one column for each variable. The names of the variables are given in the header. In the case of the solution files generated by the unstructured flow solver, the indexes of the nodes of each element are stored after the coordinates and the flow quantities. The first column represents node 1, the second one node 2, etc. The file format is described in detail in the documentation of **Vis2D**. Because of its simple form, the format can be adapted easily for other visualization tools if necessary.

The software package is organized into three highest-level directories: C++, Fortran, and Parallelization. These contain the following subdirectories:

- **C++**
 - unstructured_2d—solution of 2-D Euler/Navier-Stokes equations on unstructured triangular grids.
- **Fortran**
 - analysis—von Neumann stability analysis of 1-D model equations
 - grid_1d—1-D grid generation (Laval nozzle)
 - grid_struct_2d—2-D structured grid generation for external and internal flows
 - grid_unstr_2d—conversion of 2-D structured into unstructured triangular grids
 - structured_1d—solution of quasi 1-D Euler equations (flow through Laval nozzle); demonstration of the multigrid method
 - structured_2d—solution of 2-D Euler/Navier-Stokes equations on structured grids
 - unstructured_2d—solution of 2-D Euler/Navier-Stokes equations on unstructured triangular grids.
- **Parallelization**
 - agents—Asynchronous Agents Library
 - cuda—CUDA
 - mpi—MPI
 - openmp—OpenMP
 - serial—serial version.

The contents of the above subdirectories are further explained in the sections below. In general, the subdirectory of each simulation code contains a README.txt file. It describes the particular files, how to compile and to run the application, and how the results are stored. Additionally, the README.txt file explains the meaning of the main variables and of the input parameters. In the case of the 2-D flow solvers, the subdirectory doc contains the documentation in HTML format (index.html). The subdirectory of each flow solver also has the folder run, where you can find input files, grids, solutions, and graphics for the various demonstration cases.

² Available from CFD Consulting & Analysis (<http://www.cfd-ca.de>).

The last remark concerns the convergence history. The convergence of all flow solvers provided here is measured as the 2-norm of the difference of the density variable from two consecutive time steps, i.e.,

$$\|\Delta\rho\|_2 = \sqrt{\sum_{I=1}^N (\rho_I^{n+1} - \rho_I^n)^2}. \quad (12.1)$$

The summation is carried out over all N control volumes. It is convenient to normalize the convergence measure in Eq. (12.1) with its value from the first iteration. The programs store the normalized convergence history directly in logarithmic scale.

12.1 PROGRAMS FOR STABILITY ANALYSIS

The directory `analysis` contains two programs for the von Neumann stability analysis (Section 10.3) of linear 1-D model equations. The first program computes the Fourier symbol and the magnitude of the amplification factor for the explicit multistage time-stepping scheme (Section 6.1.1) and the hybrid scheme (Section 6.1.2). The source code is provided in the subdirectory `mstage`. The second program analyses the damping properties of a general implicit scheme (Section 6.2). It is contained in the subdirectory `implicit`. Both programs can deal with either the 1-D convection or the 1-D convection-diffusion model equation (Sections 10.3.2 and 10.3.3). The spatial discretization can be conducted either by the central scheme with artificial dissipation, by the 1st-order upwind, or by the 2nd-order upwind scheme, respectively. The same schemes can be also applied to the implicit operator. In the case of the explicit scheme, the effect of the (central or upwind) implicit residual smoothing (Section 9.3) can be investigated as well.

12.2 STRUCTURED 1-D GRID GENERATOR

The directory `grid_1d` contains a program for the generation of 1-D structured grids. Each grid node is associated with a certain area. The area distribution along the x -axis corresponds to the Laval nozzle. The area of the nozzle is evaluated using the relation

$$A(x) = \begin{cases} 1 + \frac{1}{2}(A_1 - 1) \left\{ 1 + \cos \left(\frac{\pi x}{x_{\text{thr}}} \right) \right\} & \text{for } 0 \leq x \leq x_{\text{thr}} \\ 1 + \frac{1}{2}(A_2 - 1) \left\{ 1 - \cos \left[\frac{\pi(x - x_{\text{thr}})}{1 - x_{\text{thr}}} \right] \right\} & \text{for } x_{\text{thr}} < x \leq 1, \end{cases} \quad (12.2)$$

where x_{thr} denotes the location of the throat. The length of the nozzle is supposed to be one. Furthermore, in Eq. (12.2) A_1 represents the inlet area and A_2 the outlet area

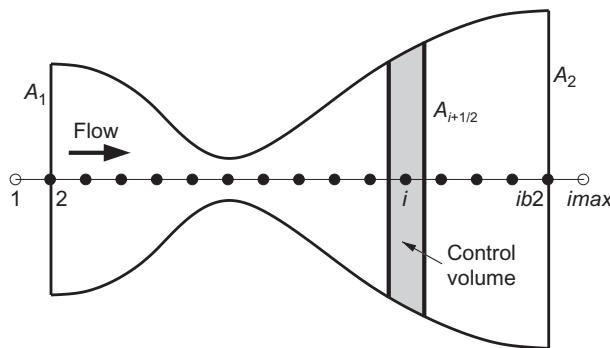


Figure 12.1 Grid and control volume for the 1-D Euler solver; points $i = 1$ and $i = imax$ are dummy points.

(see the sketch in Fig. 12.1). The area of the throat results from Eq. (12.2) to $A^*(x=x_{thr}) = 1$. The grid generated by this program serves as an input to the quasi 1-D Euler solver from Section 12.5.

12.3 STRUCTURED 2-D GRID GENERATORS

The directory `grid_struct_2d` contains three different programs for the generation of 2-D structured grids for external and internal flows. The first program, which source code is provided in the subdirectory `cgrid`, generates a C-type grid (see Section 11.1.1) around an airfoil. An example can be seen in Fig. 11.6. The initial grid is generated algebraically by using the linear TFI method Eq. (11.5). Afterward, elliptic PDEs (Section 11.1.3) are employed to produce boundary-orthogonal grid with specified wall spacing. The airfoil contour is approximated by a Bézier spline (see Refs. [1, 2] for an introduction to Bézier splines).

The second program is intended for the generation of an algebraic grid inside a 2-D channel with circular bump. The grid points are clustered around the front and the rear point of the bump. The source code can be found in the subdirectory `channel`.

The next program, which is located in the subdirectory `hgrid`, generates an H-type grid (Section 11.1.1) for a cascade. An example of grid created by the program is presented in Fig. 11.8. The linear TFI technique (Eq. (11.5)) for algebraic grid generation is employed in this case. The contour of the blade is again described by a Bézier spline, which interpolates the given points.

The programs `cgrid` and `hgrid` rely on the library provided in the subdirectory `sccom`. This library contains routines for spline interpolation, linear TFI, elliptic grid generation, and for grid stretching.

12.4 STRUCTURED TO UNSTRUCTURED GRID CONVERTER

The directory `grid_unstr_2d` contains a program for the conversion of 2-D structured grids into unstructured triangular grids. The program divides each quadrilateral cell of the structured grid into two triangles by connecting diagonal nodes with the shortest distance. The triangulated grids can be used as an input for the unstructured flow solver in [Section 12.7](#).

12.5 QUASI 1-D EULER SOLVER

The directory `structured_1d` contains a program for the solution of quasi 1-D Euler equations. The equations govern the inviscid flow in a nozzle or in a tube. They can be written in conservative, differential form as [3]

$$\frac{\partial \vec{W}}{\partial t} + \frac{\partial \vec{F}_c}{\partial x} = \vec{Q}. \quad (12.3)$$

The vectors of conservative variables, convective fluxes, and the source term read

$$\vec{W} = \begin{bmatrix} \rho A \\ \rho u A \\ \rho EA \end{bmatrix}, \quad \vec{F}_c = \begin{bmatrix} \rho u A \\ (\rho u^2 + p)A \\ \rho Hu A \end{bmatrix}, \quad \vec{Q} = \begin{bmatrix} 0 \\ p dA/dx \\ 0 \end{bmatrix}, \quad (12.4)$$

where A denotes the nozzle area. The total enthalpy H is given by the formula (2.12), and the pressure results according to Eq. (2.29) from

$$p = (\gamma - 1) \rho \left(E - \frac{u^2}{2} \right). \quad (12.5)$$

The governing equations (12.3) are discretized on a structured grid using the dual control-volume methodology (see [Section 4.2.3](#)). A sketch of the grid and of the control volume is displayed in [Fig. 12.1](#). Three different schemes are employed for the spatial discretization:

- Central scheme with scalar artificial dissipation ([Section 4.3.1](#))
- CUSP flux-vector splitting scheme ([Section 4.3.2](#))
- Roe's flux-difference splitting scheme ([Section 4.3.3](#)).

The discretized equations (12.3) are advanced in time using the explicit multistage scheme ([Section 6.1.1](#) or [6.1.2](#)). The program uses local time-stepping ([Section 9.1](#)) and the central implicit residual smoothing technique ([Section 9.3.1](#)) for convergence acceleration. The source code can be found in the subdirectory `src`. The input files and the results are provided in the subdirectory `run`.

A second version of the flow solver employs the multigrid method from [Section 9.4](#) for convergence acceleration. The spatial and temporal discretization schemes are otherwise the same as above. The multigrid scheme uses the FAS formulation for the

solution on coarse grids and FMG to initialize the flow on the finest grid. The source code is contained in the subdirectory `srcmg`, exemplary input files can be found in the subdirectory `runmg`.

The boundary conditions at the inlet and the outlet plane are implemented in characteristic variables as presented in Section 8.4. The concept of dummy points, described in Section 8.1, is utilized for this purpose (see Fig. 12.1).

12.5.1 Example run

As a test example, let us consider flow through a Laval nozzle with the inlet area $A_1 = 1.5$, outlet area $A_2 = 2.5$, and with the following boundary conditions:

- Inlet total pressure $p_{t,in} = 1.0 \times 10^5 \text{ Pa}$
- Inlet total temperature $T_{t,in} = 288.0 \text{ K}$
- Outlet static pressure $p_{out} = 7.0 \times 10^4 \text{ Pa}$

The simulation shall be carried out using Roe's upwind scheme for spatial discretization and multigrid with five grid levels: $\sigma = 4.5$, $\epsilon = 0.8$, limiter coefficient $K = 1.5$, entropy correction coefficient $\delta = 0.05 \cdot c$.

The grid for the example can be generated using the program described in Section 12.2. It is also provided as `laval.grd` in the subdirectory `runmg`. The flow solver further requires the input of various parameters in order to set up the boundary conditions, the numerical schemes, as well as the output of convergence history and of plot files. These data are provided in the user input file named `input_r`. Assuming the source code of the solver is already compiled (see `README.txt` for details), the simulation can be started by opening a terminal window (Command Prompt on Windows), changing to `runmg` and typing:

```
% eu1ldmg < input_r
```

This will produce the convergence history displayed in Fig. 12.2. Figure 12.3 shows the resulting Mach-number distribution along the nozzle length.

12.6 STRUCTURED 2-D EULER/NAVIER-STOKES SOLVER

The directory `structured_2d` contains a program for the solution of 2-D Euler and Navier-Stokes equations on structured body-fitted grids. The spatial discretization is based on the cell-centered finite-volume approach described in Section 4.2.1. It employs the central discretization scheme with scalar artificial dissipation (Section 4.3.1), as well as Roe's upwind scheme presented in Section 4.3.3. The governing equations are integrated in time using an explicit multistage scheme (Section 6.1.1 or 6.1.2), accelerated by local time-stepping (Section 9.1) and the central implicit residual smoothing (Section 9.3.1). Gradients of the velocity components and of the temperature are computed using the dual control-volume approach described in Section 4.4.1.

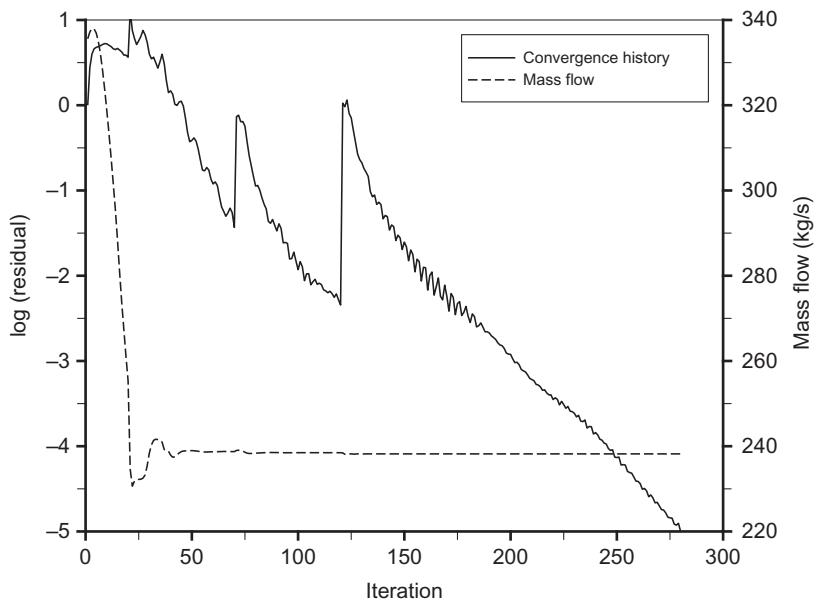


Figure 12.2 Convergence history of the density residual and of the mass flow.

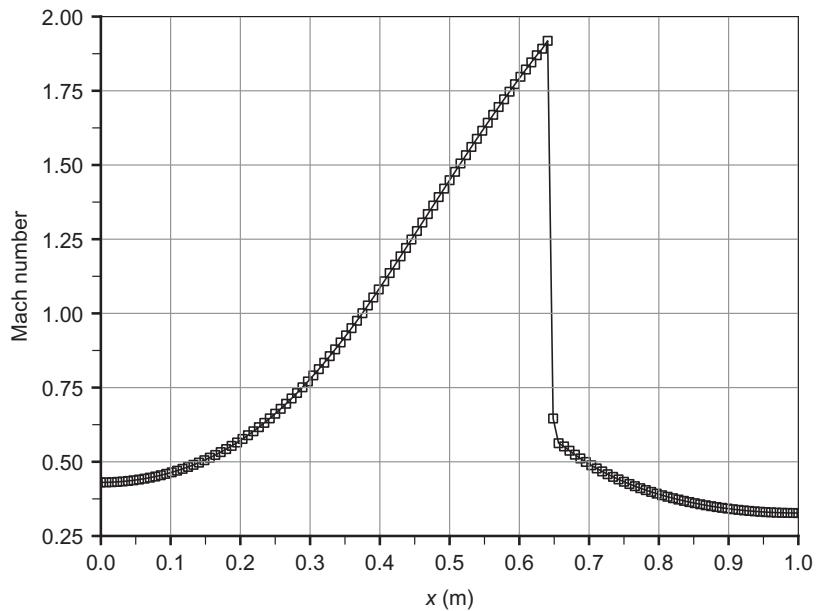


Figure 12.3 Mach-number distribution over nozzle length.

The program also offers the possibility to simulate incompressible flows at very low Mach numbers (see Section 9.5). In order to keep the code easy to understand, no turbulence model is implemented.

Source code of the flow solver is provided in the subdirectory `src`. The input files, grids, and the results can be found in the subdirectory `run`. The subdirectory `doc` contains the documentation (files, modules, functions, variables) in HTML format.

The program utilizes the concept of dummy cells (Section 8.1) for the treatment of the boundary conditions. Two layers of dummy cells are employed. A sketch of the grid in the computational domain is displayed in Fig. 12.4 (cf. Fig. 8.1). The program can deal only with single-block grids. However, it is very flexible with respect to the specification and the type of the boundary conditions. The following eight boundary types are provided:

- Coordinate cut (for C- or O-type grids)
- Far-field (optionally with vortex correction; sub- and supersonic)
- Inflow (subsonic only)
- Outflow (sub- and supersonic)
- Fluid injection (or extraction)
- Line periodic
- Solid wall (viscous or inviscid)
- Symmetry (along $x = \text{const.}$ or $y = \text{const.}$ line).

The implementation of the above boundary conditions closely follows the discussion in Chapter 8.

The four boundaries in the computational space can be divided into an arbitrary number of segments. Each of the segments can be associated with a different boundary

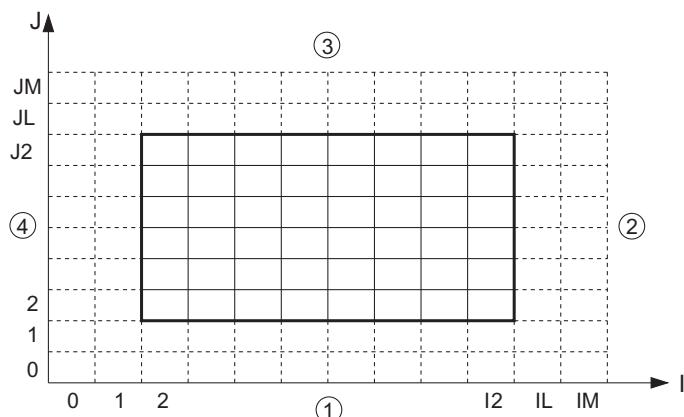


Figure 12.4 Grid in the computational space for the structured 2-D Euler/Navier-Stokes solver. There are two layers of dummy cells at each boundary (dashed line). Numbers in circles denote the sides of the computational domain.

