Adaptive High-Resolution Finite Element Schemes

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Matthias Möller

Adaptive High-Resolution Finite Element Schemes

Matthias Möller

Dissertation eingereicht am: 12.09.2008
Tag der mündlichen Prüfung: 05.12.2008

Mitglieder der Prüfungskommission

JProf. Dr. D. Kuzmin (1. Gutachter, Betreuer)
Prof. Dr. S. Turek (2. Gutachter)
Prof. Dr. R. Scharlau
Prof. Dr. H. Blum
Dr. Ch. Becker

To my parents
Berndt and Birgit
and my wife Corinna

Acknowledgments

It is impossible to acknowledge everyone who directly or indirectly contributed to the realization of this thesis. I apologize to all those not mentioned by name: You have been a great help.

In particular, I would like to express my sincere gratitude to the following people: Professors Dmitri Kuzmin and Stefan Turek guided my scientific career from the first contact to Numerical Mathematics to advanced topics of Computational Fluid Dynamics. I am deeply indebted to them for giving me the opportunity to work in a friendly atmosphere at the Institute of Applied Mathematics at the 'Technische Universität Dortmund', formerly known as 'University of Dortmund'. I want to thank them for supervising my scientific education during the last years; their ideas, fruitful discussions and encouraging criticisms helped me to improve my knowledge and, finally, led to the numerical methods and mesh adaptation algorithms presented in this thesis.

I am very grateful to Professor Rainald Löhner for supporting me in implementing his 'non-dimensional error indicator' and giving advise in adaptive mesh refinement algorithms. I am also indebted to Dr. Jaroslav Hron for sharing his experience in discrete Newton methods with me.

My appreciation goes to my colleagues of the 'Lehrstuhl III' who helped me in pointing out the obvious (and not-so-obvious) problems in this research effort. On this occasion, I would like to thank Dominik Göddeke for proofreading my manuscripts and giving valuable feedback. I am also indebted to Michael Köster for modernizing our in-house FEM package FEATFLOW so as to catch up with today's programming standards. I gave in to the temptation of reimplementing the flow solver and the adaptation library which lead to superior mesh refinement and re-coarsening algorithms. I also want to thank Jens F. Acker for assistance with mesh generation and Dr. Christian Becker and Sven H.M. Buijssen for solving all sorts of problems in daily IT-business.

I am very grateful to my parents for encouragement throughout my time at Dortmund University. Most of all, I want to express my deepest gratitude to my wife Corinna for all her support during my graduate study and for helping me keep things in perspective.

Support of this work by the DFG (German Research Foundation) is gratefully acknowledged.

Dortmund, 12. September 2008

Matthias Möller

General notation

Throughout this thesis, an attempt has been made to keep the notation as consistent as possible by adopting specific letter styles for each group of mathematical entities. The nomenclature commonly used in the literature on Computational Fluid Dynamics has been employed. The general scheme of notation is described below.

CHARACTERS. GENERAL USAGE

- *Calligraphic upper case* $\mathcal{E}$, $\mathcal{T}$, $\mathcal{V}$, $\mathcal{W}$, ... : Function spaces and sets (of elements or vertices).
- *Greek letters* Γ, Ω, Φ, γ, λ, ... : Surfaces, scalars and scalar-valued functions.
- *Italic bold-face lower case* $\mathbf{v}$, $\mathbf{x}$, ... : Multidimensional vector. *Italic bold-face lower case with subscript* $\mathbf{v}_i$, $\mathbf{x}_i$, ... : Multidimensional vector at node i. *Italic light-face lower case with subscript* v_d, x_d, ... : Spatial component of the corresponding vector.
- *Italic bold-face upper case* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{F}$, ... : Multiple (pair, triple) of scalar matrices or functions, e.g., $\mathbf{A} = (A^1, A^2, A^3)$. *Italic light-face upper case with superscript* A^d, B^d, C^d, F^d, ... : Spatial component of the corresponding matrix or function.
- *Italic bold-face lower case with subscript* $\mathbf{a}_{ij}$, $\mathbf{b}_{ij}$, $\mathbf{c}_{ij}$, $\mathbf{s}_{ij}$, ... : Multi-component coefficient vector for the edge ij, e.g., $\mathbf{a}_{ij} = (a^1_{ij}, a^2_{ij}, a^3_{ij})$. *Italic light-face lower case with sub-/superscript* a^d_{ij}, b^d_{ij}, c^d_{ij}, s^d_{ij}, ... : Spatial component of the corresponding vector.
- *Italic light-face lower case* f, g, u, v, w, ... : Scalar-valued functions. *Italic light-face upper case* $A, B, U, W, V, \ldots$: Scalar matrices or multi-component functions, e.g., $A = \{a_{ij}\}$ and $U = [U_1, U_2, U_3]^T$. *Italic light-face lower case with subscript* a_{ij}, b_{ij}, ... : Coefficient of the corresponding matrix. *Italic light-face upper case with subscript* U_k, ... : Component of the corresponding vector.
- *Sans-serif lower case* u, v, ... : Approximation of a scalar vector. *Sans-serif lower case with subscript* u_i, u_j, ... : Nodal component of the corresponding vector.
- *Sans-serif upper case* F, R, U, ... : Approximation of a multi-component vector. *Sans-serif upper case with subscript* U_k, ... : Component of the corresponding vector, e.g., k^{th} variable, or U_i, U_j ... : Nodal component of the corresponding vector (for all variables).
- *Sans-serif bold-face upper case* $\mathbf{F}$, ... : Approximation of a multiple (pair, triple) of functions, e.g., $\mathbf{F} = (\mathrm{F}^1, \mathrm{F}^2, \mathrm{F}^3)$. *Sans-serif bold-face upper case with subscript* $\mathbf{F}_i, \mathbf{F}_j \ldots$: Nodal component of the corresponding vector. *Sans-serif upper case with superscript* F^d, ... : Spatial component of the corresponding vector.

NOTATION FOR DIFFERENTIAL EQUATIONS AND FINITE ELEMENTS

Ω	open domain in $\mathbb{R}^d$
Γ	boundary $\partial\Omega$
Γ_{D}	part of the boundary, where Dirichlet boundary conditions are prescribed
Γ_{N}	part of the boundary, where Neumann boundary conditions are prescribed
Γ_{in}	inflow part of the boundary
Γ_{out}	outflow part of the boundary
θ	implicitness parameter
σ	perturbation parameter for Newton's method
∇f	gradient operator, i.e. $[\partial f/\partial x_1, \partial f/\partial x_2, \partial f/\partial x_3]$
$\nabla \cdot \mathbf{f}$	divergence operator, i.e. $\sum_{d=1}^{3} \partial \mathbf{f}_d/\partial x_d$
Δf	Laplace operator, i.e. $\sum_{d=1}^{3} \partial^2 f/\partial x_d^2$
$\mathbf{v} \otimes \mathbf{w}$	tensor product, i.e. $\mathbf{v}\mathbf{w}^T = [v_i w_j]_{i,j=1}^3$
$\partial^\alpha/\partial x^\alpha$	partial derivative with respect to variable x
$\partial/\partial t$	local time derivative
$\mathrm{d}/\mathrm{d}t$	substantial time derivative
$\mathcal{E}_m, \mathcal{V}_m$	set of elements/vertices in triangulation $\mathcal{T}_m = (\mathcal{E}_m, \mathcal{V}_m)$
$\mathcal{H}^1(\Omega)$	Sobolev space of L_2 functions with square-integrable first derivatives
$N_{\mathrm{E}}, N_{\mathrm{V}}$	number of elements/vertices of the triangulation
$\mathcal{S}^t, \mathcal{S}_h^t$	finite element space and its discrete counterpart
$\mathcal{T}_m$	triangulation resulting from m refinement/re-coarsening steps
$\mathcal{W}, \mathcal{W}_h$	space of test functions and its discrete counterpart
$ij, \vec{ij}$	edge connecting nodes i and j and its oriented counterpart
t	time variable
u	scalar quantity
$\mathbf{x}$	spatial coordinates in Euclidean space
$\hat{\mathbf{x}}$	spatial coordinates in reference system
$\rho(A)$	spectral radius of matrix A
$\mathcal{O}(\cdot)$	Landau symbol
$\|\cdot\|_p, \|\cdot\|_\infty$	p-norm/maximum norm

NOTATION FOR EULER EQUATIONS AND GAS DYNAMICS

$\mathbf{A} = (A^1, A^2, A^3)$	inviscid Jacobian matrices
$\mathbf{F} = (F^1, F^2, F^3)$	inviscid fluxes
c	speed of sound
c_p, c_v	specific heats at constant pressure/volume
e, E	internal/total energy
$\gamma = c_p/c_v$	ratio of specific heats
h, H	specific/total enthalpy
$\underline{\underline{I}}$	identity tensor
$\Lambda = \text{diag}\{\lambda_1, \ldots, \lambda_5\}$	diagonal matrix of eigenvalues
λ_k	eigenvalue for wave field k
$\mathbf{l}_k$, $\mathbf{r}_k$	left and right eigenvectors for field k
M	Mach number
$\mathbf{n}$, $\vec{N}$	physical/numerical outward unit normal
N_{D}	number of spatial dimensions
N_{U}	number of state variables
N_{n}, N_{p}	number of numerical/physical boundary conditions
p	pressure
R	specific gas constant
ρ	density
s, S	specific/total entropy
T	total temperature
$\boldsymbol{\tau}$, $\vec{T}$	physical/numerical tangential vector
$U = [U_1, \ldots, U_5]^T$	vector of conservative variables
$\mathbf{v} = [v_1, v_2, v_3]^T$	velocity vector
$V = [V_1, \ldots, V_5]^T$	vector of primitive variables
$W = [W_1, \ldots, W_5]^T$	vector of characteristic variables

SUBSCRIPTS

h	discretized quantity, e.g., approximate solution u_h
i, j	nodal point index, e.g., nodal coefficient u_i
k	number of wave field, e.g., eigenvalue λ_k
k, l	component of a set, e.g., scalar sub-matrix L_{kl} from global operator L
max, min	maximum/minimum value, e.g., maximum eigenvalue $\lambda_{\max} = \max_k\{\lvert\lambda_k\rvert\}$
n	normal component, e.g., $v_n = \mathbf{v}\cdot\mathbf{n}$
τ	tangential component, e.g., $v_\tau = \mathbf{v}\cdot\boldsymbol{\tau}$
∞	free stream quantities, e.g., U_∞
0	initial condition, e.g., U_0

SUPERSCRIPTS

n	time level t^n, e.g., $u^n = u(t^n)$
(m)	iteration number, e.g., $u^{(m)}$
$\hat{u}$	recovered values, e.g., recovered gradient value $\hat{\nabla}\mathrm{u}$
$\bar{\mathrm{u}}$	mean value or constant solution value
$\tilde{\mathrm{u}}$	density averaged Roe value, e.g., $\tilde{\mathbf{A}} = (\tilde{A}^1, \tilde{A}^2, \tilde{A}^3)$
$*$	predicted boundary value, e.g., U^*
$**$	corrected boundary value, e.g., U^{**}

Contents

1

Introduction

The design process of reliable simulation tools for Computational Fluid Dynamics involves three basic steps: a mathematical model which describes the underlying physical principles, a numerical method which is used to compute discrete solutions, and some mechanism to validate the numerical approximation. This thesis is mainly concerned with the development of numerical methods for flow problems which are commonly modeled by (systems of) partial differential equations. It is therefore instructive to consider the linear advection equation in one space dimension [154]

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \quad \text{in} \quad \mathbb{R}\times\mathbb{R}_0^+, \qquad u(x,0) = u_0(x) \tag{1.0.1}$$

for a scalar quantity u as an introductory example. The initial profile u_0 is prescribed at time $t = 0$ and advected along the x-axis with constant speed a, whereby the flow direction depends on the sign of the velocity a. The exact solution at a fixed time t can be computed adopting the method of characteristics. Let the curve $x = x(t)$ in the x,t-plane satisfy the ordinary differential equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = a \tag{1.0.2}$$

so that the rate of change of the function $u = u(x(t),t)$ along $x = x(t)$ vanishes [154]:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{\partial u}{\partial t} + \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\partial u}{\partial x} = 0 \tag{1.0.3}$$

The above relation states that the solution u remains constant along characteristics. For each initial condition $x(0) = x_0$, there exists a straight line $x(t) = x_0 + at$ passing through the point $(x_0, 0)$ in the x,t-plane so that the exact solution at future times can be computed from [154]

$$u(x,t) = u_0(x_0) = u_0(x-at), \qquad t \in \mathbb{R}_0^+ \tag{1.0.4}$$

According to this formula, the shape of the initial profile cannot change, that is, a discontinuity remains discontinuous and a smooth profile preserves its smoothness as indicated in Figure 1.1. In other words, the solution at a single point $(x_0 + a\bar{t}, \bar{t})$ in the space time domain depends solely on the initial profile given at the location x_0. Such localized propagation of information is a typical feature of so-called hyperbolic partial differential equations. In particular, we are mainly interested in conservation laws of hyperbolic type which require special solution techniques.

In general, it is difficult and for many flow problems of practical interest even impossible to determine their solution exactly so that numerical methods are required to compute an approximation to it. The design of discretization techniques for conservation laws is a challenging task which represents a field of active research. To illustrate some drawbacks of common methods, consider a rectangular initial profile and solve the linear advection equation (1.0.1) numerically.

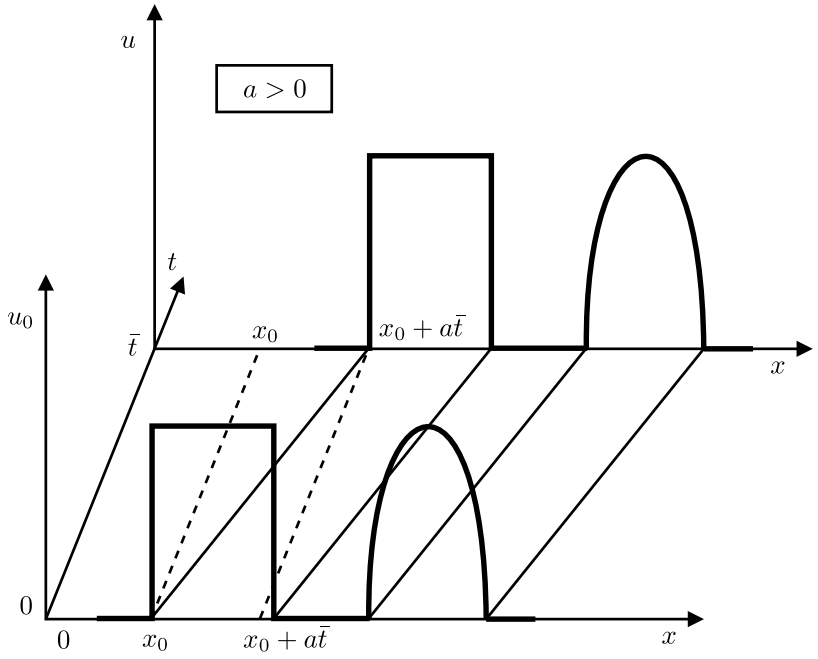


Fig. 1.1. Linear convection: initial profile and exact solution.

Standard high-order discretization techniques are prone to produce spurious undershoots and overshoots in the vicinity of steep gradients and discontinuities so that the approximate solution is polluted by non-physical oscillations as presented in Figure 1.2 (left). From the physical point of view, quantities such as density, temperature or concentration must preserve positivity. The occurrence of negative values which may be due to numerical undershoots leads to meaningless results. Moreover, the method may become unstable and the simulation may even fail for instance if the square root of a negative quantity is taken. On the other hand, low-order methods are free of these drawbacks but they engender excessive numerical diffusion so that the final solution is 'smeared' considerably as depicted in Figure 1.2 (right). Hence, low-order schemes are incapable of producing accurate solutions, and therefore, not recommended for the simulation of flow problems which are characterized by the presence of steep gradients and the formation of discontinuities.

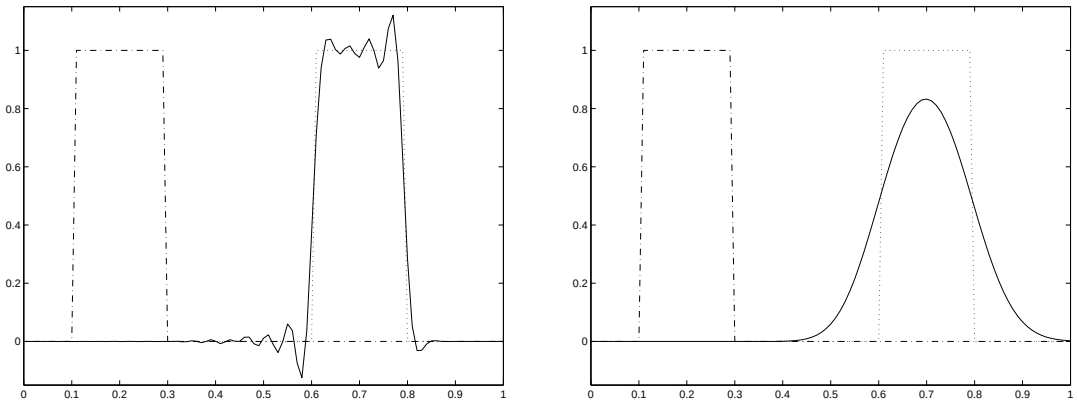


Fig. 1.2. Deficiencies of standard high-order methods (*left*) and low-order methods *(right)*.

Modern high-resolution schemes commonly in use rely on a suitable blending of both imperfect methods to recover the high accuracy of the original approximation in regions where the solution is sufficiently smooth and switch to the diffusive low-order scheme in the vicinity of discontinuities. Thus, there are three crucial ingredients to be considered: a high-order approximation for the accurate computation of smooth solutions, a non-oscillatory low-order scheme used near steep gradients, and a mechanism to automatically switch between both methods.

1.1. Research goals

In this work, the *Algebraic Flux Correction* (AFC) methodology is adopted which was developed by Kuzmin [133, 134] and Kuzmin and Turek [144, 145] for the design of high-resolution finite element schemes applicable to convection dominated partial differential equations. Starting from the standard Galerkin finite element method, the low-order discretization is constructed by performing row-sum mass lumping and elimination of negative off-diagonal entries from the discrete transport operator by adding a proper amount of artificial diffusion. This technique termed *discrete upwinding* [144] amounts to looping over pairs of off-diagonal matrix coefficients and leads to the definition of discrete diffusion operators which can be safely applied without violating mass conservation principles. The difference between the high- and low-order residuals is decomposed into sums of antidiffusive fluxes which need to be limited in order to prevent the formation of non-physical oscillations [144] which would be generated if unconstrained antidiffusion was applied. A node-based limiting strategy is employed [135, 137] which is inspired by Zalesak's multidimensional flux corrected transport algorithm [268]. All manipulations required to construct high-resolution schemes of this type are performed on the matrix level, and therefore, algebraic flux correction can be applied to discrete transport operators of any origin.

In this thesis, the focus is placed on *implicit* high-resolution finite element schemes based on the concept of algebraic flux correction which are applicable to unstructured meshes and complex domains. This is in contrast to many flow solvers commonly in use which are typically based on finite volume or discontinuous Galerkin finite element approximations making use of fully explicit time-stepping schemes. Explicit time integration techniques such as the forward Euler method may require some extra stabilization, and moreover, the size of the time step is subject to a restrictive CFL condition. In short, the largest admissible global time step is dictated by local CFL numbers which are likely to become very restrictive for locally refined meshes. In contrast, implicit time discretizations can be operated at larger time steps since stability conditions (if any) are less restrictive. However, the solution of an algebraic system of equations in each step is mandatory so that implicit schemes are frequently denounced as being non-competitive and their explicit counterparts are preferred. Furthermore, application of algebraic flux correction demands for the solution of *nonlinear* systems even for linear governing equations.

This can be accomplished by means of a fixed-point defect correction scheme [255], whereby the evolution operator for the underlying low-order discretization constitutes a viable preconditioner [144]. On the other hand, convergence rates observed for standard defect correction schemes tend to deteriorate significantly if the time step is increased and convergence may even fail completely. In this work, the algebraic flux correction methodology is equipped with *discrete Newton algorithms* [183, 184, 187] which lead to an efficient solving of nonlinear algebraic systems. Owing to the fact that the construction of the low-order scheme and the limitation of compensating antidiffusion are performed on the discrete level, there is no continuous counterpart which can be differentiated explicitly and used in standard Newton methods. As a remedy, divided differences are employed to construct an approximation to the Jacobian matrix which can be efficiently assembled edge-by-edge. The use of a node-based flux limiter may give rise to an extended sparsity pattern which is determined *a priori* by considering standard results from classical graph theory.

The design of adaptive finite element schemes for transient flow problems constitutes the primary goal of this work. The dynamic mesh adaptation algorithm developed by Hempel [102] for triangular grids in two dimensions is reviewed and the underlying concepts are generalized to arbitrary triangulations which may consist of different types of elements. The mesh adaptation procedure is based on the classical red-green strategy introduced by Bank et al. [19]. Each cell is subdivided regularly whenever the element error of the numerical solution exceeds some prescribed tolerance. Hanging nodes are eliminated by subdividing adjacent cells following so-called green refinement rules. These transition elements need to be reverted to their original macro cell prior to performing further refinement. Let us point out a key feature of the red-green adaptation strategy: For every 'atomic' refinement, e.g., one quadrilateral is refined into four quadrilaterals, there exists a unique re-coarsening operation which reverts the four cells into their macro element.

Adaptation procedures implemented in common simulation tools seem to be more concerned with refining elements and require multiple cycles to thin out the grid once the error has dropped below some tolerance. This may be attributed to the fact that automatic mesh generation techniques have gained maturity so that overly resolved regions can be regenerated without the need for user interaction. Such algorithms have been successfully used for very complex flow simulations. However, local remeshing and multistep coarsening procedures are incapable to provide a complete hierarchy of grids which is required, e.g., for multigrid algorithms [86]. In fact, each element subdivision can be reversed on paper by a unique re-coarsening operation, but the crucial part is to identify such patches efficiently and convert them to their macro cell. For triangles in two dimensions, Hempel [102] proposed to manage the grid genealogy by storing the 'date of birth' for each node. All required information such as the type of element (red/green triangle) and its relation to the unrefined macro cell is reconstructed from the nodal generation numbers so that no auxiliary tree-like structure is required to maintain parent-child relations. The re-coarsening algorithm is built on a recursive 'locking' [102] of vertices which must remain in the triangulation either due to accuracy reasons or to prevent the generation of unhandled hanging nodes.

We generalize the definition of the *nodal generation function* to arbitrary cells and give a complete characterization of elements in 2D based on the age of their corner vertices. The correctness of this approach is shown by induction so that the resulting triangulations are guaranteed to comply with all red-green rules. This approach can be extended to 3D, but the number of criteria to check for a particular cell grows considerably and is therefore prone to programming errors. To permit a fail-safe identification of elements in practical applications, we resort to *local two-level ordering* which is based on the work by Turek [254], who used global two-level ordering for an efficient implementation of geometric multigrid methods. In addition, we suggest a viable strategy to characterize elements in terms of integral states. In particular, the 'interplay' of nodal generation data is converted into bit representation, whereby the consistent application of a predefined local two-level ordering is mandatory to ensure uniqueness. Thus, the task to identify elements based on the nodal generation function reduces to simple distinction of cases for admissible states.

For the design of an efficient re-coarsening technique, we pursue the idea of excluding vertices from deletion step-by-step and devise a refined *node locking algorithm* applicable to general triangulations. It is easy to implement and shares the outstanding property of its predecessor [102], that is, the refinement and coarsening procedures feature the same 'throughput'. In other words, refinement operations can be reversed by applying the inverse re-coarsening operations so that each mesh of a transient flow computation provides an intrinsic hierarchy suitable for the application of multigrid schemes [86]. The presented mesh adaptation algorithm is developed in a most general framework, so that different types of elements can be easily combined in one triangulation. In particular, hybrid grids can be used which attempt to combine the advantages of both structured and unstructured meshes [261]. Since the algebraic flux correction paradigm [144] does not rely on geometric criteria it perfectly suits the flexibility of our dynamic mesh adaptation algorithm.

The use of mixed-type triangulations has to be accounted for in the derivation of error indicators. A common approach is to recover improved gradient values and compare them to the consistent slopes to obtain an estimate for the H_1-error of the numerical solution. The recovery of improved gradient values from the finite element solution features similarities to the accurate treatment of convective terms so that some basic ideas from the construction of high-resolution schemes for convection dominated problems can be adopted to reconstruct gradient values. Two different approaches for the evaluation of improved gradient values are presented which make use of slope limiting techniques [185, 186]. The gradient values at the edge midpoints can be computed as the limited average of consistent slopes adjacent to the corresponding edge. As an alternative, standard recovery techniques may be employed to predict gradient values either directly at edge midpoints or at the element vertices so that provisional slope values can be interpolated for each edge. A slope limiter is applied edge-by-edge to correct the intermediate values, whereby the consistent gradient values from adjacent cells serve as upper and lower bounds. The local element error can be computed from the improved slopes and used to steer the grid adaptation procedure.

In the last part of this work, high-resolution finite element schemes are used to simulate inviscid gas flows modeled by the compressible Euler equations. This first-order system consists of hyperbolic PDEs which describe the conservation of mass, momentum and energy. The design principles for scalar conservation laws have been generalized to hyperbolic systems of equations in [140, 142, 143] and a handy flux limiting procedure is proposed in a recent publication by Kuzmin [135]. These techniques which rely on the use of the group finite element formulation [78] are combined with our dynamic mesh adaptation algorithm which is extended to coupled systems. In particular, the error indicator is applied to a scalar key variable such as the density or the Mach number, whereby both quantities are sensitive to shocks, contact discontinuities and rarefaction waves. The predicted gradient error for the indicator variable is used to mark elements for refinement and identify groups of cells which can be safely re-coarsened. As an alternative, the dimensionless error indicator introduced by Löhner [161] is utilized in the simulation of more complex flows which exhibit multiple features of different intensities and length scales.

Another topic of utmost importance is the numerical treatment of boundary conditions for the compressible Euler equations. We resort to the predictor-corrector algorithm introduced by Kuzmin et al. [142] and describe a uniform approach to the implementation of characteristic and reflecting boundary conditions. In essence, a suitable combination of the local Riemann invariants evaluated for the prescribed free stream data and the nodal components of the computed solution vector is used at subsonic and supersonic inlets and outlets. Furthermore, a simplified 1D Riemann problem is solved locally for each boundary node so as to impose the no-penetration condition at solid walls. Each component of the overall solution strategy needs to be 'fine-tuned' so that time-dependent problems can be simulated at reasonable cost. We describe two different data structured for storing the global system matrix (or parts of it) in memory and utilize a segregated solution procedure [140, 142] to reduce the numerical complexity of transient flow computations.

1.2. Outline of the thesis

This work mainly deals with the design of numerical schemes for convection dominated flow problems. Besides the presentation of numerical methods, attention is also paid to practical aspects of their implementation. The main algorithms are presented in pseudo-code which should be readable by any programmer who may translate it into his/her particular programming language. Note that numerical methods for conservation laws are traditionally based on finite volume or discontinuous Galerkin finite element discretizations, whereas the use of the classical finite element approach is not as popular in the 'Conservation law community'. Therefore, a brief overview of finite element fundamentals is given in Appendix A. On the other hand, experts in finite elements

may be less familiar with the simulation of flow problems so that some classical results for hyperbolic conservation laws will be stated. Of course, this work can neither address all aspects of fluid flow computations nor can the finite element method be covered in full detail. To keep the presentation self-contained explicit formulas for Jacobian matrices as well as practical aspects of storing and traversing matrices efficiently are postponed to Appendices B and C.

Chapter 2 provides a brief introduction to mathematical aspects of Computational Fluid Dynamics. A viable classification of partial differential equations is presented and the most important properties of elliptic, parabolic and hyperbolic problems are discussed. In particular, the theory of nonlinear hyperbolic equations is briefly reviewed on the basis of the one-dimensional Burgers equation. The concept of weak solutions is introduced and the development of shocks and rarefaction waves is analyzed, whereby additional criteria are employed to satisfy the entropy condition.

Chapter 3 is devoted to the numerical treatment of scalar conservation laws. Some properties of standard discretization techniques are outlined and algebraic criteria for the design of high-resolution schemes are presented. A generic conservation law is discretized by the Galerkin finite element method, whereby the group formulation [78] is adopted to approximate the flux function. The construction of a non-oscillatory low-order scheme is addressed, and viable strategies to design node-based flux limiters of FCT and TVD type are described. As an alternative to the standard defect correction procedure, a discrete Newton method is developed for the efficient treatment of implicit algebraic flux correction schemes. The presented high-resolution schemes are applied to scalar benchmark problems for stationary and time-dependent flows in two dimensions.

The topic of error indication for convection dominated scalar problems is covered in *Chapter 4*. A survey of *a posteriori* error estimation techniques is presented and their applicability to algebraic flux correction schemes is investigated. Two recovery-based error indicators are suggested, whereby slope limiting is employed to correct provisional gradient values recovered at edge midpoints. As an alternative, the limited average of consistent finite element gradients provides a viable estimate of slope values evaluated at edge midpoints. A general framework for hierarchical mesh adaptation is proposed which builds on the design principles of the classical red-green strategy [19]. The earlier work by Hempel [102] is reviewed and a consistent extension to arbitrary triangulations is devised. In particular, we show that the elements of any mesh created by the red-green strategy can be characterized by the local values of the recursively defined nodal generation function so that no tree-based data structure is required to manage the grid genealogy. Efficient algorithms for dynamic mesh refinement and re-coarsening are devised and implementation details which may be helpful in practice are revealed. The new adaptation procedure is applied to scalar transport problems for stationary and transient flows in 2D.

Chapter 5 deals with hyperbolic systems of conservation laws, and in particular, with the compressible Euler equations. The most important properties of the governing equations are briefly discussed at the beginning of the chapter. The group finite element formulation [78] is adopted to approximate the inviscid fluxes and an edge-by-edge procedure for the efficient matrix assembly is described. The construction of the low-order scheme is based on a generalization of scalar concepts, whereby different choices for the design of artificial viscosities are discussed. Flux correction for hyperbolic systems is performed in terms of characteristic variables, whereby flux limiting of TVD type [135] is adopted. The implementation of boundary conditions for the two-dimensional Euler equations is reviewed in the framework of an implicit finite element formulation. A viable treatment of characteristic and solid wall boundary conditions is outlined which make use of the node-based predictor-corrector algorithm [140, 142]. Numerical examples are presented for transient and stationary benchmark problems in two space dimensions, whereby adaptive mesh refinement and re-coarsening is utilized to reduce the computational costs.

2

Mathematical aspects of scalar conservation laws

2.1. Introduction to partial differential equations

Engineers and scientific analysts face the need to predict the behavior of increasingly complex physical systems to a more and more detailed level. However, for realistic configurations which frequently occur in industrial applications it is impossible to predict the exact behavior even if the most sophisticated mathematical models are adopted. The challenging task of practitioners and researchers alike is to find the most appropriate model for a specific application. This requires a profound knowledge of all physical restrictions which have been introduced during the derivation of the model. As a consequence, these limitations inevitably lead to modeling errors which form the tip of a large hierarchy of error sources. In this chapter, we will provide a brief introduction into frequently adopted mathematical models and investigate the most relevant properties.

The analysis and numerical simulation of most continuous systems such as fluid dynamics, solid mechanics or heat conduction is based on partial differential equations (PDEs). In contrast to ordinary differential equations (ODEs) which involve only one independent variable and derivatives with respect to this variable, PDEs may have several independent variables and partial derivatives with respect to those variables. Hence, the theoretical analysis and the numerical treatment of partial differential equations is considered difficult and requires special tools and solution techniques. In the next paragraphs, we will consider different classes of partial differential equations and introduce some of their most important properties.

2.1.1. Classification of partial differential equations

The type of the differential equation characterizes the qualitative behavior of solutions and it impacts the choice of boundary conditions. Theoretical results concerning existence and uniqueness are frequently tailored to a single class of PDEs. In particular, there exist three types of partial differential equations, namely, *elliptic*, *parabolic* and *hyperbolic* ones. There are many ways to classify PDE problems based on rigorous mathematical analysis [6, 79, 104]. It is common practice to start from a linear first- or second-order PDE for a single independent variable and investigate the eigenvalues or the determinant of the constant coefficient matrix. The classification of PDEs can be extended to first-order systems by analyzing the existence of wave-like solutions. A plane wave traveling in the direction $\mathbf{n}$ with velocity $\lambda/|\mathbf{n}|$ can be expressed as follows [262]

$$U = \hat{U} e^{i w(t,\mathbf{x})}, \qquad \hat{U} = \text{constant}, \qquad w(t,\mathbf{x}) = \mathbf{n}\cdot\mathbf{x} - \lambda t \tag{2.1.1}$$

In fact, the classification of first-order systems suffices since it is always possible to rewrite higher-order partial differential equations as a first-order system of m equations with m unknowns [262].

This work is mainly concerned with the numerical solution of partial differential equations, and especially, concentrates on those used to model fluid dynamics applications. In order to get a

better understanding of the physical significance of the classification, it is worthwhile to present some well-known model equations for each type of PDE and discuss their respective properties. The reader who is interested in a comprehensive study of theoretical aspects of partial differential equations is referred to the classical textbook on CFD, e.g., by Hirsch [104] and Wesseling [262].

2.1.2. Partial differential equations of elliptic type

Consider the *Poisson equation* which is one of the most prominent examples of an elliptic PDE

$$-\Delta u = f \qquad \text{in} \quad \Omega \tag{2.1.2}$$

Its solution u describes the potential throughout Ω for the prescribed charge density f [41]. Its homogeneous counterpart ($f = 0$) is the well-known Laplace equation $\Delta u = 0$ which has many applications in elasticity, electromagnetism, and fluid dynamics.

Consider the rectangular two-dimensional domain $\Omega \subset \mathbb{R}^2$ sketched in Figure 2.1, where P denotes an arbitrary point in the interior. It follows from the theoretical analysis that information does not propagate in a preferred direction but travels essentially uniformly within the entire domain [8]. Hence, the solution at point P depends on all surrounding points of the closed domain Ω. On the other hand, small perturbations applied to the value at point P may influence the solution everywhere throughout the domain. Therefore, the domain of dependence and the region of influence for point P coincide which is common property of elliptic problems [104]. Numerical schemes for this class of PDEs must therefore determine the solution values at all points simultaneously, whereby the profile within the domain depends on the total boundary [8]. In fact, there are three different types of boundary conditions to be considered: Dirichlet boundary conditions which specify the value of the dependent variable at the boundary, Neumann boundary conditions which fix its normal derivatives and mixed-type boundary conditions.

In gas dynamics, solution techniques for elliptic problems need to be applied to subsonic inviscid flows which may reach steady state [8]. Owing to the fact that small disturbances travel at the speed of sound or even faster, they may propagate in all directions. In the absence of dissipative effects, no damping takes place so that the variations expand to infinity. The same applies to incompressible inviscid flows for which the speed of sound reaches infinity so that the Mach number tends to zero [8]. Flow problems may acquire elliptic character in regions of recirculation, where information may travel opposite to the principal flow direction [76]. On the other hand, unsteady problems are never of elliptic type [76].

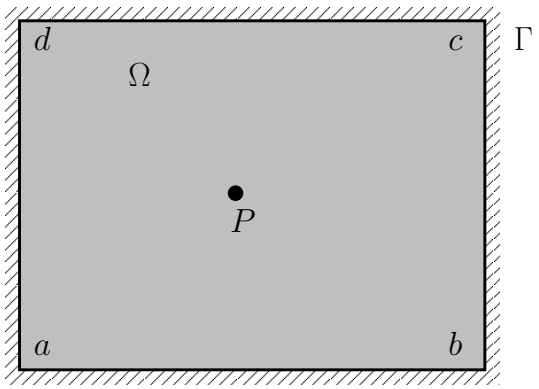


Fig. 2.1. Domain of dependence and region of influence for an elliptic problem.

2.1.3. Partial differential equations of parabolic type

A classical example of parabolic problems is the *transient heat equation* which is derived from Fourier's law of cooling. It describes the variation of temperature T in time [8]

$$\frac{\partial T}{\partial t} = \alpha \Delta T \qquad \text{in} \quad \Omega \times \mathbb{R}_0^+ \tag{2.1.3}$$

where the thermal diffusivity $\alpha = k/\rho c_v$ is a material-dependent parameter. It relates the thermal conductivity to the volumetric heat capacity and indicates the ability of the material to conduct energy [8]. If an initial temperature distribution is given, a flow of heat actuates from warmer regions to colder areas of the medium which is assumed to be homogeneous. Equation (2.1.3) is a special case of the more general diffusion equation $\frac{\partial u}{\partial t} + \nabla \cdot (D \nabla u) = 0$ with $D(u, \mathbf{x})$.

For the one-dimensional transient heat equation considered in the space-time domain, the domain of dependence is separated from the region of influence by a single line, whereby the union of both zones occupies the entire domain [172] as depicted in Figure 2.2. Consider the temperature value at the fixed point P which depends on all temperatures in the domain at previous times. Conversely, it will influence the temperature of the whole domain in the future so that the horizontal line can be interpreted as 'present time'. Due to this strict separation the use of marching techniques is favorable for the treatment of parabolic equations. Starting from an initial condition given along the straight line *ab* (see Figure 2.2), the solution is computed step by step whereby lateral boundary conditions need to be prescribed along the segments *ad* and *bc*, respectively [8]. Parabolic equations are typically related to transient problems for which the marching direction corresponds to the time variable t and an arbitrary number of space variables is admissible [8].

Another prominent example of a parabolic differential equation deals with stationary boundary-layer flows [262]. As a rule of thumb, the impact of viscosity is most pronounced in the vicinity of boundaries. In order to simplify the mathematical model and reduce the computational costs, it is possible to consider the flow to be inviscid in the interior of the domain and account for viscous effects only near the boundary. A typical application of this approach is the simulation of flow around an obstacle placed in the free stream, e.g., an airfoil or some missile. Owing to the fact that the boundary-layer equation is of parabolic type, its solution can be computed by means of time-marching methods. Away from boundaries, the flow is considered to be inviscid so that a different model equation has to be adopted. In general, their type may be different so that other and ideally cheaper solution strategies can be applied.

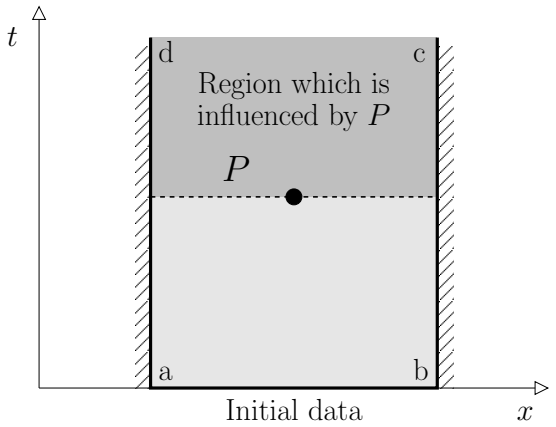


Fig. 2.2. Domain of dependence and region of influence for a parabolic problem.

2.1.4. Partial differential equations of hyperbolic type

Hyperbolic equations, like parabolic ones, can be solved using marching methods. In the hyperbolic case, the domain of dependence and the region of influence are bounded and they do not occupy the whole domain. As an example, consider the linear second-order *wave equation* [104]

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \qquad \text{in} \quad \Omega \times \mathbb{R}_0^+ \tag{2.1.4}$$

which describes the propagation of waves at constant speed c in a medium at rest. For each point P in the space-time domain, there exist two real and distinct characteristic curves intersecting at P as depicted in Figure 2.3. The domain of dependence and the region of influence are bounded by the characteristic curves and the boundary of the domain [104]. As a consequence, the solution value at point P is only influenced by a specific region of the domain which is identified by the direction of wave propagation. At the same time, information can only travel at finite speed within the region of influence 'leaving' point P. In most situations, one characteristic carries information *into* the domain whereas information travels *out of* the domain following the other characteristic. Thus, only one boundary condition can be prescribed at the boundary, namely, for the incoming characteristic [104]. The treatment of boundary conditions for hyperbolic equations is a difficult task which requires a careful investigation of the problem at hand. For inviscid compressible flows, this topic is briefly addressed in Section 5.3 and practical aspects are discussed.

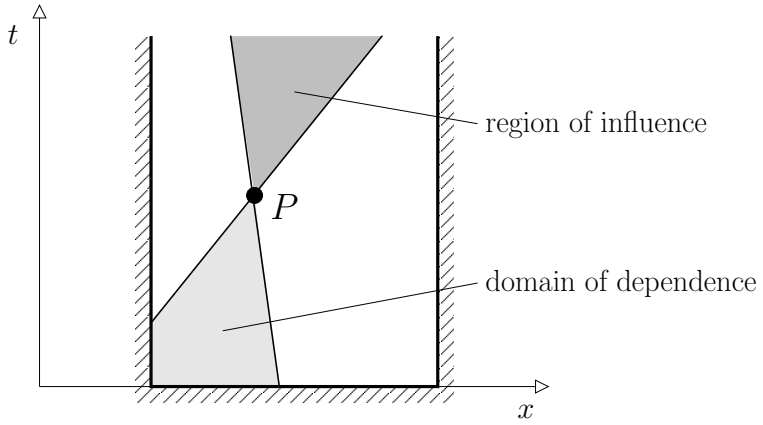


Fig. 2.3. Domain of dependence and region of influence for a hyperbolic problem.

2.1.5. Partial differential equations of mixed type

Most mathematical models adopted for practical applications cannot be classified simultaneously for all initial and boundary conditions so that the same PDE may exhibit different types subject to the concrete flow configurations. Consider the Euler-Tricomi equation [262]

$$-\frac{\partial^2 w}{\partial x^2} - y\frac{\partial^2 w}{\partial y^2} = 0 \qquad \text{in} \quad \Omega \tag{2.1.5}$$

which can be used to study steady transonic flows. The type of the above equation depends on the sign of y. In particular, $y > 0$ yields an elliptic equation which corresponds to subsonic flows. In contrast, supersonic flows, i.e. $y < 0$, give rise to a hyperbolic problem [262]. There exist many examples for mixed-type problems such as the stationary potential flow equation which can attain

each of the three types depending on the local flow speed (see Example 3.2.1 in Hirsch [104]). Due to the mixed nature of the potential equation, the simulation of transient problems is a challenging task and special solution algorithms are required. Some equations may not be classified at all since they exhibit an undefined type (see Example 2.2.1 in Wesseling [262]).

2.1.6. Summary of classification

As we have seen, many partial differential equations may be classified as elliptic, parabolic or hyperbolic. However, the type of a particular equation may be different for different initial conditions and it may even vary locally due to local flow properties. On the other hand, some problems may exhibit an undefined type either globally or for some special choice of parameters. The domain of dependence and the region of influence strongly depend on the type of the governing equation, and hence, different solution algorithms may be necessary for the different classes. Parabolic as well as hyperbolic equations can be solved by means of marching methods, whereas the solution to an elliptic equation has to be computed simultaneously for all points of the domain.

Marching techniques are a useful tool for the simulation of steady inviscid flows as long as long as the flow is supersonic so that the equations are of hyperbolic type. Subsonic regions are elliptic in character, and hence, different solution algorithms may be required [76, 262]. On the other hand, unsteady inviscid flows governed by the compressible Euler equations are unconditionally hyperbolic, no matter whether the flow is locally subsonic or supersonic. To stress this important property, such equations are frequently termed *hyperbolic in time* [8]. Regardless of the number of spatial dimensions, the distinct marching direction is dictated by the time variable t. Starting from an initial state, the solution can be computed step by step marching forward in time.

2.2. Nonlinear hyperbolic equations

In the introduction, we considered the linear advection equation (1.0.1) and analyzed some properties of its exact solution. In particular, the initial profile is transported along characteristic lines and does not change its shape. This is in contrast to nonlinear hyperbolic conservation laws which allow for the creation of discontinuities. This section deals with solutions to nonlinear hyperbolic equations for scalar variables. In particular, the one-dimensional Burgers equation is considered which represents the simplest model problem that captures some key features of gas dynamics, namely, the generation of shocks and rarefaction waves. The viscous Burgers equation does not allow for discontinuous solutions and its exact solution can be determined by means of the so-called Cole-Hopf-transformation [154, 263]. In contrast, discontinuities may be created in the absence of counterbalancing viscosity. Therefore, generalized solutions are sought in an integral sense. These weak solution may not be unique so that additional mechanisms are required to determine the entropy solution. In principle, the concepts discussed in this section also apply to nonlinear hyperbolic systems of conservation laws which are addressed in Chapter 5. It is beyond the scope of this work to present the theory of conservation laws in full detail. The aim is to point out the basic concepts and address the main difficulties. For a comprehensive coverage of this topic, the interested reader is referred to classical textbooks on conservation laws, e.g., [82, 154, 157].

2.2.1. Scalar conservation laws

As a model problem for a nonlinear scalar conservation law, consider the one-dimensional Cauchy problem for the inviscid Burgers equation in conservative form [154]

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \qquad \text{in} \quad \mathbb{R} \times \mathbb{R}_0^+, \qquad u(x,0) = u_0(x) \tag{2.2.1}$$

where $f(u) = \frac{1}{2}u^2$ denotes the flux function. By application of the chain rule, it can be written in quasi-linear form which states that the unknown solution u is advected by itself

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \qquad \text{in} \quad \mathbb{R}\times\mathbb{R}_0^+, \qquad u(x,0) = u_0(x) \tag{2.2.2}$$

This equation can be interpreted as follows [64]: For a smooth solution $u(x,t)$ fixed in time and space the total derivative with respect to time vanishes in the direction defined by the slope $\frac{\mathrm{d}x}{\mathrm{d}t} = u$. The latter quantity is termed *characteristic curve* and it follows from the quasi-linear form (2.2.2) that the solution values are constant along characteristics since its material derivative vanishes

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{\partial u}{\partial t} + \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\partial u}{\partial x} = 0 \tag{2.2.3}$$

Thus, the characteristic curve $\frac{\mathrm{d}x}{\mathrm{d}t} = u$ is a straight line as in the linear case (see Chapter 1). For continuous initial data, the value of the exact solution is implicitly defined as follows [154]

$$x_0 = x - u_0(x_0)t, \qquad u(x,t) = u_0(x_0) \tag{2.2.4}$$

provided the characteristics do not cross. The intersection of characteristics, which will eventually happen if the spatial derivative of the initial data $u_0(x)$ is negative at some point, is a special phenomenon observed for nonlinear conservation laws. First intersection takes place at [154, 251]

$$T_b = \frac{-1}{\min_x u_o'(x)} \tag{2.2.5}$$

which is referred to as the breakup time where the function $u(x,t)$ has an infinite slope. For times $t > T_b$ some of the characteristics have crossed so that there exists points x where multiple lines lead back to $t = 0$. As an example [64], consider the decreasing initial profile depicted in Figure 2.4 (left). Let us analyze the exact solution *before* the break-up at time $T_b = 1$. For $x \le t$ and $x \ge 1$, the characteristics are given by $\frac{\mathrm{d}x}{\mathrm{d}t} = 1$ and $\frac{\mathrm{d}x}{\mathrm{d}t} = 0$, respectively. As a result, the exact solution is given by the initial data. For the interval $t < x < 1$, the characteristic line can be determined by solving the ordinary differential equation $\frac{\mathrm{d}x}{\mathrm{d}t} = u$ subject to the initial condition u_0 so that [205]

$$\text{For} \quad t < x < 1: \quad x = (1-x_0)t + x_0 \qquad \Rightarrow \qquad u(x,t) = 1 - x_0 = \frac{1-x}{1-t} \tag{2.2.6}$$

As depicted in Figure 2.4 (left), the exact solution develops a discontinuity at time $t = 1$ which is due to the fact that slower waves are overtaken by faster ones propagating information from the left. The intersection of characteristics at time $t = 1$ is illustrated in Figure 2.4 (right). Beyond the breakup time T_b a classical solution to the conservation laws (2.2.2) does not exist, and hence, more general concepts for the treatment of discontinuous solutions are mandatory.

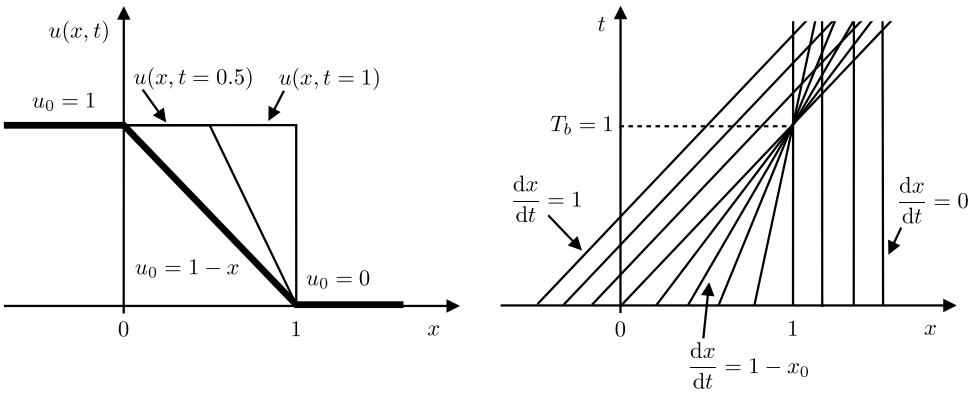


Fig. 2.4. Burgers' equation: Crossing of characteristic curves at time T_b.

2.2.2. Weak solutions

To weaken the continuity requirements on the solution $u(x,t)$ one can multiply the conservative form (2.2.1) by some test function $w(x,t) \in \mathcal{C}_0^1(\mathbb{R}\times\mathbb{R}_0^+)$ and integrate over time and space [154]

$$\int_0^\infty \int_{-\infty}^\infty w\left[\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x}\right] \mathrm{d}x\,\mathrm{d}t = 0 \tag{2.2.7}$$

Here, $\mathcal{C}_0^1$ denotes the space of continuous functions which exhibit continuous first derivatives and have compact supports, that is, $w(x,t) = 0$ outside of some bounded set. Integration by parts (using Green's formula) can be performed to 'shift' the derivatives from the solution to the test function

$$\int_0^\infty \int_{-\infty}^\infty \frac{\partial w}{\partial t}u + \frac{\partial w}{\partial x}f(u)\,\mathrm{d}x\,\mathrm{d}t = -\int_{-\infty}^\infty w(x,0)u(x,0)\,\mathrm{d}x \tag{2.2.8}$$

whereby the boundary terms vanish due to the compact support property of w. A measurable function $u(x,t)$ is termed a *weak solution* of the conservation law (2.2.1) provided the integral relation (2.2.8) is satisfied for *all* test functions $w(x,t) \in \mathcal{C}_0^1(\mathbb{R}\times\mathbb{R}_0^+)$. Of course, the (strong) solution of the differential equation (2.2.1) is a weak solution. On the other hand, the concept of weak solutions is more general since relation (2.2.8) does neither include derivatives of the unknown solution u nor of the flux function $f(u)$. Therefore, weak solutions to conservation laws are not necessarily unique so that additional criteria are required to pick out the physically relevant *entropy solution*. The correct weak solution to the inviscid Burgers equation can be recovered from the viscous counterpart of conservation law (2.2.1) in the limit of vanishing viscosity $\varepsilon \to 0$ [64]

$$\frac{\partial u_\varepsilon}{\partial t} + \frac{\partial f(u_\varepsilon)}{\partial x} = \varepsilon\frac{\partial^2 u_\varepsilon}{\partial x^2} \qquad \text{in} \quad \mathbb{R}\times\mathbb{R}_0^+, \qquad u_\varepsilon(x,0) = u_0(x) \tag{2.2.9}$$

There exist other conditions which are easier to check without resorting to the viscous equation.

Shock waves

To discuss weak solutions, it is worthwhile to consider the piecewise constant initial data

$$u(x,0) = \begin{cases} u_L & \text{if} \quad x < 0 \\ \\ u_R & \text{if} \quad x > 0 \end{cases} \qquad \text{where} \quad u_L > u_R \tag{2.2.10}$$

which is a special case of the discontinuous function generated by the initial profile depicted in Figure 2.4 (left). For each time $t \geq 0$, the solution has two constant states, whereby the characteristic lines run into the discontinuity from both sides as sketched in Figure 2.5. In gas dynamics applications, such kind of discontinuities are termed *shocks*. The unique weak solution is given by

$$u(x,t) = \begin{cases} u_L & \text{if} \quad x < st \\ \\ u_R & \text{if} \quad x > st \end{cases} \tag{2.2.11}$$

whereby the *shock speed* s must satisfy the so-called *Rankine-Hugoniot condition* [154]

$$f(u_L) - f(u_R) = s(u_L - u_R) \tag{2.2.12}$$

which relates the jump in the solution to the difference of the flux functions. For the inviscid Burgers equation (2.2.1) it is given by the average velocity $s = \frac{1}{2}(u_L + u_R)$.

Remark. The smooth solution u_ε of the viscous Burgers equation (2.2.9) reads as follows [64]

$$u_\varepsilon(x,t) = u_R + \frac{u_L - u_R}{2}\left[1 - \tanh\left(\frac{(u_L - u_R)(x - st)}{2\varepsilon}\right)\right] \tag{2.2.13}$$

so that the unique solution (2.2.11) is in fact the desired vanishing viscosity solution.

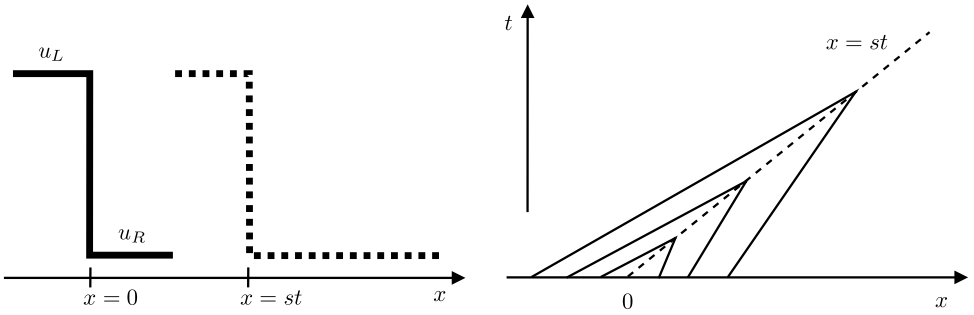


Fig. 2.5. Burgers' equation: Shock wave.

Rarefaction waves

If the initial data to the Cauchy problem is given by the following piecewise constant profile

$$u(x,0) = \begin{cases} u_L & \text{if} \quad x < 0 \\ u_R & \text{if} \quad x > 0 \end{cases} \qquad \text{where} \quad u_L < u_R \tag{2.2.14}$$

there exist infinitely many weak solutions. Again, one of them is given by expression (2.2.11), whereby the characteristics emanate from the discontinuity as illustrated in Figure 2.6.

Note that this solution is highly unstable in the sense that small perturbations of the initial data may lead to large changes in the final profile [251]. Moreover, adding a slight amount of viscosity changes the solution profile completely [64]. Such solutions are termed *entropy-violating shocks* and they are physically incorrect. Another weak solution is the *rarefaction wave* given by [154]

$$u(x,t) = \begin{cases} u_L & \text{if} \quad x < u_L t \\ x/t & \text{if} \quad u_L t \le x \le u_R t \\ u_R & \text{if} \quad x > u_R t \end{cases} \tag{2.2.15}$$

which is stable to perturbations. The smooth transition from the left state u_L to the right state u_R is depicted in Figure 2.7. It is the unique entropy solution of the viscous Burgers equation (2.2.9) in the limit of vanishing viscosity. However, the vanishing viscosity approach may be difficult to work with in practice. For stability reasons, characteristics should not emanate from a shock as time evolves (see Figure 2.6) but they must always run *into* the shock as shown in Figure 2.5.

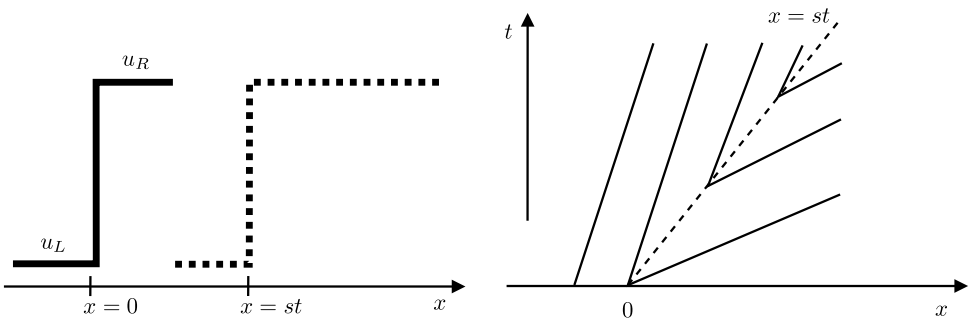


Fig. 2.6. Burgers' equation: Entropy-violating shock wave.

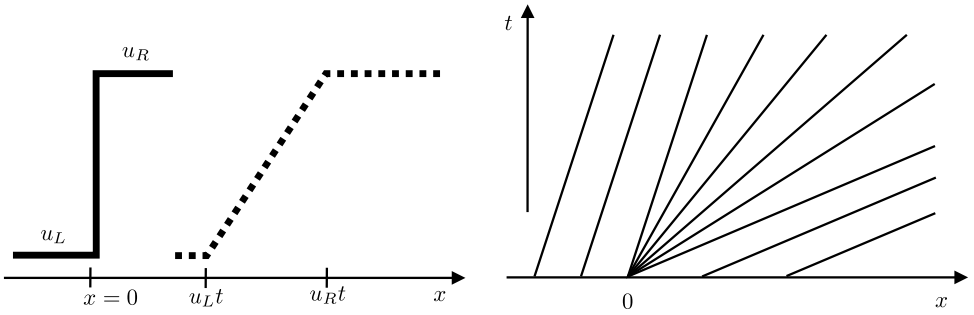


Fig. 2.7. Burgers' equation: Rarefaction wave.

2.2.3. Entropy condition

A very general criterion to check for entropy-violating shocks for arbitrary flux functions $f(u)$ is due to Oleĭnik [198]. Since the shock speed s can be computed directly from the Rankine-Hugoniot condition (2.2.12) it suffices to consider the following entropy condition

Theorem 2.2.1 (Oleĭnik's entropy condition [154])**.** A weak solution to equation (2.2.1) with initial data (2.2.10) or (2.2.14) is the physically correct entropy solution if all discontinuities satisfy

$$\frac{f(u)-f(u_L)}{u-u_L} \geq s \geq \frac{f(u)-f(u_R)}{u-u_R} \qquad \forall u = \theta u_L + (1-\theta)u_R, \quad 0 < \theta < 1 \tag{2.2.16}$$

Geometrically speaking, Oleĭnik's criterion guarantees that the flux function does not intersect the secant line connecting the two states u_L and u_R [149]. In particular, it lies entirely above that line for $u_L < u_R$ and it lies below the secant for $u_L > u_R$. The flux function for the Burgers equation is convex, i.e. $f(u)'' \geq 0$ for all u, so that Oleĭnik's condition reduces to the entropy condition [154]

$$f'(u_L) > s > f'(u_R) \tag{2.2.17}$$

Again, making use of convexity arguments, the above inequality necessarily requires $u_L > u_R$. Thus, the shock wave presented in Figure 2.5 is physically correct whereas the unstable rarefaction shock [251] depicted in Figure 2.6 violates the entropy conditions (2.2.16) and (2.2.17).

2.2.4. Summary of nonlinear hyperbolic equations

To summarize what we have said so far, nonlinear hyperbolic equations may give rise to the development of steep fronts and discontinuities even if the initial profile is smooth. Shock waves are characterized by the crossing of characteristic curves, so that classical solution techniques based on the method of characteristics can no longer be applied. As a remedy, generalized solutions are considered in a weak sense which allow for the presence of discontinuities. However, weak solutions to a given initial value problem may not be unique so that additional criteria are required to determine the physically correct solution. For an increasing initial profile, a possible weak solution to the inviscid Burgers equation may be given by entropy-violating shocks which are highly unstable to small perturbations. Moreover, such solutions do not coincide with the unique rarefaction wave recovered from the viscous Burgers equation in the limit of vanishing viscosity. Oleĭnik's entropy condition can be employed to ensure that a weak solution to the inviscid Burgers equation is in fact the physically correct entropy solution. As a rule of thumb, characteristics should always run into the shock but they must not emanate from discontinuities.

3

Numerical methods for scalar conservation laws

3.1. Introduction to numerical methods

For most conservation laws of practical interest, the computation of exact solutions is not possible so that one has to rely on numerical methods for finding approximate solutions to the problem at hand: A suitable method is chosen and applied on a particular grid consisting of a finite number of points. This step entails certain errors which need to be analyzed in order to assess the quality of the computed solution. The *Lax equivalence theorem* of numerical methods for linear differential equations can be summarized colloquially by the following statement [157]:

$$\text{consistency} + \text{stability} \Leftrightarrow \text{convergence}$$

The next section is concerned with the theory of numerical methods for differential equations. In particular, aspects of consistency, stability and convergence are discussed for one-dimensional scalar conservation laws. In addition, mathematical criteria to ensure the boundedness of the numerical approximation are presented in Section 3.1.2. Such constraints can be directly based on the total variation of the solution which should not increase as time evolves. The local extremum diminishing property constitutes a viable generalization to multidimensions and it can be used to design non-oscillatory numerical methods on unstructured meshes. Some properties of non-negative matrices as well as M-matrices are reviewed and an algebraic criterion for the construction of positivity-preserving schemes is given [144]. This section is primarily based on the textbooks on numerical methods for conservation laws by LeVeque [154, 157], Kröner [128], and Toro [251], and on the research article by Kuzmin and Turek [144]. Results from matrix analysis are mainly taken from the books by Hackbusch [87] and Varga [257].

3.1.1. Convergence and stability

Consider the one-dimensional conservation law for a scalar quantity u in divergence form

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \qquad \text{in} \quad \mathbb{R} \times \mathbb{R}_0^+ \tag{3.1.1}$$

where $f : \mathbb{R} \to \mathbb{R}$ denotes a continuous flux function. The problem at hand is equipped with the initial condition $u(x,0) = u_0$ which should be bounded whereas boundary conditions are neglected to avoid technical difficulties. For simplicity, the resulting Cauchy problem is solved over the unbounded spatial domain $\mathbb{R}$, whereby the initial data is assumed to have compact support, that is, it is non-zero only in some bounded domain and vanishes at infinity. Owing to the finite propagation speed of hyperbolic problems, the exact solution has compact support at all times $t \in \mathbb{R}_0^+$, so that all integral terms presented below extend over finite intervals [157].

For a fixed computational grid, the nodal values of the exact solution are denoted by u_i^n, where superscript n refers to the time step $t^n \in \mathbb{R}_0^+$. In general, we expect errors to grow with time, and moreover, a finite grid may not be capable to reproduce meaningful solution values for arbitrarily large times. Therefore, it makes sense to compute the solution over a finite time interval $[0,T]$ when discussing 'convergence' [157]. For simplicity, the time step Δt is assumed to be constant so that $N = T/\Delta t$ denotes the total number of time steps. In addition, a uniform spatial grid is employed, where h stands for the characteristic mesh width. It is also reasonable to assume that the ratio $\Delta t/h$ is fixed which means that grid refinement goes along with a reduction of the time step size. The numerical solution is denoted by u_h so that u_i^n represents the approximate value at node i for the time t^n. The term 'node' can be used for finite difference, finite volume and finite element discretizations alike. However, the definition of u_i^n is different in each method.

The simplest discretization one may think of is an explicit numerical scheme of the form [157]

$$\mathrm{u}_h^{n+1} = \mathcal{N}(\mathrm{u}_h^n) \tag{3.1.2}$$

where the operator $\mathcal{N}$ maps the given solution at time t^n to the new one at time $t^{n+1} = t^n + \Delta t$. Application of the numerical operator $\mathcal{N}$ to the exact solution u^n and comparison to the true solution at the next time level yields an indicator for the error introduced in a single time step. The local truncation error is defined by dividing this one-step error by the time step size [157]

$$\tau^n = \frac{\mathcal{N}(u^n) - u^{n+1}}{\Delta t} \tag{3.1.3}$$

so as to measure the error inherent to the discretization scheme. An important property of numerical schemes is consistency which can be formalized as follows [157]: A numerical method is called consistent with the differential equation if the local truncation error (3.1.3) vanishes for all smooth solutions $u(x,t)$ as the mesh width h and the size of the time step Δt tend to zero.

In addition to the local truncation error, let the global error of the numerical solution at time t^n be defined as the difference between the numerical approximation and the exact solution

$$e^n = \mathrm{u}_h^n - u^n \tag{3.1.4}$$

This quantity can also be seen as the 'pollution' that has to be applied to the exact solution so as to obtain the numerical approximation, that is, $\mathrm{u}_h^n = u^n + e^n$. Making use of the explicit numerical procedure (3.1.2), the global error at the new time step can be reformulated as follows [157]

$$e^{n+1} = \mathcal{N}(u^n + e^n) - u^{n+1} \tag{3.1.5}$$

By virtue of equation (3.1.3), one can rewrite the global error at time t^{n+1} in terms of the previous error augmented by the local truncation error multiplied by the time step size

$$e^{n+1} = [\mathcal{N}(u^n + e^n) - \mathcal{N}(u^n)] + \Delta t \tau^n \tag{3.1.6}$$

Stability theory is required to obtain an upper bound for the term in brackets which measures the effect of the discretization on the previous global error e^n. On the other hand, the one-step error $\Delta t \tau^n$ engendered in the current step can be bounded by means of consistency arguments [157].

Assume that the operator $\mathcal{N}$ is contractive, that is, $\|\mathcal{N}(v) - \mathcal{N}(w)\| \leq \|v - w\|$ for any two grid functions v and w. A contractive numerical method is stable in the particular norm, and an upper bound for the global error (3.1.6) can be readily obtained by setting $v = u^n + e^n$ and $w = u^n$ [157]:

$$\|e^{n+1}\| \leq \|\mathcal{N}(u^n + e^n) - \mathcal{N}(u^n)\| + \Delta t \|\tau^n\| \leq \|e^n\| + \Delta t \|\tau^n\| \tag{3.1.7}$$

This technique can be applied recursively to relate the global error at the final time $T = N\Delta t$ to the error in the initial data augmented by the cumulated contributions of the local truncation errors

$$\|e^N\| \leq \|e^0\| + \Delta t \sum_{n=1}^{N-1} \|\tau^n\| \tag{3.1.8}$$

The sum over local contributions can be bounded by the largest addend so as to obtain [157]:

$$\|e^N\| \leq \|e^0\| + T \max_{1\leq n\leq N-1} \|\tau^n\|, \qquad T = N\Delta t \tag{3.1.9}$$

The error in the initial data is required to vanish as $h \to 0$ to guarantee that the correct initial value problem is solved. Provided the numerical scheme is consistent, that is, each truncation error $\|\tau^n\|$ vanishes as the time step size and the mesh width tend to zero, this yields convergence $\|e^N\| \to 0$.

Alternative approaches to stability analysis for linear methods exist, such as the von Neumann stability analysis, which can be found in the literature, for instance in [128, 157].

3.1.2. Boundedness

In addition to the presented properties – consistency, stability and convergence – an essential feature of a numerical scheme is related to the boundedness of the approximate solution. As a motivation, consider the oscillatory solution profile depicted in Figure 1.2 in Chapter 1. In order to prevent the formation of spurious undershoots and overshoots in the vicinity of steep gradients it makes sense to ensure that some natural properties of the exact solution are inherited by the numerical approximation. A useful property of the entropy-satisfying weak solution to a scalar conservation law is that two sets of initial data with $u_0(x) \geq v_0(x)$ everywhere in the domain, lead to entropy solutions $u(x,t) \geq v(x,t)$ for all x and all later times $t > 0$ [154].

Monotone methods

Numerical methods mimicking this property are termed monotone schemes, if $\mathrm{u}_i^n \geq \mathrm{v}_i^n$ for all nodes i necessarily implies that $\mathrm{u}_i^{n+1} \geq \mathrm{v}_i^{n+1}$ everywhere. For linear methods expressed by the explicit formula (3.1.2), monotone schemes can be equivalently characterized as follows [154]:

$$\frac{\partial \mathrm{u}_i^{n+1}}{\partial \mathrm{u}_j^n} = \frac{\partial \mathcal{N}(\mathrm{u}_h^n; i)}{\partial \mathrm{u}_j^n} \geq 0, \quad \forall\, i,\, j \tag{3.1.10}$$

This inequality means that function $\mathcal{N}$ is monotonically non-decreasing in any argument [128] so that the new solution u_i^{n+1} at all nodes i cannot decrease, whenever the value of any u_j^n from the previous time step is increased [154]. As a direct consequence, global maxima (minima) cannot increase (decrease) from one time step to the other, and hence, the maximal (minimal) values of the initial condition may serve as upper (lower) bounds for all times [251]:

$$\begin{array}{ccccccc} \max\limits_j \mathrm{u}_j^{n+1} & \leq & \max\limits_j \mathrm{u}_j^n & \leq & \ldots & \leq & \max\limits_j \mathrm{u}_j^0 \\ \min\limits_j \mathrm{u}_j^{n+1} & \geq & \min\limits_j \mathrm{u}_j^n & \geq & \ldots & \geq & \min\limits_j \mathrm{u}_j^0 \end{array} \tag{3.1.11}$$

Consequently, no creation of spurious oscillations takes place since the numerical scheme does not generate new extrema. In other words, minima tend to increase whereas maxima are likely to decrease as time evolves. This leads to some clipping of extrema which is a common drawback of monotone numerical schemes [251]. From the above inequalities one can also conclude that the solution at any point i is bounded for all times by the initial data, i.e. $\min_j \mathrm{u}_j^0 \leq \mathrm{u}_i^n \leq \max_j \mathrm{u}_j^0$. This property is especially useful for the computation of numerical approximations to physical quantities such as the mass density ρ which must remain strictly positive.

Remark. If a general monotone scheme is also consistent with the conservation law, then it is possible to prove convergence to an entropy solution provided the initial data is bounded and the numerical flux function is sufficiently smooth [92, 93]. A comprehensive proof which covers all technical details is given in the textbook by Kröner [128]. It is applicable to linear and nonlinear discretizations alike, and moreover, it allows for explicit and implicit time-stepping schemes. The topic of convergence analysis was also addressed by Sanders in [229] who investigated the convergence behavior of monotone finite difference schemes on non-uniform grids.

In his well-known theorem, Godunov [83] proved that linear schemes which satisfy the monotonicity property can be at most first-order accurate. In other words, there are no monotone, linear schemes of second- or higher-order accuracy [251]. For practical applications, first-order approximations are insufficient to produce accurate results. Hence, other classes of numerical methods are required which exhibit the non-oscillatory behavior of monotone schemes and, at the same time, provide the accuracy of high order methods. The key idea is to sacrifice the linearity of the numerical scheme in favor of the monotone property so as to circumvent Godunov's first-order barrier. Ideally, this will prevent the formation of spurious oscillations since monotonicity does not allow for the creation of new extrema; cf. upper/lower bounds(3.1.11).

Total variation diminishing methods

The general definition of monotone schemes does not provide a suitable criterion that can be checked in practice. Moreover, the monotonicity property is very restrictive so that weaker constraints are preferable to control the boundedness of the solution. A handy criterion is based on the total variation (TV) of a scalar quantity u which is defined as follows [251]

$$\mathrm{TV}(u) = \lim_{\delta\to 0}\sup \frac{1}{\delta}\int_{-\infty}^{\infty} |u(x+\delta)-u(x)|\,\mathrm{d}x \tag{3.1.12}$$

If the function $u = u(x)$ is differentiable, then its total variation is equivalently given by [251]

$$\mathrm{TV}(u) = \int_{-\infty}^{\infty} \left|\frac{\partial u(x)}{\partial x}\right| \mathrm{d}x \tag{3.1.13}$$

Strictly speaking, the above definition can still be applied to non-differentiable functions if we consider $\partial u/\partial x$ to be the distribution derivative. For time-dependent functions $u = u(x,t)$, definitions (3.1.12) and (3.1.13) can be generalized accordingly [154]. As we are about to see, it suffices to consider the total variation at a fixed time t^n so as to control the growth of solution values. The discrete analogon of definition (3.1.13) for a valid grid function u_h^n reads as follows [251]

$$\mathrm{TV}(\mathrm{u}_h^n) = \sum_{i=-\infty}^{\infty} |\mathrm{u}_{i+1}^n - \mathrm{u}_i^n| \tag{3.1.14}$$

The initial data is assumed to have compact support. Due to the finite propagation speed inherent to hyperbolic problems, this property carries over to the solution values at all time levels t^n. Therefore, the total variation $\mathrm{TV}(\mathrm{u}_h^n)$ is bounded and does not tend to infinity. Laney [148] observed that the total variation (3.1.14) equals the sum of extrema; maxima counted positively and minima counted negatively. Each extremum is counted twice except for the two infinite boundary values $\mathrm{u}_{-\infty}^n$ and u_{∞}^n, which are supposed to vanish due to the compact support assumption.

Consider an initial value problem for a scalar conservation law in divergence form (3.1.1), where the total variation of the initial data is assumed to be finite. A fundamental theorem states that maxima do not increase, minima do not decrease and no new extrema are created [251]. In other words, the exact solution satisfies the *total variation diminishing* (TVD) property [92, 150]

$$\mathrm{TV}(u(x,t_2)) \le \mathrm{TV}(u(x,t_1)), \quad \forall t_2 \ge t_1 \in [0,T] \tag{3.1.15}$$

In his original paper on high resolution schemes [92], Harten suggested the name total variation non-increasing (TVNI) which describes the underlying principle more precisely but subsequent articles [93] adopted the less cumbersome term total variation diminishing (TVD).

Definition 3.1.1 (TVD schemes [157])**.** A two-level numerical method is said to be total variation diminishing if for any given data u_h^n the computed solution u_h^{n+1} satisfies the following inequality

$$\mathrm{TV}(\mathrm{u}_h^{n+1}) \leq \mathrm{TV}(\mathrm{u}_h^n) \tag{3.1.16}$$

For TVD methods, the above definition can be applied recursively to relate the total variation of the current solution to that of the initial data $\mathrm{TV}(\mathrm{u}_h^{n+1}) \leq \mathrm{TV}(\mathrm{u}_h^0)$ which is bounded by assumption. Let the initial profile u_h^0 be monotone, that is, it is globally non-increasing or non-decreasing

$$\mathrm{u}_i^0 \geq \mathrm{u}_{i+1}^0 \qquad \text{or} \qquad \mathrm{u}_i^0 \leq \mathrm{u}_{i+1}^0, \quad \forall i \tag{3.1.17}$$

If the numerical method features the total variation diminishing property, then the approximate solution will remain monotone in all future time steps, that is, monotonicity is preserved [128].

Definition 3.1.2 (Monotonicity-preserving schemes [157])**.** A two-level numerical method is termed *monotonicity-preserving* (MP) if the following implication holds for all nodes i:

$$\mathrm{u}_i^n \leq \mathrm{u}_{i+1}^n, \quad \forall i \qquad \Rightarrow \qquad \mathrm{u}_i^{n+1} \leq \mathrm{u}_{i+1}^{n+1}, \quad \forall i \tag{3.1.18}$$

Harten [92] demonstrated that the following hierarchy among the different classes of numerical methods holds for a general nonlinear scalar conservation law of the form (3.1.1):

$$\text{monotone} \quad \subseteq \quad \begin{matrix}\text{total variation}\\ \text{diminishing}\end{matrix} \quad \subseteq \quad \begin{matrix}\text{monotonicity}\\ \text{-preserving}\end{matrix} \tag{3.1.19}$$

Definition (3.1.10) of monotone schemes suffices to control the boundedness of the solution but it is impractical for the design of numerical methods. Hence, the aim is to base the construction of non-oscillatory discretization techniques on less restrictive constraints such as the total variation diminishing property or the MP principle which can be easily verified in practice.

One can show [251] that all three classes coincide for **linear** schemes (3.1.2) applied to the divergence form (3.1.1) with linear flux function $f(u) = au$ and constant velocity a. Hence, all such schemes are doomed to be first-order accurate at most according to Godunov's theorem [83]. In contrast, for **nonlinear** numerical methods, monotone schemes form a real subclass of total variation diminishing schemes which are contained in the even more general class of monotonicity-preserving approximations. As we are about to see in Section 3.1.3, the total variation diminishing (TVD) property can be turned into a handy criterion for the design of non-oscillatory numerical methods. Imposing upper bounds on the total variation is also useful to study the stability of nonlinear numerical methods. It is very difficult or even impossible to prove convergence for nonlinear schemes applicable to general systems of equations. On the other hand, the TVD property is a useful tool to prove convergence in the context of scalar conservation laws [157].

Consider an explicit method of the form (3.1.2) and adopt a special mapping operator

$$\mathrm{u}_i^{n+1} = \mathrm{u}_i^n - \frac{\Delta t}{h}\left[f_{i+1/2}^n - f_{i-1/2}^n\right] \tag{3.1.20}$$

where $f_{i-1/2}^n$ represents an approximation to the time-averaged flux at point $x_{i-1/2}$ [157]

$$f_{i-1/2}^n \approx \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(u(x_{i-1/2},t))\,\mathrm{d}t \tag{3.1.21}$$

In general, the above quantity depends on the solution values in the vicinity of point $x_{i-1/2}$, i.e. $f^n_{i-1/2} = F(u^n_{i-k_L}, \ldots, u^n_{i+k_R})$, whereby k_L and k_R denote two non-negative integers and the numerical flux function F is supposed to satisfy the *consistency* condition $f(\bar{u}) = F(\bar{u}, \ldots, \bar{u})$ for any constant profile $\bar{u}$ [251]. For hyperbolic problems, information travels at finite speed, and hence, it is reasonable to evaluate $f^n_{i-1/2}$ based on the data u^n_{i-1} and u^n_i, that is, $f^n_{i-1/2} = F(\mathrm{u}^n_{i-1}, \mathrm{u}^n_i)$. Then the explicit update formula (3.1.20) yields a numerical method with a three-point stencil [157]

$$\mathrm{u}^{n+1}_i = \mathrm{u}^n_i - \frac{\Delta t}{h}\left[F(\mathrm{u}^n_i, \mathrm{u}^n_{i+1}) - F(\mathrm{u}^n_{i-1}, \mathrm{u}^n_i)\right] \tag{3.1.22}$$

Such schemes are termed *conservative* since they mimic the behavior of the integral form. A fundamental theorem states that a consistent and conservative numerical method which features a Lipschitz continuous flux function is convergent in some special norm if it satisfies the total variation diminishing property given in Definition 3.1.1. A comprehensive proof can be found in the textbook by LeVeque [157], whereby the TVD property is slightly relaxed. In particular, it suffices to require that the total variation of the numerical solution be uniformly bounded.

Definition 3.1.3 (Total variation bounded [157])**.** A numerical scheme is *total variation bounded* (TVB) if for any initial data u^0_h which satisfies $\mathrm{TV}(u^0_h) < \infty$ and for any final time T there exists a constant $R > 0$ and a value $\Delta t_0 > 0$ such that the total variation is uniformly bounded, that is,

$$\mathrm{TV}(\mathrm{u}^n_h) \leq R, \quad \forall n, \Delta t \quad \text{with} \quad \Delta t < \Delta t_0, \quad n\Delta t \leq T \tag{3.1.23}$$

A TVD scheme certainly is TVB and the constant R equals the total variation of the initial data.

The above criteria guarantee that a sequence of approximations $\mathrm{u}^{(\Delta t)}_h$ converges to some fixed 'state' as the mesh width h and the time step size Δt are refined. However, it is questionable if this function is a weak solution to the original conservation laws. This question is typically answered by considering the famous Lax-Wendroff theorem which reads as follows:

Theorem 3.1.1 (Lax and Wendroff [151])**.** If a consistent and conservative method converges to some function $u(x,t)$ as h and Δt go to zero, then $u(x,t)$ is a weak solution of the conservation law.

Note that the above theorem does not provide a criterion to ensure convergence. It only states that *if* a consistent and conservative method converges then the converged quantity actually is a weak solution of the conservation law. Therefore, an additional criterion such as the TVD property or the boundedness of the total variation is necessary to guarantee convergence at all.

Let us conclude what we have said so far. Consistency and stability of a linear numerical method are sufficient to prove convergence. In addition, the approximate solution should satisfy some physical constraints, e.g., the mass density must remain positive for all times. This can be achieved by using monotone schemes for which the numerical solution is bounded by the maximum and minimum values of the initial data. Moreover, new extrema are not created so that the final solution is free of non-physical undershoots and overshoots. As an alternative, the increase of total variation can be bounded in order to prevent the formation of non-physical oscillations and ensure that monotonicity is preserved. Unfortunately, for linear approximations there is no difference between monotonicity-preserving, total variation diminishing and monotone methods so that Godunov's first-order barrier applies to all such schemes. On the other hand, there is a real hierarchy for nonlinear approximations, with monotonicity-preserving schemes being the most general class and monotone methods constituting the most specialized subclass. In particular, the total variation diminishing property lends itself to the design of non-oscillatory numerical methods since it can be verified in practice. For consistent and conservative methods, the TVB and TVD criteria can be used to prove convergence to weak solutions of scalar conservation laws.

3.1.3. Design criteria for numerical methods

This section is devoted to the design of non-oscillatory numerical methods applicable to scalar problems in multidimensions. Semi-discrete schemes are considered in the first part, and the *local extremum diminishing* property is investigated which can be used to prevent the formation of spurious oscillations. The second part mainly deals with some properties of non-negative matrices as well as *M-matrices*, which can be used to ensure that solution values do not change their sign.

Local extremum diminishing methods

The total variation diminishing (TVD) property presented in the previous section provides a valuable tool for the design of numerical methods since the total variation of the initial data represents an upper bound on $TV(\mathrm{u}_h)$ for all times. In particular, TVD schemes do not tend to create spurious oscillations since the monotonicity of the initial profile is preserved, and hence, no new extrema are generated. The true solution of a scalar conservation law is still total variation diminishing in multidimensions, whereby the definition of the total variation (3.1.12) needs to be extended accordingly [157]. On the other hand, Goodman and LeVeque [84] demonstrated that any TVD method in two space dimensions is in general at most first-order accurate except for some special cases. Hence, a genuinely multidimensional generalization of the total variation diminishing condition is required for the design of non-oscillatory schemes in multidimensions.

Let the nodal value of the scalar solution u_h satisfy the following ordinary differential equation

$$\frac{\mathrm{d}\mathrm{u}_i}{\mathrm{d}t} = \sum_j c_{ij}\mathrm{u}_j \tag{3.1.24}$$

where the coefficients c_{ij} account for the spatial approximation. Suppose that they have zero row sums so that the diagonal coefficient can be expressed in terms of off-diagonal ones

$$\sum_j c_{ij} = 0 \qquad \Rightarrow \qquad c_{ii} = -\sum_{j\neq i} c_{ij} \tag{3.1.25}$$

whereby the notation '$j \neq i$' implies that summation is performed over $j = 1,\ldots,M$ skipping index i. It was shown by Jameson that the birth and growth of non-physical undershoots and overshoots can be prevented by enforcing that all off-diagonal coefficients are non-negative [113–115].

Theorem 3.1.2 (Local extremum diminishing scheme [114])**.** A semi-discrete scheme of the form

$$m_i \frac{\mathrm{d}\mathrm{u}_i}{\mathrm{d}t} = \sum_{j\neq i} c_{ij}(\mathrm{u}_j - \mathrm{u}_i) \tag{3.1.26}$$

with $m_i > 0$ is stable in the $L_\infty-$norm if all off-diagonal coefficients c_{ij} are non-negative, i.e.

$$c_{ij} \geq 0, \quad \forall\, j \neq i \tag{3.1.27}$$

Proof. Let the solution attain its global maximum at some node, say i, so that $\mathrm{u}_i = \max_j \mathrm{u}_j$. As a consequence, the solution differences satisfy $\mathrm{u}_j - \mathrm{u}_i \leq 0$ for all indices j. Owing to the fact that all coefficients $c_{ij} \geq 0$ by hypothesis, the right-hand side of the above equation does not exceed zero. Therefore, condition $\frac{\mathrm{d}\mathrm{u}_i}{\mathrm{d}t} \leq 0$ implies that maxima do not increase. Conversely, the global minimum $\mathrm{u}_i = \min_j \mathrm{u}_j$ does not decrease since $\mathrm{u}_j - \mathrm{u}_i \geq 0$ for all nodes j, and hence, $\frac{\mathrm{d}\mathrm{u}_i}{\mathrm{d}t} \geq 0$. □

In conclusion, the above condition guarantees that a numerical method which can be represented by formula (3.1.26) is stable in the L_∞-norm provided that all off-diagonal coefficients are non-negative. As a rule, coefficient matrices resulting from finite element/finite volume discretizations are sparse so that $c_{ij} = 0$ unless i and j are adjacent nodes. Arguing as above, one can prove

that *local* maxima cannot increase, and similarly *local* minima cannot decrease. Schemes which exhibit this property are called local extremum diminishing. The LED criterion has been applied by various authors to construct oscillation-free spatial discretizations on (un-)structured meshes [39, 116, 268]. Moreover, the positivity of physical quantities such as concentrations or densities is preserved since the global minimum of the initial data serves as a lower bound for the solution.

Three-point finite difference methods in one space dimension which are local extremum diminishing necessarily satisfy Harten's total variation diminishing property [92]. To illustrate this fact, consider definition (3.1.14) of the total variation for a discrete grid function u_h. If homogeneous Dirichlet boundary conditions are prescribed at both endpoints, the total variation of the piecewise linear approximate solution can be expressed as sums of local extrema [114, 148]

$$\mathrm{TV}(\mathrm{u}_h) = \sum_{i=-\infty}^{\infty} |\mathrm{u}_{i+1} - \mathrm{u}_i| = 2\left(\sum \max \mathrm{u}_h - \sum \min \mathrm{u}_h\right) \tag{3.1.28}$$

Since local extremum diminishing schemes do not lead to an enhancement of maxima and minima as time evolves the total variation of the solution cannot increase. The equivalence of one-dimensional LED and TVD schemes was also proven by Tadmor [246]. Linear total variation diminishing schemes coincide with monotone methods, and hence, they can be first-order accurate at most [83]. This order barrier also applies to linear discretizations which satisfy the LED constraint. Thus, the non-negative coefficients in the local extremum diminishing scheme (3.1.26) have to depend on the unknown solution in order obtain a high-resolution method.

In light of the above, Jameson's LED criterion represents a multidimensional generalization of Harten's TVD constraint (3.1.23) which can be used for the design of non-oscillatory discretizations. It suffices to ensure that all off-diagonal coefficients c_{ij} have positive sign which can be accomplished for arbitrary discretizations on unstructured meshes in multidimensions.

Positivity-preserving methods

So far, only design criteria for spatial approximations have been investigated, intentionally neglecting the discretization in time. Consider a general algebraic system of the form

$$Au = Bv \qquad \text{where} \quad A, B \in \mathbb{R}^{n\times n} \quad \text{and} \quad u, v \in \mathbb{R}^n \tag{3.1.29}$$

One may think of $u = \mathrm{u}_h^{n+1}$ and $v = \mathrm{u}_h^n$ as the solution vectors at two consecutive time levels, while A and B are the discrete matrix operators representing the spatial discretization and the time-stepping scheme. The task is to find sufficient criteria to verify that some property of the old solution, i.e. non-negativity, carries over to the new one. Kuzmin and Turek [144] proved that this is the case if all entries of matrix B are non-negative and the operator A is a so-called *M-matrix*.

Definition 3.1.4 (M-matrix [257])**.** A square matrix $A = \{a_{ij}\}$ with $a_{ij} \in \mathbb{R}$ is called an M-matrix if it is non-singular, its off-diagonal coefficients satisfy $a_{ij} \leq 0$ for all $j \neq i$ and $A^{-1} \geq 0$.

The existence of the non-negative inverse matrix $A^{-1} \geq 0$ lead Kuzmin and Turek [144] to the introduction of the following positivity criterion for general two-level time-stepping schemes:

Theorem 3.1.3 (Positivity-preserving (PP) scheme [144])**.** A fully discrete numerical scheme is positivity-preserving if it can be written as an algebraic system of the form

$$A\mathrm{u}_h^{n+1} = B\mathrm{u}_h^n \tag{3.1.30}$$

where $A = \{a_{ij}\}$ is an M-matrix and $B = \{b_{ij}\}$ has no negative entries.

Proof. $\mathrm{u}_h^n \geq 0 \Rightarrow \mathrm{u}_h^{n+1} = A^{-1}B\mathrm{u}_h^n \geq 0$; cf. Positivity Theorem 2 by Kuzmin and Turek [144]. □

In short, the positivity-preserving property can be directly verified for any one-step method of the form (3.1.30) by checking the M-matrix property of A and ensuring that matrix B is non-negative. Of course, this may not be the case for arbitrary time-stepping techniques but for the standard θ-scheme it can be achieved by adjusting the employed time step size [144].

Criteria for admissible time steps

Let the semi-discrete problem (3.1.26) be discretized in time by the two-level θ−scheme

$$m_i \frac{\mathrm{u}_i^{n+1} - \mathrm{u}_i^n}{\Delta t} = \theta \sum_{j \neq i} c_{ij}^{n+1} (\mathrm{u}_j^{n+1} - \mathrm{u}_i^{n+1}) + (1-\theta) \sum_{j \neq i} c_{ij}^n (\mathrm{u}_j^n - \mathrm{u}_i^n) \tag{3.1.31}$$

where superscripts refer to the old and new time levels t^n and $t^{n+1} = t^n + \Delta t$ and $\Delta t = t^{n+1} - t^n$ denotes the time step size. The explicit forward Euler method ($\theta = 0$), the semi-implicit Crank-Nicolson schemes ($\theta = \frac{1}{2}$) and the fully implicit backward Euler time-stepping ($\theta = 1$) is recovered by varying the implicitness parameter θ between zero and unity. The above equation can be written in matrix notation which yields an algebraic system of the form (3.1.29) with coefficients

$$a_{ij} = \begin{cases} -\theta \Delta t c_{ij}^{n+1} & \text{if } j \neq i \\ m_i - \theta \Delta t c_{ii}^{n+1} & \text{otherwise} \end{cases} \qquad b_{ij} = \begin{cases} (1-\theta) \Delta t c_{ij}^n & \text{if } j \neq i \\ m_i + (1-\theta) \Delta t c_{ii}^n & \text{otherwise} \end{cases} \tag{3.1.32}$$

Suppose that the semi-discrete counterpart of equation (3.1.31) satisfies the LED criterion [114] (cf. Theorem 3.1.2), that is, all off-diagonal coefficients c_{ij} are non-negative so that the corresponding entries of both matrices A and B have the correct sign. The constraint $b_{ii} \geq 0$ for the diagonal entries of B yields a computable upper bound for admissible time steps [144]:

$$\min_i \{ m_i + (1-\theta) \Delta t c_{ii}^n \} \geq 0, \qquad 0 \leq \theta < 1 \tag{3.1.33}$$

Matrix B is unconditionally non-negative for $\theta = 1$ and/or if all diagonal entries c_{ii}^{n+1} are non-negative. Otherwise, the above time step criterion yields the following upper bound [141]

$$\Delta t_{\max}^B \leq \frac{-m_i}{(1-\theta) c_{ii}^{n+1}}, \qquad c_{ii}^{n+1} < 0, \quad 0 \leq \theta < 1 \tag{3.1.34}$$

It remains to verify that A is an M-matrix, that is, it is non-singular and exhibits a non-negative inverse. Definition 3.1.4 does not provide useful criteria that can be easily checked in practice, and therefore, equivalent characterizations of M-matrices are mandatory. A common approach is to impose the following sign conditions on the matrix coefficients (cf. Hackbusch [87], p. 49)

$$a_{ii} > 0, \qquad a_{ij} \leq 0, \quad \forall i, \forall j \neq i \tag{3.1.35}$$

and require that A be strictly diagonally dominant, that is

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad \forall i \tag{3.1.36}$$

Then it is possible to prove that A is an M-matrix, and hence, it is invertible, whereby all entries of its inverse are non-negative (see Theorem 4.3.10 in Hackbusch [87], p. 53).

All off-diagonal coefficients of matrix A as defined in (3.1.32) are non-negative, since $c_{ij}^{n+1} \geq 0$ is required by the LED constraint 3.1.2. In the case of explicit time-stepping schemes ($\theta = 0$) and/or if the coefficient matrix $C^{n+1} = \{c_{ij}^{n+1}\}$ features the zero row-sum property (3.1.25)

$$a_{ii} = m_i > 0 \qquad \text{or} \qquad a_{ii} = m_i + \theta \Delta t \sum_{j \neq i} c_{ij}^{n+1} > 0, \quad \forall i \tag{3.1.37}$$

the sign condition (3.1.35) is unconditionally satisfied. Otherwise, the largest admissible time step is bounded from above if some diagonal coefficients c_{ii}^{n+1} are strictly positive:

$$\Delta t_{\max}^{A} < \frac{m_i}{\theta c_{ii}^{n+1}}, \qquad c_{ii}^{n+1} > 0, \quad 0 < \theta \leq 1 \tag{3.1.38}$$

No restriction on the time step is required if the diagonal of matrix C^{n+1} has only negative entries.

Condition (3.1.36) can be weakened by requiring that A be diagonally dominant, that is

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|, \quad \forall i \tag{3.1.39}$$

Then the sign condition (3.1.35) suffices to guarantee the M-matrix property of A provided that the strict inequality in expression (3.1.39) holds for at least one row index i, and moreover, matrix A must be irreducible (see Theorem 4.3.10 in Hackbusch [87], p. 53). In particular, its inverse is not only non-negative but one can even show that $A^{-1} > 0$ (see Theorem 4.3.11 in Hackbusch [87]).

The reducibility of $A \in \mathbb{R}^{n \times n}$ states that there exists a permutation matrix $P \in \mathbb{R}^{n \times n}$ such that

$$PAP^T = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \tag{3.1.40}$$

where A_{11} and A_{22} are submatrices with dimensions $r \times r$, $1 \leq r < n$ and $(n-r) \times (n-r)$, respectively. If no such permutation exists, then A is irreducible [257]. A more practical definition (see Definition 4.3.2 in Hackbusch [87], p. 49) states that $A = \{a_{ij}\}$ is irreducible if each two indices i and j from the set $\{1, 2, \ldots, n\}$ are connected, that is, there exists a 'chain' of direct links

$$i = \alpha_0, \alpha_1, \ldots, \alpha_K = j \quad \text{such that} \quad a_{\alpha_{k-1}\alpha_k} \neq 0, \qquad 1 \leq k \leq K \tag{3.1.41}$$

Loosely speaking, the sparsity graph of matrix A is strongly connected [257] so that it is impossible to single out submatrices which can be treated independently from the rest of the matrix A.

Remark. Some matrix manipulations to be perform in the course of discrete upwinding (see Section 3.2.4) can lead to zero rows in the discrete transport operator [131] so that the irreducibility property and/or the (strict) diagonal dominance may be lost. It is therefore essential to apply the positive contributions of the lumped mass matrix to the diagonal coefficients of the system matrix which can be accomplished by resorting to pseudo time-stepping.

Practical aspects of M-matrices

It is worthwhile to address some properties of M-matrices which may be useful for the implementation of iterative solution strategies for nonlinear (systems of) equations. Consider, say, the m^{th} outer iteration for a generic variable x which gives rise to the following system of equations:

$$Ax^{(m)} = r \tag{3.1.42}$$

Here, A may be evaluated from the last iterate $x^{(m-1)}$ and r contains all terms that do not depend explicitly on $x^{(m)}$. Patankar [200] proposed an implicit under-relaxation strategy of the form

$$\tilde{A}x^{(m)} = \tilde{r} \tag{3.1.43}$$

which should increase the diagonal dominance of matrix $\tilde{A}$ and simplify the solution process. In essence, the main diagonal entries of A are scaled by some parameter $0 < \beta < 1$ whereas each entry of the modified vector $\tilde{r}$ exhibits some explicit contribution from the diagonal part

$$\tilde{a}_{ii} = \frac{a_{ii}}{\beta}, \qquad \tilde{r}_i = r_i + \frac{1-\beta}{\beta} a_{ii} x_i^{(m-1)} \tag{3.1.44}$$

Once the outer iteration scheme (3.1.43) has converged, the terms involving parameter β cancel out so that the solution to the original problem (3.1.42) is recovered. The modifications applied to the diagonal coefficients of A can be equivalently stated as follows

$$\tilde{A} = A + D, \qquad D = \frac{1-\beta}{\beta}\,\mathrm{diag}A \tag{3.1.45}$$

where the diagonal matrix D is non-negative provided that all diagonal coefficients of matrix A exhibit this property. Suppose that A is an M-matrix so that matrix $\tilde{A} = A + D$ satisfies

$$\tilde{a}_{ij} \le 0, \quad \forall i, \forall j \neq i \qquad \text{and} \qquad \tilde{a}_{ij} \ge a_{ij}, \quad \forall i, \forall j \tag{3.1.46}$$

One can show that $\tilde{A}$ also features the M-matrix property 3.1.4, and moreover, $A^{-1} \ge \tilde{A}^{-1} \ge 0$ (see Theorem 2.5.4 in Horn and Johnson [106], p. 117). In short, an implicit under-relaxation strategy can be employed to enhance the diagonal dominance of the system matrix. For an M-matrix A the modified operator $\tilde{A} = A + D$ inherits this amenable property so that its inverse is non-negative.

Summary

Let us briefly recall the main ideas presented in this section. The local extremum diminishing property extends the concept of total variation diminishing schemes to multidimensions. It does not rely on geometric conditions, and therefore, it can be applied to arbitrary discretizations on unstructured meshes. For semi-discrete schemes of the general form (3.1.26), the formation of non-physical undershoots and overshoots can be prevented by controlling the sign of off-diagonal coefficients. Such local extremum diminishing schemes remain positivity-preserving after the discretization in time if they can be represented as algebraic system of the form $A\mathrm{u}_h^{n+1} = B\mathrm{u}_h^n$, where A is an M-matrix and B has only non-negative entries. For the two-level θ-scheme this can be achieved by adjusting the time step size. On the one hand, matrix B is unconditionally non-negative for implicit schemes or if all diagonal entries of the operator C^n are non-negative. If this is not the case, then the largest admissible time step is subject to the upper bound (3.1.34). On the other hand, the M-matrix property of A is unconditionally satisfied for fully explicit time-stepping schemes and/or if the coefficient matrix C^{n+1} has zero row sums. In the presence of non-vanishing row sums with positive diagonal coefficients $c_{ii}^{n+1} > 0$ the time step should satisfy estimate (3.1.38) to guarantee that A is an M-matrix. In the worst case, the largest admissible time step is given by $\Delta t_{\max} = \min\{\Delta t_{\max}^A, \Delta t_{\max}^B\}$ so as to fulfill the requirements of the positivity constraint 3.1.3. Furthermore, the use of an implicit under-relaxation strategy can be justified theoretically if the iteration matrix features the M-matrix property which carries over to the relaxed operator.

3.2. High-resolution finite element schemes

The development of reliable discretization techniques for convection dominated flows is a challenging task. A variety of stabilization techniques have been presented in the literature so as to combat the formation of non-physical oscillations which would be generated otherwise. The advent of nonlinear high-resolution schemes traces its origins to the *flux-corrected transport* (FCT) methodology introduced in the early 1970s by Boris and Book [39]. The fully multidimensional generalization proposed by Zalesak [268] has formed a very general framework for the design of FCT algorithms by representing them as a blend of linear high- and low-order approximations. Unlike other flux/slope limiting techniques, which are typically based on geometric design criteria, flux correction of FCT type is readily applicable to finite element discretizations on unstructured meshes [165, 167]. A retrospective and a comprehensive summary of the state of the art can be found in the literature [138]. Moreover, the forthcoming monograph by Kuzmin [132] will cover many aspects of flux correction schemes including a presentation of recent research activities.