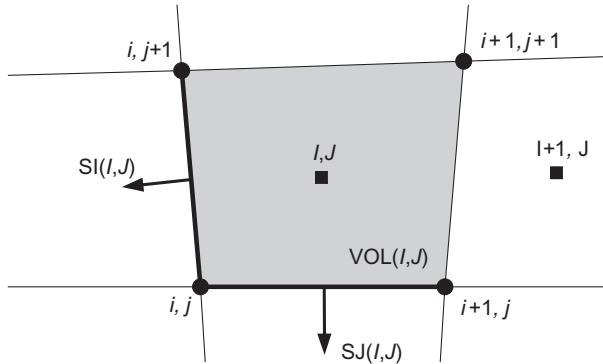


Figure 12.5 Control volume and face vectors for the structured 2-D Euler/ Navier-Stokes solver. Indexes (I, J) denote the cell, indexes (i, j) represent the grid node.

condition. This approach is quite similar to the description of block interfaces presented in Section 8.9. In fact, the program could be relatively easily extended to multiblock grids. Definitions of the segments are stored separately from the grid in the topology file (with the file extension `top`).

The definition of the face vectors SI and SJ (i.e., $\vec{n} \cdot \Delta S$) employed in the solver can be seen in Fig. 12.5. The face vectors are associated with the left and the bottom face of the control volume $VOL(I, J)$ (corresponds to $\Omega_{I,J}$). They point outwards of the control volume. The face vectors of the remaining two sides of the control volume are obtained as $-SI(I+1, J)$ and $-SJ(I, J+1)$. In this way, we have to store only two face vectors for each control volume.

12.6.1 Example run

As a test example, let us compute transonic flow past the symmetric NACA 0012 airfoil with the following boundary conditions along the far-field:

- Mach number $M_\infty = 0.8$
- Angle of attack $\alpha = 1.25^\circ$
- Static pressure $p_\infty = 1.0 \times 10^5 \text{ Pa}$
- Static temperature $T_\infty = 288.0 \text{ K}$

The simulation shall be carried out using Roe's upwind scheme for spatial discretization and explicit time-stepping: $\sigma = 5.0$, $\epsilon = 1.2$, limiter coefficient $K = 30.0$, entropy correction coefficient $\delta = 0.05 \cdot c$.

The grid for the example can be generated using the program described in Section 12.3. It is also provided as `n0012.grd` in the subdirectory `run`. The flow solver further requires the input of various parameters in order to set up the boundary conditions, the numerical schemes, as well as the output of convergence history and of plot files. These data are provided in the user input file named `n0012_input_r`.

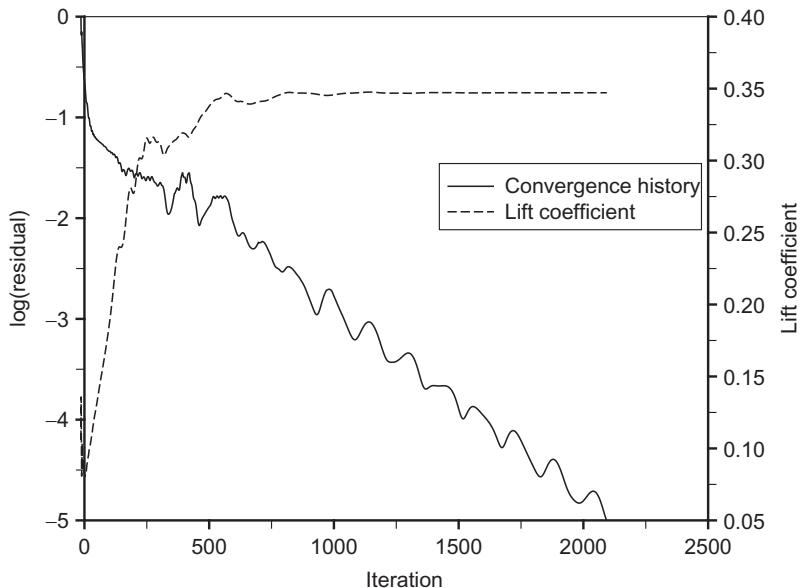


Figure 12.6 Convergence history of the density residual and of the lift coefficient.

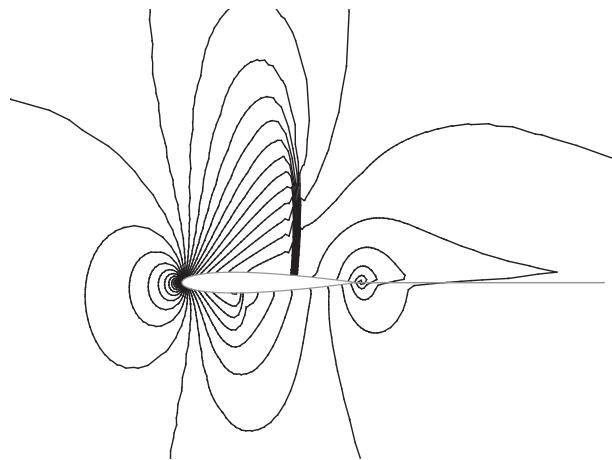


Figure 12.7 Mach-number distribution around the airfoil.

Assuming the source code of the solver is already compiled (see `README.txt` for details), the simulation can be started by opening a terminal window (Command Prompt on Windows), changing to `run` and typing:

```
% Struct2D n0012_input_r
```

This will produce the convergence history displayed in Fig. 12.6. Figure 12.7 shows the resulting Mach-number contours around the airfoil.

12.7 UNSTRUCTURED 2-D EULER/NAVIER-STOKES SOLVER

The directories `unstructured_2d` and `C++/unstructured_2d` contain a program for the solution of 2-D Euler and Navier-Stokes equations on unstructured triangular grids. The median-dual cell-vertex scheme described in Section 5.2.2 is utilized for the spatial discretization. The computation of the convective and viscous fluxes, as well as of the gradients employs an edge-based data structure. The program also offers the possibility to simulate incompressible flows at very low Mach numbers (see Section 9.5).

Source code of the flow solver is provided in the subdirectory `src`. The input files, grids, and the results can be found in the subdirectory `run`. The subdirectory `doc` contains the documentation (files, modules, functions, variables) in HTML format.

Convective fluxes are evaluated according to Roe's flux-difference splitting scheme (Sections 4.3.3 and 5.3.2). The solution at the faces of the control volume is obtained by piecewise linear reconstruction of the left and right states as given by Eq. (5.42). The gradients of the flow variables are computed with the Green-Gauss approach (Eqs. (5.49) and (5.50)). The reconstructed solution is limited using Venkatakrishnan's limiter described in Section 5.3.5 (Eq. (5.67)). The gradients at the faces of the control volume, which are required for the evaluation of the viscous fluxes, are obtained by the methodology presented in Section 5.4.2. The discretized governing equations are integrated in time using the explicit multistage scheme (Section 6.1.1 or 6.1.2), enhanced by local time-stepping (Section 9.1) and the central implicit residual smoothing (Section 9.3.2).

The same boundary conditions as described previously for the structured flow solver (except the injection and the coordinate cut) are included. The data structure is such that each boundary face could have a different boundary condition. The implementation of the far-field, inlet and outlet boundary conditions employs the concept of dummy nodes (with one layer). All topological data related to the boundaries are stored together with the grid coordinates and the node indexes in one common file (with the extension `ugr`).

12.7.1 Example run

As a test example, let us compute transonic flow through the VKI-1 turbine cascade with the following boundary conditions [4]:

- Inlet total pressure $p_{t,in} = 1.0 \times 10^5$ Pa
- Inlet total temperature $T_{t,in} = 300.0$ K
- Inlet flow angle $\alpha_{inl} = 30^\circ$
- Outlet static pressure $p_{out} = 5.283 \times 10^4$ Pa

The simulation shall be carried out using Roe's upwind scheme for spatial discretization and explicit time-stepping: $\sigma = 5.5$, $\epsilon = 0.4$, limiter coefficient $K = 1.0$, entropy correction coefficient $\delta = 0.05 \cdot c$.

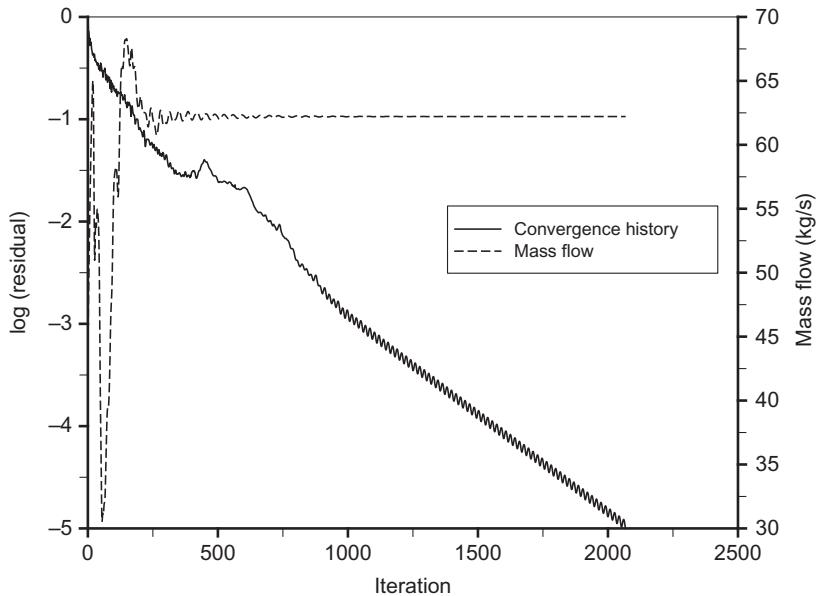


Figure 12.8 Convergence history of the density residual and of the mass flow.

The grid for the example can be generated using the programs described in Sections 12.3 and 12.4. An unstructured grid for the case generated by the advancing-front method (see Section 11.2.2) is provided as `vk1.ugr` in the subdirectory `run`. The flow solver further requires the input of various parameters in order to set up the boundary conditions, the numerical schemes, as well as the output of convergence history and of plot files. These data are provided in the user input file named `vk1_input`. Assuming the source code of the solver is already compiled (see `README.txt` for details), the simulation can be started by opening a terminal window (Command Prompt on Windows), changing to `run` and typing:

```
% Unstruct2D vk1_input
```

This will produce the convergence history displayed in Fig. 12.8. Figure 12.9 shows the resulting pressure contours inside the cascade.

12.8 PARALLELIZATION

The directory `Parallelization` contains examples of code parallelized using Asynchronous Agents Library, CUDA, MPI, and OpenMP (Section 9.6). The program to be parallelized solves the 2-D Laplace equation

$$\phi_{xx} + \phi_{yy} = 0 \quad (12.6)$$

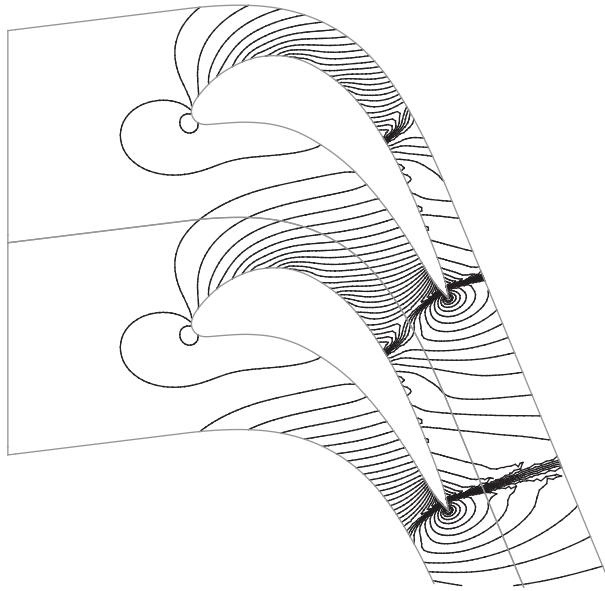


Figure 12.9 Pressure contours inside the cascade.

on a structured, rectangular grid using the Jacobi iteration. The boundary values are kept fixed during the iterations.

The examples are written in C (CUDA and MPI) and in C++. In the case of MPI, compilation is controlled by a Makefile. In the other cases, project files for Visual Studio (VS 2012) are provided. Note that all examples, except for OpenMP, require the presence of corresponding libraries and compilation tools (see references in Section 9.6). The OpenMP program does not need any special provisions, any recent C++ compiler is sufficient.

REFERENCES

- [1] Farin GE. Curves and surfaces for computer aided geometric design—a practical guide. 3rd ed. Boston: Academic Press; 1993.
- [2] Hoschek J, Lasser D. Fundamentals of computer aided geometric design. Wellesley, MA: A.K. Peters Ltd; 1993.
- [3] Shapiro AH. The dynamics and thermodynamics of compressible fluid flow. New York: Ronald Press; 1953.
- [4] Klock R, Lehthaus F, Baines NC, Sieverding CH. The transonic flow through a plane turbine cascade as measured in four European wind tunnels. *ASME J Eng Gas Turbines Power* 1986;108:277-84.

Appendix

Contents

A.1	Governing Equations in Differential Form	409
A.2	Quasilinear Form of the Euler Equations	415
A.3	Mathematical Character of the Governing Equations	416
A.3.1	Hyperbolic equations	416
A.3.2	Parabolic equations	417
A.3.3	Elliptic equations	418
A.4	Navier-Stokes Equations in Rotating Frame of Reference	419
A.5	Navier-Stokes Equations Formulated for Moving Grids	421
A.6	Thin Shear Layer Approximation	425
A.7	PNS Equations	427
A.8	Axisymmetric Form of the Navier-Stokes Equations	427
A.9	Convective Flux Jacobian	429
A.10	Viscous Flux Jacobian	431
A.11	Transformation from Conservative to Characteristic Variables	433
A.12	GMRES Algorithm	437
A.12.1	Computation of the orthonormal basis of \mathcal{K}_m	437
A.12.2	Generation of the upper Hessenberg matrix	438
A.12.3	Minimization of the residual	438
A.12.4	Q-R algorithm	439
A.13	Tensor Notation	440
	References	441

A.1 GOVERNING EQUATIONS IN DIFFERENTIAL FORM

Supposed the convective and viscous fluxes are continuous, the governing equations (2.19) can be transformed from integral to differential form by first applying Gauss's theorem. This leads to

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} \, d\Omega + \int_{\Omega} \vec{\nabla} \cdot (\bar{\bar{F}}_c - \bar{\bar{F}}_v) \, d\Omega = \int_{\Omega} \vec{Q} \, d\Omega, \quad (\text{A.1})$$

where $\bar{\bar{F}}_c$ and $\bar{\bar{F}}_v$ denote the tensors of the convective and viscous fluxes, respectively. Equation (A.1) can then be written for an arbitrary control volume Ω in the differential form as

$$\frac{\partial \vec{W}}{\partial t} + \vec{\nabla} \cdot (\bar{\bar{F}}_c - \bar{\bar{F}}_v) = \vec{Q}. \quad (\text{A.2})$$

In order to account for arbitrary body-fitted grids, we introduce a mapping function between the Cartesian (x, y, z) and a curvilinear (ξ, η, ζ) coordinate system (cf. Fig. 3.2)

$$\begin{aligned}\xi &= \xi(x, y, z, t), \\ \eta &= \eta(x, y, z, t), \\ \zeta &= \zeta(x, y, z, t).\end{aligned}\quad (\text{A.3})$$

With this, the differential form of the 3-D Navier-Stokes equations (A.2) transforms into

$$\frac{\partial \vec{W}^*}{\partial t} + \frac{\partial \vec{F}_{c,1}}{\partial \xi} + \frac{\partial \vec{F}_{c,2}}{\partial \eta} + \frac{\partial \vec{F}_{c,3}}{\partial \zeta} = \frac{\partial \vec{F}_{v,1}}{\partial \xi} + \frac{\partial \vec{F}_{v,2}}{\partial \eta} + \frac{\partial \vec{F}_{v,3}}{\partial \zeta} + \vec{Q}^*. \quad (\text{A.4})$$

The vector of the conservative variables is now given by

$$\vec{W}^* = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}. \quad (\text{A.5})$$