In addition, *total variation diminishing* (TVD) schemes were introduced in the realm of finite differences [92, 93] and extended to finite volume/finite element schemes [9, 172, 174, 231, 232]. The current trend in the unstructured grid community is to use finite volume upwinding [74], residual distribution/fluctuation splitting [47, 61] or discontinuous Galerkin methods [52, 54]. The design philosophy behind modern front-capturing methods involves a set of physical or mathematical constraints to be imposed on the discrete solution so as to prevent the formation of spurious undershoots and overshoots in the vicinity of steep gradients. To this end, the following algorithmic components are to be specified and integrated into the flow solver [139, 269]:

- A high-order approximation which may fail to possess the desired properties;
- A low-order approximation which does enjoy these properties but is less accurate;
- A way to decompose the difference between the above into a sum of skew-symmetric internodal fluxes which can be manipulated without violating mass conservation;
- A cost-effective mechanism for adjusting these antidiffusive fluxes in an adaptive fashion so that the imposed constraints are satisfied for a given solution.

In the context of the *algebraic flux correction* (AFC) paradigm devised by Kuzmin and Turek [144] the above issues are treated by imposing mathematical constraints on discrete operators to achieve certain favorable matrix properties. Remarkably, all necessary information is provided by the matrix itself, so that high-resolution schemes can be constructed as 'black-box' tools applicable to arbitrary discretizations. The remainder of this section which is mainly based on the research articles by Kuzmin and Turek [144] and Kuzmin et al. [141] is concerned with the first three items, namely the construction of semi-discrete high- and low-order finite element schemes and the derivation of a conservative flux decomposition. The topic of algebraic flux correction is postponed to Sections 3.3 and 3.4 which present node-based flux limiters of TVD and FCT type.

3.2.1. Finite element discretization

As an introductory model problem, consider the time-dependent continuity equation

$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v}u) = 0 \qquad \text{in} \quad \mathbb{R}^{N_D} \times \mathbb{R}_0^+ \tag{3.2.1}$$

which represents a mass conservation law for a scalar quantity u. The non-uniform velocity field $\mathbf{v} = \mathbf{v}(\mathbf{x},t)$ is assumed to be known analytically or computed numerically from a momentum equation solved in a parallel way. For the time being, neglect boundary conditions and let the initial data be given by $u(\mathbf{x},0) = u_0(\mathbf{x})$. The variational formulation is obtained by multiplying the strong form (3.2.1) by the weighting function $w \in \mathcal{W} := \{w \in \mathcal{H}^1(\Omega)\}$ and integrating over the domain

$$\int_\Omega w \left[\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v}u) \right] \mathrm{d}\mathbf{x} = 0 \tag{3.2.2}$$

The weighting functions which do not depend on time are considered in the Sobolev space $\mathcal{H}^1(\Omega)$, that is, the set of square integrable functions for which all first derivatives are also square integrable. The solution of the weak form (3.2.2) is sought in the space of trial functions [64]

$$\mathcal{S}^t := \{u : u(\cdot,t) \in \mathcal{H}^1(\Omega), t \in [0,T]\} \tag{3.2.3}$$

which varies as a function of time. This separation of the spatial approximation from the discretization in time is particularly useful for the treatment of time-dependent problems.

The spaces $\mathcal{W}$ and $\mathcal{S}^t$ are spanned by infinitely many functions, and thus, they are inappropriate for the numerical treatment. In the Galerkin formulation, the space of test functions $\mathcal{W}$ is approximated by the finite dimensional subspace $\mathcal{W}_h := \text{span}\{\varphi_1, \ldots, \varphi_M\}$, where φ_i denotes the shape function associated with node i. A second finite dimensional subspace $\mathcal{S}_h^t \subset \mathcal{S}^t$ is used to approximate the space of trial functions [64]. The basic ideas of the finite element method as well as some properties of interpolation functions are addressed in Appendix A. A more comprehensive presentation of finite elements applied to flow problems is given by Donea and Huerta [64].

A common practice in finite element methods for conservation laws is to interpolate the convective fluxes in the same way as the numerical solution, as proposed by Fletcher [78]:

$$\mathrm{u}_h(\mathbf{x},t) = \sum_{j=1}^{M} \mathrm{u}_j(t)\varphi_j(\mathbf{x}), \qquad (\mathbf{v}\mathrm{u})_h = \sum_{j=1}^{M} (\mathbf{v}_j \mathrm{u}_j(t))\varphi_j(\mathbf{x}) \tag{3.2.4}$$

Substitution of the above relations into (3.2.2) yields the following semi-discrete problem

$$\sum_{j=1}^{M} \left[\int_\Omega \varphi_i \varphi_j \,\mathrm{d}\mathbf{x} \right] \frac{\mathrm{d}\mathrm{u}_j}{\mathrm{d}t} + \sum_{j=1}^{M} \left[\int_\Omega \varphi_i \mathbf{v}_j \cdot \nabla\varphi_j \,\mathrm{d}\mathbf{x} \right] \mathrm{u}_j = 0 \tag{3.2.5}$$

This gives a system of ordinary differential equations for the nodal values of the approximate solution which can be compactly written in matrix form as follows [144]:

$$M_C \frac{\mathrm{d}\mathrm{u}}{\mathrm{d}t} = K\mathrm{u} \tag{3.2.6}$$

Here, $M_C = \{m_{ij}\}$ denotes the consistent mass matrix and $K = \{k_{ij}\}$ is the discrete transport operator. The corresponding matrix entries are given by the following relations

$$m_{ij} = \int_\Omega \varphi_i \varphi_j \,\mathrm{d}\mathbf{x}, \qquad k_{ij} = -\mathbf{v}_j \cdot \mathbf{c}_{ij}, \qquad \mathbf{c}_{ij} = \int_\Omega \varphi_i \nabla\varphi_j \,\mathrm{d}\mathbf{x} \tag{3.2.7}$$

The coefficients m_{ij} and $\mathbf{c}_{ij}$ remain unchanged as long as the mesh is fixed, and hence, they can be computed once at the beginning of the simulation and each time the grid has been modified. As a result, all entries k_{ij} may be computed from formula (3.2.7) so that matrix K can be efficiently assembled without resorting to costly numerical integration. The operator $\mathbf{C} = \{\mathbf{c}_{ij}\}$ corresponds to the discretized space derivatives, and it has zero row sums, i.e. $\sum_j \mathbf{c}_{ij} = 0$, as long as the sum of basis functions φ_j equals one at every point (see properties (A.7) and (A.8) in Appendix A).

The reader who is interested only in the continuity equation (3.2.1) may directly proceed to Section 3.2.4 which presents an algebraic strategy [144] to render the semi-discrete high-order scheme (3.2.6) local extremum diminishing. The remainder of this section deals with the treatment of boundary conditions in the framework of variational methods. Moreover, standard techniques to approximate general flux functions are presented. A conservative flux decomposition is reviewed, which is applicable to general finite element discretizations on arbitrary meshes [141].

3.2.2. Treatment of boundary conditions

The time-dependent continuity equation (3.2.1) considered in the previous paragraph is a special case ($\mathbf{f}(u) = \mathbf{v}u$) of the following conservation law for a scalar quantity u

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0 \qquad \text{in} \quad \mathbb{R}^{N_\mathrm{D}} \times \mathbb{R}_0^+ \tag{3.2.8}$$

where $\mathbf{f}(u) : \mathbb{R} \to \mathbb{R}$ denotes a general flux function. Again, the variational formulation is obtained by integrating the weighted residual over the domain Ω and setting the result equal to zero

$$\int_\Omega w \left[\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) \right] \mathrm{d}\mathbf{x} = 0 \tag{3.2.9}$$

Example 3.2.1. For scalar conservation laws of the form (3.2.8), one frequently makes use of so-called *flux boundary conditions* which are only prescribed at the 'inlet' of the boundary [64]:

$$\mathbf{f}(u)\cdot\mathbf{n} = b \qquad \text{on} \quad \Gamma_{\text{in}} \tag{3.2.10}$$

For the pure convection equation, i.e. $\mathbf{f}(u) = \mathbf{v}u$, the inflow part is given by $\Gamma_{\text{in}} = \{\mathbf{x}\in\Gamma : \mathbf{v}\cdot\mathbf{n} < 0\}$. The rest of the boundary $\Gamma = \overline{\Gamma_{\text{in}}\cup\Gamma_{\text{out}}}$ is referred to as 'outlet' $\Gamma_{\text{out}} = \{\mathbf{x}\in\Gamma : \mathbf{v}\cdot\mathbf{n}\geq 0\}$, where no boundary conditions are specified. It is common practice to perform integration by parts applied to the divergence of the flux function in the weak formulation (3.2.9), that is

$$\int_\Omega w\frac{\partial u}{\partial t}\,\mathrm{d}\mathbf{x} - \int_\Omega \nabla w\cdot\mathbf{f}(u)\,\mathrm{d}\mathbf{x} + \int_\Gamma w\,\mathbf{f}(u)\cdot\mathbf{n}\,\mathrm{d}s = 0 \tag{3.2.11}$$

Flux boundary conditions can be included by replacing the boundary integral by $\int_{\Gamma_{\text{in}}} w\,b\,\mathrm{d}s$.

Example 3.2.2. As a simple example for a linear model problem, consider the flux function

$$\mathbf{f}(u) = \mathbf{v}u - d\nabla u \tag{3.2.12}$$

where $\mathbf{v} = \mathbf{v}(\mathbf{x},t)$ denotes the velocity field introduced above. For a non-vanishing diffusion coefficient $d > 0$, expression (3.2.8) yields the time-dependent convection-diffusion equation which is of parabolic type (see Section 2.1.3). Hence, the boundary $\Gamma = \overline{\Gamma_{\text{D}}\cup\Gamma_{\text{N}}}$ consists of a portion Γ_{N} on which the diffusive flux is prescribed and a complementary part Γ_{D} on which the value of the solution is set directly. The first type of boundary conditions is known as Neumann (or natural) conditions whereas subscript D refers to the Dirichlet (or essential) boundary conditions [64]:

$$\begin{aligned} u &= u_{\text{D}} \qquad &&\text{on} \quad \Gamma_{\text{D}} \\ d\nabla u\cdot\mathbf{n} &= b \qquad &&\text{on} \quad \Gamma_{\text{N}} \end{aligned} \tag{3.2.13}$$

Here, $\mathbf{n}$ denotes the outward unit normal vector. Using the divergence theorem to perform integration by parts for the diffusive flux in the weak form, we obtain the integral relation

$$\int_\Omega w\frac{\partial u}{\partial t}\,\mathrm{d}\mathbf{x} + \int_\Omega w\nabla\cdot(\mathbf{v}u)\,\mathrm{d}\mathbf{x} + \int_\Omega \nabla w\cdot(d\nabla u)\,\mathrm{d}\mathbf{x} - \int_\Gamma w\,d\nabla u\cdot\mathbf{n}\,\mathrm{d}s = 0 \tag{3.2.14}$$

so that Neumann boundary conditions can be included by replacing the last integral by $\int_{\Gamma_{\text{N}}} w\,b\,\mathrm{d}s$.

A *classical solution* to the strong form (3.2.8) needs to be twice continuously differentiable, that is, it must satisfy $u\in\mathcal{C}^2(\Omega)\cap\mathcal{C}^0(\bar{\Omega})$. This regularity condition is considerably relaxed in the weak form (3.2.14) since integration by parts moves one differentiation to the weighting function $w\in\mathcal{H}^1(\Omega)$ so that $u\in\mathcal{H}^1(\Omega)$ is sufficient [64]. Classical solutions of the strong form (3.2.8) will also satisfy (3.2.14). In contrast, the integral equation does not account for Dirichlet boundary conditions which need to be explicitly built into the space of trial functions $\mathcal{S}_h^t$. Moreover, the weighting functions are assumed to vanish at the boundary, that is, $w\in\mathcal{H}_0^1(\Omega)$.

3.2.3. Approximation of the flux function

Let us briefly discuss several techniques to approximate the flux function $\mathbf{f}(u)$ present in the general conservation law (3.2.8). The simplest choice is to adopt a constant flux for each element [64]

$$\mathbf{f}_h(\mathbf{x},t) \simeq \mathbf{f}(\bar{\mathrm{u}}_h) \tag{3.2.15}$$

where $\bar{\mathrm{u}}_h$ denotes an average of the solution values. Another option is to interpolate the approximate solution u_h at the desired point $(\mathbf{x},t)$ and use it to evaluate the flux as follows [64]

$$\mathbf{f}_h(\mathbf{x},t) \simeq \mathbf{f}\left(\sum_{j=1}^{M}\mathrm{u}_j(t)\varphi_j(\mathbf{x})\right) \tag{3.2.16}$$

This approach is commonly employed in the one-step Lax-Wendroff method, whereby the flux is evaluated at the Gauss points once the nodal solution values are available. As an alternative, the flux can be interpolated in the same way as the numerical solution which is especially useful for the numerical simulation of flow problems based on finite element approximations

$$\mathbf{f}_h(\mathbf{x},t) \simeq \sum_{j=1}^{M} \mathbf{f}_j(t)\varphi_j(\mathbf{x}) \tag{3.2.17}$$

This option was promoted by Fletcher [78] who called it the *group finite element formulation*. It provides several advantages over the two-step approach inherent to relation (3.2.16). First, it simplifies the evaluation of the flux divergence since the spatial dependence of $\mathbf{f}_h(\mathbf{x},t)$ is directly built into the ansatz functions. Moreover it was found to provide an efficient treatment of nonlinear convective terms and even lead to a small gain of accuracy for the two-dimensional Burgers equation discretized by linear finite elements on a uniform grid [78]. The group formulation is used throughout this work to approximate the flux function unless indicated otherwise.

Substitution of the group representation (3.2.17) into the weak form (3.2.9) yields an ordinary differential equation for each nodal value of the approximate solution

$$\sum_{j=1}^{M} \left[\int_\Omega \varphi_i\varphi_j \,\mathrm{d}\mathbf{x}\right] \frac{\mathrm{d}u_j}{\mathrm{d}t} + \sum_{j=1}^{M} \left[\int_\Omega \varphi_i \nabla\varphi_j \,\mathrm{d}\mathbf{x}\right] \cdot \mathbf{f}_j = 0 \tag{3.2.18}$$

It is commonly known that the Galerkin finite element method is globally conservative [90]. This can be verified by summing the above equation over all nodes $i \in \{1,\ldots,M\}$. Taking into account that the basis functions sum to unity (consistency property, see equation (A.7) in Appendix A), this yields the integral form (3.2.9) of the conservation law which states that mass is conserved.

In the finite element discretization of divergence terms, there exists no natural decomposition into a sum of numerical fluxes from one node into the other. Since most high-resolution schemes operate with such fluxes, their extension to finite elements proved to be a difficult task. Peraire et al. [204] demonstrated that a conservative flux decomposition is feasible for piecewise linear finite elements on simplex meshes. A similar technique was proposed by Barth [21, 22] who investigated the relationship between finite element and finite volume discretizations.

A viable flux decomposition techniques was developed by Kuzmin et al. [141] which is applicable to general finite element approximations on arbitrary meshes. It makes use of the divergence theorem to perform integration by parts in the semi-discrete formulation (3.2.18)

$$\sum_{j=1}^{M} \left[\int_\Omega \varphi_i\varphi_j \,\mathrm{d}\mathbf{x}\right] \frac{\mathrm{d}u_j}{\mathrm{d}t} - \sum_{j=1}^{M} \left[\int_\Omega \nabla\varphi_i\varphi_j \,\mathrm{d}\mathbf{x} - \int_\Gamma \varphi_i\varphi_j \,\mathbf{n}\,\mathrm{d}s\right] \cdot \mathbf{f}_j = 0 \tag{3.2.19}$$

Of course, the same relation is obtained if the group representation of the fluxes (3.2.17) is used in the weak formulation (3.2.11). The volume integral that results from the discretization of spatial derivatives can be expressed by the auxiliary coefficient $\mathbf{c}_{ij}$ introduced in (3.2.7) [139]

$$\left[\int_\Omega \nabla\varphi_i\varphi_j \,\mathrm{d}\mathbf{x}\right] \cdot \mathbf{f}_j = \mathbf{c}_{ji} \cdot \mathbf{f}_j, \qquad \mathbf{c}_{ij} = \int_\Omega \varphi_i \nabla\varphi_j \,\mathrm{d}\mathbf{x} \tag{3.2.20}$$

Owing to the zero row sum property (cf. equation (A.8) in Appendix A) of the coefficient matrix $\mathbf{C} = \{\mathbf{c}_{ij}\}$ it is possible to express each diagonal entry in terms of off-diagonal ones

$$\sum_{j=1}^{M} \mathbf{c}_{ij} = 0 \quad \Rightarrow \quad \mathbf{c}_{ii} = -\sum_{j\neq i} \mathbf{c}_{ij} \tag{3.2.21}$$

Consequently, equation (3.2.19) can be cast into the following form [141]

$$\sum_{j=1}^{M}\left[m_{ij}\frac{\mathrm{d}u_j}{\mathrm{d}t}+\mathbf{s}_{ij}\cdot\mathbf{f}_j\right]+\sum_{j=1}^{M}g_{ij}=0 \tag{3.2.22}$$

where the entries of the mass matrices for the volume and surface triangulation are given by

$$m_{ij}=\int_{\Omega}\varphi_i\varphi_j\,\mathrm{d}\mathbf{x},\qquad \mathbf{s}_{ij}=\int_{\Gamma}\varphi_i\varphi_j\,\mathbf{n}\,\mathrm{d}s \tag{3.2.23}$$

The interior part of the discretized divergence term is assembled from the *Galerkin fluxes*

$$g_{ij}:=\mathbf{c}_{ij}\cdot\mathbf{f}_i-\mathbf{c}_{ji}\cdot\mathbf{f}_j,\qquad g_{ji}=-g_{ij} \tag{3.2.24}$$

which are associated with the edges of the sparsity pattern [141]. These skew-symmetric quantities are responsible for the bilateral mass exchange between two neighboring nodes i and j, whereby no mass is created or destroyed artificially in the interior of the domain. The total amount of the scalar quantity u is conserved in the interior and it may only change due to the external feed $\mathbf{s}_{ij}\cdot\mathbf{f}_j$ which is equal to zero unless the basis functions for both nodes are non-vanishing at the boundary.

The above flux decomposition fills the gap between the specialized data structure proposed by Peraire et al. [204] and arbitrary Galerkin discretizations. Indeed, upwinding techniques and high-resolution finite element schemes based on slope limiters which were originally developed for piecewise linear approximations on triangular meshes can be extended to more general grids [163, 172, 174, 189]. In general, flux correction algorithms try to recover the high accuracy of the underlying discretization (e.g., Galerkin/Taylor-Galerkin FEM) for smooth solutions, and resort to a non-oscillatory low-order approximation in the vicinity of steep gradients. First-order accurate upwind methods which satisfy the local extremum diminishing (LED) criterion (3.1.26) have been available for finite difference and finite volume discretizations for a long time. On the other hand, the construction of such least diffusive linear LED schemes in the framework of finite elements has been a challenging task. A viable strategy for the derivation of upwind-biased finite element methods is to adopt finite volumes for the approximation of convective terms [9, 11, 255], whereas the used of standard streamline-diffusion techniques such as SUPG does not ensure monotonicity-preservation. Kuzmin [133] and Kuzmin and Turek [144] proposed an algebraic approach for the design of 'monotone' low-order methods which is presented in the next section.

3.2.4. Semi-discrete low-order scheme

Let the discretization in space be performed by a high-order finite element (Galerkin or Taylor-Galerkin) method which yields a DAE system for the vector of time-dependent nodal values [144]

$$M_C\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}=K\mathbf{u} \tag{3.2.25}$$

where $M_C=\{m_{ij}\}$ denotes the consistent mass matrix and $K=\{k_{ij}\}$ is the discrete transport operator. The latter may contain some streamline diffusion used to stabilize the convective contribution and/or to achieve better phase accuracy in the framework of Taylor-Galerkin methods [65].

It is well known that even stabilized high-order methods may produce non-physical undershoots and overshoots in the vicinity of steep gradients. As a remedy, Jameson introduced the concept of local extremum diminishing schemes (3.1.26) which impose mathematical constraints on the sign of the off-diagonal coefficients to prevent the formation of spurious oscillations. For linear finite element discretizations of the form (3.2.25), the LED property can be readily enforced by means of *discrete upwinding* [133, 144]. The required matrix manipulations are as follows:

- Replace the consistent mass matrix M_C by its row-sum lumped counterpart $M_L = \mathrm{diag}\{m_i\}$.
- Render the operator K local extremum diminishing by adding an artificial diffusion operator $D = \{d_{ij}\}$ so as to eliminate all negative off-diagonal coefficients.

This straightforward 'post-processing' transforms (3.2.25) into its linear LED counterpart

$$M_L \frac{\mathrm{du}}{\mathrm{d}t} = L\mathrm{u}, \qquad L = K + D \tag{3.2.26}$$

As stated above the Galerkin FEM is globally conservative [90] and mass is neither created nor destroyed if row-sum mass lumping is employed, i.e. $m_i := \sum_j m_{ij} > 0$. To ensure the conservation property after the elimination of negative off-diagonal entries, Kuzmin and Turek [144] defined the diffusion operator D as a symmetric matrix with zero row and column sums

$$\sum_{i=1}^{M} d_{ij} = \sum_{j=1}^{M} d_{ij} = 0 \tag{3.2.27}$$

As a result, the application of the discrete operator D to the vector of nodal values u yields

$$(D\mathrm{u})_i = \sum_{j=1} d_{ij}\mathrm{u}_j = \sum_{j \neq i} d_{ij}(\mathrm{u}_j - \mathrm{u}_i) \tag{3.2.28}$$

Hence, the contribution of Du to node i can be decomposed into a sum of skew-symmetric internodal fluxes which are associated with the edges of the sparsity graph [144] (see also Appendix B)

$$(D\mathrm{u})_i = \sum_{j \neq i} f_{ij}^d, \qquad f_{ij}^d = d_{ij}(\mathrm{u}_j - \mathrm{u}_i) = -f_{ji}^d \tag{3.2.29}$$

In particular, there exists an *edge* between node i and node j if and only if their basis functions have overlapping supports, that is, $(\varphi_i, \varphi_j) \neq 0$, whereby $(\cdot,\cdot)$ denotes the standard inner product. In fact, this representation is valid for all discrete diffusion operators [144], whereby the flux f_{ij}^d from node j into node i is proportional to the difference between the nodal values. Depending on the sign of the coefficient d_{ij}, it is of diffusive or antidiffusive nature and leads to flattening ($d_{ij} > 0$) and steepening ($d_{ij} < 0$) of solution profiles, respectively. In any case, the symmetry of the matrix D implies that $f_{ji}^d = -f_{ij}^d$ so that there is no loss or gain of mass.

In order to eliminate all negative off-diagonal coefficients from the high-order transport operator K, the optimal choice for the artificial diffusion coefficient d_{ij} for the edge ij reads [133, 144]

$$d_{ij} = \max\{-k_{ij}, 0, -k_{ji}\} = d_{ji} \tag{3.2.30}$$

Consequently, the low-order operator $L = \{l_{ij}\}$ exhibits the LED property (3.1.26) since all off-diagonal coefficients $l_{ij} := k_{ij} + d_{ij}$ and $l_{ji} := k_{ji} + d_{ij}$ are non-negative by construction. Owing to the zero row sum property of the diffusion operator D, the diagonal coefficients of L are given by

$$l_{ii} := k_{ii} - \sum_{j \neq i} d_{ij} \tag{3.2.31}$$

Kuzmin and Turek [145] suggested the following orientation convention for the edge ij: Without loss of generality, let $\vec{ij}$ denote the *oriented edge* such that $0 \leq l_{ij} \leq l_{ji} = k_{ji} + d_{ij}$, which implies that node i is located 'upwind' and corresponds to the row number of the eliminated negative coefficient. From now on, an (unoriented) edge between nodes i and j will be denoted by ij and the notation $\vec{ij}$ implies that the above orientation convention has been applied.

The semi-discretized equation for the nodal solution value $\mathrm{u}_i(t)$ reads as follows

$$m_i \frac{\mathrm{du}_i}{\mathrm{d}t} = \sum_{j \neq i} l_{ij}(\mathrm{u}_j - \mathrm{u}_i) + \mathrm{u}_i \sum_{j=1}^{M} l_{ij} \tag{3.2.32}$$

where $m_i = \sum_j m_{ij} > 0$ denotes the corresponding entry of the lumped mass matrix and all off-diagonal coefficients l_{ij} are non-negative by construction. Owing to the zero row sum property of the discrete diffusion operator D, the last term in the above relation is not affected by discrete upwinding [145], i.e. $\mathrm{u}_i \sum_j l_{ij} = \mathrm{u}_i \sum_j k_{ij}$. For the continuity equation (3.2.1), the last term in the above expression represents a discrete counterpart of $-u\nabla \cdot \mathbf{v}$ which vanishes for divergence-free velocity fields and is responsible for a physical growth of local extrema otherwise [142, 145].

In a practical implementation, the transport operator can be initialized by $L := K$ and assembled edge-by-edge without constructing the global matrix D explicitly. Each pair of non-zero off-diagonal coefficients l_{ij} and l_{ji} is examined. If the smaller one is negative, it is set to zero and three other entries are modified so as to restore row/column sums [142, 145]:

$$\begin{aligned} l_{ii} &:= l_{ii} - d_{ij}, \qquad & l_{ij} &:= l_{ij} + d_{ij} \\ l_{ji} &:= l_{ji} + d_{ij}, \qquad & l_{jj} &:= l_{jj} - d_{ij} \end{aligned} \tag{3.2.33}$$

A general framework for the implementation of edge-based algorithms is presented in Appendix B which also addresses some common storage techniques convenient for sparse matrices.

Example 3.2.3 (Kuzmin and Turek [144])**.** To illustrate the algebraic construction procedure of the positivity-preserving low-order scheme, consider the one-dimensional convection equation

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0 \tag{3.2.34}$$

with constant velocity $v > 0$ and apply the lumped-mass Galerkin method for the discretization in space. If a piecewise linear approximation is employed on a uniform mesh of width Δx, the central difference discretization of the convective term is recovered at interior nodes

$$\frac{\mathrm{du}_i}{\mathrm{d}t} = v\frac{\mathrm{u}_{i-1} - \mathrm{u}_{i+1}}{2\Delta x} \tag{3.2.35}$$

The corresponding 2×2 *element matrices* are given by

$$\hat{M}_L = \frac{\Delta x}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \hat{K} = \frac{v}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \tag{3.2.36}$$

The negative coefficient in the upper-right corner of $\hat{K}$ violates the LED criterion (3.1.26) and should be eliminated in a conservative way. This can be done by applying the artificial diffusion operator $\hat{D}$ which is designed to be a symmetric matrix with zero row and column sums

$$\hat{D} = \frac{v}{2}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \Rightarrow \qquad \hat{L} = \hat{K} + \hat{D} = v\begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix} \tag{3.2.37}$$

After the global matrix assembly, this yields the standard upwind method

$$\frac{\mathrm{du}_i}{\mathrm{d}t} = v\frac{\mathrm{u}_{i-1} - \mathrm{u}_i}{\Delta x} \tag{3.2.38}$$

which is the least diffusive linear scheme that is local extremum diminishing.

Example 3.2.4. To illustrate the effect of *discrete upwinding* for a nonlinear conservation law, consider the one-dimensional inviscid Burgers equation in conservative form

$$\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0 \tag{3.2.39}$$

with the nonlinear flux function $f(u)=\frac{1}{2}u^2$. Let the discretization in space be performed by the lumped Galerkin method, whereby the group formulation (3.2.17) is adopted for the fluxes [78]. The use of a piecewise linear approximation on a uniform mesh of width Δx yields [79]

$$\frac{\mathrm{d}\mathrm{u}_i}{\mathrm{d}t}=\frac{1}{2}(\mathrm{u}_{i-1}+\mathrm{u}_{i+1})\frac{\mathrm{u}_{i-1}-\mathrm{u}_{i+1}}{2\Delta x} \tag{3.2.40}$$

whereby the numerical fluxes have been expressed in terms of the nodal solution values. The element matrices for the high- and low-order operators $\hat{K}$ and $\hat{L}=\hat{K}+\hat{D}$ are given by

$$\hat{K}=\frac{1}{2}\begin{bmatrix}\bar{\mathrm{u}} & -\bar{\mathrm{u}}\\ \bar{\mathrm{u}} & -\bar{\mathrm{u}}\end{bmatrix},\qquad \hat{L}=\frac{1}{2}\begin{bmatrix}\bar{\mathrm{u}}-|\bar{\mathrm{u}}| & -\bar{\mathrm{u}}+|\bar{\mathrm{u}}|\\ \bar{\mathrm{u}}+|\bar{\mathrm{u}}| & -\bar{\mathrm{u}}-|\bar{\mathrm{u}}|\end{bmatrix} \tag{3.2.41}$$

where $\bar{\mathrm{u}}$ denotes the standard average of nodal solution values on each element. Depending on the sign of $\bar{\mathrm{u}}$ either the upper or the lower row of $\hat{L}$ will have two zero entries

$$\hat{L}=\bar{u}\begin{bmatrix}0 & 0\\ 1 & -1\end{bmatrix}\quad\text{if}\quad \bar{u}>0,\qquad \hat{L}=-\bar{u}\begin{bmatrix}-1 & 1\\ 0 & 0\end{bmatrix}\quad\text{if}\quad \bar{u}<0 \tag{3.2.42}$$

which yields the following 'upwind' discretization for the one-dimensional Burgers

$$\frac{\mathrm{d}\mathrm{u}_i}{\mathrm{d}t}=\frac{1}{2}(\bar{\mathrm{u}}_L+|\bar{\mathrm{u}}_L|)\frac{\mathrm{u}_{i-1}-\mathrm{u}_i}{\Delta x}-\frac{1}{2}(\bar{\mathrm{u}}_R-|\bar{\mathrm{u}}_R|)\frac{\mathrm{u}_{i+1}-\mathrm{u}_i}{\Delta x} \tag{3.2.43}$$

where $\bar{\mathrm{u}}_L=0.5(\mathrm{u}_{i-1}+\mathrm{u}_i)$ and $\bar{\mathrm{u}}_R=0.5(\mathrm{u}_i+\mathrm{u}_{i+1})$ denote the averaged velocities in the left and right element neighbors, respectively. Concerning the LED criterion (3.1.26), both solution differences in the above formula are either applied with positive sign or canceled completely if the flow velocity has the 'wrong' direction. Note that the right-hand side of the low-order scheme (3.2.43) vanishes if both coefficients are eliminated. This can lead to an unphysical behavior engendered by discrete upwinding [131] which may require some kind of entropy-fix [157].

3.2.5. Conservative flux decomposition

The semi-discrete low-order scheme (3.2.26) is non-oscillatory but overly diffusive. Excessive artificial diffusion can be removed in regions, where the solution is sufficiently smooth, so that the high accuracy of the original method can be achieved without generating non-physical undershoots and overshoots. To this end, the difference between the residuals (3.2.25) and (3.2.26)

$$f=(M_L-M_C)\frac{\mathrm{du}}{\mathrm{d}t}-D\mathrm{u} \tag{3.2.44}$$

needs to be decomposed into sums of skew-symmetric internodal fluxes so that all further manipulations will preserve mass. The artificial diffusion operator D was designed to be a symmetric matrix with zero row and column sums. Moreover, the difference between the consistent mass matrix and its lumped counterpart exhibits the same properties [144]

$$M_C-M_L=\{d_{ij}^m\},\qquad d_{ij}^m=\int_\Omega \varphi_i(\varphi_j-\delta_{ij})\,\mathrm{d}\mathbf{x} \tag{3.2.45}$$

and thus, it is referred to as *mass diffusion operator*. In the above formula, δ_{ij} denotes the Kronecker delta symbol which equals 1 if $i = j$ and 0 otherwise. As a consequence, the contribution of (3.2.44) to node i can be decomposed into a sum of *raw antidiffusive fluxes* [139, 144]

$$f_i = \sum_{j \neq i} f_{ij}, \qquad f_{ij} = \left[m_{ij} \frac{\mathrm{d}}{\mathrm{d}t} + d_{ij} \right] (\mathrm{u}_i - \mathrm{u}_j) = -f_{ji} \tag{3.2.46}$$

The amount received by node i is subtracted from node j and vice versa so that no net loss or gain of mass can take place. In order to prevent the formation of non-physical local extrema, every antidiffusive flux f_{ij} is multiplied by some solution-dependent correction factor $\alpha_{ij} \in [0,1]$. The resulting limited antidiffusive fluxes are inserted into the right-hand side of the semi-discrete low-order scheme (3.2.32) so that the flux-corrected equation for node i reads [136]

$$m_i \frac{\mathrm{du}_i}{\mathrm{d}t} = \sum_{j \neq i} l_{ij} (\mathrm{u}_j - \mathrm{u}_i) + \mathrm{u}_i \sum_{j=1}^{M} l_{ij} + \bar{f}_i, \qquad \bar{f}_i = \sum_{j \neq i} \alpha_{ij} f_{ij} \tag{3.2.47}$$

By construction, the oscillatory high-order discretization (3.2.25) is recovered for $\alpha_{ij} \equiv 1$ whereas $\alpha_{ij} \equiv 0$ yields the local extremum diminishing counterpart (3.2.26) which is overly diffusive. For accuracy reasons, the correction factors should approach 1 in regions where the solution is sufficiently smooth. At the same time, some artificial diffusion is required in the vicinity of steep gradients so as to combat the formation of non-physical undershoots and overshoots which would be generated otherwise. The task of the flux limiter is to determine optimal correction factors α_{ij} so as to remove as much artificial diffusion as possible without generating spurious oscillations.

Over the years, a rich variety of limiting strategies has been proposed by Kuzmin [133, 134, 136], Kuzmin and Turek [144, 145], Kuzmin and Kourounis [137] and Kuzmin et al. [142]. The reader who is interested in a comprehensive survey of the entire algebraic flux correction (AFC) methodology is referred to publication [139] in the book by Kuzmin et al. [138] and to the forthcoming monograph by Kuzmin [132]. It is beyond the scope of this work to elucidate the ins and outs of *all* approaches suggested in the literature. We therefore present an assortment of flux-corrected algorithms that work well in practice. They exhibit a good compromise between ultimate accuracy and usability concerning ease of implementation and computational costs.

For strongly time-dependent flows, the recovery of the consistent mass matrix M_C that was replaced by M_L is desirable in order to improve the phase accuracy [33, 134]. A *symmetric limiting strategy* lends itself to the treatment of mass antidiffusion $(M_L - M_C)\frac{\mathrm{du}}{\mathrm{d}t}$ due to the lack of a pronounced flow direction. In contrast, *upwind-biased limiting strategies* are preferable for the treatment of convective antidiffusion. A comprehensive analysis of both approaches and a general-purpose flux limiter applicable to implicit finite element discretizations with consistent mass matrix is presented by Kuzmin [134]. The next section deals with upwind-biased limiters of TVD type, whereas the presentation of symmetric limiters of FCT type is postponed to Section 3.4.

3.3. Upwind-biased limiters of TVD type

For the time begin, assume that the problem at hand is stationary so that the contribution of the consistent mass matrix can be neglected. Hence, the raw antidiffusive flux (3.2.46) reduces to

$$f_{ij} = d_{ij} (\mathrm{u}_i - \mathrm{u}_j) = -f_{ji} \tag{3.3.1}$$

which equals the negative diffusive flux defined in relation (3.2.29), i.e. $f_{ij} = -f_{ij}^d$. Such fluxes violate the LED criterion (3.1.26) if they are of the form $f_{ij} = p_{ij}(\mathrm{u}_j - \mathrm{u}_i)$, where $p_{ij} = -d_{ij} < 0$. On the other hand, edge contributions with non-negative coefficients resemble diffusive fluxes and are harmless. The antidiffusive flux from node j into node i (or some portion of it) is also admissible if it can be interpreted as a diffusive flux from another node, say k. This is the case as long as there exist solution-dependent coefficients $q_{ik} \geq 0$ such that $f_{ij} = \sum_{k \neq i} q_{ik} (\mathrm{u}_k - \mathrm{u}_i)$ [134].

3.3.1. Node-based limiting strategy

This sufficient condition can be enforced resorting to a node-based upwind-biased limiting strategy which was proposed by Kuzmin and Turek [145] and later refined by Kuzmin [134, 135]. It is largely inspired by Zalesak's limiter [268], and hence, his notation is adopted to reflect this relationship. For each node, the net antidiffusion may consist of both positive and negative edge contributions, but in the worst case, all fluxes have the same sign. It is therefore advisable to limit them separately according to the following three-step algorithm [135]:

1. Compute the sums of positive and negative antidiffusive fluxes

$$P_i^+ = \sum_{l_{ij} \le l_{ji}} \max\{0, f_{ij}\}, \qquad P_i^- = \sum_{l_{ij} \le l_{ji}} \min\{0, f_{ij}\} \tag{3.3.2}$$

2. Define the upper and lower bounds to be imposed on the sums $P_i^\pm$

$$Q_i^+ = \sum_{j \ne i} \max\{0, -f_{ij}\}, \qquad Q_i^- = \sum_{j \ne i} \min\{0, -f_{ij}\} \tag{3.3.3}$$

3. Evaluate the nodal correction factors and perform flux limiting

$$R_i^\pm = \min\{1, Q_i^\pm / P_i^\pm\}, \qquad \alpha_{ij} = \begin{cases} R_i^+ & \text{if} \quad f_{ij} > 0 \\ R_i^- & \text{if} \quad f_{ij} \le 0 \end{cases} \tag{3.3.4}$$

Fig. 3.1. Upwind-biased flux limiting.

According to the orientation convention [145], the edges of the sparsity graph are oriented such that $l_{ij} \le l_{ji}$. Hence, the nodal quantity $P_i^\pm$ as defined in (3.3.2) accounts for the total amount of raw antidiffusion that node i receives from its downwind neighbors. As mentioned above, the special structure of the antidiffusive flux (3.3.1) implies that $-f_{ij} = f_{ij}^d$ so that the upper and lower bounds $Q_i^\pm$ can be readily defined in terms of diffusive edge contributions. It follows directly from the definition of the nodal correction factors (3.3.4) that $|R_i^\pm P_i^\pm| \le |Q_i^\pm|$ for the *upwind* node i. After flux limiting, the contribution of the edge $\vec{ij}$ to the *downwind* node j is given by [134]

$$l_{ji}(\mathrm{u}_i - \mathrm{u}_j) - \alpha_{ij} f_{ij} = (l_{ji} - \alpha_{ij} d_{ij})(\mathrm{u}_i - \mathrm{u}_j) \tag{3.3.5}$$

which also proves local extremum diminishing provided that the antidiffusion coefficient $-d_{ij}$ and the correction factor α_{ij} satisfy the inequality $l_{ji} - \alpha_{ij} d_{ij} \ge 0$. For the trivial case $d_{ij} = 0$ this is always true since $l_{ji} \ge l_{ij} \ge 0$ by construction. Let $d_{ij} > 0$ so that the above positivity constraint implies $k_{ji} + (1 - \alpha_{ij}) d_{ij} \ge 0$. This can be guaranteed by 'prelimiting' the flux according to [134]

$$f_{ij} = \min\{d_{ij}, l_{ji}\}(\mathrm{u}_i - \mathrm{u}_j) \tag{3.3.6}$$

before computing the sums $P_i^\pm$ and $Q_i^\pm$ in the above limiting procedure (see Figure 3.1). Of course, there is no need for prelimiting unless both off-diagonal coefficients of the high-order operator K are negative. In summary, algorithm (3.3.2)–(3.3.4) guarantees that there exist coefficients $q_{ij}^\pm \ge 0$ such that the sum of limited antidiffusive fluxes is bounded from below and above [134]

$$Q_i^- = \sum_{j \ne i} q_{ij}^-(\mathrm{u}_j - \mathrm{u}_i) \le \sum_{j \ne i} \alpha_{ij} f_{ij} \le \sum_{j \ne i} q_{ij}^+(\mathrm{u}_j - \mathrm{u}_i) = Q_i^+ \tag{3.3.7}$$

which proves that the resulting semi-discrete scheme is local extremum diminishing.

The upwind-biased limiting procedure described above leads to the nonlinear ODE system

$$M_L \frac{\mathrm{du}}{\mathrm{d}t} = L\mathrm{u} + \bar{f}(\mathrm{u}) \tag{3.3.8}$$

whereby the correction term $\bar{f}(\mathrm{u})$ is designed to improve the accuracy in regions of smooth solutions without reintroducing ripples in the vicinity of steep gradients. Each nodal component is given as a sum of individual edge contributions $\bar{f}_i = \sum_{j\neq i} \alpha_{ij} f_{ij}$ and the same limited flux $\alpha_{ij} f_{ij}$ is applied to node j but with opposite sign. It is instructive to define $F(u)$ as a symmetric matrix with zero row and column sums whose off-diagonal entries are given by $-\alpha_{ij} d_{ij}$ so that $\bar{f}$ can be rewritten in terms of a discrete (anti-)diffusion operator applied to the vector of nodal values

$$\bar{f}(\mathrm{u}) = F(\mathrm{u})\mathrm{u} \tag{3.3.9}$$

As a result, the high-resolution scheme (3.3.8) can be cast into compact matrix form [145]

$$M_L \frac{\mathrm{du}}{\mathrm{d}t} = K^*(\mathrm{u})\mathrm{u} \tag{3.3.10}$$

where the modified transport operator $K^*(\mathrm{u}) = \{k^*_{ij}\}$ exhibits the following structure

$$K^*(\mathrm{u}) = L + F(\mathrm{u}) = K + D + F(\mathrm{u}) \tag{3.3.11}$$

It is constructed from the original high-order operator K by adding artificial diffusion D and applying some compensating antidiffusion $F(\mathrm{u})$ so that its matrix coefficients are given by

$$k^*_{ii} = k_{ii} - \sum_{j\neq i} (1-\alpha_{ij}) d_{ij}, \qquad k^*_{ij} = k_{ij} + (1-\alpha_{ij}) d_{ij} \tag{3.3.12}$$

Of course, some off-diagonal coefficients may be **negative** and violate the LED criterion (3.1.26) at first glance. In contrast, the low-order scheme is local extremum diminishing since the diffusion matrix $D = \{d_{ij}\}$ is designed in such a way that $0 \leq k_{ij} + d_{ij} \leq k_{ji} + d_{ij}$, whereby the orientation convention implies that node i is located upwind and corresponds to the row number of the eliminated negative coefficient (if any). The two-sided estimate (3.3.7) states that there exist non-negative coefficients $q^{\pm}_{ij}$ such that the amount of limited antidiffusion can be interpreted as a sum of diffusive contributions. This suffices to ensure the existence of a matrix L^* which satisfies $K^*(\mathrm{u})\mathrm{u} = L^*(\mathrm{u})\mathrm{u}$ for a given solution u, whereby all off-diagonal coefficients l^*_{ij} are **non-negative**. Thus, the semi-discrete scheme (3.3.10) has an equivalent representation [145]

$$M_L \frac{\mathrm{du}}{\mathrm{d}t} = L^*(\mathrm{u})\mathrm{u} \tag{3.3.13}$$

which satisfies the LED constraint (3.1.26) by construction. In practice, it is not necessary to construct the modified transport operator L^* explicitly; its existence suffices to guarantee that the original high-resolution scheme (3.3.10) also satisfies the LED property (3.1.26). This equivalence will be exploited extensively in subsequent sections which deal with the discretization in time and with iterative solution techniques applicable to the linear and nonlinear algebraic systems.

Let us summarize what we have said so far and present a practical algorithm [135] for node-oriented flux correction based on the upwind-biased flux limiting procedure (3.3.2)–(3.3.4). The nodal quantities $\bar{f}_i$, $Q^{\pm}_i$, $P^{\pm}_i$ and $R^{\pm}_i$ are initialized by zeros and updated as illustrated in Figure 3.2.

For each pair of neighboring nodes i and j orient the edge $\vec{ij}$ so that $l_{ij} \le l_{ji}$.

In a loop over edges:

1. Compute the prelimited raw antidiffusive flux $f_{ij} = \min\{d_{ij}, l_{ji}\}(\mathrm{u}_i - \mathrm{u}_j)$ and apply it to sums of positive/negative fluxes at the upwind node i

$$P_i^+ := P_i^+ + \max\{0, f_{ij}\}, \qquad P_i^- := P_i^- + \min\{0, f_{ij}\} \tag{3.3.14}$$

2. Update the upper and lower bounds to be imposed on the sums $P_i^\pm$

$$\begin{aligned} Q_i^+ &:= Q_i^+ + \max\{0, -f_{ij}\}, \qquad Q_j^+ := Q_j^+ + \max\{0, f_{ij}\} \\ Q_i^- &:= Q_i^- + \min\{0, -f_{ij}\}, \qquad Q_j^- := Q_j^- + \min\{0, f_{ij}\} \end{aligned} \tag{3.3.15}$$

In a loop over nodes:

3. Evaluate the nodal correction factors to be applied

$$R_i^\pm = \min\{1, Q_i^\pm / P_i^\pm\} \tag{3.3.16}$$

In a loop over edges:

4. Check the sign of the antidiffusive flux f_{ij} and multiply the flux by the corresponding correction factor evaluated at the upwind node

$$\begin{aligned} \bar{f}_i &:= \bar{f}_i + \alpha_{ij} f_{ij} \\ \bar{f}_j &:= \bar{f}_j - \alpha_{ij} f_{ij} \end{aligned} \qquad \alpha_{ij} = \begin{cases} R_i^+ & \text{if} \quad f_{ij} > 0 \\ R_i^- & \text{if} \quad f_{ij} \le 0 \end{cases} \tag{3.3.17}$$

Fig. 3.2. Practical implementation of the generalized TVD algorithm.

Example 3.3.1. Let us illustrate the flux correction procedure for the one-dimensional linear advection equation $\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0$ with constant velocity $v > 0$. As explained in Example 3.2.3, the artificial diffusion coefficient equals $d_{ij} = v/2$ which yields the upwind discretization (3.2.38). Its accuracy can be improved by adding some portion of the raw antidiffusive flux $f_{ij} = d_{ij}(\mathrm{u}_i - \mathrm{u}_j)$, where $j = i+1$ corresponds to the downwind node. The nodal correction factor for the upwind node is given by $R_i^\pm = \min\{1, r_i^\pm\}$, where the smoothness indicator $r_i^\pm = Q_i^\pm / P_i^\pm$ reads

$$r_i^+ = \frac{\max\{0, \mathrm{u}_{i+1} - \mathrm{u}_i\} + \max\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\}}{\max\{0, \mathrm{u}_i - \mathrm{u}_{i+1}\}} = \frac{\max\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\}}{\mathrm{u}_i - \mathrm{u}_{i+1}} \quad \text{if} \quad f_{ij} > 0 \tag{3.3.18}$$

$$r_i^- = \frac{\min\{0, \mathrm{u}_{i+1} - \mathrm{u}_i\} + \min\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\}}{\min\{0, \mathrm{u}_i - \mathrm{u}_{i+1}\}} = \frac{\min\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\}}{\mathrm{u}_i - \mathrm{u}_{i+1}} \quad \text{if} \quad f_{ij} < 0 \tag{3.3.19}$$

As stated above, the correction factor is evaluated at the upwind node i, i.e. $\alpha_{ij} = R_i^\pm$, and more-

over, it depends on the sign of the antidiffusive flux from the downwind node $j = i+1$:

$$\alpha_{ij} f_{ij} = \begin{cases} \min\left\{1, \dfrac{\max\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\}}{\mathrm{u}_i - \mathrm{u}_{i+1}}\right\} d_{ij}(\mathrm{u}_i - \mathrm{u}_j) & \text{if} \quad f_{ij} > 0 \\ \min\left\{1, \dfrac{\min\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\}}{\mathrm{u}_i - \mathrm{u}_{i+1}}\right\} d_{ij}(\mathrm{u}_i - \mathrm{u}_j) & \text{if} \quad f_{ij} < 0 \end{cases} \tag{3.3.20}$$

Here, the trivial case $f_{ij} = 0$ can be safely neglected since it implies that $\mathrm{u}_i = \mathrm{u}_{i+1}$ and no correction takes place. It is easy to check that the limited antidiffusive flux can be equivalently written as

$$\alpha_{ij} f_{ij} = \max\{0, \min\{1, r_i\}\} d_{ij}(\mathrm{u}_i - \mathrm{u}_j) \tag{3.3.21}$$

where the unsigned smoothness indicator r_i is defined as the slope ratio evaluated at node i

$$r_i = \frac{\mathrm{u}_{i-1} - \mathrm{u}_i}{\mathrm{u}_i - \mathrm{u}_{i+1}} \tag{3.3.22}$$

The corrected flux (3.3.21) is also obtained from standard TVD schemes [92, 93] if the *minmod* limiter is employed [134]. It constitutes the lower bound of Sweby's second-order TVD region [245] and is the most diffusive limiter claimed to provide second-order accuracy. The relation between TVD schemes in general and the algebraic flux correction paradigm has been analyzed by Kuzmin and Turek [145] and Kuzmin [134]. In short, second-order accuracy can only be assured for a constant velocity v on a uniform mesh. The straightforward generalization of standard TVD limiters to multidimensional finite element discretizations on unstructured meshes may lead to polluted solutions in smooth regions since second-order accuracy can no longer be guaranteed.

Example 3.3.2. For the one-dimensional inviscid Burgers equation $\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0$, flux correction of TVD type is straightforward. The underlying low-order discretization (3.2.40) is constructed as explained in Example 3.2.4, whereby the group formulation is employed. For the time being let us assume that the solution remains non-negative so that the raw antidiffusive fluxes are of the form

$$\mathrm{u}_i \geq 0 \quad \forall i \qquad \Rightarrow \qquad f_{ij} = \frac{|\mathrm{u}_i + \mathrm{u}_j|}{2}(\mathrm{u}_i - \mathrm{u}_j) \tag{3.3.23}$$

where $j = i+1$ denotes the downwind node and the corresponding diffusion coefficient equals the modulus of averaged velocities $d_{ij} = 0.5|\mathrm{u}_i + \mathrm{u}_j|$. The nodal correction factor for the upwind node i is given by $R_i^{\pm} = \min\{1, r_i^{\pm}\}$ as in the linear case. The smoothness indicator $r_i^{\pm}$ needs to be redefined accordingly, whereby the sign of the flux implies that the denominator does not vanish

$$\begin{aligned} r_i^+ &= \frac{|\mathrm{u}_i + \mathrm{u}_{i-1}| \max\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\} + |\mathrm{u}_i + \mathrm{u}_{i+1}| \max\{0, \mathrm{u}_{i+1} - \mathrm{u}_i\}}{|\mathrm{u}_i + \mathrm{u}_{i+1}| \max\{0, \mathrm{u}_i - \mathrm{u}_{i+1}\}} \quad \text{if} \quad f_{ij} > 0 \\ r_i^- &= \frac{|\mathrm{u}_i + \mathrm{u}_{i-1}| \min\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\} + |\mathrm{u}_i + \mathrm{u}_{i+1}| \min\{0, \mathrm{u}_{i+1} - \mathrm{u}_i\}}{|\mathrm{u}_i + \mathrm{u}_{i+1}| \min\{0, \mathrm{u}_i - \mathrm{u}_{i+1}\}} \quad \text{if} \quad f_{ij} < 0 \end{aligned} \tag{3.3.24}$$

Since all nodal solution values are assumed to be non-negative the absolute values can be neglected. The correction factor $\alpha_{ij} = R_i^{\pm}$ is evaluated at the upwind node i which yields

$$\alpha_{ij} = \begin{cases} \min\left\{1, \dfrac{(\mathrm{u}_i + \mathrm{u}_{i-1}) \max\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\}}{(\mathrm{u}_i + \mathrm{u}_{i+1})(\mathrm{u}_i - \mathrm{u}_{i+1})}\right\} & \text{if} \quad f_{ij} > 0 \\ \min\left\{1, \dfrac{(\mathrm{u}_i + \mathrm{u}_{i-1}) \min\{0, \mathrm{u}_{i-1} - \mathrm{u}_i\}}{(\mathrm{u}_i + \mathrm{u}_{i+1})(\mathrm{u}_i - \mathrm{u}_{i+1})}\right\} & \text{if} \quad f_{ij} < 0 \end{cases} \tag{3.3.25}$$

This complicated formula can be simplified so that the corrected antidiffusive flux reads

$$\alpha_{ij} f_{ij} = \max\{0, \min\{1, r_i\}\} d_{ij}(\mathrm{u}_i - \mathrm{u}_j) \tag{3.3.26}$$

where the generalized slope ratio r_i needs to be re-defined as follows

$$r_i = \frac{(\mathrm{u}_i + \mathrm{u}_{i-1})(\mathrm{u}_{i-1} - \mathrm{u}_i)}{(\mathrm{u}_i + \mathrm{u}_{i+1})(\mathrm{u}_i - \mathrm{u}_{i+1})} = \frac{\mathrm{u}_{i-1}^2 - \mathrm{u}_i^2}{\mathrm{u}_i^2 - \mathrm{u}_{i+1}^2} \tag{3.3.27}$$

For globally non-positive solution values, essentially the same formulas can be derived by swapping the indices $i-1$ and $i+1$ in the above equations and adopting the downwind node $j = i-1$. If the solution attains both positive and negative values, the situation is more complicated. Depending on the sign of the nodal average $0.5(\mathrm{u}_i + \mathrm{u}_{i\pm1})$, node i may be an upwind node receiving either one or two antidiffusive fluxes from its downwind neighbors $i \pm 1$. On the other hand, node i can be located downwind relating to both neighbors if its solution value satisfies $-\mathrm{u}_{i-1} < \mathrm{u}_i < \mathrm{u}_{i+1}$.

3.3.2. Iterative defect correction

The upwind-biased limiting strategy is derived on the semi-discrete level which makes it possible to apply algebraic flux correction of TVD type directly to steady-state problems of the form

$$\nabla \cdot \mathbf{f}(u) = 0 \qquad \text{in} \quad \mathbb{R}^{N_\mathrm{D}} \tag{3.3.28}$$

The corresponding weighted residual formulation is approximated by finite elements to obtain the high-order transport operator K. It is turned into its low-order counterpart $L = K + D$ by adding artificial diffusion as explained in Section 3.2.4. Application of the node-based limiting algorithm presented in Figure 3.2 gives rise to the nonlinear algebraic system

$$K^*(\mathrm{u})\mathrm{u} = 0 \tag{3.3.29}$$

which must be solved iteratively. Let us adopt the method of successive approximations

$$\mathrm{u}^{(m+1)} = \mathrm{u}^{(m)} + [P(\mathrm{u}^{(m)})]^{-1} K^*(\mathrm{u}^{(m)})\mathrm{u}^{(m)}, \qquad m = 0, 1, 2, \ldots \tag{3.3.30}$$

to compute a solution to the problem at hand, whereby the 'preconditioner' $P(\mathrm{u}^{(m)})$ should be easy to invert. Starting from an initial guess $\mathrm{u}^{(0)}$, the iteration process continues until convergence.

In a practical implementation, the 'inversion' of the preconditioner P is performed by a suitable iteration procedure for solving the linear subproblem for the solution increment $\Delta\mathrm{u}^{(m)}$:

$$\left.\begin{aligned} P(\mathrm{u}^{(m)})\Delta\mathrm{u}^{(m)} &= K^*(\mathrm{u}^{(m)})\mathrm{u}^{(m)} \\ \mathrm{u}^{(m+1)} &= \mathrm{u}^{(m)} + \Delta\mathrm{u}^{(m)} \end{aligned}\right\} \qquad m = 0, 1, 2, \ldots \tag{3.3.31}$$

A viable preconditioner can be constructed from the low-order transport operator [133, 144]

$$P = -L, \qquad L = K + D \tag{3.3.32}$$

for the linear scheme (3.2.26). The operator L was designed to satisfy the LED constraint Theorem 3.1.2 so that the off-diagonal entries of the preconditioner P are all non-positive which is in compliance with the sign condition (3.1.35). However, it is impossible to enforce the positivity of diagonal coefficients (3.2.31) so that P can lack the M-matrix property and may even be singular. As a remedy, problem (3.3.28) can be replaced by its transient counterpart which is marched into a steady state limit (addressed in Section 3.3.7). In this case the pseudo time step can be tuned accordingly to guarantee that the nonlinear preconditioner is an M-matrix, whence $P^{-1} \geq 0$.

3.3.3. Newton-like techniques

As an alternative to the method of successive approximations (3.3.30) the nonlinear algebraic system (3.3.29) can be linearized by means of the multivariate Taylor expansion

$$K^*(\bar{u})\bar{u} = K^*(u)u + J_{K^*}(u)(\bar{u}-u) + \text{higher-order terms} \tag{3.3.33}$$

where $\bar{u} \neq 0$ is a non-trivial root of the original problem, u represents a suitable approximation to it and J_{K^*} denotes the Jacobian matrix of $K^*(u)u$. Let this infinite series be truncated after the first derivative term and set the result equal to zero as required by the root-condition $K^*(\bar{u})\bar{u} = 0$. This gives an update formula for the solution vector commonly referred to as Newton's method

$$u^{(m+1)} = u^{(m)} + [P(u^{(m)})]^{-1} K^*(u^{(m)})u^{(m)}, \qquad m = 0,1,2,\ldots \tag{3.3.34}$$

which is akin to (3.3.30), and hence, the auxiliary procedure (3.3.31) is employed in practice to 'invert' the operator P. Here, the 'preconditioner' is an approximation to the exact Jacobian

$$P(u^{(m)}) = -J(u^{(m)}), \qquad J(u^{(m)}) \approx J_{K^*}(u^{(m)}) \tag{3.3.35}$$

evaluated at the known state $u^{(m)}$. The choice of a good initial guess $u^{(0)}$ in the above algorithm is crucial since Newton's method is likely to diverge for crude starting values.

For the solution of the steady Euler equations, Hemker et al. [100] suggest a two-step defect correction approach. A provisional first-order solution for the stationary problem is computed directly, that is, without resorting to pseudo time-stepping. Next, the low-order profile is used as initial guess for a second-order accurate defect correction iteration, whereby the first-order operator serves as a preconditioner. Within a Newton iteration approach, this idea may be adopted as follows [183]. In order to obtain a reasonable initial guess, a small number of 'presmoothing' steps is performed, whereby the monotone evolution operator (3.3.32) can be applied either per se (without resorting to algebraic flux correction) or as a preconditioner within a high-resolution flux/defect correction scheme. After a few outer iterations, the approximate Jacobian (3.3.35) is used so that Newton's algorithm is recovered from the above iteration procedure (3.3.34).

The formal definition (3.3.35) of the Jacobian matrix requires the 'differentiation' of the modified transport operator $K^*(u^{(m)})$ which is constructed on the semi-discrete level and consists of diffusive and antidiffusive contributions (see Sections 3.2.4 and 3.3.1). Due to the lack of a continuous counterpart that could be differentiated 'by hand' no analytical expression for the derivative of $K^*(u^{(m)})$ is available. Furthermore, the node-based flux limiter (3.3.2)–(3.3.4) makes use of *min* and *max* expressions which may not be globally differentiable. It is therefore mandatory to devise a suitable approximation $J(u^{(m)})$ to the exact Jacobian matrix J_{K^*} evaluated at $u^{(m)}$.

In a practical implementation, the two-step procedure (3.3.31) is adopted to circumvent the 'inversion' of the Jacobian matrix in Newton's formula (3.3.34). The use of Krylov subspace methods [227] for the solution of the auxiliary subproblems requires the computation of Jacobian-vector-products which may be approximated by means of first-order divided differences

$$J(u^{(m)})\Delta u^{(m)} = \frac{K^*(v^+)v^+ - K^*(u^{(m)})u^{(m)}}{\sigma}, \qquad v^+ = u^{(m)} + \sigma\Delta u^{(m)} \tag{3.3.36}$$

It is even possible to improve the accuracy from first to second order by combining forward and backward Taylor expansions and so as to end up with central differences

$$J(u^{(m)})\Delta u^{(m)} = \frac{K^*(v^+)v^+ - K^*(v^-)v^-}{2\sigma}, \qquad v^\pm = u^{(m)} \pm \sigma\Delta u^{(m)} \tag{3.3.37}$$

In practice, the performance of Newton's method is quite sensitive to the size of the perturbation parameter σ which should be sufficiently small to obtain a good approximation to the derivative

[109]. Following a simple strategy proposed by Nielsen et al. [191], the step size can be determined using the relation $\sigma\|\Delta u^{(m)}\| = \sqrt{\varepsilon}$, where ε denotes the machine precision. Some alternative choices are given in a survey paper by Knoll and Keyes [123]. An effective formula proposed by Pernice et al. and successfully used in the NITSOL package [206] reads as follows

$$\sigma\|\Delta u^{(m)}\| = \sqrt[p+1]{(1+\|u^{(m)}\|)\varepsilon} \tag{3.3.38}$$

where p denotes the order of the employed finite difference formula (usually $p=1$ or $p=2$).

What makes such matrix-free approaches most attractive at first glance is their Newton-like nonlinear convergence behavior without the costs of computing and storing the Jacobian explicitly. It suffices to assemble the modified transport operators K^* based on the last iterate $u^{(m)}$ and/or the 'perturbed' solution $v^{\pm} = u^{(m)} \pm \Delta u^{(m)}$ and apply the resulting matrices to the corresponding vectors. For the second-order accurate formula (3.3.37) this can be accomplished as follows:

1. Choose the perturbation parameter $\sigma > 0$, e.g., from relation (3.3.38).

2. Assemble the high-order transport operators $K(v^{\pm})$ and perform discrete upwinding (3.2.33) so as to eliminate all negative off-diagonal entries which yields the low-order operators
$$L(v^{\pm}) = K(v^{\pm}) + D(v^{\pm}), \qquad v^{\pm} = v^{(m)} \pm \Delta u^{(m)} \tag{3.3.39}$$

3. Resort to the generalized TVD algorithm (cf. Figure 3.2) and apply it to v^+ and v^- to construct the vectors of limited antidiffusion $\bar{f}(v^+)$ and $\bar{f}(v^-)$, respectively.

4. Approximate the Jacobian-vector product by second-order divided differences
$$J(u^{(m)})\Delta u^{(m)} = \frac{L(v^+)v^+ + \bar{f}(v^+) - L(v^-)v^- - \bar{f}(v^-)}{2\sigma} \tag{3.3.40}$$

If the governing equation is linear, then the low-order operator $L = K + D$ does not depend on the solution so that the second step in the above algorithm can be omitted which yields

$$J(u^{(m)})\Delta u^{(m)} = L\Delta u^{(m)} + \frac{\bar{f}(v^+) - \bar{f}(v^-)}{2\sigma} \tag{3.3.41}$$

A word of caution is in order. Despite its attractive simplicity, the use of (3.3.36) or (3.3.37) may become quite costly since multiple linear iterations are typically required in practice to perform one outer Newton step. In addition, each inner BiCGSTAB or GMRES [228] iteration may engender more than one matrix-vector multiplication $J(u^{(m)})x$, where $x \in \mathbb{R}^M$. Consequently, a single Newton step may be prohibitively expensive due to the recurrent assembly of low-order operators and the frequent use of algebraic flux correction based on the constantly changing vectors $u^{(m)} \pm x$.

Another crucial point is that there are only few preconditioners (for the linear subproblems) that can be applied without knowing the system matrix explicitly [50]. As a consequence, the solution process may slow down significantly or even fail to converge at all. Last but not least, Jacobian matrices resulting from finite element discretizations are in general very sparse, and hence, the savings in terms of memory usage are not as overwhelming as one might think.

3.3.4. Approximation of the Jacobian matrix for transport operators

In light of the above, the explicit assembly of Jacobians gains more attraction, provided a sufficiently accurate approximation can be computed at reasonable costs. For our purpose it makes sense to introduce the divided difference operator $\mathscr{D}_k$ for a generic function $g : \mathbb{R} \to \mathbb{R}$

$$\mathscr{D}_k[g] := \frac{g(u + \sigma e_k) - g(u - \sigma e_k)}{2\sigma} \tag{3.3.42}$$

whereby e_k denotes the k^{th} unit vector. In what follows, the iteration index m is omitted to improve readability. For a differentiable function $g(\mathrm{u})$, each nodal component of the first derivative can be approximated with second-order accuracy, i.e. $g_k'(\mathrm{u}) = \mathscr{D}_k[g] + \mathcal{O}(\sigma^2)$. If relation (3.3.38) is employed to compute the perturbation parameter σ, all columns exhibit the same approximation error proportional to $\sqrt[3]{[(1+\|\mathrm{u}\|)\varepsilon]^2}$, whereas the strategy by Nielsen et al. [191] yields $\sigma = \sqrt{\varepsilon}$.

With aid of the central difference operator defined in (3.3.42), each *column* of the exact Jacobian $J_{K^*}(\mathrm{u})$ can be approximated by divided differences such that [183]

$$[J(\mathrm{u})]_{\cdot,k} = \mathscr{D}_k[K^*(\mathrm{u})\mathrm{u}] = \mathscr{D}_k[L(\mathrm{u})\mathrm{u} + \bar{f}(\mathrm{u})] \tag{3.3.43}$$

where the term $\bar{f}_i = \sum_{j\neq i}\alpha_{ij}f_{ij}$ represents the amount of limited antidiffusion that is applied to node i. Let us turn off the upwind-biased flux limiter ($\alpha_{ij} \equiv 0$) for the time being and devise an edge-based algorithm for the efficient assembly of the approximate 'upwind' Jacobian $\mathscr{D}_k[L(\mathrm{u})\mathrm{u}]$ [183]. Of course, it reduces to the standard low-order transport operator $L = K + D$ if the governing equation is linear. Therefore, let us assume that the convective term depends on the unknown solution and consider a single entry of the associated Jacobian matrix

$$[J(\mathrm{u})]_{i,k} = \frac{\left[L(\mathrm{v}_k^+)\mathrm{v}_k^+\right]_i - \left[L(\mathrm{v}_k^-)\mathrm{v}_k^-\right]_i}{2\sigma}, \qquad \mathrm{v}_k^\pm = \mathrm{u} \pm \sigma\mathrm{e}_k \tag{3.3.44}$$

Here, subscript i refers to the i^{th} row of the vectors in brackets and i,k stands for the corresponding matrix coefficient for column k. Let us adopt the short-hand notation $L^\pm = \{l_{ij}^\pm\}$ for the modified low-order transport operators and regroup terms in the above expression so as to obtain [183]

$$[J(\mathrm{u})]_{i,k} = \left[\frac{L^+ - L^-}{2\sigma}\mathrm{u}\right]_i + \left[\frac{L^+ + L^-}{2}\mathrm{e}_k\right]_i, \qquad L^\pm = L(\mathrm{u} \pm \sigma\mathrm{e}_k) \tag{3.3.45}$$

In fact, a single column of an arbitrary matrix can be extracted via multiplication by the corresponding unit vector so that the second term in the above expression reduces to the average

$$\left[\frac{L^+ + L^-}{2}\mathrm{e}_k\right]_i = \frac{l_{ik}(\mathrm{u}+\sigma\mathrm{e}_k) + l_{ik}(\mathrm{u}-\sigma\mathrm{e}_k)}{2} \tag{3.3.46}$$

In our experience, the above coefficient has only minor influence on the global Jacobian matrix and can be safely replaced by the standard operator $L(\mathrm{u})$ evaluated at the unperturbed vector u. On the other hand, all necessary information is also required for the treatment of the divided difference of the convective term so that its standard average (3.3.46) can be computed at no additional costs. Let us investigate the crucial part of equation (3.3.45) and carry out the multiplication formally

$$\left[\frac{L^+ - L^-}{2\sigma}\mathrm{u}\right]_i = \frac{1}{2\sigma}\sum_{j=1}^{M}(l_{ij}^+ - l_{ij}^-)\mathrm{u}_j, \qquad l_{ij}^\pm = l_{ij}(\mathrm{u} \pm \sigma\mathrm{e}_k) \tag{3.3.47}$$

The most interesting question that arises is which scaling factors $l_{ij}^+ - l_{ij}^-$ are different from zero and which coefficients cancel each other. In particular, the perturbation of the solution applied to node k only affects its direct neighbors which share an element with it, that is

$$l_{ij}(\mathrm{u}+\sigma\mathrm{e}_k) \neq l_{ij}(\mathrm{u}-\sigma\mathrm{e}_k) \qquad \text{if} \quad k \in \{i,j\} \tag{3.3.48}$$

Hence, an edge ij has to be considered in the assembly process if and only if any endpoint is perturbed so that the sparsity patterns of the standard finite element matrix and of the upwind Jacobian coincide [183]. Moreover, there exist explicit formulas to compute the matrix coefficients

$$\left[\frac{L^+ - L^-}{2\sigma}\mathrm{u}\right]_{i,k} = \begin{cases} \mathscr{D}_k[l_{ik}]\mathrm{u}_k - \mathscr{D}_k[d_{ik}]\mathrm{u}_i & \text{if} \quad i \neq k \\ \displaystyle\sum_{j=1}^{M}\mathscr{D}_k[l_{ij}]\mathrm{u}_j & \text{if} \quad i = k \end{cases} \tag{3.3.49}$$

The upper expression corresponds to off-diagonal entries and follows directly from the definition of the low-order operator (3.2.33). In particular, the artificial diffusion coefficient d_{ik} which is used to render $l_{ik} = k_{ik} + d_{ik}$ non-negative is subtracted from the diagonal entries (3.2.31) in order to restore zero row and column sums [133, 144]. As a result, its divided difference is multiplied by the corresponding entry in the nodal solution vector and applied with negative sign. In short, each off-diagonal coefficient of the low-order Jacobian can be equivalently computed from

$$\left[\frac{L^+ - L^-}{2\sigma}\mathrm{u}\right]_{i,k} = \mathscr{D}_k[k_{ik}]\mathrm{u}_k + \mathscr{D}_k[d_{ik}](\mathrm{u}_k - \mathrm{u}_i) \qquad \text{if} \quad i \neq k \tag{3.3.50}$$

On the other hand, the diagonal coefficients of the approximate Jacobian receive contributions from all matrix positions of the corresponding row. It is worth mentioning that relation (3.2.31) can be used to rewrite formula (3.3.49) for the diagonal coefficients as follows

$$\left[\frac{L^+ - L^-}{2\sigma}\mathrm{u}\right]_{i,k} = \sum_{j=1}^{M} \mathscr{D}_k[k_{ij}]\mathrm{u}_j + \sum_{j\neq i} \mathscr{D}_k[d_{ij}](\mathrm{u}_j - \mathrm{u}_i) \qquad \text{if} \quad i = k \tag{3.3.51}$$

Expression (3.3.49) can be directly turned into an edge-based algorithm which can be used to assemble the approximate Jacobian matrix associated with the low-order transport operator efficiently. In a practical implementation $J = \{sc\,j_{ij}\}$ is initialized by the low-order operator $L(\mathrm{u})$ or its standard average (3.3.46), and moreover, the diagonal entries are augmented by $\mathscr{D}_i[k_{ii}]\mathrm{u}_i$. Then the global matrix can be assembled efficiently in a loop over the edges of the sparsity graph

$$\begin{aligned} \mathrm{J}_{ii} &:= \mathrm{J}_{ii} + \mathscr{D}_i[k_{ij}]\mathrm{u}_j + \mathscr{D}_i[d_{ij}](\mathrm{u}_j - \mathrm{u}_i), & \mathrm{J}_{ij} &:= \mathscr{D}_j[k_{ij}]\mathrm{u}_j + \mathscr{D}_j[d_{ij}](\mathrm{u}_j - \mathrm{u}_i) \\ \mathrm{J}_{ji} &:= \mathscr{D}_i[k_{ji}]\mathrm{u}_i - \mathscr{D}_i[d_{ij}](\mathrm{u}_j - \mathrm{u}_i), & \mathrm{J}_{jj} &:= \mathrm{J}_{jj} + \mathscr{D}_j[k_{ji}]\mathrm{u}_i - \mathscr{D}_j[d_{ij}](\mathrm{u}_j - \mathrm{u}_i) \end{aligned} \tag{3.3.52}$$

Note that the suggested procedure for the construction of $J(\mathrm{u})$ does not mean to compute the approximate Jacobian for the high-order transport operator $K(\mathrm{u})$ *first* and perform discrete upwinding in order to remove negative off-diagonal entries *afterwards*. In fact, expression (3.3.45) mimics the effect of algorithmic differentiation [85] applied to the discrete upwinding procedure. An edge-based assembly can also be used to approximate the Jacobian of the operator $K(\mathrm{u})$, whereby the diffusive contributions are no longer present in (3.3.52). This yields an efficient alternative to the element-by-element evaluation which is commonly employed in finite element codes [46].

3.3.5. Approximation of the Jacobian matrix for TVD schemes

So far, the flux limiter was turned off ($\alpha_{ij} \equiv 0$) to obtain the low-order operator $L(\mathrm{u})$ and unconstrained raw antidiffusion ($\alpha_{ij} \equiv 1$) was applied to remove excessive diffusion and recover $K(\mathrm{u})$, respectively. If the problem at hand is linear, then both operators do not depend on the solution, and hence, their Jacobian matrices reduce to L and K, respectively. On the other hand, application of the node-based limiting strategy (3.3.2)–(3.3.4) leads to solution-dependent correction factors $\alpha_{ij} = \alpha_{ij}(\mathrm{u})$ which need to be accounted for in the approximation of the Jacobian matrix even if the problem at hand is linear. It is therefore expedient to distinguish between *physical nonlinearities* which reside in the governing equations and *numerical nonlinearities* which are engendered artificially by the numerical method and require special treatment.

Let us revisit equation (3.3.43) and devise a strategy for the assembly of the Jacobian matrix associated with the upwind-biased flux limiting procedure (cf. Figure 3.2). In particular, each column of the Jacobian matrix $\mathscr{D}_k[\bar{f}(\mathrm{u})]$ can be constructed by the algorithm presented in Figures 3.3 and 3.4. In the first step, the precomputed nodal quantities $P_i^{\pm}$ and $Q_i^{\pm}$ are reused to initialize their *local* counterparts $\mathsf{P}_i^{\pm}[\mathrm{u}^{\pm}]$ and $\mathsf{Q}_i^{\pm}[\mathrm{u}^{\pm}]$, whereby the stencil of node k is defined as follows [183]:

$$\mathcal{S}_k = \{i : (\varphi_i, \varphi_k) \neq 0\} \tag{3.3.53}$$

Due to this kind of initialization, it is mandatory to remove the contribution of unperturbed fluxes prior to applying their perturbed counterparts in expressions (3.3.56) and (3.3.57), respectively. Note that forward and backward perturbations indicated by $[\mathrm{u}^{\pm}] = [\mathrm{u} \pm \sigma \mathrm{e}_k]$ need to be treated separately. Finally, the nodal correction factors $\mathsf{R}_i^{\pm}[\mathrm{u}^{\pm}]$ are evaluated for all vertices in the stencil $\mathcal{S}_k$ and used to limit the antidiffusive fluxes which are affected by the perturbation of the solution at node k. In contrast to the 'upwind' Jacobian addressed in the previous section which exhibits

In a loop over each column $k = 1, 2, \ldots, M$ of the Jacobian matrix J proceed as follows:

1. Set $\mathsf{P}_k^{\pm}[\mathrm{u}^{\pm}] = 0$ and $\mathsf{Q}_k^{\pm}[\mathrm{u}^{\pm}] = 0$ and initialize the nodal quantities in the neighborhood of node k by their unperturbed counterparts

$$\mathsf{P}_i^{\pm}[\mathrm{u}^{\pm}] := P_i^{\pm}, \qquad \mathsf{Q}_i^{\pm}[\mathrm{u}^{\pm}] := Q_i^{\pm}, \qquad \forall i \in \mathcal{S}_k \setminus \{k\} \tag{3.3.54}$$

2. Loop over all oriented edges $\vec{ij}$ such that $k \in \{i, j\}$

   (a) Retrieve $f_{ij} = d_{ij}(\mathrm{u}_i - \mathrm{u}_j)$ and compute the fluxes for the perturbed node k

$$\begin{aligned} f_{ij}^{+} &= d_{ij}[\mathrm{u} + \sigma \mathrm{e}_k](\mathrm{u}_i + \sigma\delta_{ik} - \mathrm{u}_j - \sigma\delta_{jk}) \\ f_{ij}^{-} &= d_{ij}[\mathrm{u} - \sigma \mathrm{e}_k](\mathrm{u}_i - \sigma\delta_{ik} - \mathrm{u}_j + \sigma\delta_{jk}) \end{aligned} \tag{3.3.55}$$

   (b) Update the *local* sums of positive and negative antidiffusive fluxes

$$\left.\begin{aligned} \mathsf{P}_i^{\pm}[\mathrm{u}^{+}] &:= \mathsf{P}_i^{\pm}[\mathrm{u}^{+}] + \max_{\min}\{0, f_{ij}^{+}\} \\ \mathsf{P}_i^{\pm}[\mathrm{u}^{-}] &:= \mathsf{P}_i^{\pm}[\mathrm{u}^{-}] + \max_{\min}\{0, f_{ij}^{-}\} \end{aligned}\right\} \quad \text{if} \quad i = k \neq j$$

$$\left.\begin{aligned} \mathsf{P}_i^{\pm}[\mathrm{u}^{+}] &:= \mathsf{P}_i^{\pm}[\mathrm{u}^{+}] + \max_{\min}\{0, f_{ij}^{+}\} - \max_{\min}\{0, f_{ij}\} \\ \mathsf{P}_i^{\pm}[\mathrm{u}^{-}] &:= \mathsf{P}_i^{\pm}[\mathrm{u}^{-}] + \max_{\min}\{0, f_{ij}^{-}\} - \max_{\min}\{0, f_{ij}\} \end{aligned}\right\} \quad \text{if} \quad i \neq k = j \tag{3.3.56}$$

   (c) Update the *local* upper and lower bounds for admissible antidiffusion

$$\left.\begin{aligned} \mathsf{Q}_i^{\pm}[\mathrm{u}^{+}] &:= \mathsf{Q}_i^{\pm}[\mathrm{u}^{+}] + \max_{\min}\{0, -f_{ij}^{+}\} \\ \mathsf{Q}_i^{\pm}[\mathrm{u}^{-}] &:= \mathsf{Q}_i^{\pm}[\mathrm{u}^{-}] + \max_{\min}\{0, -f_{ij}^{-}\} \\ \mathsf{Q}_j^{\pm}[\mathrm{u}^{+}] &:= \mathsf{Q}_j^{\pm}[\mathrm{u}^{+}] + \max_{\min}\{0, \quad f_{ij}^{+}\} - \max_{\min}\{0, \quad f_{ij}\} \\ \mathsf{Q}_j^{\pm}[\mathrm{u}^{-}] &:= \mathsf{Q}_j^{\pm}[\mathrm{u}^{-}] + \max_{\min}\{0, \quad f_{ij}^{-}\} - \max_{\min}\{0, \quad f_{ij}\} \end{aligned}\right\} \quad \text{if} \quad i = k \neq j$$

$$\left.\begin{aligned} \mathsf{Q}_i^{\pm}[\mathrm{u}^{+}] &:= \mathsf{Q}_i^{\pm}[\mathrm{u}^{+}] + \max_{\min}\{0, -f_{ij}^{+}\} - \max_{\min}\{0, -f_{ij}\} \\ \mathsf{Q}_i^{\pm}[\mathrm{u}^{-}] &:= \mathsf{Q}_i^{\pm}[\mathrm{u}^{-}] + \max_{\min}\{0, -f_{ij}^{-}\} - \max_{\min}\{0, -f_{ij}\} \\ \mathsf{Q}_j^{\pm}[\mathrm{u}^{+}] &:= \mathsf{Q}_j^{\pm}[\mathrm{u}^{+}] + \max_{\min}\{0, \quad f_{ij}^{+}\} \\ \mathsf{Q}_j^{\pm}[\mathrm{u}^{-}] &:= \mathsf{Q}_j^{\pm}[\mathrm{u}^{-}] + \max_{\min}\{0, \quad f_{ij}^{-}\} \end{aligned}\right\} \quad \text{if} \quad i \neq k = j \tag{3.3.57}$$

Fig. 3.3. Assembly of Jacobian matrix for the upwind-biased flux limiter.

3. Evaluate the nodal correction factors for all nodes i in the neighborhood of k

$$\mathsf{R}_i^{\pm}[\mathrm{u}^{\pm}] = \min\{1, \mathsf{Q}_i^{\pm}[\mathrm{u}^{\pm}]/\mathsf{P}_i^{\pm}[\mathrm{u}^{\pm}]\}, \qquad \forall i \in \mathcal{S}_k \tag{3.3.58}$$

4. Loop over all oriented edges $\vec{ij}$ such that $i \in \mathcal{S}_k$ and $j \in \mathcal{S}_k$

   (a) Check the sign of the antidiffusive fluxes (3.3.55) and adopt the correction factors

$$\alpha_{ij}^{+} = \begin{cases} \mathsf{R}_i^{+}[\mathrm{u}^{+}] & \text{if} \quad f_{ij}^{+} > 0 \\ \mathsf{R}_i^{-}[\mathrm{u}^{+}] & \text{if} \quad f_{ij}^{+} \le 0 \end{cases} \qquad \alpha_{ij}^{-} = \begin{cases} \mathsf{R}_i^{+}[\mathrm{u}^{-}] & \text{if} \quad f_{ij}^{-} > 0 \\ \mathsf{R}_i^{-}[\mathrm{u}^{-}] & \text{if} \quad f_{ij}^{-} \le 0 \end{cases} \tag{3.3.59}$$

   (b) Apply the divided difference of the limited fluxes to the Jacobian matrix

$$\bar{f}_{ij} = \frac{\alpha_{ij}^{+} f_{ij}^{+} - \alpha_{ij}^{-} f_{ij}^{-}}{2\sigma}, \qquad \mathrm{J}_{ik} := \mathrm{J}_{ik} + \bar{f}_{ij}, \qquad \mathrm{J}_{jk} := \mathrm{J}_{jk} - \bar{f}_{ij} \tag{3.3.60}$$

5. Loop over all oriented edges $\vec{ij}$, $k \notin \{i, j\}$ such that *upwind* node $i \in \mathcal{S}_k$ and $j \notin \mathcal{S}_k$

   (a) Retrieve the precomputed raw antidiffusive flux $f_{ij} = d_{ij}(\mathrm{u}_i - \mathrm{u}_j)$ and employ the correction factors from the upwind node i

$$\alpha_{ij}^{\pm} = \begin{cases} \mathsf{R}_i^{+}[\mathrm{u}^{\pm}] & \text{if} \quad f_{ij} > 0 \\ \mathsf{R}_i^{-}[\mathrm{u}^{\pm}] & \text{if} \quad f_{ij} \le 0 \end{cases} \tag{3.3.61}$$

   (b) Apply the divided difference of the limited flux to the Jacobian matrix

$$\bar{f}_{ij} = \frac{\alpha_{ij}^{+} - \alpha_{ij}^{-}}{2\sigma} f_{ij}, \qquad \mathrm{J}_{ik} := \mathrm{J}_{ik} + \bar{f}_{ij}, \qquad \mathrm{J}_{jk} := \mathrm{J}_{jk} - \bar{f}_{ij} \tag{3.3.62}$$

Fig. 3.4. Assembly of Jacobian matrix for the upwind-biased flux limiter (continued).

the same sparsity pattern as the finite element stiffness matrix, the node-based flux limiter of TVD type gives rise to additional entries in each column $\mathscr{D}_k[\bar{f}(\mathrm{u})]$ of the Jacobian matrix. Remarkably, the structure of the connectivity graph of the Jacobian matrix for the antidiffusive contribution is known *a priori* and it can be constructed by resorting to symbolic matrix multiplication. Let us present the basic concepts by considering a simple model problem in one space dimension and postpone the extension to multidimensions to the next Section.

Example 3.3.3. Consider the one-dimensional transport equation $\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0$ and let the velocity $v > 0$ be constant. Application of discrete upwinding (cf. Example 3.2.3) yields the standard upwind method (3.2.38) and the raw antidiffusive flux $f_{ij} = \frac{v}{2}(\mathrm{u}_i - \mathrm{u}_j)$ from the downwind node $j = i+1$ is limited with aid of the correction factor evaluated at the upwind node i

$$\alpha_{ij} = \max\{0, \min\{1, r_i\}\}, \qquad r_i = \frac{\mathrm{u}_{i-1} - \mathrm{u}_i}{\mathrm{u}_i - \mathrm{u}_{i+1}} \tag{3.3.63}$$

The algorithmic details of algebraic flux correction are presented in Example 3.3.1. Owing to the linearity of the pure convection equations, the low-order Jacobian reduces to the upwind operator L. The crucial task is to approximate the derivative of the antidiffusive contribution.

In order to assemble the i^{th} column of the Jacobian matrix, let us modify the solution vector according to $\mathrm{u}_i^{\pm} = \mathrm{u}_i \pm \sigma$ which gives rise to the perturbed antidiffusive fluxes $f_{ij}^{\pm} = \frac{v}{2}(\mathrm{u}_i^{\pm} - \mathrm{u}_j)$.

Moreover, the slope ratio r_i for the upwind node i is likely to change due to variations in the numerator and denominator. Consequently, the divided difference of limited antidiffusive fluxes

$$\mathscr{D}_i[\alpha_{ij} f_{ij}] = \frac{\alpha_{ij}^+ f_{ij}^+ - \alpha_{ij}^- f_{ij}^-}{2\sigma} \tag{3.3.64}$$

contributes to the i^{th} row (with positive sign) and also to row $j = i+1$ (with negative sign). In addition, the perturbation exerted on node i may also affect the flux into the upwind node $i-1$

$$f_{i-1,i}^{\pm} = \frac{v}{2}(\mathrm{u}_{i-1} - \mathrm{u}_i^{\pm}) \tag{3.3.65}$$

and changes in the denominator of the slope ratio r_{i-1} are likely to occur so that rows $i-1$ and i of the i^{th} column may be affected by the perturbation. So far, the sparsity pattern of the standard finite element matrix may also be adopted for the Jacobian, that is,

$$\mathrm{J}_{ji} \neq 0 \qquad \text{if} \quad |i-j| \leq 1 \tag{3.3.66}$$

As we are about to see, some extension is mandatory in order to account for extra contributions to node $i+2$. Consider the nodal quantities $r_{i+1}^{\pm}$ which are proportional to the slope ratios

$$r_i^{\pm} = \frac{\mathrm{u}_i^{\pm} - \mathrm{u}_{i+1}}{\mathrm{u}_{i+1} - \mathrm{u}_{i+2}} \tag{3.3.67}$$

Due to the presence of the nodewise perturbed quantities $\mathrm{u}_i^{\pm}$, the difference of *limited* antidiffusive fluxes running between nodes $i+1$ and $i+2$ may not cancel out, and therefore, yield a non-vanishing contribution to rows $i+1$ and $i+2$ in the i^{th} column of the Jacobian. In general, the upwind direction may not be the same for all nodes which leads to the pentadiagonal matrix

$$\mathrm{J}_{ji} \neq 0 \qquad \text{if} \quad |i-j| \leq 2 \tag{3.3.68}$$

The same sparsity pattern is recovered for the mass matrix if quadratic finite elements are adopted.

3.3.6. Sparsity pattern of the Jacobian matrix

Let us generalize the one-dimensional concepts to multidimensions and present a straightforward approach to construct the connectivity graph of the Jacobian matrix. In light of the above, the sparsity pattern of the standard finite element matrix needs to be extended by one 'connectivity layer' so as to account for additional fill-in engendered by the flux limiter. In particular, non-vanishing coefficients present in the k^{th} column of the Jacobian matrix $\mathscr{D}_k[\bar{f}(\mathrm{u})]$ may be [183]

- located on the diagonal, i.e. J_{kk},
- associated with the edge ij, whereby either of its endpoints is node k,
- associated with the edge ij and there exists the edge kl such that $l \in \{i, j\}$ which indirectly 'links' node k to nodes i and j via the common vertex l.

Note that the edge orientation convention [145] has not been adopted so that the sparsity pattern of the resulting Jacobian matrix is symmetric. To illustrate the structure of the connectivity graph, consider mixed bi-/linear finite elements employed on an unstructured mesh as depicted in Figure 3.5 (left). Let the centered node correspond to the column number k so that the edges (indicated by dotted lines) are associated with non-zero off-diagonal coefficients in the finite element matrix.

In order to assemble the k^{th} column of the approximate Jacobian matrix $\mathscr{D}_k[\bar{f}(\mathrm{u})]$, the solution value at node k is perturbed which yields the extended sparsity pattern depicted in Figure 3.5

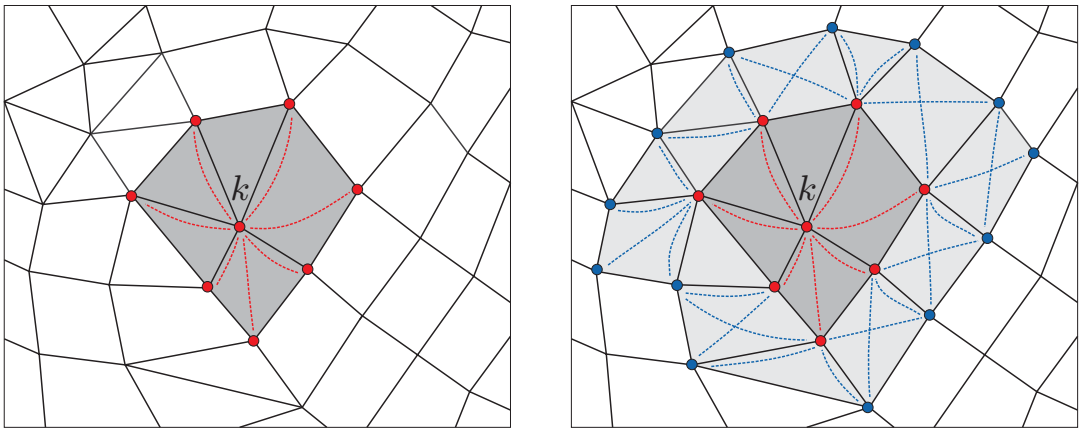


Fig. 3.5. Connectivity graph: FE matrix vs. Jacobian operator.

(right). Of course, the antidiffusive flux $f_{ij} = d_{ij}(\mathrm{u}_i - \mathrm{u}_j)$ which is associated with the edge $\vec{ij}$ may be affected by the perturbation provided that $k \in \{i, j\}$. As a result, the nodal quantities $\mathsf{P}_i^{\pm}[\mathrm{u}^{\pm}]$, $\mathsf{Q}_i^{\pm}[\mathrm{u}^{\pm}]$ and $\mathsf{R}_i^{\pm}[\mathrm{u}^{\pm}]$ computed from (3.3.56) and (3.3.57) for node k and all of its *direct neighbors* may differ from their unperturbed counterparts recovered from the upwind-biased flux limiter (3.3.2)–(3.3.4). Hence, the corresponding matrix coefficients need to be updated in the k^{th} column of the Jacobian. The divided difference of the limited flux for the edge $\vec{ij}$ is given by

$$\mathscr{D}_k[\alpha_{ij} f_{ij}] = \frac{\mathsf{R}_i^{\pm}[\mathrm{u}^+] f_{ij}^+ - \mathsf{R}_i^{\pm}[\mathrm{u}^-] f_{ij}^-}{2\sigma}, \qquad k \in \{i, j\} \tag{3.3.69}$$

whereby the appropriate correction factor for the upwind node i are chosen in accordance with the sign of the perturbed raw fluxes $f_{ij}^{\pm}$. Here, the notation $\mathsf{R}_i^{\pm}[\mathrm{u}^{\pm}]$ implies that the evaluation of the multiplier (3.3.58) is based on the perturbed solution vector $\mathrm{u}^{\pm} = \mathrm{u} \pm \sigma \mathrm{e}_k$. Without loss of generality, let node $i \neq k$ be a direct neighbor of vertex k and consider the antidiffusive flux f_{ij} with $j \neq k$ which needs to be limited by the nodal correction factor from the upwind node i. Hence, the perturbed quantities $\mathsf{R}_i^{\pm}[\mathrm{u}^{\pm}]$ may lead to a non-vanishing divided difference of the form

$$\mathscr{D}_k[\alpha_{ij}] f_{ij} = \frac{\mathsf{R}_i^{\pm}[\mathrm{u}^+] - \mathsf{R}_i^{\pm}[\mathrm{u}^-]}{2\sigma} f_{ij} \qquad k \notin \{i, j\} \wedge \exists ik \tag{3.3.70}$$

whereby the raw flux f_{ij} is not affected by the perturbed solution. It is instructive to refer to node j as an *indirect neighbor* of vertex k since there exists a pair of (unoriented) edges ij and ik which 'couples' nodes j and k by virtue of the common vertex i. It is thus that all matrix coefficients that correspond to the second 'layer' of nodes need to be considered in the k^{th} column of the Jacobian.

It is worth mentioning that the extended sparsity pattern depicted in Figure 3.5 (right) resembles that of the edge-oriented stabilization technique [43, 44]. A detailed description of the underlying data structure for non-conforming bilinear finite elements is presented by Ouazzi and Turek [199]. It is possible to adapt the suggested storage algorithm to conforming linear/bilinear finite elements but care must be taken to avoid duplicate entries in the rows of the finite element matrix. It is therefore expedient to review some ideas from graph theory and devise an alternative storage algorithm which is directly tailored to conforming (bi-)linear finite elements [183].

Let the adjacency graph of the stiffness matrix be denoted by $X = \{x_{ij}\} \in \{0,1\}^{M \times M}$. It is symmetric since $x_{ij} = x_{ji} = 1$ if and only if the finite element basis functions φ_i and φ_j have overlapping supports. In other words, either $i = j$ or there exists an edge ij connecting nodes i and j. Each coefficient of the square matrix $Y = X \cdot X \in \{0,1,2\}^{M \times M}$ satisfies $y_{ij} > 0$ if and only if

there exists a path of length not greater than two between nodes i and j since

$$y_{ij} = \sum_{k=1}^{M} x_{ik} x_{kj} > 0 \quad \Leftrightarrow \quad \exists k : x_{ik} = 1 \wedge x_{kj} = 1 \tag{3.3.71}$$

As a result, a standard algorithm for sparse matrix multiplication [18] can be employed to generate the sparsity pattern of the Jacobian matrix which is symmetric by construction.

3.3.7. Time-stepping schemes

This paragraph deals with the extension of the iterative solution strategies presented in Sections 3.3.2 and 3.3.3 to transient or pseudo-transient conservation laws of the form

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0 \qquad \text{in} \quad \mathbb{R}^{N_D} \times \mathbb{R}_0^+ \tag{3.3.72}$$

It is common practice in the computation of steady flows to introduce an artificial time derivative which is supposed to vanish if the flow becomes stationary. In other words, the above model problem is adopted and marched towards steady-state so as to obtain a converged solution.

Let the above equation be discretized in space using the upwind-biased limiting strategy proposed by Kuzmin and Turek [145] so as to obtain the semi-discrete high-resolution scheme

$$M_L \frac{\mathrm{du}}{\mathrm{d}t} = K^*(\mathrm{u})\mathrm{u} \tag{3.3.73}$$

which is designed to satisfy the local extremum diminishing constraint 3.1.2. Additional criteria are required to guarantee that such methods remain positivity-preserving [144] after the discretization in time (see Section 3.1.3). Application of the two-level θ-scheme, where the implicitness parameter θ may vary between zero and one, yields the nonlinear algebraic system

$$A(\mathrm{u}^{n+1})\mathrm{u}^{n+1} = B(\mathrm{u}^n)\mathrm{u}^n, \qquad \left\{ \begin{array}{rcl} A(\mathrm{u}^{n+1}) & = & M_L - \theta \Delta t K^*(\mathrm{u}^{n+1}) \\ B(\mathrm{u}^n) & = & M_L + (1-\theta)\Delta t K^*(\mathrm{u}^n) \end{array} \right. \tag{3.3.74}$$

According to Theorem 3.1.3, the positivity of u^n carries over to the new solution provided that all entries of the operator $B(\mathrm{u}^n)$ are non-negative and $A(\mathrm{u}^{n+1})$ satisfies the M-matrix property.

For $\theta = 0$, the explicit forward Euler method is recovered from (3.3.74) which is known to be unstable. As a remedy, Löhner et al. [167, 170] employ a two-step Taylor-Galerkin method of the Lax-Wendroff type for the design of fully explicit high-resolution methods. In general, the two-level θ-scheme method (3.3.74) is unconditionally stable for all parameter values $\theta \geq \frac{1}{2}$ and the choice $\theta = \frac{1}{2}$ corresponds to the semi-implicit Crank-Nicolson discretization which is the only θ-scheme that exhibits second-order accuracy [64]. This trapezoidal rule is positivity-preserving [144] provided the time step does not exceed an upper bound of the form (3.1.34) so that matrix $B(\mathrm{u}^n) \geq 0$ is non-negative. Furthermore, the operator $A(\mathrm{u}^{n+1})$ must satisfy the M-matrix property which may lead to additional constraints (3.1.38) to be imposed on the time step. Last but not least, the backward Euler method is recovered for $\theta = 1$. It corresponds to first-order upwind in time which makes it overly diffusive at large time steps. At the same time, this fully implicit scheme is unconditionally positive [144] so that arbitrary time steps can be employed unless iterative solvers fail to converge. In the case of an implicit time discretization ($0 < \theta \leq 1$), the physical nonlinearities inherent to the governing equation and/or the numerical nonlinearities due to the employed high-resolution scheme call for the use of an iterative solution strategy.

Several strategies for computing successive approximations to the end-of-step solution u^{n+1} of the nonlinear algebraic system (3.3.74) can be based on the fixed-point iteration scheme

$$\mathrm{u}^{(m+1)} = \mathrm{u}^{(m)} + [P(\mathrm{u}^{(m)})]^{-1} r^{(m)}, \qquad \mathrm{u}^{(0)} = \mathrm{u}^n, \qquad m = 0, 1, 2, \ldots \tag{3.3.75}$$

which is initialized by the solution from the last time step. The iteration process continues until the norm of the global defect or that of the relative changes between two successive iterates becomes small enough. The constant right-hand side and the residual for the m^{th} cycle are given by

$$b^n = B(\mathrm{u}^n)\mathrm{u}^n, \qquad r^{(m)} = b^n - A(\mathrm{u}^{(m)})\mathrm{u}^{(m)} \tag{3.3.76}$$

In a practical implementation, the contribution of the modified transport operator applied to the solution vector can be assembled edge-by-edge without the need to assemble matrix K^* explicitly. Moreover, an auxiliary iteration procedure is employed for the 'inversion' of the operator P:

$$\left.\begin{aligned} P(\mathrm{u}^{(m)})\Delta\mathrm{u}^{(m)} &= r^{(m)} \\ \mathrm{u}^{(m+1)} &= \mathrm{u}^{(m)} + \Delta\mathrm{u}^{(m)} \end{aligned}\right\} \qquad \mathrm{u}^{(0)} = \mathrm{u}^n, \qquad m = 0,1,2,\ldots \tag{3.3.77}$$

The evolution operator of the LED scheme (3.2.26) constitutes a viable preconditioner [133, 144]

$$P = M_L - \theta\Delta t L, \qquad L = K + D \tag{3.3.78}$$

Note that all off-diagonal entries of the low-order transport operator L are non-negative by construction, and therefore, those of matrix P comply with the sign condition (3.1.35). The strict positivity of diagonal coefficients may be satisfied from the outset and it can be ensured by constraining the time step subject to the upper bound (3.1.38) otherwise. Thus, the preconditioner (3.3.78) can be turned into an M-matrix, whereby its inverse is given by a non-negative matrix.

In practice, the diagonal dominance of P can be further enhanced by means of an implicit under-relation strategy [200] which is described in Section 3.1.3. Moreover, Meijerink and van der Vorst [181] showed that there exists a unique ILU(k)-decomposition for all values $k \in \mathbb{N}$ provided that the operator P satisfies the M-matrix property. Thus, the linear convergence rates of a Krylov method applied to the auxiliary problems (3.3.77) can be improved by resorting to preconditioning based on an incomplete LU-factorization of the preconditioner (3.3.78).