The vectors of the convective fluxes follow from the relationships

$$\begin{aligned}\vec{F}_{c,1} &= J^{-1} \begin{bmatrix} \rho V_1 \\ \rho V_1 u + \xi_x p \\ \rho V_1 v + \xi_y p \\ \rho V_1 w + \xi_z p \\ \rho V_1 H \end{bmatrix}, & \vec{F}_{c,2} &= J^{-1} \begin{bmatrix} \rho V_2 \\ \rho V_2 u + \eta_x p \\ \rho V_2 v + \eta_y p \\ \rho V_2 w + \eta_z p \\ \rho V_2 H \end{bmatrix}, \\ \vec{F}_{c,3} &= J^{-1} \begin{bmatrix} \rho V_3 \\ \rho V_3 u + \zeta_x p \\ \rho V_3 v + \zeta_y p \\ \rho V_3 w + \zeta_z p \\ \rho V_3 H \end{bmatrix},\end{aligned}\quad (\text{A.6})$$

and the vectors of the viscous fluxes are defined as

$$\begin{aligned}\vec{F}_{v,1} &= J^{-1} \begin{bmatrix} 0 \\ \xi_x \tau_{xx} + \xi_y \tau_{xy} + \xi_z \tau_{xz} \\ \xi_x \tau_{yx} + \xi_y \tau_{yy} + \xi_z \tau_{yz} \\ \xi_x \tau_{zx} + \xi_y \tau_{zy} + \xi_z \tau_{zz} \\ \xi_x \Theta_x + \xi_y \Theta_y + \xi_z \Theta_z \end{bmatrix}, \\ \vec{F}_{v,2} &= J^{-1} \begin{bmatrix} 0 \\ \eta_x \tau_{xx} + \eta_y \tau_{xy} + \eta_z \tau_{xz} \\ \eta_x \tau_{yx} + \eta_y \tau_{yy} + \eta_z \tau_{yz} \\ \eta_x \tau_{zx} + \eta_y \tau_{zy} + \eta_z \tau_{zz} \\ \eta_x \Theta_x + \eta_y \Theta_y + \eta_z \Theta_z \end{bmatrix},\end{aligned}\quad (\text{A.7})$$

$$\vec{F}_{v,3} = J^{-1} \begin{bmatrix} 0 \\ \zeta_x \tau_{xx} + \zeta_y \tau_{xy} + \zeta_z \tau_{xz} \\ \zeta_x \tau_{yx} + \zeta_y \tau_{yy} + \zeta_z \tau_{yz} \\ \zeta_x \tau_{zx} + \zeta_y \tau_{zy} + \zeta_z \tau_{zz} \\ \zeta_x \Theta_x + \zeta_y \Theta_y + \zeta_z \Theta_z \end{bmatrix}.$$

Finally, the source term transforms to

$$\vec{Q}^* = J^{-1} \begin{bmatrix} 0 \\ \rho f_{e,x} \\ \rho f_{e,y} \\ \rho f_{e,z} \\ \rho \vec{f}_e \cdot \vec{v} + \dot{q}_h \end{bmatrix}, \quad (\text{A.8})$$

where \vec{f}_e represents the vector of the external body forces.

The contravariant velocities in the ξ , η , ζ coordinate directions for a stationary grid are given by the formulas

$$\begin{aligned} V_1 &= \xi_x u + \xi_y v + \xi_z w, \\ V_2 &= \eta_x u + \eta_y v + \eta_z w, \\ V_3 &= \zeta_x u + \zeta_y v + \zeta_z w. \end{aligned} \quad (\text{A.9})$$

The components of the viscous stress tensor and of the thermal fluxes read

$$\begin{aligned} \tau_{xx} &= 2\mu \frac{\partial u}{\partial x} + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right), \\ \tau_{yy} &= 2\mu \frac{\partial v}{\partial y} + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right), \\ \tau_{zz} &= 2\mu \frac{\partial w}{\partial z} + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right), \\ \tau_{xy} &= \tau_{yx} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right), \\ \tau_{xz} &= \tau_{zx} = \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right), \\ \tau_{yz} &= \tau_{zy} = \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right), \\ \Theta_x &= u \tau_{xx} + v \tau_{xy} + w \tau_{xz} + k \frac{\partial T}{\partial x}, \\ \Theta_y &= u \tau_{yx} + v \tau_{yy} + w \tau_{yz} + k \frac{\partial T}{\partial y}, \\ \Theta_z &= u \tau_{zx} + v \tau_{zy} + w \tau_{zz} + k \frac{\partial T}{\partial z}. \end{aligned} \quad (\text{A.10})$$

In Eq. (A.10), k stands for the thermal conductivity coefficient, μ for the coefficient of the dynamic viscosity, and λ denotes the second viscosity coefficient ($\lambda = -(2/3)\mu$ according to Stokes's hypothesis). The diffusive thermal fluxes can also be written in the following modified form:

$$\begin{aligned} k \frac{\partial T}{\partial x} &= \frac{\mu}{(\gamma - 1)Pr} \frac{\partial c^2}{\partial x}, \\ k \frac{\partial T}{\partial y} &= \frac{\mu}{(\gamma - 1)Pr} \frac{\partial c^2}{\partial y}, \\ k \frac{\partial T}{\partial z} &= \frac{\mu}{(\gamma - 1)Pr} \frac{\partial c^2}{\partial z}, \end{aligned} \quad (\text{A.11})$$

with c representing the speed of sound and Pr the Prandtl number. The derivatives of the velocity components u , v , w , and the temperature T in Cartesian coordinates x , y , z can be expressed with the aid of the chain rule as derivatives of ξ , η , and ζ . For example,

$$\begin{aligned} \frac{\partial u}{\partial x} &= \xi_x \frac{\partial u}{\partial \xi} + \eta_x \frac{\partial u}{\partial \eta} + \zeta_x \frac{\partial u}{\partial \zeta}, \\ \frac{\partial u}{\partial y} &= \xi_y \frac{\partial u}{\partial \xi} + \eta_y \frac{\partial u}{\partial \eta} + \zeta_y \frac{\partial u}{\partial \zeta}, \quad \text{etc.} \end{aligned} \quad (\text{A.12})$$

The inverse of the determinant of the coordinate transformation Jacobian $\partial(\xi, \eta, \zeta)/\partial(x, y, z)$ is defined as

$$\begin{aligned} J^{-1} &= \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \zeta} + \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \zeta} \frac{\partial z}{\partial \xi} + \frac{\partial x}{\partial \zeta} \frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \eta} \\ &\quad - \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \zeta} \frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta} \frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \xi}. \end{aligned} \quad (\text{A.13})$$

Finally, the metric terms are specified through the relations

$$\xi_x = J \left(\frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta} \frac{\partial z}{\partial \eta} \right),$$

$$\xi_y = J \left(\frac{\partial z}{\partial \eta} \frac{\partial x}{\partial \zeta} - \frac{\partial z}{\partial \zeta} \frac{\partial x}{\partial \eta} \right),$$

$$\xi_z = J \left(\frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta} \frac{\partial y}{\partial \eta} \right),$$

$$\eta_x = J \left(\frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \eta} - \frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \xi} \right),$$

$$\begin{aligned}
\eta_y &= J \left(\frac{\partial z}{\partial \xi} \frac{\partial x}{\partial \xi} - \frac{\partial z}{\partial \xi} \frac{\partial x}{\partial \zeta} \right), \\
\eta_z &= J \left(\frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \zeta} \right), \\
\xi_x &= J \left(\frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \eta} - \frac{\partial z}{\partial \xi} \frac{\partial y}{\partial \eta} \right), \\
\xi_y &= J \left(\frac{\partial z}{\partial \xi} \frac{\partial x}{\partial \eta} - \frac{\partial x}{\partial \xi} \frac{\partial z}{\partial \eta} \right), \\
\xi_z &= J \left(\frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi} \frac{\partial x}{\partial \eta} \right).
\end{aligned} \tag{A.14}$$

It is important to see how the above metric terms from Eqs. (A.13) and (A.14) are related to geometrical quantities like the volume or the face vectors. Let us for this purpose consider a 2-D structured grid, as it is sketched in Fig. A.1. We assume that the ξ -coordinate corresponds to the i -direction, and the η -coordinate to the j -direction, respectively. Then, the derivatives of the Cartesian coordinates with respect to the curvilinear coordinates in Eq. (A.14) can be approximated at point (i, j) as

$$\begin{aligned}
\left(\frac{\partial x}{\partial \xi} \right)_{i,j} &\approx \frac{1}{2} \frac{[(x_{i,j} + x_{i+1,j}) - (x_{i,j} + x_{i-1,j})]}{(i+1/2) - (i-1/2)} = \frac{1}{2} (x_{i+1,j} - x_{i-1,j}), \\
\left(\frac{\partial y}{\partial \xi} \right)_{i,j} &\approx \frac{1}{2} \frac{[(y_{i,j} + y_{i+1,j}) - (y_{i,j} + y_{i-1,j})]}{(i+1/2) - (i-1/2)} = \frac{1}{2} (y_{i+1,j} - y_{i-1,j})
\end{aligned} \tag{A.15}$$

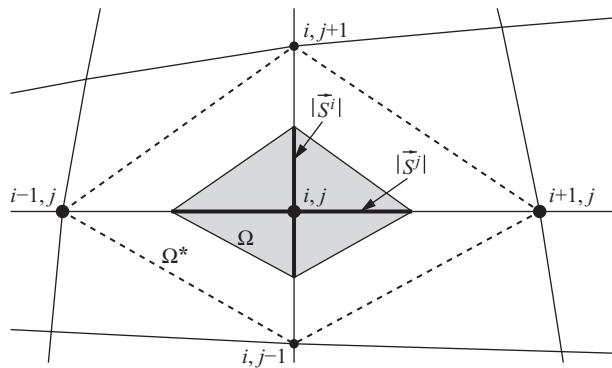


Figure A.1 Geometrical representation of the metric terms in two dimensions.

and

$$\begin{aligned}\left(\frac{\partial x}{\partial \eta}\right)_{i,j} &\approx \frac{1}{2} \frac{[(x_{i,j} + x_{i,j+1}) - (x_{i,j} + x_{i,j-1})]}{(j + 1/2) - (j - 1/2)} = \frac{1}{2} (x_{i,j+1} - x_{i,j-1}), \\ \left(\frac{\partial y}{\partial \eta}\right)_{i,j} &\approx \frac{1}{2} \frac{[(y_{i,j} + y_{i,j+1}) - (y_{i,j} + y_{i,j-1})]}{(j + 1/2) - (j - 1/2)} = \frac{1}{2} (y_{i,j+1} - y_{i,j-1}).\end{aligned}\quad (\text{A.16})$$

Thus, using the above approximations of Eqs. (A.15) and (A.16), the formulas in Eq. (A.14) become

$$\begin{aligned}\xi_x &= J \frac{\partial y}{\partial \eta} \approx J S_x^I, \\ \xi_y &= -J \frac{\partial x}{\partial \eta} \approx J S_y^I, \\ \eta_x &= -J \frac{\partial y}{\partial \xi} \approx J S_x^J, \\ \eta_y &= J \frac{\partial x}{\partial \xi} \approx J S_y^J,\end{aligned}\quad (\text{A.17})$$

where $\vec{S}^I = [S_x^I, S_y^I]^T$ denotes the average face vector in i -direction, and $\vec{S}^J = (S_x^J, S_y^J)^T$ the vector in j -direction, respectively (cf. Fig. A.1). The relationships in Eq. (A.17) can also be expressed in the form

$$\vec{S}^I = J^{-1} \begin{bmatrix} \xi_x \\ \xi_y \end{bmatrix}, \quad \vec{S}^J = J^{-1} \begin{bmatrix} \eta_x \\ \eta_y \end{bmatrix}. \quad (\text{A.18})$$

From the above we can see that the metric terms correspond to the components of the face vectors in the finite-volume scheme, divided by the determinant of the coordinate transformation Jacobian J .

The question is now, what is the geometrical interpretation of J^{-1} ? Using Eq. (A.13) which reads in two dimensions as

$$J^{-1} = \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi}, \quad (\text{A.19})$$

and inserting the expressions Eq. (A.17) for the metric terms into Eq. (A.19), it can be shown that

$$J^{-1} = 2\Omega = \frac{1}{2}\Omega^*. \quad (\text{A.20})$$

The volumes Ω and Ω^* (with constant depth $b=1$) are displayed in Fig. A.1. They are defined by the points $(i+1/2, j)$, $(i, j+1/2)$, $(i-1/2, j)$, $(i, j-1/2)$ in the case of Ω , and by the grid nodes $(i, j-1)$, $(i+1, j)$, $(i, j+1)$, $(i-1, j)$ in the case of Ω^* , respectively. It

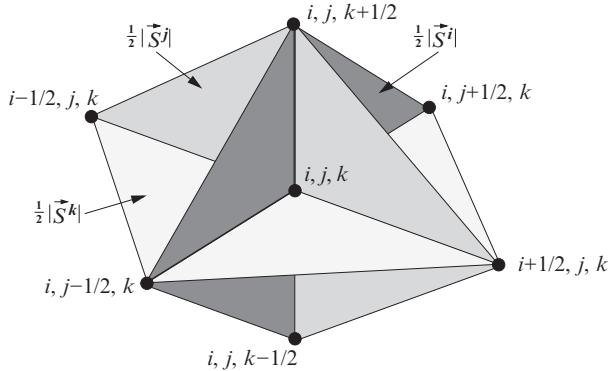


Figure A.2 Geometrical representation of the metric terms in three dimensions.

should be noted that on a rectangular grid J^{-1} equals to the dual control volume which was presented in Section 4.2.3.

The same derivations for the metric terms as above can be conducted in three dimensions. The results are shown in Fig. A.2. The inverse of the determinant of the transformation Jacobian (J^{-1}) equals to twice the volume defined by the grid nodes $(i, j, k - 1/2)$, $(i+1/2, j, k)$, $(i, j, k+1/2)$, $(i - 1/2, j, k)$, $(i, j - 1/2, k)$, and $(i, j+1/2, k)$. On a rectangular grid, this again corresponds exactly to the dual control volume of Section 4.2.3.

A.2 QUASILINEAR FORM OF THE EULER EQUATIONS

This particular form is instructive in understanding the mathematical properties of the governing equations. It is also useful in the development and analysis of numerical schemes.

The quasilinear form is derived from the governing equations in differential formulation (A.4). We will consider here only the Euler equations, i.e.,

$$\frac{\partial \vec{W}^*}{\partial t} + \frac{\partial \vec{F}_{c,1}}{\partial \xi} + \frac{\partial \vec{F}_{c,2}}{\partial \eta} + \frac{\partial \vec{F}_{c,3}}{\partial \zeta} = 0, \quad (\text{A.21})$$

where $\vec{W}^* = J^{-1} \vec{W}$ and the convective fluxes $\vec{F}_{c,1/2/3}$ are defined in Eq. (A.6). The linearization replaces the spatial derivatives of the fluxes by derivatives of the conservative variables. For example,

$$\frac{\partial \vec{F}_{c,1}}{\partial \xi} = \frac{\partial \vec{F}_{c,1}}{\partial \vec{W}} \frac{\partial \vec{W}}{\partial \xi}. \quad (\text{A.22})$$

The effect is that the spatial derivatives are now linear functions of the conserved variables. Using the definition of the convective flux Jacobian from Eq. (A.65), we obtain the quasilinear form of the Euler equations

$$J^{-1} \frac{\partial \vec{W}}{\partial t} + \bar{A}_{c,1} \frac{\partial \vec{W}}{\partial \xi} + \bar{A}_{c,2} \frac{\partial \vec{W}}{\partial \eta} + \bar{A}_{c,3} \frac{\partial \vec{W}}{\partial \zeta} = 0. \quad (\text{A.23})$$

The Jacobian $\bar{A}_{c,1}$ is given by the formula (A.68) (or Eq. (A.66) in 2D) with (n_x, n_y, n_z) replaced by $J^{-1}(\xi_x, \xi_y, \xi_z)$. The convective flux Jacobians $\bar{A}_{c,2}$ and $\bar{A}_{c,3}$ are defined in a similar way with derivatives of η and ζ , respectively. The eigenvalues of the Jacobians determine the speed and the direction of propagation of waves (with convective and acoustic modes) through the flow domain. Therefore, the eigenvalues determine the time step of explicit time-stepping schemes. The eigenvalues and eigenvectors of the Jacobians also play an important role in upwind spatial discretization schemes, where the stencil is biased in the direction of the wave propagation.

A.3 MATHEMATICAL CHARACTER OF THE GOVERNING EQUATIONS

The mathematical character of the partial differential equations (PDEs) is best explained through the classical example of the quasi-linear second-order equation

$$a \frac{\partial^2 U}{\partial x^2} + b \frac{\partial^2 U}{\partial x \partial y} + c \frac{\partial^2 U}{\partial y^2} = d, \quad (\text{A.24})$$

where U represents a general scalar function. The coefficients a , b , c , and d may be nonlinear functions of the coordinates, of U and of its first derivatives, but not of the second derivatives of U . Depending on the sign of the discriminant function $(b^2 - 4ac)$, three different classes of PDEs can be defined [1, 2]. Namely, if the discriminant is positive the equation is said to be *hyperbolic*, whereas if $(b^2 - 4ac) < 0$ it becomes *elliptic*. Finally, if $(b^2 - 4ac)$ is zero the PDE is denoted as *parabolic*.