As an alternative to the low-order operator (3.3.78) which is constant for linear problems, the fixed-point iteration scheme (3.3.75) can be preconditioned by the nonlinear LED operator [134]

$$P(\mathrm{u}^{(m)}) = M_L - \theta\Delta t L^*(\mathrm{u}^{(m)}) \tag{3.3.79}$$

The existence of the operator L^* which includes limited antidiffusion (3.3.13) is guaranteed by the upwind-biased flux limiter [145]. All intermediate solutions $\mathrm{u}^{(m)}$ remain positivity-preserving [122] by construction but convergence is a prerequisite for mass conservation [134]. However, the construction of (3.3.79) is time-consuming and quite cumbersome for arbitrary discretizations. Moreover, it has to be re-assembled in each outer iteration due to the numerical nonlinearity engendered by the flux limiter. This has led to the design of a discrete Newton method [183] which allows for an efficient treatment of time-dependent and steady-state flows alike.

Let us rewrite the nonlinear algebraic system (3.3.74) recovered from the θ-scheme as follows

$$\mathcal{F}(\bar{\mathrm{u}};\mathrm{u}^n) = A(\bar{\mathrm{u}})\bar{\mathrm{u}} - B(\mathrm{u}^n)\mathrm{u}^n \tag{3.3.80}$$

where u^n denotes the given data from the previous time step and the new solution $\mathrm{u}^{n+1} = \bar{\mathrm{u}}$ at time t^{n+1} can be determined by computing the root of the above expression. The Newton iteration for $\mathcal{F}(\bar{\mathrm{u}};\mathrm{u}^n) = 0$ is derived from the multivariate Taylor expansion about the current state $\mathrm{u}^{(m)}$

$$\mathcal{F}(\mathrm{u}^{(m+1)};\mathrm{u}^n) = \mathcal{F}(\mathrm{u}^{(m)};\mathrm{u}^n) + J_{\mathcal{F}}(\mathrm{u}^{(m)};\mathrm{u}^n)(\mathrm{u}^{(m+1)} - \mathrm{u}^{(m)}) + \text{higher-order terms} \tag{3.3.81}$$

where $J_{\mathcal{F}}$ represents the Jacobian matrix of the function $\mathcal{F}$. Neglecting terms of higher-order curvature and setting the right-hand side equal to zero one obtains the sequence of linear systems

$$\left.\begin{aligned} J_{\mathcal{F}}(\mathrm{u}^{(m)};\mathrm{u}^n)\Delta\mathrm{u}^{(m)} &= -\mathcal{F}(\mathrm{u}^{(m)};\mathrm{u}^n) \\ \mathrm{u}^{(m+1)} &= \mathrm{u}^{(m)} + \Delta\mathrm{u}^{(m)} \end{aligned}\right\} \qquad \mathrm{u}^{(0)} = \mathrm{u}^n, \qquad m = 0,1,2,\ldots \tag{3.3.82}$$

The Newton iteration is terminated based on a required drop in the norm of the nonlinear residual $\|\mathcal{F}(\mathrm{u}^{(m)};\mathrm{u}^n)\|/\|\mathcal{F}(\mathrm{u}^{(0)};\mathrm{u}^n)\| < \mathrm{tol}_{\mathrm{abs}}$ and/or a sufficiently small increment $\|\Delta\mathrm{u}^{(m)}\|/\|\mathrm{u}^{(m)}\| < \mathrm{tol}_{\mathrm{rel}}$.

It is easy to verify that the right-hand side $-\mathcal{F}(\mathrm{u}^{(m)};\mathrm{u}^n)$ equals the residual term $r^{(m)}$ defined in (3.3.76). A viable approximation to the exact Jacobian $J_{\mathcal{F}}$ can be computed as follows

$$P(\mathrm{u}^{(m)}) = M_L - \theta\Delta t J(\mathrm{u}^{(m)}), \qquad J(\mathrm{u}^{(m)}) \approx J_{K^*}(\mathrm{u}^{(m)}) \tag{3.3.83}$$

whereby the Jacobian matrix J_{K^*} for the convective contribution (3.3.33) can be approximated as explained in Sections 3.3.4 and 3.3.5. Adopting the above preconditioner in the fixed-point iteration scheme (3.3.75) yields a strict Newton method for the transient problem (3.3.72).

The operator defined in (3.3.83) is not an M-matrix in general so that intermediate solutions will not preserve positivity. Hence, convergence of the Newton iteration is a prerequisite for the numerical scheme to be positivity-preserving [144]. The use of a globalization strategy to achieve/improve convergence and some useful convergence criteria are addressed in Section 3.3.8.

A common practice in the computation of the stationary problem (3.3.28) is to resort to the corresponding transient formulation (3.3.72) and march the solution to steady state. Since temporal accuracy is non-relevant the fully implicit backward Euler method ($\theta = 1$) is adopted, whereby the artificial time step is taken as large as possible. Moreover, it can be initially chosen reasonably small to ensure convergence and adjusted in the course of the simulation to quickly reach steady state. There is a strong relation between the fully implicit backward Euler method (3.3.74) and the stationary formulation (3.3.29) equipped with an under-relaxation strategy. The use of a constant time step for the solution update (3.3.75) corresponds to taking an individual relaxation factor for each nodal equation (3.3.30). Conversely, uniform under-relaxation in the steady-state approach is equivalent to adopting a different time step for each node [76].

Let us summarize the presented algorithms which are based on the generalized TVD algorithm by Kuzmin and Turek [145]. In essence, all nonlinear solution procedures can be expressed as

$$\mathrm{u}^{(m+1)} = \mathrm{u}^{(m)} + [P(\mathrm{u}^{(m)})]^{-1} r^{(m)}, \qquad m = 0,1,2,\ldots \tag{3.3.84}$$

whereby the possible choices for the preconditioner P and the 'residual' vector $r^{(m)}$ are presented in Figure 3.1. If the steady-state problem is solved directly, both the low-order transport operator L and the nonlinear LED operator L^* may fail to possess the M-matrix property. Moreover, they can become singular due to the presence of zero rows resulting from the conservative elimination of negative off-diagonal entries. In contrast, tuning the (pseudo) time step according to constraint (3.1.38) provides a mechanism to guarantee that the low-order evolution operator $M_L - \theta\Delta L$ is an M-matrix so that its inverse exists and has non-negative entries.

Steady-state problems	$r^{(m)} = K^*(\mathrm{u}^{(m)})\mathrm{u}^{(m)}, \quad \mathrm{u}^{(0)} = \mathrm{u}_0$
Successive approximations	$P = -L, \quad$ or $\quad P = -L^*(\mathrm{u}^{(m)})$
Newton's method	$P = -J(\mathrm{u}^m), \quad$ where $\quad J(\mathrm{u}^m) \approx J_{K^*}(\mathrm{u}^{(m)})$

Transient problems	$r^{(m)} = B(\mathrm{u}^n)\mathrm{u}^n - A(\mathrm{u}^{(m)})\mathrm{u}^{(m)}, \quad \mathrm{u}^{(0)} = \mathrm{u}^n$
Successive approximations	$P = M_L - \theta\Delta t L, \quad$ or $\quad P = M_L - \theta\Delta t L^*(\mathrm{u}^{(m)})$
Newton's method	$P = M_L - \theta\Delta t J(\mathrm{u}^{(m)}), \quad$ where $\quad J(\mathrm{u}^m) \approx J_{K^*}(\mathrm{u}^{(m)})$

Tab. 3.1. Overview of nonlinear solution strategies.

3.3.8. Convergence criteria and globalization

In general, the nonlinear preconditioners (3.3.35) and (3.3.83) employed in the Newton iteration do not possess the M-matrix property so that convergence of the fixed-point iteration (3.3.84) is a prerequisite for positivity-preservation. Admittedly, the performance of Newton's method depends on the quality of the initial guess. For transient flows, the solution from the previous time step provides a reasonable starting value unless extremely large time steps are employed. In contrast, the simulation of stationary problems may require some 'pre-iterations' by means of the low-order upwind scheme in order to predict a usable initial guess for the Newton iteration.

Since the linear system (3.3.84) for the increment $\Delta \mathrm{u}^{(m)} = \mathrm{u}^{(m+1)} - \mathrm{u}^{(m)}$ is solved iteratively, the resulting algorithm is classified as an inexact Newton method [63]. The accuracy of the (inner) linear solver greatly affects the convergence behavior of the (outer) Newton algorithm. If the linear subproblems are not solved accurately enough, more Newton steps are required, and hence, the nonlinear convergence rates deteriorate. Conversely, a very small tolerance for the linear solver results in a drastic increase of inner iterations which does not pay off. Moreover, an iterate $\mathrm{u}^{(m)}$ which is not sufficiently close to the desired root may be a poor candidate for linearization by means of Taylor expansion. In fact, it does not reflect the behavior of the nonlinear residual very well. As a consequence, solving the linear system with high accuracy one may obtain a poor Newton update $\Delta \mathrm{u}^{(m)}$ which deteriorates the nonlinear convergence behavior [235, 253]. This phenomenon is typically known as oversolving [71]. Indeed, an inappropriate intermediate solution $\mathrm{u}^{(m)}$ may even crash the simulation if no precautions are taken to prevent divergence.

A common practice is to define the forcing term $\eta^{(m)} \in [0,1)$ *a priori* and require that the solution increment $\Delta \mathrm{u}^{(m)}$ be computed accurately enough by the inner solver so that

$$\| r^{(m)} - P(\mathrm{u}^{(m)}) \Delta \mathrm{u}^{(m)} \| \leq \eta^{(m)} \| r^{(m)} \| \tag{3.3.85}$$

The left-hand side of the above inequality is both the residual of the linear subproblem (3.3.84) and the model of the equation at hand linearized at the most recent state $\mathrm{u}^{(m)}$. Thus, by reducing the linear residual to satisfy (3.3.85), one will also improve the nonlinear residual provided that the truncated Taylor expansion constitutes a reasonable linearization of the nonlinear problem.

Several approaches for choosing the forcing term in an adaptive fashion are considered by Eisenstat and Walker [70, 71]. A viable strategy is to choose $\eta^{(0)} \in [0,1)$ and adopt

$$\eta^{(m+1)} = \frac{\left| \| r^{(m+1)} \| - \| r^{(m)} - P(\mathrm{u}^{(m)}) \Delta \mathrm{u}^{(m)} \| \right|}{\| r^{(m)} \|}, \qquad m = 0, 1, 2, \ldots \tag{3.3.86}$$

which reflects the agreement between the nonlinear residual and its local linear model at the last iterate. If there is a great discrepancy, then the forcing term will become large so that the linear solver will not spend too much time to find an accurate approximation to the Newton step in the next iteration. Conversely, a small value of $\eta^{(m+1)}$ indicates a good agreement between the residual and its linearized model, and moreover, it forces the linear solver to approximate the Newton step very accurately. As an alternative, the forcing term can be updated as follows [70]

$$\eta^{(m+1)} = \gamma \left(\| r^{(m+1)} \| / \| r^{(m)} \| \right)^{\alpha}, \qquad m = 0, 1, 2, \ldots \tag{3.3.87}$$

where the coefficient γ may vary between zero and one and the exponent satisfies $\alpha \in (1,2]$.

Although the use of an adaptive forcing term determined by (3.3.86) or (3.3.87) is usually effective to avoid oversolving, $\eta^{(m+1)}$ may occasionally become too small far away from a solution. As a remedy, additional safeguards [71] may be used to prevent the rapid reduction of forcing terms. In essence, $\eta^{(m+1)}$ is required to be no less than a prescribed minimum value $\eta_{\min}^{(m+1)}$ which

is given by its predecessor $\eta^{(m)}$ raised to a power associated with the expected rate of convergence. In fact, the lower bound must also exceed some fixed threshold in order to be effective. This mechanism overrides the safeguard in the vicinity of solutions and gives rise to the asymptotic convergence. In a practical implementation, the value of forcing terms is thus predicted from expressions (3.3.86) or (3.3.87) and corrected by imposing the following safeguard [71]

$$\eta^{(m+1)} := \max\left\{\eta^{(m+1)}, \eta_{\min}^{(m+1)}\right\} \qquad \text{if} \quad \eta_{\min}^{(m+1)} > \text{tol} \tag{3.3.88}$$

The threshold $\text{tol} = 0.1$ was adopted in all our experiments which is somewhat arbitrary but works well in practice. Finally, one has to take care that the forcing term is strictly bounded from unity.

Following theoretical considerations by Eisenstat and Walker, the minimum values for safeguarding the forcing terms computed from (3.3.86) and (3.3.87) are given by [71]

$$\eta_{\min}^{(m+1)} = \left[\eta^{(m)}\right]^{(1+\sqrt{5})/2} \qquad \text{and} \qquad \eta_{\min}^{(m+1)} = \gamma\left[\eta^{(m)}\right]^{\alpha} \tag{3.3.89}$$

respectively. The initial value of the forcing term is defined as $\eta^{(0)} = 0.5$ and the auxiliary parameters $\gamma = 0.5$ and $\alpha = (1+\sqrt{5})/2$ are used in all numerical examples [71].

Once the increment for the fixed-point procedure (3.3.84) has been determined by the linear solver, $\Delta \mathrm{u}^{(m)}$ can be used to update the last iterate. Since Newton's method is prone to diverge for inaccurate starting values and/or improper intermediate solutions, an appropriate globalization technique is mandatory. To this end the provisional increment is scaled by some relaxation factor s so that the new iterate $\mathrm{u}^{(m+1)} = \mathrm{u}^{(m)} + s\Delta \mathrm{u}^{(m)}$ satisfies the sufficient decrease condition [70]

$$\|r^{(m+1)}\| \leq [1 - \xi(1-\eta^{(m)})]\|r^{(m)}\| \tag{3.3.90}$$

where $\xi \in (0,1)$ represents the prescribed tolerance. In practice, line search and trust region methods are frequently employed to compute acceptable multipliers [188, 201].

In a practical implementation, backtracking [235] can be used to compute the step length reduction factor s which is supposed to lie in the safeguard interval $[s_{\min}, s_{\max}]$. The basic idea is to minimize the function $g(s) = \frac{1}{2}\|r(\mathrm{u}^{(m)} + s\Delta \mathrm{u}^{(m)})\|^2$ with respect to s. Since it can be expensive to solve this minimization problem exactly, $g(s)$ is interpolated by the quadratic polynomial [238]

$$\lambda(s) = \ [g(1) - g(0) - g'(0)]s^2 + g'(0)s + g(0) \tag{3.3.91}$$

which is found by requiring $\lambda(0) = g(0)$, $\lambda(1) = g(1)$ and $\lambda'(0) = g'(0)$. Imposing the necessary condition for a local extremum, i.e. $\lambda'(s) = 0$, yields a possible candidate for the step length

$$s = \frac{-g'(0)}{2[g(1) - g(0) - g'(0)]} \tag{3.3.92}$$

If $\lambda''(s) \leq 0$ then the quadratic model is concave down so that $s = s_{\max}$ is the best choice. On the other hand, $\lambda''(s) > 0$ implies that (3.3.92) is a valid minimizer of the interpolation function. Starting from the initial value $\Delta \mathrm{u}^{(m)}$, the solution increment is updated by $\Delta \mathrm{u}^{(m)} := s\Delta \mathrm{u}^{(m)}$ and the forcing term is adjusted accordingly, i.e. $\eta^{(m)} := 1 - s(1-\eta^{(m)})$. If the sufficient decrease condition (3.3.90) is still not satisfied by the new quantities, a new multiplier s is determined from expression (3.3.92) and the reduction process is repeated. The backtracking algorithm is stopped once inequality (3.3.90) holds or if a maximum number of reduction steps or a minimum value for the net reduction has been reached. If no acceptable solution can be determined by the backtracking algorithm, then the Newton update is rejected and the m^{th} step of the fixed-point iteration (3.3.84) is repeated using a less cumbersome preconditioner, e.g., $P = M_L - \theta \Delta t L$.

Algorithm 3.1: Inexact Newton backtracking method.

Given: $\mathrm{u}^{(0)}$, $\eta^{(0)} \in [0,1)$, $\xi \in (0,1)$, $0 < s_{\min} < s_{\max} < 1$

Result: Converged solution $\mathrm{u}^{(m+1)}$

1 **for** $m = 0,1,2,\ldots$ *(until convergence)* **do**
2 Solve the linear problem
3 $P(\mathrm{u}^{(m)})\Delta\mathrm{u}^{(m)} = r^{(m)}$
4 such that
5 $\|r^{(m)} - P(\mathrm{u}^{(m)})\Delta\mathrm{u}^{(m)}\| \leq \eta^{(m)}\|r^{(m)}\|$
6 Compute $\mathrm{u}^{(m+1)} = \mathrm{u}^{(m)} + \Delta\mathrm{u}^{(m)}$
7 **while** $\|r^{(m+1)}\| > [1-\xi(1-\eta^{(m)})]\|r^{(m)}\|$ **do**
8 Choose $s \in [s_{\min}, s_{\max}]$ from equation (3.3.92)
9 Update $\Delta\mathrm{u}^{(m)} := s\Delta\mathrm{u}^{(m)}$ and $\eta^{(m)} := 1 - s(1-\eta^{(m)})$
10 end compute new solution $\mathrm{u}^{(m+1)} = \mathrm{u}^{(m)} + \Delta\mathrm{u}^{(m)}$
11 **end**
12 **if** *Backtracking failed* **then**
13 Recompute $P(\mathrm{u}^{(m)})\Delta\mathrm{u}^{(m)} = r^{(m)}$ with different preconditioner
14 and accept solution $\mathrm{u}^{(m+1)} = \mathrm{u}^{(m)} + \Delta\mathrm{u}^{(m)}$ unconditionally
15 **end**
16 Check for convergence
17 $\frac{\|r^{(m+1)}\|}{\|r^{(0)}\|} < \mathrm{tol}_{\mathrm{abs}}$ or $\frac{\|\mathrm{u}^{(m+1)}-\mathrm{u}^{(m)}\|}{\|\mathrm{u}^{(m)}\|} < \mathrm{tol}_{\mathrm{rel}}$
18 **end**

In summary, an inexact Newton backtracking method [70] can be implemented as presented in Algorithm 3.1. The use of some forcing strategy (see line 6) is advisable to prevent oversolving in the linear iterations. On the other hand, globalization is mandatory since inaccurate starting values or unacceptable intermediate solutions would directly lead to divergence of Newton's method. A simple backtracking strategy can be employed to scale the predicted solution update $\Delta\mathrm{u}^{(m)}$ so as to satisfy the sufficient decrease condition (lines 8–11). If backtracking fails, then the outer iteration step is repeated adopting the low-order evolution operator as preconditioner.

3.3.9. Summary of upwind-biased flux limiting of TVD type

Multidimensional upwind-biased flux limiting schemes can be readily implemented in a finite element code [145]. All the necessary information is extracted from the original matrix K and there is no need to know coordinates of nodes or any other geometric details. Since the origin of the discrete transport operator K is immaterial, algebraic flux correction is also applicable to finite difference/volume discretizations if they can be represented as a DAE system of the form (3.2.25).

The node-based limiting proposed by Kuzmin and Turek yields the high-resolution method (3.3.8) and (3.3.10), respectively. Although some off-diagonal entries of the modified transport operator K^* may be negative, the two-sided estimate (3.3.7) guarantees the existence of an equivalent operator L^* which proves that the semi-discrete scheme is local extremum diminishing. After the discretization in time by the two-level θ-scheme, the numerical method (3.3.74) is positivity-preserving if it satisfies Theorem 3.1.3. This extra requirement yields computable bounds for admissible time steps. The generalized TVD algorithm (see Figures 3.2) gives rise to the nonlinear algebraic systems that call for an iterative solution strategy. The low-order evolution operator (3.3.78) constitutes a viable preconditioner which exhibits amenable matrix properties. It is easy

to invert and does not depend on the solution for linear problems so that it can be assembled once at the beginning of the simulation and stored for subsequent usage. The design of an alternative preconditioner may be based on Newton's linearization (3.3.81), whereby the approximate Jacobian matrix (3.3.83) can be efficiently assembled edge-by-edge (cf. Sections 3.3.4 and 3.3.5). The sparsity pattern of the underlying finite element matrix needs to be extended by one connectivity layer which can be achieved by means of symbolic matrix multiplication (3.3.71).

Upwind-biased flux limiting [145] is derived on the semi-discrete level, and hence, it can be directly applied to steady-state problems of the form (3.3.28). However, the M-matrix property of the preconditioner (3.3.32) may be irrecoverably lost since discrete upwinding can render the low-order transport operator singular due to the presence of zero rows. As a remedy, the use of pseudo time-stepping may be preferable for the computation of stationary flows. In particular, the pseudo time step can be adjusted according to constraint (3.1.38) to enforce the M-matrix property of the low-order evolution operator (3.3.79). Furthermore, the upper bound (3.1.34) is unconditionally satisfied for the fully implicit backward Euler method so that large time steps can be employed unless iterative solvers fail to converge. Newton's method can also be applied to the steady-state formulation (3.3.29), whereby the choice of a good initial guess is crucial. It may be worthwhile to perform a few pre-iteration steps adopting the standard defect correction scheme (3.3.30) preconditioned by the low-order operator (3.3.32) prior to resorting to the approximate Jacobian matrix (3.3.35). In all nonlinear solution algorithms, the inversion of the preconditioner is performed by solving the sequence of linear subproblems (3.3.77), e.g., with aid of a Krylov subspace method. In each Newton step, the accuracy of linear problems can be monitored by a suitable forcing strategy so as to prevent 'oversolving'. Moreover, Newton's method can be equipped with a simple backtracking procedure which is designed to ensure global convergence. The performance of the proposed algorithms will be illustrated by numerical examples for two-dimensional benchmark problems to be presented at the end of this chapter.

3.4. Symmetric limiters of FCT type

As pointed out in Section 3.2.5, the difference between the high- and low-order residuals (3.2.44) consists of convective antidiffusion proportional to $-D\mathrm{u}$ and of the contribution $(M_L - M_C)\frac{\mathrm{du}}{\mathrm{d}t}$. The latter has been intentionally neglected in the design of upwind-biased flux limiters since no genuine flow direction (upwind/downwind) is available for the mass diffusion operator $M_C - M_L$. On the other hand, mass lumping degrades the phase accuracy of the numerical method, and thus, a significant advantage of finite element schemes over finite difference and finite volume methods being applied to truly time-dependent problems is lost. The fully discretized mass flux [134]

$$f_{ij}^m = \frac{m_{ij}}{\Delta t}(\mathrm{u}_i^{n+1} - \mathrm{u}_j^{n+1}) - \frac{m_{ij}}{\Delta t}(\mathrm{u}_i^n - \mathrm{u}_j^n) \tag{3.4.1}$$

consists of an implicit part which is truly antidiffusive and a diffusive explicit part which has a strong damping effect. Kuzmin suggested a symmetric flux limiter [134] to include the consistent mass matrix in a positivity-preserving fashion and combined it with upwind-biased flux limiting [145] for the convective antidiffusive. This so-called general-purpose flux limiter [134] can be used to compute accurate solutions to time-dependent problems and it is capable of producing sharply resolved profiles for stationary flows. Thus, if the temporal evolution of the simulation is unknown, the general-purpose flux limiter may be employed without sacrificing the consistent mass matrix from the outset, and at the same time, the existence of convective antidiffusion prevents a loss of accuracy in the stationary limit. This flexibility comes at the cost of a more expensive procedure for evaluating the nodal quantities $P_i^{\pm}$, $Q_i^{\pm}$ and $R_i^{\pm}$ which need to be rebuilt in each outer iteration. Moreover, the nonlinear convergence rates deteriorate significantly for larger time steps so that steady-state computations may become time consuming.

3.4.1. Nonlinear FEM-FCT algorithm

Essentially the same limitations apply to early generalizations of Zalesak's multidimensional flux limiter [268] to implicit FCT algorithms for finite element discretizations on unstructured meshes [133, 144]. Some improvements and extensions have been proposed in a series of subsequent publications [141, 142] but there remain critical issues that need to be addressed:

- The correction factors α_{ij} produced by Zalesak's limiter depend on the time step which deteriorates the accuracy of most FCT algorithms as Δt increases.
- Classical FCT algorithms require the re-evaluation of sums of antidiffusive fluxes $P_i^{\pm}$, of admissible upper and lower bounds $Q_i^{\pm}$ and of the nodal correction factors $R_i^{\pm}$ in each outer iteration which accumulates the computational costs and leads to an extended stencil of the approximate Jacobian if Newton's linearization is employed.

The first drawback has been partially alleviated by the advent of iterative flux correction [142] which is designed to recycle the rejected antidiffusion step-by-step. On the other hand, the necessary adjustment of correction factors α_{ij} in each outer iteration increases the computational costs and deteriorates the nonlinear convergence behavior significantly. The second issue has been investigated by Kuzmin and Kourounis [137] who presented a semi-implicit FEM-FCT algorithm for the simulation the treatment of time-dependent flows. In order to increase the efficiency of the semi-implicit flux correction scheme, it has been equipped with a discrete Newton method [187]. Recently, Kuzmin [136] presented a novel approach for the design of explicit and implicit FEM-FCT schemes based on the linearization of antidiffusive fluxes.

It is expedient to introduce the underlying idea of flux correction algorithms in a more general framework and present the semi-implicit FCT limiter in the next section. Let the semi-discrete method (3.2.47) be discretized in time by the two-level θ-scheme [144]

$$A\mathrm{u}^{n+1} = B\mathrm{u}^n + \bar{f}(\mathrm{u}^{n+1}, \mathrm{u}^n) \tag{3.4.2}$$

The operator $A = M_L - \theta\Delta t L$ is designed to be an M-matrix which may involve some restrictions to admissible time steps (see constraint (3.1.38)). Furthermore, matrix $B = M_L + (1-\theta)\Delta t L$ can be rendered non-negative by adjusting the time step according to estimate (3.1.34) and it is unconditionally non-negative for the fully implicit backward Euler method ($\theta = 1$). The last term in the right-hand side of equation (3.4.2) is assembled from skew-symmetric internodal fluxes

$$\bar{f}_i = \sum_{j \neq i} \alpha_{ij} f_{ij}, \qquad f_{ji} = -f_{ij} \tag{3.4.3}$$

whereby the fully discrete counterparts of the raw antidiffusive fluxes (3.2.46) are given by [144]

$$f_{ij} = [m_{ij} + \theta\Delta t d_{ij}^{n+1}](\mathrm{u}_i^{n+1} - \mathrm{u}_j^{n+1}) - [m_{ij} - (1-\theta)\Delta t d_{ij}^n](\mathrm{u}_i^n - \mathrm{u}_j^n) \tag{3.4.4}$$

The nonlinearity of equation (3.4.2) inherent to the governing equation and/or engendered by algebraic flux correction calls for the use of an iterative solution strategy. Successive approximations to the end-of-step solution u^{n+1} can be computed by the fixed-point iteration scheme

$$\mathrm{u}^{(m+1)} = \mathrm{u}^{(m)} + [P(\mathrm{u}^{(m)})]^{-1} r^{(m)}, \qquad m = 0, 1, 2, \ldots \tag{3.4.5}$$

which is initialized by the solution from the last time step, i.e. $\mathrm{u}^{(0)} = \mathrm{u}^n$. The residual vector

$$r^{(m)} = b^n + \bar{f}(\mathrm{u}^{(m)}, \mathrm{u}^n) - A\mathrm{u}^{(m)}, \qquad b^n = B\mathrm{u}^n \tag{3.4.6}$$

needs to be updated in each outer iteration, whereby the net right-hand side $b^{(m)} = b^n + \bar{f}(\mathrm{u}^{(m)}, \mathrm{u}^n)$ consists of the constant low-order part augmented by limited antidiffusion [142, 144].

A natural choice for the preconditioner P is the monotone evolution operator $A = M_L - \theta\Delta t L$ which turns expression (3.4.5) into the standard fixed-point defect correction scheme [255]

$$A\mathrm{u}^{(m+1)} = b^{(m)}, \qquad m = 0,1,2,\ldots \tag{3.4.7}$$

To satisfy the positivity constraint 3.1.3, it suffices to define a positivity-preserving auxiliary solution $\tilde{\mathrm{u}} \geq 0$ and a matrix $B(\tilde{\mathrm{u}}) = \{b_{ij}\}$ so that the right-hand side can be written as follows [144]

$$b^{(m)} = B(\tilde{\mathrm{u}})\tilde{\mathrm{u}} \qquad \text{and} \qquad b_{ij} \geq 0, \quad \forall i,j \tag{3.4.8}$$

The non-negative matrix $B(\tilde{\mathrm{u}})$ does not need to be constructed explicitly, but its existence and the M-matrix property of A guarantee that the positivity of the right-hand side is preserved, whence

$$\tilde{\mathrm{u}} \geq 0 \quad \Rightarrow \quad b^{(m)} \geq 0 \quad \Rightarrow \quad \mathrm{u}^{(m+1)} = A^{-1}b^{(m)} \geq 0 \tag{3.4.9}$$

An explicit low-order approximation to $\mathrm{u}(t^{n+1-\theta})$ can be computed by solving the auxiliary subproblem $M_L\tilde{\mathrm{u}} = B\mathrm{u}^n$ for the positivity-preserving intermediate solution [144]

$$\tilde{\mathrm{u}} = \mathrm{u}^n + (1-\theta)\Delta t M_L^{-1} L\mathrm{u}^n \tag{3.4.10}$$

The task of the flux limiter is to determine suitable correction factors $\alpha_{ij} \in [0,1]$ which ensure that each nodal component of the updated right-hand side remains non-negative [144]

$$b_i^{(m)} = m_i\tilde{\mathrm{u}}_i + \sum_{j\neq i}\alpha_{ij}f_{ij} \geq 0 \qquad \text{if} \quad \tilde{\mathrm{u}}_j \geq 0, \quad \forall j \tag{3.4.11}$$

A viable choice is Zalesak's multidimensional FCT limiter [268] which has been successfully employed to construct flux corrected algorithms for finite element methods [137, 142, 144].

3.4.2. Semi-implicit FCT limiter

Some drawbacks of the semi-explicit FCT algorithm [133, 144] have been remedied by the advent of semi-implicit flux correction [137]. It is of predictor-corrector type, whereby the costly evaluation of nodal correction factors is only required once per time step. As a tribute to Zalesak's limiter [268], Kuzmin and Kourounis adopted the same notation for the nodal quantities $P_i^{\pm}$, $Q_i^{\pm}$ and $R_i^{\pm}$ which are initialized by zeros before the first outer iteration ($m = 0$). They can be updated following the three-step algorithm presented in Figure 3.6 and used to predict a set of limited fluxes $\tilde{f}_{ij}$ which serve as an explicit estimate for the maximum amount of admissible antidiffusion.

Note that the antidiffusive flux f_{ij}^n is not the real target flux (3.4.4) but merely an explicit predictor which is recovered in the first iteration ($\mathrm{u}^{(0)} = \mathrm{u}^n$) from the antidiffusive flux [137]

$$f_{ij}^{(m)} = \left[m_{ij} + \theta\Delta t d_{ij}^{(m)}\right]\left(\mathrm{u}_i^{(m)} - \mathrm{u}_j^{(m)}\right) - \left[m_{ij} - (1-\theta)\Delta t d_{ij}^n\right]\left(\mathrm{u}_i^n - \mathrm{u}_j^n\right) \tag{3.4.12}$$

In the worst case, all antidiffusive fluxes into node i have the same sign. Therefore, it is worthwhile to treat the positive and negative ones separately [268]. The maximum/minimum admissible increment depends on the solution values at all neighboring nodes that share an element with i. The quantities $Q_i^{\pm}$ defined in (3.4.15) measure the distance to a local maximum/minimum of the low-order solution (3.4.10), and hence, they provide upper/lower bounds for the nodal values [137]

$$\tilde{\mathrm{u}}_i^{\max} = \tilde{\mathrm{u}}_i + Q_i^+, \qquad \tilde{\mathrm{u}}_i^{\min} = \tilde{\mathrm{u}}_i + Q_i^- \tag{3.4.13}$$

Semi-implicit flux correction [137] follows the two-step algorithm presented in Figure 3.7 which is invoked in each outer iteration. The solution from the previous iteration is used to evaluate

Before the first outer iteration ($m=0$), compute a set of antidiffusive fluxes $\tilde{f}_{ij}$ which provide an explicit estimate for the admissible magnitude of $\bar{f}_{ij}=\alpha_{ij}f_{ij}^{(m)}$

1. Compute the *sums of positive and negative antidiffusive fluxes* $f_{ij}^n=\Delta t d_{ij}^n(\mathrm{u}_i^n-\mathrm{u}_j^n)$

$$P_i^+=\sum_{j\neq i}\max\{0,f_{ij}^n\},\qquad P_i^-=\sum_{j\neq i}\min\{0,f_{ij}^n\} \tag{3.4.14}$$

2. Define the *upper and lower bounds* for admissible increments

$$Q_i^+=\max\{0,\max_{j\neq i}\{\tilde{\mathrm{u}}_j-\tilde{\mathrm{u}}_i\}\},\qquad Q_i^-=\min\{0,\min_{j\neq i}\{\tilde{\mathrm{u}}_j-\tilde{\mathrm{u}}_i\}\} \tag{3.4.15}$$

3. Evaluate the *nodal correction factors* and perform flux limiting

$$R_i^\pm=\begin{cases} m_iQ_i^\pm/P_i^\pm & \text{if} \quad P_i^\pm\neq 0 \\ 1 & \text{if} \quad P_i^\pm=0 \end{cases}\qquad \tilde{f}_{ij}=\begin{cases} \min\{R_i^+,R_j^-\}f_{ij}^n & \text{if} \quad f_{ij}^n>0 \\ \min\{R_i^-,R_j^+\}f_{ij}^n & \text{if} \quad f_{ij}^n\leq 0 \end{cases} \tag{3.4.16}$$

Fig. 3.6. Semi-implicit FCT algorithm: predictor step.

the target flux (3.4.12) which is limited based on the precomputed quantities $\tilde{f}_{ij}$. In contrast to Zalesak's classical limiter, the nodal correction factors $R_i^\pm$ are allowed to exceed unity which carries over to the flux ratio $\tilde{f}_{ij}/f_{ij}^n$, but the effective correction factors [137]

$$\alpha_{ij}:=\bar{f}_{ij}/f_{ij}^{(m)} \tag{3.4.17}$$

are bounded by 0 and 1, as required for consistency. Kuzmin and Kourounis [137] give the positivity proof for the semi-implicit FCT algorithm (3.4.14)–(3.4.19) which follows the one for Zalesak's limiter [144]. For the sake of completeness it is presented in what follows. In the non-trivial case $\bar{f}_{ij}\neq 0$, the net antidiffusion received by node i does not vanish so that the i^{th} component of

At each outer iteration ($m=0,1,2,\ldots$), assemble $\bar{f}_{ij}$ and plug it into the right-hand side (3.4.6)

1. Constrain the updated target flux $f_{ij}^{(m)}$ so that its magnitude is bounded by that of $\tilde{f}_{ij}$

$$\bar{f}_{ij}=\begin{cases} \min\{f_{ij}^{(m)},\max\{0,\tilde{f}_{ij}\}\} & \text{if} \quad f_{ij}^{(m)}>0 \\ \max\{f_{ij}^{(m)},\min\{0,\tilde{f}_{ij}\}\} & \text{if} \quad f_{ij}^{(m)}\leq 0 \end{cases} \tag{3.4.18}$$

2. Insert the limited antidiffusive flux $\bar{f}_{ij}$ into the right-hand side

$$b_i^{(m)}:=b_i^{(m)}+\bar{f}_{ij},\qquad b_j^{(m)}:=b_j^{(m)}-\bar{f}_{ij} \tag{3.4.19}$$

Fig. 3.7. Semi-implicit FCT algorithm: corrector step.

the right-hand side admits the following representation

$$\bar{f}_i := \sum_{j\neq i} \bar{f}_{ij} \neq 0 \quad \Rightarrow \quad b_i^{(m)} = m_i \tilde{\mathrm{u}}_i + \bar{f}_i = (m_i - \alpha_i)\tilde{\mathrm{u}}_i + \alpha_i \tilde{\mathrm{u}}_k \tag{3.4.20}$$

The *nodal* coefficient α_i is defined in terms of the local extremum (3.4.13) of the low-order solution

$$\alpha_i := \frac{\bar{f}_i}{\tilde{\mathrm{u}}_k - \tilde{\mathrm{u}}_i}, \qquad \tilde{\mathrm{u}}_k = \begin{cases} \tilde{\mathrm{u}}_i^{\max} & \text{if} \quad \bar{f}_i > 0 \\ \tilde{\mathrm{u}}_i^{\min} & \text{if} \quad \bar{f}_i < 0 \end{cases} \tag{3.4.21}$$

which implies that $\alpha_i > 0$, and moreover, the limited antidiffusion satisfies $\bar{f}_i = \alpha_i Q_i^{\pm}$.

It follows from reformulation (3.4.20) of the right-hand side that the sign of the intermediate solution $\tilde{\mathrm{u}}$ is preserved provided the inequality $m_i - \alpha_i \geq 0$ holds for all nodes i. In the negative case $\bar{f}_i < 0$, the antidiffusive correction to node i is bounded from below by

$$m_i Q_i^- \leq R_i^- P_i^- \leq \sum_{j\neq i} \min\{0, \tilde{f}_{ij}\} \leq \bar{f}_i = \alpha_i Q_i^- \tag{3.4.22}$$

Conversely, there exists an upper bound for each strictly positive contribution $\bar{f}_i > 0$

$$\alpha_i Q_i^+ = \bar{f}_i \leq \sum_{j\neq i} \max\{0, \tilde{f}_{ij}\} \leq R_i^+ P_i^+ \leq m_i Q_i^+ \tag{3.4.23}$$

Combining both estimates, the inequality $0 \leq \alpha_i \leq m_i$ holds, which proves that the i^{th} component of the right-hand side (3.4.20) remains non-negative provided that $\tilde{\mathrm{u}}_i \geq 0$ and $\tilde{\mathrm{u}}_k \geq 0$.

Putting together what we have said so far, the semi-implicit FCT algorithm is positivity-preserving as long as the diagonal coefficients of matrix $B = M_L + (1-\theta)\Delta t L$ are non-negative. This requirement is unconditionally satisfied for the fully implicit backward Euler method ($\theta = 1$) and the largest admissible time step can be computed from the upper bound (3.1.34) otherwise.

3.4.3. Approximation of the Jacobian matrix for FEM-FCT schemes

As an efficient alternative to the fixed-point defect correction scheme (3.4.5), the nonlinear algebraic system of equation (3.4.2) can be linearized by means of Newton's method. Thus, the difference between two successive approximations is related to the residual vector as follows [187]

$$\left.\begin{aligned} P(\mathrm{u}^{(m)})\Delta\mathrm{u}^{(m)} &= r^{(m)} \\ \mathrm{u}^{(m+1)} &= \mathrm{u}^{(m)} + \Delta\mathrm{u}^{(m)} \end{aligned}\right\} \qquad P(\mathrm{u}^{(m)}) = M_L - \theta\Delta t J(\mathrm{u}^{(m)}) - \bar{J}(\mathrm{u}^{(m)}) \tag{3.4.24}$$

whereby the solution from the previous time step serves as an excellent initial guess, i.e. $\mathrm{u}^{(0)} = \mathrm{u}^n$. The solution-dependent preconditioner P comprises the approximate Jacobian operators

$$J(\mathrm{u}^{(m)}) \approx J_L(\mathrm{u}^{(m)}), \qquad \bar{J}(\mathrm{u}^{(m)}) \approx J_{\bar{f}}(\mathrm{u}^{(m)}) \tag{3.4.25}$$

A viable strategy for the calculation of the 'upwind' Jacobian J_L which corresponds to the derivative of the low-order contribution $L(\mathrm{u})\mathrm{u}$ has been described in Section 3.3.4. It is only required for nonlinear conservation laws and reduces to the low-order transport operator $L = K + D$ otherwise.

It remains to devise and efficient algorithm for approximating the Jacobian matrix $J_{\bar{f}}$ which results from the application of the semi-implicit flux correction procedure (cf. Figures 3.6 and 3.7). Since the predictor step (3.4.14)–(3.4.16) is only based on explicit contributions, the estimated fluxes $\tilde{f}_{ij}$ are available from the residual assembly at no additional cost. The k^{th} column of the approximate Jacobian $\bar{J}$ can be assembled by performing the corrector steps (3.4.18) and

At each outer iteration ($m = 0, 1, 2, \ldots$), initialize $\bar{J} \equiv 0$ and rebuild it in a loop over edges ij.

1. Evaluate the explicit antidiffusive contribution

$$f_{ij}^{\text{exp}} = \left[m_{ij} - (1-\theta)\Delta t d_{ij}^n\right](\mathrm{u}_i^n - \mathrm{u}_j^n) \tag{3.4.26}$$

*Define the difference $\Delta \mathrm{u}_{ij} := \mathrm{u}_i^{(m)} - \mathrm{u}_j^{(m)}$ and perform the following steps for $k = i$ **and** $k = j$*

2. Compute auxiliary coefficients based on the perturbed solution $\mathrm{u}^{\pm} := \mathrm{u}^{(m)} \pm \sigma \mathrm{e}_k$

$$a_{ij}^+ = m_{ij} + \theta \Delta t d_{ij}[\mathrm{u}^+], \qquad a_{ij}^- = m_{ij} + \theta \Delta t d_{ij}[\mathrm{u}^-] \tag{3.4.27}$$

3. Update the perturbed target fluxes (3.4.12) depending on the index k

$$\begin{aligned} &\text{if } k = i: \quad f_{ij}^+ = a_{ij}^+(\Delta \mathrm{u}_{ij} + \sigma), \qquad f_{ij}^- = a_{ij}^-(\Delta \mathrm{u}_{ij} - \sigma) \\ &\text{if } k = j: \quad f_{ij}^+ = a_{ij}^+(\Delta \mathrm{u}_{ij} - \sigma), \qquad f_{ij}^- = a_{ij}^-(\Delta \mathrm{u}_{ij} + \sigma) \end{aligned} \qquad f_{ij}^{\pm} := f_{ij}^{\pm} + f_{ij}^{\text{exp}} \tag{3.4.28}$$

4. Constrain each flux $f_{ij}^{\pm}$ so that its magnitude is bounded by that of $\tilde{f}_{ij}$

$$\bar{f}_{ij}^+ = \begin{cases} \min\{f_{ij}^+, \max\{0, \tilde{f}_{ij}\}\} & \text{if} \quad f_{ij}^+ > 0 \\ \max\{f_{ij}^+, \min\{0, \tilde{f}_{ij}\}\} & \text{if} \quad f_{ij}^+ \le 0 \end{cases} \tag{3.4.29}$$

$$\bar{f}_{ij}^- = \begin{cases} \min\{f_{ij}^-, \max\{0, \tilde{f}_{ij}\}\} & \text{if} \quad f_{ij}^- > 0 \\ \max\{f_{ij}^-, \min\{0, \tilde{f}_{ij}\}\} & \text{if} \quad f_{ij}^- \le 0 \end{cases} \tag{3.4.30}$$

5. Compute the divided difference and insert it into the k^{th} column of the Jacobian matrix $\bar{J}$

$$\bar{f}_{ij} = \frac{\bar{f}_{ij}^+ - \bar{f}_{ij}^-}{2\sigma} \qquad \mathrm{J}_{ik} := \mathrm{J}_{ik} - \bar{f}_{ij}, \qquad \mathrm{J}_{jk} := \mathrm{J}_{jk} + \bar{f}_{ij} \tag{3.4.31}$$

Fig. 3.8. Assembly of Jacobian matrix for the semi-implicit FCT algorithm.

(3.4.19) based on the perturbed solution vectors $\mathrm{u} + \sigma \mathrm{e}_k$ and $\mathrm{u} - \sigma \mathrm{e}_k$ so as to evaluate the limited antidiffusion $\bar{f}^+ = \bar{f}(\mathrm{u} + \sigma \mathrm{e}_k)$ and $\bar{f}^- = \bar{f}(\mathrm{u} - \sigma \mathrm{e}_k)$, respectively. Their difference can be scaled by $1/2\sigma$ and inserted into the corresponding positions in row k. However, this approach is prohibitively expensive since it does not exploit the sparsity of the Jacobian matrix. Let us emphasize the fact that the correction factors (3.4.17) are only defined 'virtually', and therefore, they do not entail a widening of the matrix stencil. Thus, the connectivity graph of the finite element matrix carries over to that of the Jacobian operator $\bar{J}$. This is in contrast to the semi-explicit FCT limiter [133, 144], where the use of correction factors $\alpha_{ij} \in [0, 1]$ yields an extended sparsity pattern which is identical to the one obtained for upwind-biased flux limiting schemes (cf. Section 3.3.6).

In light of the above, the semi-implicit FCT algorithm allows for an efficient edge-based assembly of the operator $\bar{J}$ which can be implemented as depicted in Figure 3.8. The implicit part of the target flux (3.4.12) gives rise to the following possibilities of perturbed solution differences

$$\Delta \mathrm{u}_{ij} \pm (\delta_{ik} - \delta_{jk}), \qquad j \neq i, \quad k \in \{i, j\} \tag{3.4.32}$$

They are multiplied by the corresponding coefficients $a_{ij}^{\pm}$ which account for the consistent mass matrix and the contribution of the perturbed diffusion operator. The above algorithm is applicable to linear and nonlinear transport operators alike. In case of a linear transport operator, the diffusion coefficient d_{ij} and the auxiliary quantity $a_{ij} \equiv m_{ij} + \theta d_{ij}$ are not affected by nodal solution variations so that the perturbed fluxes exhibit the following symmetry property [187]

$$\begin{aligned} k = i: \quad & f_{ij}^{\pm} = a_{ij}(\Delta u_{ij} \pm \sigma) + f_{ij}^{\exp} \\ k = j: \quad & f_{ij}^{\mp} = a_{ij}(\Delta u_{ij} \pm \sigma) + f_{ij}^{\exp} \end{aligned} \tag{3.4.33}$$

The constrained fluxes (3.4.29) and (3.4.30), which are evaluated for the i^{th} column, can be applied to column j but with opposite sign. Hence, it suffices to compute the divided difference (3.4.31) for one index, say $k = i$, and update the four coefficients of the Jacobian matrix $\bar{J}$ simultaneously

$$\begin{aligned} \bar{\jmath}_{ii} &:= \bar{\jmath}_{ii} - \bar{f}_{ij}, \qquad & \bar{\jmath}_{ij} &:= \bar{\jmath}_{ij} + \bar{f}_{ij} \\ \bar{\jmath}_{ji} &:= \bar{\jmath}_{ji} + \bar{f}_{ij}, \qquad & \bar{\jmath}_{jj} &:= \bar{\jmath}_{jj} - \bar{f}_{ij} \end{aligned} \tag{3.4.34}$$

For nonlinear problems, the above symmetry property does not hold so that the contributions to columns i and j need to be evaluated separately following the algorithmic steps (3.4.28)–(3.4.31).

3.4.4. Summary of symmetric flux limiting of FCT type

The semi-implicit approach to flux correction of FCT type leads to a robust and efficient special-purpose algorithm for time-dependent problems. The accuracy of the resulting scheme improves as the time step is refined and the consistent mass matrix can be included in a positivity-preserving fashion. The new limiting strategy makes it possible to avoid repeated computations of the nodal correction factors at each outer iteration so that semi-implicit time-stepping methods prove competitive to their fully explicit counterparts which suffer from very restrictive CFL-like conditions. The use of an explicit predictor makes it possible to apply Newton's linearization without the need to extend the sparsity pattern of the finite element matrix. The Jacobian operator can be assembled edge-by-edge using divided differences applied to the low-order transport operator and to the vector of limited antidiffusive fluxes. For linear governing equations, the 'upwind' Jacobian J_L reduces to the low-order operator L, and moreover, the contribution $J_{\bar{f}}$ can be efficiently constructed exploiting the symmetry property (3.4.33). The semi-implicit FCT algorithm is not recommended for steady-state computations which call for the use of implicit time-stepping methods operated at large time steps. In this case, the consistent mass matrix can be safely neglected so that upwind-biased flux correction as presented in Section 3.3 is preferable.

If the flow dynamics is not known *a priori*, a general-purpose flux limiter can be devised by combining upwind-biased flux correction for the convective antidiffusion with a symmetric limiter for the transient contribution [134]. The resulting algorithm is more complicated, and moreover, severe convergence problems may occur if the the monotone evolution operator $M_L - \theta\Delta t L$ is employed in the fixed-point iteration scheme (3.4.5). The author performed some preliminary tests with a discrete Newton method applied to the general-purpose flux limiter which leads to an extended stencil for the approximate Jacobian operator. On the one hand, the number of nonlinear iterations could be reduced significantly as compared to the standard defect correction procedure. On the other hand, the assembly of the approximate Jacobian matrix is time consuming, and thus, impractical for transient flow computations which require the use of moderately small time steps due to temporal accuracy. In practice, it is advisable to implement two special-purpose limiters side by side, e.g., the semi-implicit FCT algorithm for transient flows and an upwind-biased flux limiter of TVD type suitable for steady-state computations, and control the relative solution variation between two successive time steps to switch between both methods.

3.5. Numerical examples for scalar conservation laws

In this section, we apply the presented high-resolution schemes to scalar benchmark problems for stationary and time-dependent flows. Some of these configurations will be reused in Section 4.7 to assess the ability of the novel mesh adaptation procedure to deal with dynamically changing flow fields. The main objective of this section is to investigate the behavior of iterative solution procedures as applied to nonlinear algebraic equations. In particular, the potential of discrete Newton methods is compared to that of the standard defect correction scheme in the context of the algebraic flux correction paradigm. Moreover, we analyze the influence of the different discretization techniques on the quality of the approximate solution and on the challenge to compute it numerically. It is beyond the scope of this thesis to give a quantitative analysis of AFC schemes and highlight individual advantages of this methodology. The reader who is interested in an in-depth coverage of this material is referred to the series of research papers [134, 136, 137, 139, 141, 142, 144, 145] and the forthcoming monograph by Kuzmin [132]. A rigorous comparison of various stabilized finite element methods for time-dependent convection dominated problems including the latest FEM-FCT algorithms [136] is presented by John and Schmeyer [119].

3.5.1. Stationary convection-diffusion

Algebraic flux correction schemes of TVD type are derived on the semi-discrete level [145], and hence, they can be applied to stationary problems directly or within the pseudo time-stepping approach. In the latter case, the solution is computed by adopting the transient counterpart of the equation at hand which is marched into a steady state limit. Since evolution details are immaterial, the fully implicit backward Euler method is preferable, whereby the artificial time step should be taken as large as possible to reduce the computational cost. As an alternative, the stationary problem can be solved directly, whereby the use of implicit under-relaxation [200] may be advisable to ensure convergence and improve robustness of the overall algorithm.

Test problem 1: Consider the stationary convection-diffusion equation in two dimensions

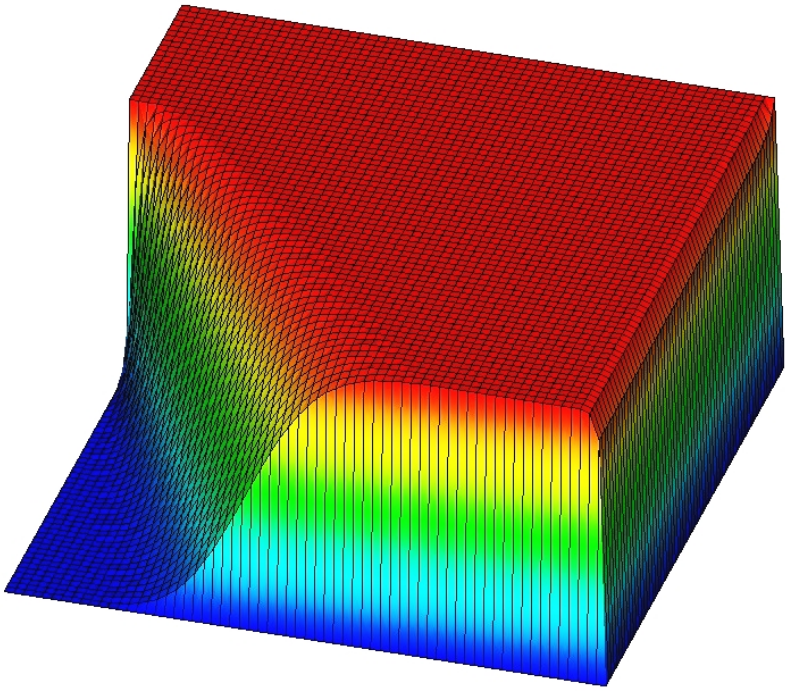
$$\mathbf{v}\cdot\nabla u - d\Delta u = 0 \qquad \text{in} \quad \Omega = (0,1)\times(0,1) \tag{3.5.1}$$

where $\mathbf{v} = (\cos(-\pi/3), \sin(-\pi/3))^T$ denotes the divergence-free velocity vector and the diffusion coefficient d =1.0e–8 is extremely small. The transient counterpart of equation (3.5.1) is addressed in Example 3.2.2. The concomitant boundary conditions of Dirichlet type read as follows

$$u(x,y) = \begin{cases} 0 & \text{if} \quad x = 1 \quad \text{or} \quad y \leq 0.7 \\ 1 & \text{otherwise} \end{cases} \tag{3.5.2}$$

This benchmark was proposed by Hughes et al. [110] and used by other authors to analyze the behavior of stabilization techniques applied to convection dominated flows, e.g. [117]. Note that the solution of the reduced problem ($d = 0$) does not satisfy the homogeneous Dirichlet boundary conditions, and hence, exponential boundary layers develop next to the lines $x = 1$ and $y = 0$. This is a common feature of singularly perturbed problems which are characterized by the presence of sharp fronts in the vicinity of the boundary. Moreover, the discontinuity which is prescribed along the line $x = 0$ produces an internal layer in the direction of the convection emanating from the point (0,0.7). For large physical diffusion coefficients, the steep front would be considerably smoothed but in the convection dominated case (d=1.0e–8) no smearing should be visible. The solutions depicted in Figure 3.9 were computed on the uniform *Grid 1* consisting of 64×64 bilinear finite elements. The upper profile (a) was obtained from the fully discrete counterpart of the low-order scheme (3.2.26), whereas the high-resolution TVD method (3.3.10) was used to produce Figure 3.9 (b). Both discretizations were applied directly without resorting to pseudo time-stepping.

(a) Low-order scheme: discrete upwinding



(b) High-resolution FEM-TVD scheme

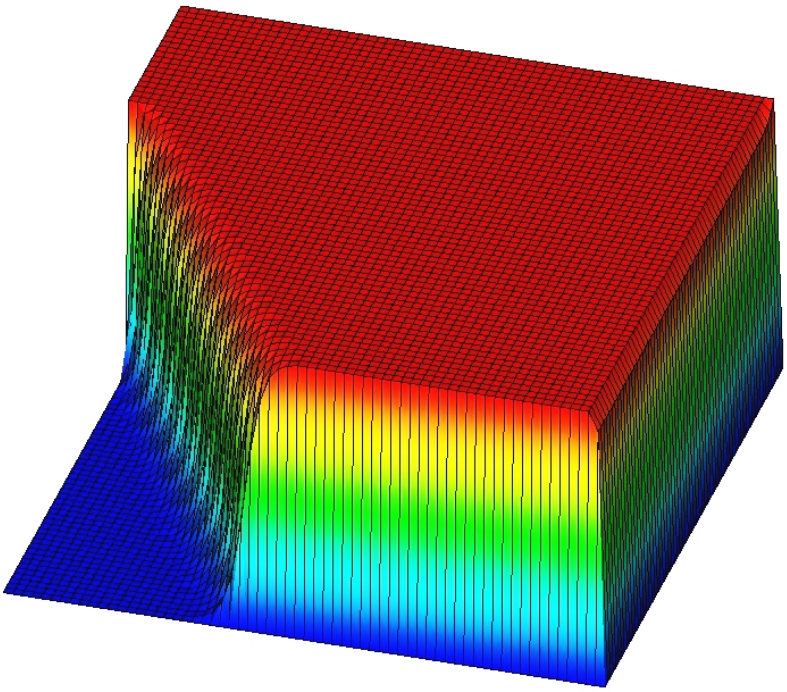


Fig. 3.9. Stationary convection-diffusion: 64×64 bilinear finite elements (Grid 1).

It is worth mentioning that the two profiles are completely free of non-physical oscillations, and moreover, the minimum/maximum solution values are bounded by zero and unity. However, a large amount of artificial diffusion is introduced by performing discrete upwinding so that the resolution of the internal layer is unacceptable due to excessive smearing. In fact, it was impossible to compute any solution by the high-order Galerkin finite element method without stabilization. We also tried solving the pseudo-transient convection-diffusion equation by the linear high-order approximation but the simulation immediately failed due to strong undershoots and overshoots.

To quantify the amount of artificial diffusion engendered by the numerical scheme and to assess its ability to abstain from generating spurious oscillations, John and Knobloch introduced the following quantities used to compare various stabilized finite element methods [117]:

$$osc_{\text{exp}} = \sqrt{\sum_{(x,y)\in\Omega_2} [\max\{0, \mathrm{u}_h(x,y)-1\}]^2} \tag{3.5.3}$$

$$osc_{\text{int}} = \sqrt{\sum_{(x,y)\in\Omega_1} [\min\{0, \mathrm{u}_h(x,y)\}]^2 + [\max\{0, \mathrm{u}_h(x,y)-1\}]^2} \tag{3.5.4}$$

$$smear_{\text{exp}} = \sqrt{\sum_{(x,y)\in\Omega_2} [\min\{0, \mathrm{u}_h(x,y)-1\}]^2} \tag{3.5.5}$$

$$smear_{\text{int}} = x_2 - x_1 \tag{3.5.6}$$

Here, x_1 and x_2 are the x-coordinates of the first points along the cutline $y = 0.25$ such that $\mathrm{u}_h(x_1, 0.25) \geq 0.1$ and $\mathrm{u}_h(x_2, 0.25) \geq 0.9$, respectively. Thus, their difference measures the thickness of the interior layer. To improve accuracy, x_1 and x_2 are evaluated on a grid with mesh width 10^{-5}. The summations in the above quantities are performed over the nodes (x,y) of the computational grids, whereby the sub-domains are defined as follows [117]:

$$\Omega_1 = \{(x,y) \in \Omega : x \leq 0.5, y \geq 0.1\} \tag{3.5.7}$$

$$\Omega_2 = \{(x,y) \in \Omega : x \geq 0.7\} \tag{3.5.8}$$

In a recent publication [118] on stabilized finite element methods designed to diminish spurious oscillations at layers, John and Knobloch redefine the quantities (3.5.3) and (3.5.4) as follows:

$$osc^*_{\text{exp}} = \max_{x\geq 0.7}\{\max\{0, \mathrm{u}_h(x,y)-1\}\} \tag{3.5.9}$$

$$osc^*_{\text{int}} = \max\left\{ \max_{(x,y)\in G}\{\mathrm{u}_h(x,y)-1\}, \left| \min_{(x,y)\in G}\{\mathrm{u}_h(x,y)\} \right| \right\} \tag{3.5.10}$$

where (x,y) are the nodes in the rectangle $G := [0, 0.5] \times [0.25, 1]$ surrounding the internal layer. The above quantities (3.5.9) and (3.5.10) are independent of the computational grid, and hence, they can be used for arbitrary meshes which may be locally adapted to the layers.

In addition to the quadrilateral *Grid 1*, we performed all computations on two different triangular meshes which result from the 64×64 grid by subdividing each cell into two triangles along the diagonals which yields exactly the computational grids utilized by John and Knobloch [117]. In particular, *Grid 2* is opposite to the internal layer and *Grid 3* is aligned to the internal layer. The three meshes feature the same number of degrees of freedom, namely 4,225, whereas the number of elements equals 4,096 for the quadrilateral Grid 1 and is twice as large for both triangular grids. The values of the benchmark quantities (3.5.3)–(3.5.6) are presented in Table 3.2.

Grid	osc_{int}	osc_{exp}	osc^*_{int}	osc^*_{exp}	$smear_{\text{int}}$	$smear_{\text{exp}}$
Discrete upwinding, BiCGSTAB, tol_{abs} =1.0e–12						
Grid 1 (Q_1)	0.0	0.0	0.0	0.0	1.929e–1	8.525e–1
Grid 2 (P_1)	0.0	0.0	0.0	0.0	2.457e–1	1.549e+0
Grid 3 (P_1)	0.0	0.0	0.0	0.0	1.176e–1	1.065e–5
Defect correction scheme, tol_{abs} =1.0e–6						
Grid 1 (Q_1)	0.0	0.0	0.0	0.0	5.730e–2	5.547e–1
Grid 2 (P_1)	0.0	0.0	0.0	0.0	6.530e–2	9.839e–1
Grid 3 (P_1)	0.0	0.0	0.0	0.0	3.930e–2	7.071e–6
Defect correction scheme, tol_{abs} =1.0e–12						
Grid 1 (Q_1)	0.0	0.0	0.0	0.0	5.730e–2	5.547e–1
Grid 2 (P_1)	0.0	0.0	0.0	0.0	6.530e–2	9.839e–1
Grid 3 (P_1)	0.0	0.0	0.0	0.0	3.930e–2	7.071e–6
Newton's method, tol_{abs} =1.0e–6						
Grid 1 (Q_1)	1.679e–6	0.0	1.570e–5	1.570e–5	5.730e–2	5.547e–1
Grid 2 (P_1)	6.755e–6	0.0	3.400e–6	3.400e–6	6.530e–2	9.839e–1
Grid 3 (P_1)	7.077e–6	0.0	3.930e–2	2.964e–6	6.00e–7	7.071e–6
Newton's method, tol_{abs} =1.0e–12						
Grid 1 (Q_1)	0.0	0.0	0.0	0.0	5.730e–2	5.547e–1
Grid 2 (P_1)	0.0	0.0	0.0	0.0	6.530e–2	9.839e–1
Grid 3 (P_1)	0.0	0.0	0.0	0.0	3.930e–2	7.071e–6

Tab. 3.2. Stationary convection-diffusion: Benchmark quantities.

As expected, the use of discrete upwinding leads to quite diffusive results, whereby the creation of non-physical undershoots and overshoots is precluded by construction. In contrast, the high-resolution FEM-TVD scheme yields more accurate solution profiles but convergence is a prerequisite for positivity-preservation. We therefore considered two different values for the absolute tolerance of the outer iteration, i.e. $\|\nabla\cdot\mathbf{f}(u)\| \approx \|K^*(\mathrm{u}^{(m)})\mathrm{u}^{(m)}\| < tol_{\text{abs}}$. For the standard defect correction scheme (3.3.30) preconditioned by the constant low-order operator $P=-L$, it suffices to adopt the moderate tolerance tol_{abs} =1.0e–6 to ensure positivity. Let us mention that less restrictive convergence criteria led to the creation of small wiggles. On the other hand, the parameter value tol_{abs} =1.0e–12 is required for Newton's method which tends to produce oscillatory results in the absence of convergence (cf. the quantity osc_{int}, osc^*_{int} and osc^*_{exp} in Table 3.2).

The amount of artificial diffusion introduced by the numerical schemes strongly depends on the computational grid. The alignment of triangles (Grid 3) along the predominant flow direction

Solver configuration	CPU	NN	NL	NL/NN
Defect correction, η =1.0e–1	**11 sec**	385	619	1.60
Defect correction, η =1.0e–2	24 sec	391	2,069	5.29
Defect correction, η =1.0e–3	47 sec	391	4,697	12.01
Newton, η =1.0e–1, $\sigma = \sqrt{\varepsilon}$	36 sec	64	1,647	25.73
Newton, η =1.0e–2, $\sigma = \sqrt{\varepsilon}$	62 sec	74	3,265	44.12
Newton, η from (3.3.86), $\sigma = \sqrt{\varepsilon}$	**22 sec**	64	727	11.36
Newton, η from (3.3.87), $\sigma = \sqrt{\varepsilon}$	49 sec	216	555	2.57
Newton, η =1.0e–1, $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	48 sec	107	1,936	26.16
Newton, η =1.0e–2, $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	86 sec	104	4,515	43.41
Newton, η from (3.3.86), $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	37 sec	121	943	7.79
Newton, η from (3.3.87), $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	83 sec	344	1,371	3.99

Tab. 3.3. Stationary convection-diffusion: 128×128 Q_1 elements (Grid 1).

permits to reduce smearing of the exponential boundary layer considerably. Conversely, false alignment of elements may lead to a significant increase of numerical diffusion. In general, the *a priori* alignment of elements is impossible for complex flows which may not even possess a global flow direction. The use of bilinear finite elements employed on quadrilateral cell provides a good compromise between aligned and unaligned triangular grids. The reader who is interested in a comprehensive comparison of stabilized finite element methods is referred to [117].

Owing to the fact that the governing equation is linear, the low-order method yields a linear algebraic system that can be easily computed, e.g., by means of the BiCGSTAB or GMRES algorithm [228]. Therefore, the focus is placed on solving the nonlinear system (3.3.29) which results from the application of algebraic flux correction [135]. Note that there are no physical nonlinearities for this benchmark problem which allows us to analyze the usability of Newton's method applied to numerical nonlinearities per se. In particular, we investigate the performance of the edge-by-edge assembly of the approximate Jacobian for the antidiffusive fluxes presented in Section 3.3.5. The convergence behavior of different nonlinear solution strategies is presented in Tables 3.3–3.5, whereby the concrete solver configuration is described in the first row. The required computing time (including pre- and postprocessing), the total number of nonlinear (NN) and linear (NL) iterations and the number of linear iterations per nonlinear step (NL/NN) are given in rows 2–5. The numerical solutions were computed on three different meshes which result from Grids 1–3 by subdividing each element regularly into four cells. In other words, 16,384 bilinear finite elements (Q_1) are compared to 32,768 linear finite elements (P_1). The degrees of freedom are located at the vertices, and hence, their number equals 16,641 in all three cases.

The outer iteration was required to satisfy the tolerance tol_{abs} =1.0e–12 to prevent the creation of spurious wiggles due to the lack of convergence. For the standard defect correction procedure (3.3.30) preconditioned by the constant low-order operator $P = -L$, the number of nonlinear iterations is considerably large but the overall computing time is unrivaled. Here, the parameter η is used to control the accuracy for the linear subproblems (3.3.31), that is, the BiCGSTAB solver is required to gain one, two and three digits per outer iteration, respectively. In particular, an

Solver configuration	CPU	NN	NL	NL/NN
Defect correction, η =1.0e–1	**10 sec**	384	843	2.20
Defect correction, η =1.0e–2	24 sec	383	2,624	6.85
Defect correction, η =1.0e–3	53 sec	383	6,517	17.02
Newton, η =1.0e–1, $\sigma = \sqrt{\varepsilon}$	50 sec	85	3,268	38.45
Newton, η =1.0e–2, $\sigma = \sqrt{\varepsilon}$	88 sec	90	6,113	67.92
Newton, η from (3.3.86), $\sigma = \sqrt{\varepsilon}$	**21 sec**	78	824	10.56
Newton, η from (3.3.87), $\sigma = \sqrt{\varepsilon}$	45 sec	246	970	3.94
Newton, η =1.0e–1, $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	70 sec	153	4,189	27.38
Newton, η =1.0e–2, $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	116 sec	161	7,672	47.65
Newton, η from (3.3.86), $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	43 sec	175	1,556	8.89
Newton, η from (3.3.87), $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	77 sec	350	2,339	6.68

Tab. 3.4. Stationary convection-diffusion: $32,768$ P_1 elements (Grid 2).

incomplete LU-factorization of the low-order evolution operator L is used to precondition the linear problems. Obviously, it suffices to solve the linear problems for the solution increment very roughly to achieve good overall convergence of the outer defect correction procedure.

The total number of nonlinear iterations can be reduced significantly ($385/64 \approx 6$) by fine-tuning the discrete Newton algorithm; cf. Table 3.3. However, an unfeasible value for the perturbation parameter σ and/or the 'wrong' strategy for choosing the forcing term η can lead to a devastating increase of nonlinear iterations and computing time. Admittedly, the latter one is twice as large in the best case (indicated by boldface) so that Newton's method cannot compete with the standard defect correction scheme. In fact, the permanent reassembly of the Jacobian matrix and the inner solver for the auxiliary subproblems turn out to be the critical parts which consume most CPU time. For the presented results, the standard BiCGSTAB algorithm [228] was employed, whereby the ILU-decomposition of the Newton operator (3.3.35) served as linear preconditioner. Note that the approximate Jacobian matrix features an extended sparsity pattern so that each linear algebra operation is more costly due to the larger number of non-zero matrix entries. Furthermore, the Jacobian matrix is not well-conditioned so that the task of the linear solver is quite demanding, and thus, it may require more inner iterations be performed per outer solution update. The nonlinear convergence rates for the *unaligned* triangular Grid 2 presented in Table 3.4 show essentially the same trends as in the case of bilinear finite elements. Of course, the number of elements is twice as large but the number of degrees of freedom equals that of the quadrilateral Grid 1. In our implementation, the residual vector and the Jacobian matrix are assembled edge-by-edge. Hence, bilinear finite elements should be slightly more costly since each element gives rise to two internal edges. If a quadrilateral is converted into two triangles, one internal edge becomes a physical one, whereas the second internal edge vanishes completely. As a result, the number of total edges is reduced by the number 128×128 if all quadrilaterals are converted into triangles. Remarkably, aligning the triangles along the flow direction not only reduces the amount of numerical diffusion but it also improves the convergence rates of all nonlinear solution strategies; cf. Table 3.5. In particular, the fixed-point defect correction schemes requires considerably

Solver configuration	CPU	NN	NL	NL/NN
Defect correction, η =1.0e–1	**7 sec**	254	442	1.74
Defect correction, η =1.0e–2	15 sec	240	1,732	7.22
Defect correction, η =1.0e–3	31 sec	240	3,235	13.48
Newton, η =1.0e–1, $\sigma = \sqrt{\varepsilon}$	26 sec	61	1,379	22.60
Newton, η =1.0e–2, $\sigma = \sqrt{\varepsilon}$	38 sec	66	2,334	35.36
Newton, η from (3.3.86), $\sigma = \sqrt{\varepsilon}$	**13 sec**	58	412	7.10
Newton, η from (3.3.87), $\sigma = \sqrt{\varepsilon}$	26 sec	172	281	1.63
Newton, η =1.0e–1, $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	25 sec	78	1,175	15.06
Newton, η =1.0e–2, $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	37 sec	82	2,081	25.38
Newton, η from (3.3.86), $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	28 sec	114	942	8.26
Newton, η from (3.3.87), $\sigma = \sqrt[3]{[(1+\|\|u\|\|)\varepsilon]^2}$	44 sec	222	896	4.04

Tab. 3.5. Stationary convection-diffusion: 32,768 P_1 elements (Grid 3).

less iterations ($385/254 \approx 1.5$), whereas the gains are moderate for Newton's method.

Let us draw an intermediate conclusion from the results presented for the stationary convection-diffusion equation. Algebraic flux correction can be directly applied to the steady-state formulation so that pseudo time-stepping is not necessary. The numerical nonlinearities engendered by the TVD limiter are of good nature so that solving the nonlinear algebraic system by the fixed-point defect correction scheme (3.3.30) yields the most efficient solution algorithm. Let us emphasize the fact that the low-order evolution operator L is constant, and thus, all the preconditioner $P = -L$ can be assembled and stored once and for all at the beginning of the simulation. On the other hand, the Jacobian matrix (3.3.35) and its incomplete LU-decomposition need to be evaluated repeatedly. As a consequence, the reduction of nonlinear steps is foiled by overhead costs of each Newton step. Strictly speaking, the simplicity of the problem at hand does not justify the use of complex solution strategies. As we are about to see in the course if this section, Newton's method will demonstrate its full potential if the governing equation is nonlinear from the outset and/or the discrete transport operator L is not constant throughout the simulation.

3.5.2. Burgers' equation in space-time

The second test problem deals with nonlinear conservation laws for a scalar quantity u, whereby the time derivative is considered as an additional space variable. This approach corresponds to computing the solution for all time levels simultaneously instead of doing it step-by-step [139]. This enables us to apply flux correction both in time and space which may lead to an improved temporal resolution. On the one hand, the algebraic systems can be very large especially in 3D so that space-time formulations may be inappropriate for the simulation of realistic applications.
Test problem 2: Consider the one-dimensional inviscid Burgers equation in space and time

$$\nabla_{\mathrm{xt}} \cdot \mathbf{f}(u) = 0 \qquad \text{in} \quad \Omega = (0,1) \times (0,0.5) \tag{3.5.11}$$

where $\mathbf{f}(u) = (\frac{1}{2}u^2, u)$ is the flux function and $\nabla_{\mathrm{xt}} := (\frac{\partial}{\partial x}, \frac{\partial}{\partial t})$ denotes the gradient in space and

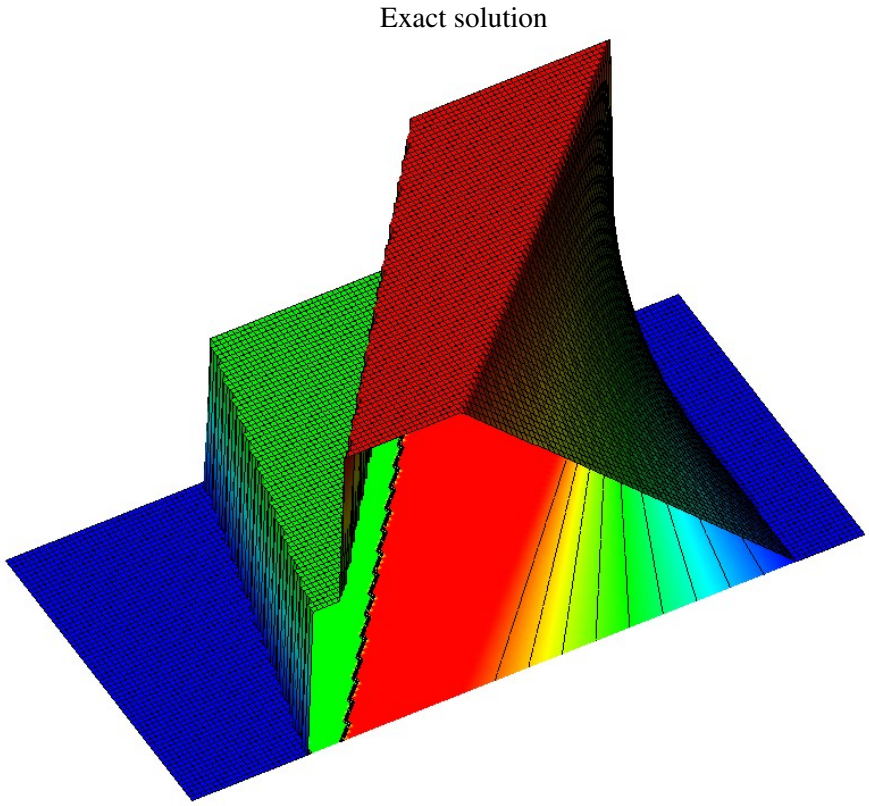


Fig. 3.10. Burger's equation in space-time: 128×64 bilinear finite elements (NLEV=7).

time. Let the following boundary conditions be imposed at the 'inlet' of the space-time domain

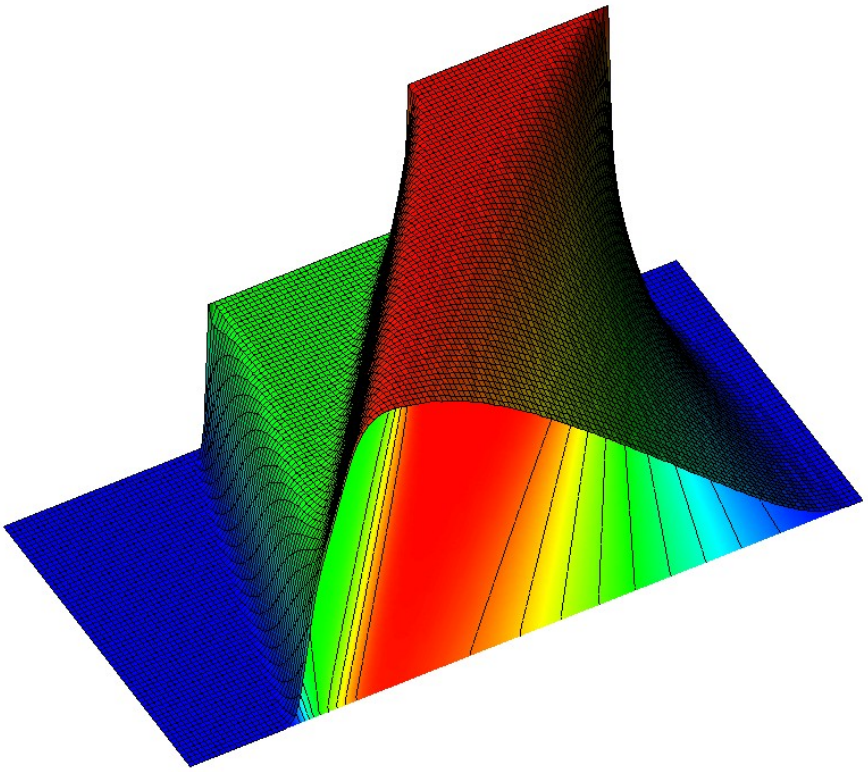
$$u(x,t) = \begin{cases} 1 & \text{if} \quad t = 0 \quad \text{and} \quad 0.1 \leq x < 0.4 \\ \frac{1}{2} & \text{if} \quad t = 0 \quad \text{and} \quad 0.4 \leq x \leq 0.7 \\ 0 & \text{if} \quad t = 0 \quad \text{and} \quad x < 0.1 \quad \text{or} \quad x > 0.7 \end{cases} \tag{3.5.12}$$

This corresponds to prescribing 'descending stairs' as initial profile and solving the transient Burgers equation up to the final time $T = 0.5$, whereby homogeneous Dirichlet conditions are imposed at the inflow, i.e. $u(0,t) = 0$. The exact solution depicted in Figure 3.10 was computed by solving the individual Riemann problems as explained in Section 2.2.2. In particular, two shock waves emanating from $(x_1,t_0) = (0.4,0.0)$ and $(x_2,t_0) = (0.7,0)$ are traveling at speeds $s_1 = 0.75$ and $s_2 = 0.25$, respectively. The Heaviside function centered about $x = 0.1$ yields the rarefaction wave

$$u(x,t) = \begin{cases} 0 & \text{if} \quad x < 0.1 \\ \frac{x-0.1}{t} & \text{if} \quad 0.1 \leq x \leq t + 0.1 \\ 1 & \text{if} \quad x > t + 0.1 \end{cases} \tag{3.5.13}$$

The numerical profiles depicted in Figure 3.11 were computed on a regular mesh consisting of 128×64 bilinear elements. The two shock waves are sharply resolved by the high-resolution FEM-TVD scheme and the solution varies smoothly across the rarefaction wave. On the other hand, discrete upwinding leads to excessive smearing so that the shocks are smoothed out and the linear transition in the rarefaction wave degenerates to a curve. In order to illustrate the strength of smoothing, the contour plots of both solutions have been included in the corresponding profiles.

(a) Low-order scheme: discrete upwinding



(b) High-resolution FEM-TVD scheme

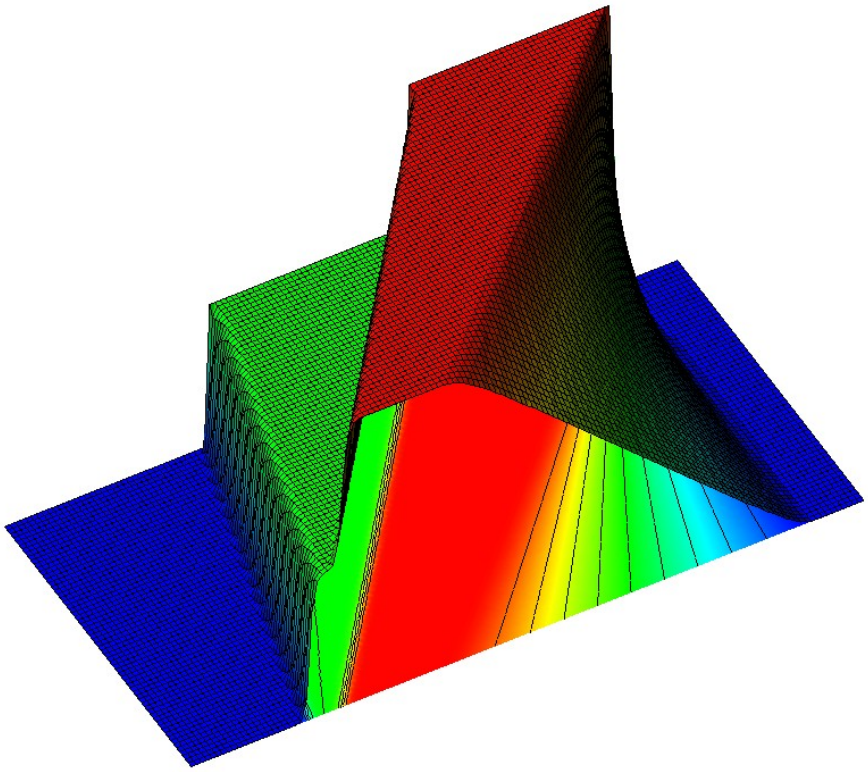


Fig. 3.11. Burger's equation in space-time: 128×64 bilinear finite elements (NLEV=7).

The difference between the exact solution u and its approximation u_h was measured in the L_1-norm

$$\|u-\mathrm{u}_h\|_1=\int_\Omega |u-\mathrm{u}_h|\mathrm{d}\mathbf{x}\approx\sum_{i=1}^{M} m_i|u(\mathbf{x}_i)-\mathrm{u}_i| \tag{3.5.14}$$

as well as in the L_2-norm defined by the following formula

$$\|u-\mathrm{u}_h\|_2^2=\int_\Omega |u-\mathrm{u}_h|^2\mathrm{d}\mathbf{x}\approx\sum_{i=1}^{M} m_i|u(\mathbf{x}_i)-\mathrm{u}_i|^2 \tag{3.5.15}$$

Their values evaluated for the low-order solution are presented in the last two columns of Table 3.6. The shock waves are considerably smeared by artificial diffusion which manifests itself in the magnitude of solution errors which decrease as the mesh is refined.

Note that the the first component of the flux function $\mathbf{f}(u)$ gives rise to a physical nonlinearity so that the low-order approximation (3.2.26) and its flux corrected counterpart (3.3.10) call for the use of a nonlinear solution strategy. This enables us to study the quality of the approximate Jacobian for the transport operator per se and assess the applicability of the edge-based assembly procedure developed in Section 3.3.4. Discrete upwinding was applied directly to the stationary problem (3.5.11) without resorting to pseudo time-stepping. The outer iteration loop was terminated once the nonlinear residual satisfied $\|\nabla\cdot\mathbf{f}(u)\|\approx\|K^*(\mathrm{u}^{(m)})\mathrm{u}^{(m)}\|<tol_{\mathrm{abs}}$, where the absolute tolerance was defined as tol_{abs} =1.0e–12. The solutions of the linear subproblems were computed by the BiCGSTAB algorithm [228] which was operated without preconditioning. It was found that the number of inner iterations could be reduced by using an incomplete LU-factorization of the preconditioner which, however, led to a significant increase of the overall CPU time.

The convergence results obtained on three levels of successively refined computational grids are presented in Table 3.6. The finer meshes consist of 32,768 and 131,072 bilinear finite elements and result from the 128×64 coarse grid by regularly subdividing each cell once and twice, respectively. In order to provide a good initial guess for the nonlinear solver, the solution from grid level $NLEV$ was prolongated to the next finer level $NLEV+1$ which amounts to performing nested iterations [88]. This technique is also known as full multigrid algorithm (FMG) [42] and it is frequently employed to accelerate steady state convergence. For the 128×64 coarse grid the solution was initialized by the boundary profile (3.5.12) extended in the direction of the t-axis. Albeit this yields a crude initial guess, no convergence problems have been observed.

The quantities presented in Table 3.6 do not include CPU times and iteration numbers required to compute the initial guess on coarser grids. The standard defect correction scheme requires more and more nonlinear iterations as the mesh is refined which results in a considerable increase of computing time. This phenomenon could not be reduced by solving the linear subproblems with higher accuracy, i.e. η =1.0e-3. In contrast, the number of Newton steps does not increase as the mesh width becomes smaller, and moreover, the nonlinear convergences rates can be improved by adopting a more restrictive forcing term η. However, there is always a trade-off between outer and inner iterations which can lead to significant differences in computing time. For this benchmark configuration, choosing the forcing term adaptively yields the best overall performance, whereby Newton's method outperforms the defect correction approach by factor 4 ($\approx$138 sec / 34 sec).

To conclude what we have said so far, the Jacobian matrix for the discrete transport operator of low order can be approximated by divided differences. The resulting Newton algorithm is quite competitive both in terms of nonlinear convergence rates and the overall computing time. Moreover, no numerical difficulties are observed for realistically fine computational meshes. Let us point out that the low-order operator L may become singular due to the presence of zero rows which are engendered by the discrete upwinding technique. If this is the case, the defect correction scheme (3.3.30) with $P=-L$ fails completely, whereas round-off errors in the approximation of the Jacobian (3.3.43) may lead to an ill-conditioned but non-singular preconditioner $P=-J$.

NLEV	grid	CPU	NN	NL	NL/NN	$\|u-u_h\|_1$	$\|u-u_h\|_2$
			Defect correction scheme, η =1.0e–1				
7	128×64	3 sec	31	823	26.58	1.4555e–2	3.5057e–2
8	256×128	18 sec	42	1,510	35.95	8.5403e–3	2.5198e–2
9	512×256	138 sec	62	2,384	38.45	4.7566e–3	1.7452e–2
			Newton's method, η =1.0e–1, $\sigma=\sqrt{\varepsilon}$				
7	128×64	1 sec	14	193	13.79	1.4555e–2	3.5057e–2
8	256×128	7 sec	13	434	33.38	8.5403e–3	2.5198e–2
9	512×256	47 sec	14	701	50.07	4.7566e–3	1.7452e–2
			Newton's method, η =1.0e–4, $\sigma=\sqrt{\varepsilon}$				
7	128×64	1 sec	8	321	40.13	1.4555e–2	3.5057e–2
8	256×128	7 sec	6	572	95.33	8.5403e–3	2.5198e–2
9	512×256	42 sec	7	675	96.43	4.7566e–3	1.7452e–2
			Newton's method, η from (3.3.86), $\sigma=\sqrt{\varepsilon}$				
7	128×64	**1 sec**	13	176	13.54	1.4555e–2	3.5057e–2
8	256×128	**5 sec**	10	303	30.30	8.5403e–3	2.5198e–2
9	512×256	**34 sec**	11	510	46.36	4.7566e–3	1.7452e–2

Tab. 3.6. Burger's equation in space-time: discrete upwinding.

The flux corrected solution could not be obtained by solving the stationary problem directly since convergence failed unless impractically small under-relaxation parameters were adopted. As a remedy, we applied algebraic flux correction to the transient counterpart of equation (3.5.11)

$$\frac{\partial u}{\partial \tau}+\nabla_{xt}\cdot\mathbf{f}(u)=0 \qquad \text{in} \quad \Omega\times\mathbb{R}_0^+ \tag{3.5.16}$$

where τ denotes the pseudo time. The fully implicit backward Euler method ($\theta=1$) was employed for the discretization in 'time' and the resulting nonlinear algebraic system

$$[M_L-\Delta\tau^{n+1}K^*(\mathrm{u}^{n+1})]\mathrm{u}^{n+1}=M_L\mathrm{u}^n \tag{3.5.17}$$

was marched into a steady state limit as explained in Section 3.3.7. The above problem was not 'solved exactly' but only a few outer iteration were performed until the nonlinear solver gained one digit or the maximum number of 10 defect correction/Newton steps was reached. Note that expression (3.5.17) can be interpreted as a relaxation scheme for the stationary problem [76], whereby each component of the quantity $\Delta\tau^{n+1}M_L^{-1}$ represents an individual relaxation parameter which is applied to the corresponding nodal equation (see Section 3.3.7). Pseudo time-stepping was terminated once the stationary part of the residual achieved the desired tolerance tol_{steady} =1.0e–12.

It is worth mentioning that the initial residual (3.3.76) of the outer iteration for time step τ^{n+1} can be used to control steady-state convergence based on the solution from the previous step:

$$\mathrm{u}^{(0)} := \mathrm{u}^n \quad \Rightarrow \quad \|\nabla_{\mathrm{xt}} \cdot \mathbf{f}(u)\| \approx \frac{1}{\Delta\tau^{n+1}} \|r^{(0)}\| = \|K^*(\mathrm{u}^n)\mathrm{u}^n\| \overset{!}{<} tol_{\mathrm{steady}} \tag{3.5.18}$$

Hence, no additional evaluations of the defect vector are required which would lead extra cost.

The computations presented in Table 3.7 were performed on the 128×64 mesh adopting different values for the time step size $\Delta\tau$. For the fixed-point defect correction scheme (3.3.75) with $P = M_L - \Delta\tau L$, the linear subproblems were solved by the BiCGSTAB algorithm [228] which was required to gain one digits per nonlinear step, i.e. η =1.0e–1. The total number of pseudo time steps (NT) is about 250 unless impractically small time steps were employed. However, the value NN/NT=9.9 indicate that the outer iteration did not converge within the admissible number of 10 steps, that is, the standard defect correction algorithm was unable to gain one digit per pseudo time step. Since evolution details are immaterial, convergence of pseudo time steps is not mandatory provided that the initial residual satisfies the steady-state criterion (3.5.18) in the long run.

As an alternative, Newton's method was applied to the nonlinear algebraic equation (3.5.17), whereby the forcing term was chosen adaptively following strategy (3.3.86). The BiCGSTAB algorithm was not able to solve the subproblems for the solution increments (3.3.81) so that the GMRES(k) method [228] had to be used, where $k = 10$ denotes the dimension of the Krylov subspace. In fact, 15-16 time steps suffice to compute the steady state solution, and moreover, the Newton iteration converges within the admissible number of 10 steps. The relaxation parameter $\Delta\tau$ plays an important role for the overall performance, and thus, it has to be determined carefully. Choosing the time step too large may lead to ill-conditioned system matrices which makes the task of the linear solver extremely difficult. It may even fail completely if the time step is chosen too large. On the other hand, the nonlinear convergence rates deteriorate for a very small parameter

$\Delta\tau$	CPU	NT	NN	NL	NN/NT	NL/NN
Defect correction scheme, BiCGSTAB, η =1.0e–1						
100.0	50 sec	249	2,488	6,521	9.99	2.62
10.0	50 sec	248	2,478	6,574	9.99	2.65
1.0	**47 sec**	253	2,526	6,561	9.98	2.60
0.1	52 sec	288	2,848	5,915	9.89	2.08
$\Delta\tau_{\mathrm{mean}} = 82.2$	47 sec	244	2,436	6,780	9.98	2.78
Newton's method, GMRES(10), η from (3.3.86), $\sigma = \sqrt{\varepsilon}$						
100.0	32 sec	16	130	5,855	8.13	45.04
10.0	**29 sec**	15	118	5,039	7.87	42.70
1.0	33 sec	23	129	6,079	5.61	47.12
0.1	102 sec	129	570	10,231	4.42	17.95
$\Delta\tau_{\mathrm{mean}} = 10.2$	36 sec	19	162	6,444	8.52	39.78

Tab. 3.7. Burger's equation in space-time: FEM-TVD scheme.

$\Delta\tau$ which corresponds to performing strong under-relaxation of the stationary problem [76]. This discrepancy between the number of outer and inner iterations manifests itself in the total CPU times required to compute the steady state solution which can differ by magnitudes.

A viable strategy to choose the pseudo time step in an adaptive fashion was suggested by Mulder and van Leer [190] and successfully implemented into a defect correction scheme by Tracy et al. [252]. In the so-called switched evolution relaxation approach, a small value $\Delta\tau^0$ is used at the beginning of the simulation and the time step is updated in each iteration as follows:

$$\Delta\tau^{n+1} = \Delta\tau^n \frac{\|K^*(\mathrm{u}^{n-1})\mathrm{u}^{n-1}\|}{\|K^*(\mathrm{u}^n)\mathrm{u}^n\|} \tag{3.5.19}$$

In essence, 'time-accurate' integration of the backward Euler scheme (3.5.17) is performed in the start-up phase, whereas $\Delta\tau^n \to \infty$ once the solution u^n is sufficiently close to the steady state limit. In a practical implementation, one should also limit the growth and reduction of the time step [131]. It is also advisable to prescribe upper and lower bounds for the absolute time step size

$$l \leq \frac{\Delta\tau^{n+1}}{\Delta\tau^n} \leq L, \qquad \Delta\tau_{\min} \leq \Delta\tau^{n+1} \leq \Delta\tau_{\max} \tag{3.5.20}$$

Otherwise, the size of the artificial time step tends to oscillate as the solution becomes nearly stationary, and it is impossible to compute the converged steady state solution.

Preliminary results for an adaptively chosen parameter $\Delta\tau \in [1,100]$ are presented in Table 3.7, where $\Delta\tau_{\text{mean}}$ denotes the average time step size. In fact, the update formula (3.5.19) turned out to be helpful in adjusting the relaxation parameter in the nonlinear solver if the 'optimal' value $\Delta\tau$ was not known *a priori* which is typically the case in practical applications. However, tuning the parameters l, L, $\Delta\tau_{\min}$, and $\Delta\tau_{\max}$ which depend on the problem at hand, the mesh width and the solution strategy to name just a few is not trivial. The general drawback of the switched evolution relaxation procedure is as follows: Formula (3.5.19) modulates the time step size based on ratio of steady state residuals but it does not include the work required to compute the solution in the particular steps. Thus, if the overall costs for approximating the solution u^n very accurately, i.e. $\|K^*(\mathrm{u}^n)\mathrm{u}^n\| \ll \|K^*(\mathrm{u}^{n-1})\mathrm{u}^{n-1}\|$, are considerably large, the relaxation parameter $\Delta\tau^{n+1}$ is increased so that the computation of the new solution is likely to be even more expensive. Strictly speaking, the time step is only reduced if the current time step diverged/failed completely. In light of the above, the optimal strategy for adjusting $\Delta\tau$ in practice is an open problem.

The resolution power of the FEM-TVD scheme is best illustrated by considering the errors

$$NLEV = 7: \qquad \|u - \mathrm{u}_h\|_1 = 4.0624\mathrm{e}-3, \qquad \|u - \mathrm{u}_h\|_2 = 1.5019\mathrm{e}-2 \tag{3.5.21}$$

$$NLEV = 8: \qquad \|u - \mathrm{u}_h\|_1 = 1.7635\mathrm{e}-3, \qquad \|u - \mathrm{u}_h\|_2 = 1.0210\mathrm{e}-2 \tag{3.5.22}$$

$$NLEV = 9: \qquad \|u - \mathrm{u}_h\|_1 = 8.0907\mathrm{e}-4, \qquad \|u - \mathrm{u}_h\|_2 = 8.0820\mathrm{e}-3 \tag{3.5.23}$$

which are significantly less than those observed for discrete upwinding; cf. Table 3.6.

In summary, the flexibility in choosing parameters for Newton's method is both a blessing and a curse. Oversolving of the linear problems can be avoided effectively by updating the forcing term following expression (3.3.86) proposed by Eisenstat and Walker [71]. In all numerical studies, this adaptive strategy was more efficient than keeping η constant throughout the simulation. The low-order solution to the inviscid Burgers equation in space-time formulation could be computed directly, whereby the discrete Newton algorithm outperformed the slowly converging defect correction procedure. Application of the high-resolution FEM-TVD scheme led to nonlinear algebraic equations which called for the use of pseudo time-stepping to ensure convergence. In practice, choosing the artificial parameter $\Delta\tau$ is a delicate task since improper choices may lead to poor overall performance. In essence, the challenge is to find the 'optimal' relaxation parameter for the stationary problem which may be different in each pseudo time step.

3.5.3. Solid body rotation

Let us proceed to transient flows and consider the time-dependent continuity equation

$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v}u) = 0 \tag{3.5.24}$$

A challenging benchmark intended to assess the ability of high-resolution transport algorithms to preserve both smooth and discontinuous profiles was proposed by LeVeque [156].
Test problem 3: A slotted cylinder, a cone and a smooth hump are exposed to the non-uniform velocity field $\mathbf{v} = (0.5-y, x-0.5)^T$ and undergo a counterclockwise rotation about the center of the unit square $\Omega = (0,1)\times(0,1)$. Each of these bodies lies within a circle of radius $r_0 = 0.15$ centered at point $(x_b, y_b) \in \Omega$ for $b = 1,2,3$. The rest of the domain is initialized by zero.

After each full revolution ($t = 2\pi k$, $k \in \mathbb{N}$), the exact solution of the pure convection equation (3.5.24) coincides with the initial data depicted in Figure 3.12 (top). The shapes of the solid bodies can be expressed by means of the normalized distance function for the reference points (x_b, y_b):

$$r_b(x,y) = \frac{1}{r_0}\sqrt{(x-x_b)^2 + (y-y_b)^2} \tag{3.5.25}$$

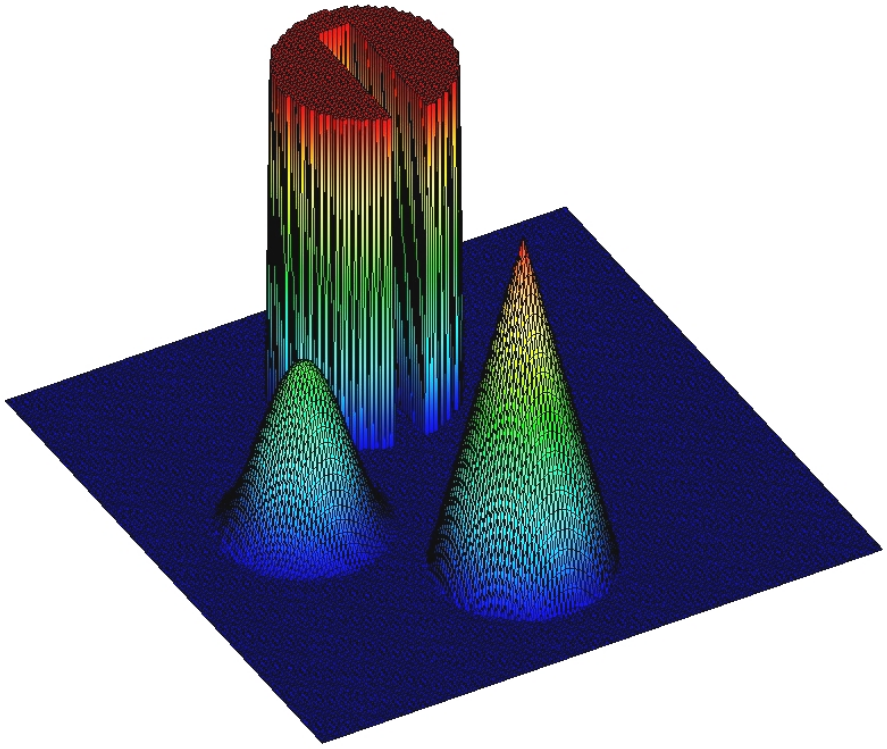
Then, the analytical expressions for the three shapes read as follows [145]:

Cylinder:	$(x_1, y_1) = (0.5, 0.75)$,	$u(x,y,0) = \begin{cases} 1 & \text{if} \quad \lvert x-x_1\rvert \geq 0.025 \vee y \geq 0.85 \\ 0 & \text{otherwise} \end{cases}$
Cone:	$(x_2, y_2) = (0.5, 0.25)$,	$u(x,y,0) = 1 - r_2(x,y)$
Hump:	$(x_3, y_3) = (0.25, 0.5)$,	$u(x,y,0) = 0.25[1 + \cos(\pi \min\{r_3(x,y), 1\})]$

Homogeneous Dirichlet boundary conditions are prescribed at the inflow boundary, i.e. $\mathbf{v}\cdot\mathbf{n} < 0$. This benchmark was repeatedly utilized to assess the potential of algebraic flux correction schemes [134, 136, 139, 145] which allows to observe the development of the AFC methodology over the years. The numerical solution depicted in Figure 3.12 (bottom) was computed by the semi-implicit FEM-FCT algorithm [137] described in Section 3.4.2, whereby a triangular grid consisting of 32,768 linear finite elements was adopted. For the discretization in time, the second-order accurate Crank-Nicolson time-stepping scheme ($\theta = 0.5$) was employed using the time step $\Delta t = 10^{-3}$. No spurious wiggles are observed in the computed solution profile and the shapes of all three bodies are well preserved. In order to quantify the error, the difference between the exact solution u and its finite element approximation u_h after one complete revolution (i.e. at time $t = 2\pi$) was measured in the L_1- and L_2-norm given by expressions (3.5.14) and (3.5.15), respectively.