The Navier-Stokes equations cannot be characterized in such an easy way. In fact, they are in general a mixture of all three classes, depending on the flow conditions and on the geometry of the problem. Nevertheless, it is worthwhile to illustrate the physical behavior of hyperbolic, parabolic, and elliptic PDEs which must be considered for proper mathematical formulation of solution methods in fluid dynamics.

A.3.1 Hyperbolic equations

If Eq. (A.24) is of hyperbolic type, it has two real characteristics. The situation is sketched in Fig. A.3. It is known from the theory that the information at point P influences only the region between the advancing characteristics. On the other hand, point P receives information only from the part of the domain between the characteristics AP and BP . Furthermore, the solution at point P depends only on that part of the boundary which is intercepted by and included within the two characteristic lines through point P , i.e., the interval AB . For example, point P obtains no information from point C , since P is not enclosed within the characteristics propagating from C . Therefore, we need to

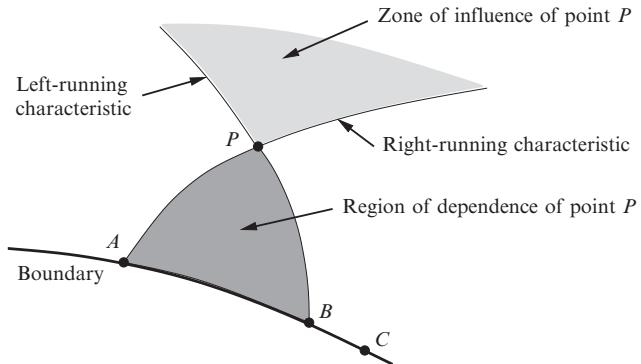


Figure A.3 Domain of dependence and influence of a hyperbolic PDE with two characteristics per point [3].

specify conditions at only one part of the boundary in order to determine the solution in a given region. Hence, the hyperbolic equations represent an *initial-value* problem.

In fluid dynamics, the following are examples of flows which are governed by hyperbolic PDEs:

1. *Steady, inviscid supersonic flow*: If the flow is 2-D, the behavior is like that already discussed above. For a 3-D flow, there are characteristic surfaces in the (x, y, z) space. The point P influences then the volume enclosed within the advancing (downstream) characteristic surface. The characteristic surface corresponds to the Mach cone in the case of the linearized potential equation.
2. *Unsteady flow*: The governing equations are in this case hyperbolic in *time*. Of course, the equations can be of different type in *space*. Hence, it then does not matter whether the flow is locally subsonic or supersonic. Because of the (partial) hyperbolic nature, it is necessary to specify an initial solution which is then advanced in time.

A.3.2 Parabolic equations

In this case, Eq. (A.24) has only one real characteristic. Figure A.4 shows the characteristic together with the domain of influence. For parabolic equations, information at point P influences the entire region on one side of the characteristic (BP in the sketch) and hence also the points C and D . On the other hand, the solution at point A is completely independent of that at point P . Hence, the information is transferred in one direction only, like for hyperbolic PDEs. Furthermore, as we can see from Fig. A.4, for example, the solution at point D depends on conditions along the whole characteristic BP as well as on those prescribed at the boundary segment BC . Thus, the parabolic equations represent a mixed *initial-* and *boundary-value* problem.

One particular form of the simplified governing equations, namely the so-called Parabolized Navier-Stokes (PNS) equations (see Section 2.4.3) exhibit the described

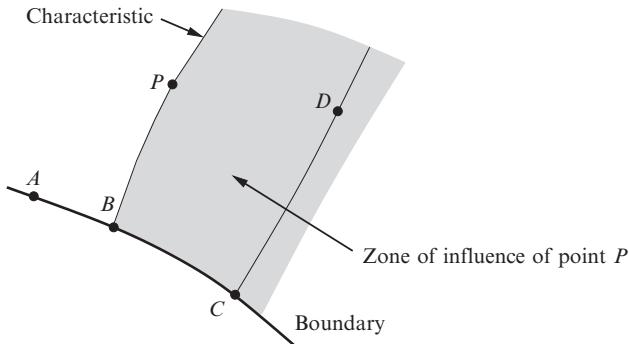


Figure A.4 Domain of influence of a parabolic PDE [3].

parabolic-type behavior. In this case, the viscous stress terms involving the derivatives with respect to the stream-wise direction are ignored in the governing equations. The solution of the PNS equations is started from some prescribed (initial) data at the inlet boundary. The simulation then proceeds by marching downstream. Each stream-wise station represents a plane (line in 2-D) which is perpendicular to the main flow direction and which is coupled in an explicit way to one or two upstream planes. In each plane, the equations have to be solved iteratively.

A further simplification of the Navier-Stokes equations for the case of high Reynolds numbers leads to the well-known boundary layer equations. These are also of parabolic type and they can be solved by a similar explicit space-marching procedure.

A.3.3 Elliptic equations

Finally, if Eq. (A.24) is of elliptic type, it has two complex characteristics. It is known from the theory that the information at some point P influences all other regions of the domain. On the other hand, the solution at point P depends on the whole surrounding domain. The situation is sketched in Fig. A.5. This implies that the solution at point P is influenced by the entire closed boundary (A, B, C). Therefore, the elliptic PDEs represent a *boundary-value* problem, where the solution at point P must be carried out simultaneously with the solution at all other points in the domain. This is in strong contrast to the space or time marching procedures applied to parabolic and hyperbolic equations. In terms of Fig. A.5, boundary conditions must be prescribed over the entire boundary (A, B, C). The boundary conditions can assume one of the two following forms:

1. Specification of the *dependent variables* along the boundary. This type of boundary condition is called the *Dirichlet condition*.
2. Specification of the *derivatives* of the dependent variables along the boundary. This type of boundary condition is named the *Neumann condition*.

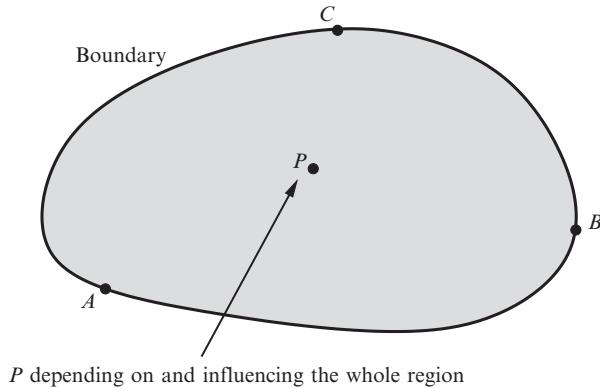


Figure A.5 Domain of influence and dependence of an elliptic PDE [3].

In fluid dynamics, the steady subsonic/incompressible inviscid flow is, for example, governed by elliptic equations. Hence, for such flows physical boundary conditions must be applied to a surface which completely surrounds the simulated flow field.

A.4 NAVIER-STOKES EQUATIONS IN ROTATING FRAME OF REFERENCE

In some instances, for example, in turbomachinery applications, for propellers or in geophysics, the computational domain is steadily rotating about some axis. In such case, it is convenient to transform the Navier-Stokes equations into a *rotating frame of reference*. Let us consider the situation which is depicted in Fig. A.6. Here, a point P rotates with the constant angular velocity $\vec{\omega}$ around a fixed axis. For the sake of simplicity, we assume that the rotation axis coincides with the x -coordinate axis. Thus, the angular velocity has the components $\vec{\omega} = [\omega_1, 0, 0]^T$. The absolute velocity \vec{v}_a results from the sum of the relative velocity \vec{v}_r and the entrainment velocity \vec{v}_e , i.e.,

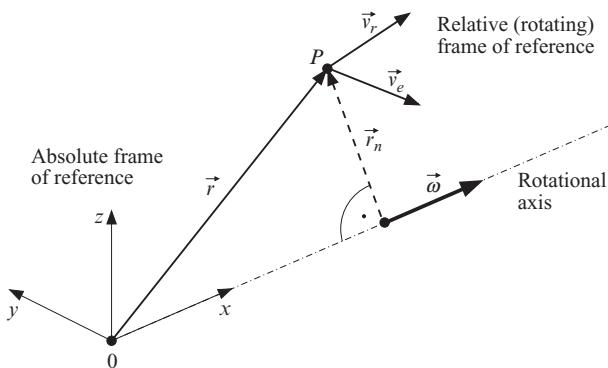


Figure A.6 Absolute and rotating frame of reference. The rotational axis coincides with the x -axis.

$$\vec{v}_a = \vec{v}_r + \vec{v}_e = \vec{v}_r + \vec{\omega} \times \vec{r}. \quad (\text{A.25})$$

If we rewrite the Navier-Stokes equations (2.19) in the relative frame of reference, we have to account for the effects due to the *Coriolis force* as well as due to the *centrifugal force*. The Coriolis force per unit mass is defined as

$$\vec{f}_{\text{Corr}} = -2(\vec{\omega} \times \vec{v}_r). \quad (\text{A.26})$$

The centrifugal force per unit mass is given by

$$\vec{f}_{\text{cen}} = -\vec{\omega} \times (\vec{\omega} \times \vec{r}) = \omega_1^2 \vec{r}_n, \quad (\text{A.27})$$

where \vec{r}_n denotes the position vector perpendicular to the rotation axis. In consequence, the source term \vec{Q} in Eq. (2.25) has to be extended by the sum of the Coriolis and the centrifugal forces for the momentum equations. Furthermore, the energy equation has to be modified because of the centrifugal force (the Coriolis force does not contribute to the energy balance). Only the continuity equation remains unchanged since the mass balance is invariant to a system rotation. Hence, the Navier-Stokes equations formulated in a relative frame of reference, which rotates with constant angular velocity about the x -axis, reads

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega + \oint_{\partial\Omega} (\vec{F}_c - \vec{F}_v) dS = \int_{\Omega} \vec{Q} d\Omega. \quad (\text{A.28})$$

The vector of the conservative variables \vec{W} consists of the following components

$$\vec{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}, \quad (\text{A.29})$$

where ρ denotes the density, u, v, w the Cartesian velocity components in the relative frame, and E the relative total energy per unit mass. The relative total energy is given by

$$E = e + \frac{|\vec{v}_r|^2}{2} - \frac{|\vec{v}_e|^2}{2} = e + \frac{u^2 + v^2 + w^2}{2} - \frac{\omega_1^2 |\vec{r}_n|^2}{2} \quad (\text{A.30})$$

with $|\vec{r}_n|^2 = y^2 + z^2$. The vector of the convective fluxes \vec{F}_c is defined as

$$\vec{F}_c = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho I V \end{bmatrix} \quad (\text{A.31})$$

with p being the static pressure, and n_x, n_y, n_z representing the components of the outward pointing unit normal vector of the surface $\partial\Omega$, respectively. Furthermore,

$$I = h + \frac{|\vec{v}_r|^2}{2} - \frac{|\vec{v}_e|^2}{2} = H - \frac{\omega_1^2 |\vec{r}_n|^2}{2},$$

$$V = n_x u + n_y v + n_z w,$$
(A.32)

where I stands for the *rothalpy*, H for the relative total enthalpy, and V for the contravariant velocity, respectively. The rothalpy represents the total energy content in a steadily rotating frame of reference.

The vector of the viscous fluxes \vec{F}_v retains the same form as presented in Eq. (2.23). Also the components of the viscous stress tensor remain formally as given by Eq. (2.15). However, the source term \vec{Q} is extended by the Coriolis and the centrifugal force to

$$\vec{Q} = \begin{bmatrix} 0 \\ \rho f_{e,x} \\ \rho \omega_1(y \omega_1 + 2w) + \rho f_{e,y} \\ \rho \omega_1(z \omega_1 - 2v) + \rho f_{e,z} \\ \rho \vec{f}_e \cdot \vec{v}_r + \dot{q}_h \end{bmatrix}$$
(A.33)

with $\rho \vec{f}_e$ being the body force per unit volume (in addition to Coriolis and centrifugal forces), and \dot{q}_h denoting the time rate of heat transfer per unit mass, respectively.

The transformed governing equations (A.28) are closed using thermodynamic relations between the state variables. In the case of a perfect gas, the pressure is computed from the formula

$$p = (\gamma - 1)\rho \left[E - \frac{u^2 + v^2 + w^2 - \omega_1^2 |\vec{r}_n|^2}{2} \right],$$
(A.34)

where γ denotes the ratio of the specific heat coefficients. Additionally, the viscosity and the thermal conductivity coefficients have to be supplied as a function of the state of the fluid.

A.5 NAVIER-STOKES EQUATIONS FORMULATED FOR MOVING GRIDS

In certain cases, for instance where the fluid-structure interaction is investigated or where a store separation is simulated, it is necessary to solve the governing equations on a moving and possibly also a deforming grid. The two most popular methodologies used to tackle such problems, are the *Arbitrary Lagrangian Eulerian* formulation [4–6] and the *dynamic grids* [7]. Both approaches are closely related and lead to the same modified form of the governing equations which accounts for the relative motion of the grid with respect to the fluid.

Written in the time-dependent integral form for a moving and/or deforming control volume Ω with a surface element dS , the Navier-Stokes equations (2.19) read

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega + \oint_{\partial\Omega} (\vec{F}_c^M - \vec{F}_v) dS = \int_{\Omega} \vec{Q} d\Omega. \quad (\text{A.35})$$

The vector of the conservative variables \vec{W} has the following components:

$$\vec{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}, \quad (\text{A.36})$$

where ρ denotes the density, u, v, w the Cartesian velocity components, and E the total energy per unit mass. The vector of the convective fluxes \vec{F}_c^M becomes on dynamic grids

$$\vec{F}_c^M = \vec{F}_c - V_t \vec{W} \quad (\text{A.37})$$

with \vec{F}_c given by Eq. (2.21) and V_t being the contravariant velocity of the face of the control volume. Hence,

$$V_t = \vec{g} \cdot \vec{n} = n_x \frac{\partial x}{\partial t} + n_y \frac{\partial y}{\partial t} + n_z \frac{\partial z}{\partial t}, \quad (\text{A.38})$$

where \vec{g} represents the grid velocity, and n_x, n_y, n_z denote the components of the outward facing unit normal vector of the surface $\partial\Omega$, respectively. Using Eq. (2.21), the convective fluxes \vec{F}_c^M can be written in the form

$$\vec{F}_c^M = \begin{bmatrix} \rho V_r \\ \rho u V_r + n_x p \\ \rho v V_r + n_y p \\ \rho w V_r + n_z p \\ \rho H V_r + V_t p \end{bmatrix}, \quad (\text{A.39})$$

where H stands for the total enthalpy and p for the static pressure, respectively. Furthermore, V_r represents the contravariant velocity relative to the motion of the grid, i.e.,

$$V_r = n_x u + n_y v + n_z w - V_t = V - V_t. \quad (\text{A.40})$$

The vector of the viscous fluxes \vec{F}_v and the source term \vec{Q} retain the same forms as already presented in Eqs. (2.23) and (2.25), respectively. The same holds also for the components of the viscous stress tensor Eq. (2.15). On the other hand, the Jacobian of the convective fluxes (Section A.9), its eigenvalues (Eq. (A.84) or A.88) and hence the spectral radii are changed due to the grid motion (V is replaced by V_r in Eq. (4.53), 5.33, or 9.77).

It was first pointed out by Thomas and Lombard [8] that besides the conservation of mass, momentum, and energy, the so-called *Geometric Conservation Law* (GCL) must be satisfied in order to avoid errors induced by a deformation of the control volumes [7, 9–11]. The GCL results from the requirement that the computation of the control volumes or of the grid velocities must be performed in such a way that the resulting numerical scheme preserves the state of a uniform flow, independently of the deformation of the grid [11]. The GCL is derived from the continuity equation. The equation for the mass conservation (2.3), formulated for moving grids, reads

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \, d\Omega + \oint_{\partial\Omega} \rho (V - V_t) \, dS = 0. \quad (\text{A.41})$$

Equation (A.41) must hold also for the case of a uniform fluid velocity (and hence V) and a constant density. Assuming that the control volume is always closed, the integral over the contravariant velocity V is zero. Herewith, we obtain the integral form of the GCL

$$\frac{\partial}{\partial t} \int_{\Omega} \, d\Omega - \oint_{\partial\Omega} V_t \, dS = 0. \quad (\text{A.42})$$

As we can see, the GCL relates the change of the control volume to the motion of its faces. It is obvious that the GCL is automatically satisfied for such moving grids, where the shapes of the control volumes do not change in time. It is essential that the GCL in Eq. (A.42) is temporally discretized using the same scheme as it is applied to the governing equations (A.35) in order to obtain a self-consistent solution method. Furthermore, the GCL has to be solved concurrently with the fluid equations. We shall demonstrate this on two exemplary temporal discretizations.