In contrast to test problems 1 and 2, this benchmark deals with transient flows, and hence, a time-accurate resolution of evolution details is essential. It is therefore insufficient to perform a few outer iterations and proceed to the next time step. In contrast, the nonlinear algebraic system (3.4.2) needs to be solved for the end-of-step solution u^{n+1}, e.g., by the fixed-point iteration scheme (3.4.5). It is worth mentioning that the low-order evolution operator $A = M_L - \theta\Delta t L$ can be turned into an M-matrix by adjusting the time step size as explained in Section 3.1.3. Thus, intermediate solutions are also positivity-preserving so that non-physical oscillations cannot occur even if the fixed-point iteration (3.4.5) is terminated too early. Of course, the errors engendered by unconverged solutions accumulate as time goes on so that the final solution profile may be poorly resolved. The Newton operator (3.4.24) lacks the M-matrix property so that convergence is a prerequisite for positivity-preservation. If the nonlinear algebraic systems are not solved accurately enough, the maximum and minimum solution values can exceed their upper and lower bounds imposed by the initial profile so that small undershoots and overshoots are likely to occur [187].

(a) Initial data / exact solution at time $t = 2\pi k$



(b) High-resolution semi-implicit FEM-FCT scheme at time $t = 2\pi$

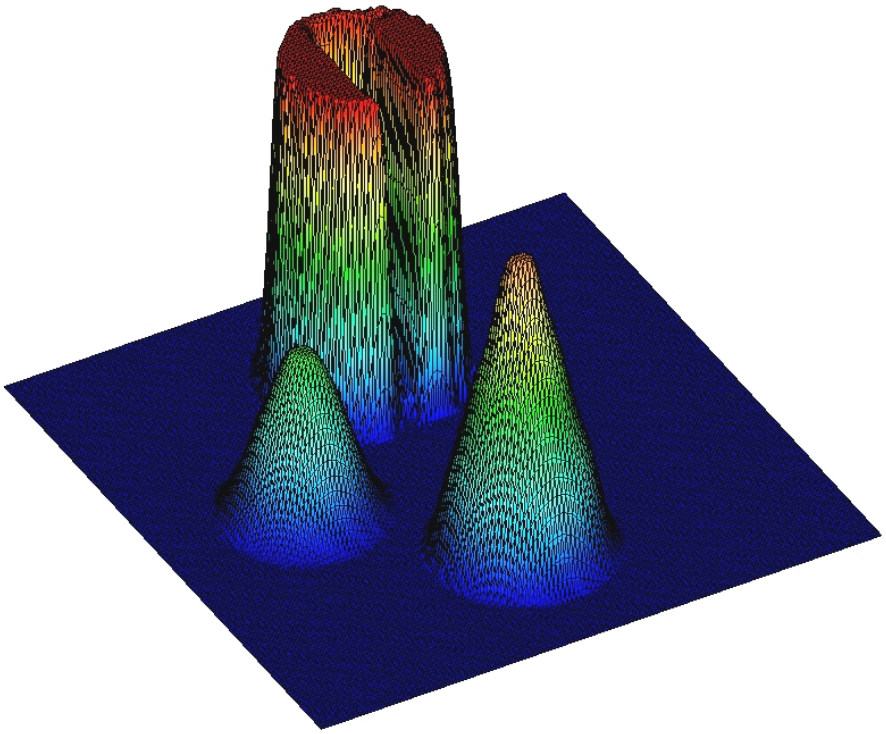


Fig. 3.12. Solid body rotation:, 32,768 linear finite elements (NLEV=7).

NLEV	CPU	NN	NN/NT	NL	NL/NN	$\|u-u_h\|_1$	$\|u-u_h\|_2$
			Defect correction scheme, η =1.0e–1				
5	48 sec	132,474	21.08	132,474	1.00	3.4905e–2	9.9643e–2
6	226 sec	127,219	20.24	127,219	1.00	2.0905e–2	7.8877e–2
7	1,230 sec	120,973	19.25	120,973	1.00	9.1127e–3	4.5320e–2
			Newton's method, η =1.0e–1, $\sigma=\sqrt{\varepsilon}$				
5	**45 sec**	27,836	4.42	47,288	1.70	3.4905e–2	9.9643e–2
6	**206 sec**	27,373	4.36	45,584	1.67	2.0904e–2	7.8876e–2
7	**1,170 sec**	26,266	4.18	44,218	1.68	9.1127e–3	4.5321e–2
			Newton's method, η from (3.3.86), $\sigma=\sqrt{\varepsilon}$				
5	46 sec	31,440	5.00	40,647	1.29	3.4905e–2	9.9643e–2
6	231 sec	31,393	5.00	39,413	1.25	2.0904e–2	7.8876e–2
7	1,260 sec	30,130	4.79	36,913	1.23	9.1127e–3	4.5321e–2

Tab. 3.8. Solid body rotation: semi-implicit FEM-FCT scheme.

In the numerical results presented in this work, we therefore accepted the approximate solution provided that the residual (3.4.6) reached the absolute tolerance $\|r^{(m)}\| < 1.0\mathrm{e}-10$. This stopping criterion was also utilized for the defect correction procedure to allow for a fair comparison. The linear problems for the solution increment were solved by the BiCGSTAB algorithm [228].

The convergence behavior of the different solution procedures is presented in Table 3.8, whereby two alternative strategies for choosing the forcing term η were studied. Application of Newton's method leads to a small gain of computing time (about $5-8\%$) as compared to the standard defect correction scheme. On the other hand, the number of outer iterations can be reduced by factor $4-5$. The discrepancy between CPU time and nonlinear convergence rates can be explained by the 'simplicity' of the linear model equation (3.5.24). The low-order evolution operator $A = M_L - \theta\Delta t L$ is constant as long as the mesh is fixed, and therefore, it can be assembled and stored once and for all at the beginning of the simulation. Conversely, the Jacobian matrix (3.4.25) for the antidiffusive fluxes needs to be evaluated in each outer iteration. It is worth mentioning that the resulting Newton operator $P = M_L - \theta\Delta t J - \bar{J}$ exhibits the same sparsity pattern as the stiffness matrix, and therefore, the costs of linear algebra operations are comparable for A and P. Of course, the matrix condition numbers may be different so that the BiCGSTAB algorithm needs slightly more iterations to solve the linear subproblems if Newton's method is employed.

In light of the above, we expect the potential of Newton's method to be more pronounced for time-dependent velocity fields for which the discrete transport operator $L(t)$ needs to be updated in each time step. Hence, the preconditioner $A(t) = M_L - \theta\Delta t L(t)$ is no longer constant, and thus, it cannot be computed *a priori*. Furthermore, system matrices need to be assembled/updated if the grid is modified, e.g., if adaptive mesh refinement and/or coarsening is performed. We will reinvestigate this issue in Section 4.7 which is concerned with dynamic mesh adaptivity.

3.5.4. Swirling flow problem

Let us consider another numerical example for the time-dependent continuity equation (3.5.24). Test problem 4: This two-dimensional benchmark introduced by LeVeque [156] deals with the swirling deformation of the initial data subject to the non-uniform velocity field

$$\mathbf{v}(\mathbf{x},t) = (\sin^2(\pi x)\sin(2\pi y)g(t), -\sin^2(\pi y)\sin(2\pi x)g(t))^T \tag{3.5.26}$$

where $g(t) = \cos(\pi t/T)$ on the time interval $0 \le t \le T$. The initial velocity profile at time $t = 0$ is shown in Figure 3.13 (left). Since $\mathbf{v} = (0,0)$ on the boundary of the unit square, no boundary conditions need to be prescribed in the case of pure convection. The initial condition equals unity within a circular sector of $\pi/2$ radians and is set to zero elsewhere:

$$u(x,y,0) = \begin{cases} 1 & \text{if} \quad (x-1)^2 + (y-1)^2 < 0.8 \\ 0 & \text{otherwise} \end{cases} \tag{3.5.27}$$

The initial mass distribution is deformed by the time-dependent velocity field which gradually slows down and reverses at time $T/2$ so that the initial profile is recovered as exact solution at the final time $T = 1.5$. The numerical solutions were computed by the semi-implicit FEM-FCT algorithm [137] with linear finite elements. The solution profiles recovered at two different time instants are depicted in Figure 3.14. They are completely free of spurious oscillations, and moreover, the steep front which assumes a spiral shape as time goes on is sharply resolved.

Owing to the time dependency of the velocity field (3.5.26) it is expedient to adjust the time step size dynamically in the course of the simulation. This can be accomplished in the framework of adaptive control [89] by using the Proportional-Integral-Differential (PID) controller [256]

$$\Delta t^{n+1} = \left(\frac{e_{n-1}}{e_n}\right)^{k_P} \left(\frac{e_{\text{target}}}{e_n}\right)^{k_I} \left(\frac{e_{n-1}^2}{e_n e_{n-2}}\right)^{k_D} \Delta t^n, \qquad e_n = \frac{\|\mathrm{u}^{n+1} - \mathrm{u}^n\|}{\|\mathrm{u}^{n+1}\|} \tag{3.5.28}$$

Here, e_{target} denotes the desired target tolerance for relative changes and the quantity e_n measures the actual variation of solution values from time t^n to time t^{n+1}. The PID variables $k_P = 0.075$, $k_I = 0.175$ and $k_D = 0.01$ are defined as suggested in [256]. The objective of the above equation is to make the output quantity e_n follow the desired reference value e_{target} along a smooth curve

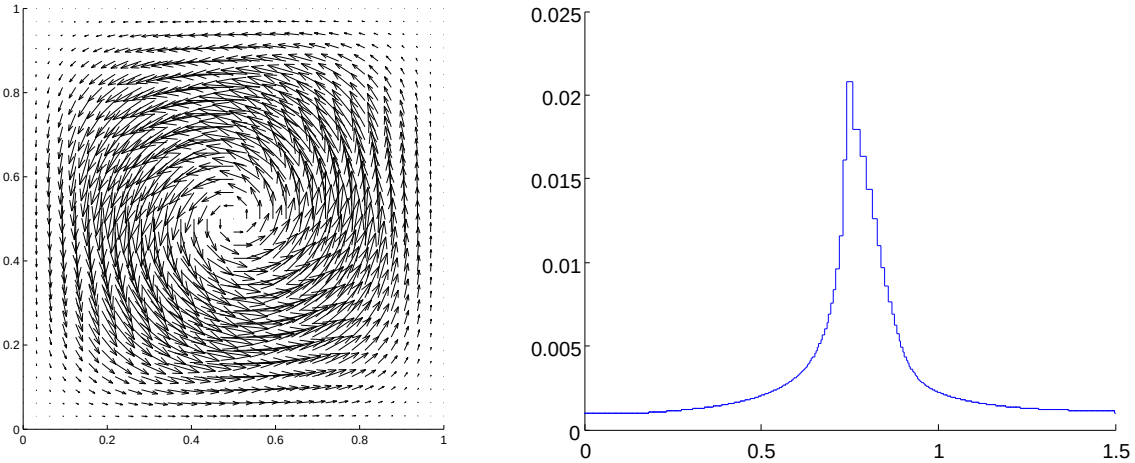
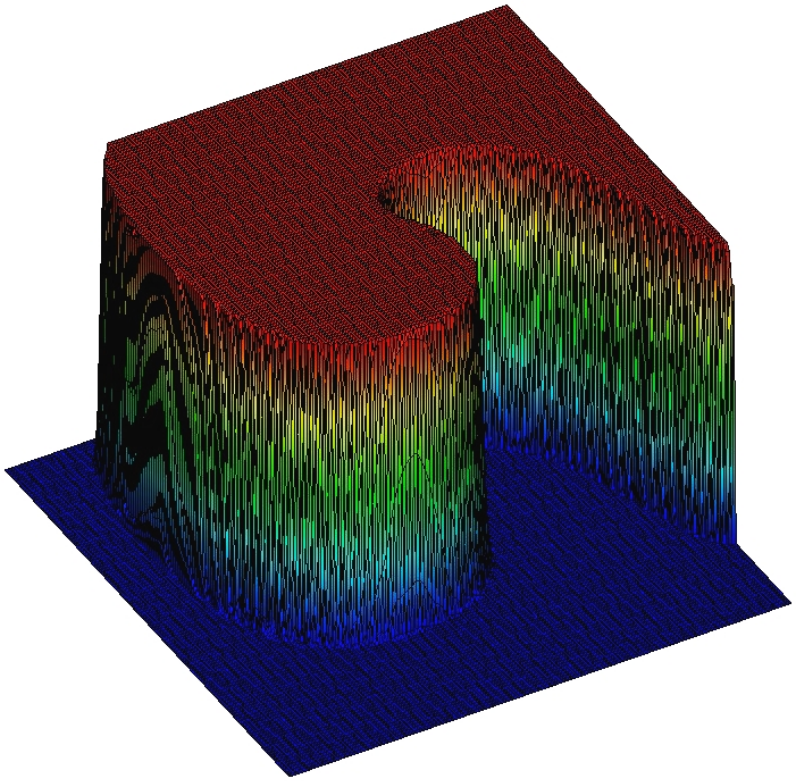


Fig. 3.13. Swirling flow: initial velocity field and evolution of the time step size.

(a) Intermediate solution at time $t = 0.75$



(b) Final solution at time $t = 1.5$

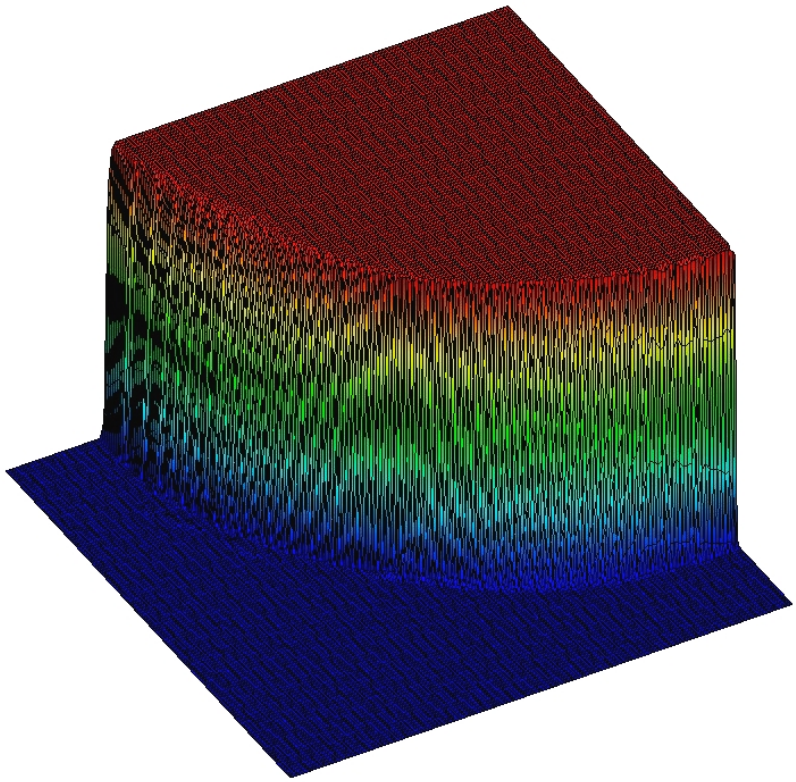


Fig. 3.14. Swirling flow: 32,768 linear finite elements, semi-implicit FEM-FCT scheme.

(vs. time), while minimizing abrupt changes in Δt. If a time step yields an unacceptable value $e_n > e_{\max}$, then it is rejected and the solution u^{n+1} is recomputed adopting the scaled time step size $\Delta t^n := \frac{e_{\max}}{e_n}\Delta t^n$. In a practical implementation, one should also limit the growth and reduction of the time step [131] and prescribe absolute bounds to countervail the wind-up effect [256]

$$\Delta t_{\min} \le \Delta t^{n+1} \le \Delta t_{\max}, \qquad l \le \frac{\Delta t^{n+1}}{\Delta t^n} \le L \tag{3.5.29}$$

In our computations we adopted the value e_{target} =5.0e–3 which yields Δt ≈1.0e–3 as the initial/minimal time step size and set $l = 1/L = 0.5$. The evolution of the step size for the semi-implicit FEM-FCT scheme is illustrated in Figure 3.13 (right). The creeping flow velocity near time $T/2$ correlates with the increase of the time step and the initial value Δt ≈1.0e–3 is recovered at the end of the simulation. Note that the upper bound $\Delta t_{\max} = 0.1$ is never reached which demonstrates that the control algorithm (3.5.28) does not drive the step size ad infinitum but monitors the relative changes precisely. The same computations were performed with a constant time step Δt =1.0e–3 in order to measure the gain in computing time and to estimate the error engendered by using larger time steps during the simulation. The results obtained from the application of Newton's method and the fixed-point defect correction scheme (3.4.5) are presented in Table 3.9.

The total number of time steps (NT) reduces by factor 1.6 (≈ 1,500 sec/905 sec) by letting the PID controller (3.5.28) adjust Δt dynamically. The price to be paid is a slightly less accurate temporal resolution of the final solution. However, the deterioration of solution errors measured in the L_1- and L_2-norm is not more than $5-6\%$ which is acceptable for practical applications. The smaller number of total time steps goes along with a reduction of the overall computing time. Newton's method converged within $5-6$ iterations per time step even if Δt varies between 1.0e–3 and 2.0e–2. In essence, the discrete Newton algorithm scales linearly in the number of time steps so that the CPU time reduces by factor 1.5 by adopting the evolutionary PID controller (3.5.28).

On the other hand, the nonlinear convergence rates of the fixed-point defect correction scheme degrade if larger time steps are employed so that the gain in computing time is less pronounced (1.2≈ 1,367 sec/1,119 sec). In conclusion, the final solution can be determined about three times faster by choosing the time step adaptively and solving the nonlinear algebraic systems by Newton's method. If temporal accuracy is most important, sufficiently small time steps need to be employed so that $P = M_L$ is an excellent preconditioner for the fixed-point iteration scheme (3.4.5).

Time-stepping	CPU	NT	NN	NN/NT	NL/NN	$\|u-\mathrm{u}_h\|_1$	$\|u-\mathrm{u}_h\|_2$
Defect correction scheme, η =1.0e–1							
e_{target} =5.0e–3	1,119 sec	905	22,196	24.53	1.00	7.6305e–3	4.0892e–2
Δt =1.0e–3	1,367 sec	1,500	32,890	21.93	1.00	7.1581e–3	3.8926e–2
Newton's method, η from (3.3.86), $\sigma = \sqrt{\varepsilon}$							
e_{target} =5.0e–3	**336 sec**	905	5,281	5.84	1.18	7.6305e–3	4.0893e–2
Δt =1.0e–3	**512 sec**	1,500	7,947	5.30	1.19	7.1580e–3	3.8926e–2

Tab. 3.9. Swirling flow: semi-implicit FEM-FCT scheme.

3.5.5. Taxonomy of nonlinear solution techniques

In summary, our numerical studies did not reveal the 'ultimate' solver which performed best in all disciplines. It is therefore important to know about the pros and cons of the presented solution strategies to chose the most suitable approach for a concrete application in practice.

Symmetric flux limiting of FCT type is favorable for the simulation of transient flows, whereby the use of moderately small time steps is required to achieve good temporal accuracy. The computing time can be significantly reduced by adjusting the size of the physical time step adaptively, e.g., by means of the PID controller (3.5.28) which monitors the relative changes of the solution.

For the solution of the nonlinear algebraic equations, we considered the standard defect correction scheme and compared its performance to that of the discrete Newton method. As a rule of thumb, the complexity of Newton's method pays off if (a) physical nonlinearities are present in the governing equation, (b) sufficiently large (pseudo) time steps can be employed, (c) the grid is refined (e.g., by using adaptive mesh refinement), and (d) the system matrix changes in time and needs to be updated in each outer iteration. In other words, Newton's method seems to benefit from the numerical difficulties inherent to the problem at hand, whereas the convergence behavior of defect correction schemes deteriorates for increasingly demanding applications. On the other hand, the ease of implementation and its unrivaled performance observed for the stationary convection-diffusion equation make it the perfect solution strategy for simple problems.

In a practical implementation, both solution procedures may act in concert as follows: If Newton's method fails to determine the numerical solution, the particular (time) step can be recomputed adopting the more robust defect correction approach. Conversely, slow convergence of the latter one may serve as an indicator to include the Jacobian matrix into the global preconditioner. In conclusion, algebraic flux correction schemes were equipped with discrete Newton methods which – in most situations – led to an improvement of the overall performance.

For stationary problems, one has to decide whether the steady-state formulation is solved directly or its transient counterpart is marched into a steady state limit making use of the fully implicit backward Euler method. Theoretically, upwind-biased flux limiters of TVD type can be applied directly to the stationary problem so that there is no need for pseudo time-stepping. In practice, the resulting system matrices may become extremely ill-conditioned so that most standard approaches fail to solve the linear problems for the solution increments. Furthermore, the conservative elimination of negative off-diagonal coefficients from the discrete transport operator may lead to zero rows [131], and hence, singular matrices can be produced by performing discrete upwinding. This drawback is circumvented by resorting to pseudo time-stepping, whereby the artificial time step $\Delta\tau$ serves as relaxation parameter that can be tuned to improve robustness [76]. However, finding its optimal value is quite an art. In principle, it should be sufficiently large to achieve steady state convergence in a few steps. On the other hand, the stationary formulation is obtained for $\Delta\tau \to \infty$ so that the linear solver converges slowly and it may even fail if the artificial time step is too large. It is therefore advisable to employ a moderately small $\Delta\tau$ at the beginning and adjust its value adaptively in the course of the simulation. However, further research is required to devise a robust strategy to control the time step size in an optimal manner. A viable approach may be based on the (asymptotic) convergence rate of the nonlinear solution procedure

$$\rho^n = \left(\frac{\|\boldsymbol{r}^{(m)}\|}{\|\boldsymbol{r}^{(0)}\|} \right)^{\frac{1}{m}} \tag{3.5.30}$$

where m denotes the number of iterations performed in the current step. The above quantity accounts for the actual 'work' required to compute the end-of-step solution u^{n+1}, and $\rho^n \geq 1$ indicates that the initial residual did not reduce. The new time step size may be determined by the PID controller (3.5.28) applied to the convergence rate (3.5.30) subject to the target value ρ_{target}.

4

Mesh adaptivity

4.1. Introduction to mesh adaptivity

Progress in computer performance and the improvement of numerical methods for CFD have enabled analysts to simulate more and more challenging problems for which no or at least little *a priori* knowledge of the solution structure is available. This complexity has made it increasingly difficult and time consuming to construct usable grids – not to speak of optimality in any sense – by trial and error. The situation gets even worse for the treatment of transient flows, where local phenomena such as discontinuities and steep gradients may appear, travel in space and disappear again as time evolves. More and more effort is required to verify the reliability of numerical schemes and to validate the simulation results produced by existing CFD codes [192, 221].

The main ingredients for the design of an adaptive finite element scheme are as follows:

- A tool that *estimates* the error between the exact solution and its finite element approximation or at least a mechanism which *indicates* where the difference is unacceptably large;
- An algorithm that *modifies the grid* so as to reduce the numerical error (h-adaptation);
- A sensor which *adjusts the order* of approximation for each element (p-adaptation).

Convection dominated problems give rise to the formation of steep gradients and shock waves which may propagate and interact with each other. Standard discretization techniques are prone to generate non-physical overshoots and undershoots in the vicinity of shocks and discontinuities so that a non-oscillatory low-order approximation should be used locally. In the framework of algebraic flux correction (AFC) [133, 134, 137, 139, 144, 145], flux limiting is employed to blend discretizations of high and low order so as to prevent the formation of wiggles. Hence, such high-resolution schemes can be interpreted as a rudimentary p-adaptation approach, whereby the largest order of approximation is given by the original Galerkin discretization. It is used provided the local solution is sufficiently smooth whereas the discrete upwind method is recovered in the vicinity of discontinuities. Adaptive grid refinement improves the resolution of shocks significantly without leading to a prohibitively expensive increase in computational costs caused by global refinement. Note that for hyperbolic problems information travels at finite speed along characteristic curves so that adaptive mesh refinement is most likely to improve the solution locally without corrupting its global behavior. The original mesh can be safely restored in locally refined regions once the feature of interest that caused refinement is no longer present.

The variation of the approximation order is a common approach in the framework of discontinuous Galerkin finite element schemes (DGFEM) since the discontinuity of adjacent cells is acceptable by construction [35, 52, 107, 244]. In the context of classical Galerkin schemes variation of the approximation order is not straightforward [1] and beyond the scope of this thesis. Therefore, only mesh adaptation strategies are considered which are based on grid modifications.

In particular, we will present a hierarchical mesh adaptation approach applicable to hybrid meshes, which attempt to combine the advantages of both structured and unstructured meshes [233, 261]. The use of triangles or tetrahedra is particularly useful for the triangulation of complex domains which can be accomplished for instance by advancing front algorithms [170, 203]. Unstructured grids can be also used to replace parts of structured grid of poor quality, e.g., in the vicinity of singular points, where highly stretched distorted cells are likely to occur. On the other hand, numerical methods may be implemented more efficiently on structured meshes since indirect addressing of elements can be avoided. This has led to the use of Cartesian cores [166] in the interior of the domain, whereas unstructured meshes are employed at the boundary and/or to join multiple regions of structured grids. For the solution of the Navier-Stokes equations, the use of orthogonal grids in the boundary layer is considered by some authors an essential property to obtain a good discretization of the diffusive operator. Hence, structured meshes can be generated near the boundary, whereas unstructured and probably coarser triangulations are employed elsewhere. In short, the flexibility of hybrid meshes makes them a favorable tool for a wide range of applications.

This chapter is structured as follows. A brief survey of existing error estimation strategies applicable to convection dominated problems is given in Section 4.2. The aim is to point out the basic ideas and discuss, whether the underlying principles can be applied to finite element approximations based on algebraic flux correction [139]. Moreover, error indicators commonly in use are presented since they represent an efficient alternative to rigorous error estimation techniques which are computationally more expensive. In particular, recovery-based error indicators are reviewed and a slope-limited variant of the standard gradient recovery procedure is proposed in Section 4.3. As an alternative, an efficient error indicator based on limited slope averaging is suggested.

The second part of this chapter starts with an overview of recent trends in mesh adaptive procedures (cf. Section 4.5). In particular, the work by Hempel [101, 102] is reviewed and a generalized approach to hierarchical mesh adaptation for transient flow problems is proposed in Section 4.6. It is based on the red-green refinement strategy introduced by Bank et al. [20] but special care is taken to allow for an efficient re-coarsening of previously refined mesh regions. The genealogy of the entire triangulation is managed by means of generation indices which store the date of birth of each vertex. These nodal quantities allow to control the maximum refinement level, and moreover, the type of element (green or red) and its relation to adjacent cells can be determined in constant time. The grid re-coarsening algorithm is based on the recursive locking of vertices which must not be deleted either due to accuracy reasons or to ensure that the resulting grid is free of 'hanging nodes'. Implementation details are discussed and some basic properties of the proposed algorithms are analyzed. Numerical examples which illustrate the performance of the proposed algorithm for two-dimensional benchmark problems are presented in Section 4.7.

4.2. Survey of error estimation techniques

Starting with the pioneering work of Babuška and Rheinboldt [12, 13] on *a posteriori* error analysis for finite element approximations of the Poisson and Cauchy-Riemann equations, an extensive development of theories and methods for *a posteriori* error estimation has come to life. For a review of some of the main developments, the interested reader is referred to the monographs by Ainsworth and Oden [194], Babuška and Strouboulis [14], Bangerth and Rannacher [17] and Verfürth [259] as well as the comprehensive review articles by Becker and Rannacher [26] and Eriksson et al. [73]. The recent book edited by Barth and Deconinck [23] provides a compilation of survey articles addressing error estimation techniques for fluid dynamics applications. Rigorous *a posteriori* error estimation techniques for scalar conservation laws in multidimensions can be found in the publications by Cockburn [51], Johnson [121], Hartmann, Houston [94], Kröner, Ohlberger [129, 130, 197], Mackenzie et al. [175] and Süli, Houston [243] to name just a few.

4.2.1. Types of errors

In contrast to elliptic and parabolic problems, where solution deficiencies have only local effects due to global minimization principles, the mechanism of error propagation is more complex in the hyperbolic case. Consider an initial coarse mesh which is incapable of resolving small scale features. Then the error estimator may completely miss these unresolved features and suggest grid refinement in the wrong place or no adaptation at all. A similar behavior can be observed if the employed discretization scheme engenders too much artificial dissipation so as to suppress important small scale features and smear shock waves considerably [210]. It follows that the initial grid should be sufficiently fine so that essential flow features are captured. If some *a priori* knowledge of the flow behavior is available, it is worthwhile to pre-adapt the computational grid 'by hand' and introduce more cells in regions, where one expects the solution to exhibit localized structures. Moreover, the use of a high-resolution discretization scheme is advisable so as to minimize the error due to excessive artificial diffusion and non-physical oscillations, respectively.

Houston et al. [108] decompose the error $e_\omega = e|_\omega$ on an open subset ω of the domain Ω into its cell contribution e_ω^{cell} which is governed by the local residual and the transmitted error e_ω^{trans} which accounts for the disturbances generated outside of ω and propagated through its boundary

$$e_\omega = e_\omega^{\text{cell}} + e_\omega^{\text{trans}} \tag{4.2.1}$$

A numerical study [243] revealed that the error correlates well to the transmitted error but shows little similarity to the cell error as well as to the local residual. This has significant implications in the context of mesh adaptation. In particular, grid refinement based on monitoring the local residuals may fail to refine all elements where $\|e_\omega\|$ is large since the transmitted error is ignored [243]. On the other hand, the local cell errors can be used to control the error e_0 restricted to some element Ω_0 at least for scalar conservation laws. In particular, the sum of local errors e_k^{cell} for all elements Ω_k located in the domain of dependence of Ω_0 provide an upper bound of e_0 [242].

In most engineering applications, not only the global flow field is required but also specific, scalar-valued quantities of interest $J(u)$ that depend on the solution u. Hence, a second type of error comes into play, namely, the error of the linear or nonlinear output functional

$$e_J = J(u) - J(\mathrm{u}_h) \tag{4.2.2}$$

Typical quantities of interest are the drag and lift coefficients for the simulation of flow around an airfoil or boundary fluxes and pressure-drop between inflow and outflow in internal flows. It is also possible to measure localized quantities such as the average velocity of the computed field $\mathbf{v}$ in the subset $\omega \subset \Omega$ in a prescribed direction $\mathbf{d}$ by adopting the linear functional [211]

$$J(\mathbf{v}) = \frac{1}{|\omega|}\int_\omega \mathbf{v}\cdot\mathbf{d}\,\mathrm{d}\mathbf{x} \tag{4.2.3}$$

Owing to the linearity of the above functional, the error in $J(\mathbf{v})$ is then defined by

$$e_J = J(\mathbf{v}) - J(\mathbf{v}_h) = J(\mathbf{e}_v) = \frac{1}{|\omega|}\int_\omega \mathbf{e}_v\cdot\mathbf{d}\,\mathrm{d}\mathbf{x}, \qquad \mathbf{e}_v = \mathbf{v} - \mathbf{v}_h \tag{4.2.4}$$

If the error of a nonlinear output functional such as the kinetic energy of the velocity field $\mathbf{v}$

$$N(\mathbf{v}) = \frac{1}{2}\int_\Omega \mathbf{v}^2\,\mathrm{d}\mathbf{x} \tag{4.2.5}$$

is sought, it is advisable to linearize the resulting error relation [211]

$$N(\mathbf{v}) - N(\mathbf{v}_h) = \frac{1}{2}\int_\Omega (\mathbf{v}_h + \mathbf{e}_v)^2 - \mathbf{v}_h^2\,\mathrm{d}\mathbf{x} = \int_\Omega \mathbf{v}_h\cdot\mathbf{e}_v\,\mathrm{d}\mathbf{x} + \frac{1}{2}\int_\Omega \mathbf{e}_v^2\,\mathrm{d}\mathbf{x} \tag{4.2.6}$$

by neglecting the second term which corresponds to the second derivative of $N(\cdot)$ with respect to the variable $\mathbf{v}_h$. This simplification allows to define the error in terms of the linear functional

$$e_J = \int_\Omega \mathbf{v}_h \cdot \mathbf{e}_v \, \mathrm{d}\mathbf{x} \tag{4.2.7}$$

provided the error is reasonably small so that its square can be safely neglected. Most quantities of interest can be expressed as linear or linearized functionals so that goal-oriented error estimation strategies can be developed in a general framework. Note that global error measures such as the commonly used energy norm error can also be expressed in terms of output functionals [26]

$$J(\varphi) = \frac{(\nabla\varphi, \nabla e)}{\|\nabla e\|} \qquad \Rightarrow \qquad e_J = J(u) - J(\mathrm{u}_h) = J(e) = \|\nabla e\| \tag{4.2.8}$$

whereby the error $e = u - \mathrm{u}_h$ of the scalar solution vector is considered as a fixed quantity.

In this work, the numerical error will only serve as an indicator which is used to steer the mesh adaptation algorithm. Hence, the process of error indication should be computationally inexpensive and easy to implement into existing finite element codes. We therefore neglect the temporal as well as the transmitted error and perform grid refinement and re-coarsening based on the local element error of the current solution. In what follows, a short and certainly incomplete overview of existing approaches is given which are used to estimate the solution error *a posteriori*.

4.2.2. Reconstruction-based techniques

The basic idea of this class of estimators is to compare the approximate solution or its gradient to an improved reconstruction of the corresponding quantity so as to obtain an estimate of the true error. By construction, reconstruction-based error estimators can only detect features which are already present in the solution computed on a given grid. In other words, they are doomed to fail if the computational grid is too coarse or if the numerical scheme is overly diffusive so that shock waves are only resolved as humps. Moreover, such techniques do not account for the temporal development of the error since they only investigate the solution values at a given time instant.

Error estimation by Richardson's extrapolation

A common approach to the adaptive treatment of hyperbolic problems is based on Richardson's extrapolation which is used to estimate the truncation error of the numerical scheme and equidistribute the error by means of local grid refinement [28, 31]. Due to its simplicity it can be easily integrated into existing codes such as CLAWPACK [155] extending the algorithm by Berger and Colella [28] to wave-propagation schemes applied on rectangular curvilinear grids [30].

In its original form, Richardson's extrapolation requires the *a priori* knowledge of the order of approximation p which is problem-dependent. Let u_h and u_{rh} denote the solutions computed on two nested grids at the same time instant. Here, r represents the refinement factor and h stands for the mesh width of the fine grid which is contained inside the coarse grid. The refinement also applies to the time step. If the coarse grid solution is advanced from time t^n to $t^{n+1} = t^n + s\Delta t$ then s small time steps Δt are required to obtain the solution on the refined grid. To simplify the presentation, let the ratio of mesh width and time step size be constant so that $r = s$. Moreover, assume that the order of accuracy p is the same in time and space. It is worth mentioning that this restriction can be circumvented by adopting a more expensive convergence study as explained in a research article by Berger and Oliger [31]. If the true solution is sufficiently smooth, the local truncation error can be expressed by means of the following Taylor series expansion [31]

$$u(\mathbf{x}, t^{n+1}) - \mathrm{u}_h(\mathbf{x}, t^{n+1}) \approx r\tau, \qquad \tau = \Delta t \left[C_1 \Delta t^p + C_2 h^p\right] \tag{4.2.9}$$

where the higher-order terms have been neglected and the auxiliary coefficients C_1 and C_2 do not depend on the mesh width h. The truncated Taylor series for the coarse grid solution reads

$$u(\mathbf{x},t^{n+1}) - \mathrm{u}_{rh}(\mathbf{x},t^{n+1}) \approx r^{p+1}\tau \tag{4.2.10}$$

and subtraction of equation (4.2.9) yields an estimate of the local truncation error [31]

$$\frac{\mathrm{u}_h(\mathbf{x},t^{n+1}) - \mathrm{u}_{rh}(\mathbf{x},t^{n+1})}{r^{p+1}-r} \approx \tau \tag{4.2.11}$$

at time t^{n+1}. As a matter of fact, Richardson's extrapolation constitutes a usable tool to obtain a global *a posteriori* estimate of the true error $e = u - \mathrm{u}_h$ on the finest grid in a suitable norm [76]

$$\|e_h\| = \frac{\|\mathrm{u}_h - \mathrm{u}_{rh}\|}{|r^p - 1|} \approx \|e\| \tag{4.2.12}$$

It has been already mentioned that the spatial order of accuracy p is problem-dependent. Moreover, the observed order of convergence may differ from its theoretical value due to the presence of discontinuities, the way in which boundary conditions are implemented or the use of nonlinear flux/slope limiters. In practice, it needs to be determined numerically by means of a grid refinement study. Ferziger and Perić [77] employ a sequence of three nested grids with constant refinement factor r. Note that the error due to the time discretization needs to be negligibly small if the spatial approximation is of interest. Let u_h, u_{rh} and u_{r^2h} denote a sequence of consecutive solutions which are required to be in the asymptotic convergence range, that is, they change monotonically as the grid is refined. Then the rate of convergence can be estimated globally by means of the L_2-norm as follows (e.g., $r = 2$ if the spacing is halved [76])

$$p = \log\left(\frac{\|\mathrm{u}_{r^2h} - \mathrm{u}_{rh}\|}{\|\mathrm{u}_{rh} - \mathrm{u}_h\|}\right)\frac{1}{\log r} \tag{4.2.13}$$

Roache [222] extends this type of error estimation technique by means of Richardson's extrapolation to sequences of grids which are not systematically refined or coarsened. An alternative approach was proposed by Schall et al. [230] to certify the accuracy of the numerical solution by means of a grid convergence analysis. To be more precise, a sequence of three anisotropic meshes consisting of N_V, $4N_\mathrm{V}$ and $16N_\mathrm{V}$ vertices is generated which are adapted individually so as to obtain the best approximate solution with a fixed number of nodes. From that, all solutions are transferred to the finest grid so that the global order of convergence can be determined similarly to (4.2.13). If *early second-order mesh convergence* cannot be observed, that is, $p \notin [1.6, 2.2]$, then a finer mesh consisting of $32N_\mathrm{V}$ vertices is generated and the whole procedure is repeated.

Since the order p may vary locally due to the presence of discontinuities and/or the application of nonlinear flux/slope limiting schemes Roy [226] suggested the use of first- and second-order extrapolation, whereby the error can be estimated from the sequence of approximate solutions:

$$u(\mathbf{x},t) - \mathrm{u}_h(\mathbf{x},t) \approx \frac{(r^2+r-1)(\mathrm{u}_{rh} - \mathrm{u}_h) - (\mathrm{u}_{r^2h} - \mathrm{u}_{rh})}{(r+1)(r-1)^2} \tag{4.2.14}$$

The main disadvantage of extrapolation based error estimators is that this approach relies on the local smoothness of the solution. Thus, this technique *may* be unreliable for hyperbolic problems which exhibit shock waves and discontinuities. Moreover, coarser grids may completely fail to resolve small scale features. Hence, the coarsest of the three grids already needs to be sufficiently fine so as to guarantee asymptotic convergence behavior so that Richardson's extrapolation can become quite costly when applied to hyperbolic problems. Despite this lack of theoretical backing, several researchers successfully employed this estimation technique for the simulation of compressible flows [223] and found it even more reliable than other approaches [111].

Recovery-based error estimators

In 1987, Zienkiewicz and Zhu [274] suggested a new type of error estimator based on gradient recovery. The 'simple error estimator for practical engineering analysis' presented for linear elastic problems is motivated by the observation that the solution gradient $\nabla \mathrm{u}_h$ of a piecewise continuous finite element approximation u_h exhibits discontinuities at element interfaces. Provided the true solution u is sufficiently smooth, these slope jumps may serve as an indicator for the numerical error. If the unknown gradient of the exact solution is replaced by a smoothed slope $\hat{\nabla}\mathrm{u}_h$ recovered from the finite element approximation, a computable error estimator can be derived [274]:

$$\|\hat{e}\| = \sqrt{\int_\Omega |\hat{\nabla}\mathrm{u}_h - \nabla \mathrm{u}_h|^2 \,\mathrm{d}\mathbf{x}} \approx \|e\| \tag{4.2.15}$$

The idea of using recovery techniques to obtain improved gradient values exhibits quite a long tradition in finite elements (cf. Oden et al. [193, 195] or Hinton and Campbell [103]). To the author's best knowledge, Cantin et al. [45] were the first to consider so-called averaging projection schemes for calculating an improved approximation to the consistent stresses. Zienkiewicz and Zhu employed this technique in their original article on recovery-based error estimation [274]. Ainsworth et al. [4] consider averaging projection schemes in a more general framework to analyze their convergence properties and give some advice concerning the aspect of efficiency.

In 1992, Zienkiewicz and Zhu proposed the superconvergent patch recovery (SPR) technique [275, 276] which is described in more detail in Section 4.3.2 of this thesis. As an alternative, polynomial preserving recovery (PPR) schemes [271] compute the smoothed gradient vector by differentiating a higher-order polynomial approximation of the solution values. Recently, Zhang and Naga [272] introduced a meshless recovery procedure which abandons the concept of element patches in favor of spherical patches which can be expanded adaptively. This strategy overcomes the solvability problem for the patchwise systems of equations which may be under-determined for an insufficient number of sampling points likely to occur at the boundary.

The ease of implementation, robustness, and accuracy in many situations have promoted the popularity of recovery based adaptive schemes especially in the engineering community. However, a word of caution is mandatory: Ainsworth and Oden [194] construct a one dimensional example in which the estimated error equals zero while the exact error can be arbitrary large. Problems have also been reported in applying this methodology to compressible flows using classical finite element or finite volume schemes [211]. In essence, shock waves are typically smeared across several elements and captured as linear approximation with steep gradients. As a consequence, the jumps across element interfaces are very small and the error predicted by the recovery procedure tends to zero at the location of the 'discontinuity' [210]. Hence, mesh refinement is forced in the vicinity of the shock but not at its core. Yet, it is questionable if this phenomenon can be attributed to the gradient reconstruction or to the overly diffusive discretization scheme employed.

4.2.3. Residual-based error estimators

Another class of *a posteriori* error estimates which has been extensively considered in the literature relates the residual of the discrete solution to the numerical error. As an example [211], consider the steady-state convection diffusion equation for the scalar-valued quantity u

$$\mathbf{v} \cdot \nabla u - d\Delta u = f \qquad \text{in} \quad \Omega \tag{4.2.16}$$

$$u = 0 \qquad \text{on} \quad \Gamma_\mathrm{D} \tag{4.2.17}$$

$$\mathbf{n} \cdot \nabla u = g \qquad \text{on} \quad \Gamma_\mathrm{N} \tag{4.2.18}$$

where $f \in L^2(\Omega)$, $d > 0$ denotes the diffusion coefficient and the velocity $\mathbf{v}$ is assumed to be solenoidal, i.e. $\nabla \cdot \mathbf{v} = 0$. Its weak form is obtained by multiplying the equation by a suitable test function, integrating over the domain and setting the result equal to zero. Moreover, integration by parts is applied to the high-order term which yields the weak form

$$\int_\Omega w(\mathbf{v}\cdot\nabla u) + d\nabla w \cdot \nabla u \,\mathrm{d}\mathbf{x} = \int_\Omega w f \,\mathrm{d}\mathbf{x} + \int_{\Gamma_\mathrm{N}} w g \,\mathrm{d}s \tag{4.2.19}$$

whereby the boundary conditions (4.2.17)–(4.2.18) have been replaced in the boundary integral. It is common practice to introduce the following short hand notation

$$A(u,w) := \int_\Omega w(\mathbf{v}\cdot\nabla u) + d\nabla w \cdot \nabla u \,\mathrm{d}\mathbf{x}, \qquad F(w) := \int_\Omega w f \,\mathrm{d}\mathbf{x} + \int_{\Gamma_\mathrm{N}} w g \,\mathrm{d}s \tag{4.2.20}$$

so that the problem at hand can be formulated as follows [211]: Find $u \in \mathcal{W}$ such that

$$A(u,w) = F(w), \qquad \forall w \in \mathcal{W} \tag{4.2.21}$$

where the Dirichlet boundary values are built into the space $\mathcal{W} = \{w \in \mathcal{H}^1(\Omega) : w = 0 \text{ on } \Gamma_\mathrm{D}\}$. Consider a conforming finite element approximation and let the standard Galerkin approach be adopted which amounts to finding the approximate solution $\mathrm{u}_h \in \mathcal{W}_h \subset \mathcal{W}$ such that

$$A(\mathrm{u}_h, w) = F(w), \qquad \forall w \in \mathcal{W}_h \tag{4.2.22}$$

Due to the use of conforming finite elements, the error $e = u - \mathrm{u}_h$ belongs to $\mathcal{W}$ as well. Replacing u by $\mathrm{u}_h + e$ in (4.2.21) and subtracting $A(\mathrm{u}_h, w)$ from both sides of the equation yields

$$A(\mathrm{u}_h + e, w) - A(\mathrm{u}_h, w) = R_h(w), \qquad \forall w \in \mathcal{W} \tag{4.2.23}$$

where the linear functional $R_h(w) = F(w) - A(\mathrm{u}_h, w)$ denotes the residual. In particular, the residual vanishes on $\mathcal{W}_h$ which is typically referred to as Galerkin orthogonality property

$$R_h(w) = 0, \qquad \forall w \in \mathcal{W}_h \tag{4.2.24}$$

In general, computing an approximation to equation (4.2.23) is as expensive as finding the solution of the original problem (4.2.21). Hence, the task is to *estimate* the norm of the residual so as to obtain computable upper and lower bounds for the error with positive constants C_1 and C_2 [211]

$$C_1 \|e\|_\mathcal{W} \leq \|R_h\|_* \leq C_2 \|e\|_\mathcal{W} \tag{4.2.25}$$

Here, $\|e\|_\mathcal{W}$ is a suitable norm of the error e in space $\mathcal{W}$ and $\|R_h\|_*$ denotes the dual norm of the weak residual with respect to $\|\cdot\|_\mathcal{W}$ which is typically defined by the following expression

$$\|R_h\|_* := \sup_{\substack{w\in\mathcal{W}\\ w\neq 0}} \frac{|R_h(w)|}{\|w\|_\mathcal{W}} \tag{4.2.26}$$

For the convection diffusion problem (4.2.16), the error can be measured as follows [211]:

$$\|e\|_\mathcal{W} = \sqrt{\int_\Omega \nabla e \cdot \nabla e \,\mathrm{d}\mathbf{x}} \tag{4.2.27}$$

Moreover, it is possible to prove that $C_1 = 1$ and $C_2 = 1 + M_0$ in the estimate (4.2.25), where the positive constant M_0 depends on the solenoidal velocity field $\mathbf{v}$ so as to ensure that [211]

$$\left| \int_\Omega w(\mathbf{v}\cdot\nabla u) \,\mathrm{d}\mathbf{x} \right| \leq M_0 \|u\|_\mathcal{W} \|w\|_\mathcal{W}, \qquad \forall u, w \in \mathcal{W} \tag{4.2.28}$$

In light of the above, the norm of the residual should in general provide a reasonable estimate of the solution error provided the values of the constants C_1 and C_2 are not far away from unity. For many practical applications, their value depends on various other constants, and hence, it is difficult or even impossible to obtain sharp estimates of the form (4.2.25). For a detailed presentation of residual-based error estimation techniques the interested reader is referred to the survey article by Prudhomme and Oden [211], and the monographs by Ainsworth and Oden [3] and Verfürth [259].

In general, it is impossible to compute the supremum norm (4.2.26) directly so that an *a posteriori* estimate of the residual is needed. Since the residual vanishes for all $w \in \mathcal{W}_h$, finer meshes and/or different function spaces need to be considered for the evaluation of the residual term. In contrast, a numerical solution u_h^* computed by some high-resolution finite element scheme based on algebraic flux correction [139] does not satisfy the Galerkin orthogonality property (4.2.24). In particular, the high-order Galerkin solution u_h is 'perturbed' by some discrete stabilization so that $R_h(w) = F(w) - A(\mathrm{u}_h^*, w)$ does not vanish on the function space $\mathcal{W}_h$. However, it can be directly computed for a given function $w \in \mathcal{W}_h$ and used within standard residual-based error estimators.

Remark. For the convection diffusion problem (4.2.16), the solution error is naturally related to the residual via the error equation (4.2.23). This approach can be extended to general conservation laws of the form $\partial_t u + \nabla \cdot \mathbf{f}(u) = 0$, whereby the residual R_h serves as a source term [147]

$$\frac{\partial e}{\partial t} + \nabla \cdot \tilde{\mathbf{f}}(e) = R_h \tag{4.2.29}$$

and the modified flux function is given by $\tilde{\mathbf{f}}(e) = \int_0^1 \nabla \cdot \mathbf{f}(\xi \mathrm{u}_h + (1-\xi)(\mathrm{u}_h - e))\,\mathrm{d}\xi$. Zhang et al. [270] utilize a similar approach for the treatment of the compressible Euler equations, whereby the above equation is solved explicitly adopting a numerical method of higher-order approximation.

4.2.4. Goal-oriented error estimators

As pointed out in the introduction, engineers are more interested in controlling the error of local quantities which are given by linear or nonlinear functionals of the field variables. To illustrate the basic ideas of goal-oriented error estimation techniques consider the variational form (4.2.21) for the steady convection diffusion problem (4.2.16)–(4.2.18) and let $J(\cdot)$ be some linear output functional as discussed in Section 4.2.1. The task is to control the error of the quantity of interest

$$e_J = J(u) - J(\mathrm{u}_h) = J(e), \qquad e = u - \mathrm{u}_h \tag{4.2.30}$$

Becker and Rannacher [26] utilized an 'optimal control approach' so as to relate the output functional $J(\cdot)$ to the exact solution $z \in \mathcal{W}$ of the adjoint/dual problem as follows

$$A(w,z) = J(w), \qquad \forall w \in \mathcal{W} \tag{4.2.31}$$

By construction, the dual solution z and the exact solution $u \in \mathcal{W}$ of the primal problem (4.2.21) are mutually adjoint to each other so that the following relation holds [26]

$$J(u) = A(u,z) = F(z) \tag{4.2.32}$$

Let the numerical solution $\mathrm{u}_h \in \mathcal{W}_h \subset \mathcal{W}$ to the primal problem (4.2.21) be computed by means of Galerkin finite elements. As a result, the error (4.2.30) can be expressed as follows

$$J(u) - J(\mathrm{u}_h) = A(u,z) - A(\mathrm{u}_h, z) = A(u - \mathrm{u}_h, z) \tag{4.2.33}$$

Owing to the Galerkin orthogonality property $A(u - \mathrm{u}_h, \mathrm{z}_h) = 0$, $\forall \mathrm{z}_h \in \mathcal{W}_h \subset W$, the error in the output functional can be related to the errors of the primal and dual problems [26]

$$J(u) - J(\mathrm{u}_h) = A(u - \mathrm{u}_h, z - \mathrm{z}_h) \tag{4.2.34}$$

Moreover, the above expression yields $J(u) - J(\mathrm{u}_h) = F(z - \mathrm{z}_h)$, provided that the dual solution z_h also exhibits the Galerkin orthogonality property, that is, $A(\mathrm{u}_h, z - \mathrm{z}_h) = 0$, $\forall\, \mathrm{u}_h \in \mathcal{W}_h \subset \mathcal{W}$.

In essence, there exist two alternative approaches to estimate the modulus of the right-hand side of equation (4.2.34). Johnson et al. [73] make use of the Cauchy-Schwartz inequality and 'eliminate' the term $z - \mathrm{z}_h$ to obtain the global *a posteriori* estimate

$$|J(u) - J(\mathrm{u}_h)| \leq C_{\mathrm{int}} C_{\mathrm{stab}} \|h^s R_h\|_*, \qquad s \geq 0 \tag{4.2.35}$$

where the constants C_{int} and C_{stab} strongly depend on the problem at hand and are difficult to obtain in practice. In the dual-weighted residual (DWR) method proposed by Becker and Rannacher [26] the residual is decomposed into element contributions first. Afterwards, each cell contribution can either be estimated by means of the Cauchy-Schwartz inequality or computed numerically adopting local higher-order approximations or difference quotients.

Since the Galerkin orthogonality property is crucial for the derivation of equation (4.2.34) goal-oriented error estimation techniques cannot be applied directly to numerical solutions which were computed by the nonlinear high-resolution AFC schemes described in the previous chapter. Thus, the extension of the dual-weighted residual approach to this type of non-standard Galerkin schemes is outstanding, and in general, the design of error estimators for output functionals applicable in the context of the algebraic flux correction paradigm is a topic of ongoing research.

4.2.5. Error indicators

From the theoretical point of view, so-called feature indicators are less attractive but enjoy great popularity in practice. As the name suggests, these techniques are designed to identify the location and strength of special flow features such as discontinuities or large variations of the solution in smooth regions. We will present some popular error indicators presently in use.

First- or second-difference indicator

Assume that the numerical scheme used to compute the approximate solution u_h is first-order accurate so that the error is proportional to the H_1-seminorm of the exact solution u, that is

$$\|u - \mathrm{u}_h\|_0 \leq Ch|u|_1, \qquad |u|_1 = \sqrt{\sum_{|\alpha|=1} \|\partial^\alpha u\|_0^2} \tag{4.2.36}$$

Here, h denotes the characteristic mesh width and the constant C is independent of h. The jump in u may serve as an indicator for large variations in the solution. This so-called first-difference indicator empirically derived in [59, 60] computes the undivided first differences of the solution u_h. However, a suitable normalization strategy is mandatory. A detailed description is given by Shapiro [236] who promoted this indicator especially for transonic flows. In particular, Shapiro considered the compressible Euler equations and used the undivided first difference of the density as an indicator for mesh adaptation. If we assume the solution to be sufficiently smooth, then the error can be also approximated in the H_2-seminorm, that is, its second derivatives [196]

$$\|u - \mathrm{u}_h\|_0 \leq Ch^2|u|_2, \qquad |u|_2 = \sqrt{\sum_{|\alpha|=2} \|\partial^\alpha u\|_0^2} \tag{4.2.37}$$

Error indicators of such type have a common drawback: they are not dimensionless. As a result, they are more sensitive to strong features (e.g., shock waves) than to weaker features (e.g., contact discontinuities and shear layers). To overcome this drawback so-called multi-pass strategies have been suggested [2]. In the first pass, only strong features are detected and mesh adaptation is performed accordingly, and the focus is placed on weaker flow features in the second pass.

First- and second-difference indicators

Löhner [161] introduced a non-dimensional error indicator which is based on (4.2.37) scaled by the absolute values of the gradient so as to avoid the 'eating-up' effect in the presence of strong shocks. To illustrate this approach consider linear finite elements of constant length h in one space dimension so that Löhner's indicator [161] evaluated for the internal node i reads as follows

$$e_i = \frac{|\mathrm{u}_{i+1} - 2\mathrm{u}_i + \mathrm{u}_{i-1}|}{|\mathrm{u}_{i+1} - \mathrm{u}_i| + |\mathrm{u}_i - \mathrm{u}_{i-1}| + \delta_i}, \qquad \delta_i = C_n(|\mathrm{u}_{i+1}| + 2|\mathrm{u}_i| + |\mathrm{u}_{i-1}|) \tag{4.2.38}$$

The term δ_i serves as a 'noise' filter which is required to prevent the indicator from approaching unity due to small wiggles in the solution profile. The parameter C_n depends on the problem and the numerical algorithm used to solve the governing equations. In our experience, it is worthwhile to replace δ_i in the denominator by $\delta_i^* = \max\{\delta_i, \varepsilon\}$, where ε denotes some absolute tolerance, since δ_i does not reduce noise if the approximate solution u_h tends to zero. The above error indicator is normalized, i.e. $0 \leq e_i \leq 1$, so that preset tolerances may be employed for general classes of problems. Moreover, it can be applied to multiple indicator variables if mesh adaptation is to be performed for systems of conservation laws (see Chapter 5). In the context of finite element approximations, the multidimensional generalization of equation (4.2.38) reads

$$e_i = \sqrt{\sum_{\substack{\alpha,\beta \geq 1 \\ |\alpha+\beta|=2}} \frac{(D''_{\alpha,\beta})^2}{(D'_{\alpha,\beta} + D_{\alpha,\beta})^2}} \tag{4.2.39}$$

where the derivative terms for the node i are evaluated as follows [163]

$$D''_{\alpha,\beta} = \sum_j \left[\int_\Omega \left(\frac{\partial \varphi_i}{\partial x_\alpha} \right) \left(\frac{\partial \varphi_j}{\partial x_\beta} \right) \mathrm{d}\mathbf{x} \right] \mathrm{u}_j \tag{4.2.40}$$

$$D'_{\alpha,\beta} = \int_\Omega \left| \frac{\partial \varphi_i}{\partial x_\alpha} \right| \left| \sum_j \frac{\partial \varphi_j}{\partial x_\beta} \mathrm{u}_j \right| \mathrm{d}\mathbf{x} \tag{4.2.41}$$

$$D_{\alpha,\beta} = C_n \sum_j \left[\int_\Omega \left| \frac{\partial \varphi_i}{\partial x_\alpha} \right| \left| \frac{\partial \varphi_j}{\partial x_\beta} \right| \mathrm{d}\mathbf{x} \right] |\mathrm{u}_j| \tag{4.2.42}$$

The first term represents the second derivative of the finite element solution $\mathrm{u}_h = \sum_j \varphi_j \mathrm{u}_j$. Since linear finite elements are employed integration by parts is performed to shift one derivative to the test function φ_i, whereby the contribution from the arising boundary integral is neglected. The second term is proportional to the modulus of the gradient, i.e. $|\nabla \mathrm{u}_h| = |\sum_j \nabla \varphi_j \mathrm{u}_j|$ which is multiplied by the absolute values of the derivatives of nodal shape functions $|\nabla \varphi_i|$. The third term which serves as noise filter requires the absolute value of nodal solution values $|\mathrm{u}_j|$, and hence, it needs to be scaled by the moduli of the first derivatives of basis functions [160]. Since the gradient of linear finite elements is constant on each triangle, the above quantities can be computed efficiently without resorting to costly numerical integration. Explicit expressions for the first derivatives of linear basis functions are given in equation (A.19) in Appendix A.

The dimensionless error indicator (4.2.39) has been analyzed by Löhner [161, 162] and applied by various authors to perform challenging flow simulations in two and three space dimensions [24, 25, 164]. In this work, we make use of mixed computational grids consisting of triangles and quadrilaterals, whereby linear and bilinear finite elements are adopted. In light of the above, it is expedient to divide quadrilateral elements into two non-overlapping triangles so that all quantities (4.2.40)–(4.2.42) can be efficiently evaluated from constant derivative values.

4.2.6. Summary of error estimation techniques

The development of reliable *a posteriori* error estimators for convection dominated problems is an issue of utmost importance. Owing to the complicated error propagation mechanism it may be insufficient to measure only cell errors which correlate to local residuals. Thus, the transmitted error should be taken into account if rigorous *a posteriori* error estimates are required.

Error estimation based on Richardson's extrapolation has been successfully employed to steer mesh adaptation algorithms for the simulation of compressible flows. In practice, the solution is computed on a sequence of three nested grids in order to estimate the error of the solution on the finest mesh. This approach relies on a uniform order of spatial accuracy which is not satisfied for nonlinear high-resolution schemes based on flux limiters. A generalization to discretizations which exhibit a varying order of approximation is feasible but the main disadvantage of extrapolation based error estimators remains: the Taylor expansion relies on the local smoothness of the solution which may be not the case in convection dominated problems. Moreover, the coarsest grid must already be fine enough so as to capture all essential flow features such as shock waves and discontinuities. This restriction also applies to error estimation techniques based on gradient recovery. In essence, the consistent finite element gradient is compared to improved slope values which are computed, e.g., by averaging projection methods, the superconvergent patch recovery approach or so-called polynomial preserving recovery schemes.

Residual-based error estimation techniques make use of the fact that the cell error correlates well to the residual evaluated for each element. Standard finite element schemes exhibit the Galerkin orthogonality property which states that the residual vanishes for all functions from the discretized weighting space. To evaluate the residual term explicitly, it is common practice to employ higher-order basis functions and/or enrich the given function space, e.g., by edge shape functions or interior bubble shape functions. As an alternative, the residual can also be computed on a finer grid. Note that high-resolution schemes based on algebraic flux correction fail to possess the Galerkin orthogonality property. Hence, the residual does not vanish but its contribution is computable, and hence, it can be used within standard error estimators. Moreover, the flux function of the problem at hand can be modified to derive a conservation law for the solution error, whereby the residual serves as a source term. This error estimation technique has been applied to compressible flows with moderate success. However, the error equations needs to be solved with higher-order accuracy so that the computational costs are quite high in practice.

For engineering applications, the error is frequently measured in term of quantities of interest which are typically given by linear or nonlinear output functionals. Goal-oriented error estimation techniques can be based on an optimal control approach so as to relate the error of the (linearized) output functional to the global errors of the primal and the adjoint solution. The derivation relies on the Galerkin orthogonality of the finite element method and an extension of the dual-weighted residual approach applicable to algebraic flux correction schemes is still under development.

From the practical point of view, error estimators/indicators should be easy to implement into existing simulation tools and inexpensive to compute so that they can be applied to realistic problems. The first- and second-difference of the approximate solution have been used to control the solution error. Both approaches have difficulties if the flow field exhibits strong shocks as well as weak features which are not detected reliably unless multi-pass strategies are employed. As an efficient alternative, a dimensionless error indicator can be constructed by dividing the second derivative by the gradient values as suggested by Löhner [161], whereby the solution average serves as a noise filter to prevent the 'activation' of the indicator due to small wiggles. Since the indicator is bounded by zero and unity, it can be applied to multiple key variables simultaneously and a suitable combination thereof may be used to steer the grid adaptation algorithm.

4.3. Recovery-based error indicators

This section focuses on the design of recovery based *a posteriori* error indicators [185, 186] applicable to high-resolution finite element schemes for convection dominated flows. In particular, the developed techniques can be employed for scalar governing equations and systems of hyperbolic conservation laws alike. In the latter case, the recovery procedure is applied to some indicator variable which may be either a primary variable present in the solution vector or a derived quantity such as the Mach number. It is also possible to consider a set of indicator variables and use a combination of the estimated error to steer the mesh adaptation procedure. To keep the presentation self-contained, let us derive the *a posteriori* error indicator for a generic finite element solution

$$u \approx \mathrm{u}_h = \sum_{j=1}^{M} \mathrm{u}_j \varphi_j(\mathbf{x}) \tag{4.3.1}$$

where φ_j denote the set of conforming linear or bilinear basis functions spanning the finite-dimensional subspace $\mathcal{W}_h \subset \mathcal{W}$. Moreover, the temporal evolution of the flow is neglected so that u_h represents a scalar indicator variable evaluated at a fixed time t^n. In this section, we will concentrate on the numerical error resulting from the approximation of spatial derivatives. In particular, an *a posteriori* error indicator for the vector-valued gradient error

$$\mathbf{e} = \nabla u - \nabla \mathrm{u}_h \tag{4.3.2}$$

is devised. The consistent finite element gradient is computed from the approximate solution

$$\nabla \mathrm{u}_h = \sum_{j=1}^{M} \mathrm{u}_j \nabla \varphi_j(\mathbf{x}) \tag{4.3.3}$$

and the unknown values of the exact slopes ∇u are replaced by a smoothed gradient field $\hat{\nabla}\mathrm{u}_h$. As a result, a viable approximation to the true gradient error (4.3.2) may be computed as follows

$$\mathbf{e} \approx \hat{\mathbf{e}} = \hat{\nabla}\mathrm{u}_h - \nabla \mathrm{u}_h \tag{4.3.4}$$

Pointwise error estimates are difficult to obtain in general, so integral measures are typically employed in the finite element framework. Let the computational domain Ω be partitioned into a set of non-overlapping finite elements Ω_k so that the L_2-norm represents a usable measure for the global error, and moreover, its square can be assembled from local element errors

$$\|\hat{\mathbf{e}}\| = \left(\sum_{\Omega_k} (\hat{\mathbf{e}}, \hat{\mathbf{e}})_{\Omega_k} \right)^{1/2} \qquad (\hat{\mathbf{e}}, \hat{\mathbf{e}})_{\Omega_k} = \int_{\Omega_k} \hat{\mathbf{e}}^T \hat{\mathbf{e}} \, \mathrm{d}\mathbf{x} \tag{4.3.5}$$

For piecewise linear (P_1) finite elements on triangular meshes, the consistent gradient $\nabla \mathrm{u}_h$ is constant on each triangle, whereas linear variations are observed for bilinear (Q_1) elements. The details of linear and bilinear basis functions and explicit formulas for their first derivatives are given in Appendix A. In order to obtain an improved approximation to the exact gradient ∇u, let us specify slope values $\hat{\nabla}\mathrm{u}_{ij} = \hat{\nabla}\mathrm{u}_h(\mathbf{x}_{ij})$ at all midpoints of edges, i.e. $\mathbf{x}_{ij} := \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)$. For triangular elements, the smoothed gradient $\hat{\nabla}\mathrm{u}_h$ varies linearly in each Ω_k and is allowed to exhibit discontinuous jumps across inter-element boundaries. This approach can be interpreted as seeking the nodal values for a non-conforming gradient approximation by means of linear Crouzeix-Raviart elements [57] for which the local degrees of freedom are located at edge midpoints. As a generalization to bilinear finite elements used on quadrilateral meshes, a similar gradient approximation can be based on the non-conforming Rannacher-Turek element [212]. The reconstruction of gradient values at edge midpoints enables us to impose mathematical constraints on the recovered gradient which are inspired by common slope limiting techniques [185, 186].

4.3.1. Limited gradient averaging

The first approach to obtaining a smoothed edge gradient is inspired by slope limiting techniques employed in the context of high-resolution finite volume schemes and later carried over to discontinuous Galerkin finite element methods [53]. The basic ideas can be best illustrated by considering a finite volume discretization of the one-dimensional advection equation $\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0$.

Let the interval $I = (x_0, x_1)$ be partitioned into a set of finite volumes $I_j = (x_{j-1/2}, x_{j+1/2})$ on which the mean value $\bar{\mathrm{u}}_j$ of the solution is constant. The task is to define a suitable slope value u'_j on each cell so that a piecewise linear approximation can be defined as follows

$$\mathrm{u}_h(x) = \bar{\mathrm{u}}_j + \mathrm{u}'_j(x - x_j), \qquad \forall x \in I_j. \tag{4.3.6}$$

In the simplest case, one-sided or centered slopes can be utilized to obtain first- and second-order accurate schemes which lead to rather diffusive profiles and are quite likely to produce non-physical oscillations in the vicinity of steep fronts, respectively. For a numerical scheme to be non-oscillatory, it should possess certain properties [139], e.g., be monotone, positivity preserving, total variation diminishing (TVD) or local extremum diminishing (LED) as presented in Sections 3.1.2 and 3.1.3. In order to construct LED schemes [113], Jameson introduced a family of limited average operators $\mathfrak{L}(a,b)$ which are characterized by the following properties [114]:

$$\begin{aligned} &\text{(P1)} \quad \mathfrak{L}(a,b) = \mathfrak{L}(b,a) && \text{(P3)} \quad \mathfrak{L}(a,a) = a \\ &\text{(P2)} \quad \mathfrak{L}(ca,cb) = c\mathfrak{L}(a,b) && \text{(P4)} \quad \mathfrak{L}(a,b) = 0 \quad \text{if} \quad ab \le 0 \end{aligned}$$

While conditions (P1)–(P3) are natural properties of an average, (P4) is to be enforced by means of a limiter function. Jameson demonstrated that the most widely used TVD limiters can be cast into two-parameter form as presented in Figure 4.1. Here, the modified sign function is given by

$$\mathfrak{S}(a,b) = \frac{\operatorname{sign}(a) + \operatorname{sign}(b)}{2} \tag{4.3.7}$$

which equals zero for $ab \le 0$ and returns the common sign of a and b otherwise. As a result, the adjusted counterpart of u'_j in (4.3.6) can be computed as the limited average of left and right slopes

$$\hat{\mathrm{u}}'_j = \mathfrak{L}\left(\frac{\bar{\mathrm{u}}_{j-1} - \bar{\mathrm{u}}_j}{\Delta x}, \frac{\bar{\mathrm{u}}_{j+1} - \bar{\mathrm{u}}_j}{\Delta x}\right) \tag{4.3.8}$$

1. minmod: $\mathfrak{L}(a,b) = \mathfrak{S}(a,b)\,\min\{|a|,|b|\}$
2. maxmod: $\mathfrak{L}(a,b) = \mathfrak{S}(a,b)\,\max\{|a|,|b|\}$
3. Van Leer: $\mathfrak{L}(a,b) = \mathfrak{S}(a,b)\,\dfrac{2|a||b|}{|a|+|b|}$
4. MC: $\mathfrak{L}(a,b) = \mathfrak{S}(a,b)\,\min\left\{\dfrac{|a+b|}{2}, 2|a|, 2|b|\right\}$
5. superbee: $\mathfrak{L}(a,b) = \mathfrak{S}(a,b)\,\max\{\min\{2|a|,|b|\}, \min\{|a|,2|b|\}\}$

Fig. 4.1. Standard TVD limiters in two-parameter form.

Let us return to our original task that involves the reconstruction of smoothed gradients at edge midpoints. This is where the benefit of an edge based formulation comes into play. Except for vertices which are located at the boundary, *exactly* two elements are adjacent to each edge *ij* such that an improved gradient can be determined efficiently as the limited average of the constant slope values to the left and to the right by letting each component be defined as follows [185, 186]

$$\hat{\nabla}\mathrm{u}_{ij} = \mathcal{L}(\nabla\mathrm{u}_{ij}^{+}, \nabla\mathrm{u}_{ij}^{-}) \tag{4.3.9}$$

It is easy to verify that for all limiter functions $\mathcal{L}$ presented in Figure 4.1, the recovered edge gradient is naturally bounded from below and above by the constant slope values

$$\nabla\mathrm{u}_{ij}^{\min} \leq \hat{\nabla}\mathrm{u}_{ij} \leq \nabla\mathrm{u}_{ij}^{\max}, \qquad \text{where} \qquad \nabla\mathrm{u}_{ij}^{\substack{\max\\\min}} = \begin{matrix}\max\\\min\end{matrix}\{\nabla\mathrm{u}_{ij}^{+}, \nabla\mathrm{u}_{ij}^{-}\} \tag{4.3.10}$$

The above inequality holds separately for each spatial component of the vector-valued gradients. Suppose the upper and lower bounds have different sign for the d^{th} spatial component:

$$\nabla\mathrm{u}_{ij}^{+}\big|_d \cdot \nabla\mathrm{u}_{ij}^{-}\big|_d < 0 \qquad \Rightarrow \qquad \hat{\nabla}\mathrm{u}_{ij}\big|_d = \mathcal{L}\left(\nabla\mathrm{u}_{ij}^{+}, \nabla\mathrm{u}_{ij}^{-}\right)\big|_d = 0 \tag{4.3.11}$$

According to the mean value theorem for continuous functions, the corresponding slope value should attain zero somewhere in-between. If this is true for all components of the gradient, the solution may attain a local extremum across the edge. In light of the above, property P4 of limited average operators acts as a discrete analog to the necessary condition for local extrema in the continuous case which requires the derivative to be zero [185].

The recovered gradient (4.3.9) depends on the choice of the limiter $\mathcal{L}$ to some extent. In the author's experience, MC has proven to be robust in practice as it tries to select the standard average of the left and right slopes whenever possible without violating the natural upper and lower bounds.

4.3.2. Limited gradient reconstruction

As an alternative to the limited averaging approach, traditional recovery procedures can be used to predict provisional gradient values at edge midpoints which are corrected by means of edgewise slope limiting so as to satisfy the geometric constraints set up by inequality (4.3.10). In their first paper on recovery-based error estimation [274], Zienkiewicz and Zhu make use of so-called averaging projection schemes to construct improved stress values from the finite element solution

$$\hat{\nabla}\mathrm{u}_h = \sum_{j=1}^{\hat{M}} \hat{\nabla}\mathrm{u}_j \phi_j(\mathbf{x}) \tag{4.3.12}$$

where the coefficients $\hat{\nabla}\mathrm{u}_j$ are obtained by solving the discrete problem

$$\int_\Omega \phi_i(\hat{\nabla}\mathrm{u}_h - \nabla\mathrm{u}_h)\,\mathrm{d}\mathbf{x} = 0, \qquad \forall\phi_i \in \hat{\mathcal{W}}_h \tag{4.3.13}$$

In general, the element shape functions used to construct the basis functions ϕ_j may differ from those adopted in the finite element approximation (4.3.1). A detailed analysis by Ainsworth et al. [4] revealed the fact that the corresponding polynomial degrees should satisfy $\deg\phi \geq \deg\varphi$ whereby the original choice $\phi = \varphi$ by Zienkiewicz and Zhu [274] 'is not only effective, but also the most economical' [4] one. This has also been mentioned in an earlier publication by Oden and Brauchli [193]. The substitution of expansion (4.3.12) into equation (4.3.13) yields

$$\sum_{j=1}^{\hat{M}} \left[\int_\Omega \phi_i\phi_j\,\mathrm{d}\mathbf{x}\right] \hat{\nabla}\mathrm{u}_j - \sum_{j=1}^{M} \left[\int_\Omega \phi_i\nabla\varphi_j\,\mathrm{d}\mathbf{x}\right] \mathrm{u}_j = 0 \tag{4.3.14}$$

which amounts to solving a linear algebraic system for each component of the smoothed gradient

$$M_C \hat{\nabla} \mathrm{u}_h = \mathbf{C} \mathrm{u}_h \tag{4.3.15}$$

The consistent mass matrix $M_C = \{m_{ij}\}$ and the multi-component operator $\mathbf{C} = \{\mathbf{c}_{ij}\}$ which corresponds to the discretized spatial derivatives are assembled from the integral terms

$$m_{ij} = \int_\Omega \phi_i \phi_j \, \mathrm{d}\mathbf{x}, \qquad \mathbf{c}_{ij} = \int_\Omega \phi_i \nabla \varphi_j \, \mathrm{d}\mathbf{x} \tag{4.3.16}$$

The coefficients m_{ij} and $\mathbf{c}_{ij}$ remain unchanged as long as the mesh is kept fixed, and hence, they need to be evaluated once during the initialization step and each time the grid has been adapted. If the same set of basis functions is used for the numerical solution and the approximate gradient, i.e. $\phi \equiv \varphi$, then the coefficients defined in (4.3.16) coincide with those required to assemble the finite element matrices and are thus available at no additional costs. The discrete operator $\mathbf{C}$ has zero row sums as long as the sum of basis functions equals one at every point so that each nodal component of the right-hand side in equation (4.3.15) can be assembled edge-by-edge [185, 186]

$$\sum_{j=1}^{M} \mathbf{c}_{ij} = 0 \qquad \Rightarrow \qquad (\mathbf{C} \mathrm{u}_h)_i = \sum_{j \neq i} \mathbf{c}_{ij} (\mathrm{u}_j - \mathrm{u}_i) \tag{4.3.17}$$

Note that the system of algebraic equations (4.3.15) can also be obtained by applying the Galerkin approximation to the weak form of the continuous problem $\hat{\nabla} u = \nabla u$, where the weighting and basis functions may or may not be the same. Therefore, projection schemes of the form (4.3.12)–(4.3.13) are called variational recovery schemes [172], and moreover, they can be applied repeatedly so as to approximate higher-order derivative [163]. In any case, the solution to the linear problem (4.3.15) can be computed iteratively by means of successive approximation, whereby the lumped mass matrix $M_L = \mathrm{diag}\{m_i\}$, where $m_i = \sum_j m_{ij}$ provides an excellent preconditioner

$$\hat{\nabla} \mathrm{u}_h^{(m+1)} = \hat{\nabla} \mathrm{u}_h^{(m)} + M_L^{-1} \left[\mathbf{C} \mathrm{u}_h - M_C \hat{\nabla} \mathrm{u}_h^{(m)} \right], \qquad m = 0, 1, 2, \ldots \tag{4.3.18}$$

Mass lumping can also be applied directly to the algebraic equation (4.3.15) which yields an explicit formula for computing the values of the projected gradient at each node

$$\hat{\nabla} \mathrm{u}_i = \frac{1}{m_i} \sum_{j \neq i} \mathbf{c}_{ij} (\mathrm{u}_j - \mathrm{u}_i) \tag{4.3.19}$$

In general, provisional slopes at edge midpoints can be interpolated from the nodal values obtained either from (4.3.15) or (4.3.19) making use of the finite element representation (4.3.12). For linear and bilinear finite elements this corresponds to taking the mean values for each edge *ij*, that is,

$$\hat{\nabla} \mathrm{u}_{ij} = \hat{\nabla} \mathrm{u}_h(\mathbf{x}_{ij}) = \frac{\hat{\nabla} \mathrm{u}_i + \hat{\nabla} \mathrm{u}_j}{2} \tag{4.3.20}$$

It is also possible to recover smoothed slope values directly in the edge midpoints by using nonconforming Crouzeix-Raviart and/or Rannacher-Turek elements in the projection scheme (4.3.15).

An alternative superconvergent patch recovery technique (SPR) has been suggested by Zienkiewicz and Zhu [275]. It is based on the observation that the finite element solution exhibits superconvergence at some exceptional points which are known *a priori* for common finite elements. Let the smoothed gradient be represented in terms of a polynomial expansion of the form

$$\hat{\nabla} \mathrm{u}_h = P(\mathbf{x}) \mathbf{a} \tag{4.3.21}$$

where $P(\mathbf{x})$ denotes a polynomial that is of the same degree and completeness as the one employed to approximate the original solution u_h. For linear and bilinear finite elements it contains all first-order monomials, that is, $P(\mathbf{x}) = [1, x, y]$ and $P(\mathbf{x}) = [1, x, y, xy]$, respectively. The multi-component vector of coefficients $\mathbf{a} = [a_1, a_2, \ldots, a_m]^T$ contains the nodal values of the recovered gradient.

For each vertex there exists a patch $\mathcal{P}_k$ of elements surrounding this node. As an alternative to this node-base approach, element-based patches can be constructed by 'gluing' together all cells which are adjacent to a given element. Last but not least, the selection can be restricted to face-neighboring cells [5]. All possible choices for constructing element-based and node-based patches are illustrated in Figure 4.2. Let the set of sampling points for a given patch be defined as follows:

$$\mathcal{S}_k = \left\{ j : \mathbf{x}_j \in \mathcal{P}_k \right\} \tag{4.3.22}$$

To compute the improved gradient $\hat{\nabla}\mathrm{u}_h$ it suffices to solve the minimization problem [275]

$$F(\mathbf{a}) = \sum_{j=1}^{|\mathcal{S}_k|} (\hat{\nabla}\mathrm{u}_j - \nabla\mathrm{u}_j)^2 \quad \rightarrow \quad \min \tag{4.3.23}$$

The unknown coefficients $\mathbf{a}$ can be computed by seeking a least squares fit through all (superconvergent) sampling points. Substitution of interpolations (4.3.21) and (4.3.12) in the corresponding least squares minimization leads to the local algebraic problem $M\mathbf{a} = \mathbf{f}$, where

$$M = \sum_{j \in \mathcal{S}_k} P^T(\mathbf{x}_j) P(\mathbf{x}_j), \qquad \mathbf{f} = \sum_{j \in \mathcal{S}_k} P^T(\mathbf{x}_j) \nabla\mathrm{u}_h(\mathbf{x}_j) \tag{4.3.24}$$

The number of rows in the least squares system is equal to the number of coefficients in the polynomial expansion P. Thus, the total number of sampling points $|\mathcal{S}_k|$ must be equal or greater than that quantity in order to obtain a unique solution. For linear and bilinear finite elements it

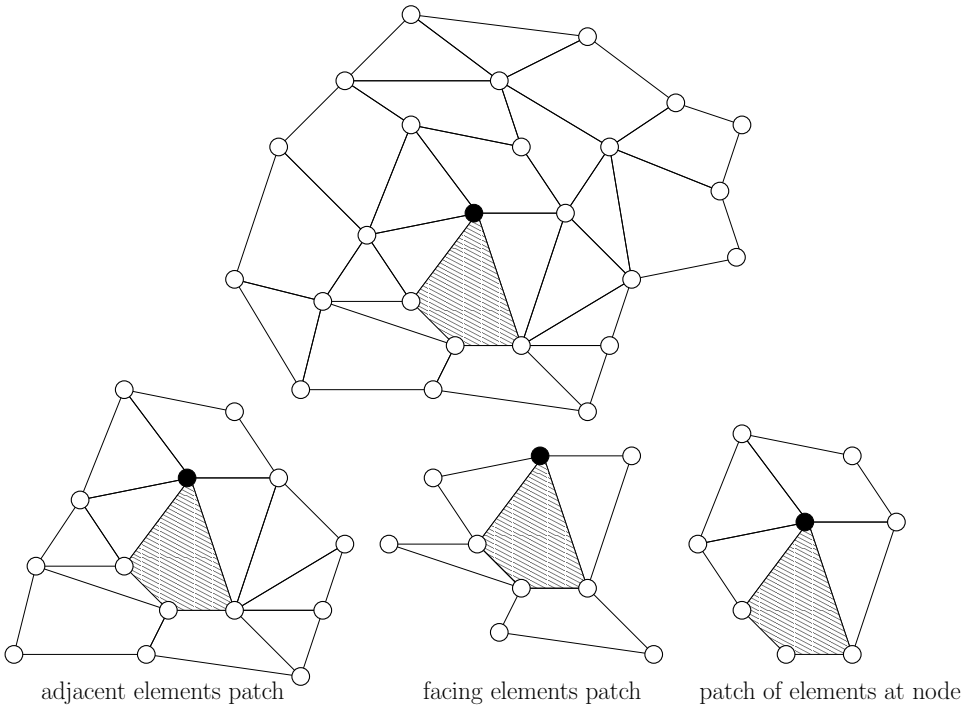


Fig. 4.2. Examples of element-based and node-based patches.

suffices to sample the consistent slopes at the superconvergent Gaussian quadrature points in the center of the element [275]. If patches are constructed in an element-based fashion at least three or four elements are adjacent to each cell in the interior so that the local minimization problems can be solved unconditionally. However, the number of adjacent elements may be insufficient to solve the minimization problem at the boundary. In particular, this problem occurs for corner vertices especially if node-based patches are employed. As a remedy, the slope values at all boundary nodes can be recovered with aid of interior patches as suggested by Zienkiewicz and Zhu [275] but it is easy to construct meshes for which two boundary components are only separated by one layer of elements such that no interior patches are available [146]. In the author's experience, an element-based construction of patches may be slightly more expensive due to a larger number of sampling points but solvability problems have rarely been observed in practice.

The practical implementation of the SPR algorithm has been addressed by Akin [5] who suggests a unified approach for mixed finite elements using a constant Jacobian for each patch. To illustrate the basic ideas consider a group of cells surrounding the 'master' element Ω_k as presented in Figure 4.3. For element-based patches, Ω_k corresponds to the cell for which the current patch $\mathcal{P}_k$ is created and it is the element with the highest polynomial degree otherwise. It is common practice in finite elements to introduce a one-to-one mapping $F_k : \hat{\Omega}_k \rightarrow \Omega_k$ which converts the physical cell into the reference element $\hat{\Omega}_k$ and vice versa. The details of this coordinate transformation are presented in Appendix A. As a generalization to the normalized reference element, let us make use of the bounding element $\bar{\Omega}_k$ which comprises all sampling points in the patch and exhibits the same coordinate alignment as $\hat{\Omega}_k$. In other words, the mapping $\bar{F}_k : \hat{\Omega}_k \rightarrow \bar{\Omega}_k$ features a constant Jacobian since points undergo a linear transformation and no deformation takes place. As a result, all sampling points $\mathbf{x}_j \in \mathcal{S}_k$ in the physical space can be efficiently converted to the non-dimensional coordinates $\bar{\mathbf{x}}_j$ adopting the constant Jacobian. The least squares fit process (4.3.24) is carried out in the patch-induced frame of reference and the reconstructed gradient values are converted back to physical coordinates afterwards. A very detailed description of this approach including a working package of Fortran 90 subroutines is provided by Akin [5].

The ease of implementation, generality and ability to produce quite accurate estimators boosted the popularity of recovery-based techniques especially in the engineering community. However, any of the above-mentioned strategies to predict the high-order gradient values may fail if the solution exhibits jumps or the gradient is too steep. This is typically the case for hyperbolic conservation laws such as the compressible Euler equations featuring strong shock waves.

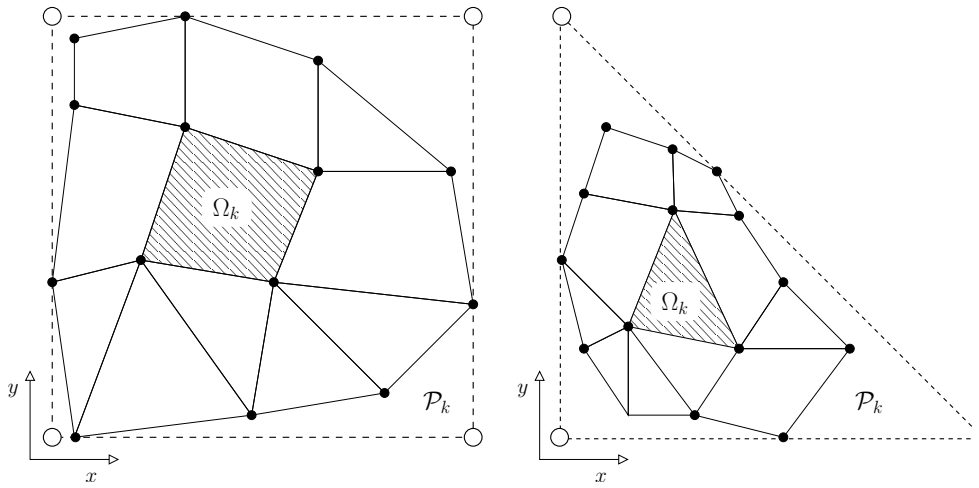


Fig. 4.3. Bounding a group of elements with a constant Jacobian patch.

Based on our experience in the design of high-resolution finite element schemes for convection dominated problems we suggested a suitable post-processing of the recovered gradient values of high order [185, 186]. The averaging process extends over an unsettled number of surrounding element gradients which may strongly vary in magnitude and even possess different signs. It is therefore difficult to find admissible upper and lower bounds to be imposed on such nodal gradients. On the other hand, the transition to an edge based formulation makes it possible to correct the provisional slope values at edge midpoints subject to their low-order counterparts evaluated in adjacent elements. It is also advisable to enforce the sign-preserving property (P4) of limited average operators so as to mimic the necessary condition of a local extremum [185, 186]

$$\hat{\nabla}\mathrm{u}_{ij}^{*} = s_{ij}\left|\max\{\nabla\mathrm{u}_{ij}^{\min},\min\{\hat{\nabla}\mathrm{u}_{ij},\nabla\mathrm{u}_{ij}^{\max}\}\}\right|, \qquad s_{ij} := \mathcal{S}(\nabla\mathrm{u}_{ij}^{\min},\nabla\mathrm{u}_{ij}^{\max}) \tag{4.3.25}$$

As a result, the corrected slope $\hat{\nabla}\mathrm{u}_{ij}^{*}$ is bounded from below and above by the low-order gradient values adjacent to the edge ij, and moreover, it equals zero if the consistent gradient changes its sign. For an interior edge, the interplay of quantities involved in this predictor-corrector *edgewise limited recovery* (ELR) procedure are illustrated in Figure 4.4. The generality of this approach allows for the use of arbitrary reconstruction techniques in the prediction step (L_2-projection and superconvergent patch recovery as well as polynomial-preserving recovery schemes [271] or meshless variants [272]) and apply edgewise slope-limiting as a black-box post-processing tool.

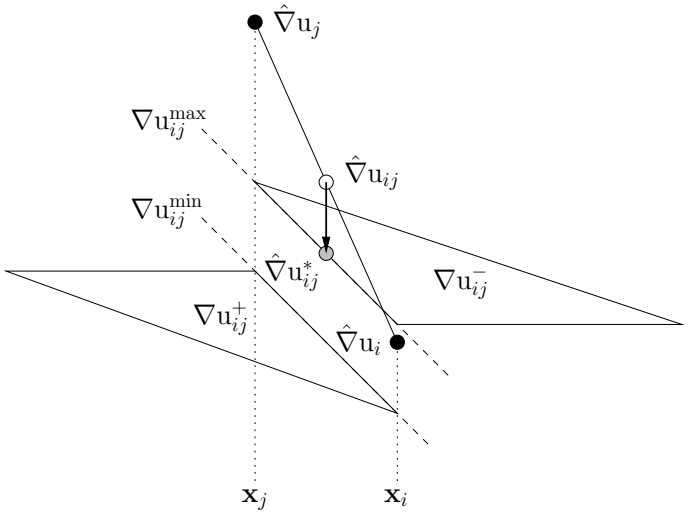


Fig. 4.4. Edgewise slope-limited gradient recovery.

4.3.3. Numerical examples for gradient-recovery techniques

The examples that follow illustrate the influence of the presented techniques on the quality of the recovered gradient. As a first test problem (TP1), consider the Gaussian hill function

$$u(x) = \alpha\exp(-\beta x^2), \qquad \alpha = 0.4, \quad \beta = 5 \tag{4.3.26}$$

which is smooth and attains its maximum value at the origin $x = 0$. The exact 'gradient' which is given by $u'(x) = -2\alpha\beta x\exp(-\beta x^2)$ is well-defined for all $x \in \mathbb{R}$. It exhibits two global extrema at $x_{\max,\min} = \pm 1/\sqrt{2\beta}$ as long as β is positive so that the profile (4.3.26) changes its curvature twice.

The various recovery procedures – edgewise limited averaging (ELA), superconvergent patch recovery (SPR), and L_2-projection with consistent and lumped mass matrix – have been applied

to the above model problem making use of successively refined meshes. The results obtained on the coarsest grid consisting of 10 uniform cells are depicted in Figure 4.5, where the exact slope values are indicated by dotted lines and the horizontal bars represent the constant finite element gradient. All recovery techniques fail to reproduce the nodal gradient values at the turning points except for the L_2-projection method (4.3.15) if the consistent mass matrix is adopted. On the other hand, this scheme tends to produce small overshoots and undershoots which can be observed in the upper right profile of Figure 4.5. All other methods are free of this drawback since the recovered nodal quantities are bounded by the constant finite element slopes of the adjacent elements. This property is a blessing and a curse. It prevents the formation of unphysical wiggles in the improved gradient profile, and at the same time, it leads to a poor resolution of local extrema. This so-called peak-clipping phenomenon is well known for high-resolution schemes based on flux/slope limiters which resort to overly diffusive low-order approximations in the vicinity of steep gradients.

The generation of undershoots and overshoots becomes less pronounced if the mesh is refined. The gradient values which have been recovered on a grid consisting of 40 uniform elements are depicted in Figure 4.6. The various recovery techniques slightly differ in the facility to capture the minimum and maximum gradient values accurately. On the other hand, it is nearly impossible to distinguish the computed gradient profiles visually apart from local extrema.

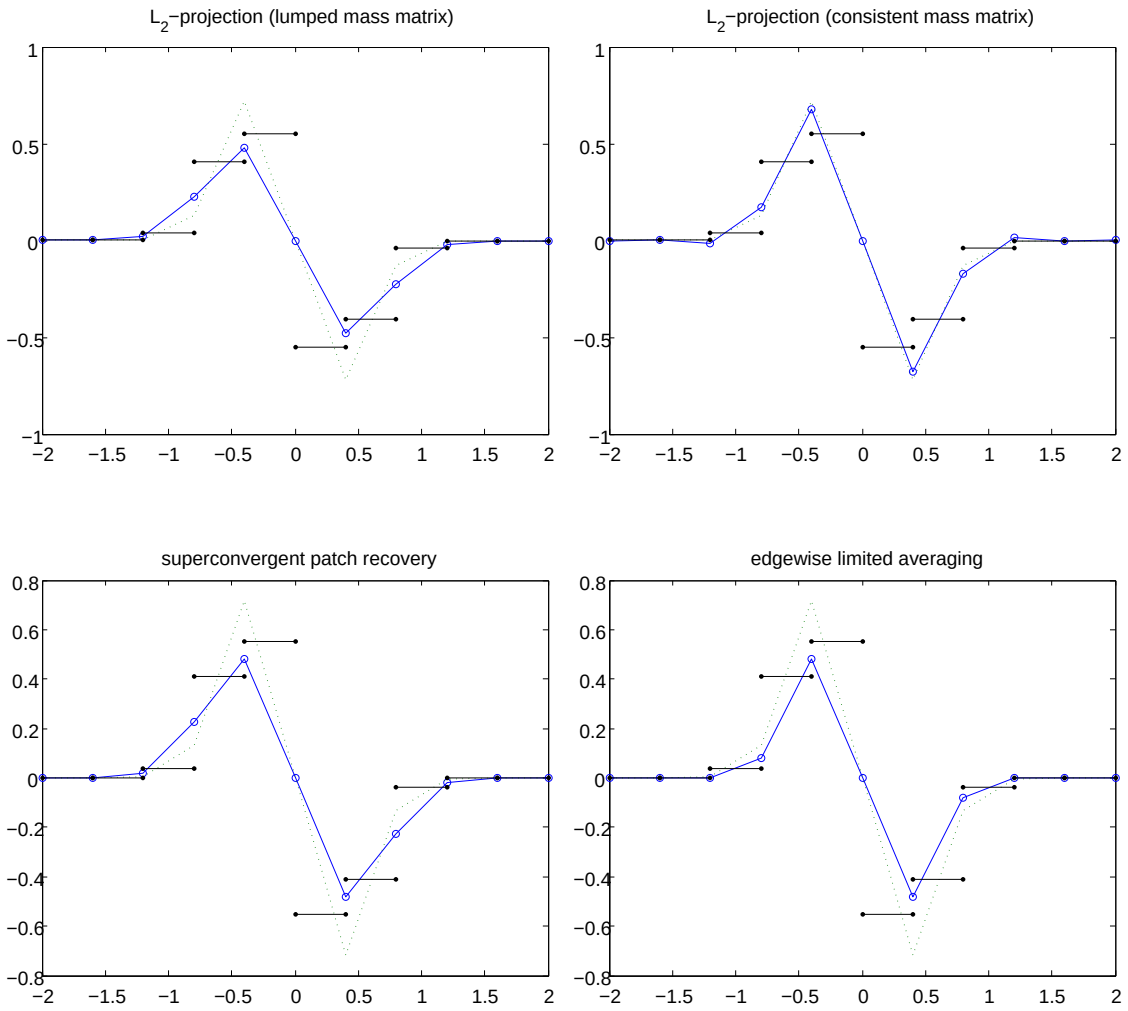


Fig. 4.5. TP1: Comparison of consistent and recovered gradient with 10 elements.

In order to assess the quality of the improved gradient values quantitatively, we measured the pointwise error between the exact slopes and the recovered gradient at $x_1 = 0.4$ and $x_2 = 1.2$ which are both present on the coarsest grid. The first point is located nearest to $x_{\min} = 1/\sqrt{10} \approx 0.3162$, where $u'(x_{\min}) = -\sqrt{1.6/\exp} \approx -0.7672$ attains its global minimum value. The second point $x_2 = 1.2$ is placed such that the recovered gradient values may suffer from unphysical oscillations if the variational recovery scheme (4.3.15) is employed with consistent mass matrix. The nodal convergence rates for the recovered slopes are depicted in Figure 4.7. The consistent finite element gradient ∇u_h is piecewise constant and exhibits jumps across element boundaries. Its 'nodal' values have been computed as the standard average of the two adjacent slope values which yields a linear reduction of the gradient error. All recovery techniques presented in Sections 4.3.1 and 4.3.2 are at least superconvergent, whereby the differences in the pointwise error is negligible for sufficiently fine grids. However, the absolute error of the 'improved' quantity $\hat{\nabla} u_h$ at point x_2 may even exceed that of the consistent finite element gradient unless edgewise limited averaging is employed to compute the nodal slope values. Consequently, equation (4.3.4) may provide an unusable estimate of the true gradient error for very coarse grids. It is worth mentioning that the error reduction rate is improved significantly if the nodal gradient values are recovered by means of consistent L_2-projection, whereby the same limitations apply on insufficiently fine meshes.

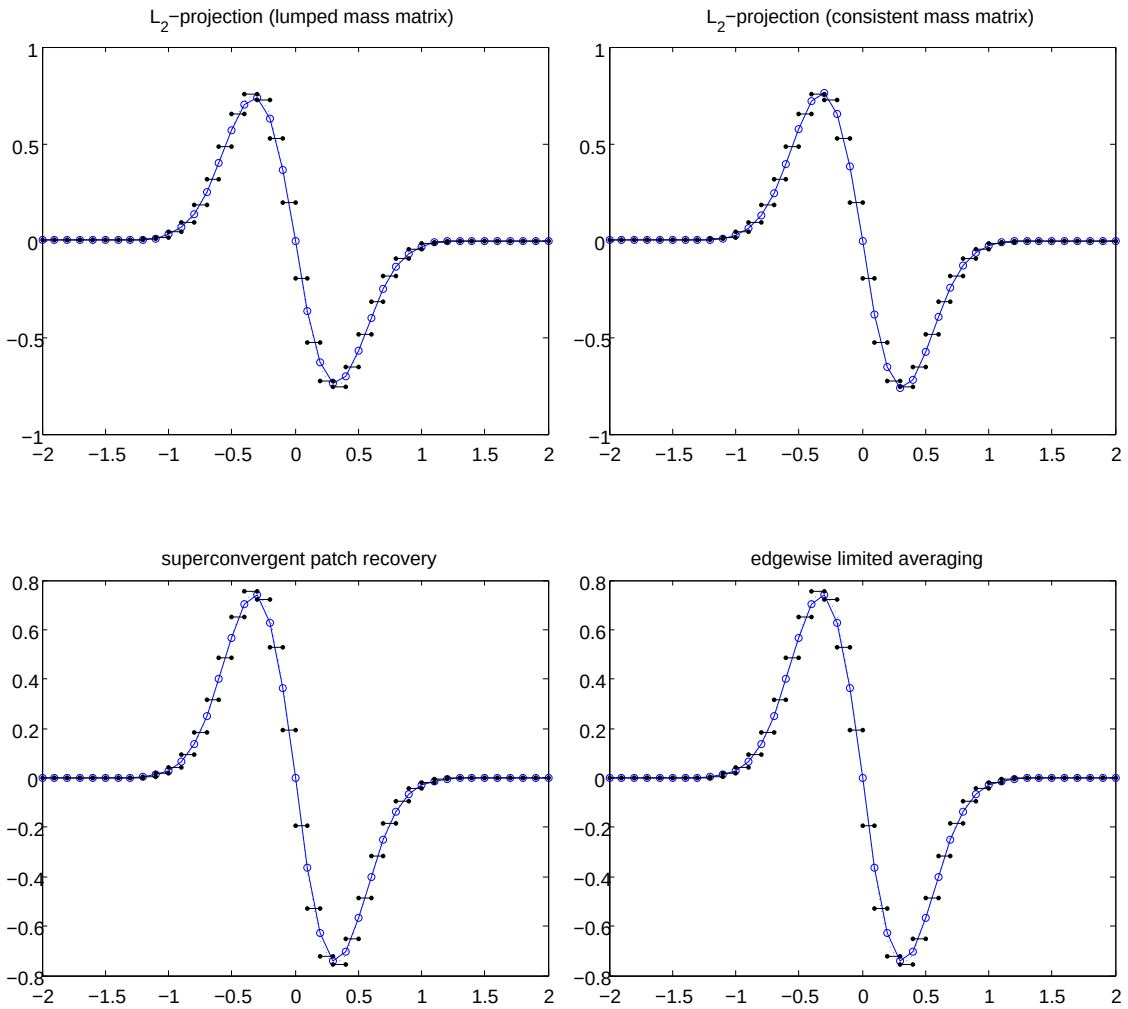


Fig. 4.6. TP1: Comparison of consistent and recovered gradient with 40 elements.