Explicit Time-Stepping Scheme

Let us begin with a basic explicit time-stepping scheme (forward Euler) given in Eq. (6.4). With this particular temporal discretization, Eq. (A.42) becomes

$$\Omega^{n+1} - \Omega^n = \int_t^{t+\Delta t} \oint_{\partial\Omega} (\vec{g} \cdot \vec{n}) \, dS \, dt. \quad (\text{A.43})$$

Since the faces of the control volume Ω consist of convex facets, the total change of the volume in Eq. (A.43) is equal to the sum of local changes at each facet. If we consider now one particular facet m of the control volume, like it is sketched in Fig. A.7, we can formulate the local GCL as

$$\Delta\Omega_m = \Omega_m^{n+1} - \Omega_m^n = \Delta t [(\vec{g} \cdot \vec{n}) \Delta S]_m^{n+1/2}. \quad (\text{A.44})$$

Under the assumption the grid motion is linear in time and in space, the term $\Delta\Omega_m$ in Eq. (A.44) represents the volume $ABC'A'B'C'$ in Fig. A.7 swept out by the facet during Δt . Since the local change of the control volume is known (in general with a certain

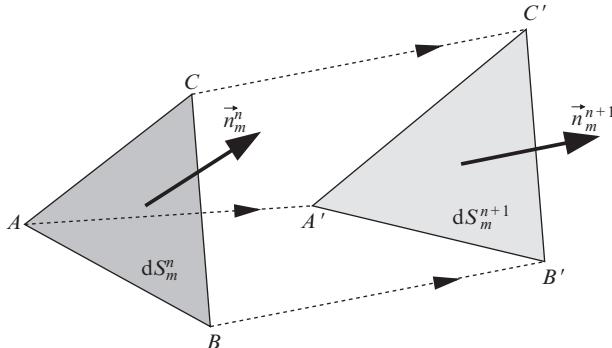


Figure A.7 Movement of a triangular facet m of the control volume between the time levels n and $n + 1$.

approximation in 3D), we can compute $V_t \Delta S$, which is needed for the evaluation of the convective fluxes \vec{F}_c^M in Eq. (A.35), directly from Eq. (A.44). The normal vector in Eq. (A.44) is defined as the arithmetic average of the normals at t and $t + \Delta t$, i.e.,

$$(\vec{n} \Delta S)_m^{n+1/2} = \frac{1}{2} [(\vec{n} \Delta S)_m^{n+1} + (\vec{n} \Delta S)_m^n]. \quad (\text{A.45})$$

It is used for the computation of the convective as well as the viscous fluxes in Eq. (A.35).

When we employ the explicit multistage scheme from Section 6.1.1 for the temporal discretization of Eq. (A.35), the flow solution at a stage k will be given by the relation

$$\vec{W}^{(k)} = \vec{W}^{(0)} \frac{\Omega^n}{\Omega^{n+1}} - \frac{\alpha_k \Delta t}{\Omega^{n+1}} \left[\sum_{m=1}^{N_F} (\vec{F}_c^M - \vec{F}_v)_m \Delta S_m^{n+1/2} - \vec{Q} \Omega^n \right], \quad (\text{A.46})$$

where N_F denotes the number of facets¹ of the control volume Ω . The convective and viscous fluxes as well as the source term are functions of the solution from the previous stage $\vec{W}^{(k-1)}$. Furthermore, the grid velocity is obtained from

$$[(\vec{g} \cdot \vec{n}) \Delta S]_m^{n+1/2} = \frac{\Omega_m^{n+1} - \Omega_m^n}{\alpha_k \Delta t}, \quad (\text{A.47})$$

in order to be consistent with the multistage scheme Eq. (A.46). Finally, the normal vector is computed by Eq. (A.45).

Implicit Time-Stepping Scheme

Let us consider the three-point backward Euler scheme from Eq. (6.79) for the second example of temporal discretization. Omitting the index I , the right-hand side of Eq. (6.79) becomes on a moving grid

¹ Note that the number of facets may be different from the number of faces; this is especially the case for the median-dual scheme on unstructured grids.

$$\vec{R}^{n+1} = \sum_{m=1}^{N_F} (\vec{F}_c^M - \vec{F}_v)_m \Delta S_m^{n+1} - \vec{Q}\Omega^{n+1}. \quad (\text{A.48})$$

The convective and viscous fluxes as well as the source term are in this case functions of the new solution \vec{W}^{n+1} . The grid velocity is computed in line with the time integration scheme from

$$[(\vec{g} \cdot \vec{n}) \Delta S]_m^{n+1} = \frac{3 \Omega_m^{n+1} - 4 \Omega_m^n + \Omega_m^{n-1}}{2\Delta t}. \quad (\text{A.49})$$

The normal vector, which is required for the evaluation of the convective and the viscous fluxes, has to be computed by extrapolation from two previous time levels

$$(\vec{n} \Delta S)_m^{n+1} = \frac{3}{2}(\vec{n} \Delta S)_m^{n+1/2} - \frac{1}{2}(\vec{n} \Delta S)_m^{n-1/2}, \quad (\text{A.50})$$

where the terms at the levels $(n + 1/2)$ and $(n - 1/2)$ are obtained from Eq. (A.45). The definitions in Eqs. (A.49) and (A.50) make sure that the temporal discretization remains of second order on a moving grid (compare to the conditions given in Ref. [12]).

Wall Boundary Conditions

As we discussed in Section 8.2, the first condition is that there is no flow normal to the wall. For a moving boundary, this translates to $V_r = 0$. When applied to Eq. (A.39), only the terms $\vec{n} p_w$ and $V_t p_w$ remain in $(\vec{F}_c^M)_w$. The wall pressure p_w is obtained in the same way as presented in Section 8.2, the wall velocity is given by Eq. (A.47) or (A.49), respectively. The second condition, which arises in the case of viscous flows, states that the fluid moves with the wall velocity, i.e., $\vec{v}_w = \vec{g}_w$. This effects the extrapolation into the dummy cells discussed in Section 8.2.2.

More details on the numerical implementation of the governing equations for moving and/or deforming grids and on the discretization of the GCL can be found in the above cited references and also, e.g., in Refs. [12–20]. A very detailed description of schemes for unstructured grids was presented in Refs. [15, 20].

A.6 THIN SHEAR LAYER APPROXIMATION

In the case of high Reynolds numbers, the flow is influenced by the viscous stresses only in a narrow region around the body. It is then reasonable to assume that the gradients in the wall normal direction dominate, and hence that the gradients in the other direction(s) can be neglected [21, 22]. We speak in such case of the so-called Thin Shear Layer (TSL) approximation of the Navier-Stokes equations.

Let us consider for illustration the differential form of the Navier-Stokes equations formulated for a general curvilinear grid (A.4). If we further assume that, as rendered in Fig. 2.4, the wall is located at $\eta = \text{const.}$, all derivatives in the stream-wise ($\partial/\partial\xi$) and in the cross-flow direction ($\partial/\partial\zeta$) are omitted from the diffusive terms. The 3-D TSL

Navier-Stokes equations in differential form then appear as follows in the curvilinear coordinate system (ξ , η , ζ)

$$\frac{\partial \vec{W}^*}{\partial t} + \frac{\partial \vec{F}_{c,1}}{\partial \xi} + \frac{\partial \vec{F}_{c,2}}{\partial \eta} + \frac{\partial \vec{F}_{c,3}}{\partial \zeta} = \frac{\partial \vec{F}_{v,2}}{\partial \eta} + \vec{Q}^*. \quad (\text{A.51})$$

The vectors of the conservative variables (\vec{W}^*), of the convective fluxes ($\vec{F}_{c,1/2/3}$), and of the diffusive flux ($\vec{F}_{v,2}$), respectively, are given by the relationships (A.5)–(A.7). Furthermore, the source term (\vec{Q}^*) also remains in the form given by Eq. (A.8). However, the components of the viscous stress tensor (Eq. (A.10)) and of the thermal fluxes transform to

$$\begin{aligned} \tau_{xx} &= 2\mu \eta_x \frac{\partial u}{\partial \eta} + \lambda \left(\eta_x \frac{\partial u}{\partial \eta} + \eta_y \frac{\partial v}{\partial \eta} + \eta_z \frac{\partial w}{\partial \eta} \right), \\ \tau_{yy} &= 2\mu \eta_y \frac{\partial v}{\partial \eta} + \lambda \left(\eta_x \frac{\partial u}{\partial \eta} + \eta_y \frac{\partial v}{\partial \eta} + \eta_z \frac{\partial w}{\partial \eta} \right), \\ \tau_{zz} &= 2\mu \eta_z \frac{\partial w}{\partial \eta} + \lambda \left(\eta_x \frac{\partial u}{\partial \eta} + \eta_y \frac{\partial v}{\partial \eta} + \eta_z \frac{\partial w}{\partial \eta} \right), \\ \tau_{xy} &= \tau_{yx} = \mu \left(\eta_y \frac{\partial u}{\partial \eta} + \eta_x \frac{\partial v}{\partial \eta} \right), \\ \tau_{xz} &= \tau_{zx} = \mu \left(\eta_z \frac{\partial u}{\partial \eta} + \eta_x \frac{\partial w}{\partial \eta} \right), \\ \tau_{yz} &= \tau_{zy} = \mu \left(\eta_z \frac{\partial v}{\partial \eta} + \eta_y \frac{\partial w}{\partial \eta} \right), \\ \Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k \eta_x \frac{\partial T}{\partial \eta}, \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k \eta_y \frac{\partial T}{\partial \eta}, \\ \Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + k \eta_z \frac{\partial T}{\partial \eta}. \end{aligned} \quad (\text{A.52})$$

In the above formulas (A.52), the partial derivatives with respect to the Cartesian coordinates were approximated as

$$\begin{aligned} \frac{\partial u}{\partial x} &\approx \eta_x \frac{\partial u}{\partial \eta}, \\ \frac{\partial u}{\partial y} &\approx \eta_y \frac{\partial u}{\partial \eta}, \\ \frac{\partial u}{\partial z} &\approx \eta_z \frac{\partial u}{\partial \eta}, \quad \text{etc.} \end{aligned} \quad (\text{A.53})$$

in accordance with the TSL assumption.

A.7 PNS EQUATIONS

In certain cases, where the following three conditions are met:

- the flow is steady (i.e., $\partial \vec{W}/\partial t = 0$),
- the fluid moves predominantly in one main direction (there must be no boundary layer separation),
- the cross-flow components are negligible,

the Navier-Stokes equations (2.19) can be simplified to a form called the PNS equations [23]. Then, the derivatives of the velocity components and of the temperature with respect to the stream-wise direction can be dropped from the viscous stress terms. Furthermore, the components in the stream-wise direction of the viscous stress tensor $\bar{\tau}$ and of its work $(\bar{\tau} \cdot \vec{v})$ can be neglected in the viscous flux vector in Eq. (2.23).

In order to demonstrate the PNS approach, let us consider the differential form of the Navier-Stokes equations written for a general curvilinear grid as given by Eq. (A.4). Let us also assume that the stream-wise direction corresponds to the coordinate ξ and the cross-flow directions to the coordinates η and ζ , respectively. Then, the differential form of the 3-D PNS equations reads in a curvilinear coordinate system (ξ, η, ζ)

$$\frac{\partial \vec{W}^*}{\partial t} + \frac{\partial \vec{F}_{c,1}}{\partial \xi} + \frac{\partial \vec{F}_{c,2}}{\partial \eta} + \frac{\partial \vec{F}_{c,3}}{\partial \zeta} = \frac{\partial \vec{F}_{v,2}}{\partial \eta} + \frac{\partial \vec{F}_{v,3}}{\partial \zeta} + \vec{Q}^*. \quad (\text{A.54})$$

Hence, the viscous flux in the stream-wise direction ($\vec{F}_{v,1}$) was assumed to be zero. The partial derivatives of the Cartesian velocity components and of the temperature, which appear in the viscous stress tensor and in the thermal fluxes (A.10), are approximated as

$$\begin{aligned} \frac{\partial u}{\partial x} &\approx \eta_x \frac{\partial u}{\partial \eta} + \zeta_x \frac{\partial u}{\partial \zeta}, \\ \frac{\partial u}{\partial y} &\approx \eta_y \frac{\partial u}{\partial \eta} + \zeta_y \frac{\partial u}{\partial \zeta}, \\ \frac{\partial u}{\partial z} &\approx \eta_z \frac{\partial u}{\partial \eta} + \zeta_z \frac{\partial u}{\partial \zeta}, \quad \text{etc.} \end{aligned} \quad (\text{A.55})$$

within the PNS approach instead of using the exact formulas (A.12). The definition of the conservative variables (A.5), the relations for the convective fluxes (A.6), for the two remaining viscous fluxes (A.7), as well as for the source term (A.8) remain all unchanged.

A.8 AXISYMMETRIC FORM OF THE NAVIER-STOKES EQUATIONS

In cases where flow is simulated around cylindrical bodies or inside cylindrical cavities, it can often be assumed that it is symmetrical with respect to the longitudinal axis. This means that there is no swirl, and also that the derivatives of the flow quantities

in the circumferential direction are zero. Under these conditions, one momentum equation can be dropped from the Navier-Stokes equations (2.19). Thus, the governing equations are reduced from three to two dimensions, which lead to a substantially reduced computational effort.

The axisymmetric form of the Navier-Stokes equations reads

$$\frac{\partial}{\partial t} \int_{\Omega} \vec{W} d\Omega + \oint_{\partial\Omega} (\vec{F}_c - \vec{F}_v) r dS = \int_{\Omega} \vec{Q} d\Omega, \quad (\text{A.56})$$

where r is the coordinate in the radial direction. The vector of the conservative variables consists of the four components

$$\vec{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \quad (\text{A.57})$$

with u and v being here the velocity components in the axial and the radial direction, respectively. The vector of the convective fluxes is given by

$$\vec{F}_c = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_r p \\ \rho H V \end{bmatrix}, \quad (\text{A.58})$$

where n_x is the unit normal vector in the axial direction, n_r in the radial direction, and the contravariant velocity $V = n_x u + n_r v$. The vector of the viscous fluxes is defined as

$$\vec{F}_v = \begin{bmatrix} 0 \\ n_x \tau_{xx} + n_r \tau_{xr} \\ n_x \tau_{rx} + n_r \tau_{rr} \\ n_x \Theta_x + n_r \Theta_r \end{bmatrix}, \quad (\text{A.59})$$

where

$$\begin{aligned} \Theta_x &= u \tau_{xx} + v \tau_{xr} + k \frac{\partial T}{\partial x}, \\ \Theta_r &= u \tau_{rx} + v \tau_{rr} + k \frac{\partial T}{\partial r} \end{aligned} \quad (\text{A.60})$$

are the terms describing the work of the viscous stresses and of the heat conduction in the fluid. Finally, the source term reads

$$\vec{Q} = \frac{1}{r} \begin{bmatrix} 0 \\ 0 \\ p - \tau_{\theta\theta} \\ 0 \end{bmatrix}. \quad (\text{A.61})$$

The components of the viscous stress tensor are given by the relations

$$\begin{aligned}\tau_{xx} &= -\frac{2\mu}{3} \vec{\nabla} \cdot \vec{v} + 2\mu \frac{\partial u}{\partial x} \\ \tau_{rr} &= -\frac{2\mu}{3} \vec{\nabla} \cdot \vec{v} + 2\mu \frac{\partial v}{\partial r} \\ \tau_{\theta\theta} &= -\frac{2\mu}{3} \vec{\nabla} \cdot \vec{v} + 2\mu \frac{v}{r} \\ \tau_{xr} = \tau_{rx} &= \mu \left(\frac{\partial u}{\partial r} + \frac{\partial v}{\partial x} \right)\end{aligned}\tag{A.62}$$

with the divergence of the velocity

$$\vec{\nabla} \cdot \vec{v} = \frac{\partial u}{\partial x} + \frac{1}{r} \frac{\partial (rv)}{\partial r} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial r} + \frac{v}{r}.\tag{A.63}$$

The boundary conditions at the symmetry line read

$$\begin{aligned}\rho v &= 0 \\ \frac{\partial u}{\partial r} &= \frac{\partial T}{\partial r} = \frac{\partial p}{\partial r} = 0.\end{aligned}\tag{A.64}$$

It should be noted that the volume Ω is the product of the area of the control volume with an average radius r . Thus as we can see, the axisymmetric equations (A.56) are like the 2-D Navier-Stokes equations, except for the multiplication by the radius, the additional source term $(p - \tau_{\theta\theta})/r$, and the changed definition of $\vec{\nabla} \cdot \vec{v}$.

A.9 CONVECTIVE FLUX JACOBIAN

The convective flux Jacobian represents the gradient of the convective fluxes with respect to the conservative variables, i.e.,

$$\bar{A}_c = \frac{\partial \vec{F}_c}{\partial \vec{W}},\tag{A.65}$$

where \vec{F}_c is given by Eq. (2.21).