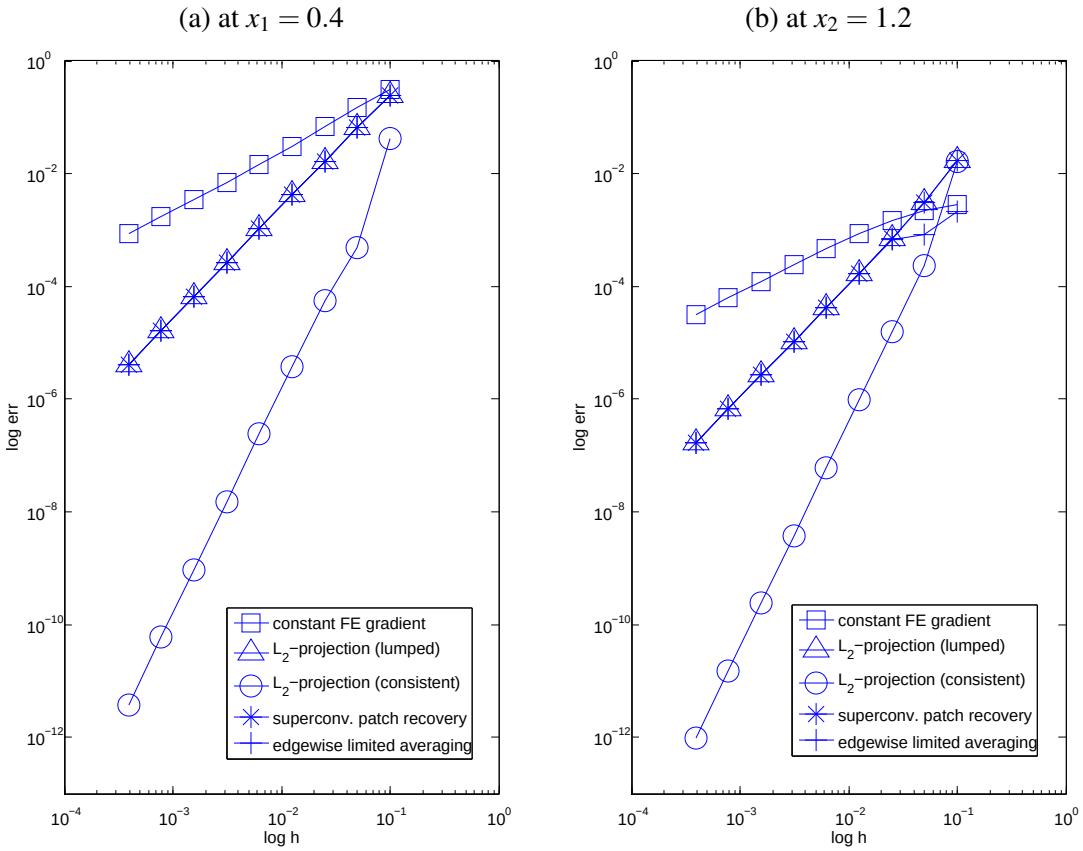


Fig. 4.7. TP1: Nodal convergence rate for recovered gradient.

The second test problem (TP2) deals with the steady convection-diffusion equation

$$a\frac{\partial u}{\partial x} - d\frac{\partial^2 u}{\partial x^2} = 0 \quad \text{in} \quad (0,1), \qquad u(0) = 1, \quad u(1) = 0 \tag{4.3.27}$$

which may give rise to the formation of a steep boundary layer if the convective term dominates, i.e. $a \gg d > 0$. The exact solution to the above model problem and its gradient are given by

$$u(x) = \frac{\exp(\gamma) - \exp(\gamma x)}{\exp(\gamma) - 1}, \qquad u'(x) = \frac{-\gamma \exp(\gamma x)}{\exp(\gamma) - 1} \tag{4.3.28}$$

where the parameter $\gamma = a/d$ denotes the ratio of convective and diffusive effects. For $a = 1$ and $d = 10^{-2}$, the gradient is nearly constant everywhere in the domain and abruptly tends to $-\gamma = -100$ near the right boundary. Nodal slopes have been computed on a sequence of successively refined equidistant meshes adopting the recovery techniques presented above. A representative assortment of smoothed gradient profiles is depicted in Figure 4.8, whereby 10 (left) and 40 (right) linear elements have been employed. All methods fail to predict the steep descent $u'(1) \approx -100$ at the boundary correctly. Moreover, the consistent L_2-projection scheme tends to produce overshoots and undershoots which also persist on the twice refined mesh.

The convergence rates of the pointwise gradient error measured at three exceptional nodes is illustrated in Figure 4.9. The absolute error observed for the consistent L_2-projection schemes differs by orders of magnitude as compared to the other recovery techniques. This can be attributed to the presence of spurious wiggles engendered on moderately coarse grids in the vicinity of steep fronts. On the other hand, the consistent L_2-projection scheme features the fastest error reduction

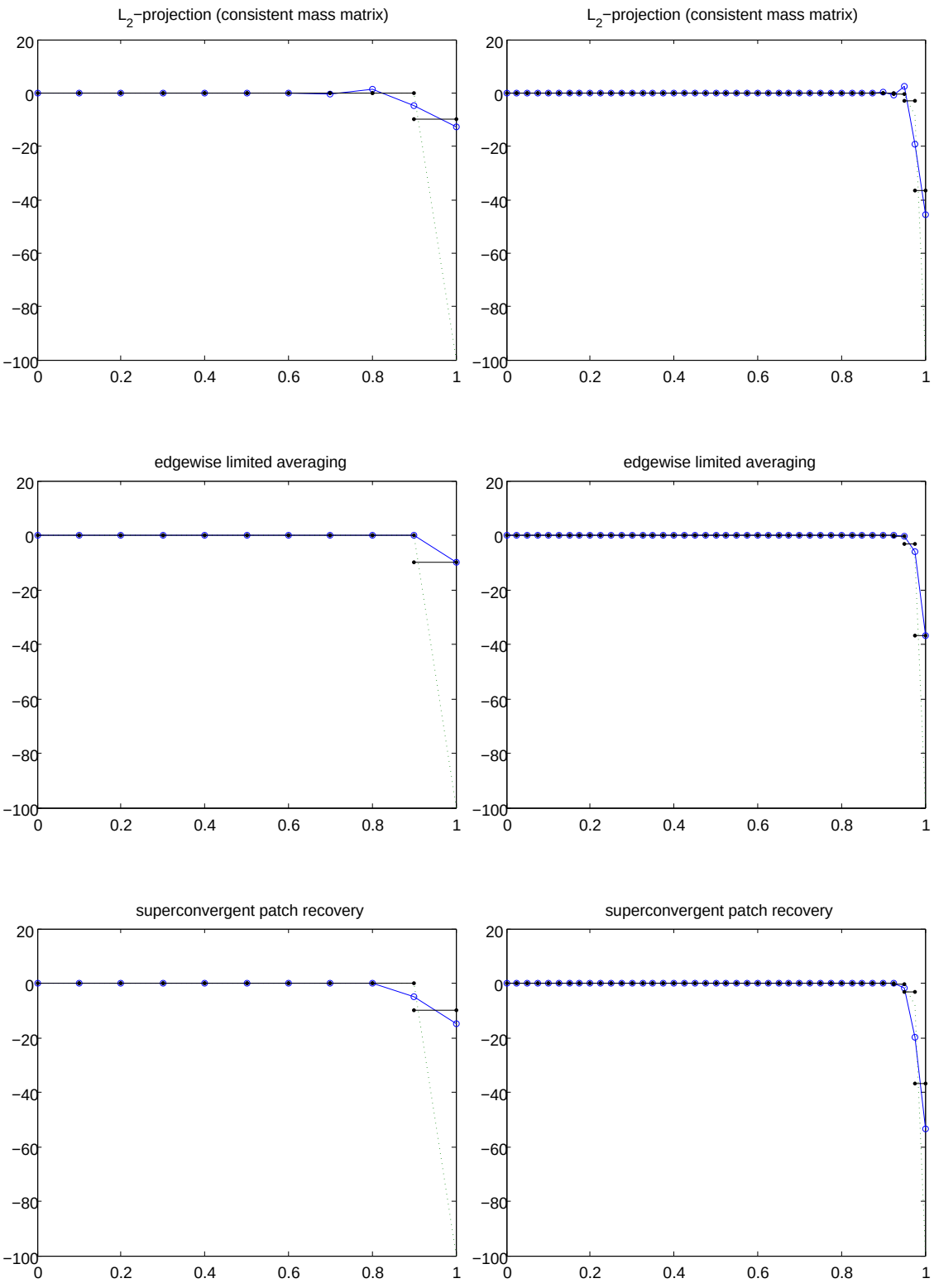


Fig. 4.8. TP2: Comparison of consistent and recovered gradient with 10 and 40 elements.

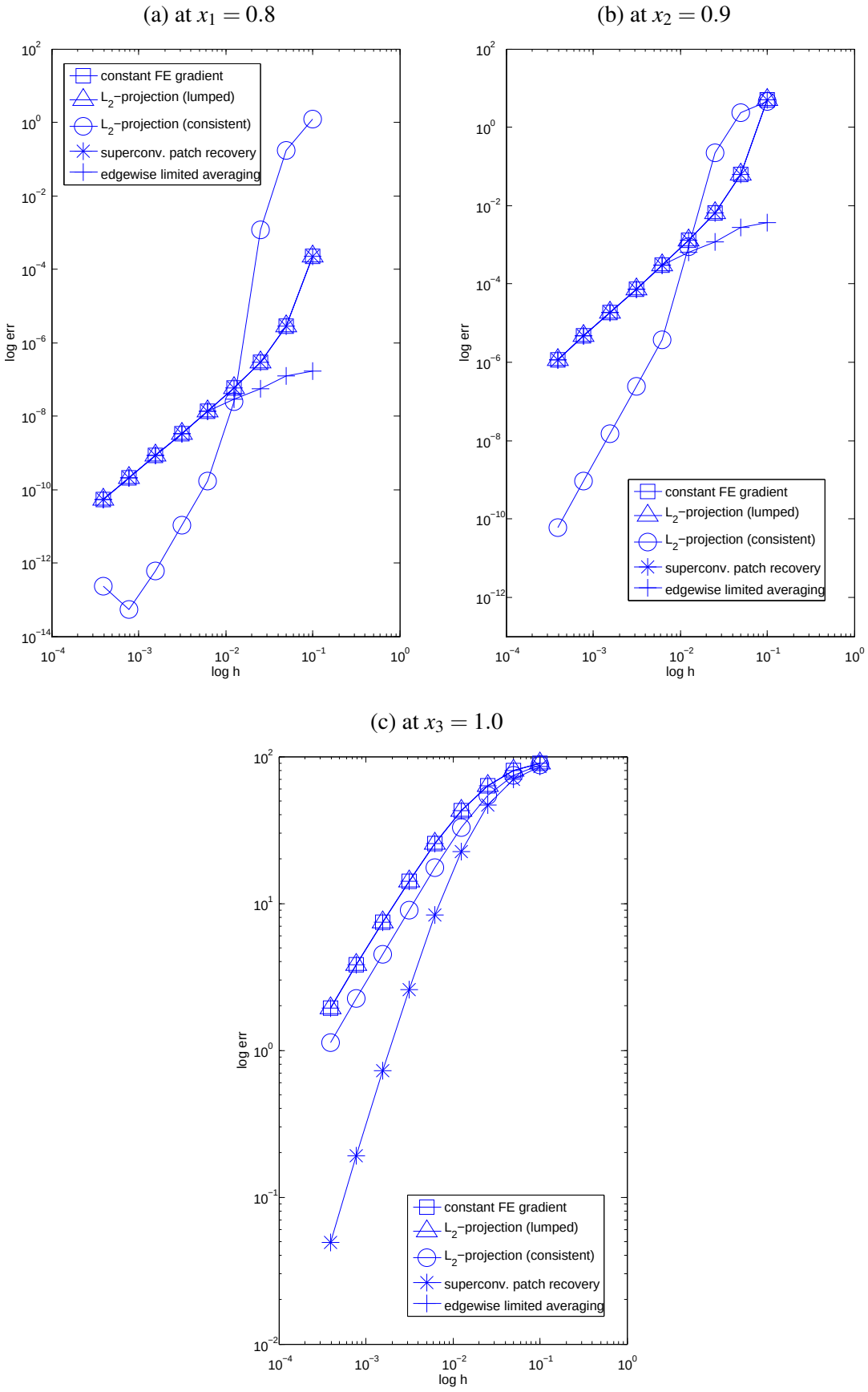


Fig. 4.9. TP2: Nodal convergence rate for recovered gradient.

rate once the grid spacing has become sufficiently small. It is worth mentioning that all other recovery procedures exhibit the same rate of convergence, whereby edgewise limited averaging yields superior gradient values in the interior (e.g., at points x_1 and x_2) if moderately coarse grids are employed. In contrast, superconvergence at the boundary, e.g., near $x_3 = 1.0$, is only observed for the SPR technique, whereas all other recovery schemes provide linear error reduction.

The preliminary results obtained for the two test problems deliver insight into the behavior of recovery-based error indicators applied to solution profiles with steep fronts. Variational recovery schemes are inexpensive and easy to implement, and moreover, the consistent L_2-projection method provides superior error reduction rates in many situations. On the other hand, spurious oscillations are likely to occur in the vicinity of steep gradients so that the predicted nodal slopes may be erroneous on coarse grids. This can be remedied by correcting the provisional gradient values with aid of the edgewise slope-limiting procedure (4.3.25). The results computed by edgewise limited averaging are promising especially on coarse grids, and hence, this approach represents an efficient alternative to the more elaborate recovery schemes such as the SPR technique.

4.4. Adaptation strategy

Error estimation techniques provide reasonable tools to verify the quality of the computed solution per se. In practice, they are frequently used to identify regions, where the grid should be further refined in order to improve the accuracy of the approximate solution. The advantage of dimensionless error indicators is as follows: Each element or vertex is given a value from a finite domain, e.g., $[0,1]$ for Löhner's difference indicator [161]. Hence, it suffices to define some tolerance C_{ref} and mark those elements for refinement for which the indicator exceeds the prescribed tolerance. In addition, the constant C_{crs} is used to identify patches of cells which can be safely re-coarsened provided the corresponding values of the indicator are less than the parameter C_{crs}.

Another simple adaptation strategy known as fixed fraction criterion involves the two parameters $C_{\text{ref}}, C_{\text{crs}} \in (0,100)$ such that $C_{\text{ref}} + C_{\text{crs}} < 100$. Let the elementwise error indicators be ordered according to their size. Those elements which correspond to the C_{ref} percent of the largest entries in the ordered sequence (say, the top 20%) are refined. In contrast, a portion of C_{crs} percent of elements with smallest error (say, the bottom 10%) are coarsened. Further variations on this approach, e.g., with dynamically changing parameters C_{ref} and C_{crs} are also possible [243].

In practice, the accuracy requirement may be given in terms of an overall percentage error

$$\zeta = \frac{\|e\|}{\|u\|} \tag{4.4.1}$$

where $\|\cdot\|$ denotes some norm suitable for the problem at hand. Since both the exact solution u and the error e are unknown, the available approximations are used instead so that the predicted values of the percentage error (4.4.1) can be computed as follows [274]

$$\zeta_h = \sqrt{\frac{\|\mathrm{e}_h\|^2}{\|\mathrm{u}_h\|^2 + \|\mathrm{e}_h\|^2}} \tag{4.4.2}$$

Here, u_h denotes the finite element solution and the global norm of the estimated error e_h can be decomposed into individual element contributions (cf. equation (4.3.5)), that is

$$\|\mathrm{e}_h\|^2 = \sum_{\Omega_k} \|\mathrm{e}_h\|^2_{\Omega_k} \tag{4.4.3}$$

In general, the user will specify the permissible value $\zeta_{\max}$ of the percentage error (4.4.1) and try to achieve $\zeta_h < \zeta_{\max}$ economically, that is, by refining the mesh locally. Assume that the error is

distributed equally between elements so that the permissible element error can be defined as [274]

$$\|\mathrm{e}_h\|^2 \approx N_\mathrm{E}\,\mathrm{e}_{\max}^2 \qquad \Rightarrow \qquad \mathrm{e}_{\max} = \zeta_{\max}\sqrt{\frac{\|\mathrm{u}_h\|^2 + \|\mathrm{e}_h\|^2}{N_\mathrm{E}}} \tag{4.4.4}$$

where N_E denotes the total number of elements. In order to identify elements which should be refined it suffices to check the ratio of predicted and permissible element errors:

$$\|\mathrm{e}_h\|_{\Omega_k} > \mathrm{e}_{\max} \qquad \Rightarrow \qquad \text{refine element } \Omega_k \tag{4.4.5}$$

A similar strategy may be employed to determine the set of element which can be coarsened. In particular, a lower bound $\zeta_{\min}$ for the percentage error is defined and cells are marked for coarsening if the estimated error is less than the smallest permissible element error:

$$\mathrm{e}_{\min} = \zeta_{\min}\sqrt{\frac{\|\mathrm{u}_h\|^2 + \|\mathrm{e}_h\|^2}{N_\mathrm{E}}}, \qquad \|\mathrm{e}_h\|_{\Omega_k} < \mathrm{e}_{\min} \qquad \Rightarrow \qquad \text{coarsen element } \Omega_k \tag{4.4.6}$$

Of course, there are many more strategies to select elements for refinement and coarsening based on a given error distribution per element. Let us mention some iterative error-balancing strategy which tries to equilibrate the element error such that [17]

$$\|\mathrm{e}_h\|_{\Omega_k} \approx \zeta_{\max}/N_\mathrm{E}, \qquad k = 1, 2, \ldots, N_\mathrm{E} \tag{4.4.7}$$

Since the total number of mesh cells N_E is only obtained at the end of the adaptation cycle this yields an implicit criterion to mark elements for refinement. Let the local errors be ordered in decreasing order. Starting from the largest value, i.e. $k = 1$ and $n = 0$, we check if

$$\|\mathrm{e}\|_{\Omega_k} \leq \frac{\zeta_{\max}}{N_\mathrm{E} + 3n} \tag{4.4.8}$$

and refine the element Ω_k otherwise. In this case, the counter n is increased by one to account for the new number of elements. Once the above condition is satisfied for some element, the process is stopped since all further elements exhibit smaller local errors.

4.5. Survey of mesh adaptation methods

The primary objective of mesh adaptation is to construct an underlying mesh which yields the numerical solution to a given problem with a desired accuracy at minimal computational costs. For pure *h*-adaptivity this coincides with the requirement of using the smallest possible number of degrees of freedom. If higher order finite elements are employed, e.g., in *p*- or *hp*-adaptivity, solution costs may also grow due to a denser connectivity pattern of the finite element matrix. With this observation in mind, mesh adaptation can be seen as an optimization problem for the underlying triangulation, whereby the choice of optimization criteria is problem dependent.

Equidistribution schemes try to minimize the displacement globally in a suitable energy norm so that the error is distributed uniformly in space which yields a finite element mesh with as few degrees of freedom as possible [12, 13]. This strategy is tailored to elliptic problems for which the domain of dependence and the region of influence coincide and occupy the entire domain. Since there is a mutual interaction between any two grid points reduction of the global error is preferable. From an engineering point of view, it frequently makes sense to bound the absolute error locally, e.g., for sub-domains or even at some single point. To illustrate this idea, consider the steel cables of a rope bridge. For the safe operation it is most important to guarantee that each rope bears the load applied to it under realistic weather conditions and rush-hour traffic.

The solution of convection dominated flows is characterized by the appearance of discontinuities and steep gradients. The location of these zones of high activity is not known *a priori* and may even change in the course of the simulation so that adaptive mesh refinement/re-coarsening is mandatory. The accuracy of the approximation is greatly improved by locally refining the mesh in the vicinity of 'interesting' features. At the same time, considerable savings in terms of memory usage and CPU time are obtained by coarsening the distribution of grid points in regions of uniform activity. In practical CFD application, an adaptive remeshing procedure for steady state flow computations is as follows: An initial mesh is chosen which is most unlikely optimal [168], since one does not know about the exact location of the regions of high activity *a priori*. As a rule of thumb, the initial mesh should be coarse enough to reduce the computational costs. At the same time, a 'good' point distribution is essential to capture all salient flow features and to reduce the number of adaptivity cycles for steady state computations [179]. The solution is marched to a quasi steady state and one or more adaptation criteria are adopted to create a new – and hopefully more suitable – mesh. The previous solution is transferred to the new grid and advanced towards steady state again. A global indicator is used to stop the iterative process of computing a converged solution and remeshing the grid. For time-dependent problems, the situation is slightly more complicated since 'interesting' flow features are likely to travel through the domain as time goes on. Of course, the grid can be adapted in each time step so as to resolve the current solution in an optimal way. However, this strategy is prohibitively expensive so that multiple time steps are performed on the same mesh before the grid is re-adapted to the solution. This demands for some foresight of the error estimator so as to refined the grid in regions, where interesting flow features will be generated in forthcoming time steps. In practice, error estimation techniques based on the current solution are employed to determine a set of elements which need to be refined and some protection layers of cells are included to account for the flow field of forthcoming time steps.

Even though this ad hoc approach lacks theoretical justification it works well in practice [161]. More sophisticated error estimation techniques for time-dependent flows may be based on a space-time splitting of the adjoint error representation [242]. As an alternative, the problem at hand can be considered in the space-time domain which amounts to computing the solution to all time levels simultaneously. As a result, standard error estimation techniques can be employed. This approach is very expensive for realistic configurations, and hence, not applicable to three-dimensional real-life applications. In what follows, we will consider different types of mesh adaptation in use.

Grid redistribution

Grid redistribution schemes (r-adaptation) [169] are quite easy to implement into an existing code since the grid topology remains unchanged. The basic idea is to align the fixed number of grid points to the flow field in an optimal sense without changing the connectivity. A good overview of various grid redistribution schemes is given in the survey articles by Eiseman [69] and Thompson [249, 250] and in the book by Carey [48]. The early attempts date back to Miller's moving finite element method [182], in which the new coordinates of the grid points are subject to some functional minimization process. Another redistribution method based on optimization approaches has been proposed by Jacquotte [112]. In the field of hypersonic flows, Gnoffo [81] utilized a spring analogy energy minimization in which the stiffness of the springs is proportional to the error indicator. His work was extended to parabolized Navier-Stokes solutions for simple configurations [98, 99], but in general, redistribution methods revealed a lack of robustness [97].

In addition, grid redistribution schemes can be based on variational principles at the expense of solving an additional partial differential evolution equation in each adaptation step. Recently, moving mesh methods have been applied successfully to hyperbolic conservation laws [247]. Notwithstanding the impressive results for two-dimensional Euler flow calculations by Tang [248], the 'background' mesh needs to be sufficiently fine so that enough cells can be concentrated in the

vicinity of shock waves. Many popular approaches allow to prescribe the volume of cells of the new grid but lack the facility to control their shapes, and hence, such techniques may lead to highly distorted triangulations. The use of grid redistribution schemes in complex and/or non-convex domains is more complicated and requires special care. Moreover, some schemes may completely fail if interior obstacles such as airfoils give rise to several boundary components.

It is now believed [169] that redistributing points is not flexible enough to be successfully applied to highly unstructured grids and complex geometries for general compressible flow problems since the elements tend to become extremely distorted. Even predicting the number of required grid points *a priori* is a non-trivial task. On the other hand, high-performance computations can thrive on the fact that the point connectivity remains unchanged throughout the whole simulation. If specialized (re)numbering strategies are employed to construct optimized hardware-oriented numerical schemes then mesh redistribution may be a viable candidate for grid adaptation. Additionally, grid smoothing techniques can be employed as post-processing to improve the grid quality after another mesh adaptation strategy has been applied [80]. Strictly speaking, they can also be regarded as grid redistribution method since the connectivity is preserved.

Grid regeneration

An alternative approach is grid regeneration (m-adaptation) which came along with the development of automatic grid generators. For a given solution the new distribution of grid points is determined with aid of error estimation techniques. Then local parts or even the entire computational domain are regenerated by means of advancing front algorithms [170, 203], Delaunay triangulations [40, 178, 260] and modified quadtree or octree techniques [237], respectively. A detailed description of grid regeneration schemes is given by Peraire et al. [202, 203] and Hassan et al. [209]. The grid points can be aligned with the flow field in an optimal sense without producing skewed elements. Local remeshing provides a reasonable tool for computing flow fields with moving bodies [209], for which pure grid redistribution would lead to unacceptably distorted cells. On the other hand, the high computational costs required by grid regeneration and (conservative) interpolation of the solution between two completely different grids does not pay off in the context of steady state problems for which a usable initial grid can be constructed quite efficiently. Furthermore, automatic grid generation of hexahedral meshes is still a matter of research so that m-adaptivity is commonly applied to triangular and tetrahedral grids in practice.

Grid enrichment

The approach we will follow in this thesis is based on enriching the computational grid. This can be either done by inserting new nodes into the grid and subdividing existing elements into new ones (h-adaptivity) or by enhancing the degree of basis functions (p-adaptivity). The latter strategy can be accomplished by means of higher-order polynomials [15], spectral functions [177] or by hierarchical shape functions [273]. As pointed out earlier, solutions to convection dominated flow problems are characterized by singularities which can be approximated with first order accuracy at most without violating the monotonicity constraint [83]. In light of the above, the high-resolution schemes presented in Chapter 3 can be regarded as rudimentary p-adaptation. Algebraic criteria are employed to detect regions in which the high-order discretization ($p \approx 2$) has to be gradually reduced to low-order ($p \approx 1$) by introducing artificial diffusion. Mesh enrichment by means of p-adaptivity ($p \gg 1$) is frequently employed in the context of discontinuous Galerkin finite element discretization which allow for an unproblematic treatment of different approximation orders.

Mesh refinement by means of h-enrichment has been widely used for fluid dynamic problems both with embedded grids aligned to the initial one [168] and unaligned grids [31]. In addition to the fact that conservation of physical quantities is naturally maintained, the classical h-refinement

procedure does not introduce numerical diffusion, since no interpolation except for the basis functions is required. In the context of aligned embedded mesh methods we have to distinguish between those which engender so-called 'hanging nodes' [236] and others which do not give rise to interface regions [168]. The latter strategy is sometimes referred to as conformal mesh refinement which results in regular grids [19, 27]. Following the method proposed by Rivara [215], for all triangles/tetrahedra that are marked for refinement the mid-point of the longest edge is connected to the opposite vertex. In a recursive algorithm, hanging nodes are connected in the same manner until global conformality is achieved, whereby the longest-edge refinement algorithm terminates in a finite number of steps [214]. Moreover, it has been shown that the smallest interior angle in a triangulation resulting from a repeated use of this algorithm is greater than or equal to half the smallest angle of the initial grid. In a series of publications [213, 216–218, 220], various improvements of this algorithm have been suggested and its properties such as linear run-time have been analyzed. Although an analogous strategy has been successfully applied in three space dimensions [219] some mathematical results which hold for triangles and give the theoretical justification for this approach are not available in 3D. Even more important, longest-side bisection can only be applied to triangles/tetrahedra and lacks a natural counterpart for quadrilaterals/hexahedra.

This severe restriction has been overcome by another bisection-type mesh refinement approach which was proposed by Bank et al. in an early paper [19]. The basic idea is to subdivide each element in two space dimensions into four self-similar sub-triangles and sub-quadrilaterals, respectively. This strategy is commonly known as *red*-refinement. Afterwards, global conformality of the triangulation is restored by introducing transition elements which are referred to as *green* elements. These auxiliary cells are recombined to their original macro element prior to performing further refinement so that the quality of the resulting grid does not deteriorate due to acute angles. Pattern-based mesh refinement can be directly extended to three space dimensions [163], whereby the quality of the resulting grids may slightly deteriorate. As an example consider a tetrahedron which is regularly refined into four tetrahedra and one interior octahedron. The latter needs to be further subdivided into four tetrahedra, whereby the new cells should be placed around the shortest inner diagonal so as to produce the fewest distortion in the refined grid [163]. Alternative refinement patterns have been considered by Mavriplis [180], Bieterman et al. [36], Leitner and Selberherr [152], Korotov and Křížek [124]. For simplicity, we will consider hierarchical mesh adaptation algorithms based on the red-green refinement strategy in two dimensions and address its generalization to three space dimensions in a forthcoming publication.

4.6. Hierarchical mesh adaptation

Mesh adaptivity is an essential component of modern simulation software since it allows to simulate real-life applications at reasonable costs. In light of the above, the design of an efficient adaptation strategy is a challenging task which requires a profound knowledge of the problem to be solved. Some approaches – like grid redistribution schemes – perform well for moderate grid deformation but they may fail completely in case of strong mesh anisotropies due to the development of steep fronts and/or shock waves. In this thesis, we present a mesh adaptation strategy which is guaranteed to produce a series of conformal triangulations that possess the same quality as the initial triangulation. In addition, this approach maintains the grid hierarchy to the largest extent which may be exploited in a geometric multigrid approach. This technique is largely inspired by the work by Hempel [102] who considered inviscid compressible flows discretized by explicit finite volume schemes. What makes the difference is the fact, that Hempel proposed a special purpose algorithm which is only applicable to triangles in two space dimensions. In this work, we utilize the same ideas and extend them to arbitrary – and possible mixed – triangulations. Despite the fact that only two-dimensional numerical examples are presented in this thesis the presented

strategy is readily applicable in three dimensions. Admittedly, the complexity of implementation is more demanding but in principal there are no difficulties from the theoretical point of view.

The design goal of our mesh adaptation procedure can be summarized as follows: *What can be refined in a single step, can also be re-coarsened in one subsequent step undoing the previous modifications* [102]. The importance of this paradigm is due to several reasons. First, it states that for every atomic refinement operation there exists an inverse re-coarsening procedure so that their consecutive application yields the identity which amounts to performing no mesh modification at all. As a result, any grid resulting from the application of several refinement and re-coarsening steps can equivalently be constructed from the same initial mesh by only making use of local refinement. It is worth mentioning, that both mesh refinement and re-coarsening have the same 'throughput' [101] which is an outstanding property among today's grid coarsening strategies. Most coarsening algorithms try to identify simply connected blocks of elements which can be collectively coarsened [101, 163]. A common drawback of such methods is the necessity to repeatedly apply mesh coarsening in order to restore the outcome of *one* refinement step.

The flow chart of algorithmic steps involved in the hierarchical mesh adaptation procedure is sketched in Figure 4.10. First, an error indicator/estimator is used to obtain an error distribution of the solution which was computed in the given mesh. This information is used to mark cells for refinement and identify patches of elements which can be re-coarsened to their original macro cell. The actual refinement and re-coarsening steps are performed one after the other.

1. Mark elements for refinement based on a given error distribution

$$\mathcal{M} : \mathcal{E}_m \to \mathbb{N}_0, \quad \forall \Omega_k \in \mathcal{E}_m, \quad \text{where} \quad \mathcal{T}_m = (\mathcal{E}_m, \mathcal{V}_m)$$

2. Lock vertices from removal and update element marker for re-coarsening

$$d : \mathcal{V}_m \to \mathbb{Z}, \quad \forall \mathrm{v}_i \in \mathcal{V}_m, \quad \text{where} \quad \mathcal{T}_m = (\mathcal{E}_m, \mathcal{V}_m)$$

3. Insert new vertices and perform local mesh refinement

$$g : \mathcal{V}'_{m+1} \to \mathbb{N}_0, \quad \forall \mathrm{v}_i \in \mathcal{V}'_{m+1}, \quad \text{where} \quad \mathcal{T}'_{m+1} = (\mathcal{E}'_{m+1}, \mathcal{V}'_{m+1})$$

4. Delete unneeded vertices and perform local mesh re-coarsening

$$\mathcal{T}'_{m+1} = (\mathcal{E}'_{m+1}, \mathcal{V}'_{m+1}) \to \mathcal{T}_{m+1} = (\mathcal{E}_{m+1}, \mathcal{V}_{m+1})$$

Fig. 4.10. Roadmap of the mesh adaptation procedure: $\mathcal{T}_m \to \mathcal{T}_{m+1}$.

The outline of this section is as follows: In Section 4.6.1 we take up an abstract position and introduce a hierarchical mesh refinement algorithm which is based on the red-green strategy suggested by Bank et al. [19]. A conformal re-coarsening strategy is developed in Section 4.6.2 extending the previous work by Hempel [102] to general triangulations. Section 4.6.3 is concerned with the theoretical properties of the presented mesh adaptation algorithm and gives a complete characterization of elements which is purely based on nodal information. Issues of practical implementation are addressed in Sections 4.6.5 and 4.6.6 which may be particularly worth reading for practitioners who want to include the suggested algorithms in an existing CFD/CSM code in an efficient way. A summary of the proposed mesh adaptation procedure is given in Section 4.6.7.

4.6.1. Conformal mesh refinement algorithm

Let $\mathcal{T} = \mathcal{T}(\mathcal{E},\mathcal{V})$ denote a triangulation of the computational domain $\Omega \subset \mathbb{R}^{N_D}$ subdivided into non-overlapping elements $\mathcal{E} = \{\Omega_k : k = 1,\ldots,N_E\}$ which are numbered globally by positive integers. The set of vertices which constitute the corner nodes of the elements is denoted by $\mathcal{V} = \{v_i : i = 1,\ldots,N_V\}$, whereby the quantities N_E and N_V stand for the total number of elements and nodes, respectively. In the context of the finite element method, a triangulation is said to be conformal if all pairs of elements (Ω_k,Ω_l) with $k \neq l$ satisfy one of the following conditions:

- the intersection of both elements $\Omega_k \cap \Omega_l$ is empty
- the elements Ω_k and Ω_l share a common vertex
- the elements Ω_k and Ω_l share an edge between two common vertices
- the elements Ω_k and Ω_l share a common face (only in 3D)

We will restrict ourselves to h-adaptivity and neglect post-processing techniques such as grid smoothing for the time being so that the position of vertices is fixed once they have been inserted into the mesh. Then the task of the adaptation procedure is to transform a given conformal triangulation $\mathcal{T}_m$ into a new grid $\mathcal{T}_{m+1}$ which is also conformal, whereby $\mathcal{T}_0 = \mathcal{T}(\mathcal{E}_0,\mathcal{V}_0)$ denotes the initial triangulation whose vertices $\mathcal{V}_0$ must be preserved in all refinement and coarsening steps.

Note that we carefully distinguish between the termini *operation* and *step*. A refinement operation is always applied to a single element or a localized group of neighboring cells whereas the term step denotes the ensemble of *all* operations of the same type. To illustrate this difference, consider the initial triangulation $\mathcal{T}_0$. In order to generate $\mathcal{T}_1$ by performing one refinement step $\mathcal{T}_0 \rightarrow \mathcal{T}_1$, a sequence of, say S, elemental refinement operations is applied, that is

$$\mathcal{T}_0 = \mathcal{T}_0^{(0)},\ldots,\mathcal{T}_0^{(s)} \rightarrow \mathcal{T}_0^{(s+1)},\ldots,\mathcal{T}_0^{(S)} = \mathcal{T}_1 \tag{4.6.1}$$

It is worth mentioning that an intermediate triangulation $\mathcal{T}_0^{(s)}$ need not be conformal provided that the final grid $\mathcal{T}_1$ is. This concept naturally carries over to arbitrary triangulations $\mathcal{T}_m \rightarrow \mathcal{T}_{m+1}$, whereby $S = S_r + S_c$ denotes the sum of element refinement and re-coarsening operations.

Hempel introduced the so-called *generation function* $g : \mathcal{V}_m \rightarrow \mathbb{N}_0$ which defines the 'date of birth' for each vertex of the triangulation [102]. By definition, all vertices of the initial triangulation belong to generation zero, and hence, they are considered unremovable throughout the simulation. If a new vertex $v_p \notin \mathcal{V}_m$ is inserted at the *midpoint* of the edge between the two nodes $v_i \in \mathcal{V}_m$ and $v_j \in \mathcal{V}_m$ then its date of birth is defined as the maximum of the generation numbers $g(v_i)$ and $g(v_j)$, increased by one. For general triangulations in arbitrary space dimensions Hempel's idea can be extended as follows: If a new vertex is placed in the *interior* of an element, say Ω_k, then the largest generation number of all corner nodes constituting cell Ω_k is adopted and increased by one for the new node. In three space dimensions, new vertices may be introduced at element faces shared by two adjacent elements. Then, the age of the new vertex is computed as the maximum generation number of the three/four corner nodes increased by one. In general, the generation function for the set of vertices is defined recursively as follows:

$$g(v_p) := \begin{cases} 0 & \text{if} \quad v_p \in \mathcal{V}_0 \\ \max\limits_{v_i \in \Gamma_{kl}} g(v_i) + 1 & \text{if} \quad v_p \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\ \max\limits_{v_i \in \partial\Omega_k} g(v_i) + 1 & \text{if} \quad v_p \in \Omega_k^\circ := \Omega_k \setminus \partial\Omega_k \end{cases} \tag{4.6.2}$$

where $\partial\Omega_k$ represents the boundary of element Ω_k. Let us remark that the second condition in the definition of the generation function (4.6.2) covers the cases of common edges in 2D and of common edges *and* common faces in three space dimensions, respectively.

As a consequence, it is trivial to prescribe a maximum number of admissible refinement levels which is particularly useful to control the computational costs of transient flow computations if dynamic mesh adaptation is employed. Of course, the youngest vertex of an element, that is, the one with the largest generation number, is always the node which was last inserted. At the same time, the vertex generation coincides with the number of refinement operations which were required to generate the element from the initial triangulation whose points all possess index zero. As we are about to see, the generation information suffices to determine the complete genealogy of any *element* present in an arbitrary triangulation produced by hierarchical red-green adaptation.

Insertion of elements and vertices

Let us review the classical red-green refinement algorithm introduced by Bank et al. [19] by considering an element Ω_k of the current triangulation $\mathcal{T}_m$ which is marked for refinement. Such elements are uniformly subdivided into four self-similar elements which are of the same type as the original cell, i.e. triangles or quadrilaterals in two space dimensions. This procedure which is commonly referred to as *red-refinement* may give rise to hanging nodes located in the midpoint of adjacent edges if no global refinement of the whole triangulation is performed.

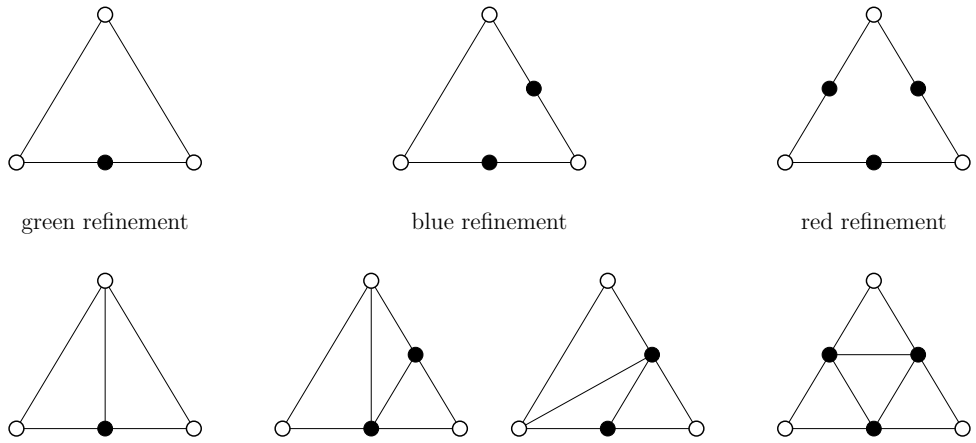


Fig. 4.11. Types of refinement for triangles in 2D.

For triangles appearing in two space dimensions, all possible cases are illustrated in Figure 4.11. If three hanging nodes are present in an element, it suffices to perform regular red-refinement for this cell. On the other hand, a single subdivided edge can be easily cured by connecting the new vertex in the edge midpoint to its opposite corner node. This so-called *green-refinement* resembles the longest-edge bisection strategy proposed by Rivara [215]. In general, the quality of the resulting triangulation may deteriorate but the smallest interior angle of any triangle can at most be halved. In order to avoid permanent deterioration due to successive subdivision of the same edge, a green element is recombined with its corresponding green neighbor prior to performing further refinement steps. This process is sketched in Figure 4.12, where the hatched triangle is marked for regular refinement. Suppose the *a priori* removal of green elements is neglected for the time being. Then the dotted edges need to be introduced which will produce four acute triangles. As a remedy, the affected green triangles are converted back to their macro cells which are subdivided into four similar elements. Finally, the red strategy is applied to the hatched

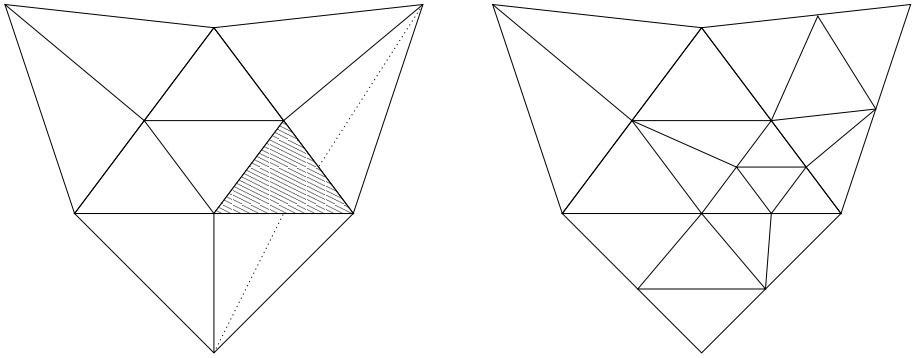


Fig. 4.12. Red-green refinement for triangles in 2D.

triangle and hanging nodes are removed by means of green refinement of adjacent cells.

The so-called *blue-refinement* is depicted in Figure 4.11 (middle). On the one hand, it is not obvious which of the two hanging nodes should be connected with the opposite vertex. On the other hand, interior angles may become very small so that distorted triangles can already result from one refinement operation. That is why blue refinement is avoided in the standard red-green algorithm [19]. Consequently, an additional vertex is introduced at the midpoint of the untouched edge and regular refinement following the red strategy is used to treat the three hanging nodes.

A similar technique can be applied to quadrilateral elements as depicted in Figure 4.13, whereby the different possibilities of blue refinement patterns have been omitted for obvious reasons. Let us emphasize the fact that this approach may turn a pure quadrilateral mesh into a mixed triangulation by virtue of the green transition elements. This may be a problem for discretization techniques which rely on uniform triangulations and/or are strongly based on geometric construction principles which are difficult or even impossible to generalize to mixed grids. Kuzmin and Turek [144, 145] demonstrated that high-resolution methods can be constructed on the basis of algebraic criteria, and hence, they are completely free of such drawbacks. Therefore, the algebraic flux correction schemes presented in Section 3.2 are convenient for the use of hybrid meshes.

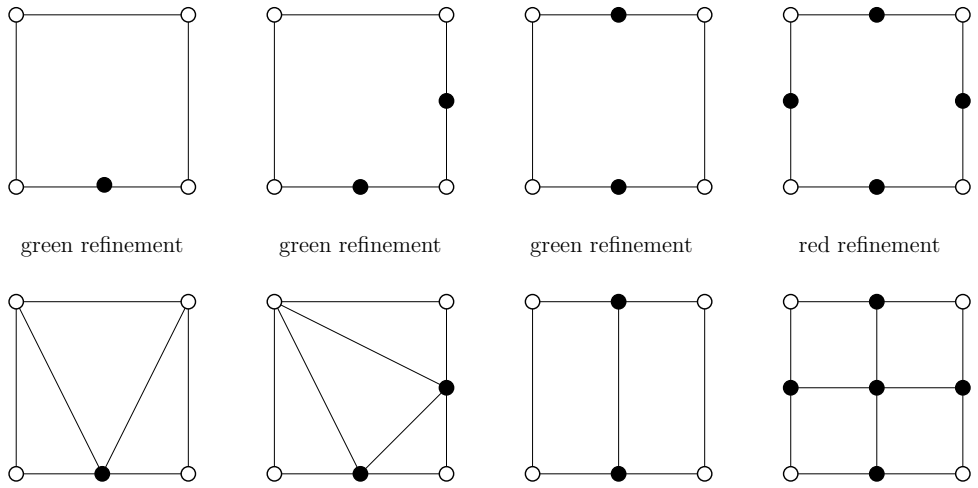


Fig. 4.13. Types of refinement for quadrilaterals in 2D.

4.6.2. Conformal mesh re-coarsening algorithm

In addition to local grid refinement required in regions where the solution error is unacceptably large, the mesh should be coarsened in those parts of the domain with errors smaller than average. As an example, consider a straight single shock wave which travels through the domain as time evolves. The locally adapted mesh should 'follow' the shock wave so as to provide a sufficiently high resolution of the discontinuity. At the same time, as few elements as possible should be used to reduce the computational costs. This is typically accomplished by removing some vertices and elements whose contribution to the error is negligible. Various techniques applicable to unstructured meshes can be found in the literature. The main difficulty is to identify sets of elements that can be agglomerated so as to remove some vertices and/or cells from the triangulation.

This can be done for instance by performing successive edge-swapping [91] until the vertex to be deleted is surrounded by exactly three triangles as illustrated in Figure 4.14. In three space dimensions, the node needs to be isolated in an analog fashion so that the four adjacent elements can be combined to one tetrahedron. If the node to be removed is located at the boundary, an artificial element is introduced so as to virtually 'move' the vertex into the interior. Afterwards the standard procedure can be applied to eliminate the marked vertices. However, edge-swapping is tailored to triangular and tetrahedral elements and is not applicable to quadrilaterals and hexahedra.

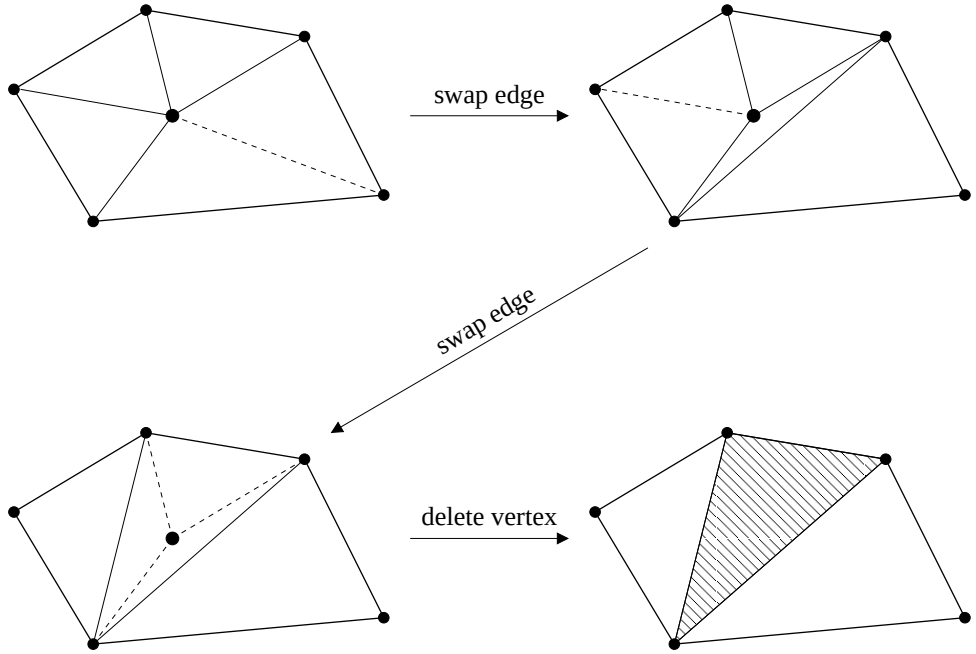


Fig. 4.14. Deletion of an interior vertex.

As an alternative, a family of admissible *blocking patterns* [163] can be defined *a priori*. When it comes to the coarsening of a patch of elements, one tries to recognize some of these patterns for which suitable algorithms have been implemented in the CFD code. In particular, the topic of pattern recognition has been a field of active research for decades [2, 29, 31, 60, 161, 179]. As another alternative, one can remove all cells in an element patch marked for deletion and perform local remeshing in this region of the domain. In any case, nodal connectivity may change completely from one grid to the other so that no hierarchy of grids is available.

Deletion of elements and vertices

A hierarchical approach for removing elements arises naturally in the context of red-green adaptivity. The adaptation process starts from those cells which are marked for refinement due to accuracy reasons and performs regular subdivision (red refinement) into four triangles/quadrilaterals in 2D. The green rule is applied *afterwards* in order to restore global conformality. Consequently, the deletion process must be initiated by patches of regularly refined cells whose fine resolution is no longer required so that they can be converted back into the original macro element. All green cells which are no longer needed to absorb 'hanging nodes' are eliminated in a second step.

For practical applications, an efficient algorithm to delete vertices and remove unneeded elements is crucial for the simulation of transient flows. Hempel [102] devised an elegant re-coarsening algorithm applicable to triangular grids in two space dimensions. In this thesis, we extend his work to arbitrary triangulations and present a more general algorithm for hierarchical mesh re-coarsening. Let us introduce the deletion indicator function $d : \mathcal{V}_m \to \mathbb{Z}$ whose absolute values coincide with those of the nodal generation function $g : \mathcal{V}_m \to \mathbb{N}_0$ defined in (4.6.2). Following the strategy by Hempel [102], the sign of function d is used to select vertices for removal:

- Node $\mathrm{v}_p \in \mathcal{V}_m$ can be removed from the triangulation $\mathcal{T}_m$ if $d(\mathrm{v}_p) > 0$;
- otherwise node $\mathrm{v}_p \in \mathcal{V}_m$ is 'locked' and must not be deleted if $d(\mathrm{v}_p) \leq 0$.

Of course, all vertices of the initial mesh $\mathcal{T}_0$ belong to generation zero, and therefore, they are unconditionally locked by construction. All other nodes are potential candidates for removal so that the deletion indicator is initialized as $d(\mathrm{v}_p) := g(\mathrm{v}_p)$ at the beginning of each re-coarsening step. For triangular meshes, Hempel [102] presented a set of rules which are applied to exclude vertices from the list of nodes that will be deleted. This so-called locking algorithm is divided into two parts, whereby the second step requires the re-investigation of adjacent elements once a node has been locked. Note that vertices cannot be unlocked by the re-coarsening procedure so that an infinite toggling between the two states – 'free' and 'locked' – cannot occur. In order to extend Hempel's approach [102] to arbitrary triangulations we established additional rules and reformulated some node-locking criteria. To our belief, this also improves the clarity of the basic principles of the re-coarsening strategy. The resulting algorithm is depicted in Figure 4.15.

Following the road map depicted in Figure 4.10, the node-locking algorithm is applied *after* elements have been marked for refinement based on an error indicator/estimator and *before* the actual refinement takes place. In the first loop, the term edges refers to the three and four physical edges of triangular and quadrilateral elements, respectively. They are not to be confused with the edges of the sparsity pattern which have been repeatedly mentioned in Chapter 3. All vertices which must unconditionally remain in the triangulation $\mathcal{T}_m$ are locked, once rule (R1) has been applied. Note that a vertex, say $\mathrm{v}_i \in \mathcal{V}_m$, can only be eliminated if it has been engendered by the most recent refinement operation. In other words, there exists an edge ij such that node v_j also belongs to the previous triangulation $\mathcal{T}_{m-1}$, and thus, it has smaller generation number, i.e. $g(\mathrm{v}_j) < g(\mathrm{v}_i)$. If both nodes feature the same age, then they have been introduced due to the same refinement operation, and thus, they are potential candidates for removal. Strictly speaking, condition (4.6.3) ensures that all vertices which belong to some macro element(s) are excluded from the provisional list of nodes that can be deleted from the current triangulation $\mathcal{T}_m$.

Rules (R2) and (R3) are responsible to transform the values of the cellwise error indicator/estimator into re-coarsening patterns applicable to groups of elements. Obviously, pending element subdivision conflicts with the removal of associated nodes. This is why all corners vertices of an element, say Ω_k, are locked by rule (R2) if the cell is marked for refinement. As stated above, the removal process starts from patches of red elements which are located in regions, where fine grid resolution is no longer required. In particular, rule (R3) ensures that elements which are neither marked for further refinement nor accepted for re-coarsening are kept 'as is'.

In a loop over physical edges ij:

(R1) Check if $g(\mathrm{v}_i) \neq g(\mathrm{v}_j)$ and lock the vertex with smaller generation number

$$d(\mathrm{v}_l) := -|d(\mathrm{v}_l)|, \quad l \in \{i, j\} \quad \text{such that} \quad g(\mathrm{v}_l) < g(\mathrm{v}_k), \quad k = i + j - l \tag{4.6.3}$$

In a loop over nodes:

(R2) Check if node v_i belongs to an element which is marked for refinement, then lock that vertex, i.e. $d(\mathrm{v}_i) := -|d(\mathrm{v}_i)$.

(R3) Check if node v_i belongs to a red element which is not marked for re-coarsening, then also lock that vertex.

In a (recursive) loop over elements (in 2D):

(R4) Check if two vertices of an inner red triangle are locked, then lock the third node.

(R5) Check if the 'interior' node of a patch ω of four red quadrilateral elements is locked or if more then six vertices associated with that group of elements are locked, then lock all nodes comprised in the patch ω.

Fig. 4.15. Locking algorithm to prevent vertices from removal.

In a practical implementation, all criteria (R1)–(R3) may be applied in a single loop over the elements, whereby condition (R1) can be ignored if (R2) or (R3) is already satisfied. It is worth mentioning that the rules (R1)–(R3) can be utilized without modification in three dimensions so that the first part of the locking algorithm can be carried out in linear time, i.e. $\mathcal{O}(N_{\mathrm{E}})$. An additional post-processing step is required to prevent the creation of blue elements. In this work, we present sufficient conditions for mixed triangulations in two space dimensions. The extension to 3D follows essentially the same strategy but it has not been implemented by the author so far.

Let us reconsider the possible refinement patterns for triangles illustrated in Figure 4.11. If two vertices of the inner triangle are locked the situation corresponds to the blue strategy (middle) which was avoided intentionally during refinement. In light of the above, rule (R4) is mandatory to prevent the generation of blue triangles, whereas condition (R5) ensures that re-coarsening of quadrilateral elements is restricted to the admissible patterns depicted in Figure 4.13.

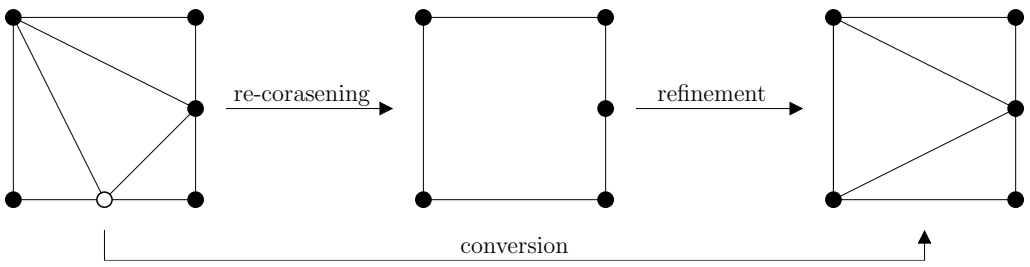


Fig. 4.16. Conversion of four triangles into three triangles.

Obviously, the locking of node v_i, engendered by some element, say Ω_k, requires the reinvestigation of all cells surrounding v_i. Consequently, the last part of the locking algorithm needs to be applied recursively until the deletion indicator d remains unchanged for all nodes. Once the list of 'free' vertices (i.e., $d(\mathrm{v}_i) > 0$) is assembled, the corresponding nodes are removed from the triangulation $\mathcal{T}_m$ and the neighboring cells are 'combined' to their original macro elements. In some situations, this process may lead to hanging nodes which need to be eliminated by appropriate green refinement. As an example, consider a group of four green triangles which are marked for re-coarsening, whereby one additional vertex at the edge midpoint is locked as illustrated in Figure 4.16. In this situation, the four cells are combined to their original macro quadrilateral which is refined following the green rule afterwards. In a practical implementation, these two successive re-coarsening/refinement operations can be carried out simultaneously in a simple conversion.

4.6.3. Formalized red-green adaptivity

At first glance, the task of finding appropriate patches of elements and converting them into coarser cells sounds complicated and quite time consuming. In Section 4.6.6 we will present a smart implementation of our hierarchical mesh adaptation procedure which does not need additional space and requires only linear time. It is worthwhile to analyze some basic properties of the suggested strategies and think about how they can be turned into an efficient algorithm.

For our purpose it makes sense to introduce a formalized naming convention for all red and green refinement/re-coarsening operations which allows for an unambiguous addressing.

Definition 4.6.1. An element subdivision or recombination will be referred to as $n_1E_1 \rhd n_2E_2$ operation, whereby $n_1, n_2 \in \{1,2,3,4\}$ and $E_1, E_2 \in \{\text{"Q(uadrilateral)"}, \text{"T(riangle)"}\}$.

According to this convention, the green and red refinement operations depicted in Figure 4.11 are termed $1T \rhd 2T$ and $1T \rhd 4T$ refinement, respectively. Furthermore, the grid modifications presented in Figure 4.13 are called $1Q \rhd 3T$, $1Q \rhd 4T$, $1Q \rhd 2Q$ and $1Q \rhd 4Q$ refinement operations (from left to right). In any case, $n_1 = 1$ and the subdivision of cells implies $n_2 > n_1$.

For each refinement operation there exists an atomic re-coarsening procedure which can be interpreted as the corresponding inverse operation, e.g., $4Q \rhd 1Q$ or $3T \rhd 1Q$ re-coarsening. It is easy to see that $n_2 = 1$ and $n_1 > n_2$ for any kind of element re-coarsening operation. In particular, all grid modifications required to generate a refined triangulation $\mathcal{T}_{m+1}$ starting from $\mathcal{T}_m$ can be 'undone' in a single step by using the corresponding inverse operations. Hence, a mesh $\mathcal{T}_m$ that results from $\mathcal{T}_0$ by application of m refinement *and* re-coarsening steps can be constructed by purely refining the initial triangulation $m' \leq m$ times. As a result, both grids are topologically equivalent, i.e. $\mathcal{T}_m \sim \mathcal{T}_{m'}$, whereby the actual numbering of vertices/elements may be different. This observation is useful in several respects. On the one hand, it simplifies the theoretical analysis which can be based on triangulations resulting from pure refinement. On the other hand, it allows to easily validate the correct interplay of mesh refinement and re-coarsening procedures in a concrete implementation. For instance, a green element is only introduced by the refinement procedure to 'absorb' a hanging node engendered by the regular subdivision (red refinement) of an adjacent cell. Hence, a vertex must not be completely surrounded by green elements.

Last but not least, let us introduce some shorthand notation for so-called conversions. Each element subdivision can be considered as an atomic refinement operation for which there exists a unique inverse, also atomic re-coarsening procedure. Of course, it is possible to combine a group of four triangles into the corresponding macro quadrilateral and perform green refinement into three triangles afterwards. This bi-atomic operation eliminates one of the two nodes recently inserted at the edge midpoints as depicted in Figure 4.16. For simplicity, we will refer to this composition of two operations as $4T \rhd 3T$ conversion which means that a $1Q \rhd 3T$ refinement of the macro element is performed *after* the application of a $4T \rhd 1Q$ re-coarsening.

In general, mesh refinement is initiated by those cells which exceed some prescribed error tolerance. The regular subdivision of cells may spread across several element layers until it stops either at the boundary or by application of a particular green refinement rule. An important design principle is that *a green element cannot appear out of nowhere but it has to be necessitated by some red refinement operation applied to one of its edge or face neighbors.* This property of the red-green strategy ensures that vertices are introduced at edge midpoints, if at least one adjacent cell has been refined regularly. In other words, a recently created node cannot belong to several green elements which result from different refinement *operations*. Moreover, the locking algorithm presented in Figure 4.15 complies with that rule which means that repeated re-coarsening does not lead to 'isolated' vertices which are surrounded by only green elements.

4.6.4. Implicit mesh genealogy

In many adaptive flow solvers, mesh adaptation is accomplished with aid of so-called *genealogy trees* which store the relation between all cells. In essence, the root consists of all elements present in the initial triangulation $\mathcal{T}_0$. Whenever a cell, say Ω_k, is subdivided into n_2 parts, the appropriate number of *leaves* are inserted into the tree as children below the corresponding parent item. Afterwards, all parents are deactivated, that is, they are virtually removed from the list of elements. Note that these items physically remain in the genealogy tree, and thus, additional storage is required for all non-active cells. By construction, the depth of the tree cannot exceed the maximum level of admissible refinement operations $R_{\max}$ so that the total memory is bounded by [56]

$$|\mathcal{E}_0| \sum_{r=0}^{R_{\max}} Q^r = |\mathcal{E}_0| \frac{Q^{R_{\max}+1} - 1}{Q - 1} \tag{4.6.4}$$

where $|\mathcal{E}_0|$ denotes the number of elements of the initial triangulation $\mathcal{T}_0$ and the regular subdivision of a single cell typically gives rise to $Q = 4$ and $Q = 8$ new elements in two and three space dimensions, respectively. The elements of the current triangulation $\mathcal{T}_m$ are given by all leaves of the genealogy tree. The removal of elements is performed in reverse order. For all leaves which are marked for coarsening, the corresponding parent nodes are re-activated and all of their children are destroyed. In a practical implementation, an additional array may be introduced to allow for a direct 'look-up' of leaves, i.e. active elements. Moreover, each child may keep a link to its parent to avoid expensive search operations which typically require logarithmic time on average.

In contrast, the nodal generation function (4.6.2) is defined for all vertices present in the current triangulation so that no unnecessary information needs to be stored in order to revert to some coarser mesh. In other words, the number of vertices $N_{\mathrm{V}} = \max_{m=1,\dots,M} |\mathcal{V}_M|$ present in the 'finest' computational mesh yields an upper bound for the total space required to store the complete genealogy of a sequence of triangulations $\{\mathcal{T}_m\}_{m=0}^{M}$ which is generated from an initial grid $\mathcal{T}_0$ by application of a finite number of refinement and/or re-coarsening steps. Remarkably, each element, say $\Omega_k \in \mathcal{E}_m$ of a particular triangulation $\mathcal{T}_m$ can be 'characterized' by the generation numbers of the corner vertices. Moreover, this information suffices to determine the relation of Ω_k to its macro element. Hempel [102] demonstrated that four different cases have to be considered to identify triangles resulting from red-green adaptation. We have generalized this strategy to mixed triangulations in two dimensions and present a viable characterization in Theorem 4.6.1. It is complete in the sense that each cell can be uniquely identified so that no ambiguous element constellations can occur. The proof of this proposition is quite technical, and therefore, it is postponed to Appendix D. As we are about to see in Section 4.6.6, the tedious identification of elements can be simplified considerably by defining element states *a priori* which can be checked in constant time. The extension of Theorem 4.6.1 to tetrahedral and/or hexahedral meshes in three space dimensions is still outstanding but no conceptual difficulties are to be expected.

Theorem 4.6.1. (Characterization of elements in 2D) Let $\mathcal{T}_m = (\mathcal{E}_m, \mathcal{V}_m)$, $m = 0, 1, \ldots$ be an arbitrary conformal triangulation that is constructed from $\mathcal{T}_0$ by m refinement/re-coarsening steps.

(a) If all nodes of a triangle/quadrilateral Ω_k have generation number zero, then $\Omega_k \in \mathcal{E}_0$, that is, the element belongs to the initial triangulation $\mathcal{T}_0$.

(b) If three nodes of a quadrilateral Ω_k have the same generation number, then Ω_k is a red element which results from $1Q \triangleright 4Q$ refinement.

(c) If all nodes of a triangle Ω_k have the same generation number, then Ω_k is an inner red element resulting from $1T \triangleright 3T$ refinement.

(d) If exactly two consecutive nodes of a quadrilateral Ω_k have largest (positive) generation number (in the cell), then the connecting edge was introduced by a green $1Q \triangleright 2Q$ refinement and the adjacent element Ω_l constitutes the corresponding green neighbor.

(e) Let two nodes of a triangle Ω_k have largest generation number (in the cell).

  (e.1) If the neighbor along the edge connecting these two vertices is an inner red triangle, then Ω_k is an outer red triangle resulting from $1T \triangleright 3T$ refinement. The third vertex which has smaller generation number coincides with one node of the macro element.

  (e.2) Otherwise (i.e. there is no adjacent inner red triangle), element Ω_k is a green triangle which results from $1Q \triangleright 4T$ refinement of a quadrilateral into four triangles.

(f) Let one node, say v_i, of a triangle Ω_k possess largest generation number (in the cell).

  (f.1) If there exists a neighboring triangle Ω_l with two vertices with largest generation number, then Ω_k is one of the two outer green triangles resulting from $1Q \triangleright 4T$ refinement.

  (f.2) If there exists exactly *one* triangle Ω_l meeting at the common node v_i which has largest generation number (in cell Ω_l), then both elements result from $1T \triangleright 2T$ refinement of the original macro triangle $\Omega_k \cup \Omega_l$ into two green triangles.

  (f.3) If there exist exactly *two* triangles Ω_l meeting at the common node v_i which has largest generation number (in all cells), then the three elements result from $1Q \triangleright 3T$ refinement of the original macro quadrilateral into three green triangles.

Proof. See Appendix D. □

4.6.5. Local two-level ordering of vertices

The complete characterization of elements presented in Theorem 4.6.1 is theoretically attractive, but in practice, it may be cumbersome to repeatedly check multiple adjacent cells in order to determine the state of a *single* element. As a remedy, we resort to the two-level ordering strategy suggested by Turek [254] for multigrid algorithms and extend his approach to locally refined meshes. The basic idea is to define some numbering strategy for node insertion/element subdivision *a priori* so that some 'knowledge' about adjacent elements is available implicitly.

As an example, consider a single quadrilateral in two space dimensions which is subdivided into four regular cells as depicted in Figure 4.17. In the two-level ordering procedure, the numbers of existing vertices are preserved and new nodes are numbered in 'ascending' order [254]. The global element numbers are indicated by italics, and again, some predefined ordering is applied. Asterisks are used to indicated the starting position of the local node numbering within each element. For example, element $\Omega_2 \in \mathcal{T}_1$ is spanned by nodes $\mathrm{v}_3 \in \mathcal{V}_0$ and $\mathrm{v}_7, \mathrm{v}_9, \mathrm{v}_6 \in \mathcal{V}_1 \setminus \mathcal{V}_0$. In practice, regular subdivision is applied repeatedly so as to obtain the computational grid $\mathcal{T}_m$.

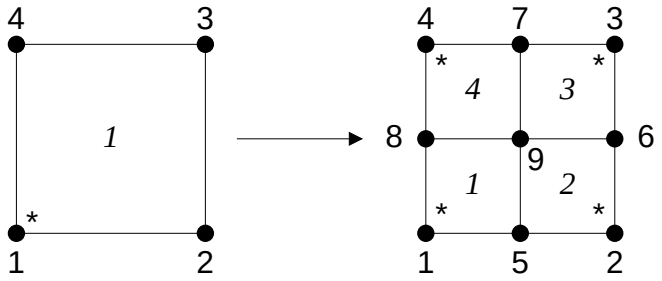


Fig. 4.17. Two-level ordering of vertices for regular refinement.

By construction, nodes of the same generation are grouped contiguously so that vertex coordinates for all multigrid levels can be stored in a single vector and addressed as follows: All vertices $\mathrm{v}_i \in \mathcal{V}_0$ of the initial coarse grid $\mathcal{T}_0$ are numbered from $i = 1,\ldots,|\mathcal{V}_0|$. Nodes $\mathrm{v}_i \in \mathcal{V}_1 \setminus \mathcal{V}_0$ which have been introduced on the next finer level reside in positions $i = |\mathcal{V}_0|+1,\ldots,|\mathcal{V}_1|$ and so forth.

In what follows, we extend the two-level ordering strategy [254] to meshes which exhibit locally refined subregions. Of course, the strict separation of node generations into contiguous memory blocks cannot be maintained if local mesh refinement and re-coarsening is performed. However, all nodes of the initial triangulation are stored en bloc, and they are numbered in compliance with the two-level ordering strategy so that standard multigrid procedures can be adopted.

In this thesis, we dwell on the two-dimensional triangulation $\mathcal{T}_m = (\mathcal{E}_m, \mathcal{V}_m)$ but the generalization to higher space dimensions is straightforward. All vertices $\mathrm{v}_i \in \mathcal{V}_m$ are globally numbered according to $i = 1,\ldots,N_\mathrm{V}$, whereas elements $\Omega_k \in \mathcal{E}_m$ are addressed by $k = 1,\ldots,N_\mathrm{E}$. In addition, there exists a local numbering of nodes per element, $iloc_1, iloc_2, \ldots, iloc_{N_\mathrm{VE}}$, whereby a counterclockwise ordering is adopted without loss of generality as indicated in Figure 4.17. This enables us to refer to the edge succeeding the vertex $iloc_l$ as the l^{th} edge of element Ω_k and the cell adjacent to this edge is termed the l^{th} element neighbor. Note that in three dimensions, the local numbering of nodes is not unique. Moreover, the number of corner nodes does not coincide with the number of faces and edges so that local edge-/face-numbering strategies need to be defined explicitly.

Let us return to red-green adaptation in two space dimensions and consider a triangle Ω_k which is marked for $1T \rhd 4T$ refinement. Starting with the first local edge, three new vertices are inserted at the midpoints of edges. They are assigned the global numbers $N_\mathrm{V} + l$, where N_V stands for the total number of vertices present in the triangulation just *before* the subdivision of element Ω_k and $l = 1,2,3$ denotes the local edge numbers. The inner red triangle preserves the number of the macro element Ω_k by definition, whereas all three outer triangles are labeled as depicted in Figure 4.19 (bottom). Without loss of generality, the local numbering within each outer triangle starts at the oldest vertex, that is, the unique vertex which is inherited by the original macro element. Consequently, the inner red triangle can always be reached from an outer red triangle 'via' the second edge. The local numbering of the interior element can be chosen arbitrarily as long as the same strategy is used consistently in all adaptation steps.

The application of local two-level ordering employed for green $1T \rhd 2T$ refinement is illustrated in Figure 4.19 (top). It makes no difference whether we start from element Ω_k or $\Omega_{N_\mathrm{E}+1}$, the corresponding green neighbor can always be reached 'via' the second edge. In general, there is some flexibility in choosing the local ordering strategy as long as a set of concrete numbering rules is defined *a priori* and strictly applied in each refinement and/or re-coarsening operation. For quadrilateral elements, a viable local two-level numbering for all possible red and green refinement patterns is presented in Figure 4.18. In both Figures 4.18 and 4.19, the encircled numbers denote so-called element states which will be defined in Section 4.6.6.

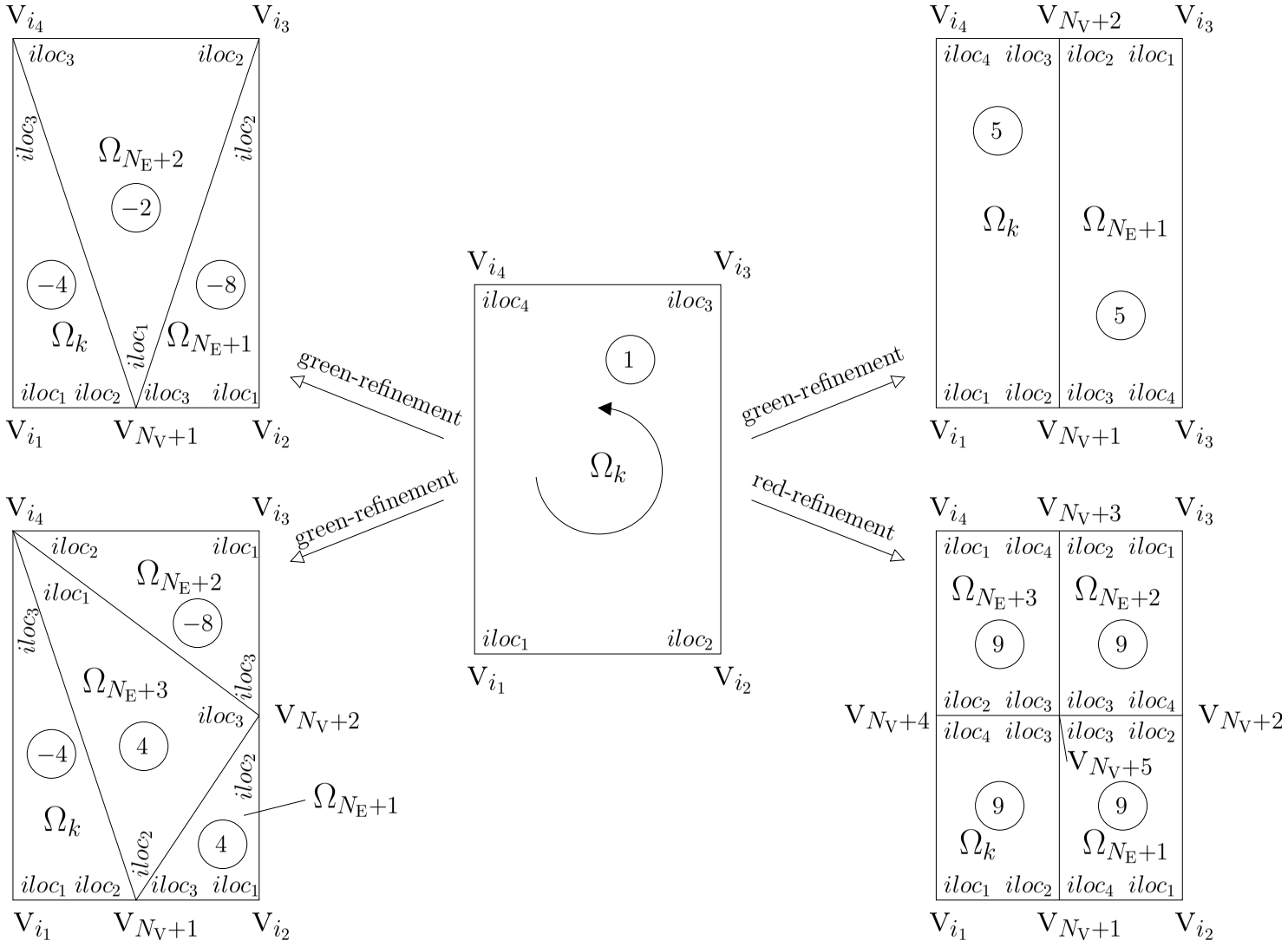


Fig. 4.18. Local two-level ordering of vertices for quadrilaterals in 2D.

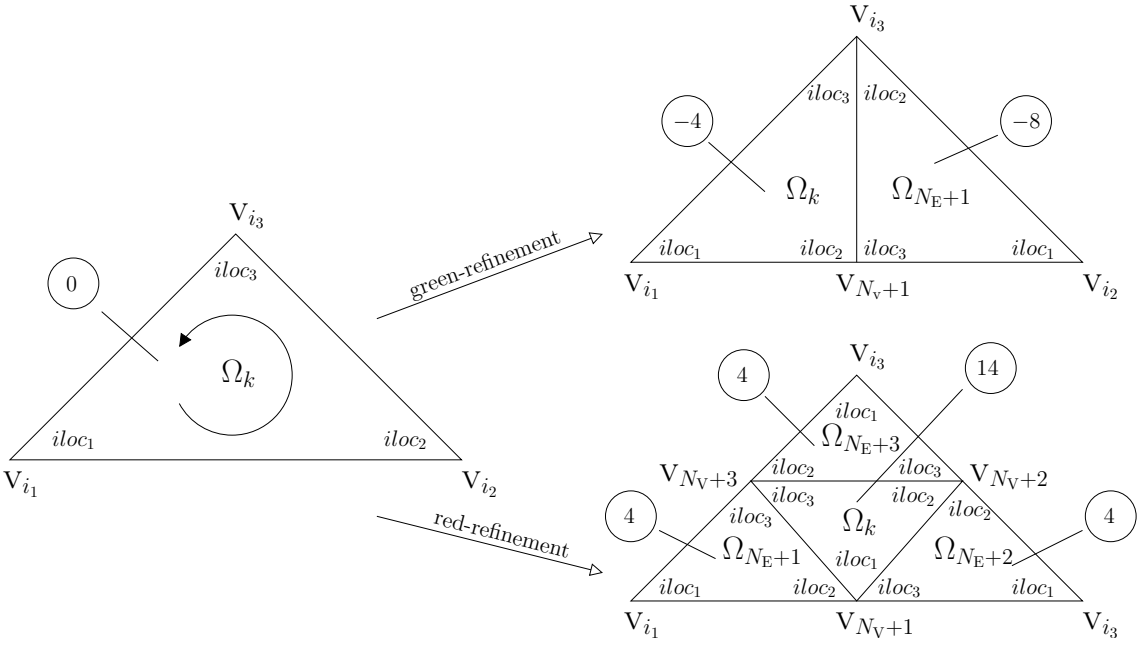


Fig. 4.19. Local two-level ordering of vertices for triangles in 2D.

By virtue of two-level ordering, the identification of elements can be simplified significantly. Let triangle Ω_k possess a single node with largest generation number and consider criteria (f.1)–(f.3) of Theorem 4.6.1 which may require the investigation of multiple neighboring elements. Due to the particular numbering strategy sketched in Figures 4.18 and 4.19, it suffices to investigate the adjacent cell Ω_l along the second edge. It is easy to verify that element Ω_l can possess:

- one vertex with largest generation number at local position 3, then (f.2) applies;
- one vertex with largest generation number at local position 1, then (f.3) applies;
- two vertices with equal generation numbers at local positions 2 and 3, then (f.1) applies.

Thus, the brute-force analysis of all cells surrounding Ω_k can be replaced by selective checks applied to the single elements Ω_l whose position is known *a priori*. Such 1-check criteria can be devised for all statements of Theorem 4.6.1 so that an efficient identification of elements is feasible. The entire presentation of all 'technical subtleties' is beyond the scope of this work.

4.6.6. Practical implementation

To our belief, the usability of any mesh adaptation algorithm rises and falls with its flexibility and the ease of implementation. In this section, we explain some technical details which may be helpful for programming red-green adaptivity in an efficient way. In particular, the local two-level ordering strategy presented in Section 4.6.5 is adopted consistently so that a wealth of *a priori* knowledge about the internal structure of the triangulation is available at no additional costs.

The marking of elements for red and green refinement can be performed with aid of the so-called *marker function* $\mathcal{M} : \mathcal{E}_m \to \mathbb{N}_0$. Its task is to assign a natural number to each element $\Omega_k \in \mathcal{E}_m$ from which the refinement operation to be performed can be determined directly. A viable approach is to use binary arithmetic (in MSB representation) and associate the edge locally numbered by l with `bit[l]`. In two space dimension 4 bits suffice to represent all edges of a

type of element Ω_k	value of state function $\mathcal{S}(\Omega_k)$
triangle / quadrilateral of initial triangulation	$00000_2 = 0_{10}$ $00001_2 = 1_{10}$
green quadrilateral from $1Q \triangleright 2Q$ refinement	$00101_2 = 5_{10}$ $10101_2 = 21_{10}$
red quadrilateral from $1Q \triangleright 4Q$ refinement	$01101_2 = 13_{10}$
inner red triangle from $1T \triangleright 3T$ refinement	$01110_2 = 14_{10}$
outer red triangle from $1T \triangleright 3T$ refinement or green triangle from $1Q \triangleright 4T$ refinement	$00100_2 = 4_{10}$
(left) green triangle from $1T \triangleright 2T$, $1Q \triangleright 3T$, $1Q \triangleright 4T$ refinement	$-00100_2 = -4_{10}$
(right) green triangle from $1T \triangleright 2T$, $1Q \triangleright 3T$, $1Q \triangleright 4T$ refinement	$-01000_2 = -8_{10}$
centered green triangle from $1Q \triangleright 3T$ refinement	$-00010_2 = -2_{10}$

Tab. 4.1. List of admissible element states in 2D.

quadrilateral cell. Moreover, the least significant bit is used to distinguish triangles (`bit[0]=0`) from quadrilaterals (`bit[0]=1`) so that $\mathcal{M}(\Omega_k) \in [0_2, 11111_2] = [0_{10}, 31_{10}]$. In three dimensions, we need to account for 12 edges and possibly more than two different types of elements (tetrahedra, hexahedra, prisms) so that cell markers can be stored as Bytes or Words.

If an element Ω_k is destined for regular refinement, then all edges will be subdivided, that is, the corresponding bits of $\mathcal{M}(\Omega_k)$ are set to 1. Of course, the same edge needs to be marked for bisection in the adjacent cell Ω_l until all elements have been processed. The flipping of bits in neighboring cells may yield indication for blue refinement. If the element is a triangle, it suffices to replace all unacceptable markers $0110_2 = 6_{10}$, $1010_2 = 10_{10}$ and $1100_2 = 12_{10}$ by $1110_2 = 14_{10}$, that is, regular refinement is enforced. For quadrilaterals, the set of invalid markers can be easily deduced from the admissible refinement patters depicted in Figure 4.13. Note that the detection and elimination of blue refinement markers needs to be performed recursively for neighboring cells until all invalid identifiers have been removed. In fact, it suffices to consider only those elements which are adjacent to edges that have been recently marked for subdivision.

For an efficient implementation of mesh refinement and re-coarsening procedures it is essential that the element type can be determined from the nodal generation function (4.6.2) in no time. Let us introduce the *element state function* $\mathcal{S} : \mathcal{E}_m \to \mathbb{Z}$ which is used to assign an integer value to each element of the triangulation. This concept is inspired by the strategy suggested to construct the refinement marker by resorting to binary arithmetic. In two dimensions, the state $\mathcal{S}(\Omega_k)$ of the element Ω_k can be expressed by 5 bits, whereby `bit[0]` is used to distinguish triangles from quadrilaterals. A viable approach to construct the state function is as follows:

- Let `bit[0]=1` for quadrilaterals and `bit[0]=0` for triangles and exit if all nodes have generation number zero, that is, the element belongs to the initial triangulation.
- Set `bit[l]=1` if the two endpoints of the l^{th} edge have the same positive generation number.
- If `bit[1..4]` are zero, then determine the local number $l \in \{1,\ldots,4\}$ of the node with largest generation number, set `bit[l]=1` and negate the state. This also applies to triangles if the bit that corresponds to the unique vertex with largest generation number is not set.

The last rule is slightly more complex but it can be implemented efficiently since only bit modifications are performed. If the particular two-level ordering strategy of Figures 4.19 and 4.18 is employed, the different types of elements can be characterized by the ten different states given in Table 4.1. To make the presentation self-contained, these states are indicated by encircled numbers in aforementioned Figures. In essence, the states of all cells can be evaluated at the beginning of each adaptation step and stored in linear space. As a result, the identification of element types by means of Theorem 4.6.1 reduces to a few case selections, whereby the local two-level ordering of vertices can be used exclude sets of impossible constellations *a priori*.

4.6.7. Summary of hierarchical mesh adaptation

Let us summarize what we have said so far. A hierarchical grid adaptation algorithm based on the red-green strategy can be used for the efficient treatment of transient flow problems. Cells which are marked for refinement are subdivided regularly (red refinement) and hanging nodes are eliminated by means of transition elements (green refinement). Green elements are removed prior to performing further refinement so as to avoid the degeneration of the mesh quality due to acute triangles. The re-coarsening of element patches is an inverse operation which deletes superficial vertices and restores the original macro element. Both the refinement and the re-coarsening procedure can be based on the generation function (4.6.2) which stores the date of birth for each node. If a vertex is inserted at the midpoint of an edge, the maximum age of the two endpoints is increased by one and used as the new generation number. For internal nodes, all surrounding vertices are considered and their maximum age increased by one is adopted. As a result, the youngest node(s) of a particular element always correspond to vertices which can be deleted at first in order to recover the original macro element. The presented mesh re-coarsening algorithm starts from a list of nodes which are potential candidates for removal and a recursive procedure is applied to 'lock' vertices which must remain in the triangulation. Once the algorithm has terminated, the nodes which are unlocked are eliminated and the corresponding macro elements are restored.

In a practical implementation, both the cell marker and the element state can be expressed by integers, whereby individual bits are associated with the numbers that denote the local position of edges/vertices in the element. It is expedient to adopt the two-level ordering strategy so that the structure of cell adjacencies is known *a priori*. In principle, the local ordering strategy can be chosen arbitrarily for each type of elements. On the other hand, labeling vertices, edges (and faces in 3D) locally per elements demands skill in order to extract most information from the element states. In particular, ten different states suffice to identify elements in two space dimension. Remarkably, the element states provide all necessary information required to address the 'correct' cell neighbor directly without performing an extensive investigation of neighboring elements.

The main steps of the hierarchical red-green adaptation algorithm are presented in Figure 4.20. Let us remark that the two interpolation steps can be combined and performed at the very end of the adaptation procedure if stationary flow computations are performed. In this case, loss of mass and/or reduced accuracy of the interpolated profile on the adapted grid can be neglected since the new solution only serves as an 'initial guess' that is marched towards steady state. For time-dependent flows it is worthwhile to introduce separate interpolation steps after refinement and re-coarsening. For the first one, the finite element interpolation will do since the triangulation $\mathcal{T}_m$ is only *enriched* by new vertices and elements. It does not introduce any numerical diffusion and the numerical solution is conserved naturally. However, the grid re-coarsening may violate the conservation principle so that care must be taken to prevent a spurious loss of mass which may occur otherwise. On the other hand, cells are typically combined to their macro elements in regions of nearly uniform flow so that the loss of mass is expected to be sufficiently small.

Given: conformal initial triangulation $\mathcal{T}_0 = (\mathcal{E}_0, \mathcal{V}_0)$

For $m = 0, 1, \ldots, M$

1. *Compute numerical solution on triangulation $\mathcal{T}_m$.*
   - Solve for the steady-state solution u^n ($n \to \infty$), or
   - perform k time steps to update the solution $\mathrm{u}^{n-k} \to \mathrm{u}^n$.

2. *Compute an error distribution e^n for the solution u^n on triangulation $\mathcal{T}_m$.*
   - Employ *a posteriori* error estimation, or
   - use one or more error/feature indicators.

3. *Mark elements for refinement based on error distribution e^n.*
   - Allocate temporal memory and construct marker $\mathcal{M}(\Omega_k)$ for all elements $\Omega_k \in \mathcal{E}_m$.
   - Determine new dimensions of $\mathcal{E}'_{m+1}$ and $\mathcal{V}'_{m+1}$ and allocate memory accordingly.

4. *Mark elements for re-coarsening based on error distribution e^n.*
   - Initialize the deletion indicator $d(\mathrm{v}_i) := |g(\mathrm{v}_i)|$ for all nodes $\mathrm{v}_i \in \mathcal{V}_m$.
   - Apply the algorithm from Fig. 4.15 to lock vertices which should not be deleted.
   - Determine new dimensions of $\mathcal{E}_{m+1}$ and $\mathcal{V}_{m+1}$.

5. *Perform adaptive mesh refinement $\mathcal{T}_m \to \mathcal{T}'_{m+1}$.*
   - Loop over all elements $\Omega_k \in \mathcal{E}_m$ and refine them according to the marker $\mathcal{M}(\Omega_k)$.
   - Interpolate solution values u^n to the new triangulation $\mathcal{T}'_{m+1}$.

6. *Perform adaptive mesh re-coarsening $\mathcal{T}'_{m+1} \to \mathcal{T}_{m+1}$.*
   - Loop over all elements $\Omega_k \in \mathcal{E}'_{m+1}$ and re-coarsen patches based on locked vertices.
   - Interpolate intermediate solution values from step 5. to the final triangulation $\mathcal{T}_{m+1}$.
   - Adjust dimensions of $\mathcal{E}_{m+1}$ and $\mathcal{V}_{m+1}$ according to their values determined in step 3.

7. *Update/re-assembly all system matrices based on $\mathcal{T}_{m+1}$ and proceed to step 1.*

Fig. 4.20. Summary of the mesh adaptation procedure.

4.7. Numerical examples for scalar conservation laws

In this paragraph, we apply recovery-based error indicators to convection dominated flow problems and perform adaptive mesh refinement to improve the accuracy of solution profiles at reasonable cost. The numerical solutions were computed by the high-resolution schemes addressed in Chapter 3. In particular, upwind-biased flux limiting of TVD type (Section 3.3) is employed for stationary problems and transient flow simulations are performed by the semi-implicit FEM-FCT algorithm (Section 3.4). A recovery-based error indicator is used to identify elements which have to be refined. In particular, we employ the gradient-recovery techniques presented in Section 4.3 for the computation of smoothed slope values which are compared to the consistent finite element gradients to obtain an estimate for the true gradient error (4.3.2). The global L_2-norm of the predicted gradient error is decomposed into individual element contributions which serve as indicators for insufficiently resolved regions in the computational grid. The adaptation strategy described in Section 4.4 is used to mark cells for refinement and re-coarsening based on the elementwise error indicator. Mesh adaptation is performed as explained in Section 4.6, whereby pure triangular as well as mixed triangulations are considered to demonstrate the flexibility of our hierarchical approach. The examples that follow are meant to illustrate the potential of adaptive finite element schemes applied to stationary and time-dependent flows. For a detailed description of the underlying benchmark configurations, the interested reader is referred to Section 3.5.

4.7.1. Stationary convection-diffusion

Mesh adaptivity for stationary flows follows the algorithm outlined in Figure 4.20. In essence, the steady state solution is computed on the (initial) triangulation $\mathcal{T}_m$, for $m = 0, 1, \ldots$ which is locally refined based on the elementwise error distribution e. Ideally, this process continues until mesh convergence is achieved, that is, the global solution error (4.2.1) and/or a particular quantity of interest (4.2.2) satisfies a desired tolerance so that no further refinement is required. In practice, shock waves and steep gradients may lead to 'infinitely' small cells if no maximum refinement level is prescribed *a priori*. In this thesis, we decided on a maximum number of elements/vertices so that numerical solutions could be computed in acceptable time by the underlying flow solver and determined the number of admissible refinement levels accordingly.

Shapiro [236] suggested to allow for 'halfway' converged intermediate solutions, that is, the square root of tol_{steady} is used to monitor steady state convergence, i.e. $\|K^*(\mathrm{u})\mathrm{u}\|^2 < tol_{\text{steady}}$. The original convergence parameter is only adopted on the last grid so as to obtain a fully converged final solution. In the simulation of stationary problems we observed a phenomenon also reported by Shapiro [236]: Each adaptation step $\mathcal{T}_m \rightarrow \mathcal{T}_{m+1}$ tends to increase the number of elements N_{E} until a maximum is reached. The largest number of elements is typically present in the triangulation, when the finest admissible refinement level is first introduced. Successive adaptation steps tend to remove cells near discontinuities until the number of elements 'converged'.

Test problem 1: Our first numerical example proposed by Hughes et al. [110] deals with the stationary convection-diffusion equation (3.5.1) in two space dimensions. The details of this benchmark including concomitant boundary conditions are given in Section 3.5.1. John and Knobloch [117, 118] used this test case to compare various stabilization techniques for finite element methods based on the quantities (3.5.3)–(3.5.6). Obviously, the sums over solution values depend on the underlying grid spacing so that only the thickness of the internal layer $smear_{\text{int}} = x_2 - x_1$ yields a meaningful benchmark quantity for locally refined meshes. Here, x_1 and x_2 correspond to the smallest x-coordinates such that $\mathrm{u}_h(x_1, 0.25) \geq 0.1$ and $\mathrm{u}_h(x_2, 0.25) \geq 0.9$, respectively. The nodal values of the numerical solution along the cutline $y = 0.25$ were interpolated on a background grid of width 10^{-7} so that the coordinates x_1 and x_2 could be evaluated with high accuracy. We also evaluated the quantities (3.5.9) and (3.5.10) whose values were below machine precision.

In the original benchmark [117], the numerical solutions were computed by linear finite elements adopting three different grids, whereby the number of degrees of freedom was 4,225 in all triangulations. We expect that locally refined meshes are capable of reproducing the layers more accurately, that is, smearing due to numerical diffusion is likely to reduce if sufficiently small cells are clustered in the vicinity of steep fronts. Therefore, we started from a very coarse background grid $\mathcal{T}_0$ and performed adaptive mesh refinement based on the local L_2-norms of the estimated gradient error. Repeated subdivision of cells was stopped once the number of vertices N_{V} exceeded the value 4,225. Let $\mathcal{T}_m = (\mathcal{E}_m, \mathcal{V}_m)$ be the triangulation, where this happens first. Then the largest generation number (4.6.2) was determined by the following expression

$$g_{\max} = \max_{\mathrm{V}_i \in \mathcal{V}_m} g(\mathrm{V}_i) \tag{4.7.1}$$

and used as upper bound for the admissible refinement level. Successive adaptation steps were performed to delete excess elements until the number of vertices reached a steady state. The aim of this benchmark can be summarized as follows: Impose an upper bound on the number of degrees of freedom and construct an 'optimal' mesh adjusted to the local flow field so that the internal and external layers are captured with least numerical diffusion possible (see also [118]).

For lack of space, it is impossible to present intermediate profiles and partially refined grids for all combinations of error indication techniques and initial triangulations so that we had to make a selection. The numerical solutions depicted in Figures 4.21–4.22 (left) were computed on a sequence of gradually refined meshes, whereby the consistent L_2-projection scheme (4.3.15) was employed to compute improved gradient values $\hat{\nabla}\mathrm{u}_h$ at the nodes. The consistent slopes $\nabla\mathrm{u}_h$ were evaluated from the approximate solution u_h using linear or bilinear finite elements depending on the cell type. Figures 4.21–4.22 (right) display the L_2-norm of the estimated gradient error $\|\nabla\mathrm{u}_h - \hat{\nabla}\mathrm{u}_h\|_{\Omega_k}$ for each element. A logarithmic scaling was used for the visualization neglecting values below 10^{-8} (indicated by purple color in the error distributions). The adaptation strategy presented in Section 4.4 was adopted to equidistribute the gradient error over all cells, whereby the upper/lower bounds $\zeta_{\max} = 5\%$ and $\zeta_{\min} = 1\%$ were imposed on the percentage error (4.4.1).

The initial triangulation consists of 64 quadrilaterals and 81 degrees of freedom. The error indicator reliably detects the poorly resolved internal and external layers and triggers mesh refinement in their vicinity. In the course of successive refinement, a maximum number of 8,060 cells and 7,189 vertices is created. The zone of highest grid point concentration confines itself more and more to the layer regions so that the 'converged' mesh exhibits 20% less elements/vertices; cf. Figure 4.23 (a). It is worth mentioning that the computational grids mainly consist of quadrilateral cells since triangles are only introduced as transition elements (less than 35%).

The crisp resolution of the solution profiles leads to a significant reduction of artificial diffusion. Let the benchmark quantity $smear_{\mathrm{int}}$ =5.73e–2 from Section 3.5.1 serve as 'reference' value which was obtained on a uniform grid using 64×64 bilinear finite elements. For the adaptively refined mesh depicted in Figure 4.22 (top), the number of degrees of freedom decreases by 65% without changing the strength of smearing. The thickness of the internal layer diminishes by factor 2.3 ($\approx$ 5.729e–2/2.447e–2) if two more levels of local refinement are accepted. A similar value for $smear_{\mathrm{int}}$ was computed on a structured grid consisting of 66,049 vertices and 256×256 Q_1 elements. Thus, 10% of the vertices from the regularly refined grid suffice to reproduce the solution profile with the same accuracy if local mesh refinement is performed.

As an alternative to the L_2-projection scheme (4.3.15) for the computation of improved gradient values, we implemented the superconvergent patch recovery (SPR) technique (4.3.21) introduced by Zienkiewicz and Zhu [275]. We also utilized limited gradient averaging (4.3.9) to evaluate smoothed slope values directly at the midpoints of edges adopting the MC limiter (cf. Figure 4.1). As before, the computational grids were adjusted based on the local L_2-norms of the estimated gradient errors. Their distribution on the final mesh is depicted in Figures 4.23 (b) and

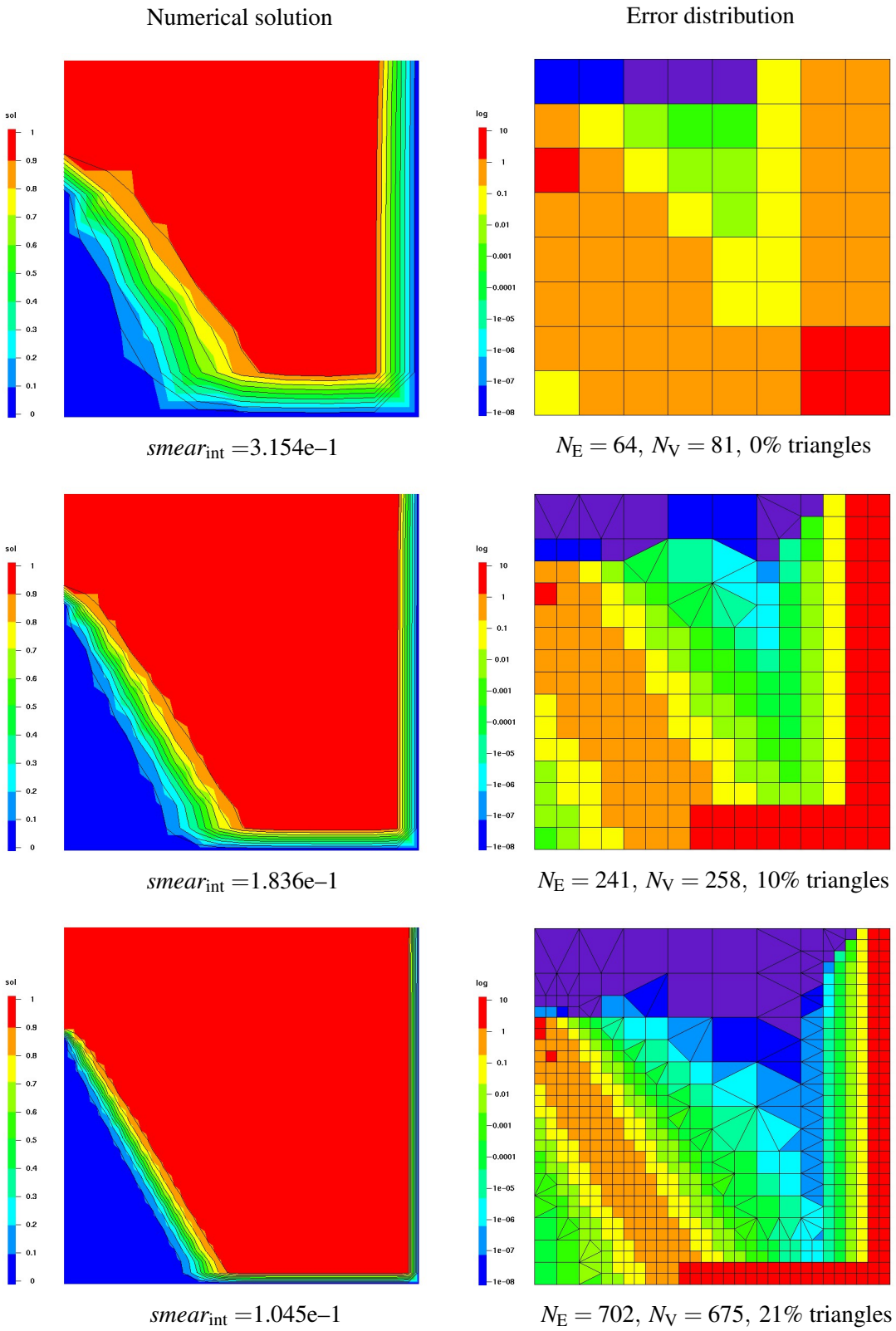


Fig. 4.21. Stationary convection-diffusion: consistent L_2-projection.

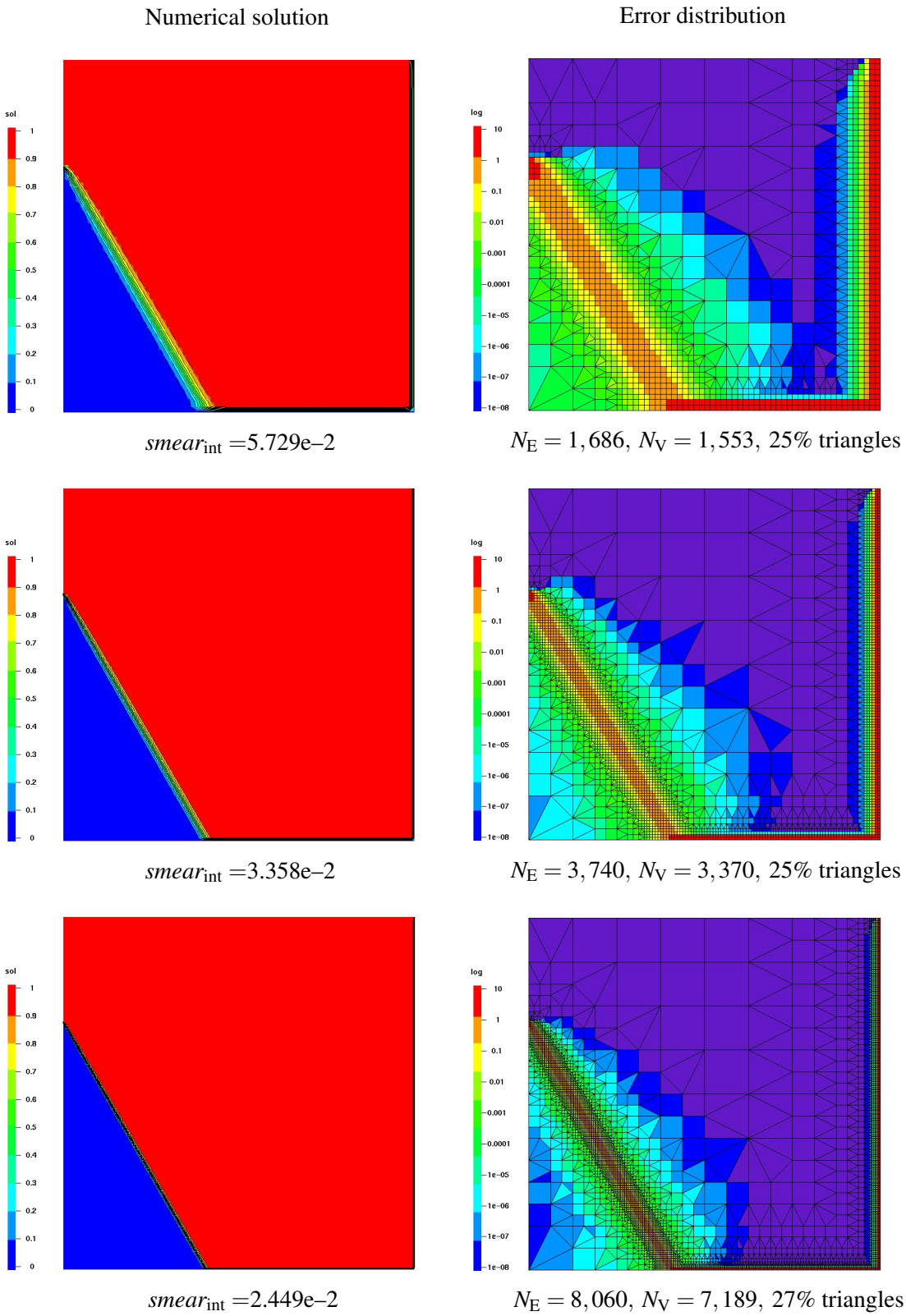
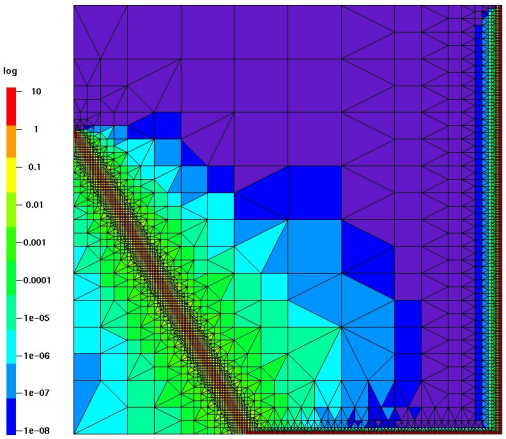


Fig. 4.22. Stationary convection-diffusion: consistent L_2-projection (continued).

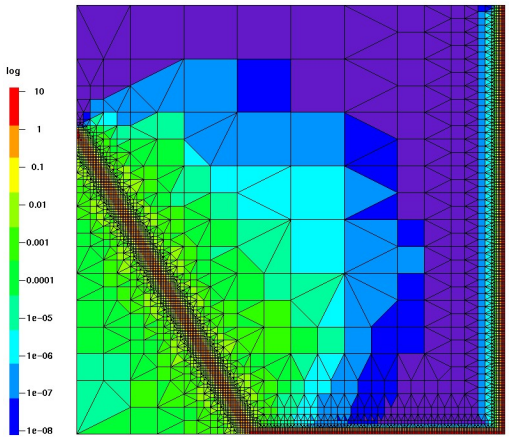
(a) Consistent L_2-projection



$N_{\mathrm{E}} = 6,593,\ N_{\mathrm{V}} = 5,694$

34% triangles, $smear_{\mathrm{int}}$ =2.44713e–2

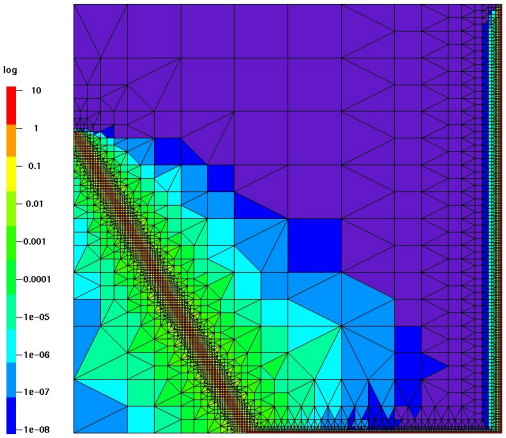
(b) Superconvergent patch recovery



$N_{\mathrm{E}} = 7,279,\ N_{\mathrm{V}} = 6,393$

31% triangles, $smear_{\mathrm{int}}$ =2.44790e–2

(c) Limited gradient averaging



$N_{\mathrm{E}} = 6,469,\ N_{\mathrm{V}} = 5,565$

35% triangles, $smear_{\mathrm{int}}$ =2.44764e–2

(d) Grid history

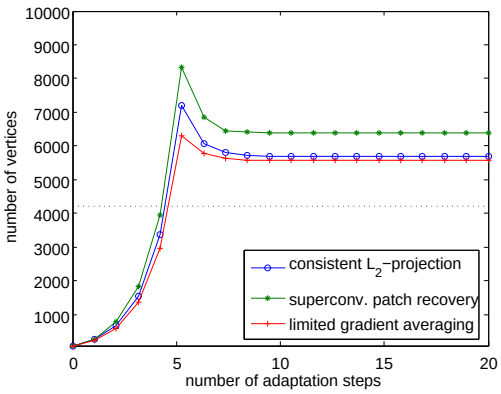


Fig. 4.23. Stationary convection-diffusion: error distribution on the 'converged' mesh.

(c). In essence, the resolution of the internal and external layers is comparable to that produced by the consistent L_2-projection strategy which can be seen from the benchmark quantity $smear_{\mathrm{int}}$. However, the total number of degrees of freedom present in the 'converged' grid is different for all three approaches, whereby the superconvergent patch recovery technique yields the triangulation with the largest number of elements/vertices. Away from steep fronts, the magnitude of the estimated slope errors decays slowly so that less elements were re-coarsened in successive adaptation steps. Conversely, limited gradient averaging identifies the internal layer most precisely, and therefore, it produces the coarsest mesh which is best adjusted to the flow field. The history of grid convergence for all three gradient-recovery techniques is illustrated in Figure 4.23 (d).

The hierarchical adaptation procedure developed in Section 4.6 can be readily applied to pure triangular meshes. The structured mesh depicted in Figure 4.24 (top, right) serves as initial tri-

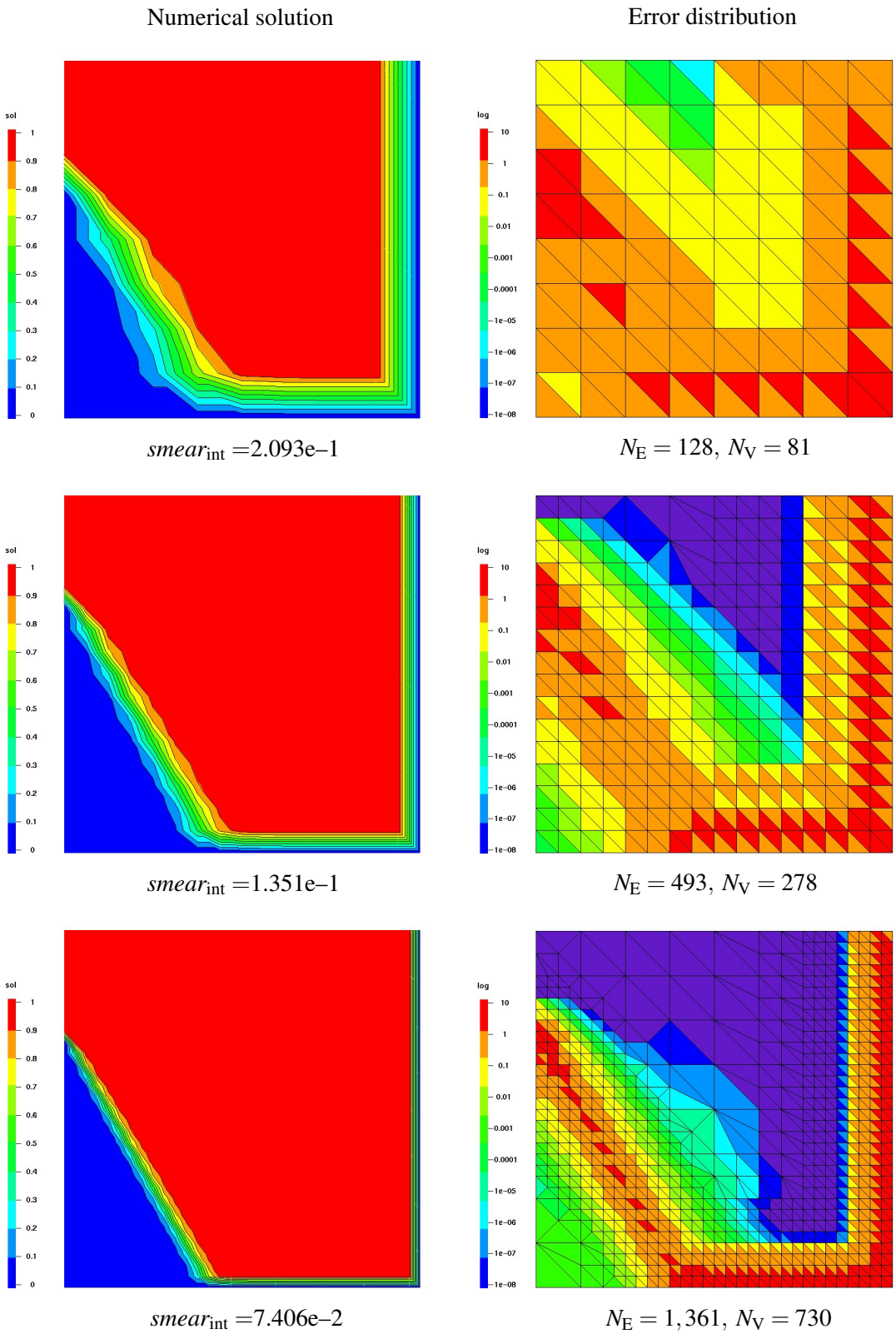


Fig. 4.24. Stationary convection-diffusion: superconvergent patch recovery.

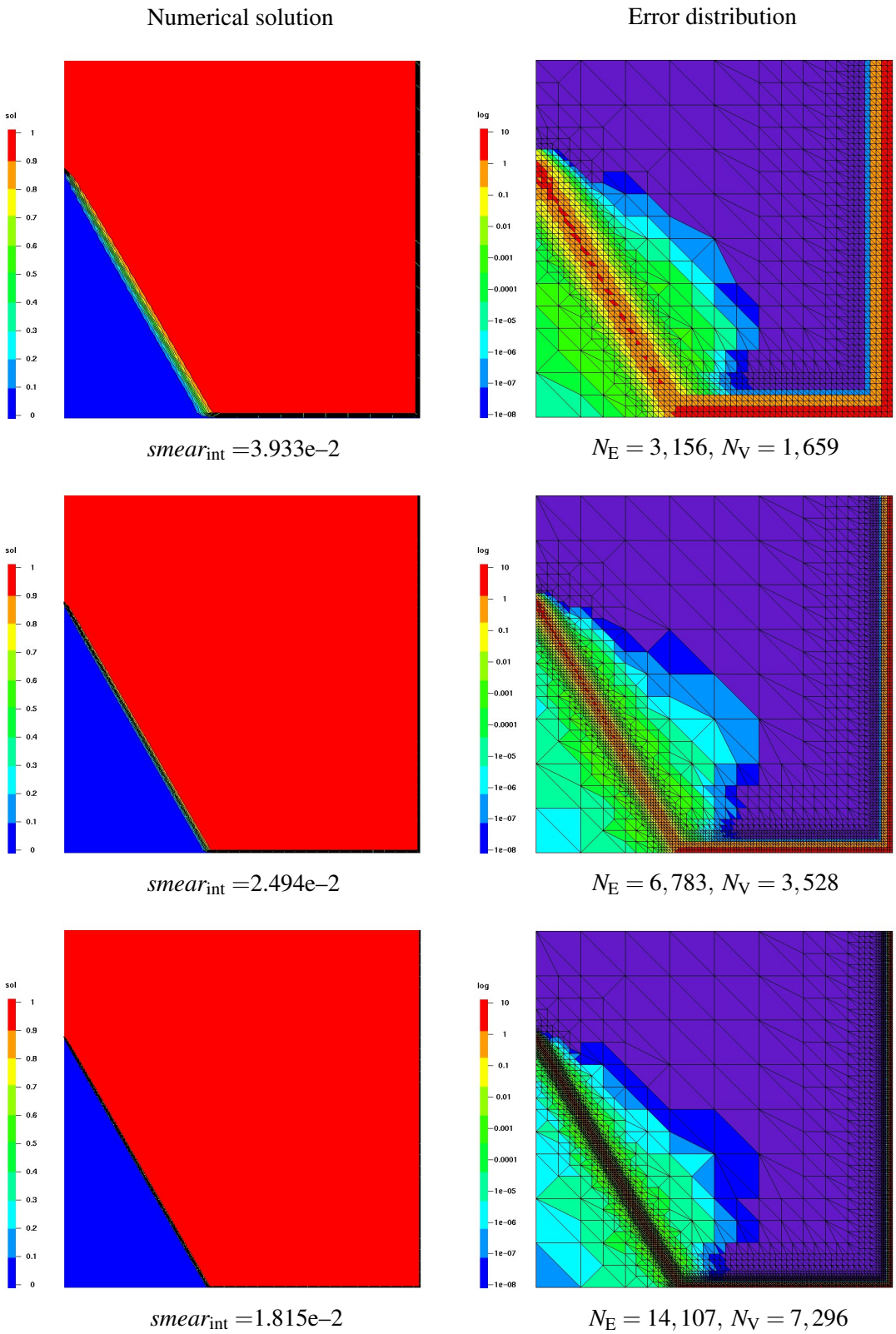
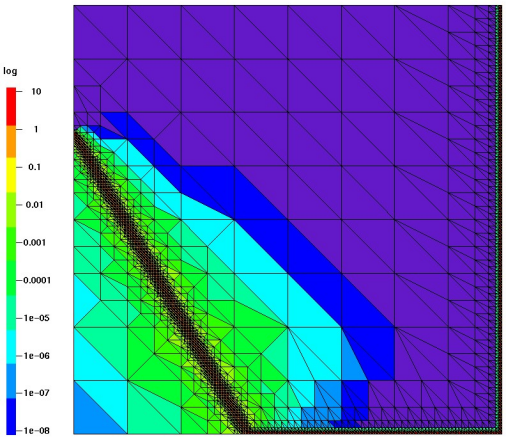


Fig. 4.25. Stationary convection-diffusion: superconvergent patch recovery (continued).

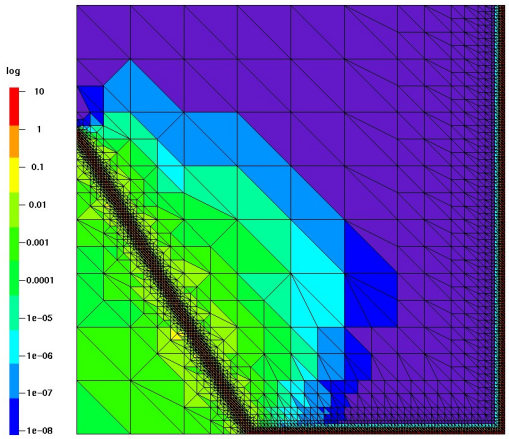
(a) Consistent L_2-projection



$N_{\mathrm{E}} = 6,946,\ N_{\mathrm{V}} = 3,703$

$smear_{\mathrm{int}}$ =1.80484e–2

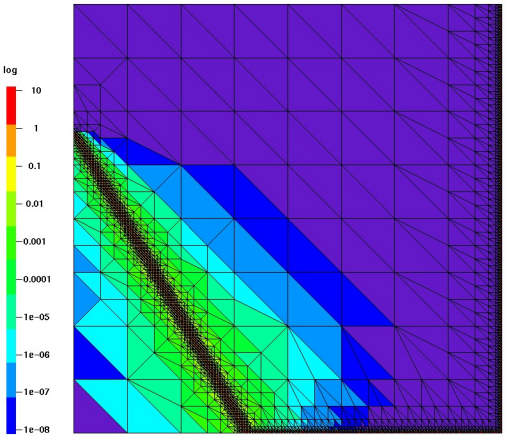
(b) Superconvergent patch recovery



$N_{\mathrm{E}} = 9,319,\ N_{\mathrm{V}} = 4,894$

$smear_{\mathrm{int}}$ =1.80606e–2

(c) Limited gradient averaging



$N_{\mathrm{E}} = 6,688,\ N_{\mathrm{V}} = 3,574$

$smear_{\mathrm{int}}$ =1.80046e–2

(d) Grid history

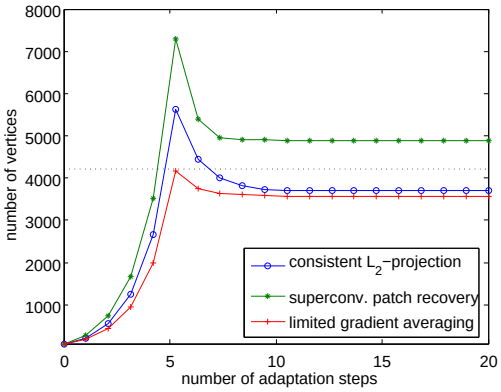


Fig. 4.26. Stationary convection-diffusion: error distribution on the 'converged' triangular mesh.

angulation which results from the regular 8×8 grid by subdividing each quadrilateral into two triangles along the diagonal in the direction of the internal layer. The improvement of the numerical solutions in the course of mesh adaptation and the distributions of local gradient errors predicted by the superconvergent patch recovery technique are depicted in Figures 4.24–4.25. As in the previous example, successive adaptation steps were performed once the number of degrees of freedom reached the upper bound 4,225 until a 'converged' mesh was obtained. Note that 1,659 adaptively positioned vertices suffice to reduce the thickness of the internal layer to the reference value $smear_{\mathrm{int}}$ =3.93e–2 which is an effective saving of 60%. The final computational grids produced by recovery-based error indication based on consistent L_2-projection, the SPR-technique and limited gradient averaging are presented in Figure 4.26 (a)–(c). The zones of high grid point concentration are indistinguishable, whereas the total number of vertices differs considerably.

4.7.2. Swirling flow problem

Let us proceed to transient flows and consider the continuity equation solved in the unit square

$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v}u) = 0 \qquad \text{in} \quad (0,1)^2 \times (0,T) \tag{4.7.2}$$

where T is the final time of the simulation. The non-uniform velocity field $\mathbf{v}$ is given by

$$\mathbf{v}(\mathbf{x},t) = (\sin^2(\pi x)\sin(2\pi y)g(t), -\sin^2(\pi y)\sin(2\pi x)g(t))^T \tag{4.7.3}$$

The details of this benchmark problem proposed by LeVeque [156] are given in Section 3.5.4.
Test problem 2: Let us adopt the time-dependent function $g(t) = \cos(\pi t/T)$ which causes the swirling velocity field to come to rest at $t = T/2$ and reverse its direction for $t > T/2$. Hence, the initial profile is recovered as exact solution, whereby intermediate solutions attain a spiral shape; cf. Figure 3.14. In Section 3.5.4, this test problem was solved by using linear finite elements on a structured grid consisting of 32,768 triangles and 129×129 degrees of freedom. The L_1- and L_2-norm of the solution errors presented in Table 3.9 will serve as reference values.

Let us reduce the number of elements/vertices by resorting to dynamic mesh adaptation as explained in Section 4.6 so as to produce numerical solutions of the same accuracy in *shorter computing time*. For the simulation, a 17×17 coarse mesh was 'pre-adapted' three times. That is, the gradient-based error indicator was applied to the analytically given profile

$$u(x,y,0) = \begin{cases} 1 & \text{if} \quad (x-1)^2 + (y-1)^2 < 0.8 \\ 0 & \text{otherwise} \end{cases} \tag{4.7.4}$$

which was repeatedly prescribed on locally refined grids. The initial mesh produced by performing three steps of local refinement based on the limited gradient averaging approach is shown in Figure 4.27. It consists of 3,960 triangles and comprises 2,022 degrees of freedom. The structured 129×129 grid which was used to compute the reference solution corresponds to applying three steps of global refinement, and hence, the maximum level of *local* refinement was set to 3 to obtain comparable results. Thus, the shortest length of the legs of the triangles is $1/128$ in both cases.

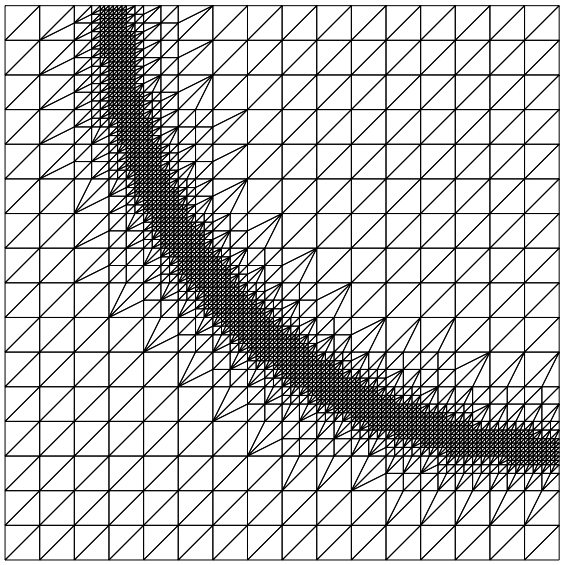


Fig. 4.27. Swirling flow: pre-adapted initial mesh.

Every five time steps (which corresponds to Δt_{adapt} =5.0e–2 for constant Δt =1.0e–3), limited gradient averaging was performed to compute improved slope values which were compared to the consistent finite elements gradient. The computational mesh was adapted with the objective of equidistributing the gradient error as explained in Section 4.4. In our simulations, we adopted the values $\zeta_{\max} = 5\%$ and $\zeta_{\min} = 2\%$ as tolerances for the percentage error (4.4.1). In transient flow computations, the mesh has to be adapted not only to the current solution profile but also to its expected shape in the future. In practice, it is therefore advisable to introduce one or more protection layers to account for future evolution details. In our implementation, all cells adjacent to elements which were selected for refinement by the error indicator were also marked for subdivision. This procedure was applied to the new set of elements, whereby those cells which had already been flagged in the first pass were neglected to avoid the redundant investigation of neighboring elements. More sophisticated strategies may be devised but the above algorithm to construct protection layers of width two is simple to implement and it worked well for our purpose.

The sequence of triangulations shown in Figure 4.28 demonstrates that the dynamic mesh adaptation algorithm developed in Section 4.6 is capable of 'following' the evolution details of transient flows. We performed numerical experiments using the constant value Δt =1.0e–3 and by letting the PID controller (3.5.28) adjust the time step size adaptively based on the relative changes of the solution. The quality of the numerical results is assessed by comparing the approximate solution at the final time $T = 1.5$ to its exact/initial value given by expression (4.7.4). The differences are measured in the L_1- and L_2-norm defined in (3.5.14) and (3.5.15), respectively. The solution errors presented in Table 4.2 are in very good agreement with the reference values shown in Table 3.9 which were computed without mesh adaptivity. The deviation between the different recovery techniques is insignificant and the slight deterioration due to the use of variable time steps is consistent with the effects observed in the reference case.

Time-stepping	$\|u-\mathrm{u}_h\|_1$	$\|u-\mathrm{u}_h\|_2$	$\|u-\mathrm{u}_h\|_1$	$\|u-\mathrm{u}_h\|_2$	$\|u-\mathrm{u}_h\|_1$	$\|u-\mathrm{u}_h\|_2$
	L_2-projection		SPR-technique		lim. averaging	
e_{target} =5.0e–3	7.2480e–3	3.9241e–2	7.2600e–3	3.9300e–2	7.2399e–3	3.9200e–2
Δt =1.0e–3	7.1742e–3	3.8925e–2	7.1697e–3	3.8924e–2	7.1807e–3	3.8945e–2

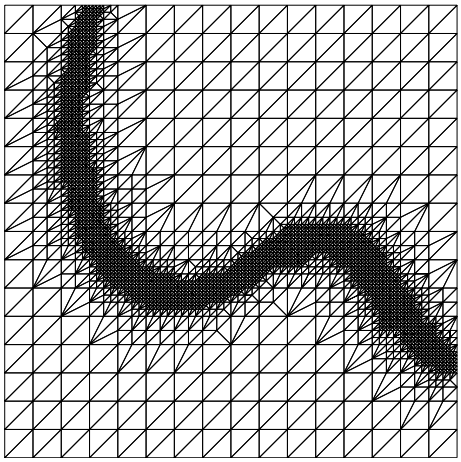
Tab. 4.2. Swirling flow: solution error for dynamic mesh adaptivity.

The quality of the adapted mesh depends on the reliability of the error indicator to a large extent. Hence, the hierarchical mesh adaptation algorithm may create more grid points than required if the local gradient error is not well predicted. To investigate the facilities of the underlying gradient-recovery techniques, we plotted the number of degrees of freedom versus simulation time in Figure 4.29. Obviously, all three error indicators provoke refinement as the initial profile is deformed by the velocity field and succeed in re-coarsening the mesh once the flow reverts to its initial data. In fact, the computational grids look similar to those shown in Figure 4.28, and therefore, they are not displayed in this thesis. However, the total number of vertices present in the sequence of triangulations is different which can also be seen from their maximum values

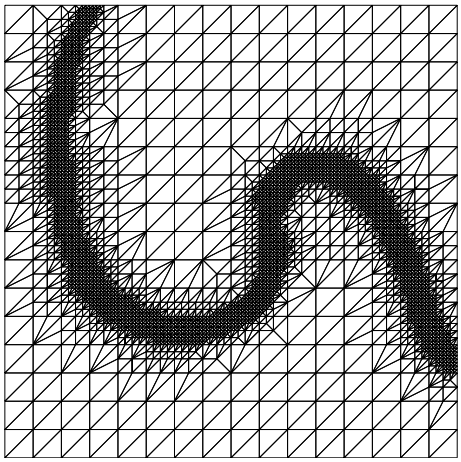
$$L_2\text{-proj}: \quad 3,682, \qquad \text{SPR}: \quad 4,403, \qquad \text{lim. avg.}: \quad 3,301 \tag{4.7.5}$$

In essence, the superconvergent patch recovery technique tends to predict larger values for the local gradient errors so that more elements located at the foothills of the discontinuity are marked for refinement. On the other hand, consistent L_2-projection and limited gradient averaging identify the moving front more precisely, and hence, they do not give rise to excess elements.

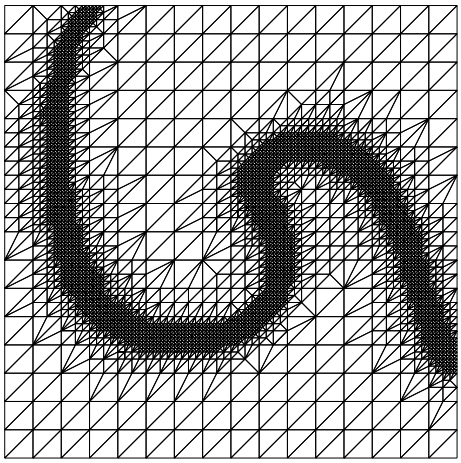
(a) simulation time $t = 0.25$



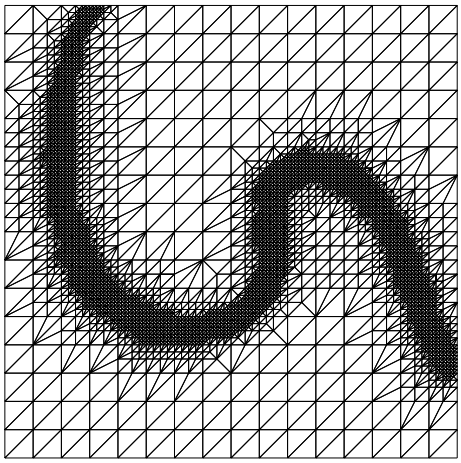
(b) simulation time $t = 0.5$



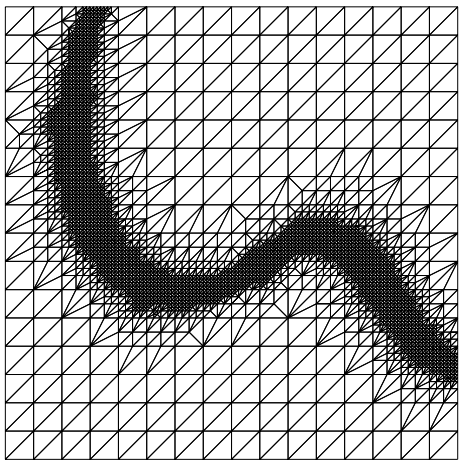
(c) simulation time $t = 0.75$



(d) simulation time $t = 1.0$



(e) simulation time $t = 1.25$



(f) simulation time $t = 1.5$

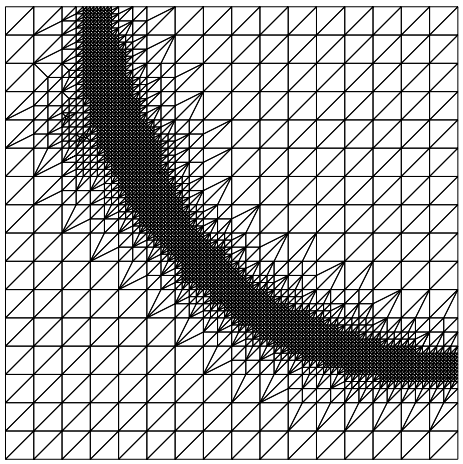


Fig. 4.28. Swirling flow: dynamically adapted computational meshes.

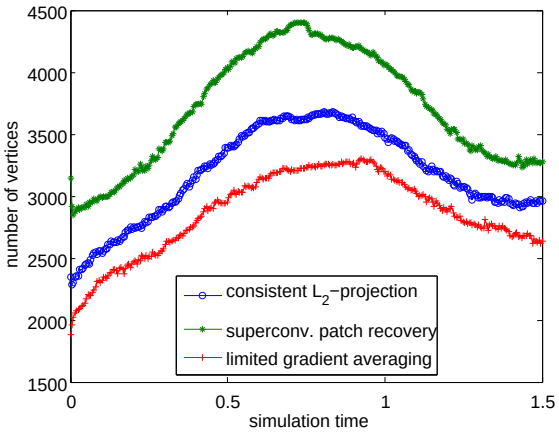


Fig. 4.29. Swirling flow: evolution of the vertex number.

Error indicator	CPU	NT	NN	NL	NN/NT	NL/NN
Defect correction scheme, η =1.0e–1, e_{target} =5.0e–3						
L_2-projection	180 sec	1,262	28,924	170,257	22.92	5.89
SPR-technique	262 sec	1,242	28,560	164,901	23.00	5.77
lim. averaging	165 sec	1,277	29,216	174,966	22.88	5.99
Defect correction scheme, η =1.0e–1, Δt =1.0e–3						
L_2-projection	220 sec	1,500	32,778	191,211	21.85	5.83
SPR-technique	298 sec	1,500	32,776	187,472	21.85	5.72
lim. averaging	196 sec	1,500	32,802	195,126	21.87	5.95
Newton's method, η from (3.3.86), $\sigma = \sqrt{\varepsilon}$, e_{target} =5.0e–3						
L_2-projection	96 sec	1,262	8,301	58,115	6.58	7.00
SPR-technique	163 sec	1,242	7,884	55,087	6.35	6.99
lim. averaging	**87 sec**	1,277	8,474	60,398	6.64	7.13
Newton's method, η from (3.3.86), $\sigma = \sqrt{\varepsilon}$, Δt =1.0e–3						
L_2-projection	113 sec	1,500	9,585	66,525	6.39	6.94
SPR-technique	185 sec	1,500	9,259	64,465	6.17	6.96
lim. averaging	**102 sec**	1,500	9,684	68,362	6.46	7.06

Tab. 4.3. Swirling flow: nonlinear vs. linear solution behavior.

Let us dwell on the solution strategies discussed in Chapter 3 and reconsider the numerical results for the swirling flow problem presented in Table 3.9. In short, Newton's method operated with the adaptively chosen forcing term (3.3.86) and a variable time step size provides the most efficient solution algorithm. The same solver configurations were adopted for the numerical results given in Table 4.3, whereby dynamic mesh adaptation based on all three gradient-recovery techniques was performed. Note that the smallest element size (i.e. the length of the legs of the triangles is $1/128$) coincides with that of the structured 129×129 grid which was utilized to compute the reference computations. The relative changes of solution values seem to be larger if locally adapted meshes are employed, and hence, the PID controller (3.5.28) determines smaller Δt so that more steps are required to reach the final time $T = 1.5$. In essence, the convergence behavior of the nonlinear and linear solution strategies coincide with their counterparts in Table 3.9. That is, Newton's method converged within 6 iterations per time step, whereas 3–4 times more cycles were performed by the defect correction scheme. Consequently, the CPU time required to compute the final solution reduced by using variable time steps and adopting the discrete Newton algorithm in lieu of the defect correction approach. Remarkably, the choice of the gradient-recovery techniques has a great impact on the overall computing time. The solution of local least squares problems required by the superconvergent patch recovery procedure described in Section 4.3.2 is rather time-consuming. For elliptic equations, the overhead cost may pay off by turning expression (4.3.4) into a robust estimator for the global solution error [274, 276]. However, the identification of steep fronts suffices for our purpose, and thus, limited gradient averaging yields the most efficient error indicator which is simple to integrate into existing codes. It is worth mentioning that the adaptation procedure required less than 1% of the total CPU time running in serial mode.

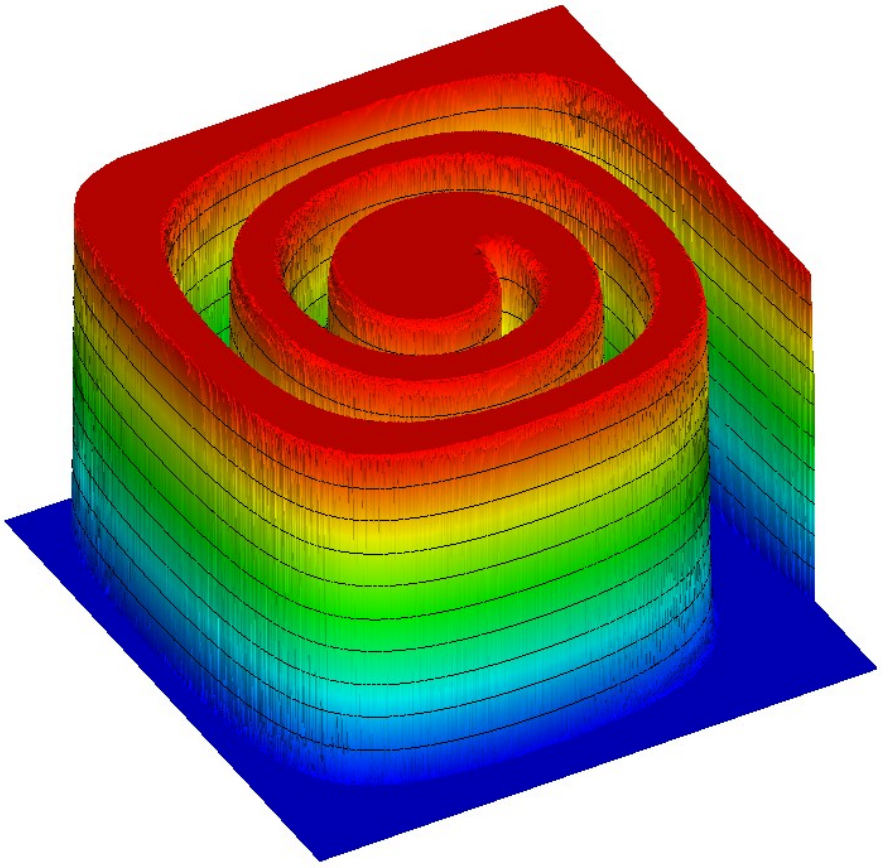
Let us summarize what we have said so far. The dynamic mesh adaptation algorithm developed in Section 4.6 is particularly useful to adjust the grid to local flow features so as to reduce the computational cost of transient flow simulations. In fact, the performance of the simplest solution strategy, namely, the fixed-point defect correction scheme operated with constant time step size, improves by factor 7 ($\approx$ 1367sec/165sec) if locally adapted meshes are employed. The most efficient solution algorithm for this benchmark problem builds on Newton's method and makes use of the PID controller (3.5.28) to adjust the time step dynamically. At the end of the day, the total CPU time can be reduced to only 6% (i.e. 87 sec) of its worst case value 1,367 sec.

4.7.3. Swirling deformation

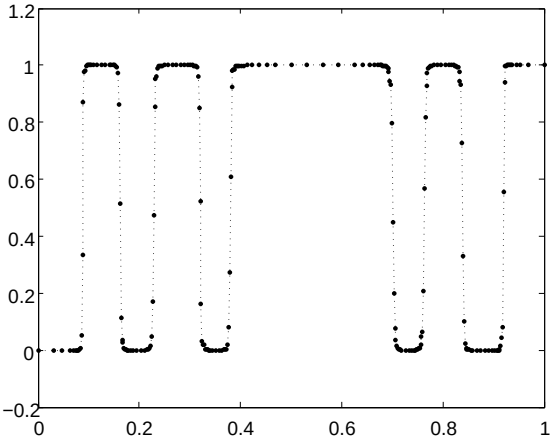
In our last numerical example for scalar conservation laws, mesh adaptation is used to *improve the spatial accuracy* of the high-resolution FEM-FCT scheme without performing successive global refinement which is prohibitively expensive, and therefore, not feasible for practical applications.

Test problem 3: Consider the time-dependent continuity equation (4.7.2) and let the solution be initialized by expression (4.7.4). The initial mass distribution is exposed to the incompressible velocity field (4.7.3) which is constant for $g(t) \equiv 1$ so that the complex spiral shape depicted in Figure 4.30 is obtained at time $t = 2.5$. This benchmark proposed by LeVeque [156] was employed in [139, 187] to assess the resolution power of flux-correction algorithms using linear and bilinear finite elements on structured grids. In this thesis, we present numerical results for highly unstructured meshes which are created from the coarse grid shown in Figure 4.31 by means of local mesh refinement. Limited gradient averaging was repeatedly applied to the profile (4.7.4) so as to construct the initial computational mesh which consists of 15,547 vertices and 10,996 elements. For this test case, we utilized the parameter values $\zeta_{\max} = 5\%$ and $\zeta_{\min} = 2\%$ for the percentage error (4.4.1) and let each cell of the initial grid be refined five times at most. Note that global subdivision of the coarse grid up to the maximum refinement level 5 yields 513×513 degrees of freedom and nearly 400,000 elements which is impractical for non-parallel codes.

(a) Numerical solution at time $t = 2.5$



(b) Cross section along $x = 0.5$



(c) Cross section along $y = 0.5$

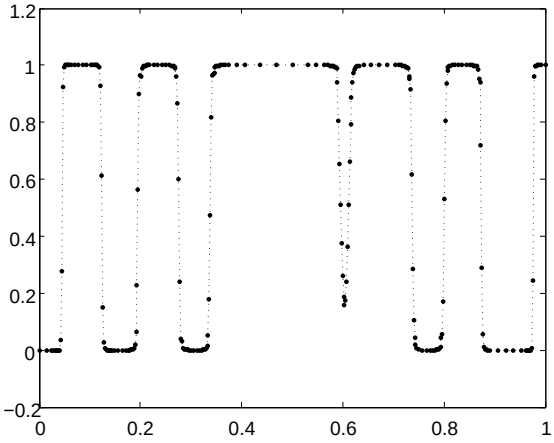


Fig. 4.30. Swirling deformation: adaptive finite element scheme.

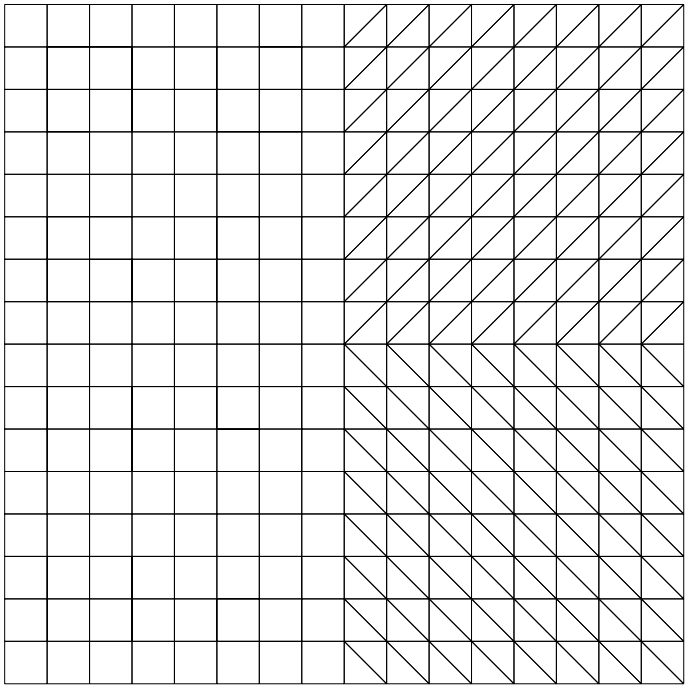


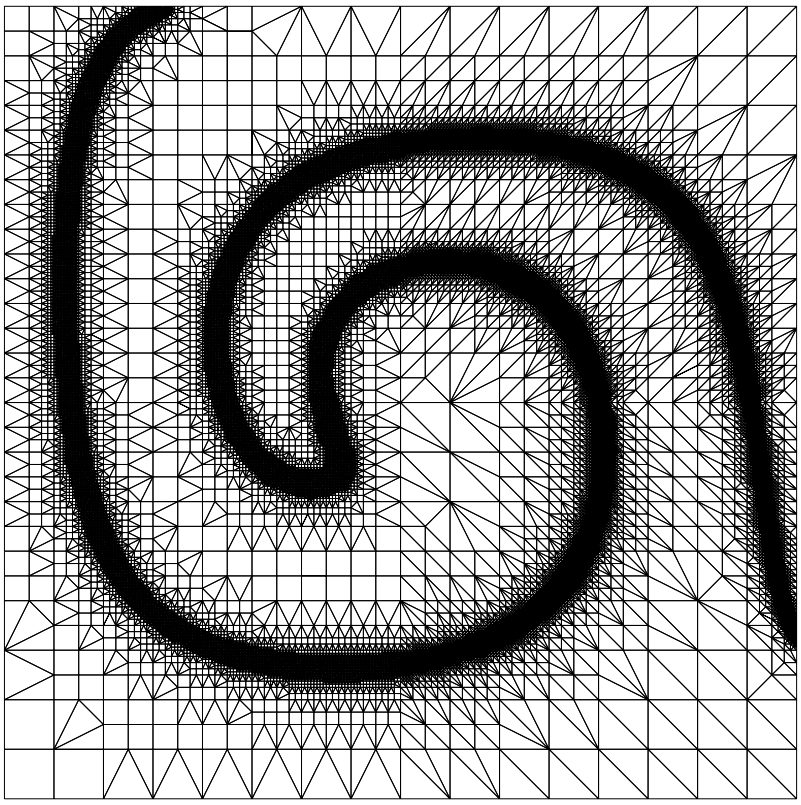
Fig. 4.31. Swirling deformation: initial coarse grid.

The aim of this benchmark is to produce highly accurate numerical solutions at a fraction of the computational costs engendered by globally refined grids. Note that the initial profile (4.7.4) separates the unit square into two simply shaped zones of constant density, whereas the computational domain is entirely occupied by the swirling flow field as time evolves. Hence, the challenging task of the adaptation algorithm is to keep the mesh as coarse as possible and perform local refinement only in the vicinity of steep fronts. We utilized the semi-implicit FEM-FCT algorithm [137] for the discretization in space and performed integration in time by the second-order accurate Crank-Nicolson scheme ($\theta = 0.5$). The constant value Δt =1.0e–3 was adopted which corresponds to performing 2,500 time steps to compute the solution at the final time. The mesh was adapted every five steps to ensure accurate spatial resolution, whereby a protection layer of width two was introduced to account for variations of the solution within the time interval Δ_{adapt} =5.0e–3.

The crisp resolution of the solution produced on the locally adapted mesh at time $t = 2.5$ can also be seen from the cross sections depicted in Figure 4.30 (b) and (c). The bold dots correspond to solution values sampled at the actual grid points located on the cutline $x = 0.5$ and $y = 0.5$, respectively. Note that the finest mesh width is $1/512 \approx 1.9$e–3 so that the tolerance 1.0e–3 was adopted for the search radius. The cross section $x = 0.5$ separates the region of triangular elements ($x > 0.5$) from the mixed triangulation ($x < 0.5$), whereby 221 common grid points are located on the cutline. Conversely, the cross section along $y = 0.5$ depicted in Figure 4.30 (c) runs through both regions. It is worth mentioning that no spurious transition between triangles and quadrilaterals is visible in the distribution of the 240 solution values sampled at the vertices.

The sequence of triangulations recovered at simulation times 1.0, 1.5, 2.0 and 2.5 are presented in Figures 4.32–4.33. The counterclockwise furling of the initial profile as time goes on is very well represented by the underlying grids. The computational mesh at the final time consist of 137,390 elements and 89,576 degrees of freedom which is only 35% of the computer memory necessary for global mesh refinement. Hence, dynamic mesh adaptation provides a useful tool to lower the storage requirement which leads to the reduction of the overall computing time.

(a) time $t = 1.0$



(b) time $t = 1.5$

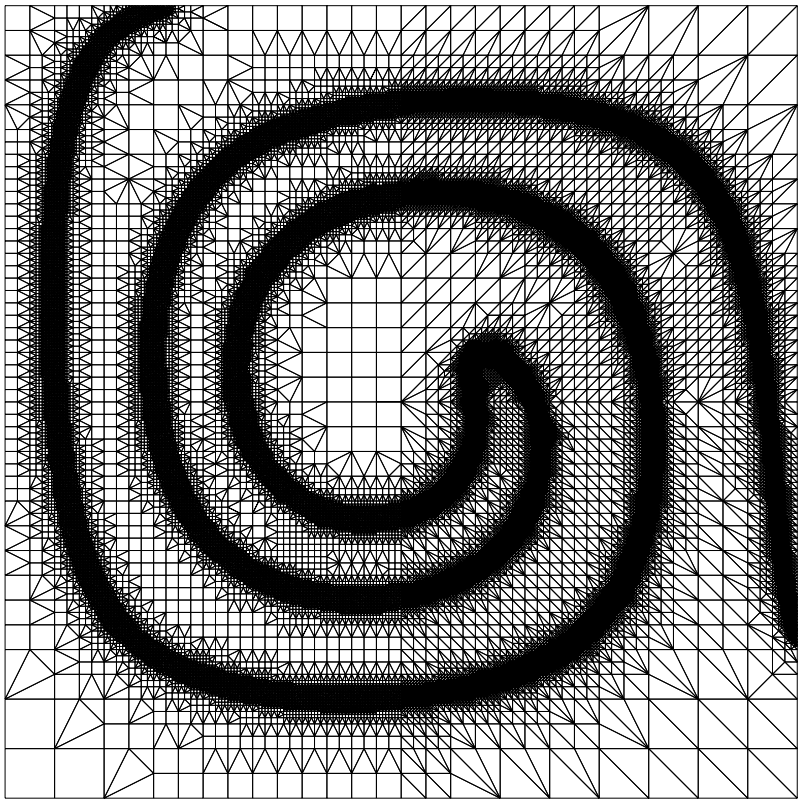
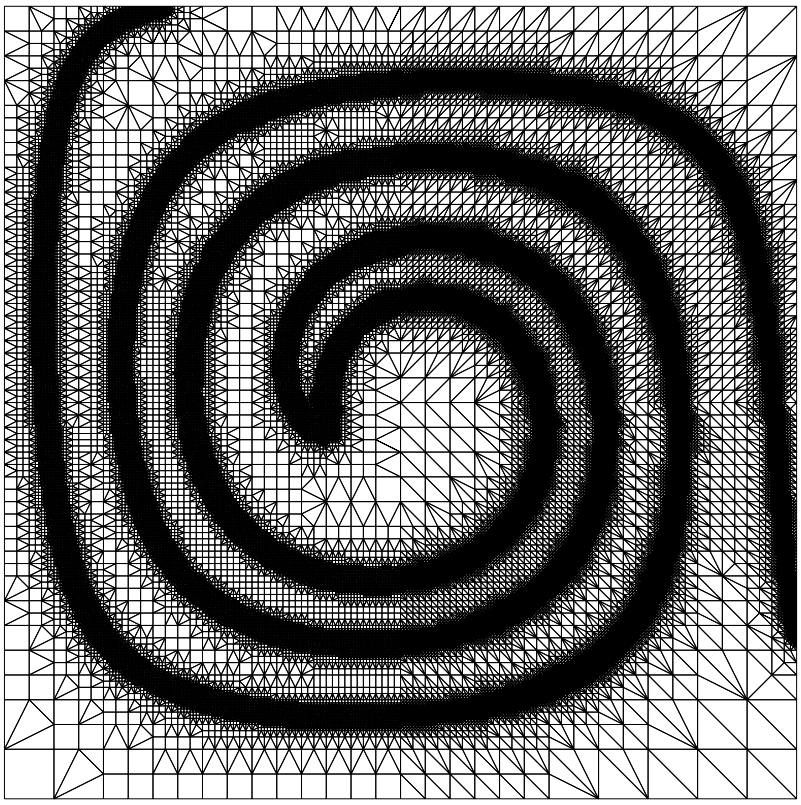


Fig. 4.32. Swirling deformation: dynamic mesh adaptation.

(c) time $t = 2.0$



(d) time $t = 2.5$

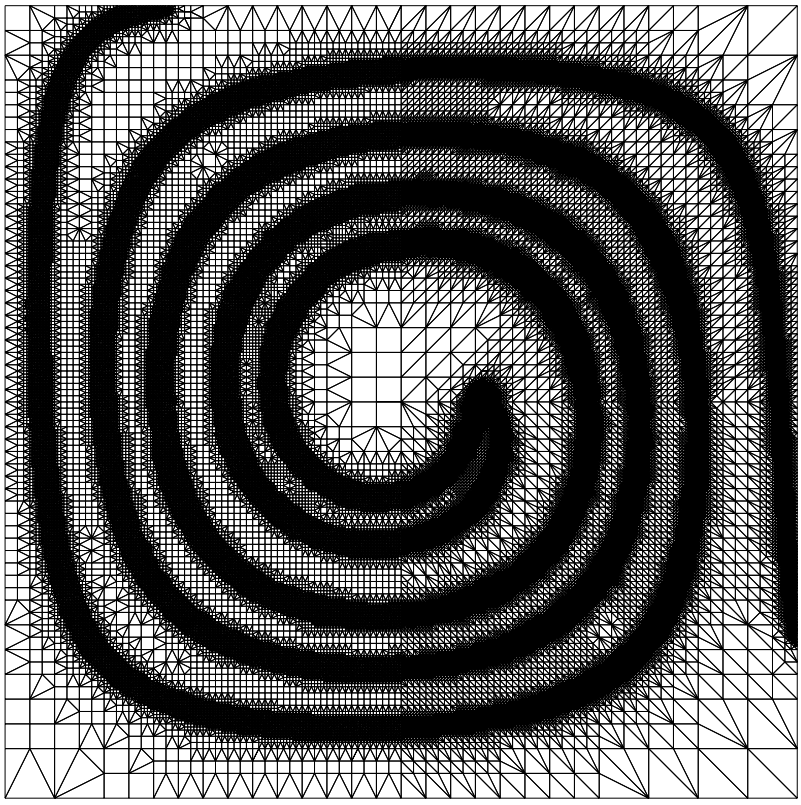
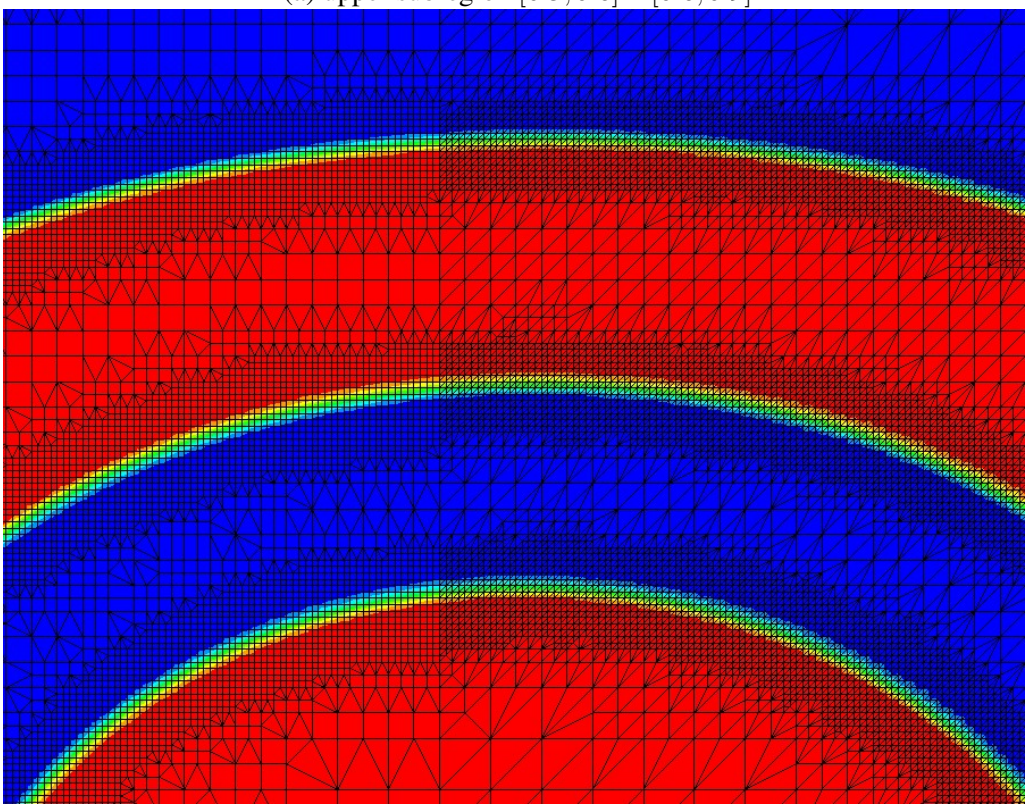


Fig. 4.33. Swirling deformation: dynamic mesh adaptation (continued).

(a) upper subregion $[0.3, 0.6] \times [0.6, 0.9]$



(b) lower subregion $[0.3, 0.6] \times [0.2, 0.5]$

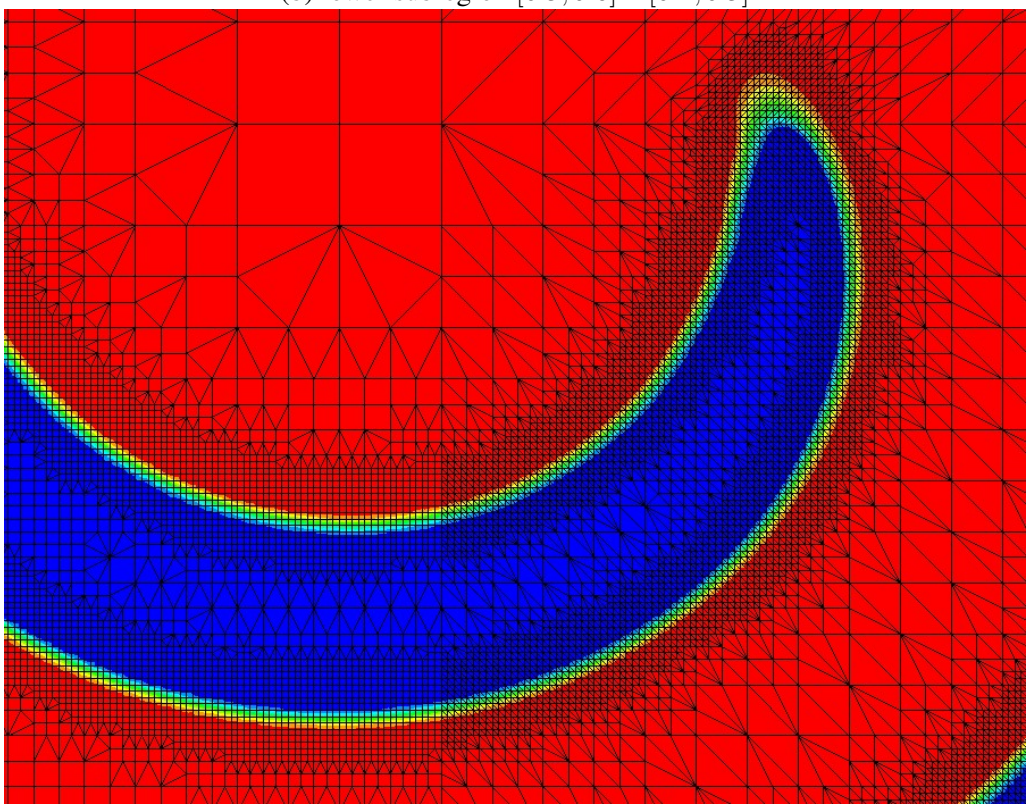


Fig. 4.34. Swirling deformation: closeup view of adapted mesh at time $t = 2.5$.

A closeup view of the final solution profile computed on the locally adapted mesh is shown in Figure 4.34. By construction, the grid contains only triangles for $x > 0.5$, whereas the use of triangular transition elements give rise to a mixed triangulation for $x < 0.5$. However, either pure linear ($x > 0.5$) or bilinear ($x < 0.5$) finite elements are employed in the vicinity of steep fronts. Note that there is no difference in the approximation quality in the transition between triangles and quadrilaterals, i.e. along $x = 0.5$. The presented numerical results show that algebraic flux correction schemes are readily applicable to hybrid grids which may consist of different cell types from the outset or make use of triangular transition elements in the course of mesh adaptation.

```
================================================
Time for ...              % time     self seconds
================================================
computing solution        72.34%     1.38253341E+04
mesh adaptivity            0.46%     8.79990000E+01
error indication          11.37%     2.17369880E+03
triangulation              0.31%     5.97498000E+01
coefficient assembly       1.15%     2.19738300E+02
matrix assembly           11.49%     2.19514410E+03
residual/rhs assembly      2.62%     5.00788100E+02
pre-/post-processing       0.01%     2.80150000E+00
================================================
total simulation                     1.91119527E+04
```

Fig. 4.35. Swirling deformation: profiling of computational cost.

The simulation ran 318 minutes on an AuthenticAMD Opteron^{TM}Processor 250 using a single core, whereby less than 1% of the CPU time was spent on mesh adaptivity. An overview of the computing time for each part of the code is presented in Figure 4.35. In fact, the solution of the nonlinear and linear problems represents the most expensive task. In summary, the assembly of the global system matrix which needs to be updated each time the grid was modified and the right-hand side/defect vector require less than 15% of the total CPU time. Another 15% of the computing time is spent on error indication, mesh adaptivity and the re-assembly of coefficient matrices and auxiliary data structured required by the kernel routines of the finite element library.

4.7.4. Conclusion on adaptive schemes for scalar conservation laws

In the present chapter, we considered recovery-based error indicators and developed a hierarchical mesh adaptation algorithm which was adopted for stationary and time-dependent flow simulations. In all computations, the nodal generation number recursively defined in (4.6.2) was used to impose an upper bound on the maximum refinement level which was prescribed *a priori*. Three to five refinement levels were typically accepted in the computations performed for this thesis.

The usability of various recovery techniques was analyzed for the stationary convection-diffusion equation adopting a very small diffusion coefficient. All approaches succeeded in locally refining the mesh in the vicinity of steep fronts so as to reduce the amount of numerical diffusion and improve the accuracy of the approximate solution. The superconvergent patch recovery procedure is computationally more expensive than consistent L_2-projection and limited gradient averaging. Moreover, the SPR-technique tends to predict larger values for the gradient error so that cells were also refined in regions, where the flow was smooth or nearly constant. However, this did not reduce the amount of numerical diffusion and the resolution of the numerical approximation did not improve noticeably. Thus, the overhead cost of the superconvergent patch recovery

technique cannot be justified by more accurate solution profiles. The L_2-projection scheme and the even less expensive limited gradient averaging represent efficient alternatives for the design of recovery-based error indicators for convection dominated problems.

The employed grid re-coarsening algorithm was effective in removing excess elements in regions, where the gradient error was sufficiently small. The largest number of vertices/elements was typically obtained when the finest refinement level was reached for the first time. It is therefore advisable to invoke the mesh adaptation procedure successively until a 'converged' grid is recovered. In a practical implementation, fully converged solutions are required only on the final grid. Following Shapiro [236], the stopping criterion can be relaxed on all intermediate meshes, e.g., by using the square root of the parameter tol_{steady} as tolerance to monitor steady state convergence.

Hierarchical mesh adaptivity based on the red-green strategy [19] is applicable to triangulations consisting of triangles and/or quadrilaterals, whereby triangular transition elements may be introduced in the latter case. In our computations performed for the convection-diffusion problem, we did not observe numerical difficulties for hybrid meshes. The essential flow features were resolved either by pure linear or by pure bilinear finite elements, and moreover, the amount of artificial diffusion ($smear_{\text{int}}$) was similar to the reference values evaluated on structured grids.

Dynamic mesh adaptation was studied for the time-dependent continuity equation adopting both transient and constant velocity fields. The mesh refinement and re-coarsening algorithms succeeded in adjusting the computational mesh to local flow features so as to 'follow' steep fronts as time evolves. The simple strategy to refine elements in a protection layer of width two by way of precaution was sufficient for our purpose as long as mesh adaptation was performed frequently enough. Otherwise, essential flow features such as steep fronts are likely to drop out of locally refined mesh regions, and hence, they are irretrievably corrupted by excessive numerical diffusion.

The numerical results produced for the test problem 2 demonstrate that mesh adaptivity can be used to reduce the computing time of transient flow calculations without degrading the accuracy of numerical solutions. In particular, there is a very large gap between the most expensive solution strategy (defect correction scheme on a regular grid using a constant time step) and Newton's method combined with dynamic adjustment in time (evolutionary PID controller) and space (h-adaptivity). For this particular benchmark, the most efficient algorithm determined the numerical solution in only 6% of the worst case CPU time, whereby the accuracy deteriorated by $1-2\%$ which is negligibly in practice. In light of the above, the use of mesh adaptivity combined with efficient solution algorithms is beneficial to reduce the computational cost of transient simulations.

The potential of dynamic mesh adaptation in improving the resolution of complex flow fields was investigated in the third numerical example. The computational grids were fully adjusted to the spiral shape of the profile, and moreover, there was apparently no difference in the resolution of the approximate solution when using linear or bilinear finite elements. Admittedly, the separation of the unit square into a quadrilateral region and a triangular part left and right to the vertical line $x = 0.5$ is somehow artificial. In practical applications, the decision on triangles and quadrilaterals should be based on the facilities of the employed mesh generation software and/or on physically motivated criteria. It is commonly thought that rectangular elements are particularly useful to resolve boundary layers present in viscous flows. Hence, test problem 3 serves as a proof-of-concept to demonstrate that hierarchical mesh refinement based on the red-green strategy [19] is applicable to hybrid meshes. To our belief, the dynamic mesh adaptation procedure suggested in this thesis can be easily integrated into existing CFD software packages and it is suited for the treatment of complex flows which arise in real-world applications.

5

The compressible Euler equations

Unstructured grid finite element methods appear to be particularly attractive for the numerical treatment of aerodynamic applications governed by the compressible Euler equations [163, 172, 189]. Most of the algorithms currently in use are explicit and subject to the CFL condition. For hyperbolic systems, the local CFL number is given by the ratio of the maximum eigenvalue and the smallest mesh width multiplied by the time step. For highly unstructured and/or locally adapted grids, the largest admissible time step is subject to the local CFL number on the smallest cell unless local time-stepping schemes are employed. Such techniques are frequently combined with the pseudo time-stepping approach used for the simulation of stationary flows which do not call for temporal accuracy and mass conservation in each time step. In essence, the use of a common global time step is abandoned in favor of local steps at which each cell is individually marched to steady-state. Recently, local time-stepping techniques have been applied to space-time expansion discontinuous Galerkin (DG) schemes [171] for the two dimensional unsteady compressible Navier-Stokes equations. Apart from this particular algorithm, the use of different time steps at different mesh points may result in a loss of 'mass' and allow shocks to move at wrong speeds so that local time stepping is not feasible for transient flows in general.

In light of the above, there is a need for the development of *implicit* high-resolution finite element schemes for the above-mentioned class of CFD applications. On the one hand, fully implicit schemes are unconditionally stable, and thus, they can be operated at large time steps which makes them a favorable tool for the computation of steady-state flows. For time-dependent flows, the semi-implicit Crank-Nicolson discretization represents a viable alternative to the explicit Euler method which requires further stabilization. On the other hand, implicit schemes are more difficult to implement and their actual performance strongly depends on the choice of data structures, the configuration of linear and nonlinear solvers, the efficiency of the global matrix assembly and the interplay of all components. In this chapter, we present a Galerkin flux decomposition for the Euler equations. Roe's linearization of the Jacobian matrix is used to devise the high-order semi-discrete scheme which lends itself to an implicit time discretization. The concept of local extremum diminishing schemes is generalized to hyperbolic systems and a viable low-order method is constructed by elimination of negative eigenvalues of the Jacobian matrix. Different choices for the construction of artificial viscosities are discussed and the dimensional splitting approach is adopted to extend node-based flux limiting of TVD type to systems of hyperbolic conservation laws. Some strategies for the iterative solution of the nonlinear algebraic systems of equations are reviewed. The implementation of boundary conditions is addressed and a unified predictor-corrector algorithm applicable to characteristic as well as wall boundary conditions is suggested. The grid adaptation algorithm introduced in the previous chapter is generalized to hyperbolic systems. The developed numerical methods are applied to some two-dimensional benchmark problems for the compressible Euler equations to assess their usability.

5.1. Governing equations

The Euler equations of gas dynamics represent a system of conservation laws for the mass, momentum, and total energy of an ideal, inviscid, compressible fluid [105]:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) &= 0 \\ \frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v}) + \nabla p &= 0 \\ \frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho H \mathbf{v}) &= 0 \end{aligned} \tag{5.1.1}$$

Here, the scalar quantity ρ denotes the fluid density, p is the thermodynamic pressure and $\mathbf{v}$ represents the vector-valued flow velocity. The total energy E is assembled from the internal energy e, which is not a conserved quantity [64], augmented by the specific kinetic energy, that is

$$E = e + \frac{|\mathbf{v}|^2}{2}, \qquad |\mathbf{v}|^2 = v_1^2 + v_2^2 + v_3^2 \tag{5.1.2}$$

Moreover, the specific enthalpy h and the stagnation, or total enthalpy H are given by

$$h = e + \frac{p}{\rho}, \qquad H = E + \frac{p}{\rho} \tag{5.1.3}$$

The first-order system (5.1.1) of partial differential equations is a simplification of the more realistic compressible Navier-Stokes equations, whereby the effects of body forces, viscous stresses and heat conductivity are neglected [104]. Note that the above system is under-determined since it has less equations than variables. In order to solve the compressible Euler equations additional equations of state are required to relate the pressure to the conserved variables. In this work, only perfect gases are considered which leads to a single equation of state for the pressure.

This section is concerned with analytical aspects of the Euler equations. In addition to the divergence form, we will present the quasi-linear form of the equations (5.1.1) which is appropriate for the design of numerical methods. Some important properties such as the hyperbolicity of the Jacobian matrix are reviewed. The reader who is already familiar with the theory of inviscid flows may want to skip this paragraph and continue reading at Section 5.2 which deals with the construction of numerical methods for hyperbolic systems of conservation laws.

5.1.1. Equations of state

Let the fluid of interest be an ideal gas, that is, it is required to obey the thermal equation of state

$$p = \rho R T \tag{5.1.4}$$

where T denotes the total temperature measured in Kelvin. The specific gas constant R differs for different species, and a typical value for air at sea-level is $R = 287 N \cdot m/kg \cdot K$. Moreover, let the fluid of interest satisfy the caloric equation of state which is given by

$$e = c_v T \tag{5.1.5}$$

where c_v denotes the specific heat at constant volume. Making use of the specific enthalpy defined in (5.1.3), a similar expression for the specific heat at constant pressure reads as follows

$$h = c_p T \tag{5.1.6}$$

For thermally perfect gases which additionally satisfy the ideal gas law (5.1.4), internal energy e and enthalpy h are functions of temperature T alone and the specific heats are constant. Combining the three relations (5.1.4)–(5.1.6) leads to the definition of so-called perfect gases for which

$$R = c_p - c_v \tag{5.1.7}$$

Substitution into the ideal gas law (5.1.4) and elimination of the temperature T with aid of the caloric equation of state (5.1.5) yields the following constitutive relation for the pressure [105]

$$p = (\gamma-1)\rho e = (\gamma-1)\rho\left(E - \frac{|\mathbf{v}|^2}{2}\right) \tag{5.1.8}$$

Here, $\gamma = c_p/c_v$ is the ratio of specific heats which in general satisfies $1 < \gamma \leq 5/3$. In particular, $c_v = 717N \cdot m/kg \cdot K$ and $c_p = 1004N \cdot m/kg \cdot K$ for air at sea-level so that $\gamma = 1.4$. All other gases not satisfying equations (5.1.4) and (5.1.5) are known as real gases which may be either dense or rarefied. For perfect gases microscopic particles are assumed to interact only upon direct collision. In contrast, molecules are close enough in dense gases so that intermolecular forces play an important role. On the other hand, the number of microscopic particles per unit volume in rarefied gases is very small, so that the statistical average used for perfect gases cannot be applied. We will only consider perfect gases in this work, and in particular, air at sea-level is adopted.

In addition to conservation laws, any physical system must satisfy the second law of thermodynamics which states, that the total entropy of the universe never decreases. This fundamental quantity measures the 'disorder' of the thermodynamic system. Roughly speaking, the thermodynamic equations presented above describe the average state of microscopic particles, whereas entropy measures the deviation from the average. An equation of state for the specific entropy s is

$$s = c_v \log p - c_p \log \rho + const \tag{5.1.9}$$

which is valid for all perfect gases. The exact value of the additive constant is usually unobtainable and at the same time unimportant. Zero entropy corresponds to perfect knowledge of the microscopic particles but increasing entropy goes along with the increase of uncertainty about the microscopics. From the practical point of view, the most important property of entropy is given by the fact that for smooth flows its value remains constant along particle paths and changes to a higher value if particles are crossing a shock. This is a natural consequence of neglecting viscous phenomena and other sources of entropy generation in the Euler equations except for shocks [149].

5.1.2. Mach number and speed of sound

For an isentropic flow which exhibits no entropy variation, the speed of sound is defined as [251]

$$c = \sqrt{\frac{\partial p}{\partial \rho}\Big|_s} \tag{5.1.10}$$

where c represents the speed at which small disturbances, i.e., acoustic waves, propagate through a fluid measured relative to the motion of the flow, whereby the entropy s is kept fixed. Note that in general the relation for the speed of sound is nonlinear due to the fact that the equation of state for the pressure typically depends on density and internal energy, i.e. $p = p(\rho, e)$. However, for perfect gases the thermal equation of state (5.1.4) implies that [251]

$$c = \sqrt{\gamma RT} = \sqrt{\frac{\gamma p}{\rho}} \tag{5.1.11}$$

Using the state equation for perfect gases (5.1.8) and definition (5.1.3) the speed of sound can also be written in terms of total energy/enthalpy and kinetic energy, respectively

$$c^2 = \gamma(\gamma-1)\left(E - \frac{|\mathbf{v}|^2}{2}\right), \qquad c^2 = (\gamma-1)\left(H - \frac{|\mathbf{v}|^2}{2}\right) \tag{5.1.12}$$

The compressibility effect can be analyzed by considering the dimensionless Mach number

$$M = \frac{|\mathbf{v}|}{c} \tag{5.1.13}$$

which relates the modulus of the flow velocity to the speed of sound. In (nearly) incompressible fluids, sound waves travel at infinite speed ($c \to \infty$) so that the Mach number tends to zero. On the other hand, it can be used to classify subsonic ($0 < M < 1$), transonic ($M \approx 1$), and supersonic ($M > 1$) compressible flows. Hypersonic flows ($M \gg 1$) give rise to chemical reactions which cannot be modeled by the Euler equations due to the lack of heat conductive effects.

5.1.3. Vector notation

For further analysis, it is expedient to represent the system of compressible Euler equations (5.1.1) in a compact vector notation. Let $U = U(\mathbf{x},t)$ be the column vector of conservative variables which depend both on the spatial coordinates $\mathbf{x} \in \Omega$ and time $t \in \mathbb{R}_0^+$, whereby $\mathbb{R}_0^+$ denotes the set of non-negative real numbers. Without loss of generality, the computational domain is embedded into the Cartesian space, i.e. $\Omega \subset \mathbb{R}^{N_\mathrm{D}}$. Mathematically speaking, the vector of conservative variables $U = [U_1, \ldots, U_{N_\mathrm{U}}]^T$ is given by the following function [105]

$$U = \begin{bmatrix} \rho \\ \rho\,\mathbf{v} \\ \rho\,E \end{bmatrix} \qquad U(\mathbf{x},t) : \mathbb{R}^{N_\mathrm{D}} \times \mathbb{R}_0^+ \to \mathbb{R}^{N_\mathrm{U}} \tag{5.1.14}$$

Here, $N_\mathrm{D} \in \{1,2,3\}$ denotes the number of space dimensions and $N_\mathrm{U} \in \mathbb{N} \setminus \{0\}$ represents the number of state variables, whereby the relation $N_\mathrm{U} = N_\mathrm{D} + 2$ holds for the Euler equations.

The initial conditions of the state vector at time $t = 0$ is typically given as a function of space alone, i.e. $U_0 = U(\mathbf{x},0) : \mathbb{R}^{N_\mathrm{D}} \to \mathbb{R}^{N_\mathrm{U}}$. The divergence terms present in the system of compressible Euler equations (5.1.1) can be written uniformly in terms of the multi-component inviscid flux function $\mathbf{F} = \mathbf{F}(U) : \mathbb{R}^{N_\mathrm{U}} \to \mathbb{R}^{N_\mathrm{U}}$. In three space dimensions it is given by the triple of fluxes $\mathbf{F} = (F^1, F^2, F^3)$ for each direction of the Cartesian coordinate system [105]

$$\mathbf{F} = \begin{bmatrix} \rho\,\mathbf{v} \\ \rho\,\mathbf{v}\otimes\mathbf{v} + p\underline{\underline{I}} \\ \rho\,H\mathbf{v} \end{bmatrix} \qquad F^d = \begin{bmatrix} \rho v_d \\ \rho\mathbf{v}v_d + p\underline{\underline{I}} \\ \rho H v_d \end{bmatrix} \tag{5.1.15}$$

Thus, the Cauchy problem for the Euler equations (5.1.1) in divergence form reads as follows

$$\begin{aligned} \frac{\partial U}{\partial t} + \nabla\cdot\mathbf{F} &= 0 \qquad \text{in} \quad \mathbb{R}^{N_\mathrm{D}} \times \mathbb{R}_0^+ \\ U(\mathbf{x},0) &= U_0(\mathbf{x}) \qquad \text{in} \quad \mathbb{R}^{N_\mathrm{D}} \end{aligned} \qquad \nabla\cdot\mathbf{F} := \sum_{d=1}^{N_\mathrm{D}} \frac{\partial F^d}{\partial x_d} \tag{5.1.16}$$

Whenever possible, the value $N_\mathrm{D} = 3$ will be adopted so as to keep the presentation most general and restriction to one or two space dimensions will be explicitly indicated. Finally, the under-determined system (5.1.16) is closed by the equation of state for a perfect gas (5.1.8).

5.1.4. Quasi-linear form of equations

For the design of numerical methods it is convenient if the spatial derivatives are applied directly to the conservative variables rather than the inviscid fluxes. This can be accomplished by making use of the chain rule so as to obtain the quasi-linear form of the Euler equations

$$\frac{\partial U}{\partial t} + \mathbf{A}\cdot\nabla U = 0, \qquad \mathbf{A}\cdot\nabla U = \sum_{d=1}^{N_D} A^d \frac{\partial U}{\partial x_d}, \qquad A^d(U) = \frac{\partial F^d}{\partial U} \tag{5.1.17}$$

Here, $\mathbf{A} = (A^1, A^2, A^3)$ denotes the triple of Jacobian matrices which can be found in any textbook on gas dynamics [105], and for the two-dimensional case in the Appendix C.

We would like to emphasize the fact that application of the chain rule requires both functions $\mathbf{F}(U)$ and $U(\mathbf{x},t)$ to be differentiable. Nonlinear conservation laws give rise to the formation of shock waves and other discontinuities so that their solution can only be defined in a weak sense (see Section 2.2.2). In essence, integration by parts is employed to shift the derivatives to some properly chosen test function, so that the solution U and the fluxes $\mathbf{F}(U)$ only need to be integrable.

In light of the above, it is desirable to relate the triple of inviscid fluxes to the values of the solution vector directly without resorting to the chain rule. For the compressible Euler equations, this can be achived by using the homogeneity property of the flux function.

Theorem 5.1.1 (Homogeneity Property [82])**.** Each component of the flux function $\mathbf{F} = (F^1, F^2, F^3)$ is a homogeneous function of degree one, that is, for an arbitrary constant $\alpha \in \mathbb{R}$ it satisfies

$$F^d(\alpha U) = \alpha F^d(U), \qquad d = 1,2,3 \tag{5.1.18}$$

Proof. See Godlewski and Raviart [82]; Lemma 5.1. □

Let equation (5.1.18) be differentiated with respect to α and set $\alpha = 1$ which yields

$$F^d(U) = A^d(U)U, \qquad A^d(U) = \frac{\partial F^d}{\partial U}, \qquad d = 1,2,3 \tag{5.1.19}$$

The above relation provides a valuable tool for the design of numerical methods based on the flux vector splitting approach. The homogeneity property is also satisfied by the Euler equations with an equation of state that is slightly more general than (5.1.8) only valid for perfect gases [240].

5.1.5. Hyperbolicity of equations

The hyperbolicity of the Euler equations of gas dynamics is an important feature which is frequently employed in the design of numerical solution algorithms. To introduce the basic ideas consider the one-dimensional Euler equations in quasi-linear form

$$\frac{\partial U}{\partial t} + A(U)\frac{\partial U}{\partial x} = 0, \qquad A(U) = \frac{\partial F}{\partial U} \tag{5.1.20}$$

whereby the solution U is assumed to be differentiable. The Jacobian matrix is given by [154]

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)v^2 & (3-\gamma)v & \gamma-1 \\ \frac{1}{2}(\gamma-1)v^3 - vH & H-(\gamma-1)v^2 & \gamma v \end{bmatrix} \tag{5.1.21}$$

where v is the fluid velocity and $H = E + p/\rho$ denotes the total enthalpy defined in (5.1.3). The diagonal matrix of eigenvalues of the Jacobian A reads as follows

$$\Lambda = \mathrm{diag}\{v-c, v, v+c\} \tag{5.1.22}$$

where $c = \sqrt{\gamma p/\rho}$ is the local speed of sound which is always greater than zero. As a consequence, all eigenvalues are real and, moreover, they are distinct. This leads directly to the following

Definition 5.1.1 (Hyperbolic system [251])**.** A system of the form (5.1.20) is said to be hyperbolic if for all vectors U, the coefficient matrix $A(U)$ has three real eigenvalues

$$\lambda_1(U) \leq \lambda_2(U) \leq \lambda_3(U) \tag{5.1.23}$$

with a complete family of right eigenvectors $\mathbf{r}_k = \mathbf{r}_k(U)$, $k = 1,2,3$. Moreover, the system is called strictly hyperbolic if all eigenvalues are distinct for all possible values of U.

The one-dimensional Euler equations are strictly hyperbolic which can be seen from the matrix of eigenvalues (5.1.22) and the family of right eigenvectors stored columnwise [140]

$$R = \begin{bmatrix} 1 & 1 & 1 \\ v-c & v & v+c \\ H-vc & \frac{1}{2}v^2 & H+vc \end{bmatrix} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3] \tag{5.1.24}$$

In contrast, the rows of its inverse correspond to the left eigenvectors given by [140]

$$R^{-1} = \begin{bmatrix} \frac{1}{2}(b_1 + \frac{v}{c}) & \frac{1}{2}(-b_2 v - \frac{1}{c}) & \frac{1}{2}b_2 \\ 1-b_1 & b_2 v & -b_2 \\ \frac{1}{2}(b_1 - \frac{v}{c}) & \frac{1}{2}(-b_2 v + \frac{1}{c}) & \frac{1}{2}b_2 \end{bmatrix} = \begin{bmatrix} \mathbf{l}_1 \\ \mathbf{l}_2 \\ \mathbf{l}_3 \end{bmatrix} \tag{5.1.25}$$

where the auxiliary coefficients b_1 and b_2 are defined according to

$$b_1 = \frac{b_2 v^2}{2}, \qquad b_2 = \frac{\gamma - 1}{c^2} \tag{5.1.26}$$

From the physical point of view, the eigenvalues represent the speed at which information propagates through the flow field. The hyperbolicity of the one-dimensional Euler equations can be equivalently stated as follows [157]: *A system of the form (5.1.20) is said to be hyperbolic if for all vectors U, there exists a real diagonal matrix $\Lambda(U)$ and a real non-singular matrix $R(U)$ such that $A(U)R(U) = \Lambda(U)R(U)$ hold.* Here, the diagonal matrix Λ represents the set of real eigenvalues of A and the columns of R comprise the family of right eigenvectors (5.1.24). Due to the non-singularity of the latter one, there exists a characteristic decomposition of the form

$$A = R\Lambda R^{-1} \tag{5.1.27}$$

The wave-like solution behavior of hyperbolic systems can be studied by substituting the decomposition (5.1.27) into the quasi-linear form (5.1.20) and multiplying the resulting system by R^{-1}

$$R^{-1}\frac{\partial U}{\partial t} + \Lambda R^{-1}\frac{\partial U}{\partial x} = 0 \tag{5.1.28}$$

This leads to the definition [105] of a new set of characteristic variables $\delta W = [\delta w_1, \delta w_2, \delta w_3]^T$

$$\delta W = R^{-1}\delta U \tag{5.1.29}$$

where δ denotes an arbitrary variation either ∂_t or ∂_x. In other words, the increments δW can be expressed as a linear combination of the increments of the conservative variables δU with coefficients equal to the left eigenvectors (5.1.25). On the other hand, the solution δU to the original

problem can be decomposed into simple waves which are described by the right eigenvectors $\mathbf{r}_k$ with amplitudes equal to the characteristic components δw_k [105]

$$\delta U = \sum_{k=1}^{3} \delta w_k \mathbf{r}_k \tag{5.1.30}$$

In general, the left and right eigenvectors are functions of the flow variables so that the coefficients in the above expansion are not constant. For the time being, assume that the coefficient matrix A is constant so that the one-dimensional Euler can be transformted into the canonical form [251]

$$\frac{\partial W}{\partial t} + \Lambda \frac{\partial W}{\partial x} = 0, \qquad W := R^{-1} U \tag{5.1.31}$$

For an isentropic flow (i.e. no variation of entropy occurs) of a polytropic ideal gas the characteristic variables W which are also called Riemann variables read as follows [105]:

$$W = \left[v - \frac{2c}{\gamma - 1}, s, v + \frac{2c}{\gamma - 1} \right]^T \tag{5.1.32}$$

Here, s denotes the entropy per unit mass of fluid defined in (5.1.9). Since the matrix of eigenvalues Λ is diagonal, system (5.1.31) yields a sequence of decoupled scalar convection equations

$$\frac{\partial w_k}{\partial t} + \lambda_k \frac{\partial w_k}{\partial x} = 0, \qquad k = 1,2,3 \tag{5.1.33}$$

which state that the quantity w_k propagates along the corresponding characteristic with constant speed λ_k. The exact solution is constant along the straight lines satisfying the ODEs

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \lambda_k, \qquad \text{where} \quad \lambda_k = \text{const} \tag{5.1.34}$$

In particular, entropy is transported along the path line of the fluid and it is conserved along the characteristic $\mathrm{d}x/\mathrm{d}t = v$ in the absence of discontinuities. The quantities $v \pm 2c/(\gamma - 1)$ travel at speeds $v \pm c$ along the so-called Mach lines defined by $\mathrm{d}x/\mathrm{d}t = v \pm c$.

The solution to the original problem can be recovered as $U = RW$, where the Riemann variables w_k are the coefficients of the right eigenvectors $\mathbf{r}_k$ in the following decomposition

$$U = \sum_{k=1}^{3} w_k \mathbf{r}_k, \qquad w_k(x,t) = w_k(x - \lambda_k t, 0) \tag{5.1.35}$$

which depends only on the initial data [154]. It is important to note that general nonlinear hyperbolic systems give rise to non-constant matrices, and hence, the eigenvalues and eigenvectors of the Jacobian matrix A vary in space and time. Consequently, Riemann variables W can only be defined locally for a 'frozen' state of the unknown solution U. It is thus that the characteristics associated with the Euler equations are not in general straight lines but curves in the x,t-plane

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v - c, \qquad \frac{\mathrm{d}x}{\mathrm{d}t} = v, \qquad \frac{\mathrm{d}x}{\mathrm{d}t} = v + c \tag{5.1.36}$$

Moreover, they do not transport constant values of the unknowns unless the variation of the characteristic variables equals zero. If the characteristic variables W remain constant, then they are termed Riemann invariants [105]. These one-dimensional concepts turn out to be useful for the design of numerical methods and the specification of boundary conditions in multidimensions.

The concept of hyperbolicity can be extended to multidimensions by restriction to a prescribed direction [105]. Then it suffices to consider the eigenstructure of the associated Jacobian matrix which results from a suitable linear combination of uni-directional Jacobians $\mathbf{A} = (A^1, A^2, A^3)$.

Definition 5.1.2 (Hyperbolic system [72])**.** A system of the form (5.1.17) is termed hyperbolic if for all admissible vectors U and all 'directions' $\mathbf{e} \in \mathbb{R}^{N_\mathrm{D}}$, $|\mathbf{e}| = 1$ the linear combination

$$\mathbf{A}(U,\mathbf{e}) = \sum_{d=1}^{N_\mathrm{D}} e_d A^d(U) \tag{5.1.37}$$

of the uni-directional Jacobian matrices A^d, $\forall d = 1,\ldots,N_\mathrm{D}$ has real (possibly multiple) eigenvalues $\lambda_k = \lambda_k(U,\mathbf{e}), \forall k = 1,\ldots,N_\mathrm{U}$ and a complete family of right eigenvectors $\mathbf{r}_k = \mathbf{r}_k(U,\mathbf{e})$. Moreover, the system is termed strictly hyperbolic if the eigenvalues are distinct for all U.

The above property is also known as hyperbolicity in time [105, 251]. The multidimensional Euler equations (5.1.16) are hyperbolic in time [105], which can be shown by proving an equivalent formulation of the above definition. In particular, it suffices to show that any linear combination of the three uni-directional Jacobian matrices is diagonalizable with real eigenvalues

$$\mathbf{A}(U,\mathbf{e}) = R(U,\mathbf{e})\,\Lambda(U,\mathbf{e})\,[R(U,\mathbf{e})]^{-1} \tag{5.1.38}$$

Here, $\Lambda(U,\mathbf{e})$ is the diagonal matrix of real (but not distinct) eigenvalues and the matrix of right eigenvectors $R(U,\mathbf{e})$ is non-singular. As in the one-dimensional case, its inverse is composed of the left eigenvectors. Analytical expressions for all matrices in two space dimensions are given in Appendix C, and their three-dimensional counterparts can be found in many textbooks on gas dynamics, e.g. [105, 225]. Owing to the fact that the Euler equations exhibit multiple eigenvalues

$$\Lambda(U,\mathbf{e}) = \mathrm{diag}\{\mathbf{v}\cdot\mathbf{e} - c, \mathbf{v}\cdot\mathbf{e}, \mathbf{v}\cdot\mathbf{e} + c, \mathbf{v}\cdot\mathbf{e}, \mathbf{v}\cdot\mathbf{e}\} \tag{5.1.39}$$

the strict hyperbolicity is lost in multidimensions. Unfortunately, the uni-directional Jacobians do not commute, and therefore, they have different eigenvalues [105]. Consequently, it is impossible to diagonalize all A^d simultaneously while this can be done for any linear combination.

The characteristic formulation of the multidimensional Euler equations is not unique, that is, there exists infinitely many wave descriptions of a given flow in multidimensions [149]. Moreover, waves may have any dimension between 1 and N_D in the $\mathbf{x},t$-plane, and they can travel in an infinite number of directions. For a detailed characteristic analysis the interested reader is referred to the textbook by Hirsch [105] and the work by Kröger in characteristic theory [127].

5.1.6. Properties of the Euler equations

Let us summarize the main results presented in this section. The compressible Euler equations (5.1.1) represent a system of conservation laws for the mass, momentum and energy, which needs to be equipped with an equation of state. In this thesis, the fluid is assumed to be a perfect, that is, it is an ideal gas that is thermally perfect, so that equation (5.1.8) can be employed. The governing equation were cast into compact vector notation and the resulting first-order system (5.1.16) can be transformed into an equivalent quasi-linear formulation (5.1.17) which relates the Jacobian tensor $\mathbf{A}$ to the derivative of the state vector U. The one-dimensional Euler equations are strictly hyperbolic which is due to the fact that all eigenvalues are distinct. The canonical form can be derived by transition to the characteristic variables so as to decouple the one-dimensional Euler equations into a sequence of scalar transport equations for individial waves. For a constant Jacobian A, all characteristics are straight lines and the exact solution can be determined as the superposition of simple waves. For nonlinear hyperbolic systems, the characteristics are curves in the space-time domain so that Riemann variables can be only defined locally. In multidimensions, each linear combination of uni-directional Jacobians is diagonalizable with real eigenvalues. On the other hand, the individual matrices do not commute so that it is impossible to diagonalize all Jacobians simultaneously. Moreover, $\lambda_k = \mathbf{v}\cdot\mathbf{e}$ is a multiple eigenvalue so that the *strict* hyperbolicity is lost.

5.2. High-resolution schemes for the Euler equations

This section deals with the application of algebraic flux correction schemes [133, 144, 145] to the equations of gas dynamics [142]. The presentation of this material follows closely that in [140]. Implicit finite element methods are still rarely used in compressible flow simulations, and therefore, the construction of the global Jacobian matrix is addressed in detail. In particular, an efficient edge-based assembly procedure is presented which is applicable to general Galerkin finite element schemes. A suitable low-order method is constructed by adding a tensor of artificial diffusion which is designed to eliminate negative eigenvalues from the off-diagonal blocks [140]. Different approaches for the construction of artificial viscosities are discussed and dimensional splitting is adopted to employ characteristic flux limiting of TVD type as proposed by Kuzmin [135]. Some strategies for the iterative solution of nonlinear systems of equation are reviewed and a segregated approach is promoted for moderately small time steps.

5.2.1. High-order scheme

Consider the weighted residual formulation of the compressible Euler equations (5.1.16)

$$\int_\Omega W\left[\frac{\partial U}{\partial t}+\nabla\cdot\mathbf{F}(U)\right]\mathrm{d}\mathbf{x}=0 \tag{5.2.1}$$

where W denotes the vector-valued weighting function. A common practice in the simulation of conservation laws is to employ the group representation proposed by Fletcher [78]

$$\mathrm{U}_h(\mathbf{x},t)=\sum_{j=1}^{M}\mathrm{U}_j(t)\varphi_j(\mathbf{x}),\qquad \mathbf{F}_h(\mathbf{x},t)=\sum_{j=1}^{M}\mathbf{F}_j(t)\varphi_j(\mathbf{x}) \tag{5.2.2}$$

which yields a system of semi-discretized equations for the time-dependent nodal values

$$\sum_{j=1}^{M}\left[\int_\Omega\varphi_i\varphi_j\,\mathrm{d}\mathbf{x}\right]\frac{\mathrm{d}\mathrm{U}_j}{\mathrm{d}t}+\sum_{j=1}^{M}\left[\int_\Omega\varphi_i\nabla\varphi_j\,\mathrm{d}\mathbf{x}\right]\cdot\mathbf{F}_j=0 \tag{5.2.3}$$

For the numerical treatment, it is expedient to eliminate the dependent variables $\mathbf{F}_j$ in favor of the unknows U_j which yields a DAE system for the vector of time-dependent nodal values

$$\mathrm{M}_C\frac{\mathrm{d}\mathrm{U}}{\mathrm{d}t}=\mathrm{K}\,\mathrm{U} \tag{5.2.4}$$

Here, M_C is the block-diagonal mass matrix and K denotes a discrete counterpart of the operator $-\mathbf{A}\cdot\nabla$ from (5.1.17). A viable strategy applicable to (5.2.3) was suggested by Kuzmin et al. [142].

Consider the coefficient matrices $\mathbf{C}=\{\mathbf{c}_{ij}\}$ which account for spatial derivatives and features zero row sums so that its diagonal entries can be expressed in terms of the off-diagonal ones:

$$\mathbf{c}_{ij}=\int_\Omega\varphi_i\nabla\varphi_j\,\mathrm{d}\mathbf{x},\qquad \mathbf{c}_{ii}=-\sum_{j\neq i}\mathbf{c}_{ij} \tag{5.2.5}$$

Substitution of the above expression into the semi-discrete formulation (5.2.3) yields [142]

$$\sum_{j=1}^{M}\mathrm{M}_{ij}\frac{\mathrm{d}\mathrm{U}_j}{\mathrm{d}t}+\sum_{j\neq i}\mathbf{c}_{ij}\cdot(\mathbf{F}_j-\mathbf{F}_i)=0 \tag{5.2.6}$$

where the block $\mathrm{M}_{ij}=m_{ij}\mathrm{I}$ contains the coefficients $m_{ij}=\int_\Omega\varphi_i\varphi_j\,\mathrm{d}\mathbf{x}$ of the consistent mass matrix $M_C=\{m_{ij}\}$ and the dimension of the identity operator I equals the number of state variables.

Hence, the right-hand side of the semi-discrete system (5.2.4) can be constructed in a loop over the edges of the sparsity pattern, whereby edge *ij* gives rise to the following contribution [142]:

$$(\mathrm{KU})_i \quad \longleftarrow \quad \mathbf{c}_{ij} \cdot (\mathbf{F}_i - \mathbf{F}_j) \tag{5.2.7}$$

$$(\mathrm{KU})_j \quad \longleftarrow \quad \mathbf{c}_{ji} \cdot (\mathbf{F}_j - \mathbf{F}_i) \tag{5.2.8}$$

The triple of coefficient matrices $\mathbf{C} = (C^1, C^2, C^2)$ engendered by first-order space derivatives can be assembled once and for all at the beginning of the simulation and needs to be updated each time the mesh is modified. Thus, the group finite element formulation [78] allows for an assembly of the right-hand side vector without resorting to costly numerical integration.

Since we are mainly interested in implicit time discretizations, the global operator K (or parts of it) is required explicitly rather than its application to the vector of unknown solution values. In his pioneering work on approximate Riemann solver [224], Roe presented a linearization strategy which can be used to express the jump of fluxes in terms of solution differences multiplied by the linearized Jacobian matrix. In particular, any linear combination of flux differences can be rewritten as the solution jumps multiplied by the so-called cumulative Roe matrix [142, 143]

$$\mathbf{e} \cdot (\mathbf{F}_j - \mathbf{F}_i) = \mathbf{e} \cdot \tilde{\mathbf{A}}_{ij} (\mathrm{U}_j - \mathrm{U}_i) \tag{5.2.9}$$

In the above expression, the vector $\mathbf{e} = (e_1, e_2, e_3)$ is an arbitrary 'direction' in the three-dimensional space. The operator $\tilde{\mathbf{A}}_{ij}$ denotes the triple of Jacobian matrices $\mathbf{A} = (A^1, A^2, A^3)$ evaluated at the density averaged Roe mean values which are given by the following expressions [224]

$$\tilde{\mathbf{v}}_{ij} = \frac{\sqrt{\rho_i}\mathbf{v}_i + \sqrt{\rho_j}\mathbf{v}_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}, \qquad \tilde{H}_{ij} = \frac{\sqrt{\rho_i}H_i + \sqrt{\rho_j}H_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}, \qquad \tilde{c}_{ij} = \sqrt{(\gamma - 1)\left(\tilde{H}_{ij} - \frac{|\tilde{\mathbf{v}}_{ij}^2|}{2}\right)} \tag{5.2.10}$$

An explicit value for the density is not required for computing the Jacobian matrices and the average $\tilde{\rho}_{ij} = \sqrt{\rho_i \rho_j}$ can be used otherwise [251]. By virtue of this linearization, the edge contributions (5.2.7) and (5.2.8) can be rewritten as a combination of unidirectional Roe matrices multiplied by the difference between the values of the conservative variables at nodes *i* and *j* [142]:

$$(\mathrm{KU})_i \quad \longleftarrow \quad \mathbf{c}_{ij} \cdot \tilde{\mathbf{A}}_{ij} (\mathrm{U}_i - \mathrm{U}_j) \tag{5.2.11}$$

$$(\mathrm{KU})_j \quad \longleftarrow \quad \mathbf{c}_{ji} \cdot \tilde{\mathbf{A}}_{ij} (\mathrm{U}_j - \mathrm{U}_i) \tag{5.2.12}$$

For our purpose it makes sense to split the coefficients that correspond to the discretized spatial derivatives into their symmetric and skew-symmetric parts [142, 143]

$$\mathbf{c}_{ij} = \mathbf{a}_{ij} + \mathbf{b}_{ij}, \qquad \mathbf{a}_{ij} := \frac{\mathbf{c}_{ij} - \mathbf{c}_{ji}}{2} = -\mathbf{a}_{ji}, \qquad \mathbf{b}_{ij} := \frac{\mathbf{c}_{ij} + \mathbf{c}_{ji}}{2} = \mathbf{b}_{ji} \tag{5.2.13}$$

Integration by parts applied to the volume integral in (5.2.5) reveals that [163]

$$\mathbf{c}_{ij} = -\mathbf{c}_{ji} + \mathbf{s}_{ij}, \qquad \mathbf{s}_{ij} = \int_\Gamma \varphi_i \varphi_j \mathbf{n} \, \mathrm{d}s \tag{5.2.14}$$

whereby $\mathbf{s}_{ij}$ is an entry of the 'mass matrix' for the surface triangulation. It is symmetric and vanishes unless both nodes are located at the boundary so that $\mathbf{b}_{ij} = \frac{1}{2}\mathbf{s}_{ij}$ equals zero in the interior. Furthermore, coefficient $\mathbf{a}_{ij} = \mathbf{c}_{ij}$ for an interior edge *ij* and it can be evaluated from expression (5.2.13) otherwise. It is instructive to apply splitting (5.2.13) to the cumulative Roe matrices [142]

$$\mathrm{A}_{ij} = \mathbf{a}_{ij} \cdot \tilde{\mathbf{A}}_{ij}, \qquad \mathrm{B}_{ij} = \mathbf{b}_{ij} \cdot \tilde{\mathbf{A}}_{ij} \tag{5.2.15}$$

whereby the symmetric contribution B_{ij} is only needed at the boundary and vanishes in the interior. Thus, only the skew-symmetric part A_{ij} of the averaged Jacobians is to be evaluated unless edge ij is located at the boundary. In general, its contribution to the right-hand side of (5.2.4) reads [142]

$$(\mathrm{KU})_i \quad \longleftarrow \quad (\mathrm{A}_{ij}+\mathrm{B}_{ij})(\mathrm{U}_i-\mathrm{U}_j) \tag{5.2.16}$$

$$(\mathrm{KU})_j \quad \longleftarrow \quad (\mathrm{A}_{ij}-\mathrm{B}_{ij})(\mathrm{U}_i-\mathrm{U}_j) \tag{5.2.17}$$

This representation leads to a very efficient edge-based algorithm for matrix assembly. The coefficients $\mathbf{a}_{ij}$ and $\mathbf{b}_{ij}$ can be computed and store once and for all at the beginning of the simulation. They depend solely on the underlying mesh and on the type of approximation so that an update is only required after the grid has been modified. The global operator K can be evaluated without resorting to costly numerical integration. It is initialized by zeros and updated in a loop over edges of the sparsity graph making use of the local Jacobians (5.2.15) (see Kuzmin et at. [142]):

$$\begin{aligned} \mathrm{K}_{ii} &:= \mathrm{K}_{ii} + \mathrm{A}_{ij} + \mathrm{B}_{ij}, \qquad & \mathrm{K}_{ij} &:= -\mathrm{A}_{ij} - \mathrm{B}_{ij} \\ \mathrm{K}_{jj} &:= \mathrm{K}_{jj} - \mathrm{A}_{ij} + \mathrm{B}_{ij}, \qquad & \mathrm{K}_{ji} &:= \quad \mathrm{A}_{ij} - \mathrm{B}_{ij} \end{aligned} \tag{5.2.18}$$

In particular, the local blocks A_{ij} and B_{ij} (if required) are evaluated edge-by-edge based on the Roe mean values (5.2.10) and the precomputed coefficients (5.2.13). Their entries are scattered individual to the corresponding positions in the Jacobian operator adopting the correct signs from formula (5.2.18). A general framework for edge-based algorithms and some useful data structures for storing large sparse block matrices are discussed in Appendix B. It is worth mentioning that it is also possible to assemble only parts of the global operator and avoid explicit matrix-vector multiplications by evaluating KU edge-by-edge as described in expressions (5.2.16)–(5.2.17).