2-D Formulation

The Jacobian matrix of the convective fluxes takes in two dimensions the following form [24]:

$$\bar{A}_c = \begin{bmatrix} -V_t & n_x & n_y & 0 \\ n_x\phi - uV & V - V_t - a_3n_xu & n_yu - a_2n_xv & a_2n_x \\ n_y\phi - vV & n_xv - a_2n_yu & V - V_t - a_3n_yv & a_2n_y \\ V(\phi - a_1) & n_xa_1 - a_2uV & n_ya_1 - a_2vV & \gamma V - V_t \end{bmatrix}, \quad (\text{A.66})$$

where

$$\begin{aligned} a_1 &= \gamma E - \phi, \\ a_2 &= \gamma - 1, \\ a_3 &= \gamma - 2, \\ V &= n_xu + n_yv, \\ \phi &= \frac{1}{2}(\gamma - 1)(u^2 + v^2) \end{aligned} \quad (\text{A.67})$$

and n_x , n_y denote the Cartesian components of the unit normal vector \vec{n} (Fig. 2.1). Furthermore, γ stands for the ratio of specific heat coefficients and V represents the contravariant velocity. The contravariant velocity of the face of the control volume (V_t —see Eq. (A.38)) is set to zero for stationary grids.

3-D Formulation

The expression for the convective flux Jacobian reads in three dimensions [24]

$$\bar{A}_c = \begin{bmatrix} -V_t & n_x & n_y & 0 \\ n_x\phi - uV & V - V_t - a_3n_xu & n_yu - a_2n_xv & a_2n_x \\ n_y\phi - vV & n_xv - a_2n_yu & V - V_t - a_3n_yv & a_2n_y \\ n_z\phi - wV & n_xw - a_2n_zu & n_yw - a_2n_zv & a_2n_z \\ V(\phi - a_1) & n_xa_1 - a_2uV & n_ya_1 - a_2vV & 0 \\ n_z & & & \\ n_zu - a_2n_xw & a_2n_x & & \\ n_zv - a_2n_yw & a_2n_y & & \\ V - V_t - a_3n_zw & a_2n_z & & \\ n_za_1 - a_2wV & \gamma V - V_t & & \end{bmatrix} \quad (\text{A.68})$$

with the abbreviations

$$\begin{aligned} a_1 &= \gamma E - \phi, \\ a_2 &= \gamma - 1, \\ a_3 &= \gamma - 2, \\ V &= n_xu + n_yv + n_zw, \\ \phi &= \frac{1}{2}(\gamma - 1)(u^2 + v^2 + w^2). \end{aligned} \quad (\text{A.69})$$

In the above relations (A.68) and (A.69), respectively, n_x , n_y , and n_z denote the Cartesian components of the unit normal vector \vec{n} (see Fig. 2.1). In the case of stationary grids, we have to set $V_t = 0$.

A.10 VISCOUS FLUX JACOBIAN

The viscous flux Jacobian represents the gradient of the viscous fluxes with respect to the conservative variables, i.e.,

$$\bar{A}_v = \frac{\partial \vec{F}_v}{\partial \vec{W}}, \quad (\text{A.70})$$

where \vec{F}_v is given by Eq. (2.23). In the following, we shall consider for simplicity the TSL approximation of the Navier-Stokes equations only (cf. Sections 2.4.3 and A.6). Furthermore, it is assumed that the dynamic viscosity and the thermal conductivity coefficients are frozen with respect to changes of the conservative variables and in space.

2-D Formulation

The 2-D viscous flux Jacobian reads in curvilinear coordinates for the TSL approximation [24]

$$\bar{A}_v = \frac{\mu}{J} \begin{bmatrix} 0 & 0 & 0 & 0 \\ b_{21} & a_1 \partial_\psi(\rho^{-1}) & a_2 \partial_\psi(\rho^{-1}) & 0 \\ b_{31} & a_2 \partial_\psi(\rho^{-1}) & a_3 \partial_\psi(\rho^{-1}) & 0 \\ b_{41} & b_{42} & b_{43} & a_4 \partial_\psi(\rho^{-1}) \end{bmatrix}, \quad (\text{A.71})$$

where J represents the determinant of the coordinate transformation Jacobian correspondingly to Eq. (A.13). The coefficients a and b are given by

$$\begin{aligned} b_{21} &= -a_1 \partial_\psi(u/\rho) - a_2 \partial_\psi(v/\rho), \\ b_{31} &= -a_2 \partial_\psi(u/\rho) - a_3 \partial_\psi(v/\rho), \\ b_{41} &= a_4 \partial_\psi [(u^2 + v^2)/\rho - E/\rho] - a_1 \partial_\psi(u^2/\rho) \\ &\quad - 2a_2 \partial_\psi(uv/\rho) - a_3 \partial_\psi(v^2/\rho), \\ b_{42} &= -a_4 \partial_\psi(u/\rho) - b_{21}, \\ b_{43} &= -a_4 \partial_\psi(v/\rho) - b_{31}, \\ a_1 &= (4/3)\psi_x^2 + \psi_y^2, \\ a_2 &= (1/3)\psi_x\psi_y, \\ a_3 &= \psi_x^2 + (4/3)\psi_y^2, \\ a_4 &= (\gamma/Pr) (\psi_x^2 + \psi_y^2). \end{aligned} \quad (\text{A.72})$$

In the above relations, μ stands for the dynamic viscosity coefficient, γ for the ratio of specific heat coefficients and Pr denotes the Prandtl number, respectively.

The terms $\partial_\psi()$ = $\partial()/\partial\psi$ must be conceived as operators. Let us consider for illustration the term $a_1\partial_\psi(\rho^{-1})$ in Eq. (A.71). Within many implicit schemes, the viscous flux Jacobian \bar{A}_v is multiplied by the change of the conservative variables $\Delta \vec{W} = \vec{W}^{n+1} - \vec{W}^n$. Hence, the complete term would read

$$[J^{-1}\mu a_1\partial_\psi(\rho^{-1})] \Delta W_2 = A_{22} \Delta W_2. \quad (\text{A.73})$$

If we now discretize the above term on structured grid using backward differences, we obtain at the mid-point ($i+1/2$)

$$[A_{22} \Delta W_2]_{i+1/2} = J_{i+1/2}^{-1} \mu_{i+1/2} (a_1)_{i+1/2} \frac{\left(\frac{\Delta W_2}{\rho} \right)_{i+1} - \left(\frac{\Delta W_2}{\rho} \right)_i}{\Delta \psi}. \quad (\text{A.74})$$

Jacobian matrices for specific coordinate directions result if we set $\psi = \xi$ or $\psi = \eta$, respectively.

3-D Formulation

The 3-D viscous flux Jacobian takes in curvilinear coordinates the following form, if we utilize the TSL approximation [24]

$$\bar{A}_v = \frac{\mu}{J} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ b_{21} & a_1\partial_\psi(\rho^{-1}) & a_2\partial_\psi(\rho^{-1}) & a_3\partial_\psi(\rho^{-1}) & 0 \\ b_{31} & a_2\partial_\psi(\rho^{-1}) & a_4\partial_\psi(\rho^{-1}) & a_5\partial_\psi(\rho^{-1}) & 0 \\ b_{41} & a_3\partial_\psi(\rho^{-1}) & a_5\partial_\psi(\rho^{-1}) & a_6\partial_\psi(\rho^{-1}) & 0 \\ b_{51} & b_{52} & b_{53} & b_{54} & a_7\partial_\psi(\rho^{-1}) \end{bmatrix}, \quad (\text{A.75})$$

where

$$\begin{aligned} b_{21} &= -a_1\partial_\psi(u/\rho) - a_2\partial_\psi(v/\rho) - a_3\partial_\psi(w/\rho), \\ b_{31} &= -a_2\partial_\psi(u/\rho) - a_4\partial_\psi(v/\rho) - a_5\partial_\psi(w/\rho), \\ b_{41} &= -a_3\partial_\psi(u/\rho) - a_5\partial_\psi(v/\rho) - a_6\partial_\psi(w/\rho), \\ b_{51} &= a_7\partial_\psi[(u^2 + v^2 + w^2)/\rho - E/\rho] - \\ &\quad a_1\partial_\psi(u^2/\rho) - a_4\partial_\psi(v^2/\rho) - a_6\partial_\psi(w^2/\rho) \\ &\quad - 2a_2\partial_\psi(uv/\rho) - 2a_3\partial_\psi(uw/\rho) - 2a_5\partial_\psi(vw/\rho), \\ b_{52} &= -a_7\partial_\psi(u/\rho) - b_{21}, \\ b_{53} &= -a_7\partial_\psi(v/\rho) - b_{31}, \\ b_{54} &= -a_7\partial_\psi(w/\rho) - b_{41} \end{aligned} \quad (\text{A.76})$$

and

$$\begin{aligned}
a_1 &= (4/3)\psi_x^2 + \psi_y^2 + \psi_z^2, \\
a_2 &= (1/3)\psi_x\psi_y, \\
a_3 &= (1/3)\psi_x\psi_z, \\
a_5 &= (1/3)\psi_y\psi_z, \\
a_4 &= \psi_x^2 + (4/3)\psi_y^2 + \psi_z^2, \\
a_6 &= \psi_x^2 + \psi_y^2 + (4/3)\psi_z^2, \\
a_7 &= (\gamma/Pr) (\psi_x^2 + \psi_y^2 + \psi_z^2).
\end{aligned} \tag{A.77}$$

Again, the terms $\partial_\psi() = \partial()/\partial\psi$ have to be treated as operators. Jacobian matrices for specific coordinate directions are obtained by setting $\psi = \xi$ or $\psi = \eta$ or $\psi = \zeta$, respectively.

In the case of a finite-volume discretization, the metric terms ψ_x , ψ_y , and ψ_z , can be transformed into the components of the face vector $\vec{S} = \vec{n}\Delta S$ according to Eq. (A.17). Furthermore, the determinant of the Jacobian of the coordinate transformation J is replaced by the inverse of the volume formed around the face of the control volume as indicated by Fig. A.1 and by Eq. (A.20). Thus, if we repeat the example of Eq. (A.73) for a finite-volume scheme, the product $J^{-1}a_1$ in Eq. (A.72) will read

$$J^{-1}a_1 = \frac{2}{\Omega^*} \left(\frac{4}{3}S_x^2 + S_y^2 \right). \tag{A.78}$$

It should be noted that the face area was included in Eq. (A.78) to reflect the multiplication of the viscous fluxes with ΔS when the residual is computed— \vec{F}_v itself contains only the components of the unit normal vector. The discretization at the mid-point ($i+1/2$) then becomes

$$[A_{22} \Delta W_2]_{i+1/2} = \frac{2\mu_{i+1/2}}{\Omega_{i+1/2}^*} \left(\frac{4}{3}S_x^2 + S_y^2 \right) \left[\left(\frac{\Delta W_2}{\rho} \right)_{i+1} - \left(\frac{\Delta W_2}{\rho} \right)_i \right], \tag{A.79}$$

where we assumed $\Delta\psi = 1$ due to the definition of \vec{S}^I and \vec{S}^J (see Fig. A.1 and Eqs. (A.15) and (A.16)).

A.11 TRANSFORMATION FROM CONSERVATIVE TO CHARACTERISTIC VARIABLES

The convective flux Jacobian (Section A.9) has real eigenvalues and a complete set of eigenvectors. Therefore, the Jacobians can be diagonalized as [25]

$$\bar{A}_c = \bar{T} \bar{\Lambda}_c \bar{T}^{-1}, \tag{A.80}$$

where \bar{T}^{-1} denotes the matrix of the left eigenvectors, \bar{T} of the right eigenvectors and $\bar{\Lambda}_c$ represents the diagonal matrix of the eigenvalues, respectively. The transformation from conservative (\bar{W}) to characteristic variables (\bar{C}) reads for an arbitrary variation δ (like ∂_t or $\vec{\nabla}$)

$$\delta \bar{C} = \bar{T}^{-1} \delta \bar{W}. \quad (\text{A.81})$$

The diagonalization of the convective flux Jacobian can be viewed as a decomposition into different waves. The right eigenvectors represent the waves, the characteristic variables are the wave amplitudes, and finally the eigenvalues are the associated wave speeds. In the context of the diagonalization that will be presented below, we differentiate between two types of waves. First, there are *convective* or *linear* waves, which are connected to eigenvalues of the type $\Lambda_c = \vec{v} \cdot \vec{n}$. Second, there are *acoustic* waves which are related to eigenvalues of the type $\Lambda_c = \vec{v} \cdot \vec{n} \pm c$. It should be noted that the diagonalization cannot be conducted simultaneously in multiple spatial directions [26]. This is the reason why the flux-difference splitting and the TVD schemes employ only 1-D wave decomposition. Only the fluctuation-splitting schemes (see Section 3.1.5) use an *approximate* multidimensional decomposition.

2-D Formulation

The matrix of the left eigenvectors of \bar{A}_c appears as follows in two dimensions [24]

$$\bar{T}^{-1} = \begin{bmatrix} (1 - \phi c^{-2}) & a_1 u c^{-2} \\ -(n_x u - n_y v) \rho^{-1} & n_y \rho^{-1} \\ a_2(\phi - cV) & a_2(n_x c - a_1 u) \\ a_2(\phi + cV) & -a_2(n_x c + a_1 u) \\ a_1 v c^{-2} & -a_1 c^{-2} \\ -n_x \rho^{-1} & 0 \\ a_2(n_y c - a_1 v) & a_1 a_2 \\ -a_2(n_y c + a_1 v) & a_1 a_2 \end{bmatrix}. \quad (\text{A.82})$$

The matrix of the right eigenvectors assumes the form [24]

$$\bar{T} = \begin{bmatrix} 1 & 0 & a_3 & a_3 \\ u & n_y \rho & a_3(u + n_x c) & a_3(u - n_x c) \\ v & -n_x \rho & a_3(v + n_y c) & a_3(v - n_y c) \\ \phi a_1^{-1} & \rho(n_y u - n_x v) & a_3(a_4 + cV) & a_3(a_4 - cV) \end{bmatrix}. \quad (\text{A.83})$$

The diagonal matrix contains the real eigenvalues

$$\bar{\Lambda}_c = \begin{bmatrix} \Lambda_1 & 0 & 0 & 0 \\ 0 & \Lambda_2 & 0 & 0 \\ 0 & 0 & \Lambda_3 & 0 \\ 0 & 0 & 0 & \Lambda_4 \end{bmatrix}. \quad (\text{A.84})$$

The following definitions apply in the above relationships:

$$\begin{aligned}
a_1 &= \gamma - 1, \\
a_2 &= \frac{1}{\rho c \sqrt{2}}, \\
a_3 &= \frac{\rho}{c \sqrt{2}}, \\
a_4 &= \frac{\phi + c^2}{\gamma - 1}, \\
V &= n_x u + n_y v, \\
\phi &= \frac{1}{2}(\gamma - 1)(u^2 + v^2), \\
\Lambda_1 &= \Lambda_2 = V - V_t, \\
\Lambda_3 &= V - V_t + c, \\
\Lambda_4 &= V - V_t - c.
\end{aligned} \tag{A.85}$$

Furthermore, γ denotes the ratio of specific heat coefficients, c the speed of sound, $\vec{n} = [n_x, n_y]^T$ the unit normal vector, and V the contravariant velocity, respectively. Finally, V_t represents the contravariant velocity of the face of the control volume as given by Eq. (A.38). In the case of stationary grids, V_t has to be set to zero.