5.2.2. Low-order scheme

To a large extent, the ability of a high-resolution scheme to withstand the formation of wiggles depends on the quality of the underlying low-order method. For scalar conservation laws, the construction of a local extremum diminishing scheme can be based on two main principles [144]: Row-sum mass lumping is performed in order to remove antidiffusion from the consistent mass matrix and discrete upwinding amounts to a conservative elimination of negative off-diagonal entries from the high-order transport operator. The first strategy can be readily applied whereas a generalization of the LED principle to hyperbolic systems is required [142]. Let row-sum mass lumping be performed in the Galerkin discretization (5.2.4) which is replaced by

$$\mathrm{M}_L \frac{\mathrm{dU}}{\mathrm{d}t} = \mathrm{L}\,\mathrm{U} \tag{5.2.19}$$

Here, M_L is the lumped mass matrix and L represents the low-order Jacobian operator [142] to be defined below. Jameson's LED criterion 3.1.2 for scalar equations relates the $L_\infty-$stability of a numerical scheme (3.1.26) to the sign of the off-diagonal coefficients [114]. It provides a good starting point for the construction of non-oscillatory discretizations for systems of conservation laws, where the edge contributions to the global matrix L are no longer scalar quantities but local tensors $\mathrm{L}_{ij} \in \mathbb{R}^{N_\mathrm{U}\times N_\mathrm{U}}$. Consider the semi-discretized equation for the solution value at node i [140]

$$\mathrm{M}_i \frac{\mathrm{dU}_i}{\mathrm{d}t} = \sum_{j\neq i} \mathrm{L}_{ij}(\mathrm{U}_j - \mathrm{U}_i) \tag{5.2.20}$$

where $\mathrm{M}_i = m_i\mathrm{I}$ denotes the coefficients of the lumped mass matrix $m_i = \sum_j m_{ij} > 0$ multiplied by the local identity matrix I and L_{ij} corresponds to the low-order Jacobian which is not symmetric

in general. As a natural generalization of Theorem 3.1.2 to hyperbolic systems [142] assume that all off-diagonal matrix blocks $\mathrm{L}_{ij} \geq 0$ are positive semi-definite, that is,

$$\mathrm{v}^T \mathrm{L}_{ij} \mathrm{v} \geq 0 \qquad \forall \mathrm{v} \in \mathbb{R}^{N_\mathrm{U}} \quad \mathrm{v} \neq 0 \tag{5.2.21}$$

Let us mention that the discussion of definiteness is often restricted to only Hermitian matrices and there is no agreement in the literature on the proper definition of positive semi-definite for non-Hermitian matrices. The above inequality implies that all eigenvalues of the local block L_{ij} are non-negative. By saying this, we tacitly assume that the hyperbolicity of the original Jacobian carries over to its low-order counterpart such that each block L_{ij} is diagonalizable with real eigenvalues λ_k. The vector $\mathrm{v} \in \mathbb{R}^{N_\mathrm{U}}$ can be chosen arbitrarily, so that

$$\mathrm{v}^T \mathrm{L}_{ij} \mathrm{v} = \mathrm{v}^T \lambda_k \mathrm{v} = \lambda_k \|\mathrm{v}\|^2 \geq 0 \tag{5.2.22}$$

necessarily implies $\lambda_k \geq 0$ due to the norm property $\|\mathrm{v}\| = 0$ if and only if $\mathrm{v} = 0$. As a consequence, a possible generalization of Theorem 3.1.2 to hyperbolic systems is to require that the eigenvalues of all off-diagonal matrix blocks L_{ij} be non-negative [142]. For the trivial case $N_\mathrm{U} = 1$, this requirement is consistent with Jameson's LED [114] criterion 3.1.2 since the 'eigenvalue' of a scalar quantity is the coefficient itself. On the other hand, condition (5.2.21) is much less restrictive than the requirement that all off-diagonal entries of the global operator L be non-negative.

Theorem 5.2.1 (LED principle for systems [140])**.** A semi-discrete system of the form (5.2.20) is local extremum diminishing for a certain set of local characteristic variables if all off-diagonal matrix blocks L_{ij} are positive semi-definite, that is, their eigenvalues are all non-negative.

To satisfy this generalized LED criterion the symmetric boundary contribution B_{ij} is neglected in the assembly procedure (5.2.18) and tensorial artificial viscosity D_{ij} is applied so that the low-order operator L can be initialized by zeros and updated edge-by-edge as follows [142]

$$\begin{aligned} \mathrm{L}_{ii} &:= \mathrm{L}_{ii} + \mathrm{A}_{ij} - \mathrm{D}_{ij}, \qquad & \mathrm{L}_{ij} &:= -\mathrm{A}_{ij} + \mathrm{D}_{ij} \\ \mathrm{L}_{jj} &:= \mathrm{L}_{jj} - \mathrm{A}_{ij} - \mathrm{D}_{ij}, \qquad & \mathrm{L}_{ji} &:= \quad \mathrm{A}_{ij} + \mathrm{D}_{ij} \end{aligned} \tag{5.2.23}$$

Hence, there is no need to assemble the high-order operator K at all. Note that subtraction of D_{ij} from the diagonal blocks guarantees that mass is conserved. In fact, the global operator $\mathrm{D} = \{\mathrm{D}_{ij}\}$ is block-symmetric ($\mathrm{D}_{ij} = \mathrm{D}_{ji}$) and features blockwise zero row- and column-sums

$$\sum_{i=1}^{M} \mathrm{D}_{ij} = \sum_{j=1}^{M} \mathrm{D}_{ij} = \vec{0} \in \mathbb{R}^{N_\mathrm{U} \times N_\mathrm{U}} \tag{5.2.24}$$

The symmetric part B_{ij} which is just present at the boundary and vanishes in the interior as well as the contribution from the consistent mass matrix belong into the raw antidiffusive flux [142]

$$\mathrm{F}_{ij} = \left[\mathrm{M}_{ij} \frac{\mathrm{d}}{\mathrm{d}t} + \mathrm{D}_{ij} + \mathrm{B}_{ij} \right] (\mathrm{U}_i - \mathrm{U}_j), \qquad \mathrm{F}_{ji} = -\mathrm{F}_{ij} \tag{5.2.25}$$

which is skew-symmetric. It offsets the error induced by mass lumping and removes the amount of artificial viscosity which is required to enforce the generalized LED constraint. As in the scalar case it is limited to prevent the formation of spurious oscillations and inserted into the right-hand side of the semi-discrete low-order scheme (5.2.20) so as to obtain its flux-corrected counterpart

$$\mathrm{M}_i \frac{\mathrm{d}\mathrm{U}_i}{\mathrm{d}t} = \sum_{j \neq i} \mathrm{L}_{ij} (\mathrm{U}_j - \mathrm{U}_i) + \bar{\mathrm{F}}_i, \qquad \bar{\mathrm{F}}_i = \sum_{j \neq i} \bar{\mathrm{F}}_{ij} \tag{5.2.26}$$

A remark is in order in case the low-order scheme (5.2.19) is employed per se. If no flux correction is applied, it is worthwhile to account for the symmetric part B_{ij} in the right-hand side of equation (5.2.20) which can be assembled from individual edge contributions (see [142])

$$(\mathrm{LU})_i \quad \longleftarrow \quad (\mathrm{A}_{ij} + \mathrm{B}_{ij} - \mathrm{D}_{ij})(\mathrm{U}_i - \mathrm{U}_j) \tag{5.2.27}$$

$$(\mathrm{LU})_j \quad \longleftarrow \quad (\mathrm{A}_{ij} - \mathrm{B}_{ij} + \mathrm{D}_{ij})(\mathrm{U}_i - \mathrm{U}_j) \tag{5.2.28}$$

It remains to design the dissipation tensor D_{ij} so as to enforce the generalized LED constraint.

5.2.3. Design of artificial viscosities

As pointed out earlier, the compressible Euler equations represent a hyperbolic system of conservation laws so that any linear combination of the uni-directional Jacobian matrices is diagonalizable with real eigenvalues (see Definition 5.1.2). In particular, there exists a diagonal matrix Λ_{ij} of eigenvalues and a regular matrix R_{ij} of right eigenvectors such that the symmetric part of the cumulative Roe matrix $\mathrm{A}_{ij} = \mathbf{a}_{ij} \cdot \tilde{\mathbf{A}}_{ij}$ admits the following factorization [142]

$$\mathrm{A}_{ij} = |\mathbf{a}_{ij}|\, \mathrm{R}_{ij} \Lambda_{ij} \mathrm{R}_{ij}^{-1} \tag{5.2.29}$$

The scaling factor is given by the Euclidean norm of the skew-symmetric coefficient vector

$$|\mathbf{a}_{ij}| = \sqrt{\mathbf{a}_{ij} \cdot \mathbf{a}_{ij}}, \qquad \mathbf{a}_{ij} = (a_{ij}^1, a_{ij}^2, a_{ij}^3) \tag{5.2.30}$$

which is defined in (5.2.13). Furthermore, the entries of the diagonal matrix $\Lambda_{ij} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_5\}$ are given by real eigenvalues which represent the characteristic speeds of wave propagation [140]

$$\lambda_1 = \tilde{v}_{ij} - \tilde{c}_{ij}, \qquad \lambda_2 = \lambda_3 = \lambda_4 = \tilde{v}_{ij}, \qquad \lambda_5 = \tilde{v}_{ij} + \tilde{c}_{ij} \tag{5.2.31}$$

The speed of sound $\tilde{c}_{ij}$ for Roe's linearization can be computed from formula (5.2.10) whereas the 'projection' of the density-averaged velocity vector $\mathbf{v}_{ij}$ onto the edge ij yields [142]

$$\tilde{v}_{ij} = |\mathbf{a}_{ij}|^{-1}\, \mathbf{a}_{ij} \cdot \tilde{\mathbf{v}}_{ij} \tag{5.2.32}$$

In the continuous case, there are two superimposed acoustic waves traveling at speeds $\pm c$ relative to the gas. Moreover, the characteristics associated with the multiple eigenvalue $\lambda_2 = \lambda_3 = \lambda_4$ represent the trajectories of fluid particles [64]. In contrast to the scalar case, there may be different 'upwind' directions for different waves depending on the sign of the eigenvalues.

The artificial dissipation D_{ij} is meant to eliminate the negative eigenvalues in the off-diagonal blocks of the discrete Jacobian, i.e. $\mathrm{L}_{ij} = -\mathrm{A}_{ij} + \mathrm{D}_{ij}$ and $\mathrm{L}_{ji} = \mathrm{A}_{ij} + \mathrm{D}_{ij}$. At the same time, the diagonalizability property of the original operator should be preserved. A viable strategy is to introduce matrix $|\Lambda_{ij}|$ which contains the absolute values of the eigenvalues

$$|\Lambda_{ij}| = \mathrm{diag}\{|\lambda_1|, \ldots, |\lambda_5|\} \tag{5.2.33}$$

and define the artificial viscosity tensor D_{ij} for the edge ij as follows [142]

$$\mathrm{D}_{ij} = |\mathbf{a}_{ij}|\, \mathrm{R}_{ij}\, |\Lambda_{ij}|\, \mathrm{R}_{ij}^{-1} \tag{5.2.34}$$

It is instructive to separate the eigenvalues of the cumulative Roe matrix A_{ij} into their positive and negative contributions $\Lambda_{ij}^{\pm} = \frac{1}{2}(\Lambda_{ij} \pm |\Lambda_{ij}|)$ (see [64]). As a consequence, application of the artificial viscosity D_{ij} to the off-diagonal blocks of the low-order Jacobian matrix yields

$$\mathrm{L}_{ij} = -2|\mathbf{a}_{ij}|\, \mathrm{R}_{ij} \Lambda_{ij}^{-} \mathrm{R}_{ij}^{-1}, \qquad \mathrm{L}_{ji} = 2|\mathbf{a}_{ij}|\, \mathrm{R}_{ij} \Lambda_{ij}^{+} \mathrm{R}_{ij}^{-1} \tag{5.2.35}$$

which proves that the corresponding eigenvalues are non-negative as required by the generalized LED constraint 5.2.1. In particular, the above strategy corresponds to the flux difference splitting approach. In one dimension, Roe's approximate Riemann solver [224] is recovered which is one of the most popular discretization techniques for the Euler equations of gas dynamics.

Instead of dealing with the cumulative Roe matrix for the edge ij, it is also possible to employ dimensional splitting and diagonalize the uni-directional Jacobians one at a time [105, 157, 251]

$$\mathrm{A}_{ij} = \mathbf{a}_{ij} \cdot \tilde{\mathbf{A}}_{ij} = \sum_{d=1}^{N_\mathrm{D}} a_{ij}^d \tilde{A}_{ij}^d, \qquad \tilde{A}_{ij}^d = \mathrm{R}_{ij}^d \Lambda_{ij}^d [\mathrm{R}_{ij}^d]^{-1} \tag{5.2.36}$$

The generalized LED criterion can be satisfied by elimination of negative eigenvalues for each spatial dimension which leads to the following definition of artificial dissipation [140]

$$\mathrm{D}_{ij} = \sum_{d=1}^{N_\mathrm{D}} \mathrm{R}_{ij}^d |a_{ij}^d \Lambda_{ij}^d| [\mathrm{R}_{ij}^d]^{-1} \tag{5.2.37}$$

As a result, the off-diagonal blocks of the low-order Jacobian operator are given by

$$\mathrm{L}_{ij} = \sum_{d=1}^{N_\mathrm{D}} -\mathrm{R}_{ij}^d \min\{0, 2a_{ij}^d \Lambda_{ij}^d\} [\mathrm{R}_{ij}^d]^{-1}, \qquad \mathrm{L}_{ji} = \sum_{d=1}^{N_\mathrm{D}} \mathrm{R}_{ij}^d \max\{0, 2a_{ij}^d \Lambda_{ij}^d\} [\mathrm{R}_{ij}^d]^{-1} \tag{5.2.38}$$

Of course, this decomposition into one-dimensional wave patterns brings about strong numerical diffusion in the crosswind direction. Moreover, the cost of evaluating D_{ij} triples in comparison to (5.2.34) but parallel implementation is straightforward. On the other hand, dimensional splitting can be used to perform flux limiting in terms of characteristic variables [135, 136, 140].

Last but not least, the spectral radius of the cumulative Roe matrix provides a reasonable measure for the amount of viscosity required to eliminate negative eigenvalues [163, 172, 269]

$$\mathrm{D}_{ij} = |\mathbf{a}_{ij}| \lambda_{\max} \mathrm{I}, \qquad \lambda_{\max} = \max_k \{|\lambda_k|\} \tag{5.2.39}$$

Note that D_{ij} just affects the diagonal blocks of the discrete Jacobian operator, and moreover, it is the same for all variables. In particular, this so-called scalar dissipation approach corresponds to discrete upwinding applied to the fastest wave propagating at the characteristic speed

$$\lambda_{\max} = |\tilde{v}_{ij}| + \tilde{c}_{ij} \tag{5.2.40}$$

It leads to a very efficient assembly of the global matrix L without resorting to characteristic factorizations of the form (5.2.29). Surprisingly enough, a slightly over-diffusive low-order method may even be preferable due to the resulting improvement of phase accuracy [142, 269] as long as excessive artificial viscosity is removed in the course of flux correction. Furthermore, scalar dissipation can be used as a cost-effective preconditioner for the approximate Riemann solver.

Example 5.2.1 (Kuzmin and M. [140])**.** To illustrate the construction of the non-oscillatory low-order scheme (5.2.19), consider the one-dimensional Euler equations in quasi-linear form

$$\frac{\partial U}{\partial t} + A(U) \frac{\partial U}{\partial x} = 0 \tag{5.2.41}$$

An explicit expression for the Jacobian matrix $A(U) = \frac{\partial F}{\partial U}$ is given in equation (5.1.21) and the characteristic decomposition $A = R\Lambda R^{-1}$ is discussed in Section 5.1.5. The use of linear finite elements for discretization in space employed on a uniform mesh of width Δx yields

$$m_i = \Delta x, \qquad c_{ij} = \begin{cases} 1/2 & \text{for} \quad j = i+1 \\ -1/2 & \text{for} \quad j = i-1 \end{cases} \tag{5.2.42}$$

At interior nodes, the high-order element/edge contribution is of the form

$$\mathrm{K}_{ij} = c_{ij}\tilde{A}_{ij} = -\mathrm{A}_{ij}, \qquad j = i \pm 1 \tag{5.2.43}$$

where $\tilde{A}_{ij}$ stands for the Jacobian matrix which is evaluated using the Roe mean values $(\tilde{v}_{ij}, \tilde{H}_{ij})$. The tensor of viscous dissipation D_{ij} defined in (5.2.34) and (5.2.37) simplifies to

$$\mathrm{D}_{ij} = \frac{1}{2}\mathrm{R}_{ij}\,|\Lambda_{ij}|\,\mathrm{R}_{ij}^{-1}, \qquad \Lambda_{ij} = \mathrm{diag}\{\tilde{v}_{ij} - \tilde{c}_{ij}, \tilde{v}_{ij}, \tilde{v}_{ij} + \tilde{c}_{ij}\} \tag{5.2.44}$$

where the density-averaged speed of sound is given by $\tilde{c}_{ij} = \sqrt{(\gamma-1)(\tilde{H}_{ij} - \frac{1}{2}\tilde{v}_{ij}^2)}$. Substitution of the above expression for the artificial viscosity into formula (5.2.28) yields the numerical flux

$$\mathrm{G}_{ij}^{\mathrm{Roe}} = \frac{\mathrm{F}_i + \mathrm{F}_j}{2} - \frac{1}{2}\mathrm{R}_{ij}\,|\Lambda_{ij}|\,\Delta\mathrm{W}_{ij}, \qquad j = i+1 \tag{5.2.45}$$

which is identical to that recovered from Roe's approximate Riemann solver [224]. Let the difference between two nodal solution values be transformed into characteristic variables

$$\Delta\mathrm{W}_{ij} = \mathrm{R}_{ij}^{-1}(\mathrm{U}_j - \mathrm{U}_i) \tag{5.2.46}$$

This representation reveals that Roe's first-order scheme, which is described in many textbooks on gas dynamics [82, 105, 157, 251, 262], can be interpreted as discrete upwinding (see Section 3.2.4) applied to the decoupled scalar equations for the local characteristic variables. In particular, the amount of artificial diffusion for wave k is proportional to the absolute value of the eigenvalue $\frac{1}{2}|\lambda_k|$ which is just enough to render the scalar semi-discrete scheme for the k^{th} field LED.

5.2.4. Characteristic limiters of TVD type

In the last decades, remarkable progress has been made in the development of high-resolution schemes for scalar conservations laws and a genuinely multidimensional framework is available for finite element discretizations. What makes its rigorous generalization to hyperbolic systems difficult, is the lack of reliable physical and mathematical criteria for the design of flux limiters. The intricate coupling of the Euler equations (5.1.1) precludes the use of a flux correction algorithm developed for scalar problems which is independently applied to the continuity equation, momentum equation and energy equation. In particular, this naïve adjustment of the conservative fluxes may give rise to non-physical undershoots and overshoots in certain dependent variable such as pressure, internal energy or entropy. In light of the above, the design of flux limiters for hyperbolic system is more involved than that for scalar conservation laws since the evolution of *all* physically relevant quantities needs to be controlled simultaneously [140]. On the one hand, the numerical solution is strongly influences by the type of the flux limiter as it is the case for scalar equations. On the other hand, the set of variables to which flux correction is applied plays an important role. For an in-depth coverage of this topic, the reader is referred to [269].

In this work, we neglect the contribution of the consistent mass matrix to the raw antidiffusive fluxes (5.2.25) and employ the characteristic limiter of TVD type which was recently proposed by Kuzmin [135] for the treatment of coupled systems. As for scalar conservation laws [145], the resulting semi-discretized problem can be cast into the general form; cf. equation (3.3.10)

$$\mathrm{M}_L\frac{\mathrm{dU}}{\mathrm{d}t} = \mathrm{K}^*\mathrm{U} \tag{5.2.47}$$

where the modified Jacobian $\mathrm{K}^* = \{\mathrm{K}_{ij}^*\}$ represents a discrete counterpart of the operator $-\mathbf{A}\cdot\nabla$ designed to be local extremum diminishing for a set of local characteristic variables. It is composed from the low-order Jacobian (5.2.23) and a limited amount of compensating antidiffusion so that its structure is comparable to that of its scalar counterpart (3.3.11).

The origin of characteristic TVD schemes for the one-dimensional Euler equations dates back to the work by Yee et al. [266, 267] in the 1980s. Various ad hoc extensions to finite elements have been proposed in the literature [9, 66, 172, 231, 232] and applied with considerable success. However, the derivation of a genuinely multidimensional generalization remains a challenging task since waves can travel in an infinite number of directions so that the wave decomposition is no longer unique. It is therefore unclear how to transform the linearized hyperbolic system into a set of decoupled advection equations, for which robust numerical techniques are available.

Consider the semi-discrete system (5.2.26) and let the raw antidiffusive flux be given by

$$\mathrm{F}_{ij} = [\mathrm{D}_{ij} + \mathrm{B}_{ij}](\mathrm{U}_i - \mathrm{U}_j), \qquad \mathrm{F}_{ji} = -\mathrm{F}_{ij} \tag{5.2.48}$$

where D_{ij} denotes the tensor of artificial viscosity and B_{ij} is the symmetric boundary contribution defined in (5.2.15). Since the cumulative Roe matrix $\mathrm{A}_{ij} = \mathbf{a}_{ij} \cdot \tilde{\mathbf{A}}_{ij}$ is evaluated in the direction of the coefficient vector $\mathbf{a}_{ij}$ it can be diagonalized as presented in equation (5.2.29).

Characteristic TVD schemes can be based on uni-directional slope limiting [174] so that the use of a single direction for the definition of local characteristic variables is feasible. On the other hand, the design of node-oriented flux limiters is based on the following algorithmic steps [140]:

1. Collect upstream/downstream edge contributions to individual nodes;
2. Compute nodal correction factors on the basis of this information;
3. Check the sign of antidiffusive fluxes and limit them edge-by-edge.

The above operations demand for consistency, that is, the nodal data collected in the first step should correspond to the same set of characteristic variables. However, the transformation matrices R_{ij} and R_{ij}^{-1} for the cumulative Roe matrix A_{ij} depend not only on the density averaged solution values but also on the coefficient vector $\mathbf{a}_{ij}$ which represents the direction for the edge at hand.

As a remedy, dimensional splitting can be adopted which amounts to performing flux correction independently for each coordinate direction as proposed by Yee et al. [267]. Negative eigenvalues of the uni-directional Roe matrices $\tilde{A}_{ij}^d$ are eliminated to satisfy the generalized LED constraint 5.2.1 for each spatial dimension [135, 140]. Excessive artificial viscosities are removed by performing characteristic flux limiting based on the raw antidiffusive flux [135]

$$\mathrm{F}_{ij} = \sum_{d=1}^{N_\mathrm{D}} \mathrm{F}_{ij}^d, \qquad \mathrm{F}_{ij}^d := \mathrm{R}_{ij}^d \left(|a_{ij}^d \Lambda_{ij}^d| + b_{ij}^d \Lambda_{ij}^d \right) [\mathrm{R}_{ij}^d]^{-1} (\mathrm{U}_i - \mathrm{U}_j) \tag{5.2.49}$$

Since dimensional splitting has to be performed also for the low-order contribution it is worthwhile to initialize the right-hand side of (5.2.47) by the skew-symmetric part of the high-order operator

$$(\mathrm{K}^* \mathrm{U})_i := \sum_{j \neq i} \mathrm{A}_{ij} (\mathrm{U}_i - \mathrm{U}_j), \qquad \mathrm{A}_{ij} = \mathbf{a}_{ij} \cdot \tilde{\mathbf{A}}_{ij} \tag{5.2.50}$$

and update it following the algorithm presented in Figures 5.1. It is ivoked individually for each spatial dimension $d = 1, 2, 3$, whereby the nodal quantities $P_i^\pm$ and $Q_i^\pm$ are initialized by zeros.

A few remarks are in order. The definition of nodal correction factors implies $|R_{I,k}^\pm P_{I,k}^\pm| \leq |Q_{I,k}^\pm|$ for all waves k at the upwind node I. Moreover, the coefficient $a_{ji}^d = -a_{ij}^d$ is skew-symmetric so that the off-diagonal entries of the low-order operator in characteristic variables satisfy

$$l_{ij}^k = -\min\{0, 2a_{ij}^d \lambda_k^d\} \geq 0, \qquad l_{ji}^k = \max\{0, 2a_{ij}^d \lambda_k^d\} \geq 0 \tag{5.2.51}$$

Let node i be located 'upstream' so that $a_{ij}^d \lambda_k^d \geq 0$ according to expression (5.2.57). Hence, the LED property for the downstream node j follows from the fact that

$$l_{ji}^k - \mathrm{R}_{i,k}^\pm \max\{0, \min\{|a_{ij}^d \lambda_k^d| + b_{ij}^d \lambda_k^d, 2|a_{ij}^d \lambda_k^d|\}\} \geq (1 - \mathrm{R}_{i,k}^\pm) 2a_{ij}^d \lambda_k^d \geq 0 \tag{5.2.52}$$

In a loop over edges:

1. Decompose the difference between the nodal solution values U_j and U_i given in conservative variables into its characteristic components

$$\Delta \mathrm{W}_{ij} = [\mathrm{R}_{ij}^d]^{-1}(\mathrm{U}_j - \mathrm{U}_i) \tag{5.2.53}$$

whereby the rows of the transformation matrix R_{ij}^{-1} are populated by the left eigenvectors of the uni-directional Roe matrix $\tilde{A}_{ij}^d$ corresponding to the eigenvalues

$$\lambda_1^d = \tilde{v}_{ij}^d - \tilde{c}_{ij}, \qquad \lambda_2^d = \lambda_3^d = \lambda_4^d = \tilde{v}_{ij}, \qquad \lambda_5^d = \tilde{v}_{ij}^d + \tilde{c}_{ij} \tag{5.2.54}$$

2. Compute the raw (anti-)diffusive characteristic flux for each field number k as follows

$$\mathrm{C}_{ij}^k = -\max\{0, \min\{|a_{ij}^d \lambda_k^d| + b_{ij}^d \lambda_k^d, 2|a_{ij}^d \lambda_k^d|\}\}\Delta \mathrm{W}_{ij}^k = -\mathrm{C}_{ji}^k \tag{5.2.55}$$

where the interior and boundary coefficients $\mathbf{a}_{ij}$ and $\mathbf{b}_{ij}$ are defined in (5.2.13)

3. Update the sums of edge contributions to nodes i and j for each field number k

$$\left.\begin{aligned} P_{i,k}^{\pm} &:= P_{i,k}^{\pm} + \max_{\min}\{0, \mathrm{C}_{ij}^k\} \\ Q_{i,k}^{\mp} &:= Q_{i,k}^{\mp} - \max_{\min}\{0, \mathrm{C}_{ij}^k\} \\ Q_{j,k}^{\pm} &:= Q_{j,k}^{\pm} + \max_{\min}\{0, \mathrm{C}_{ij}^k\} \end{aligned}\right\} \quad \begin{array}{l} \text{if} \quad a_{ij}^d \lambda_k^d \geq 0 \quad \text{otherwise} \\ \text{exchange node numbers } i \text{ and } j \end{array} \tag{5.2.56}$$

In a loop over nodes:

3. Evaluate the nodal correction factors $R_{i,k}^{\pm} = \min\{1, Q_{i,k}^{\pm}/P_{i,k}^{\pm}\}$

In a loop over edges:

4. Determine the number of the 'upwind node' *individually* for each scalar wave propagating at the characteristic speed λ_k^d relative to the edge ij and limit the flux accordingly

$$I = \begin{cases} i & \text{if} \quad a_{ij}^d \lambda_k^d \geq 0 \\ j & \text{if} \quad a_{ij}^d \lambda_k^d < 0 \end{cases} \qquad \bar{\mathrm{C}}_{ij}^k = \begin{cases} R_{I,k}^{+} \mathrm{C}_{ij}^k & \text{if} \quad \mathrm{C}_{ij}^k \geq 0 \\ R_{I,k}^{-} \mathrm{C}_{ij}^k & \text{if} \quad \mathrm{C}_{ij}^k < 0 \end{cases} \tag{5.2.57}$$

5. Add the limited antidiffusive correction $\bar{\mathrm{C}}_{ij}$ to the raw *diffusive* flux $|a_{ij}^d \Lambda_{ij}^d| \Delta \mathrm{W}_{ij}$

$$\bar{\mathrm{F}}_{ij}^d = \mathrm{R}_{ij}^d (|a_{ij}^d \Lambda_{ij}^d| \Delta \mathrm{W}_{ij} + \bar{\mathrm{C}}_{ij}), \qquad \bar{\mathrm{F}}_{ji}^d = -\bar{\mathrm{F}}_{ij}^d \tag{5.2.58}$$

and transform the result back to the conservative variables prior to its application

$$(\mathrm{K}^* \mathrm{U})_i := (\mathrm{K}^* \mathrm{U})_i + \bar{\mathrm{F}}_{ij}^d, \qquad (\mathrm{K}^* \mathrm{U})_j := (\mathrm{K}^* \mathrm{U})_j - \bar{\mathrm{F}}_{ij}^d \tag{5.2.59}$$

Fig. 5.1. Characteristic flux correction algorithm of TVD type [135].

as long as $R^{\pm}_{i,k} \leq 1$. Conversely, $a^d_{ij}\lambda^d_k < 0$ implies that the negative off-diagonal entry has been eliminated from row j, i.e. $l^k_{ji} = 0$, and the LED constraint for row i is satisfied for $R^{\pm}_{j,k} \leq 1$:

$$l_{ij} - \mathrm{R}^{\pm}_{j,k} \max\{0, \min\{|a^d_{ij}\lambda^d_k| + b^d_{ij}\lambda^d_k, 2|a^d_{ij}\lambda^d_k|\}\} \geq (R^{\pm}_{j,k} - 1)a^d_{ij}\lambda^d_k \geq 0 \tag{5.2.60}$$

In particular, the raw antidiffusive flux is proportional to $|a^d_{ij}\lambda^d_k| + b^d_{ij}\lambda^d_k$. However, the symmetric part which is only present at the boundary has been intentionally neglected in the construction of the low-order scheme [142]. In order to enforce the LED constraint at both nodes, the flux needs to be 'prelimited' according to (5.2.55) which is quite similar to expression (3.3.6) valid for scalar conservation laws. On the other hand, the 'safeguard' $2|a^d_{ij}\lambda^d_k|$ can be employed without knowing the orientation of the edge ij explicitly (cf. equation (5.2.51) above).

To conclude what we have said so far implicit high-resolution finite element schemes for the compressible Euler equations can be based on the conservative flux decomposition (5.2.6). Roe's linearization strategy can be used to express jumps of inviscid fluxes in terms of solution differences multiplied by Jacobian matrices which are evaluated at density-averaged mean values. A viable low-order scheme can be constructed by adding tensorial artificial viscosity designed to eliminate negative eigenvalues from off-diagonal Jacobians so as to satisfy the generalized LED constraint 5.2.1. Excessive diffusion can be removed by performing flux limiting of TVD type which is individually applied to characteristic fluxes for each spatial dimensions.

5.2.5. Iterative solution techniques

The nonlinear algebraic systems resulting from a (semi-)implicit time-discretization of the compressible Euler equations exhibit the same structure as their scalar counterparts considered in Chapter 3. This makes it possible to design iterative solution strategies based on the fixed-point iteration scheme (3.3.84) whose generalization to systems of equations reads as follows

$$\mathrm{U}^{(m+1)} = \mathrm{U}^{(m)} + \mathrm{P}^{-1}\mathrm{R}^{(m)}, \qquad \mathrm{U}^{(0)} = \mathrm{U}^n, \qquad m = 0, 1, 2, \ldots \tag{5.2.61}$$

In the above expression, P is a suitable preconditioner and the constant right-hand side for the FEM-TVD methodology and the residual of the m^{th} outer iteration are given by

$$\mathrm{B}^n = [\mathrm{M}_L + (1-\theta)\Delta t \mathrm{K}^*(\mathrm{U}^n)]\mathrm{U}^n \tag{5.2.62}$$

$$\mathrm{R}^{(m)} = \mathrm{B}^n - [\mathrm{M}_L - \theta\Delta t \mathrm{K}^*(\mathrm{U}^{(m)})]\mathrm{U}^{(m)} \tag{5.2.63}$$

The latter vanishes for the exact solution of the discrete problem and it is supposed to approach zero in an actual simulation. For implicit time-stepping schemes, the criteria imposed by the TVD limiting strategy are only satisfied upon convergence and an insufficient number of outer iterations may lead to non-physical solutions [140]. It is possible to employ divided differences and approximate the Jacobian operator following the strategy proposed in Section 3.3.5. As of this writing, the design of a discrete Newton method for the Euler equations has not been pursued by the author in practice. As an alternative, the system matrix for the low-order approach (5.2.19) discretized in time by the standard $\theta-$scheme constitutes a viable preconditioner [142]

$$\mathrm{P} = \mathrm{M}_L - \theta\Delta t \mathrm{L}(\mathrm{U}^{(m)}) \tag{5.2.64}$$

The 'inversion' of the global operator P is prohibitively expensive, and moreover, the following two-step implementation of (5.2.61) constitutes a good starting point for further simplifications

$$\mathrm{P}\Delta\mathrm{U}^{(m)} = \mathrm{R}^{(m)}, \qquad m = 0, 1, 2, \ldots \tag{5.2.65}$$

$$\mathrm{U}^{(m+1)} = \mathrm{U}^{(m)} + \Delta\mathrm{U}^{(m)}, \qquad \mathrm{U}^{(0)} = \mathrm{U}^n \tag{5.2.66}$$

As in the scalar case, a certain number of inner iterations is performed to compute the solution increment $\Delta \mathrm{U}^{(m)}$ which is used to update the last iterate. Adopting definition (5.2.64) for P, the linear subproblem to be solved in each outer iteration exhibits the following structure

$$\theta\Delta t \begin{bmatrix} \frac{M_L}{\theta\Delta t} - L_{11} & L_{12}^{(m)} & L_{13}^{(m)} & L_{14}^{(m)} & L_{15}^{(m)} \\ L_{21}^{(m)} & \frac{M_L}{\theta\Delta t} - L_{22}^{(m)} & L_{23}^{(m)} & L_{24}^{(m)} & L_{25}^{(m)} \\ L_{31}^{(m)} & L_{32}^{(m)} & \frac{M_L}{\theta\Delta t} - L_{33}^{(m)} & L_{34}^{(m)} & L_{35}^{(m)} \\ L_{41}^{(m)} & L_{42}^{(m)} & L_{43}^{(m)} & \frac{M_L}{\theta\Delta t} - L_{44}^{(m)} & L_{45}^{(m)} \\ L_{51}^{(m)} & L_{52}^{(m)} & L_{53}^{(m)} & L_{54}^{(m)} & \frac{M_L}{\theta\Delta t} - L_{55}^{(m)} \end{bmatrix} \begin{bmatrix} \Delta \mathrm{U}_1^{(m)} \\ \Delta \mathrm{U}_2^{(m)} \\ \Delta \mathrm{U}_3^{(m)} \\ \Delta \mathrm{U}_4^{(m)} \\ \Delta \mathrm{U}_5^{(m)} \end{bmatrix} = \begin{bmatrix} \mathrm{R}_1^{(m)} \\ \mathrm{R}_2^{(m)} \\ \mathrm{R}_3^{(m)} \\ \mathrm{R}_4^{(m)} \\ \mathrm{R}_5^{(m)} \end{bmatrix}$$

Here, the row index refers to the vector of unknowns for a particular conservative variable (density, momentum component or energy) which interacts with the others via the off-diagonal blocks.

As an example, consider the first row of the preconditioner which represents a low-order approximation to the continuity equation $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$. The density is only related to the momentum flux which leads to zero entries in the upper left/right corner of the Jacobian operator (cf. Appendix C for the two-dimensional case). This property carries over to the discrete Jacobian K, that is, the sub-matrices K_{11} and K_{15} vanish if the standard Galerkin method is employed without stabilization. Moreover, the scalar matrices K_{1k} and K_{k5} for $k = 2,3,4$ depend only on the directional vector $\mathbf{e}$ along which the Jacobian is evaluated, and hence, they are constant as long as the mesh is fixed. However, the use of artificial viscosities required to turn the high-order operator K into its low-order counterpart L may give rise to non-vanishing contributions to all matrix blocks of the global preconditioner P. On the other hand, the use of scalar dissipation (5.2.40) proportional to the spectral radius of the Roe matrix only affects the diagonal blocks L_{kk} of the preconditioner, whereas the off-diagonal blocks are equal to their high-order counterparts K_{kl} for $k \neq l$.

Due to the strong coupling of equations, it may be necessary to solve the global problem en bloc so as to achieve convergence. Of course, the use of a robust solution strategy for very large linear problems is mandatory. In our experience, the BiCGSTAB and GMRES algorithms [228] work well in practice. In order to improve the linear convergence rates, a block incomplete LU-decomposition of P may be employed as preconditioner. Various strategies for the design of (B)ILU preconditioners including a comparison between pointwise and blockwise factorizations are presented in the work by van der Ploeg et al. [207, 208] and the references therein.

On the other hand, the fully coupled solution of the above system is quite demanding in terms of computational time and memory required to assemble the global preconditioned P and store it, respectively. As an alternative, a block-Jacobi method [55] may be adopted, whereby all off-diagonal blocks P_{kl} for $k \neq l$ are set to zero. As a consequence, the global update procedure (5.2.65)–(5.2.66) decouples into a sequence of scalar subproblems for each variable [140, 142]

$$P_{kk}\Delta \mathrm{U}_k^{(m)} = \mathrm{R}_k^{(m)}, \qquad k = 1,\ldots,N_{\mathrm{U}} \tag{5.2.67}$$

$$\mathrm{U}_k^{(m+1)} = \mathrm{U}_k^{(m)} + \Delta \mathrm{U}_k^{(m)}, \qquad \mathrm{U}_k^{(0)} = \mathrm{U}_k^n \tag{5.2.68}$$

which can be treated separately or, better yet, in parallel. The number of inner iterations needed to obtain the solution increment $\Delta \mathrm{U}_k^{(m)}$ with a prescribed tolerance may vary from one variable to the other and, for instance, the update of the density value may require less effort than that of the energy. The decoupling of equations makes is possible to apply all techniques developed in the previous chapter for scalar problems to the sequence of subproblems (5.2.67)–(5.2.68), and moreover, a different solution strategy can be used for each equation. The implementation of this segregated approach is very easy and its performance is satisfactory as long as the time step

remains moderately small [140]. However, the nonlinear convergence rates deteriorate and the simulation may even fail completely as the time step is increased. If larger time steps are employed, e.g., in steady-state comutations, use of the fully coupled approach is advisable to ensure convergence. Another possibility is to adopt a block-Gauß-Seidel method [55] which represents a compromise between the fully coupled and the segregated approch [140]. Furthermore, full multigrid algorithms may be employed for the computation of steady-state solutions [100, 159, 239].

Let us consider two alternative approaches for storing the global preconditioner or parts of it. The aforementioned structure of the global system corresponds to the so-called globally blocked memory layout which is described in Appendix B in more detail. In short, each scalar sub-matrix, i.e. $P_{kl} = \delta_{kl} M_L + L_{kl}$, is stored contiguously (cf. Figure B.2) so that individual blocks can be included or neglected individually in the assembly of the preconditioner. This representation seems to be particularly useful for the segregated approach (5.2.67)–(5.2.68) if a different solution strategy is applied to each scalar subproblem. On the other hand, both the residual vector and the global preconditioner P are evaluated edge-by-edge, whereby local Jacobian matrices need to be scattered to their global positions. Hence, the locally blocked memory layout depicted in Figure B.3 is likely to be more efficient in an edge-based solution algorithm. The advantage of this storage technique becomes even more pronounced for the fully coupled approach.

5.3. Boundary conditions

The correct treatment of boundary conditions for the Euler equations is crucial for the development of accurate numerical schemes. A comprehensive coverage of this topic is available in a number of textbooks [79, 105, 262]. One can distinguish between internal and external flows. The first type frequently occurs in channel flow computations which may develop complicated wave patterns due to obstacles present in the interior. For such problems, the domain is naturally bounded by impermeable walls that may interact with impinging waves. Nonetheless, some care is required at the inlet and outlet in order to reflect the physical behavior of the flow outside the channel. On the other hand, in aerodynamic applications such as flow over airfoils one typically starts with an infinite domain that needs to be bounded artificially so that the problem setup does not exceed the limited resources of a computer. Hence, apart from physical boundary conditions dictated by the wave pattern, some artificial ones must be imposed when it comes to the numerical simulation of flow problems. Mathematically speaking, the N_p physical boundary conditions (PBC) are required to secure the existence and uniqueness of the exact solution, whereas the N_n numerical boundary conditions (NBC) are supposed to ensure that various perturbations generated in the interior of the computational domain leave it without being reflected at the boundaries [105]. For hyperbolic systems with N_U state variables a proper combination of PBC and NBC must be imposed such that

$$N_\mathrm{U} = N_\mathrm{p} + N_\mathrm{n} \qquad \text{where} \quad N_\mathrm{p} \leq N_\mathrm{U} \quad \text{and} \quad N_\mathrm{n} \leq N_\mathrm{U} \tag{5.3.1}$$

The issue of deciding which boundary conditions to impose is a non-trivial task since inappropriate choices can easily ruin the existence and uniqueness of solutions and even lead to wrong solutions.

5.3.1. Types of boundary conditions

In essence, there are two classes of boundaries that need to be considered for the Euler equations: The first family of boundaries is known as solid surfaces which occur at walls in interior channels or at obstacles such as airfoils immersed in the flow. The other group is termed open boundaries which need to be considered for instance at the inlet and outlet of a channel. Whenever an infinite domain is bounded artificially, e.g. in the computation of flow around an airfoil, open boundaries come into play and in this case, it is common practice to refer to them as far-field boundaries.

For simplicity, let us present the basic ideas for the one-dimensional Euler equations and refer to the literature for an extension to multidimensions [105, 127]. As pointed out in Section 5.1.5, the one-dimensional Euler equations are strictly hyperbolic, that is, the Jacobian matrix (5.1.21) is diagonalizable with distinct real eigenvalues (5.1.22) which define the three characteristics [105]

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v - c, \qquad \frac{\mathrm{d}x}{\mathrm{d}t} = v, \qquad \frac{\mathrm{d}x}{\mathrm{d}t} = v + c \tag{5.3.2}$$

As a rule of thumb, the number of physical boundary conditions to be prescribed at a point on the boundary must be equal to the number of incoming characteristics. Consequently, the number of numerical boundary conditions is given by the number of outgoing characteristics.

Depending on the Mach number $M = v/c$, one can distinguish between different flow configurations as depicted in Figure 5.2. Consider the point $P_1 = P_1(x,t)$ which is located in a supersonic region, i.e. $M > 1$. If it is placed at the inflow boundary then all characteristics are directed into the domain, and hence, boundary conditions need to be prescribed for all three variables. In contrast, it is not allowed to prescribe any boundary value if the point P_1 is located on the outflow. For the subsonic case ($M < 1$) two boundary conditions are required at the inflow boundary, whereas only one boundary condition can be imposed if the point $P_2 = P_2(x,t)$ is located at the outlet [105].

The physical conditions to be prescribed are the entropy and the values of the Riemann variables (5.1.32), whereby the sign of the eigenvalues defines the type of variables which need to be set [105]. However, these variables are generally not known for practical applications, and physical boundary conditions are typically prescribed in terms of velocities and pressure. It is thus that physical quantities need to be transformed to the Riemann variables to allow for a characteristic treatment. Moreover, numerical schemes require the values of all variables at the boundary so that numerical boundary conditions are mandatory which need to be compatible with the physical flow behavior [105]. In light of the above, the numerical treatment of boundary conditions for the Euler equations is a challenging task which may influence the simulation results to a large extent.

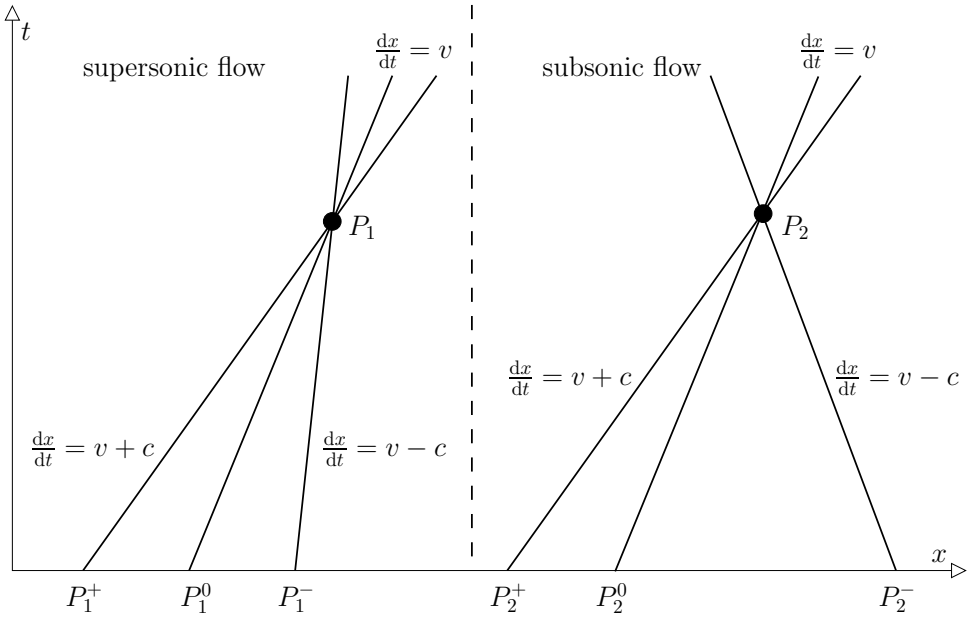


Fig. 5.2. Propagation of flow quantities in a one-dimensional inviscid flow.

5.3.2. Implementation of boundary conditions

In practice, physical boundary conditions are typically imposed on the primitive variables or they are given in terms of input data like total enthalpy, entropy, temperature, inclination angle or some combination thereof. On the other hand, the numerical solution is sought in terms of conservative variables so that it is impossible to impose the Dirichlet boundary conditions in the usual way. It is common practice to recover the boundary values by changing to the characteristic variables, evaluating the incoming Riemann invariants from the physical boundary conditions and extrapolating the outgoing ones from the interior of the domain [105, 262, 265]. The inverse transformation is used to recover the desired values in conservative variables which can be used to evaluate the numerical fluxes for an (explicit) finite volume method or imposed as Dirichlet boundary conditions for a finite difference scheme. The extrapolation of Riemann invariants is expensive to perform on unstructured meshes, and moreover, its theoretical justification is not clear.

Many publications are related to finite volume discretizations which are typically employed for fluid dynamics applications. In the cell-centered framework, the state variables are given at the centroids of cells, so that special care must be taken to recover accurate boundary values from the interior. In contrast, most finite element schemes are free of this drawback since the degrees of freedom are located at the vertices and/or at the midpoints of edges. To the author's best knowledge, the implementation of characteristic boundary conditions in finite element codes has received only little attention in the literature [176]. Some valuable information can be found in the book by Shapiro [236] who presents an adaptive finite element solution algorithm for the Euler equations based on an explicit multi-step time integration method. Some guidance for the accurate implementation of wall boundary conditions in curved domains is given by Berger and Krivodonova [126]. The numerical treatment of wall boundary conditions in an implicit finite element code is addressed by Sens and Mortchelewicz [234] who resort to an element-based evaluation of the wall pressure from velocity-corrected boundary fluxes. The remaining variables are computed from the updated pressure values and extrapolated quantities such as the entropy.

5.3.3. Node-based predictor-corrector algorithm

Our implementation of boundary conditions is based on the predictor-corrector algorithm introduced by Kuzmin et al. [142] which is applicable to all types of (characteristic) boundary conditions. In implicit schemes for the Euler equations, the new solution value U^{n+1} has to be computed iteratively, e.g. by performing a number of fixed-point iteration defect correction steps (5.2.61). In each outer iteration, a sufficient number of inner iterations (5.2.65) is carried out to compute the solution increment $\Delta\mathrm{U}^{(m)}$ which is applied to the last iterate (5.2.66). Once the outer iteration has converged the last iterate yields an approximation to the end-of-step solution. Several strategies for choosing the preconditioner P have been discussed in Section 5.2.5.

In order to *predict* the solution values at boundary nodes, it is possible to adopt a point Jacobi method [55], that is, all entries of the preconditioner P are neglected except for the diagonal ones. Thus, an estimate of the solution can be computed by dividing the components of the defect vector $\mathrm{R}_i^{(m)}$ by the diagonal entries of the preconditioner so as to update the old iterate explicitly [142]:

$$\forall \mathbf{x}_i \in \Gamma: \quad \mathrm{U}_i^* = \mathrm{U}_i^{(m)} + \mathrm{P}_{ii}^{-1}\mathrm{R}_i^{(m)}, \qquad \mathrm{P}_{ii} = \mathrm{diag}\{P_{kk}^{i,i} : k = 1,\ldots,N_{\mathrm{U}}\} \tag{5.3.3}$$

Here, subscript *kk* denotes the diagonal block of the global operator $\mathrm{P} = \{P_{kl}\}$ and the superscript notation refers to the corresponding row/column entry within each block. In the next step, the provisional solution U_i^* is transformed to the local characteristic variables W_i^* so that boundary conditions can be imposed directly. Finally, the modified vector W_i^{**} is converted back to conservative variables so as to obtain the *corrected* values U_i^{**}. The flow chart [140] of variable transformations and manipulations to be performed for all boundary nodes is displayed in Figure 5.3.

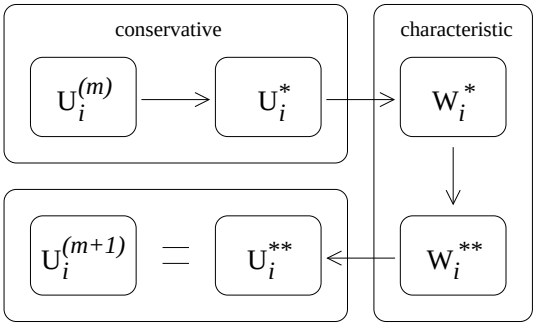


Fig. 5.3. Variable transformation for boundary nodes.

The corrected boundary values U_i^{**} may serve as standard Dirichlet boundary values which need to be imposed on the end-of-step solution. To this end, the corresponding entries of the defect vector are nullified and *all* off-diagonal coefficients residing in the rows of the preconditioner P which correspond to some boundary node are set to zero [140]:

$$\forall \mathbf{x}_i \in \Gamma: \quad P_{kl}^{i,j} := 0, \quad \forall k \neq l, \forall j, \qquad P_{kk}^{i,j} := 0, \quad \forall k, \forall j \neq i, \qquad \mathrm{R}_i^{(m)} := 0 \tag{5.3.4}$$

Moreover, the old solution values at boundary nodes are replaced by their precomputed counterparts U_i^{**} prior to solving the linear system (5.2.65), whereby the modified preconditioner (5.3.4) is adopted. This strategy implies that all nodal components of the increment vector $\Delta\mathrm{U}^{(m)}$ equal zero at the boundary so that the corrected boundary values carry over to the new solution:

$$\forall \mathbf{x}_i \in \Gamma: \quad \Delta\mathrm{U}_i^{(m)} = \mathrm{P}_{ii}^{-1}\mathrm{R}_i^{(m)} = 0 \qquad \Rightarrow \qquad \mathrm{U}_i^{(m+1)} = \mathrm{U}_i^{(m)} := \mathrm{U}_i^{**} \tag{5.3.5}$$

Note that there is no need for an *ad hoc* extrapolation of data from the interior since all computations are based on nodal values at the boundary. Of course, the concrete algorithm to be employed for the transition $\mathrm{W}_i^* \rightarrow \mathrm{W}_i^{**}$ depends on the type of boundary conditions. A viable implementation of numerical boundary conditions for the Euler equations is elaborated upon [140] (see Appendix A). An alternative approach for the treatment of characteristic boundary conditions is described in the next Section which follows the more general strategy presented by Shapiro [236].

5.3.4. Characteristic boundary conditions

It is worthwhile to summarize the different types of boundary conditions and give some guidance in selecting the variables to set from the physical boundary conditions and those to be evaluated from compatibility relations. To simplify the presentation, let us omit the subscript i and denote the predicted values of the density, momentum, and total energy by ρ^*, $(\rho\mathbf{v})^*$ and $(\rho E)^*$, respectively. Moreover, velocity, pressure and the speed of sound are given by the following expressions

$$\mathbf{v}^* = \frac{(\rho\mathbf{v})^*}{\rho^*}, \qquad p^* = (\gamma-1)\left((\rho E)^* - \rho^*\frac{|\mathbf{v}^*|^2}{2}\right), \qquad c^* = \sqrt{\frac{\gamma p^*}{\rho^*}} \tag{5.3.6}$$

The free stream state vector reads $\mathrm{U}_\infty = [\rho_\infty, (\rho\mathbf{v})_\infty, (\rho E)_\infty]$ and all derived quantities are indicated by subscript ∞. As we are about to see, a suitable combination of the provisional solution at the boundary and the free stream values is employed to evaluate the final boundary values U^{**}.

Open boundary in 2D

An important class of boundary conditions for the compressible Euler equations (5.1.1) is referred to as 'open' boundary conditions which are based on quasi one-dimensional characteristic theory. The governing equations are locally transformed from Euclidean space into the coordinate system (ξ,η) which is aligned normal to the boundary, whereby derivatives tangential to the boundaries are neglected. Moreover, the flow is assumed to be locally isentropic, that is, no entropy variation takes place. The diagonalization of the resulting system yields the characteristic equations [236]:

$$\begin{aligned} \frac{\partial Q}{\partial t}+(v_\xi+c)\frac{\partial Q}{\partial \xi}=0 \qquad & \qquad \frac{\partial v_\eta}{\partial t}+v_\xi\frac{\partial v_\eta}{\partial n}=0 \\ \frac{\partial R}{\partial t}+(v_\xi-c)\frac{\partial R}{\partial \xi}=0 \qquad & \qquad \frac{\partial s}{\partial t}+v_\xi\frac{\partial s}{\partial \xi}=0 \end{aligned} \tag{5.3.7}$$

In the above relation, v_ξ and v_η denote the velocities normal and tangential to the boundary, s is the entropy and the Riemann invariants Q and R are given by [158]

$$Q=v_\xi+\frac{2c}{\gamma-1}, \qquad R=v_\xi-\frac{2c}{\gamma-1} \tag{5.3.8}$$

These invariants are exact as long as there is no entropy variation normal to the boundary and they provide a reasonable approximation otherwise [236]. The above system represents a set of decoupled wave equations which model the propagation of characteristic variables normal to the boundary, whereby the direction is determined from the sign of the associated wave velocity.

To avoid confusion, let us emphasize the fact that $\mathbf{n}=[n_x,n_y]^T$ was defined as the *outward* unit normal vector so that a negative normal velocity $v_n=\mathbf{v}\cdot\mathbf{n}$ corresponds to an *ingoing* characteristic curve whereas an *outgoing* characteristic is associated with positive normal velocity. The wave speeds are given by the real eigenvalues of the projected Jacobian matrix $A_n=\mathbf{n}\cdot\mathbf{A}$:

$$\lambda_1=v_n-c, \qquad \lambda_2=\lambda_3=v_n, \qquad \lambda_4=v_n+c \tag{5.3.9}$$

The corresponding 1D Riemann invariants are given by the following expressions [236, 262]:

$$w_1=v_n-\frac{2c}{\gamma-1}, \qquad w_2=\frac{p}{\rho^\gamma}, \qquad w_3=u_\tau, \qquad w_4=v_n+\frac{2c}{\gamma-1} \tag{5.3.10}$$

where $u_\tau=\mathbf{v}\cdot\boldsymbol{\tau}$ denotes the velocity in the direction of the tangential vector $\boldsymbol{\tau}=[\tau_x,\tau_y]^T$. Then, a unified approach to the treatment of boundary conditions can be implemented as follows [236]:

1. Compute the invariants using the predicted quantities U^* *and* the free stream values U_∞:

$$W^*=[w_1^*,w_2^*,w_3^*,w_4^*], \qquad W_\infty=[w_{1,\infty},w_{2,\infty},w_{3,\infty},w_{4,\infty}] \tag{5.3.11}$$

2. Check the sign of the wave speed λ_k (from the interior v_n) and adopt the invariant based on the free stream values, or the invariant evaluated from the predicted boundary values:

$$W^{**}=[w_1^{**},w_2^{**},w_3^{**},w_4^{**}], \qquad w_k^{**}:=\begin{cases} w_{k,\infty} & \text{if} \quad \lambda_k<0 \\ w_k^* & \text{if} \quad \lambda_k\geq 0 \end{cases} \tag{5.3.12}$$

3. Transform vector W^{**} back to conservative variables and update the solution from (5.3.5).

The primitive variables can be calculated as follows [236] (see also Appendix A in [140]):

$$v_n^{**} = \frac{w_1^{**} + w_4^{**}}{2} \tag{5.3.13}$$

$$c^{**} = (\gamma - 1)\frac{w_4^{**} - w_1^{**}}{4} \tag{5.3.14}$$

$$\rho^{**} = \left(\frac{(c^{**})^2}{\gamma w_2^{**}}\right)^{\frac{1}{\gamma-1}} \tag{5.3.15}$$

$$p^{**} = \frac{\rho^{**}(c^{**})^2}{\gamma} \tag{5.3.16}$$

$$u_\tau^{**} = w_3^{**} \tag{5.3.17}$$

Note that the order of evaluation is crucial, for instance, the speed of sound computed in (5.3.14) is necessary to update the density by equation (5.3.15). Moreover, the new values c^{**} and ρ^{**} are a prerequisite for the correct evaluation of the pressure from (5.3.16). The above quantities are used to update the nodal vector of conservative variables at the boundary. The suggested procedure can be implemented as a 'black-box' tool for the treatment characteristic boundary conditions. In fact, the user may only prescribe the free stream quantities and the actual type of boundary condition is determined automatically by the algorithm. To get a better understanding of characteristic boundary conditions, let us consider the different inflow and outflow conditions which may be characterized based on the local Mach number $M = |v_n|/c$.

Supersonic open boundary

By definition, a supersonic flow implies that $|v_n| > c$. At a supersonic inlet $v_n < 0$ and all eigenvalues are negative so that all boundary conditions need to be prescribed. In fact, the free stream vector given in conservative variables U_∞ can be adopted so that no transition to the Riemann invariants and back-transformation is required. On the other hand, a supersonic outlet exhibits only outgoing characteristics ($v_n > c$) so that no physical boundary conditions may be imposed at all.

Subsonic inflow boundary

At a subsonic inlet $v_n < 0$ so that $\lambda_4 = v_n + c$ is positive while the other eigenvalues are negative. As a consequence, all Riemann invariants need to be specified except for w_4 which is computed from the provisional solution U^*. The normal and tangential flow velocities are given by [236]

$$v_n^{**} = \frac{w_{1,\infty} + w_4^*}{2}, \qquad v_\tau^{**} = w_{3,\infty} = v_{\tau,\infty} \tag{5.3.18}$$

and the remaining primitive variables to be transformed into conservative ones read as follows:

$$c^{**} = (\gamma - 1)\frac{w_4^* - w_{1,\infty}}{4}, \qquad \rho^{**} = \left(\frac{(c^{**})^2}{\gamma w_{2,\infty}}\right)^{\frac{1}{\gamma-1}}, \qquad p^{**} = \frac{\rho^{**}(c^{**})^2}{\gamma} \tag{5.3.19}$$

The remaining quantities of the corrected boundary values $\mathrm{U}^{**} = [\rho^{**}, (\rho\mathbf{v})^{**}, (\rho E)^{**}]^T$ are [236]

$$(\rho\mathbf{v})^{**} = \rho^{**}\mathbf{v}^{**}, \qquad (\rho E)^{**} = \frac{p^{**}}{\gamma - 1} + \frac{(v_n^{**})^2 + (v_\tau^{**})^2}{2} \tag{5.3.20}$$

where the vector $\mathbf{v}^{**} = v_n^{**}\mathbf{n} + v_\tau^{**}\boldsymbol{\tau}$ is determined from the normal and tangential velocities.

Subsonic outflow boundary

At a subsonic outlet $v_n > 0$ so that only the first eigenvalue $\lambda_1 = v_n - c$ is negative, and hence, the Riemann invariant w_1 has to be prescribed. All other characteristic variables retain their boundary values computed at the predictor step. Thus, the normal and tangential flow velocities yield

$$v_n^{**} = \frac{w_{1,\infty} + w_4^*}{2}, \qquad v_\tau^{**} = w_3^* = v_\tau^* \tag{5.3.21}$$

The corrected values of the density, the pressure and the speed of sound are given by

$$c^{**} = (\gamma - 1)\frac{w_4^* - w_{1,\infty}}{4}, \qquad \rho^{**} = \left(\frac{(c^{**})^2}{\gamma w_2^*}\right)^{\frac{1}{\gamma-1}}, \qquad p^{**} = \frac{\rho^{**}(c^{**})^2}{\gamma} \tag{5.3.22}$$

and they can be readily used to compute the final vector of conservative variables from expressions (5.3.20). For many problems, specifying the exit pressure p_s yields a more physical boundary condition than evaluating w_1 based on the free stream values. To this end, all outgoing Riemann invariants $w_2^* - w_4^*$ are evaluated from the provisional solution values U^* and the single incoming characteristic $w_{1,\infty}$ is set to the following value adopting the exit pressure p_S [236]:

$$w_{1,\infty} = \frac{4}{\gamma - 1}\sqrt{\frac{\gamma p_s}{\rho^*}\left(\frac{p^*}{p_s}\right)^{\frac{1}{\gamma}}} - w_4^* \tag{5.3.23}$$

Substitution of the above quantity into (5.3.13)–(5.3.16) leads to the desired relation $p = p_s$.

Solid wall boundary in 2D

The reflecting boundary conditions state that no flow penetrates a solid wall, that is, the normal velocity at the wall vanishes. All eigenvalues except $\lambda_1 = -c$ are non-negative so that exactly one (bi-)characteristic enters the domain, and hence, the natural boundary condition

$$\mathbf{v} \cdot \mathbf{n} = 0 \tag{5.3.24}$$

is the only constraint to be prescribed. All other information for defining the state variables on the wall needs to be recovered from within the flow domain. In essence, there exist two common approaches to implement wall boundary conditions. In the weak formulation, the slip condition is only enforced in an integral sense by means of the flux function which reduces to

$$\mathbf{F} \cdot \mathbf{n} = \begin{bmatrix} 0 \\ p n_x \\ p n_y \\ 0 \end{bmatrix} \tag{5.3.25}$$

normal to the wall. It is important to note that zero mass flux ($v_n = 0$) is not satisfied exactly by the nodal solution values at the boundary. In a practical implementation, the no-penetration condition can be enforced by modifying the inviscid fluxes as proposed by Shapiro [236]

$$F^1 = \begin{bmatrix} \rho v_1^* \\ \rho v_1 v_1^* + p \\ \rho v_2 v_1^* \\ \rho H v_1^* \end{bmatrix} \qquad F^2 = \begin{bmatrix} \rho v_2^* \\ \rho v_1 v_2^* \\ \rho v_2 v_2^* + p \\ \rho H v_2^* \end{bmatrix} \tag{5.3.26}$$

and enforcing flow tangency once the solution at the next time step has been computed. The corrected velocity $\mathbf{v}^* = \mathbf{v} - (\mathbf{n} \cdot \mathbf{v})\mathbf{n}$ is defined such that the total flux normal to the boundary satisfies relation (5.3.25) and an explicit formula for the modified velocities is as follows [236]:

$$v_1^* = v_1 - (v_1 n_y - v_2 n_x) n_y \tag{5.3.27}$$

$$v_2^* = v_2 - (v_2 n_x - v_1 n_y) n_x \tag{5.3.28}$$

Some additional weak formulations are considered in the finite volume context, e.g. [16]. Their main drawback is that the zero mass flux condition is only satisfied in an integral sense and a non-vanishing normal velocity may be present in the nodal boundary values.

As an alternative, wall boundary conditions can be imposed in a strong sense. In essence, the state variables at the boundary are computed by the standard scheme and modified *a posteriori* so as to reflect the zero mass flux condition exactly. For the time being, let us neglect the node-based predictor-corrector algorithm presented in Section 5.3.3 and assume the state vector U^* be given for each boundary node. It is common practice to employ reflecting boundary conditions which amounts to introducing artificial ghost cells or states on the part of the boundary corresponding to the solid wall. In these ghost elements, all components are set equal to the interior values except for the normal velocity which is negated. If both the interior and the ghost states are passed to a Riemann solver, the resulting normal velocity vanishes due to the symmetry of the reflection [251]. This approach works well for straight-sided bodies but unphysical effects such as spurious solution variations near the boundary are observed for curved domains and/or higher-order approximations. As a remedy, Krivodonova and Berger [126] considered high-order schemes based on the discontinuous Galerkin method and suggested so-called curved boundary conditions.

It makes sense to adopt this approach also for (bi-)linear finite element approximations if an analytical or at least more accurate than straight-sided description of the boundary is available. Consider the situation at the boundary depicted in Figure 5.4. Obviously, the normal vector $\vec{N}(\mathbf{x}_i)$ for the finite element Ω_k does not coincide with vector $\mathbf{n}(\mathbf{x}_i)$ normal to the boundary Γ at point $\mathbf{x}_i$. Let us convert the conservative variables computed at node i into their primitive counterparts, i.e. $\mathrm{v}_i = [\rho_i, \mathbf{v}_i, p_i]^T$, and define the ghost state at the wall boundary as $\mathrm{v}_i^g = [\rho_i, \mathbf{v}_i^g, p_i]^T$. The velocity vector $\mathbf{v}(\mathbf{x}_i)$ is reflected to the ghost state with respect to the analytical normal $\mathbf{n}(\mathbf{x}_i)$ so that the normal and tangential components relative to the physical geometry are given by

$$v_n^g = -v_n, \qquad v_\tau^g = v_\tau \tag{5.3.29}$$

respectively. The above quantities are converted back into the Euclidean space so that the modified velocity components of the ghost state at boundary node i read as follows [126]

$$v_1^g = v_1(n_y^2 - n_x^2) - 2n_x n_y v_2, \qquad v_2^g = v_2(n_x^2 - n_y^2) - 2n_x n_y v_1 \tag{5.3.30}$$

In the above expression, n_x and n_y denote the x- and y-component of the *physical* normal vector

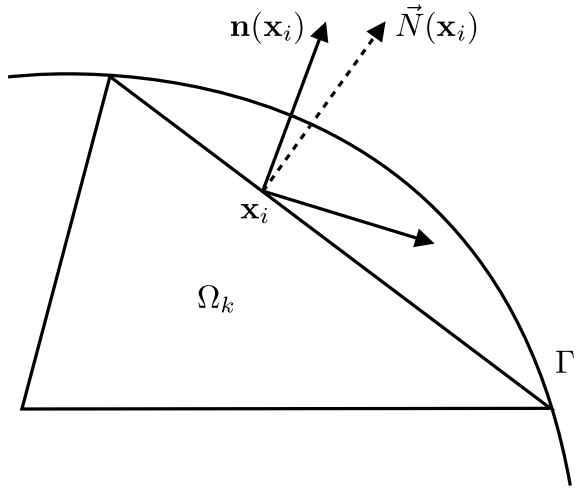


Fig. 5.4. Curvature boundary conditions at the boundary element Ω_k.

n, respectively. Finally, the Riemann problem for the two states v_i and v_i^g is solved in the direction of the *approximate* normal vector $\vec{N}(\mathbf{x}_i)$. The uniform density and pressure values at both states ensure that the exact solution of the Riemann problem consists of either two shocks or two rarefaction waves [251], whereby the normal velocity at the interface is given by

$$v_n(\mathbf{x}_i) = \frac{1}{2}(v_N + v_N^g) \tag{5.3.31}$$

Depending on the sign of $\mathbf{v} \cdot \vec{N}$, the tangential velocity $v_\tau(\mathbf{x}_i)$ is defines as v_T or v_T^*. The resulting velocity may not be exactly orthogonal to the approximate normal $\vec{N}$ to the boundary, whereby the error depends on the original velocity vector and the curvature of the boundary [126].

In our implementation of solid-wall boundaries we make use of a similar approach which is tailored to finite element discretizations. Note that for (bi-)linear elements the degrees of freedom are located in the cell vertices. The edges of the elements yield a polygonal approximation of the physical boundary, whereby each boundary node has two neighboring segments as illustrated in Figure 5.5. Hence, the numerical normal $\vec{N}(\mathbf{x}_i)$ at vertex $\mathbf{x}_i$ can be defined as the weighted average

$$\vec{N}(\mathbf{x}_i) := \frac{w_{k_1}\vec{N}_{k_1} + w_{k_2}\vec{N}_{k_2}}{w_{k_1} + w_{k_2}} \tag{5.3.32}$$

which still needs to be normalized so as to obtain an approximation to the outward unit normal vector at point $\mathbf{x}_i$. Here, $\vec{N}_{k_1}$ and $\vec{N}_{k_2}$ denote the uniquely defined normal vectors of the adjacent elements Ω_{k_1} and Ω_{k_2}, respectively. The weighting coefficients w_{k_1} and w_{k_2} may be chosen inversely proportional to the length of the edges left and right to the boundary vertex $\mathbf{x}_i$. In general, the approximate vertex normal (5.3.32) will not coincide with the physical normal $\mathbf{n}(\mathbf{x}_i)$. As a remedy, the presented algorithm developed by Krivodonova and Berger [126] for the implementation of wall boundary conditions in curved geometries can be adopted and applied node-by-node.

In practice, poor convergence and various numerical difficulties are to be expected if the initial data fail to satisfy the no-penetration condition (5.3.24). In fact, the abrupt change of the normal velocity results in an *impulsive start* which is physically impossible due to inertia [140]. This problem was studied by Lyra [172, 173] who promoted the use of the more robust condition

$$\mathbf{n} \cdot \mathbf{v}^{n+1} = \omega \mathbf{n} \cdot \mathbf{v}^n, \qquad \omega \in (0, 1] \tag{5.3.33}$$

In essence, the fluid is allowed to seep through wall during the startup, whereby the relaxation parameter ω controls its permeability. Thus, the immediate blow-up of solution values at solid walls is unlikely to occur. By virtue of expression (5.3.33), the normal velocity is gradually driven to zero as the flow evolves so that (5.3.24) is satisfied in the steady-state limit.

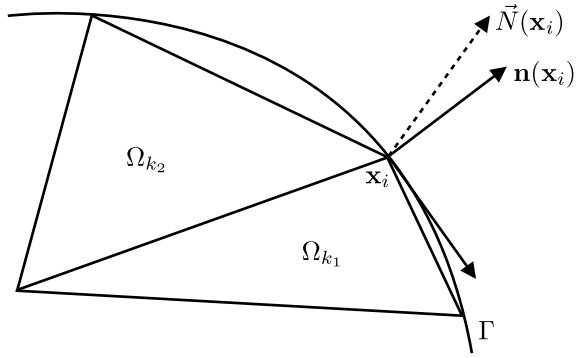


Fig. 5.5. Curvature boundary conditions at the boundary point $\mathbf{x}_i$.

The relaxed no-penetration condition can be implemented in the framework of curved boundary conditions developed by Krivodonova and Berger [126]. In particular, the velocity components (5.3.30) of the ghost state need to be augmented by the contributions from the old time step

$$v_1^g := v_1^g + 2\omega\,(v_1^n n_x^2 + v_2^n n_x n_y), \qquad v_2^g := v_2^g + 2\omega\,(v_2^n n_y^2 + v_1^n n_x n_y) \tag{5.3.34}$$

and the Riemann problem for the two states v_i and v_i^g is solved as in the standard approach. Note that the resulting normal velocity at the interface satisfies formula (5.3.31) so that

$$v_n(\mathbf{x}_i) \approx \frac{1}{2}(v_n + v_n^g) = \omega\,\mathbf{n}\cdot\mathbf{v}^n \tag{5.3.35}$$

is a good approximation to the relaxed no-penetration condition (5.3.33). The above relation can be verified by simple manipulations exploiting the unit length of $\mathbf{n}$, that is $n_x^2 + n_y^2 = 1$.

The use of an (approximate) Riemann solver for each boundary node may sound complicated at first glance, but in fact, the uniform density and pressure values at both states allow for significant simplifications. As a starting point, consider the following pressure equation [251]

$$f(p) = f_L(p, \mathrm{v}_L) + f_R(p, \mathrm{v}_R) + v_R - v_L \tag{5.3.36}$$

It relates the left and right states by virtue of the auxiliary function [251]

$$f_k(p) = \begin{cases} (p - p_k)\left[\frac{2}{(\gamma+1)\rho_k}\right]^{\frac{1}{2}}\left[p + \left(\frac{\gamma-1}{\gamma+1}\right)p_k\right]^{-\frac{1}{2}} & \text{if} \quad p > p_k \qquad \text{(shock)} \\ \frac{2c_k}{\gamma-1}\left[\left(\frac{p}{p_k}\right)^z - 1\right], \qquad z = \frac{\gamma-1}{2\gamma} & \text{if} \quad p \le p_k \qquad \text{(rarefaction)} \end{cases} \tag{5.3.37}$$

where ρ_k, p_k and c_k denote the density, pressure and the speed of sound at the two states $k = L$ and $k = R$. For uniform pressure ($p_L = p_R$), both contributions of $f_k(p)$ are equal so that either two shocks or two rarefaction waves can occur. The unknown pressure p^* can be determined as the root of the equation $f(p) = 0$. Toro [251] suggested the use of a Newton-Raphson method. Due to the uniform pressure and density values at the left and right states, it suffices to compute a provisional pressure from the simple primitive variable Riemann solver (PVRS) and decide – based on the ratio p_{PVRS}/p – whether to employ the two-rarefaction Riemann solver (TRRS) or the two-shock Riemann solver (TSRS) to obtain an initial guess p^* for the Newton iteration.

For non-vacuum initial data, there exists a unique solution $p^* > 0$ for the pressure such that $f(p^*) = 0$, provided that the two states satisfy the pressure positive condition [251]

$$\frac{2c_L + 2c_R}{\gamma - 1} > v_R - v_L \tag{5.3.38}$$

In the framework of the relaxed no-penetration condition (5.3.33), this may be accomplished by adjusting the relaxation parameter accordingly if the above inequality is violated for $\omega = 0$. In any case, the density value at the boundary can be calculated from either state as follows [251]

$$\rho^* = \begin{cases} \rho\left[\dfrac{\frac{p^*}{p_k} + \frac{\gamma-1}{\gamma+1}}{\frac{\gamma-1}{\gamma+1}\frac{p^*}{p_k} + 1}\right] & \text{if} \quad p^* > p_k \qquad \text{(shock)} \\ \rho\left[\dfrac{p^*}{p_k}\right]^{\frac{1}{\gamma}} & \text{if} \quad p^* \le p_k \qquad \text{(rarefaction)} \end{cases} \tag{5.3.39}$$

A comprehensive discussion on Riemann solvers and a complete solution is due to Toro [251].

5.4. Numerical examples for the compressible Euler equations

The examples that follow illustrate the performance of the algebraic flux correction scheme applied to some academic test problems for inviscid flows at different Mach numbers. In this thesis, we consider characteristic limiters of TVD type which are favorable for the treatment of stationary flows. The design of flux limiting schemes for time-dependent problems is addressed in the collection of articles [138] and the references therein. A recent paper by Kuzmin [136] dwells on FEM-FCT algorithms with flux linearization including numerical results for the scalar transport equation as well as inviscid flow problems. For a comprehensive description of the latest research activities on AFC schemes for hyperbolic systems, the interested reader is also referred to [132].

Another aim of this section is to validate the usability of recovery-based error indicators being applied to the compressible Euler equations and to illustrate the potential of the new mesh adaptation strategy for inviscid flow computations. Its predecessor which is restricted to pure triangular meshes was considered in [185, 186] for a few stationary test problems. It was not capable to produce numerical meshes which possessed an intrinsic hierarchy, and moreover, it did not provide mechanism to preserv the initial triangulation. These drawbacks may be tolerable for steady state computations but the novel approach to dynamic mesh adaptation is preferable for the simulation of transient flows. At the current stage of development, there are still many aspects that call for further investigation and an extension to three dimensions necessary for the simulation of industrial applications is outstanding. However, the numerical examples that follow indicate that algebraic flux correction schemes utilized in combination with local mesh adaptivity constitute a promising approach to compute accurate solutions to inviscid flows at reasonably cost.

5.4.1. 5° Converging channel flow

Our first test problem deals with supersonic flow at $M_\infty = 2$ though a converging channel, where the bottom wall is sloped at $\theta = 5^\circ$ as depicted in Figure 5.6. This benchmark was also used by Shapiro [236] to compare some variants of the Galerkin method adopting bilinear and biquadratic finite element. The supersonic stream is alternately deflected from the bottom and top walls giving rise to a multi-reflected shock wave which separates five zones of essentially constant states. Table 5.1 shows the values of pressure, density and Mach number for the exact solution [236] which can be computed analytically by means of shock wave theory [7]. The oblique shock wave separating regions I and II exhibits the inclination angle $\beta = 34.265^\circ$, that is, the angle between the shock wave and the sloped bottom wall equals $\beta - \theta = 29.265^\circ$. The strength of the reflected shocks and all derived quantities can be calculated from the particular upstream Mach number M_{II}, M_{III}, etc. and the slope angle $\theta = 5^\circ$ (see Anderson [7], pp. 111–120). For the numerical simulation,

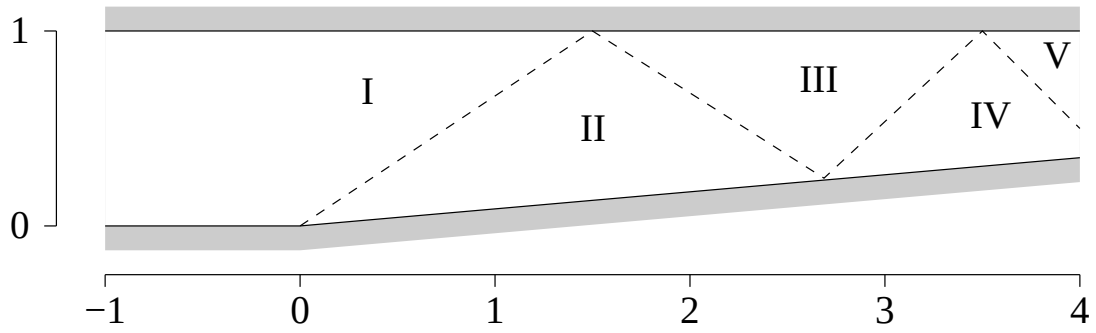


Fig. 5.6. 5° Converging channel flow: geometry.

Region	Density	Pressure	Mach number
I	1.000	0.714	2.000
II	1.216	0.940	1.821
III	1.463	1.218	1.649
IV	1.747	1.563	1.478
V	2.081	1.998	1.302

Tab. 5.1. 5° Converging channel flow: exact solution values.

each cell of the initial 8×40 grid depicted in Figure 5.7 (a) is subdivided twice to obtain the computational *Grid 1* which consists of 5,120 bilinear finite elements and comprises 5,313 degrees of freedom. *Grid 2* has the same number of vertices, whereby each quadrilateral is subdivided into two triangles by connecting the lower-left node with the upper-right one giving rise to 10,240 linear finite elements. This benchmark problem can be simulated without placing some extra elements upstream of the wedge, that is, the computational domain can be restricted to $0 \leq x \leq 4$. Below we will extend this test problem and consider supersonic flows at higher Mach numbers, where the use of auxiliary elements located upstream of the wedge is necessary to calculate the solution values at the lower wall correctly (see also Shapiro [236]). To simplify the comparison of numerical results, the same grids are used for all supersonic channel flow computations.

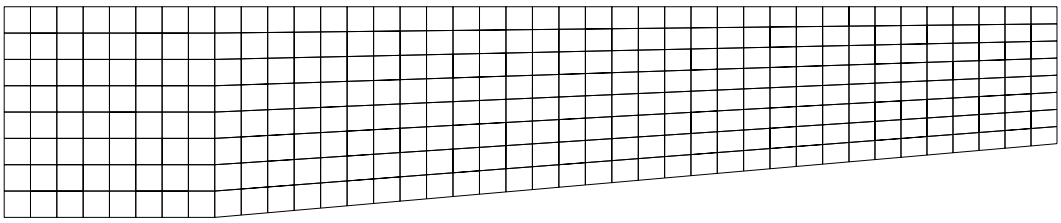
The numerical solutions were computed by the high-resolution FEM-TVD scheme [135]. In essence, the group finite element formulation [78] was employed to construct the high-order Galerkin discretization (5.2.4). It was turned into the non-oscillatory high-resolution scheme (5.2.47) by adding artificial viscosities and performing characteristic flux limiting of TVD type so as to remove excessive diffusion in regions of smooth solution without generating non-physical undershoots and overshoots (cf. Section 5.2). Solving the stationary problem directly was impracticable so that the vector of nodal solution values was initialized by the free stream data [236]

$$\rho_\infty = 1, \qquad \mathbf{v}_\infty = (2,0), \qquad p_\infty = \gamma^{-1} \approx 0.714 \tag{5.4.1}$$

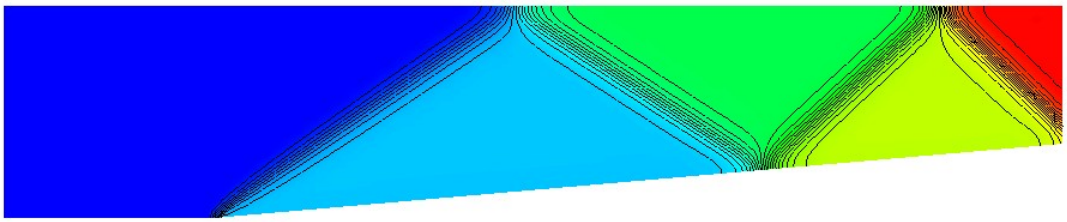
and marched into a steady state adopting the fully implicit backward Euler method. We employed the segregated approach presented in Section 5.2.5 for solving the nonlinear algebraic equations. The diagonal blocks of the preconditioner (5.2.64) were constructed using scalar dissipation proportional to the spectral radius of the cumulative Roe matrix. The free stream quantities (5.4.1) also served as boundary conditions at the supersonic inlet, whereas no boundary condition were prescribed at the supersonic outlet. The no-slip condition at the lower and upper walls was imposed in the strong sense following the implementation details given in Section 5.3.4.

The numerical solutions computed on Grids 1 and 2 are depicted in Figure 5.7 (b) and (c), respectively. It is hardly possible to see a difference between the density distributions computed using linear and bilinear finite elements. The resolution of the shock wave is acceptable taking into account that the grids were relatively coarse. It is worth mentioning that the nodal values of the approximate solution coincide with their exact counterparts in the 'interior' of each region. Conversely, the shock wave is strongly smeared by numerical diffusion and the transitions between the five zones of uniform flow are anything else but discontinuous if the low-order method is employed based on scalar dissipation; cf. Figure 5.7 (d). The use of tensorial dissipation alleviates the devastating effect of excessive dissipation to some extent but it does not provide a sufficiently accurate resolution of the shock wave. The solution profiles computed by the low-order method using Q_1 elements are not presented since they are similar to Figure 5.7 (d) and (e).

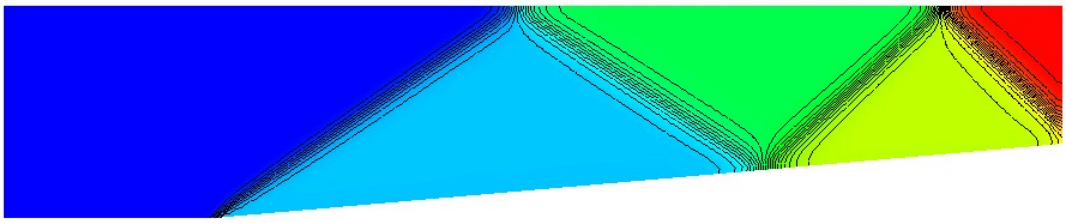
(a) Initial coarse grid, 320 elements, 391 vertices



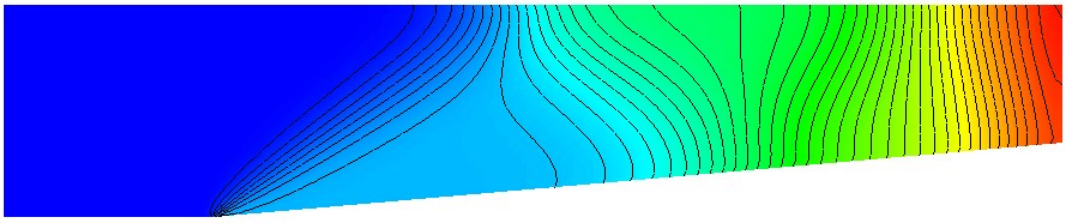
(b) 5,120 Q_1 elements (Grid 1), FEM-TVD scheme



(c) 10,240 P_1 elements (Grid 2), FEM-TVD scheme



(d) 10,240 P_1 elements (Grid 2), low-order method, scalar dissipation



(e) 10,240 P_1 elements (Grid 2), low-order method, tensorial dissipation

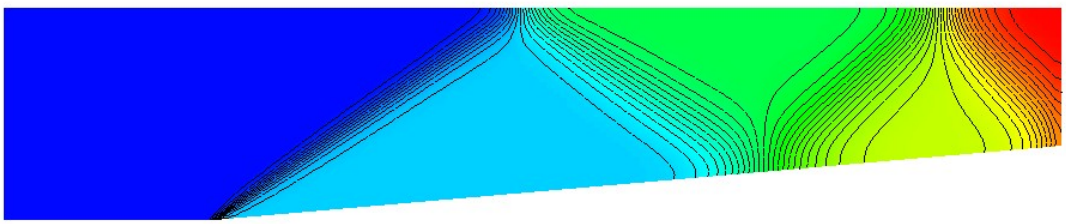


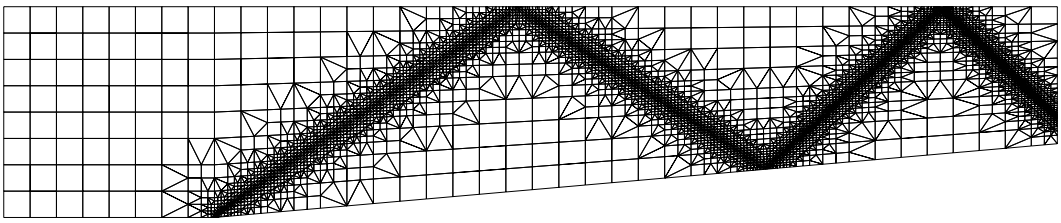
Fig. 5.7. 5° Converging channel flow at $M_\infty = 2$ on regular grid (50 contours).

In light of the above, algebraic flux correction constitutes a valuable tool for the design of high-resolution schemes for inviscid flows on quadrilateral and triangular finite elements. In fact, the smearing of discontinuities due to numerical dissipation is confined to a thin layer of 1–3 cells in the vicinity of the shock. This observation suggests local mesh adaptivity as an economical approach to improve the accuracy of the solution without the need to globally refine elements everywhere in the domain. The computational mesh shown in Figure 5.8 (a) was constructed from the initial coarse grid by computing a steady state solution and refining the mesh making use of recovery-based error indication techniques. This procedure was repeated prescribing the maximum number of three refinement levels until a 'converged' mesh was obtained.

Owing to the fact that the density ρ is discontinuous at shocks and contact discontinuities and varies smoothly across rarefaction waves [157], it constitutes a reasonable key variable for mesh adaptation [236]. In our computations, we utilized the L_2-projection method (4.3.15) to recover improved slope values for the flow density and compared them to its consistent finite element gradient. The predicted gradient error for the indicator variable was decomposed into element contributions which were used to steer mesh adaptation based on the equidistribution strategy described in Section 4.4. In particular, elements were marked for refinement and re-coarsening using the parameters $\zeta_{\max} = 10\%$ and $\zeta_{\min} = 5\%$ as tolerances for the percentage error. Since we were interested in the steady-state limit, no protection layers were introduced, that is, mesh adaptivity was directly based on the generated element marker without post-processing.

The benefits of local mesh adaptation are demonstrated by the accurate resolution of the density distribution shown in Figure 5.8 (b). Here, 50 contour levels were used to visualize the qualitative improvements of the adaptively computed solution compared to that depicted in Figure 5.7 (c). In fact, the five zones of uniform flow are sharply delimited from each other by steep fronts. Remarkably, the computational grid has 20% less degrees of freedom as the structured Grid 1 and features nearly the same number of elements. However, discontinuities are resolved with higher accuracy by concentrating more vertices in the vicinity of the shock wave and preserving the initial coarse grid in regions, where the solution is constant. Note that a structured grid would require 20,480 quadrilaterals to provide the same grid spacing. The accuracy of the discontinuities is also illustrated by the solution profiles depicted in Figure 5.9. The density values were evaluated in

(a) Computational mesh: 5,145 elements, 4,451 vertices



(b) Density isolines, FEM-TVD scheme, mixed Q_1/P_1 elements

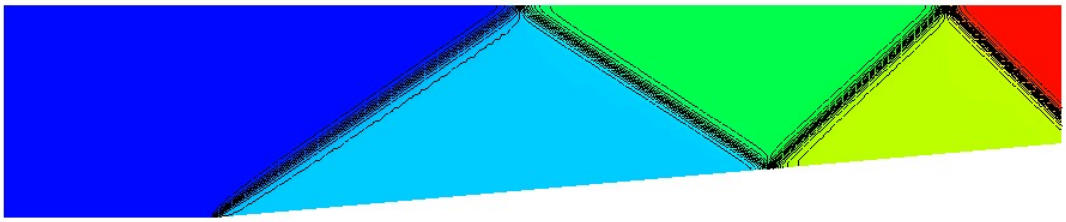


Fig. 5.8. 5° Converging channel flow at $M_\infty = 2$ on locally adapted mesh (50 contours).

(a) Density along cutline $y = 0.6$ (b) Surface Mach number

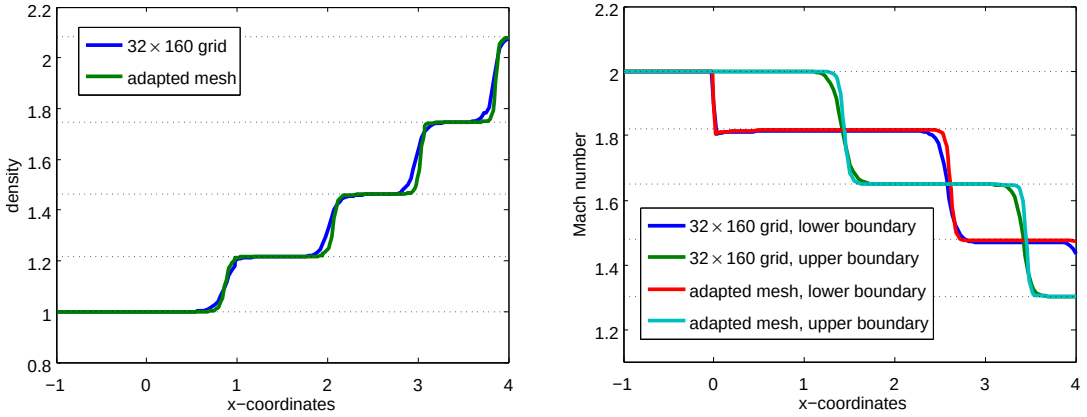
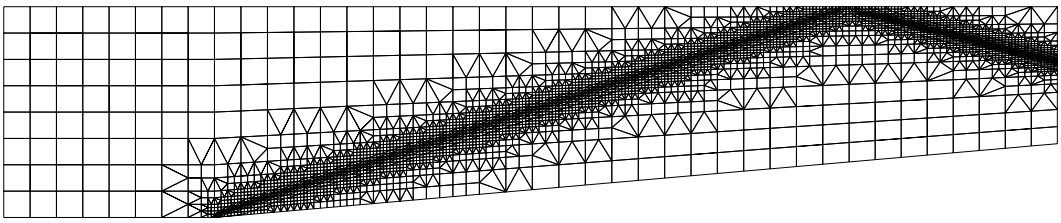


Fig. 5.9. 5° Converging channel flow at $M_\infty = 2$: 32×160 grid vs. locally adapted mesh.

the interior of the domain along the cutline $y = 0.6$ so that all regions I–V of constant states are traversed from left to right. Furthermore, the nodal values of the Mach number were sampled at the upper and lower wall boundary to compare the accuracy improvement due to adaptive mesh refinement. In fact, all profiles are in very good agreement with the value of the exact quantities given in Table 5.1 which are denoted by dotted lines in the particular diagrams.

Let us extend this test problem to higher Mach numbers and consider the supersonic flow at $M_\infty = 4$ leaving all other parameters unchanged. The resulting oblique shock wave exhibits the inclination angle $\beta = 18^\circ$ and gives rise to the downstream Mach number $M_{II} = 3.637$. It is once reflected downward by the upper wall before it leaves the computational domain at $M_{III} = 3.319$. The locally adapted mesh and the density distribution predicted by the high-resolution FEM-TVD scheme are depicted in Figure 5.10, whereby 20 contour levels are used for the visualization. The resolution of the shock wave is crisp, and moreover, the values for the Mach number are in very good agreement with their analytic counterparts. It is worth mentioning that the placement of

(a) Computational mesh: 4,424 elements, 3,867 vertices



(b) Density isolines, FEM-TVD scheme, mixed Q_1/P_1 elements

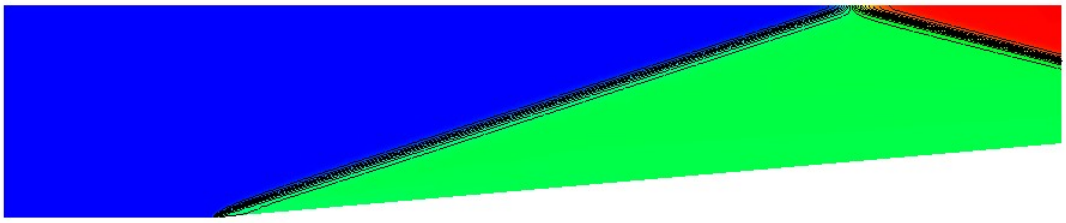


Fig. 5.10. 5° Converging channel flow at $M_\infty = 4$ on locally adapted mesh (20 contours).

elements upstream of the wedge is crucial for this configuration. Otherwise, the flow quantities on the lower wall were not calculated correctly. On the other hand, only 100 'auxiliary' cells (about 2%) are located in the half-plane $x \leq 0$ which is negligible compared to the total number of elements. If global mesh refinement were performed, 64×64 cells (i.e. 1/5 of the total number of elements) would be located left to the wedge, where the flow equals the free stream values.

Let us investigate the ability of the adaptation procedure to react to dynamic changes in the quasi-stationary flow field. The free stream Mach number is therefore modulated in time

$$M_\infty(t) = 3 - \cos(\omega t), \qquad t \geq 0 \tag{5.4.2}$$

which yields a supersonic flow that is slowly pulsating at the periodic rate $\omega = 0.05$. The supersonic $M_\infty = 2$ flow is recovered at $t = 2\pi k/\omega$ for $k \in \mathbb{N}_0$ while $t = (2\pi k - \pi)/\omega$ corresponds to the free stream Mach number $M_\infty = 4$. The simulation was started from the locally adapted mesh depicted in Figure 5.8 (a) and ran for a complete period, that is, $0 \leq t \leq 2\pi/\omega \approx 125.66$.

The sequence of locally adapted meshes for $t = k\pi/8$ with $k = 1, \ldots, 7$ is shown in Figure 5.12, and the computational grid presented in Figure 5.10 (a) corresponds to $t = \pi/\omega$. The acceleration of the flow in the first half-period flattens the inclination angle of the oblique shock wave, and hence, it passes through the converging channel being reflected just once from the upper wall as the Mach number tends to 4. It is clearly visible that the dynamic mesh adaptation algorithm has the ability to keep track of the shock movement such that the grid is re-coarsened in regions, where locally refined cells are no longer required. The sequence of locally adapted grids produced in the second half-period $\pi \leq t\omega \leq 2\pi$ are not displayed since they resemble the presented ones in reverse order. In essence, the flow slows down step-by-step so that the shock wave is reflected multiple times from the upper and lower wall before it leaves the channel.

The temporal evolution of the vertex number is illustrated in Figure 5.11, whereby the simulation time is normalized to the interval $[0, 2\pi]$. The number of vertices decreases during the acceleration of the flow ($0 \leq t \leq \pi$) and returns to its initial value as the free stream Mach number approaches $M_\infty = 2$. We also performed a long-time simulation over multiple periods and observed the same behavior of the vertex number. It is worth mentioning that the CPU time spent on dynamic mesh adaptivity is less than 1% of the total computing time so that the benefit of using flow adjusted grids is not destroyed by the need to generate them in time-consuming grid refinement and re-coarsening procedures. We therefore belief, that our dynamic mesh adaptation algorithm is ready for use in modern compressible flow solvers for unstructured triangulations.