3-D Formulation

The matrix of the left eigenvectors of \bar{A}_c becomes in three dimensions [24]

$$\bar{T}^{-1} = \begin{bmatrix} n_x a_5 - (n_z v - n_y w) \rho^{-1} & n_x a_1 u c^{-2} & n_x a_1 v c^{-2} + n_z \rho^{-1} \\ n_y a_5 - (n_x w - n_z u) \rho^{-1} & n_y a_1 u c^{-2} - n_z \rho^{-1} & n_y a_1 v c^{-2} \\ n_z a_5 - (n_y u - n_x v) \rho^{-1} & n_z a_1 u c^{-2} + n_y \rho^{-1} & n_z a_1 v c^{-2} - n_x \rho^{-1} \\ a_2(\phi - cV) & -a_2(a_1 u - n_x c) & -a_2(a_1 v - n_y c) \\ a_2(\phi + cV) & -a_2(a_1 u + n_x c) & -a_2(a_1 v + n_y c) \\ n_x a_1 w c^{-2} - n_y \rho^{-1} & -n_x a_1 c^{-2} \\ n_y a_1 w c^{-2} - n_x \rho^{-1} & -n_y a_1 c^{-2} \\ n_z a_1 w c^{-2} & -n_z a_1 c^{-2} \\ -a_2(a_1 w - n_z c) & a_1 a_2 \\ -a_2(a_1 w + n_z c) & a_1 a_2 \end{bmatrix}. \tag{A.86}$$

The matrix of the right eigenvectors reads [24]

$$\bar{T} = \begin{bmatrix} n_x & n_y & n_z \\ n_x u & n_y u - n_z \rho & n_z u + n_y \rho \\ n_x v + n_z \rho & n_y v & n_z v - n_x \rho \\ n_x w - n_y \rho & n_y w + n_x \rho & n_z w \\ n_x a_6 + \rho(n_z v - n_y w) & n_y a_6 + \rho(n_x w - n_z u) & n_z a_6 + \rho(n_y u - n_x v) \\ a_3 & a_3 & a_3 \\ a_3(u + n_x c) & a_3(u - n_x c) & a_3(v + n_y c) \\ a_3(v - n_y c) & a_3(w + n_z c) & a_3(w - n_z c) \\ a_3(a_4 + cV) & a_3(a_4 - cV) & \end{bmatrix}. \quad (\text{A.87})$$

The diagonal matrix contains the real eigenvalues

$$\bar{\Lambda}_c = \begin{bmatrix} \Lambda_1 & 0 & 0 & 0 & 0 \\ 0 & \Lambda_2 & 0 & 0 & 0 \\ 0 & 0 & \Lambda_3 & 0 & 0 \\ 0 & 0 & 0 & \Lambda_4 & 0 \\ 0 & 0 & 0 & 0 & \Lambda_5 \end{bmatrix}. \quad (\text{A.88})$$

The following abbreviations were used:

$$\begin{aligned} a_1 &= \gamma - 1, \\ a_2 &= \frac{1}{\rho c \sqrt{2}}, \\ a_3 &= \frac{\rho}{c \sqrt{2}}, \\ a_4 &= \frac{\phi + c^2}{\gamma - 1}, \\ a_5 &= 1 - \frac{\phi}{c^2}, \\ a_6 &= \frac{\phi}{\gamma - 1}, \\ V &= n_x u + n_y v + n_z w, \\ \phi &= \frac{1}{2}(\gamma - 1)(u^2 + v^2 + w^2), \\ \Lambda_1 &= \Lambda_2 = \Lambda_3 = V - V_t, \\ \Lambda_4 &= V - V_t + c, \\ \Lambda_5 &= V - V_t - c. \end{aligned} \quad (\text{A.89})$$

In the above relations Eqs. (A.86)–(A.89), γ denotes the ratio of specific heat coefficients, c the speed of sound, $\vec{n} = [n_x, n_y, n_z]^T$ the unit normal vector, and V the contravariant velocity, respectively. Finally, V_t represents the contravariant velocity of the face of the control volume as given by Eq. (A.38). In the case of stationary grids, V_t has to be set to zero.

A.12 GMRES ALGORITHM

Consider the system of linear equations

$$\bar{A}\vec{x} = \vec{b}. \quad (\text{A.90})$$

We are looking for an approximate solution of the form

$$\vec{x} = \vec{x}_0 + \vec{z}, \quad (\text{A.91})$$

where \vec{x}_0 represents an initial guess and \vec{z} is a member of the Krylov subspace

$$\vec{z} \in \mathcal{K}_m, \quad \mathcal{K}_m \equiv \text{span} \left\{ \vec{r}_0, \bar{A}\vec{r}_0, \bar{A}^2\vec{r}_0, \dots, \bar{A}^{m-1}\vec{r}_0 \right\} \quad (\text{A.92})$$

with $\vec{r}_0 = \vec{b} - \bar{A}\vec{x}_0$, and m being the dimension of \mathcal{K} . The parameter m is also termed the number of search directions. The Generalized Minimal Residual (GMRES) algorithm [27] determines \vec{z} in such a way that the 2-norm of the residual, i.e.,

$$\| \vec{b} - \bar{A}(\vec{x}_0 + \vec{z}) \| \quad (\text{A.93})$$

is minimized. In the following, we present the particular steps of the GMRES algorithm.

A.12.1 Computation of the orthonormal basis of \mathcal{K}_m

We employ the modified Gram-Schmidt procedure

```

 $\vec{r}_0 = \vec{b} - \bar{A}\vec{x}_0$ 
 $\vec{v}_1 = \vec{r}_0 / \| \vec{r}_0 \|$ 

DO  $j = 1, m$ 
 $\vec{v}_{j+1} = \bar{A}\vec{v}_j$ 
DO  $i = 1, j$ 
 $h_{i,j} = \vec{v}_{j+1} \cdot \vec{v}_i$ 
 $\vec{v}_{j+1} = \vec{v}_{j+1} - h_{i,j} \vec{v}_i$ 
ENDDO
 $h_{j+1,j} = \| \vec{v}_{j+1} \|$ 
 $\vec{v}_{j+1} = \vec{v}_{j+1} / h_{j+1,j}$ 
ENDDO

```

where $h_{i,j}$ denotes the coefficients of the upper Hessenberg matrix ($i = \text{line}, j = \text{column}$). However, the matrix is extended by the elements $h_{j+1,j}$. Therefore, the dimensions become $(m + 1) \times m$.

A.12.2 Generation of the upper Hessenberg matrix

The matrix has the following almost triangular form

$$\bar{H}_m^* = \begin{bmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,m-1} & h_{1,m} \\ h_{2,1} & h_{2,2} & \dots & h_{2,m-1} & h_{2,m} \\ 0 & h_{3,2} & \ddots & \vdots & \vdots \\ \vdots & 0 & \ddots & h_{m-1,m-1} & h_{m-1,m} \\ \vdots & \vdots & \ddots & h_{m,m-1} & h_{m,m} \\ 0 & 0 & \dots & 0 & h_{m+1,m} \end{bmatrix}^{(m+1) \times m}. \quad (\text{A.94})$$

It is used further below to formulate and solve the minimization problem for the residual (Eq. (A.93)).

A.12.3 Minimization of the residual

The correction of the start solution \vec{x}_0 is defined as [27]

$$\vec{z} = \sum_{j=1}^m \gamma_j \vec{v}_j, \quad (\text{A.95})$$

where γ_j are the components of the vector

$$\vec{\gamma} = [\gamma_1, \gamma_2, \dots, \gamma_m]^T. \quad (\text{A.96})$$

Furthermore, it can be shown that

$$\bar{A}\bar{V}_m = \bar{V}_{m+1}\bar{H}_m^* \quad (\text{A.97})$$

with

$$\bar{V}_m = [\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m] \quad (\text{A.98})$$

being a matrix with \vec{v}_j as columns. Let us introduce the notation

$$\vec{e} = [\|\vec{r}_0\|, 0, \dots, 0]^T, \quad (\text{A.99})$$

where \vec{e} has $(m+1)$ elements. Using the definition of Eq. (A.99), we observe that $\vec{r}_0 = \vec{b} - \bar{A}\vec{x}_0 = \bar{V}_{m+1}\vec{e}$. Hence, we obtain for the residual Eq. (A.93)

$$\begin{aligned}
\| \vec{b} - \bar{A}(\vec{x}_0 + \vec{z}) \| &= \| \vec{r}_0 - \bar{A} \left(\sum_{j=1}^m y_j \vec{v}_j \right) \| \\
&= \| \vec{r}_0 - \bar{A} \bar{V}_m \vec{y} \| \\
&= \| \bar{V}_{m+1} (\vec{e} - \bar{H}_m^* \vec{y}) \| \\
&= \| \vec{e} - \bar{H}_m^* \vec{y} \| .
\end{aligned} \tag{A.100}$$

We employed the orthonormality of \bar{V}_{m+1} in the last step (this means $\vec{v}_i^T \cdot \vec{v}_j = 0$ for $i \neq j$ and $\vec{v}_i^T \cdot \vec{v}_i = 1$ for $i = j$). Therefore, the problem of the minimization of the residual can be simplified as

$$\min_{\vec{z} \in \mathcal{K}_m} \| \vec{b} - \bar{A}(\vec{x}_0 + \vec{z}) \| = \min_{\vec{y} \in \mathcal{R}^m} \| \vec{e} - \bar{H}_m^* \vec{y} \| . \tag{A.101}$$

The solution of the minimization problem can be obtained with the aid of the Q-R algorithm which is described next.

A.12.4 Q-R algorithm

Let us define $\bar{R}_m = \bar{Q}_m \bar{H}_m^*$ with

$$\bar{Q}_m \stackrel{\text{def}}{=} \bar{F}_m \bar{F}_{m-1} \dots \bar{F}_1 \tag{A.102}$$

being the product of the Givens rotation matrices

$$\bar{F}_j = \begin{bmatrix} \bar{I}_{j-1} & & \\ & \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} & \\ & & \bar{I}_{m-j} \end{bmatrix}^{(m+1) \times (m+1)} . \tag{A.103}$$

In Eq. (A.103), \bar{I}_j denotes the identity matrix of dimension j . Further, c_j and s_j ($c_j^2 + s_j^2 = 1$) represent the sine/cosine of the rotation angle. The rotations are chosen such that \bar{H}_m^* is transformed into an upper triangular matrix \bar{R}_m which has the dimensions $(m+1) \times m$ and which last line contains only zeros. Since $\bar{Q}_m^T \bar{Q}_m = \bar{I}$, we can write in Eq. (A.101)

$$\begin{aligned}
\| \vec{e} - \bar{H}_m^* \vec{y} \| &= \| \bar{Q}_m^T (\bar{Q}_m \vec{e} - \bar{Q}_m \bar{H}_m^* \vec{y}) \| \\
&= \| \vec{g} - \bar{R}_m \vec{y} \| ,
\end{aligned} \tag{A.104}$$

where $\vec{g} = \bar{Q}_m \vec{e}$ denotes the transformation of the vector \vec{e} (Eq. (A.99)). The last line of \bar{R}_m consists of zeros, therefore only the term g_{m+1} is nonzero in the row $(m+1)$ of the vector $(\vec{g} - \bar{R}_m \vec{y})$. If we denote the first m -components of $(\vec{g} - \bar{R}_m \vec{y})$ as p_j ($j = 1, \dots, m$), then the norm in Eq. (A.104) becomes

$$\|\vec{g} - \bar{R}_m \vec{\gamma}\| = \sqrt{s_{m+1}^2 + \sum_{j=1}^m p_j^2}. \quad (\text{A.105})$$

If we choose the components γ_j of $\vec{\gamma}$ in such a way that $p_j^2 = 0$ for all $j = 1, \dots, m$, we obtain for the minimization problem in Eq. (A.101)

$$\min_{\vec{\gamma} \in \mathcal{R}^m} \|\vec{g} - \bar{R}_m \vec{\gamma}\| = |g_{m+1}|. \quad (\text{A.106})$$

The components γ_j results from the solution of the following system of linear equations

$$\begin{bmatrix} R_{1,1} & \dots & R_{1,m-1} & R_{1,m} \\ 0 & \ddots & \vdots & \vdots \\ \vdots & \ddots & R_{m-1,m-1} & R_{m-1,m} \\ 0 & \dots & 0 & R_{m,m} \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_{m-1} \\ \gamma_m \end{bmatrix} = \begin{bmatrix} g_1 \\ \vdots \\ g_{m-1} \\ g_m \end{bmatrix} \quad (\text{A.107})$$

by back-substitution. The solution of the system of equations (A.90) is then obtained with the known γ_j from Eq. (A.95).

It is important to remark that

$$\begin{aligned} \|\vec{r}_m\| &= \|\vec{b} - \bar{A}(\vec{x}_0 + \vec{z})\| \\ &= \|\vec{e} - \bar{H}_m^* \vec{\gamma}\| \\ &= \|\vec{g} - \bar{R}_m \vec{\gamma}\| \\ &= |g_{m+1}|. \end{aligned} \quad (\text{A.108})$$

This means that the actual residual can be easily determined as $|g_{m+1}|$.

A.13 TENSOR NOTATION

Expressions like coordinate (x_i) or velocity components (v_i) represent *first-order tensors*. They have three components and thus correspond to vectors. Hence,

$$\begin{aligned} x_i &= [x_1, x_2, x_3] = [x, y, z] = \vec{r}, \\ v_i &= [v_1, v_2, v_3] = [u, v, w] = \vec{v}. \end{aligned} \quad (\text{A.109})$$

Second-order tensors consist of nine components and can be written as 3×3 matrices, e.g.,

$$v_i v_j \equiv \begin{bmatrix} v_1 v_1 & v_1 v_2 & v_1 v_3 \\ v_2 v_1 & v_2 v_2 & v_2 v_3 \\ v_3 v_1 & v_3 v_2 & v_3 v_3 \end{bmatrix}. \quad (\text{A.110})$$

Similarly, the tensor of viscous stresses $\bar{\tau} = \tau_{ij}$ reads

$$\tau_{ij} \equiv \begin{bmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{bmatrix}. \quad (\text{A.111})$$

The Kronecker symbol δ_{ij} is a special second-order tensor. It corresponds to a 3×3 identity matrix. Thus, the relation holds

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \quad (\text{A.112})$$

The last important rule is the so-called *Einstein summation convention*. It states that whenever two identical indexes occur in an expression, it means a sum over all three coordinate directions. With this, the scalar product between the vectors \vec{u} and \vec{v} can be expressed as

$$u_i v_i = u_1 v_1 + u_2 v_2 + u_3 v_3 = \vec{u} \cdot \vec{v}. \quad (\text{A.113})$$

Furthermore, the divergence of the vector \vec{v} becomes in tensor notation

$$\frac{\partial v_i}{\partial x_i} = \frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2} + \frac{\partial v_3}{\partial x_3} = \vec{\nabla} \cdot \vec{v}. \quad (\text{A.114})$$

REFERENCES

- [1] Sommerfeld AJW. Partial differential equations in physics. New York: Academic Press; 1949.
- [2] Courant R, Hilbert D. Methods of mathematical physics. New York: Interscience; 1962.
- [3] Hirsch C. Numerical computation of internal and external flows, vol. 1. Chichester: John Wiley and Sons; 1988.
- [4] Hirt CW, Amsden AA, Cook JL. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J Comput Phys* 1974;14:227–53.
- [5] Pracht WE. Calculating three-dimensional fluid flows at all speeds with an Eulerian-Lagrangian computing mesh. *J Comput Phys* 1975;17:132–59.
- [6] Donea J. An arbitrary Lagrangian-Eulerian finite element method for transient fluid-structure interactions. *Comput Methods Appl Mech Eng* 1982;33:689–23.
- [7] Batina JT. Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Paper* 89-0115; 1989.
- [8] Thomas PD, Lombard CK. Geometric conservation law and its application to flow computations on moving grids. *AIAA J* 1979;17:1030–37.
- [9] Tamura Y, Fujii K. Conservation law for moving and transformed grids. *AIAA Paper* 93-3365; 1993.
- [10] Lesoinne M, Farhat C. Geometric conservation laws for flow problems with moving boundaries and deformable meshes and their impact on aeroelastic computations. *AIAA Paper* 95-1709; 1995 [also in *Comput Methods Appl Mech Eng* 1996;134:71–90].
- [11] Guillard H, Farhat C. On the significance of the GCL for flow computations on moving meshes. *AIAA Paper* 99-0793; 1999.

- [12] Geuzaine P, Farhat C. Design and time-accuracy analysis of ALE schemes for inviscid and viscous flow computations on moving meshes. *AIAA Paper 2003-3694*; 2003.
- [13] Nkonga B, Guillard H. Godunov type method on non-structured meshes for three-dimensional moving boundary problems. *Comput Methods Appl Mech Eng* 1994;113:183–204.
- [14] Singh KP, Newman JC, Baysal O. Dynamic unstructured method for flows past multiple objects in relative motion. *AIAA J* 1995;33:641–49.
- [15] Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. *J Comput Phys* 1996;127:380–97.
- [16] Slater JW, Freund D, Sajben M. Study of CFD methods applied to rapidly deforming boundaries. *AIAA Paper 97-2041*; 1997.
- [17] Vierendeels J, Riemsdagh K, Dick E. Flow calculation in complex shaped moving domains. *AIAA Paper 97-2044*; 1997.
- [18] Koobus B., Farhat C. Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes. *Comput Methods Appl Mech Eng* 1999;170:103–30.
- [19] Dubuc L, Cantarini F, Woodgate M, Gribben B, Badcock KJ, Richards BE. A grid deformation technique for unsteady flow computations. *Int J Numer Methods Fluids* 2000;32:285–11.
- [20] Sørensen KA, Hassan O, Morgan K, Weatherill NP. A multigrid accelerated time-accurate inviscid compressible fluid flow solution algorithm employing mesh movement and local remeshing. *Int J Numer Methods Fluids* 2003;43:517–36.
- [21] Steger JL. Implicit finite difference simulation of flows about arbitrary geometries. *AIAA J* 1978;17:679–86.
- [22] Pulliam TH, Steger JL. Implicit finite difference simulations of 3-D compressible flows. *AIAA J* 1980;18:159–67.
- [23] Patankar SV, Spalding DB. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *Int J Heat Mass Transfer* 1972;15:1787–806.
- [24] Pulliam TH, Steger JL. Recent improvements in efficiency, accuracy, and convergence for implicit approximate factorization algorithms. *AIAA Paper 85-0360*; 1985.
- [25] Warming RF, Beam R, Hyett BJ. Diagonalization and simultaneous symmetrization of the gas-dynamic matrices. *Math Comput* 1975;29:1037.
- [26] Whitham GB. Linear and nonlinear waves. New York: Wiley; 1974.
- [27] Saad Y, Schulz MH. GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J Sci Stat Comput* 1986;7:856–69.

INDEX

A

Additive Runge-Kutta, 200
Additive Schwarz Method, 326
ADI: Alternating Direction Implicit, 52, 186, 372
Advancing-front Delaunay, 373
Advancing-front method, 373, 378
Advancing-layers method, 381
Advancing-normal point placement, 383
Agglomeration multigrid, 304
ALE: Arbitrary Lagrangian Eulerian, 421
Algebraic grid generation, 361, 365
Algebraic models, 57
All-speed flow, 308
AMG: Algebraic Multigrid, 293, 303
Amplification factor, 341
Approximate Riemann solver, 43, 103
Arrhenius formulae, 22
ARS: Algebraic Reynolds Stress, 56
Artificial compressibility method, 31, 308
Artificial dissipation, 42, 92, 140
Asynchronous Agents Library, 324
AUSM: Advection Upstream Splitting Method, 43, 95, 97
Automatic differentiation, 184
Average of fluxes, 81, 86, 132, 135
Average of variables, 81, 86, 132, 135
Axisymmetric form, 427

B

Background grid, 374
Backscatter, 240
Backward Euler, 202, 424
Baldwin-Lomax model, 57
Bi-CGSTAB: Bi-Conjugate Gradient Stabilized, 53, 194
Body-fitted grid, 30
Boundary conditions, 58
Boundary value problem, 417
Boussinesq hypothesis, 57, 221
Bradshaw's assumption, 233
Bubble packing method, 373, 381
Bulk viscosity, 16

C

C++ AMP: Accelerated Massive Parallelism, 324
Carbuncle phenomenon, 44, 105
Cartesian grids, 31, 385

Cell-centered scheme, 37, 76, 80, 125, 132
Cell-vertex scheme, 37, 76, 82, 85, 125, 134
Central scheme, 42, 92, 140
Centrifugal force, 420
CFL condition, 173, 344, 353, 354
CFL number, 170, 173, 344, 353
CGS: Conjugate Gradient Squared, 53, 194
Characteristic boundary conditions, 58
Characteristic time-stepping, 49
Characteristic variables, 43, 104, 107, 112, 262, 433
Characteristics of PDE's, 416
Chimera technique, 34, 276
Circulation, 265
CIRS: Central Implicit Residual Smoothing, 287
Co-located grid scheme, 76, 125
Coarse grid correction, 295
Computational molecule, 90
Computational space, 33, 360
Condition number, 307
Conjugate gradient method, 53, 193
Conservation laws, 7
Conservative variable preconditioning matrix, 310
Conservative variables, 17, 76
Consistency, 338
Constrained Delaunay triangulation, 377
Containment-dual control volume, 138
Continuum, 7
Contravariant velocity, 17, 422
Control functions, 368
Control volume, 77, 78, 126, 128
Convective flux Jacobian, 174, 179, 183, 185, 187, 189, 190, 415, 429
Convective flux tensor, 9
Convective fluxes, 8, 16, 89, 139
Coordinate cut, 272, 362, 365
Coriolis force, 420
CUDA: Compute Unified Device Architecture, 327
Curvilinear grid, 33
CUSP: Convective Upwind Split Pressure, 43, 95, 100
Cyclic directions, 277

D

Dalton's law, 22
Damköhler number, 20

Damping functions, 228, 230
 Damping properties, 340
 Decoupling, 92, 115, 162
 Degrees of freedom, 38
 Delaunay method, 373, 374
 Density (mass) weighted decomposition, 55, 218
 Density-based schemes, 31
 Dependent variables, 76
 DES: Detached Eddy Simulation, 243
 Determinant of Jacobian of coordinate transformation, 412
 Differential form, 409
 Diffusive flux, 9
 Diffusive flux tensor, 9
 Dilatation, 15
 Direct methods, 52, 178
 Dirichlet condition, 369, 418
 Dirichlet tessellation, 374
 DIRK: Diagonally Implicit Runge-Kutta, 198
 Discontinuous Galerkin scheme, 38
 Discretization accuracy, 339
 Discretization error, 337
 DNS: Direct Numerical Simulation, 55, 214
 Domain decomposition, 34
 Dual control volumes, 37, 76, 85, 134
 Dual time-stepping approach, 51, 202
 Dummy cells, 254
 Dummy points, 254
 DXF: Drawing eXchange Format (AutoCAD), 360
 Dynamic SGS models, 241
 Dynamic viscosity coefficient, 15

E

E-CUSP scheme, 101
 Eddy viscosity, 57, 214, 221
 Edge collapsing, 304
 Eigenvalue, 101, 104, 105, 317, 434, 436
 Eigenvector, 104, 106, 318, 434, 435
 Einstein summation convention, 441
 Elliptic equation, 418
 Elliptic grid generation, 361, 367
 Embedded scheme, 200
 ENO: Essentially Non-Oscillatory, 45, 149
 Ensemble averaging, 218
 Enthalpy, 19
 Enthalpy damping, 284
 Entropy condition, 38
 Entropy correction, 44, 105, 107
 Equation of state, 19
 Ergodic hypothesis, 218

Euler equations, 25
 Explicit operator, 177
 Explicit scheme, 168
 Explicit time-stepping scheme, 48
 External body forces, 11, 411
 External volume forces, 11

F

Face vector, 77–79, 127–129
 Factored scheme, 178
 Factorisation error, 52, 187
 Far-field, 262
 FAS: Full Approximation Storage, 50, 295
 Favre (mass) averaging, 55, 218
 Favre decomposition, 55, 218
 Fick's law, 9, 13
 Finite control volume, 8
 Finite-difference method, 32, 36
 Finite-element method, 32, 38
 Finite-volume method, 32, 37
 First-order closures, 56, 214, 225
 Fluctuation-splitting scheme, 43, 44
 Fluid dynamics, 7
 Flux, 8
 Flux Jacobian, 33, 42, 43, 49, 51, 177, 178, 182
 Flux-difference splitting scheme, 43, 103
 Flux-vector splitting scheme, 43, 95
 FMG: Full Multigrid, 50, 297
 Forcing function, 295
 Forward Euler, 168, 423
 Fourier symbol, 341
 Fourier's law, 13
 Frontal Delaunay, 373

G

Gauss quadrature points, 149
 Gauss's theorem, 409
 Gaussian filter, 237
 GCL: Geometric Conservation Law, 18, 203, 423
 Germano identity, 242
 Global coarsening, 306
 Global time stepping, 174
 GMRES: Generalised Minimal Residual, 53, 193, 194, 437
 Green-Gauss gradient computation, 150, 278
 Grid cells, 29
 Grid converged solution, 41, 340
 Grid generation, 29
 Grid lines, 29
 Grid nodes, 29
 Grid points, 29

Grid refinement, 304
 Grid smoothing, 388
 Grid velocity, 422
 Grid vertexes, 29
 Gridless Method, 41

H

H-CUSP scheme, 101
 Hanging nodes, 33
 High Reynolds number model, 232
 HLL: Harten–Lax–van Leer (solver), 43
 Horse-shoe vortex, 265, 266
 Hybrid grids, 33, 122, 384
 Hyperbolic equation, 416
 Hyperbolic grid generation, 361, 370

I

IGES: Initial Graphics Exchange Specification, 360
 ILU: Incomplete Lower Upper, 54, 196
 Implicit operator, 51, 177, 178, 293
 Implicit scheme, 176
 Implicit time-stepping scheme, 48, 51
 Implicit-explicit residual smoothing, 49
 Incompressible, 31, 308
 Incremental point insertion, 374
 Inflow, 10
 Initial value problem, 417
 Injection boundary, 270
 Inlet boundary, 268
 Integral formulation, 9
 Internal energy, 12
 Interpolation operator, 294
 IRK: Implicit Runge–Kutta, 173, 197
 IRS: Implicit Residual Smoothing, 49, 286
 Iterative methods, 52, 178

J

Jacobi preconditioning, 49
 Jacobian coordinate transformation, 36
 Jacobian matrix, 429, 431
 JST scheme, 93

K

K- ω model, 58
 K- ε model, 58, 228
 Kinematic eddy viscosity, 217
 Kinematic viscosity coefficient, 15
 Kronecker symbol, 441
 Krylov subspace, 53, 193, 194

L

Laplace equation, 368, 370
 Laplacian operator, 140, 388

Lattice Boltzmann Method, 40
 LDFSS: Low-Diffusion Flux-Splitting Scheme, 43, 95

Least-squares gradient computation, 151
 Left state, 43, 81, 87, 90, 133

Left/right preconditioning, 53, 195

LES: Large-Eddy Simulation, 55, 235

Lifting body, 264

Limiter, 46, 91, 107, 155

Limiter for CUSP scheme, 112

Limiter for TVD scheme, 112

Limiter function, 46, 91, 107, 110, 155

Line-implicit methods, 52

Linear reconstruction, 146

Linear TFI, 367

Linelets, 52

Load balancing, 325

Local time-stepping, 49, 174

Low Reynolds number model, 229

LU-SGS: Lower–Upper Symmetric Gauss–Seidel, 52, 188

LU-SSOR: Lower–Upper Symmetric Successive Overrelaxation, 52, 188

Lumped mass matrix, 47, 168, 177

M

MAPS: Mach Number-Based Advection Pressure Splitting, 43, 95

Mass matrix, 47, 138, 168, 177, 202, 204

Matrix dissipation scheme, 42, 94, 143

Matrix-free approach, 54, 195

Max-min triangulation, 373

Median-dual control volume, 125, 134

Mesh, 29

Method of lines, 31, 46, 73, 122, 167

Method of weighted residuals, 39

Metric terms, 412

Metrics, 77, 126

MILES: Monotonically Integrated LES, 240

Mixed grids, 33, 122, 373, 384

Monotonicity preserving scheme, 44, 107, 155

Morkovin’s hypothesis, 55, 220

Moving grid, 421

MPI: Message-Passing Interface, 324

Multiblock approach, 33, 275

Multigrid cycle, 50

Multigrid level, 49

Multigrid method, 49, 293

Multiphase flow, 23

Multistage scheme, 169, 197

Multistage time-stepping scheme, 48

MUSCL: Monotone Upstream-Centered Schemes for Conservation Laws, 90, 109, 145

N

Navier-Stokes equations, 18
 Neumann condition, 369, 418
 Newton-Krylov method, 54, 178, 193, 195
 Newtonian fluid, 15
 No-slip boundary condition, 59
 Non-linear eddy viscosity, 57, 223
 Nonnested grids, 303
 Normal-momentum relation, 257
 Noslip boundary condition, 260
 NURBS: Nonuniform Rational B-Splines, 382

O

OpenACC, 324
 OpenCL: Open Computing Language, 329
 OpenMP: Open Multi-Processing, 326
 Order of accuracy, 36, 339
 Outflow, 10
 Outlet boundary, 268
 Overlapping control volumes, 37, 76, 82, 125
 Overrelaxation parameter, 190, 192, 193

P

Parabolic equation, 417
 Parallelization, 320
 PDE: partial differential equation, 361
 Perfect gas, 19
 Periodic boundary, 273, 362
 Phase angle, 341
 Physical space, 29
 Piecewise linear prolongation, 307
 PNS: Parabolized Navier-Stokes, 24, 417, 427
 Point implicit, 172
 Prandtl number, 19, 57, 222, 412
 Preconditioning, 31, 308
 Preconditioning matrix, 309, 314
 Preconditioning parameter, 315, 316
 Pressure sensor, 94
 Pressure-based schemes, 31, 308
 Prolongation operator, 296
 Pseudo-Laplacian, 140

Q

Quadratic reconstruction, 148
 Quasilinear form, 308, 415

R

RANS: Reynolds-Averaged Navier-Stokes, 55, 219

RCM: Reverse Cuthill-McKee, 182, 196

Real gas, 19, 46
 Reconstruction, 45, 125, 132, 135, 140, 144, 149
 Reflected cells, 271
 Residual, 47, 75, 124, 168
 Residual averaging, 49
 Residual distribution, 84
 Residual smoothing, 49, 286
 Restriction operator, 295
 Reynolds averaging, 55, 217
 Reynolds-stress tensor, 56, 219, 220
 Riemann solver, 43
 Right state, 43, 81, 87, 90, 133
 Roe average, 102, 103
 Roe matrix, 101, 103
 Roe upwind scheme, 43, 103, 318, 320
 Rotating frame of reference, 18, 419
 Rotation-rate tensor, 216
 Rotational periodicity, 273, 274
 Rothalpy, 421
 Rotor-stator interaction, 385
 RST: Reynolds-Stress Transport, 56, 224
 Runge-Kutta Time-Stepping Scheme, 48, 169

S

Scalar dissipation scheme, 93, 311
 SDIRK: Singly Diagonally Implicit Runge-Kutta, 198
 Search directions, 53, 194
 Second viscosity coefficient, 15
 Second-order closures, 56, 214
 Seed points, 304
 Semi-coarsening, 294
 SGS: Subgrid-Scale stress, 238
 Shape functions, 38
 Sharp Fourier cut-off filter, 236
 Single-stage time-stepping scheme, 48
 Singletons, 306
 SLIP: Symmetric Limited Positive, 92
 Slope limiter, 109, 110
 Smagorinsky SGS model, 241
 Smooth grid, 30, 123
 Solid wall, 255
 Solution reconstruction, 132, 135, 140, 144, 149
 Solution update, 168
 Source term, 74, 124, 172, 177
 Spalart-Allmaras model, 57, 225
 Spatial averaging, 217
 Spatial discretization, 32, 74, 124
 Spectral radius, 93, 143, 174–176, 318, 422
 Spectral-Element Method, 39

SST turbulence model, 232
 Stability, 337
 Stability analysis, 340
 Staggered grid scheme, 76, 125
 Steger-Warming flux-vector splitting, 183
 Steiner point, 378
 Stencil, 90
 STEP: STandard for the Exchange of Product model data, 360
 Stiffness, 172, 177
 STL: STereoLithography (rapid prototyping), 360
 Stokes's hypothesis, 412
 Strain-rate tensor, 216
 Structured grids, 32, 360
 Structured scheme, 73
 Subgrid-scale model, 55, 235, 240
 Surface forces, 11
 Surface grid, 360
 Surface grid generation, 383
 Sutherland formula, 19
 Switched evolution relaxation, 196
 Symmetric TVD scheme, 44, 106
 Symmetry boundary, 271
 System matrix, 177, 293

T

Tensor notation, 216, 440
 TFI: Transfinite Interpolation, 361, 365
 TFQMR: Transpose-Free Quasi-Minimum Residual, 53, 194
 Thermal conductivity coefficient, 13
 Thermal diffusivity coefficient, 9, 13
 Time averaging, 217
 Time step, 173
 Time-stepping operator, 341
 Tophat filter, 236
 Total energy, 12, 420
 Total enthalpy, 14, 421
 Translational periodicity, 273
 Truncation error, 36, 338
 TSL: Thin Shear Layer, 23, 116, 185, 425
 Turbulence modeling, 55
 Turbulent dissipation rate, 224
 Turbulent eddy viscosity, 57, 221
 Turbulent heat-flux vector, 56, 221, 222
 Turbulent kinetic energy, 219, 220
 Turbulent Prandtl number, 57, 222
 Turbulent thermal conductivity coefficient, 57, 222

TVD: Total Variation Diminishing, 43, 44, 106
 Two-equation models, 228

U
 UIRS: Upwind Implicit Residual Smoothing, 49, 290
 Unit normal vector, 8, 17, 75, 77–79, 126, 127, 129, 132, 136, 231, 421, 422
 Unsteady flows, 51, 202
 Unstructured grids, 33, 35, 373
 Unstructured scheme, 121
 Upwind prolongation, 298, 301, 303
 Upwind restriction, 298, 303
 Upwind scheme, 43, 76, 89, 90, 95, 103, 106, 144
 Upwind TVD scheme, 44, 106, 318

V

V-Cycle, 296
 Validation, 338
 Van Albada limiter, 108, 111
 Van Leer's flux-vector splitting, 96
 Variational principle, 39
 Vector of convective fluxes, 16
 Vector of viscous fluxes, 16
 Verification, 338
 Virtual edges, 154
 Viscous flux Jacobian, 174, 185, 187, 189, 191, 192, 431
 Viscous fluxes, 16, 113, 157
 Viscous stress tensor, 11, 216
 Viscous stresses, 14
 Visibility criterion, 382
 Volume agglomeration, 304
 Volume grid, 360
 Von Neumann stability analysis, 340
 Voronoi diagram, 374
 Vortex correction, 265, 266

W

W-Cycle, 296
 Wall functions, 232
 Weak formulation, 39
 Weak solutions, 38
 WENO: Weighted Essentially Non-Oscillatory, 45, 149