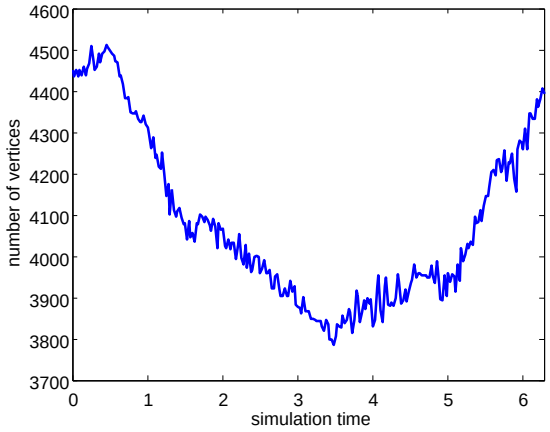


Fig. 5.11. 5° Converging channel flow: evolution of the vertex number.

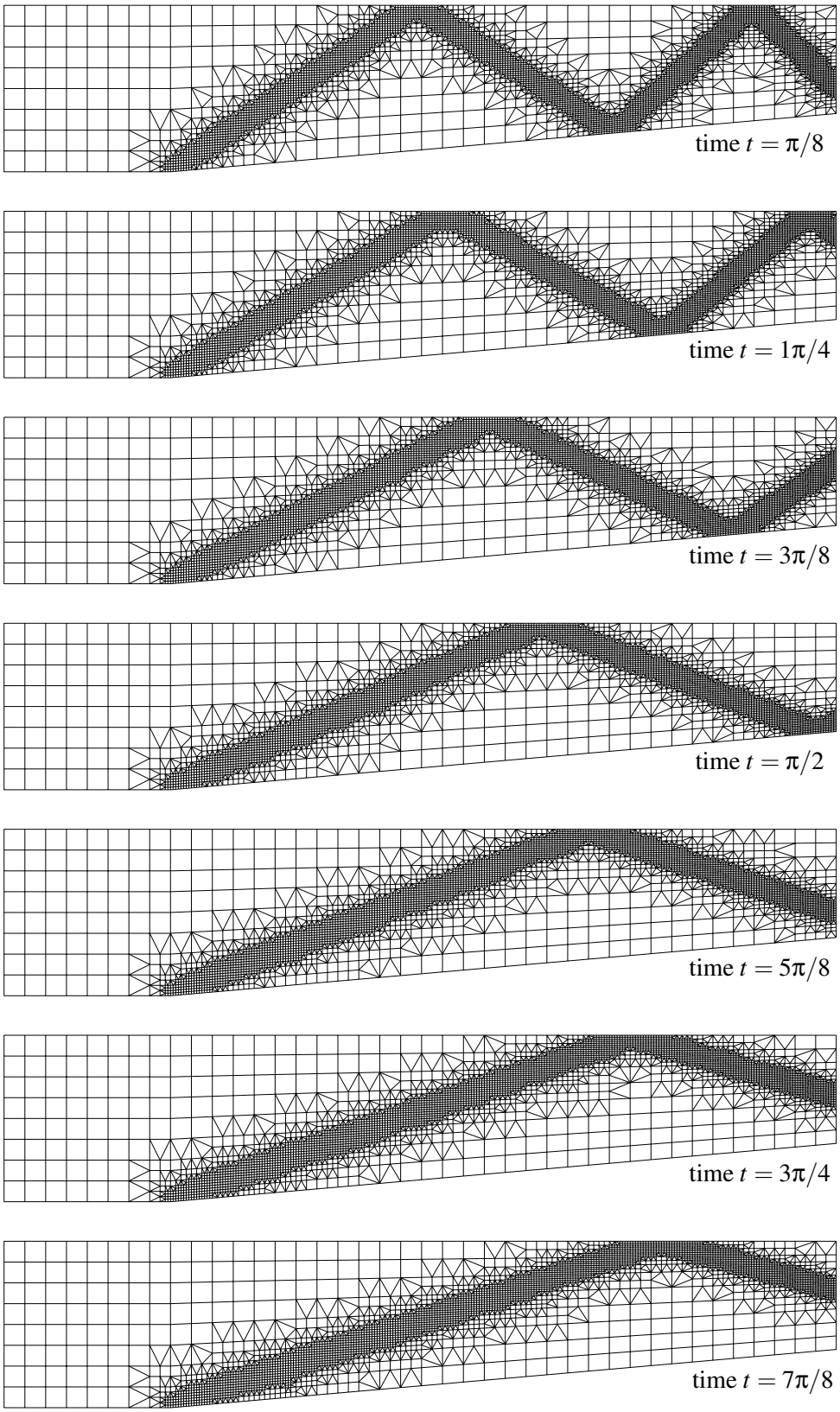


Fig. 5.12. 5° Converging channel flow: dynamic mesh adaptation.

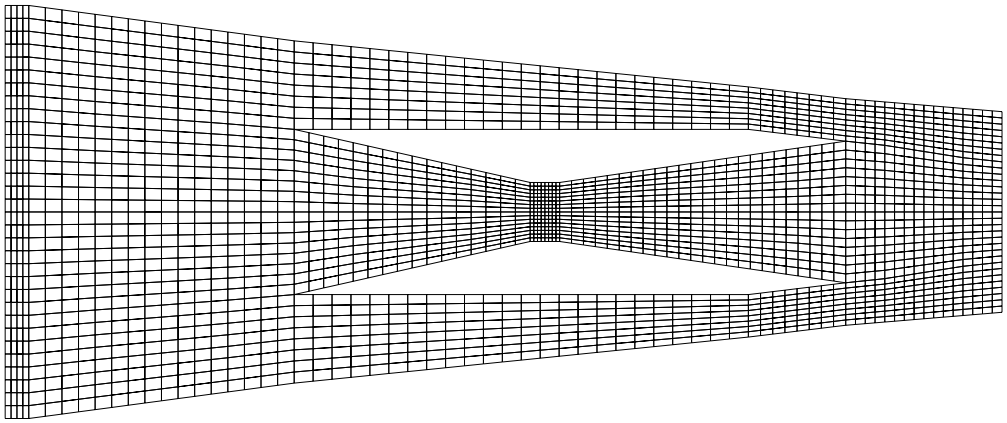
5.4.2. Supersonic scramjet inlet

Another interesting two-dimensional benchmark configuration was introduced in [49]. The problem starts with a steady Mach 3 flow in a narrowing channel which exhibits two sharp-cornered internal obstacles giving rise to multiple shock wave reflections. The geometry of the wind tunnel and the initial coarse grid is depicted in Figure 5.13 (a). The computational domain is delimited by three piecewise linear boundary components passing through the following set of points [75]

	upper wall						upper obstacle				
x_i	0.0	0.4	4.9	12.6	14.25	16.9	4.9	8.9	9.4	14.25	12.6
y_i	3.5	3.5	2.9	2.12	1.92	1.7	1.4	0.5	0.5	1.2	1.4

Owing to the symmetry of the scramjet inlet, the horizontal position of the polygons remains unchanged, whereas the y-coordinates of the corner points for the lower wall/obstacle are given by $-y_i$. A supersonic gas flow with density 1, pressure $1/(\gamma M_{\text{in}}^2)$ and velocity 1 is continually fed in from the left boundary along $x = 0$, where $M_{\text{in}} = 3$ denotes the inlet Mach number [75]. The same values are used to initialize the nodal solution vector which is somehow artificial but it makes the problem very easy to set up. The exit velocity is always supersonic, and hence, no boundary conditions need to be prescribed along $x = 16.9$. Moreover, reflecting boundary conditions are applied at the upper and lower walls and the internal obstacles. A detailed numerical analysis of

(a) Initial computational grid: 2,432 elements, 2,611 vertices



(b) Mach number isolines, FEM-TVD scheme, Q_1 elements

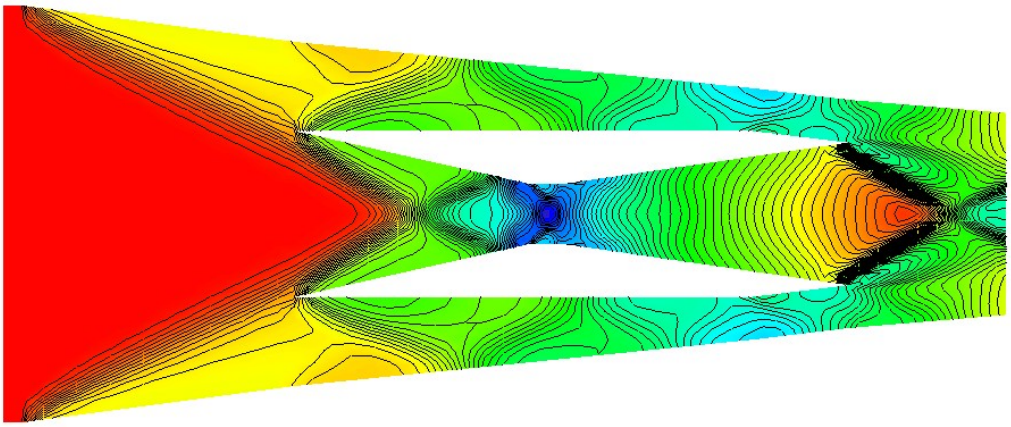


Fig. 5.13. Scramjet inlet at Mach 3 on regular grid (50 contours).

scramjet inlets for different inlet Mach numbers and flow angles (termed yaws) is presented by Shapiro [236]. Following his suggestions, grid singularities such as the leading and trailing edges of internals are treated as interior nodes, that is, no boundary conditions are imposed at all.

As in the previous example, the stationary solution displayed in Figure 5.13 (b) was produced by means of pseudo time-stepping based on the fully implicit backward Euler method, whereby the characteristic FEM-TVD scheme [135] was adopted for the spatial discretization. The qualitative behavior of the strong shocks is reproduced correctly on the, admittedly, coarse grid but the interplay of weak shocks is completely missed. Moreover, spurious oscillations were note created neither for the components of the state vector nor for dependent flow variables such as the Mach number. Moreover, no unphysical phenomena such as the development of an artificial boundary layer engendered by the leading edges of the internal obstacles are visible.

To improve the resolution of shock waves, adaptive mesh refinement was performed based on the hierarchical adaptation strategy developed in Section 4.6. The flow field exhibits various features of different intensities, and therefore, Löhner's [161] dimensionless indicator was used to identify elements which need to be refined, whereby the maximum number of three refinement levels was prescribed. The density served as key variable and the noise filter in expression (4.2.42) was set to $C_n = 0.005$. Since the density is always greater than unity for this benchmark, no absolute tolerance for the solution average had to be prescribed. The nodal quantities e_i were distributed to the elements by computing their arithmetic mean values. In our numerical experiments we observed that the indicator was sensitive to the choice of parameters, e.g., larger values for the noise filter C_n wiped out essential flow features. A viable strategy was to refine cells if the error indicator exceeded $tol_{\text{ref}} = 0.15$ and to adopt the value $tol_{\text{crs}} = 0.05$ for re-coarsening.

The density and Mach number distribution of the approximate solution as well as the 'converged' computational mesh are depicted in Figure 5.14 (a)–(c). The interaction of shock waves is reproduced reasonably well while regions of essentially uniform flow do not require excessive mesh refinement. Furthermore, the solution profiles and the computational grids are symmetric with respect to the x-axis. Let us dwell on some interesting features of the flow topology [236]. The shocks created at the inlet entrance are reflected six times from the upper and lower walls and the particular boundary of the internal obstacles, whereas the strut leading edge shocks experience three reflections passing through the internal chamber. Note that bending of the entrance shock towards the center line (barely visible) occurs when its sixth reflection intersects with the third reflection of the corresponding interior shock wave. Another interesting flow feature is the slip line off the trailing edge which also causes bending of the entrance shock towards the center line.

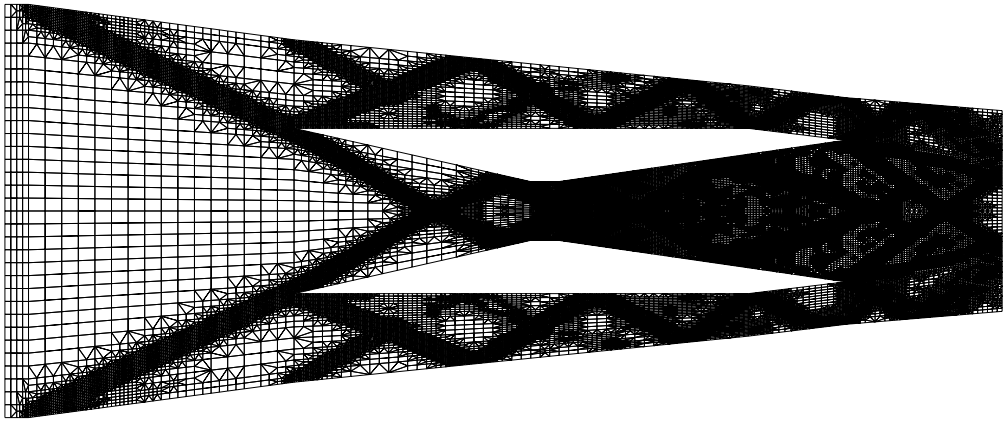
Let us proceed to the Mach number distribution which is in good agreement with the numerical results published by other researchers [49, 75] who made use of anisotropic mesh adaptation. The locally adapted computational mesh shown in Figure 5.14 (a) is likely to exceed the resolution of the printer but it is well adapted to the local flow field. This is illustrated by the close-up of the throat depicted in Figure 5.15. The two strut shocks interact with each other and with the expansion fans centered at $(x_i, y_i) = (\pm 0.5, 8.9)$ giving rise to Mach 1 flow. In fact, a very small region of subsonic flow (indicated by blue color) is formed in the center passage.

An important quantity to measure the performance of inlets is the total pressure loss [236]

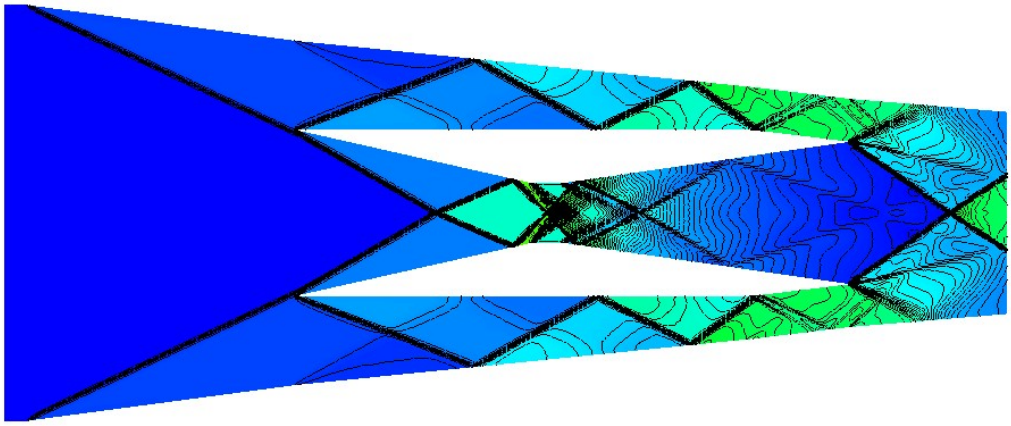
$$p_{\text{loss}} = \frac{p_{0,\infty} - p_0}{p_{0,\infty}}, \qquad p_0 = p\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} \tag{5.4.3}$$

where $p_{0,\infty}$ denotes the total pressure for the free stream condition. The inlet should be designed so as to minimize the pressure loss at the cruising Mach number, whilst an adequate loss is desirable at off-design points [236]. The Mach number and the total pressure loss along the center line and at the exit are displayed in Figure 5.16. Each compression increases the total pressure loss step-by-step which attains the maximum value 31.18%. In fact, the small subsonic region in the center of

(a) Adapted computational mesh: 81,199 elements, 72,662 vertices



(b) Density isolines, FEM-TVD scheme, mixed Q_1/P_1 elements



(c) Mach number isolines, FEM-TVD scheme, mixed Q_1/P_1 elements

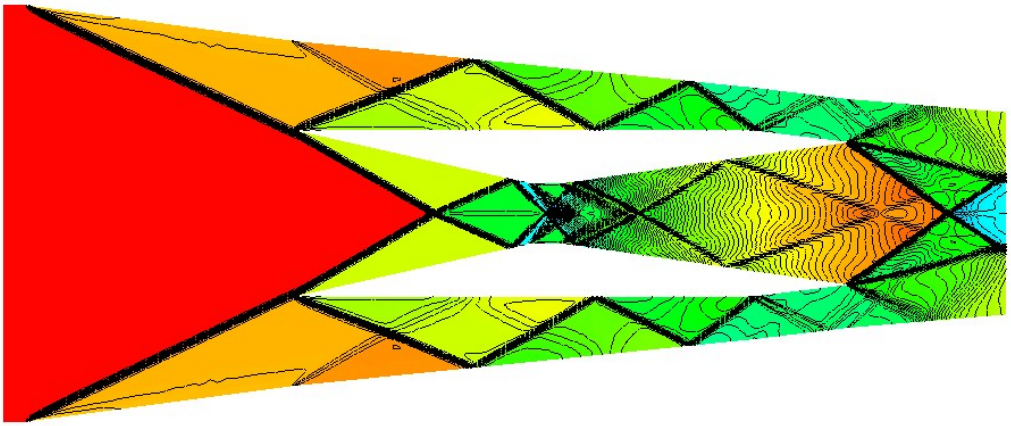


Fig. 5.14. Scramjet inlet at Mach 3 on locally adapted mesh (100 contours).

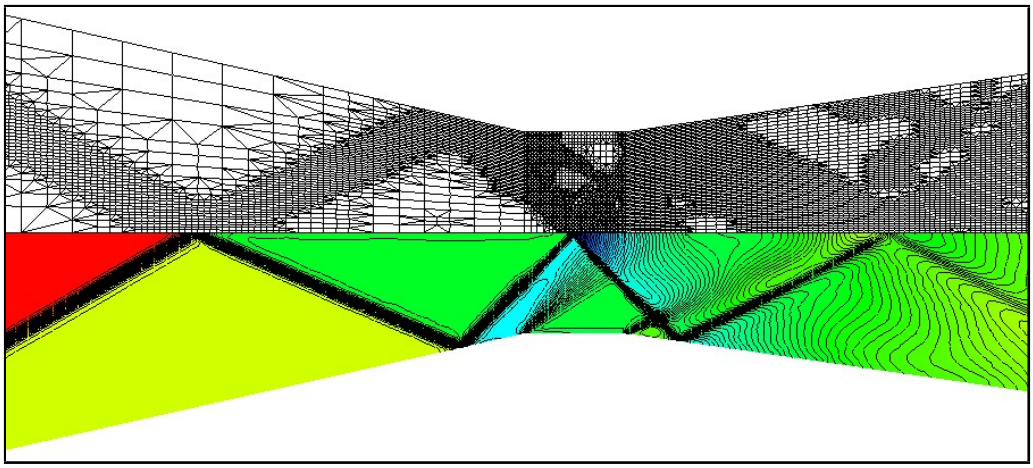


Fig. 5.15. Scramjet inlet at Mach 3: close-up of the center passage (100 contours).

the throat at $(x_i, y_i) \approx (9.2, 0.0)$ is clearly visible from the mid-channel Mach number. Moreover, the steep fronts near $y \approx \pm 0.7$ in the total pressure loss at the exit indicate the presence of the slip lines emanating from the trailing edges of the internal obstacles. In summary, the depicted profiles are in good agreement with the numerical results presented by Shapiro [236]. It is worth mentioning that no spurious oscillations due to numerical errors occur in the vicinity of shocks.

We do not want to conceal that a large number of simulation runs had to be performed until usable configurations for the error indicator were found. As an example consider the adapted grid and the computed density distributions shown in Figure 5.17 (a) which result from the slightly less restrictive parameters $tol_{\text{ref}} = 0.3$ and $tol_{\text{crs}} = 0.1$ for refinement and re-coarsening, respectively. At first glance, the solution profile looks reasonable but some essential small scale features such as the third reflection of the strut leading edge shock are missing. This is due to the fact that fewer elements are refined in the center passage. We also performed the same simulation with linear finite elements on unstructured triangles using both configurations for mesh adaptivity; see Figure 5.17 (b) and (c). Let us summarize what we have said so far. Local mesh refinement constitutes a viable approach to simulate complex inviscid flows at reasonable costs. In practice, it typically suffices to adjust the grid to local features to get an idea of the flow physics. Rigorous error estimation

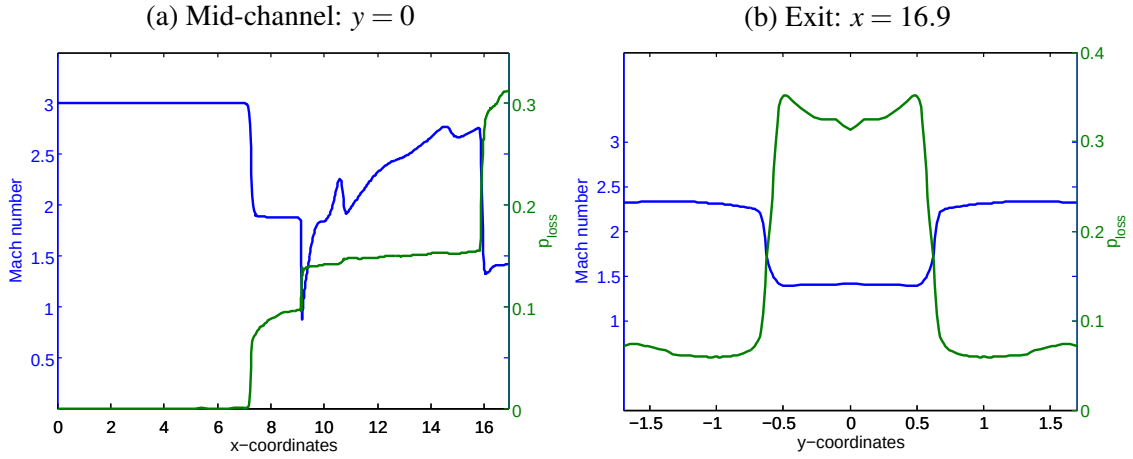
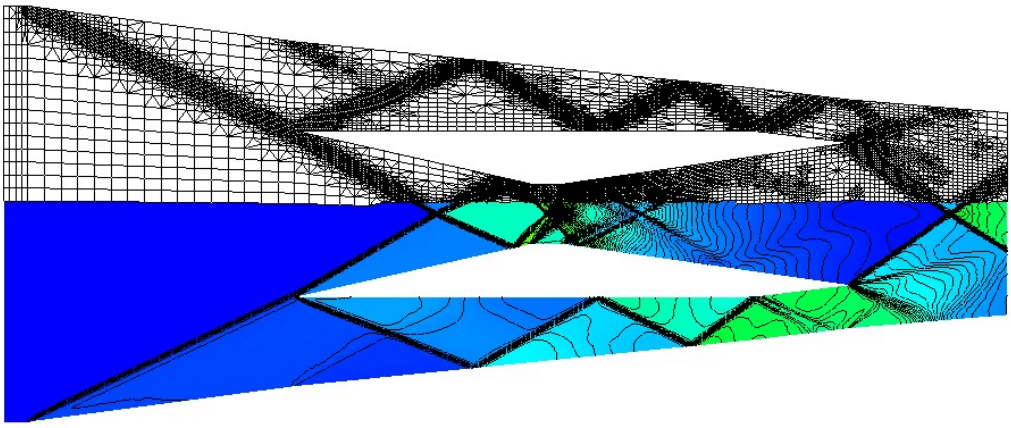
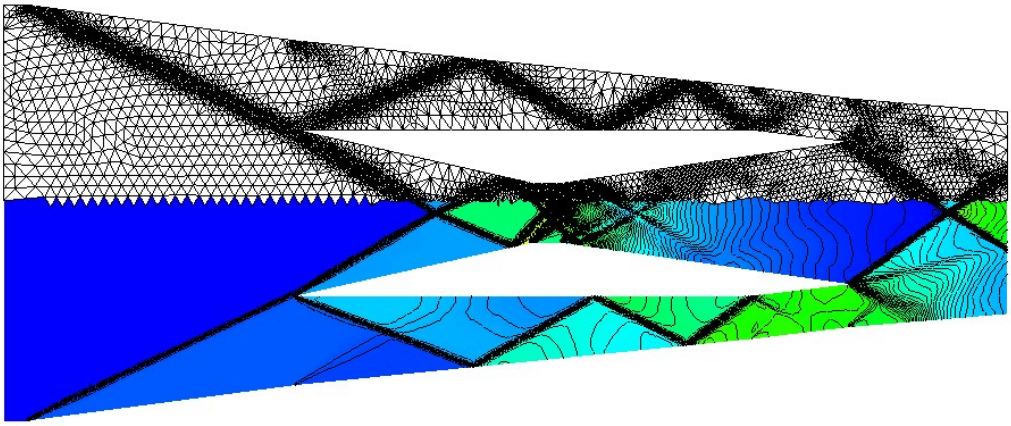


Fig. 5.16. Scramjet inlet at Mach 3: Mach number and total pressure loss.

(a) 40,886 elements, 47,834 vertices, $tol_{\text{ref}} = 0.3$, $tol_{\text{crs}} = 0.1$



(b) 52,280 elements, 26,652 vertices, $tol_{\text{ref}} = 0.3$, $tol_{\text{crs}} = 0.1$



(c) 84,937 elements, 43,220 vertices, $tol_{\text{ref}} = 0.15$, $tol_{\text{crs}} = 0.05$

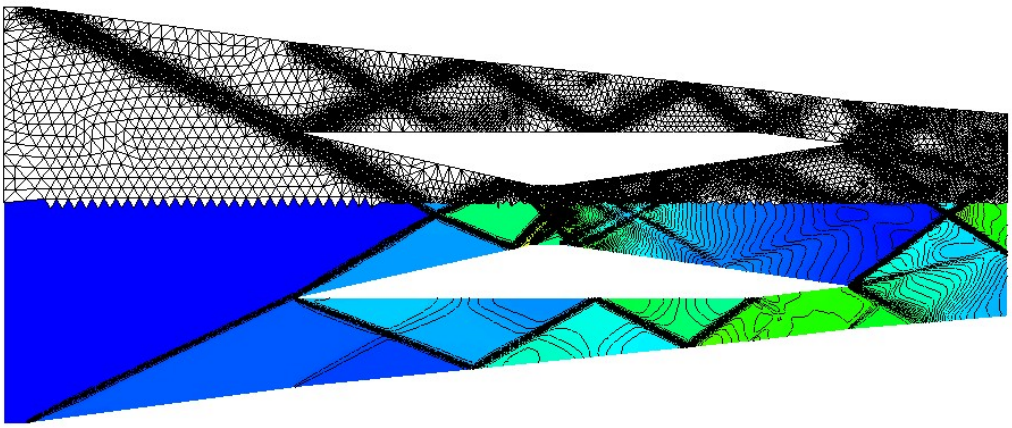


Fig. 5.17. Scramjet inlet at Mach 3: Comparison of adapted meshes (100 contours).

techniques for real-world problems are rarely available and/or they are prohibitively expensive so that mesh adaptivity for large-scale computations is frequently based on indicators such as the one proposed by Löhner [161]. On the other hand, his dimensionless indicator is no complete 'black-box' tool since it requires some user experience to prescribe reasonable parameters for refinement and re-coarsening. Of course, it is possible to post-process the grid, e.g., by performing Laplacian smoothing [80] or by implementing mechanisms to improve the mesh quality [32, 34]. However, such techniques cannot be effective if the predicted error distribution is erroneous.

5.4.3. Double Mach reflection

Our last numerical example proposed by Woodward and Colella [264] deals with a horizontally moving Mach 10 shock wave in air ($\gamma = 1.4$) which impinges on a wedge of 60° forming a jet of denser gas. The system is equivalent to sending a shock wave diagonally into a reflecting wall as presented in Figure 5.18. The initial data at time $t = 0$ is given by the two states [10]

$$\begin{bmatrix} \rho_{\text{pre}} \\ u_{\text{pre}} \\ v_{\text{pre}} \\ p_{\text{pre}} \end{bmatrix} = \begin{bmatrix} 8.0 \\ 8.25\cos(30^\circ) \\ -8.25\sin(30^\circ) \\ 116.5 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \rho_{\text{post}} \\ u_{\text{post}} \\ v_{\text{post}} \\ p_{\text{post}} \end{bmatrix} = \begin{bmatrix} 1.4 \\ 0.0 \\ 0.0 \\ 1.0 \end{bmatrix} \tag{5.4.4}$$

which are separated by the diagonal shock front. In particular the pre-shock conditions are adopted for $x < x_0 + y/\sqrt{3}$ with $x_0 = 1/6$ and the post-shock data is used elsewhere. The boundary conditions are constructed so as to mimic the physical setup of a vertical Mach 10 shock being reflected from the bottom wall sloped at 60°. Hence, standard no-penetration/reflecting boundary conditions are prescribed at the solid wall Γ_{wall}. Since the flow is supersonic at the outflow Γ_{out} no boundary conditions need to be specified, whereas the pre-shock initial conditions are imposed at the supersonic inlet Γ_{in}. The speed of sound ahead of the shock is 1 and the shock propagates at speed 10 so that the intersection point of the diagonal line with the upper boundary $y = 1$ moves at speed $s = 10/\cos(30^\circ)$. Hence, the pre- and post-shock initial data (5.4.4) serve as boundary conditions for $\Gamma_{\text{pre}}(t) = \{(x,y) : x < x_s(t), y = 1\}$ and $\Gamma_{\text{post}}(t) = \{(x,y) : x \geq x_s(t), y = 1\}$, whereby the separation point satisfies $x_s(t) = x_0 + (1+20t)/\sqrt{3}$. The simulation was run for the short time period $0 \leq t \leq 0.2$. A detailed description of this benchmark configuration which has become a challenging test problem for hydrodynamic algorithms can be found in reference [264].

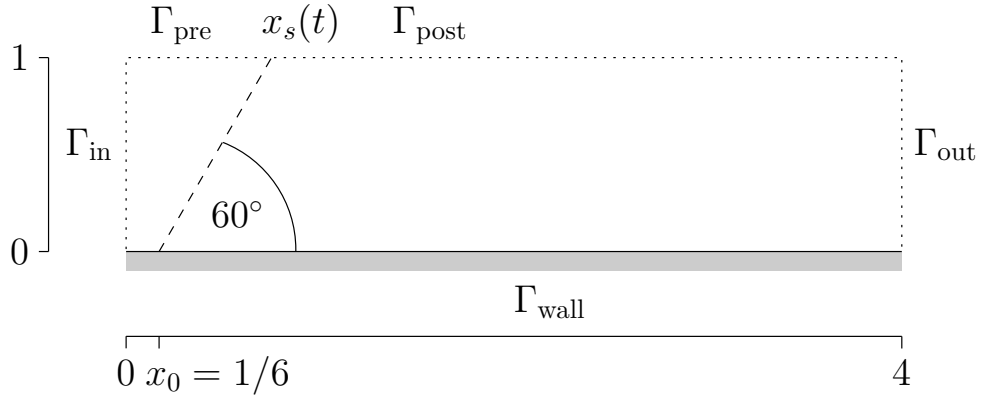


Fig. 5.18. Double Mach reflection: computational domain.

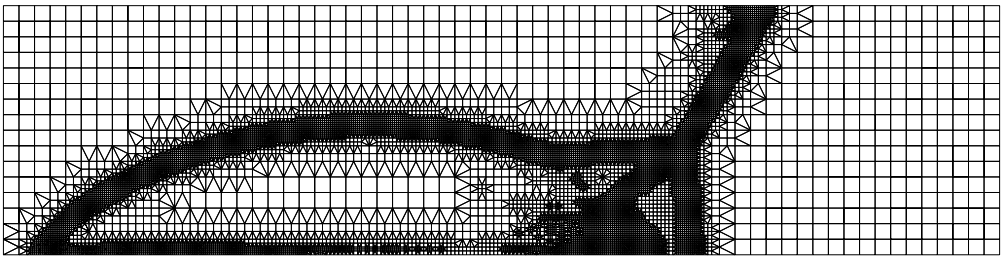
Albeit the fact that the temporal evolution of the flow is essential, we performed characteristic flux limiting of TVD type so that the antidiffusive contribution of the consistent mass matrix was not partially recovered with aid of flux correction. Of course, the resolution can be improved by resorting to FCT algorithms [136, 269] which are preferable for the simulation of time-dependent problems but our main objective was to test the performance of the *dynamic* mesh adaptation strategy proposed in Section 4.6. Moreover, Crank-Nicolson time-stepping ($\theta = 0.5$) was employed, whereby the time step Δt =3.0e–4 was used in accordance with the numerical experiments published by Woodward and Colella [264]. They considered finite difference schemes so that a structured grid with $\Delta x = \Delta y = 1/120$ was employed to compute the reference solution. Thus, we adopted an initial 16×64 coarse grid ($\Delta x = \Delta y = 1/16$) and allowed up to three levels of local refinement which corresponds to the minimum mesh width $\Delta x = \Delta y = 1/128$. The marking of elements for refinement and re-coarsening was based on Löhner's [161] dimensionless error indicator which was applied to the density variable. Three pre-adaptation steps were performed to adapt the initial triangulation to the diagonal shock wave. Due to the rapid movement of the shock wave, the computational grid was adjusted rather frequently at rate Δt_{adapt} =1.0e–3 and a protection layer of width two was introduced to refine elements in advance (see Section 4.7.2 for details). Otherwise, the Mach 10 shock would have moved out of the refined mesh region and its crisp resolution would have been irrecoverably lost. Furthermore, the thresholds tol_{ref} and tol_{crs} were determined automatically as proposed by Shapiro [236]. That is, he root mean square value of the error indicator was calculated and the tolerances for refinement and re-coarsening were defined as a fraction and a multiple of e_{rms}, respectively. Experimentally, the configuration $tol_{\text{ref}} = 110\% \times e_{\text{rms}}$ and $tol_{\text{crs}} = 30\% \times e_{\text{rms}}$ seem to work well and the noise filter was set to $C_n = 0.01$.

Figure 5.19 shows the computational mesh at time $t = 0.2$ and the density and pressure values of the approximate solution which are similar to the numerical profiles presented by other researchers [62, 136, 264, 269]. In fact, the density ranges from 1.4 to 22.19 which is in very good agreement with published reference values [10, 264]. Another important quantity is the adiabatic constant $A = p/\rho^\gamma$. It is a function of the entropy and provides a useful measure for numerical errors. The isolines shown in Figure 5.19 (d) are similar to those of the reference solution [264]. Note that 'staircase structures' and other peculiar features which were observed by Woodward and Colella [264] in the numerical solutions produced by Boris' ETBFCT method [38] are not generated by Kuzmin's FEM-TVD scheme [135]. This benefit of AFC schemes was also reported for the novel FEM-FCT limiter [136] being applied to the same test problem.

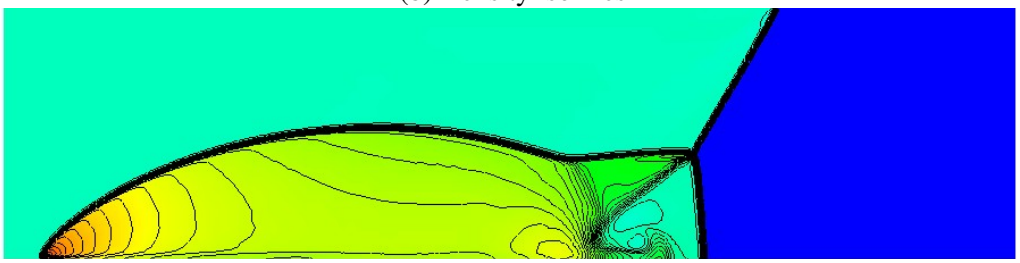
This benchmark illustrates the importance of choosing a meaningful key variable for mesh adaptivity. The pressure variable is sensitive to shock waves but it remains constant across the contact discontinuity which 'drives' the straight shock [10]. Hence, application of the error indicator to the quantity p could miss essential features of the flow which might lead to poorly adapted meshes. On the other hand, the adiabatic constant 'feels' the flow features near the bottom wall but the bowed shock would not be detected if $A = p/\rho^\gamma$ was used as indicator variables. Conversely, the density provides valuable information about shocks waves, contact discontinuities *and* rarefaction fans, and hence, it constitutes an excellent key variable for mesh adaptation.

The numerical results obtained by using linear finite elements are shown in Figure 5.20. Note that the triangles of the initial mesh, and hence, those of all computational grids are not aligned with the diagonal shock wave. By construction, the number of elements is larger since each quadrilateral is divided into two triangles. In essence, the solution profiles obtained by using bilinear and linear finite elements are comparable. However, there is some slight difference in the angle at which the first shock wave impinges on the bottom wall. In our opinion, the bilinear finite element solution depicted in Figure 5.19 agrees well with the reference plots published by Woodward and Colella [264], whereas the angle of the first shock wave is a little bit too steep in the contour plots depicted in Figure 5.20 which were computed by linear finite elements.

(a) Adapted computational mesh: 13,171 elements, 11,631 vertices



(b) Density isolines



(c) Pressure isolines

(d) Isolines for $A = p/\rho^{\gamma}$

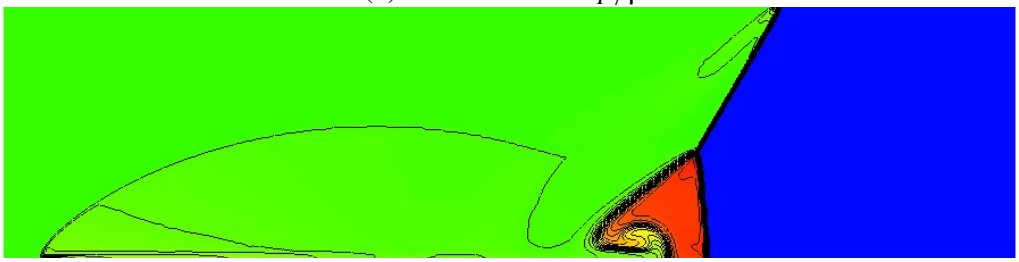
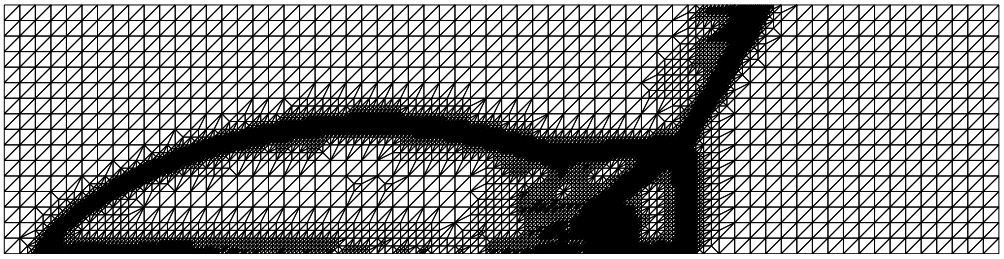


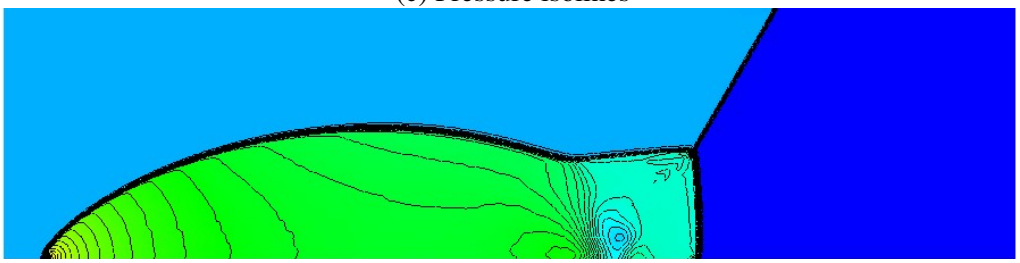
Fig. 5.19. Double Mach reflection at $t = 0.2$: FEM-TVD scheme (50 contours).

(a) Adapted computational mesh: 19,195 elements, 9,799 vertices



(b) Density isolines

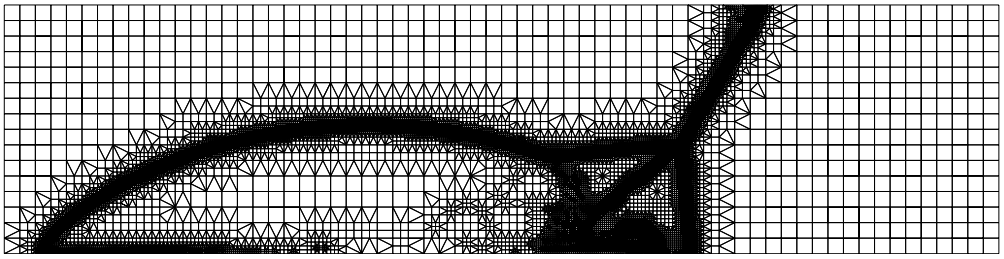
(c) Pressure isolines



(d) Isolines for $A = p/\rho^{\gamma}$

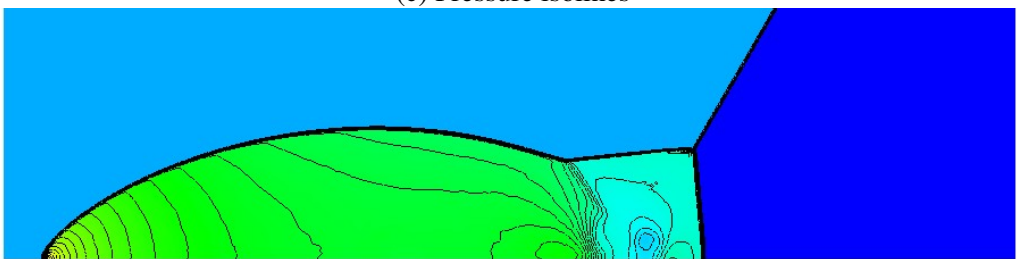
Fig. 5.20. Double Mach reflection at $t = 0.2$: FEM-TVD scheme (50 contours).

(a) Adapted computational mesh: 27,015 elements, 23,164 vertices



(b) Density isolines

(c) Pressure isolines



(d) Isolines for $A = p/\rho^{\gamma}$

Fig. 5.21. Double Mach reflection at $t = 0.2$: FEM-TVD scheme (50 contours).

As a very last example, we reconsidered the 16×64 coarse mesh and increased the maximum number of admissible refinement levels by one, that is, each quadrilateral of the initial triangulation could be subdivided four times. It is worth mentioning that global refinement would lead to 262,144 cells which corresponds to $\Delta x = \Delta y = 1/256$, whereas 23,164 degrees of freedom and 27,015 bilinear and linear (30%) finite elements were used to compute the solution profiles depicted in Figure 5.21. The essential features of the flow field are sharply resolved and no spurious undershoots and overshoots are present. The scrawly isolines in the density plot are due to the visualization software which is misled by the layer of triangular transition elements.

5.4.4. Conclusion on adaptive schemes for the compressible Euler equations

In the present chapter we applied the algebraic flux correction methodology introduced by Kuzmin and Turek [144, 145] to systems of hyperbolic conservation laws [142] and studied the performance of the dynamic mesh adaptation algorithm described in Section 4.6. In particular, three different benchmark problems for the compressible Euler equations were considered which illustrated the potential of high-resolution finite element schemes combined with local mesh refinement. The first example, the Mach 2 flow in a converging channel, was used to demonstrate that upwind biased flux limiting of TVD type [135] can be adopted to resolve shock waves with high accuracy. The solution profiles predicted by linear and bilinear finite elements were completely free of spurious undershoots and overshoots which would be generated if high-order discretizations such as the standard Galerkin method were employed without stabilization. The underlying low-order method was based on the generalized LED constraint 5.2.1 which amounts to a conservative elimination of negative eigenvalues from local Jacobian matrices [142]. Scalar dissipation proportional to the spectral radius of the Roe matrix as well as tensorial artificial viscosities were utilized to design low-order schemes. The computed profiles were free of non-physical oscillations but the accuracy of the approximate solution leaves a lot to be desired.

In general, the resolution of the solution can be improved by performing local mesh refinement. However, recovery-based error indicators can only detect 'features' which are present in the predicted flow fields. Hence, an overly diffusive low-order method may not work well together with mesh adaptation since the error indicator does not provide meaningful information where to refine the grid. In contrast, the high-resolution FEM-TVD scheme [142] was capable to reproduce the flow field accurately on coarse grids so that 3–4 local refinement steps were sufficient to resolve shock waves and small scale feature very crispy. In addition to mesh refinement it is advisable to allow for mesh re-coarsening also for stationary problems. In our numerical studies, the number of elements could be reduced by performing some extra adaptation steps once the maximum number of refinement levels was reached which was also observed by Shapiro [236].

The resolution power of the high-resolution FEM-TVD procedure equipped with the hierarchical mesh adaptation algorithm was investigated by considering a Mach 3 flow exposed to a prototypical scramjet inlet which led to complex shock interactions due to the presence of internal obstacles. This challenging benchmark which was studied by various researchers, e.g., in [49, 75, 140, 236], served as an illustration for the applicability of the presented mesh refinement strategy to 'realistic' problems which would be expensive to simulate if globally refined grids were employed. The interplay of multiple shocks featuring different intensities required the use of Löhner's dimensionless error indicator [161] in order to detect small scale features such as the expansion fans off the throat. However, choosing the tolerance for refinement and re-coarsening was a non-trivial task which required several simulation runs until viable parameters were found. As a remedy, we implemented the automatic threshold mechanism proposed by Shapiro [236] which worked well in practice. Finding absolute criteria for refinement and re-coarsening valid for the complete simulation turned out to be more difficult than evaluating the root mean square of

the error indicator and adopting a multiple and a fraction thereof as tolerance parameters. However, there remains some empiricism in the detection of under- and overresolved regions of the computational grid by means of error indicators which demands for experienced practitioners.

The effectiveness of the re-coarsening algorithm was demonstrated by the pseudo-transient supersonic flow in a 5° converging channel and the truly time-dependent double Mach reflection problem suggested by Woodward and Colella [264]. In the first benchmark, the inlet Mach number was periodically modulated between 2 and 4 so that the quasi-stationary shock wave moved back and forth in the two-dimensional channel. The mesh adaptation procedure succeeded in following the multiply reflected shock wave and produced computational grids in accordance with the sawtooth shape of the solution. In particular, the initial triangulation was recovered after one complete period which demonstrated the reliability of the re-coarsening strategy. The complicated flow field formed by the double Mach reflection of a planar shock impinging on a wedge gave insight into the performance of dynamic mesh adaptivity for time-dependent inviscid flows. The computational grid depicted in Figure 5.21 (a) is precisely adjusted to the approximate solution. In fact, ten times more cells would be necessary to reproduce the flow field with comparable accuracy on a structured grid. To cut a long story short, the adaptive high-resolution finite element schemes presented in this thesis seem to be a viable strategy for the simulation of stationary and time-dependent inviscid flows governed by the compressible Euler equations.

Let us finally mention that it was impossible to generate usable computational grids by means of the equidistribution strategy applied to the local gradient error except for the converging channel problem (see Section 4.4 for details). This may be attributed to the fact that expressions (4.4.4) and (4.4.6) are based on the assumption that the local error be equally distributed which is not the case if the flow field exhibits strong shocks and weak features simultaneously. As a result, the error indicator provoked mesh refinement in the vicinity of strong features missing the weak ones. In practice, it is therefore expedient to adopt dimensionless error indicators such as the one proposed by Löhner [161] unless rigorous and efficient error estimation techniques are available.

6

Conclusions and outlook

In this thesis, we endeavored to develop adaptive high-resolution finite element methods applicable to convection dominated flow problems. One major highlight was the design of discrete Newton algorithms which were tailored to the peculiarities of the algebraic flux correction (AFC) methodology founded by Kuzmin and Turek [133, 144]. A common feature of AFC schemes is, that they are based on conservative matrix manipulations so as to fulfill rigorous mathematical constraints. In particular, negative off-diagonal entries of the discrete transport operator were eliminated by adding artificial diffusion to the corresponding positions in the upper *and* lower triangular matrix and subtracting the same amount of numerical dissipation from the diagonal coefficients.

The high-resolution schemes presented in this thesis made use of nodal flux limiters to predict the amount of antidiffusion which was applied to remove excessive artificial diffusion without producing spurious oscillations. By construction, there exists no continuous formulation like in traditional stabilized Galerkin methods which can be differentiated *a priori* to obtain analytical expressions for the Jacobian matrix. As a remedy, we resorted to numerical differentiation and approximated the Newton operator by divided differences. In particular, efficient edge-by-edge procedures [183, 187] were presented for assembling the Jacobian matrix for upwind-biased TVD limiters [135, 145] and for the semi-implicit FEM-FCT algorithm [137]. Moreover, the construction of the 'discrete upwind Jacobian' was addressed which was required for nonlinear problems such as the inviscid Burgers equation. Owing to the nodal correction factors predicted by the limiter and used to correct the antidiffusive fluxes there was an indirect coupling between nodes not sharing a common edge. Thus, the discrete Newton operator gave rise to additional fill-in, whereby its sparsity pattern was determined *a priori* by symbolic matrix multiplication.

Numerical experiments were performed to compare the discrete Newton method to the standard defect correction scheme which was typically used in our finite element code. It was impossible to find the optimal solution strategy which was superior in all disciplines. In fact, the number of nonlinear iteration could always be reduced by using Newton's method provided that the perturbation parameter and the forcing term were chosen carefully. However, the CPU time required to produce the numerical solution should be taken into account to analyze the total efficiency of a numerical scheme. As a rule of thumb, the fixed point defect correction approach preconditioned by the low-order evolution operator proved to be more efficient for 'simple' problems such as the stationary convection-diffusion equation for which the system matrix was assembled once and for all at the beginning of the simulation. On the other hand, Newton's method was preferable for the treatment of complex flows, e.g., Burgers' equation in space-time and the continuity equation with a time-dependent velocity field, whereby the time step size was adjusted dynamically with aid of a PID controller. It is therefore advisable to implement both solution strategies and let the user select the nonlinear solver which is most suitable for the particular problem. As an alternative, the solution procedure may be adjusted dynamically in the course of the simulation, e.g., by monitoring the asymptotic convergence rates of the nonlinear solver in each (pseudo-) time step.

Another main contribution of this thesis was the development of a hierarchical mesh adaptation algorithm based on the red-green strategy by Bank et al. [19] and the work by Hempel [102] for triangles in 2D. Let us subsume the essential properties of the novel adaptation procedure:

- it was applicable to stationary and time-dependent problems without modifications
- the use of arbitrary triangulations consisting of triangles and/or quadrilaterals was feasible, whereby the conformality of the grid was accomplished by introducing transition elements
- all required information (refinement level, characterization of elements, relationship of adjacent elements) was extracted from the nodal generation number so that tree-based data structures were not necessary to maintain the genealogy of the triangulation
- each refinement operation was reversible by application of the corresponding unique re-coarsening operation so that the initial triangulation could be recovered at any time
- the selection of (patches of) elements for re-coarsening was based on a recursive vertex-locking algorithm which could be applied to an arbitrary set of 'pre-locked' nodes that should not be deleted from the triangulation, e.g., all vertices of the initial coarse grid
- a local two-level ordering strategy was adopted to allow for an efficient identification of cells based on a limited number of element states which could be defined *a priori*

Our dynamic mesh adaptation algorithm was shown to be effective in the simulation of convection dominated flows governed by scalar conservation laws and by the compressible Euler equations. The solution profiles computed on structured and locally adapted meshes possessed the same accuracy if the finest grid spacing was comparable, whereby the overall computing time was reduced by using h-adaptivity in lieu of of global refinement. Furthermore, local mesh refinement was successfully employed to improve the resolution of shock waves and other flow features in situation, where global refinement was impracticable. It is worth mentioning that less than 1% of the CPU time was typically spent on mesh adaptation not including the computing time required to predict the solution error. In our opinion, the dynamic mesh adaptation procedure constitutes a black-box tool that can be easily integrated into CFD software packages. Moreover, the presented concepts are applicable to stationary and time-dependent flow simulations alike so that dynamic h-adaptivity following the red-green strategy [19] can be implemented as a reusable library.

Three dimensional simulations are computationally expensive so that adaptive mesh refinement and re-coarsening is indispensable for the efficient treatment of real-world applications. Our mesh adaptation algorithm was derived in a general framework and an efficient implemented in 2D was presented as a proof-of-concept. From the theoretical point of view, all ingredients – the characterization of elements by means of the nodal generation function, the recursive vertex-locking algorithm and the local two-level ordering – can be readily extended to three dimensions. Viable refinement patterns for tetrahedral elements can be found in the book by Löhner [163] and the references therein. Refinement strategies for hexahedral, prismatic and pyramidal elements are presented by Mavriplis [180] and subdivision of octahedra is addressed for instance by Leitnet and Selberherr [152]. These element types can be directly integrated into the adaptation library including the admissible rules for refinement. By construction, each refinement operation can be reversed by 'gluing together' the group of cells which form the original macro element. The nodes not belonging to coarse grid elements which can therefore be eliminated from the given triangulation are determined by the recursive vertex-locking algorithm. In practice, the integration of new element types into the adaptation library may be performed step-by-step starting from tretrahedra which may serve as transition elements, e.g., in the subdivision of hexahedral cell.

In this thesis, gradient-recovery techniques such as the superconvergent patch recovery (SPR) procedure by Zienkiewicz and Zhu [274, 276] as well as L_2-projection schemes [4] were investigated for one-dimensional test cases. The consistent L_2-projection method was prone to create oscillations in the vicinity of steep fronts and local extrema which could be rectified *a posteriori* [185, 186]. In essence, the consistent finite element slopes served as natural upper and lower bounds for the predicted gradient values which were corrected by means of a slope limiter. This post-processing technique was generalized to multidimensions by performing edgewise slope limiting. Limited gradient averaging [185] was suggested as an efficient alternative to the computationally more expensive superconvergent patch recovery if a rough estimate of the gradient error was sufficient to mark elements for refinement, e.g., in the vicinity of steep fronts.

Complex flow fields which gave rise to various phenomena at different intensities called for the use of Löhner's [161] dimensionless error indicator to resolve all essential flow features equally well. In our numerical experiments we observed that the indicator was sensitive to noise filtering and to the choice of the parameter values used to mark elements for refinement and re-coarsening. This difficulty could be alleviated by resorting to the automatic threshold method suggested by Shapiro [236] but there still was some empiricism in the error detection mechanism.

In light of the above, the area of computable *a posteriori* error estimates for convection dominated flows requires further investigation. At the moment, computing time was the limiting factor which kept us from allowing finer grids for the computation of stationary flows and the maximum refinement level was defined *a priori*. In the future, mathematical criteria based on the error of the approximate solution should be used to steer mesh adaptivity. In fact, the nodal generation function used to manage the grid genealogy allows to define individual refinement levels for (groups of) vertices. Hence, a precise control of local mesh refinement is feasible provided that reliable estimates for the solution error are available. Goal-oriented error estimation in terms of output functionals [194, 258] seems to be a promising direction for future research. In the context of discontinuous Galerkin finite element schemes, this methodology was successfully applied to the compressible Euler and Navier-Stokes equations by Hartmann and Houston [95, 96]. Classical approaches such as the DWR method by Becker and Rannacher [26] rely on the Galerkin orthogonality property which is violated if algebraic flux correction is performed [132–137, 139–145]. Recently, Kuzmin [131] devised a generalized error estimator which is applicable if the Galerkin orthogonality property is not satisfied but an exhaustive numerical analysis is outstanding.

The design of rigorous error estimates for time-dependent flows is another topic of utmost importance. In our numerical tests, the evolutionary PID controller [256] provided an efficient mechanism to adjust the time step size dynamically by monitoring the relative changes of the solution. Moreover, the use of protection layers of one or two elements was sufficient to account for the evolution of the flow field provided that mesh adaptation was performed frequently enough. However, some physically motivated criteria may be preferable due to the following reasons: The protection layer may be constructed 'in the direction' of the flow so that the creation of excess elements, e.g., behind the moving front, is prevented. Furthermore, the update frequency of the computational grid may be reduced if reliable predictions for the temporal evolution are available. Hence, the design of space-time adaptive methods for conservation laws [130, 243] may be helpful to improve the grid quality and to reduce the CPU time required for transient flow calculations.

In summary, the family of algebraic flux correction schemes needs to be equipped with computable *a posteriori* estimates for the spatial and possibly the temporal solution errors. In essence, flux correction constitutes a rudimentary p-adaptation approach which may be combined with existing p-adaptive algorithms using hierarchical basis functions. In conjunction with the dynamic h-adaptation procedure developed in this thesis, the design of hp-FEM-FCT and hp-FEM-TVD methods may be feasible so as to create powerful CFD tools in the long run.

A

Finite element fundamentals

In this appendix, we present the main concepts of the finite element method which is adopted throughout this thesis. The next paragraph is concerned with general principles of weighted residual methods. In particular, the Galerkin finite element discretization is briefly described. The numerical schemes presented in this thesis rely on certain properties of coefficient matrices, e.g., the zero row-sum property of $\mathbf{C} = \{\mathbf{c}_{ij}\}$ has been utilized to deduce a conservative flux decomposition (cf. equation (3.2.24) in Section 3.2.3). It is therefore advisable to shed light on general properties of interpolation functions which are used in the assembly of matrices.

The algorithms presented in this work, have been developed in a most general framework so that there is no restriction to simplex elements and/or two space dimensions unless indicated otherwise. Since numerical tests have been performed for two-dimensional problems we find it helpful to give explicit formulas for linear and bilinear finite elements.

Weighted residual formulation

Let $\mathcal{L}$ denote a generic partial differential operator and consider the linear model problem

$$\mathcal{L}u = f \tag{A.1}$$

The basic idea of the weighted residual method is to replace the above equation by its weak form

$$\int_{\Omega} w\,[\mathcal{L}u - f]\,\mathrm{d}\mathbf{x} = 0 \tag{A.2}$$

which is obtained by multiplying the residual by the weighting function w and setting the integral over the domain $\Omega \subseteq \mathbb{R}^{N_\mathrm{D}}$ equal to zero. In order to compute an approximate solution u_h to the above problem by the finite element method, the computational domain is subdivided into a collection of N_E non-overlapping sub-domains Ω_k called finite elements such that

$$\bigcup_{k=1}^{N_\mathrm{E}} \Omega_k \subseteq \Omega \tag{A.3}$$

In two space dimensions, such partitions typically consist of triangular or quadrilateral elements comprising three or four vertices. In the three-dimensional case, triangles and quadrilaterals can be generalized to tetrahedra and hexahedra, respectively. The flexibility of the finite element method makes it possible to employ different types of elements in a single computational mesh.

As a result, the approximate solution $\mathrm{u}_h = \sum_j \varphi_j \mathrm{u}_j$ can be defined separately on each element, whereby the trial functions φ_j are required to satisfy some basic properties. For a comprehensive introduction to the theory of finite elements, the interested reader is referred to the classical textbook [58, 120]. The application of finite elements to fluid flow problems is discussed by Donea and Huerta [64]. A detailed description of practical aspects and guidelines for the efficient implementation of finite element flow solvers can be found in the book by Löhner [163].

Properties of interpolation functions

From the nodal coefficients $\mathrm{u}_j(t) = \mathrm{u}_h(\mathbf{x}_j, t)$ given at time t the solution values within each element Ω_k can be interpolated by means of local basis functions $\varphi_j^{(k)}$ such that [236]

$$\mathrm{u}_h(\mathbf{x}, t) = \sum_{j=1}^{m} \mathrm{u}_j(t)\varphi_j^{(k)}, \qquad \forall \mathbf{x} \in \Omega_k \tag{A.4}$$

where m denotes the number of local degrees of freedom for each element. Summing the local basis functions over all elements one obtains a set of functions $\{\varphi_j\}_{j=1}^{M}$ which are used to expand the approximate solution for the whole domain in terms of the nodal coefficients u_j, i.e. [236]

$$\mathrm{u}_h(\mathbf{x}, t) = \sum_{j=1}^{M} \mathrm{u}_j(t)\varphi_j(\mathbf{x}), \qquad \forall \mathbf{x} \in \Omega \tag{A.5}$$

In the above equation M is the total number of degrees of freedom. To obtain a valid finite element approximation at reasonable costs, it is expedient to impose the following criteria [163]:

1. Relation (A.5) requires that the basis function φ_j is one at node j and vanishes elsewhere

$$\mathrm{u}_h(\mathbf{x}_i, t) = \sum_{j=1}^{M} \mathrm{u}_j(t)\varphi_j(\mathbf{x}_j) = \mathrm{u}_j(t) \qquad \Rightarrow \qquad \varphi_j(\mathbf{x}_i) = \delta_{ij} = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases} \tag{A.6}$$

2. The sum of local basis functions $\varphi_j^{(k)}$ should be identically one for each element Ω_k such that constant functions are represented exactly which is required for *consistency*. This property carries over to the global basis functions φ_j which are required to feature unit sums

$$\sum_{j=1}^{M} \varphi_j(\mathbf{x}) = 1, \qquad \forall \mathbf{x} \in \Omega_k \tag{A.7}$$

3. For computational efficiency, it is desirable that the local basis function $\varphi_j^{(k)}$ vanishes outside of element Ω_k to obtain a finite element approximation which leads to sparse matrices. As a consequence, the global basis function φ_j turns into a hat function with local support comprising the set of elements meeting at vertex j and equals zero outside.

Another feature of shape functions directly follows from the constant sum property (A.7):

4. The sum of derivatives vanishes in each element (*conservation property* [163])

$$\sum_{j=1}^{M} \nabla\varphi_j(\mathbf{x}) = 0, \qquad \forall \mathbf{x} \in \Omega_k \tag{A.8}$$

Note that these properties do not require the shape functions to be continuous across element boundaries, and therefore, non-conforming finite elements such as the linear Crouzeix-Raviart element [57] or the rotated bilinear Rannacher-Turek element [212] have been devised. Furthermore, discontinuous Galerkin formulations [153] which are nowadays quite popular for the simulation of compressible flows can be employed for the spatial discretization. We will only consider conforming finite element discretizations based on the 'classical' Galerkin formulation.

According to the analysis given by Strang [241], the use of polynomial interpolation functions φ_j is the optimal choice in the following sense: In order to obtain a k^{th} order accurate approximation to an s^{th} derivative on a regular grid the interpolation functions must be capable of representing all polynomials of degree $k+s-1$ exactly. This requirement can be satisfied with a minimal number of components if polynomials are applied. In this work, Lagrangian interpolation functions will be employed to construct linear and bilinear finite element approximations.

Natural coordinate transformation and derivative calculation

It is important to notice the unstructured nature of the finite element method which allows for using arbitrary mixed triangular and/or quadrilateral (and even degenerated) elements. All operations are performed at the element level, that is, each element contribution is assembled locally and distributed to the global degrees of freedom afterwards. The price for this flexibility is the computational overhead cost resulting from indirect addressing and non-vectorized instructions.

Even though it is possible to express the finite element method completely in terms of physical coordinates $\mathbf{x} = (x, y, z)$ of the computational domain the interpolation functions are typically defined on some geometrically simple reference element $\hat{\Omega}_k$ in terms of a natural coordinate system $\hat{\mathbf{x}} = (\hat{x}, \hat{y}, \hat{z})$. The transformation between the physical and the reference element is carried out by some one-to-one mapping $F_k : \hat{\Omega}_k \rightarrow \Omega_k$. It is defined such that the following relation holds [120]

$$\mathbf{x} = F_k(\hat{\mathbf{x}}) = \sum_{j=1}^{m} \mathbf{x}_j \hat{\varphi}_j^{(k)}(\hat{\mathbf{x}}), \qquad \forall \mathbf{x} \in \Omega_k \tag{A.9}$$

Here, $\mathbf{x}_j$ denotes the physical coordinates of local degrees of freedom and. Once the basis functions have been defined on the reference element in terms of natural coordinates the inverse mapping can be employed for the transformation back to the physical space as follows [120]

$$\varphi_j^{(k)}(\mathbf{x}) = \hat{\varphi}_j^{(k)}(F_k^{-1}(\mathbf{x})), \qquad \forall \mathbf{x} \in \Omega_k, \qquad j = 1, \ldots, m \tag{A.10}$$

The first property of element shape functions, namely, $\varphi_j^{(k)}(\mathbf{x}_i) = \delta_{ij}$ remains valid on the reference element since $\varphi_j^{(k)}(\mathbf{x}) = \hat{\varphi}_j^{(k)}(\hat{\mathbf{x}})$. In order to compute derivatives of shape functions directly in the physical space, the chain rule can be applied to the above relation so as to obtain [120]

$$\hat{\nabla} \hat{\varphi}_j^{(k)}(\hat{\mathbf{x}}) = J_k(\mathbf{x}) \nabla \varphi_j^{(k)}(\mathbf{x}), \qquad \forall \mathbf{x} \in \Omega_k, \qquad j = 1, \ldots, m \tag{A.11}$$

Here, J_k denotes the Jacobian of the inverse transformation $F_k^{-1} : \Omega_k \rightarrow \hat{\Omega}_k$ which is defined as

$$J_k(\mathbf{x}) = \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{x}} & \frac{\partial z}{\partial \hat{x}} \\ \frac{\partial x}{\partial \hat{y}} & \frac{\partial y}{\partial \hat{y}} & \frac{\partial z}{\partial \hat{y}} \\ \frac{\partial x}{\partial \hat{z}} & \frac{\partial y}{\partial \hat{z}} & \frac{\partial z}{\partial \hat{z}} \end{bmatrix} \tag{A.12}$$

Let us assume that J_k is non-singular for all $\mathbf{x}$ in the element Ω_k which is a necessary condition for the mapping F_k to be invertible. In this case, its inverse J_k^{-1} exists and it can be used to express the derivatives of the solution vector in element Ω_k in terms of reference coordinates [236]

$$\nabla \mathrm{u}_h(\mathbf{x}, t) = J_k^{-1}(\hat{\mathbf{x}}) \sum_{j=1}^{m} \mathrm{u}_j(t) \hat{\nabla} \hat{\varphi}_j^{(k)}(\hat{\mathbf{x}}), \qquad \forall \mathbf{x} \in \Omega_k \tag{A.13}$$

For simplicity, the analysis is performed in two dimensions but the extension to three-dimensional problems is straightforward. Hence, let the physical coordinate system be denoted by $\mathbf{x} = (x, y)$ while the coordinates of the reference space are indicated by $\hat{\mathbf{x}} = (\hat{x}, \hat{y})$ so that [120]

$$J_k = \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{x}} \\ \frac{\partial x}{\partial \hat{y}} & \frac{\partial y}{\partial \hat{y}} \end{bmatrix} \qquad J_k^{-1} = \frac{1}{\det J_k} \begin{bmatrix} \frac{\partial y}{\partial \hat{y}} & -\frac{\partial x}{\partial \hat{y}} \\ -\frac{\partial y}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{x}} \end{bmatrix} \tag{A.14}$$

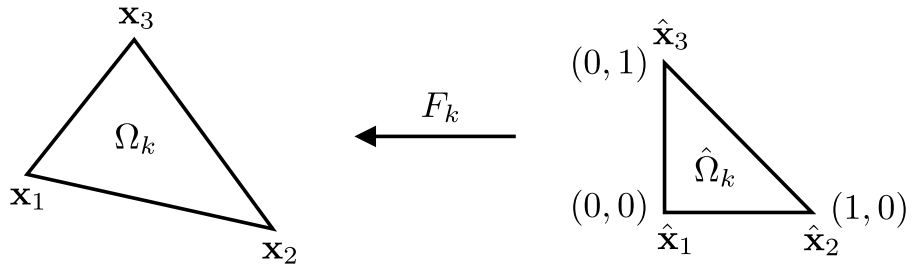


Fig. A.1. Linear finite element.

Linear elements

The simplest two-dimensional Lagrangian finite element basis consists of piecewise linear polynomials. The situation for a triangular grid cell is illustrated in Figure A.1. In this case, the number of local degrees of freedom $m = 3$ coincides with the number of nodes at the three corners. The shape functions defined on the reference triangle $\hat{\Omega}_k$ are as follows [163]

$$\hat{\varphi}_1 = 1 - \hat{x} - \hat{y}, \qquad \hat{\varphi}_2 = \hat{x}, \qquad \hat{\varphi}_3 = \hat{y} \tag{A.15}$$

The transformation to the physical element can be carried out by the linear (affine) mapping

$$F_k(\hat{\mathbf{x}}) = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \hat{\mathbf{x}} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{A.16}$$

It exhibits some amenable properties which will turn out to be quite useful [120]:

1. Vertices in the reference element are mapped onto the vertices of the triangle.
2. Midpoints of edges remain midpoints under the linear transformation.
3. The barycenter $\frac{1}{3}\sum_{j=1}^{3} \hat{\mathbf{x}}_j$ of the reference element is mapped onto $\frac{1}{3}\sum_{j=1}^{3} \mathbf{x}_j$

Furthermore, the Jacobian of the inverse mapping and its determinant are given by [163]

$$J_k = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{bmatrix} \qquad \det J_k = 2|\Omega_k| \tag{A.17}$$

where $|\Omega_k|$ stands for the area covered by the sub-domain Ω_k. Note that J_k is constant on each element so that its determinant vanishes if and only if the triangle degenerates to some line segment. Consequently, the constant inverse of the Jacobian exists for any non-degenerate triangle so that the derivatives of the solution vector can be readily computed from (A.13) taking [163]

$$J_k^{-1} = \frac{1}{2|\Omega_k|} \begin{bmatrix} y_3 - y_1 & y_1 - y_2 \\ x_1 - x_3 & x_2 - x_1 \end{bmatrix} \tag{A.18}$$

Finally, let us present an analytic expression for the derivatives of shape functions in physical coordinates which can be used, e.g., for the direct computation of constant gradient values [163]

$$\frac{\partial}{\partial x}\begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{pmatrix} = \frac{1}{2|\Omega_k|}\begin{pmatrix} y_2 - y_3 \\ y_3 - y_1 \\ y_1 - y_2 \end{pmatrix} \qquad \frac{\partial}{\partial y}\begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{pmatrix} = \frac{1}{2|\Omega_k|}\begin{pmatrix} x_3 - x_2 \\ x_1 - x_3 \\ x_2 - x_1 \end{pmatrix} \tag{A.19}$$

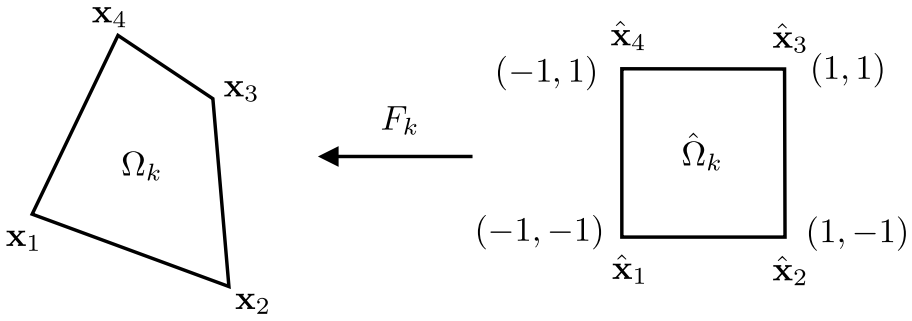


Fig. A.2. Bilinear finite element.

Bilinear elements

Multidimensional interpolants on quadrilaterals can be constructed by taking the tensor product of one-dimensional Lagrange polynomials defined on the reference element. Figure A.2 illustrates the situation for a bilinear finite element in physical and referential space. The number of local degrees of freedom is four which coincides with the number of vertices in the rectangle. The nodal interpolation functions in natural coordinates are given by the following expressions

$$\begin{aligned} \hat{\varphi}_1 &= (1-\hat{x})(1-\hat{y})/4, \qquad \hat{\varphi}_3 = (1+\hat{x})(1+\hat{y})/4 \\ \hat{\varphi}_2 &= (1+\hat{x})(1-\hat{y})/4, \qquad \hat{\varphi}_4 = (1-\hat{x})(1+\hat{y})/4 \end{aligned} \tag{A.20}$$

The bilinear mapping from the reference space to the physical coordinate system reads [236]

$$F_k(\hat{\mathbf{x}}) = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \hat{\mathbf{x}} + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \hat{x}\hat{y} + \begin{bmatrix} a_4 \\ b_4 \end{bmatrix} \tag{A.21}$$

In contrast to (A.16), it features an additional mixed term $\hat{x}\hat{y}$. To improve readability, auxiliary coefficients a_1 to a_4 and b_1 to b_4 have been introduced which are defined as follows [236]

$$\begin{aligned} a_1 &= (-x_1+x_2+x_3-x_4)/4, \qquad b_1 = (-y_1+y_2+y_3-y_4)/4 \\ a_2 &= (-x_1-x_2+x_3+x_4)/4, \qquad b_2 = (-y_1-y_2+y_3+y_4)/4 \\ a_3 &= (\ \ x_1-x_2+x_3-x_4)/4, \qquad b_3 = (\ \ y_1-y_2+y_3-y_4)/4 \\ a_4 &= (\ \ x_1+x_2+x_3+x_4)/4, \qquad b_4 = (\ \ y_1+y_2+y_3+y_4)/4 \end{aligned} \tag{A.22}$$

Note that for bilinear finite elements, the Jacobian matrix and its inverse are no longer constant but vary linearly in $\hat{x}$ and $\hat{y}$ due to the presence of the mixed term [236]

$$J_k = \begin{bmatrix} a_1+a_3\hat{y} & b_1+b_3\hat{y} \\ a_2+a_3\hat{x} & b_2+b_3\hat{x} \end{bmatrix} \qquad J_k^{-1} = \frac{1}{\det J_k} \begin{bmatrix} b_2+b_3\hat{x} & -(b_1+b_3\hat{y}) \\ -(a_2+a_3\hat{x}) & a_1+a_3\hat{y} \end{bmatrix} \tag{A.23}$$

The determinant of the Jacobian can be computed explicitly by means of the following formulation

$$\det J_k = (a_1b_2-a_2b_1)+(a_1b_3-a_3b_1)\hat{x}+(a_3b_2-a_2b_3)\hat{y} \tag{A.24}$$

A remark concerning the existence of J_k^{-1} is in order. Consider the case that $\det J_k$ which is linear in each element changes its sign from one node to the other. Then the mean value theorem states that the determinant attains zero somewhere in between such that the Jacobian J_k becomes singular. It

can be shown that this is the case if the largest interior angle between two edges exceeds the value π. To put it another way, the Jacobian for bilinear finite elements is non-singular if and only if the quadrilateral defined in physical coordinates is convex [241].

As for the linear finite elements (A.19), it is possible to give analytical expressions for the derivatives of shape functions in physical coordinates so that gradient values of the solution can be computed directly. However, the formulas are far more complicated than those for linear elements

$$\frac{\partial}{\partial x}\begin{pmatrix}\varphi_1\\ \varphi_2\\ \varphi_3\\ \varphi_4\end{pmatrix} = \frac{1}{4\det J_k}\begin{pmatrix} b_1-b_2+(-b_1-b_3)\hat{x}+(\ \ b_2+b_3)\hat{y}\\ b_1+b_2+(\ \ b_1+b_3)\hat{x}+(-b_2+b_3)\hat{y}\\ -b_1+b_2+(-b_1+b_3)\hat{x}+(\ \ b_2-b_3)\hat{y}\\ -b_1-b_2+(\ \ b_1-b_3)\hat{x}+(-b_2-b_3)\hat{y}\end{pmatrix} \tag{A.25}$$

$$\frac{\partial}{\partial y}\begin{pmatrix}\varphi_1\\ \varphi_2\\ \varphi_3\\ \varphi_4\end{pmatrix} = \frac{1}{4\det J_k}\begin{pmatrix} -a_1+a_2+(\ \ a_1+a_3)\hat{x}+(-a_2-a_3)\hat{y}\\ -a_1-a_2+(-a_1-a_3)\hat{x}+(\ \ a_2-a_3)\hat{y}\\ \ \ a_1-a_2+(\ \ a_1-a_3)\hat{x}+(-a_2+a_3)\hat{y}\\ \ \ a_1+a_2+(-a_1+a_3)\hat{x}+(\ \ a_2+a_3)\hat{y}\end{pmatrix} \tag{A.26}$$

The correctness of the above relations can be easily checked with aid of computer algebra systems.

B

Edge-based matrix assembly and storage techniques

This appendix is concerned with practical aspects of the edge-based matrix assembly utilized in Chapters 3 and 5. In particular, some useful data structures and techniques suitable for storing large sparse matrices which are typically encountered in finite elements are reviewed and a general framework for an edge-by-edge traversal is presented. Two alternative strategies are considered for storing global block matrices resulting from the discretization of systems of equations, whereby the block-wise CSR approach relies on the group finite element formulation [78].

2.1. Practical aspects of scalar matrices

Owing to the local support property of basis functions the connectivity graph of finite element matrices is sparse. This has led to the design of specialized storage techniques for sparse matrices [67]. It is common practice to adopt the compact row storage (CRS) or the compact column storage (CCS) – also called Harwell-Boeing [68] – sparse matrix formats. Both techniques are largely similar and differ only in the leading matrix dimension (row or column) that is stored en bloc. In other words, the CCS format for matrix A is equivalent to the CRS format for the transposed matrix A^T [67]. Therefore, the decision on which format to use for a concrete application should be based on the fact whether the matrix or its transposed are required more frequently.

Consider the scalar finite element matrix $K = \{k_{ij}\}$ and let the number of non-zero matrix entries, i.e. $(\varphi_i, \varphi_j) \neq 0$, be denoted by `nnz`. The idea of the compressed row storage (CRS) technique is to traverse the matrix row-by-row and put subsequent non-zeros in contiguous memory locations of the array `K(1:nnz)`. In addition, two auxiliary integer arrays `ColumnIndex(1:nnz)` and `RowPtr(1:nr+1)` are required to reconstruct the original matrix structure. As the name suggests, the number of rows stored in the matrix is denoted by `nr`. For sufficiently sparse matrices the required amount of `2nnz+nr+1` memory positions is significantly less than that needed for the `nr×nr` data array storing the full matrix including all zero entries [67].

The vector `RowPrt` stores the location in the array `K` that starts a row. In other words, all of its entries satisfying `RowPtr(i)` $\leq$ `ipos` $\leq$ `RowPtr(i+1)-1` belong to the i^{th} row. For each item `ipos`, the corresponding column number is stored in the `ColumnIndex` vector. That is, the matrix entry located at the global position `ipos` belongs to the j^{th} column if `ColumnIndex(ipos)=j`. By convention, `RowPtr(nr+1)=nnz+1` so that no violation of the upper bound of the `ColumnIndex` array takes place [67]. Moreover, the column indices of each row are assumed to be stored in ascending order so that `RowPtr(i)` $\leq$ `ipos1` $<$ `ipos2` $\leq$ `RowPtr(i+1)-1` implies that the corresponding entries of the column index vector satisfy `ColumnIndex(ipos1)` $<$ `ColumnIndex(ipos2)`. This ordering convention is not a compulsory property of the standard CRS format but it is crucial for an efficient implementation of the edge-based matrix assembly [131].

Due to the importance of the matrix diagonal, it is worthwhile to introducing a third integer array `DiagonalPtr(1:nr)` which stores the absolute position of the diagonal coefficient for each

row [125]. In short, `ipos=DiagonalPtr(i)` denotes the global position of the diagonal matrix entry k_{ii} = `K(ipos)`, and hence, `ColumnIndex(ipos)=i`. As a result, diagonal positions can be addressed without investigating all row entries one after the other. In what follows, this approach will be referred to as CRS9 storage format. As an alternative, the CRS7 format with 'leading diagonals' slightly modifies the vector `RowPtr` so that the first item `ipos=RowPtr(i)` points to the location of the diagonal entry and the remaining entries of the i^{th} row are stored in positions `RowPtr(i)+1` to `RowPtr(i+1)-1` with ascending column indices [37].

Let us illustrate the two different storage techniques for the square matrix

$$K = \begin{bmatrix} 1 & 2 & 0 & 7 \\ 2 & 4 & 3 & 0 \\ 0 & 3 & 6 & 5 \\ 7 & 0 & 5 & 8 \end{bmatrix}$$

The corresponding representation in the CRS7 format with leading diagonal entry reads

K	1.0	2.0	7.0	4.0	2.0	3.0	6.0	3.0	5.0	8.0	7.0	5.0
ColumnIndex	1	2	4	2	1	3	3	2	4	4	1	3
RowPtr	1	4	7	10	13							

Here and below, underlined column numbers indicate the start of a new row. If the CRS9 format is adopted the additional index vector for the diagonal entries comes into play

K	1.0	2.0	7.0	2.0	4.0	3.0	3.0	6.0	5.0	7.0	5.0	8.0
ColumnIndex	1	2	4	1	2	3	2	3	4	1	3	4
RowPtr	1	4	7	10	13							
DiagonalPtr	1	5	8	12								

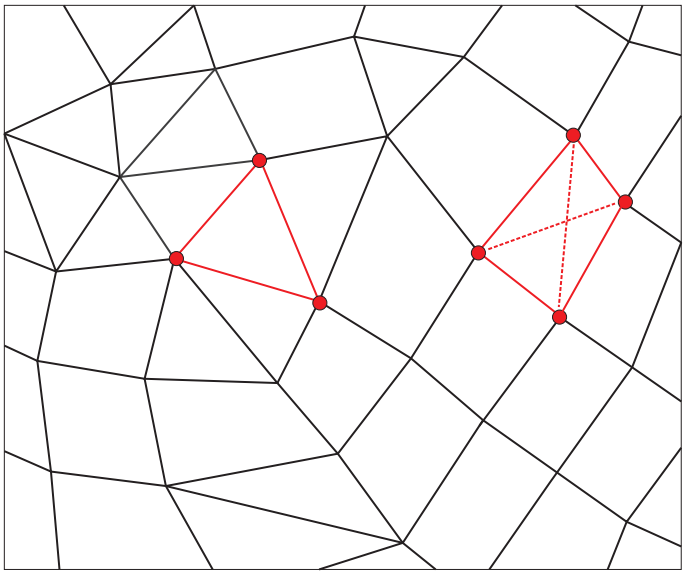


Fig. B.1. Types of edges for mixed bi-/linear finite elements.

For the algebraic flux correction approach [144] used in this work, edge-by-edge traversal of the matrix K is crucial since discrete diffusion coefficients and (anti-)diffusive fluxes are associated with the edges of the sparsity graph. In particular, there exists an edge between nodes i and j if they share a common element as depicted in Figure B.1 for a mixed linear/bilinear finite element discretization employed on an unstructured grid in two space dimensions. For linear finite elements, the above definition of edges coincides with that of the physical edges (indicated by straight lines) connecting the corner vertices. However, the use of bilinear elements gives rise to additional internal edges (indicated by dashed lines) connecting opposite vertices. In three dimensions, twelve internal edges need to be considered for hexahedral elements; two edges for each face plus four edges connecting diametrically opposed vertices.

An elegant algorithm for an edge-by-edge traversal of sparse finite element matrices in CRS7 format was developed by Kuzmin [131]. In essence, the extra vector `SepPtr(1:nr)` is introduced and used to separate the lower left from the upper right triangular matrix. To this end, each entry is initialized by the global position of the first off-diagonal entry right to the diagonal, i.e. `SepPtr(i)=ipos` such that `ColumnIndex(ipos)>i` and `ColumnIndex(ipos-1)<i` except for the first row. Furthermore, the auxiliary vector `AuxPtr(1:nr)` is required which is initialized by the values of `SepPtr(1:nr)` and updated step-by-step. The lower left part of the matrix is traversed row-by-row (i.e. from `RowPtr(i)+1` to `SepPtr(i)`) and the off-diagonal coefficient that corresponds to the entry ij is processed. The global position of its counterpart ji is computed as follows: `AuxPtr(j)=AuxPtr(j)+1` so that `ji=AuxPtr(j)`. Note that two additional vectors are required for this algorithm, and moreover, the re-initialized needs to be performed before each traversal.

We propose a modified version of the above algorithm applicable to sparse matrices in arbitrary CRS formats which requires only one extra vector `SepPtr(1:nr)`. It can be operated forward and backward so that multiple matrix traversals can be performed without the need to re-initialize some vectors. An edge-based traversal algorithm for matrices stored in CRS9 format is as follows:

```
1  SepPtr(1:nr)=RowPrt(1:nr)
2  do i=1,nr
3    ii=DiagonalIndex(i)
4    do ij=SepPtr(i)+1,RowPtr(i+1)-1
5         j=ColumnIndex(ij)
6        jj=DiagonalIndex(j)
7        ji=SepPtr(j)
8           SepPtr(j)=SepPtr(j)+1
9
10          !! do something !!
11
12   end do
13 end do
```

Alg. B.1: Edge-based traversal of matrices stored in CRS9 format.

At the beginning, the array `SepPtr` points to the positions that start each row. The first loop (lines 2–13) iterates over the rows of the matrix from top to bottom. In the second loop (lines 4–12), all off-diagonal positions in the upper right part are visited consecutively. If the matrix position `ij` is processed, then the separator for the j^{th} row points to its corresponding counterpart `ji` in the lower-left triangular part. This is due to the fact that the connectivity graph of finite element matrices is symmetric. Finally, the entry j of the separator array is increased by one (line 8) so as to point to the location of the next matrix coefficient in the j^{th} row. This update guarantees that `SepPtr(j)` provide the position of the diagonal entry when the outer loop has reached row j,

and hence, the inner loop starts at the first off-diagonal entry in the upper right part. The actual procedure which performs the matrix and/or vector manipulation strongly depends on the concrete algorithm and can be placed within the inner loop as indicated in the above algorithm.

The presented traversal strategy resembles its predecessor [131] which was designed for sparse matrices stored in the CRS7 format. In fact, procedure B.1 can be easily adapted to leading diagonals. To this end, the vector `DiagonalIndex` is replaced by `RowPtr` in lines 3 and 6 since it allows to directly address the diagonal coefficients. Owing to the leading position of the matrix diagonal, `SepPtr` initially points to the position of the diagonal coefficients which need to be 'skipped'. Thus, lines 7 and 8 need to be swapped so that the diagonal separator `SepPtr(j)` for row j is increased prior to recovering the position of the lower-left off-diagonal entry `ji`.

2.2. Practical aspects of block matrices

The compressed row storage format devised for a scalar matrix K can be readily extended to block matrices used to represent systems of equations such as the compressible Euler equations discussed in Chapter 5. Consider the global operator $\mathrm{K} = \{\mathrm{K}_{ij}\}$ which represents a $N_\mathrm{U} \times N_\mathrm{U}$ block matrix applicable to the multi-component solution vector $\mathrm{U} = [\mathrm{U}_1, \ldots, \mathrm{U}_{N_\mathrm{U}}]^T$, where each scalar sub-vector U_k comprises the nodal solution values of the particular variable k. Of course, each block can be stored individually as scalar sub-matrix. By virtue of the group formulation [78], all blocks exhibit the same connectivity graph so that it suffices to maintain the auxiliary vectors `RowPtr`, `ColumnIndex` and `DiagonalIndex` only once. The actual data can be saved in the three-dimensional array `K(1:nnz,1:nu,1:nu)`, whereby the leading dimension is assumed to be stored contiguously in memory. The resulting memory layout of the matrix K is sketched in Figure B.2.

Let the operator K be assembled edge-by-edge following the update procedure [142]

$$\begin{aligned} \mathrm{K}_{ii} &:= \mathrm{K}_{ii} + \mathrm{A}_{ij} + \mathrm{S}_{ij}, & \mathrm{K}_{ij} &:= -\mathrm{A}_{ij} - \mathrm{S}_{ij} \\ \mathrm{K}_{jj} &:= \mathrm{K}_{jj} - \mathrm{A}_{ij} + \mathrm{S}_{ij}, & \mathrm{K}_{ji} &:= \quad \mathrm{A}_{ij} - \mathrm{S}_{ij} \end{aligned} \tag{B.1}$$

Provided the indices ii, ij, ji and jj are known, the local matrices A_{ij} and S_{ij} can be evaluated efficiently and scattered to their corresponding position in the global operator. Note that for each edge there are `4×nu×nu` *isolated* memory positions involved in the update procedure (B.1). As a consequence, matrix assembly may be time consuming since memory alignment techniques and optimized prefetching and caching strategies of modern hardware processors cannot be exploited. On the other hand, each block represents a scalar finite element matrix so that many routines of an existing CFD code such as matrix-vector multiplication can be reused without modifications.

As an alternative to the blockwise memory alignment, the matrix can be grouped locally as illustrated in Figure B.3. In this case, the matrix data resides in the three-dimensional array `K(1:nu,1:nu,1:nnz)`, whereby the local blocks constitute the two leading dimensions and the actual sparsity structure lags behind. Let K be assembled in a loop over edges adopting the update procedure (B.1). As a matter of fact, each edge gives rise to only four jumps in memory; twice to address the diagonal positions and two times for the off-diagonal entries. Once the starting address of a block has been reached in memory, the local matrices A_{ij} and S_{ij} can be applied en bloc. However, the reimplementation of most matrix routines such as matrix-vector multiplication, matrix factorization or matrix preconditioning is mandatory if such localized memory alignment is employed. On the other hand, this extra effort clearly pays off since the global matrix assembly is a crucial component of implicit schemes that consumes most of the CPU time. This specialized memory arrangement also needs to be reflected in the global vectors for the solution and the right-hand side and/or defect vector. Moreover, this approach relies on the group finite element formulation [78] which guarantees that all matrix blocks share the same sparsity pattern.

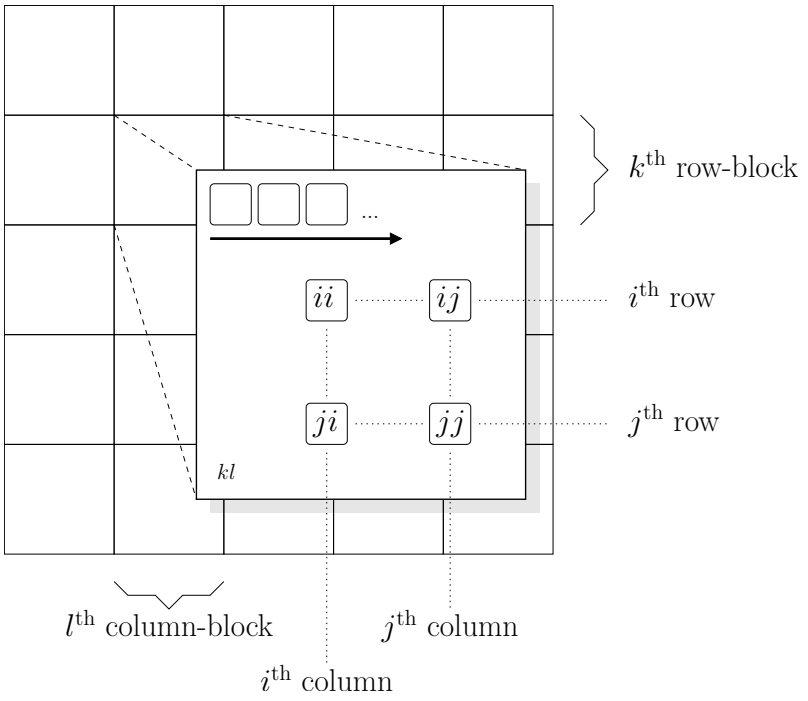


Fig. B.2. Memory layout for globally blocked matrices.

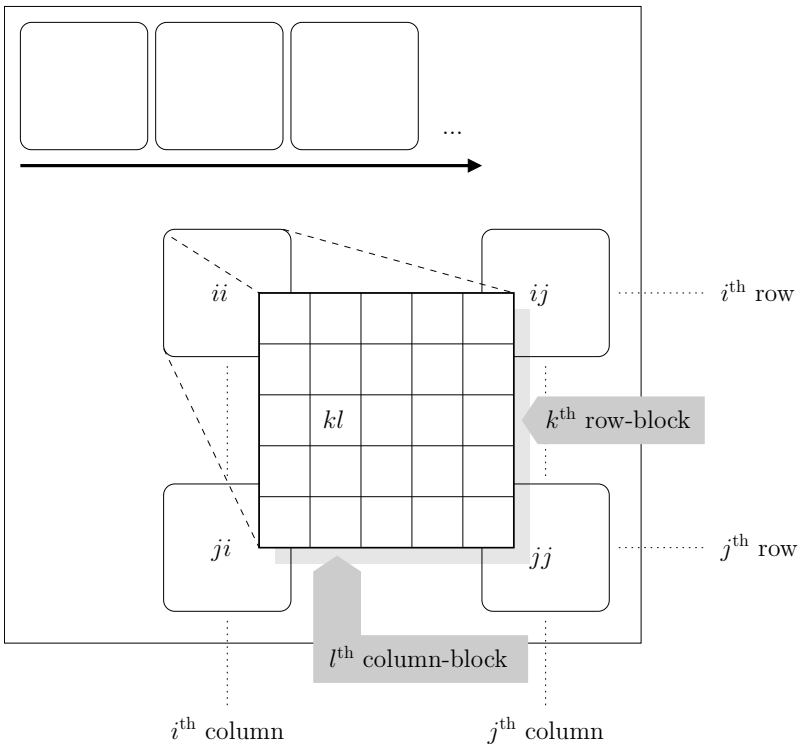


Fig. B.3. Memory layout for locally blocked matrices.

C

Jacobian and Transformation Matrices

In this appendix, we present the Jacobian matrices for the two-dimensional Euler equations and the corresponding transformation matrices given by the left and right eigenvectors. The Jacobian matrices $\mathbf{A} = (A^1, A^2)$ for the inviscid fluxes in two dimensions are given by [149, 251]

$$A^1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \gamma_2 v_1^2 + \gamma_1 v_2^2 & (3-\gamma)v_1 & (1-\gamma)v_2 & \gamma - 1 \\ -v_1 v_2 & v_2 & v_1 & 0 \\ \gamma_1(v_1^3 + v_2^2 v_1) - H v_1 & H - (\gamma-1)v_1^2 & (1-\gamma)v_1 v_2 & \gamma v_1 \end{bmatrix} \tag{C.1}$$

and

$$A^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -v_1 v_2 & v_2 & v_1 & 0 \\ \gamma_2 v_2^2 + \gamma_1 v_1^2 & (1-\gamma)v_1 & (3-\gamma)v_2 & \gamma - 1 \\ \gamma_1(v_1^2 v_2 + v_2^3) - H v_2 & (1-\gamma)v_1 v_2 & H - (\gamma-1)v_2^2 & \gamma v_2 \end{bmatrix} \tag{C.2}$$

where the auxiliary quantities γ_1 and γ_2 are defined as follows:

$$\gamma_1 = \frac{\gamma - 1}{2}, \qquad \gamma_2 = \frac{\gamma - 3}{2} \tag{C.3}$$

For the numerical treatment it may be worthwhile to consider an arbitrary linear combination

$$A_e := \mathbf{A}(U, \mathbf{e}) = e_1 A^1 + e_2 A^2, \qquad \mathbf{e} = (e_1, e_2), \qquad e_1^2 + e_2^2 = 1 \tag{C.4}$$

Let $v_e := \mathbf{e} \cdot \mathbf{v}$ denote the 'projected' velocity so that the Jacobian A_e reads as follows [225]

$$A_e = \begin{bmatrix} 0 & e_1 & e_2 & 0 \\ (\gamma-1)q e_1 - v_1 v_e & v_e - (\gamma-2)v_1 e_1 & u_1 e_2 - (\gamma-1)v_2 e_1 & (\gamma-1)e_1 \\ (\gamma-1)q e_2 - v_2 v_e & v_2 e_1 - (\gamma-1)v_1 e_2 & v_e - (\gamma-2)v_2 e_2 & (\gamma-1)e_2 \\ [(\gamma-1)q - H]v_e & H e_1 - (\gamma-1)v_1 v_e & H e_2 - (\gamma-1)v_2 v_e & \gamma v_e \end{bmatrix} \tag{C.5}$$

where the auxiliary quantity $q = |\mathbf{v}|/2$ denotes the magnitude of the velocity. Two eigenvalues of the cumulated Jacobian A_e are distinct and two are repeated, that is,

$$\Lambda_e = \mathrm{diag}\{v_e - c, v_e, v_e + c, v_e\} \tag{C.6}$$

A viable choice for the matrix of right eigenvectors $R_e := R(U,\mathbf{e})$ is as follows: [225]

$$R_e = \begin{bmatrix} 1 & 1 & 1 & 0 \\ v_1 - ce_1 & v_1 & v_1 + ce_1 & e_2 \\ v_2 - ce_2 & v_2 & v_2 + ce_2 & -e_1 \\ H - cv_e & q & H + cv_e & v_1e_2 - v_2e_1 \end{bmatrix} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4] \tag{C.7}$$

From the above expression, the matrices for the uni-directional Jacobians $A^d, d = 1,2$ can be readily obtained by letting the directional vector satisfy $\mathbf{e} = [1,0]$ and $\mathbf{e} = [0,1]$, respectively [251]:

$$R^1 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ v_1 - c & v_1 & v_1 + c & 0 \\ v_2 & v_2 & v_2 & -1 \\ H - cv_1 & q & H + cv_1 & -v_2 \end{bmatrix} \qquad R^2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ v_1 & v_1 & v_1 & 1 \\ v_2 - c & v_2 & v_2 + c & 0 \\ H - cv_2 & q & H + cv_2 & v_1 \end{bmatrix} \tag{C.8}$$

It should be noted that the eigenvectors of the repeated eigenvalues $\lambda_2 = \lambda_4 = v_e$ are not linearly dependent. Mathematically speaking, both the geometric and the algebraic multiplicity coincide so that a complete family of right eigenvectors is available, i.e., R_e. This is not the case in three space dimensions [225]. The algebraic multiplicity of the repeated eigenvalue v_e is three but there exist only two linearly independent eigenvectors which span a two dimensional subspace of $\mathbb{R}^5$.

Let us return to the two-dimensional Euler equations. The left eigenvectors can be determined from the rows of the inverse of the nonsingular matrix R_e which yields [225]

$$L_e = \begin{bmatrix} \frac{1}{2}(b_1 + \frac{v_e}{c}) & \frac{1}{2}(-b_2v_1 - \frac{e_1}{c}) & \frac{1}{2}(-b_2v_2 - \frac{e_2}{c}) & \frac{1}{2}b_2 \\ 1 - b_1 & b_2v_1 & b_2v_2 & -b_2 \\ \frac{1}{2}(b_1 - \frac{v_e}{c}) & \frac{1}{2}(-b_2v_1 + \frac{e_1}{c}) & \frac{1}{2}(-b_2v_2 + \frac{e_2}{c}) & \frac{1}{2}b_2 \\ \frac{v_2 - v_ee_2}{e_1} & e_2 & \frac{e_2^2 - 1}{e_1} & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{l}_1 \\ \mathbf{l}_2 \\ \mathbf{l}_3 \\ \mathbf{l}_4 \end{bmatrix} \tag{C.9}$$

where the auxiliary quantities q, b_1 and b_2 are defined as follows

$$q = \frac{|\mathbf{v}|}{2}, \qquad b_1 = b_2q, \qquad b_2 = \frac{\gamma - 1}{c^2} \tag{C.10}$$

Note that the matrix of left eigenvector is singular for $e_1 = 0$ due to the fourth row. However, this singularity can be eliminated by replacing the first and third entry of the fourth row by [225]

$$\frac{v_2 - v_ee_2}{e_1} = \frac{v_2 - (e_1v_1 + e_2v_2)e_2}{e_1} = \frac{v_2 - v_1e_1e_2 - v_2(1 - e_1^2)}{e_1} = e_1v_e - e_2v_1 \tag{C.11}$$

and

$$\frac{e_2^2 - 1}{e_1} = \frac{-e_1^2}{e_1} = -e_1 \tag{C.12}$$

respectively. It is worth mentioning that this elegant remedy is no longer possible in three space dimensions. Hence, different matrices of left eigenvectors need to be utilized depending on the direction of $\mathbf{e} = [e_1, e_2, e_3]$ in order to avoid the formation of singularities [225].

D

Proof of Theorem 4.6.1

Let us represent the characterization of elements introduced in Section 4.6.3 under consideration:

Theorem 4.6.1 (Characterization of elements in 2D) Let $\mathcal{T}_m = (\mathcal{E}_m, \mathcal{V}_m)$, $m = 0, 1, \ldots$ be an arbitrary conformal triangulation that is constructed from $\mathcal{T}_0$ by m refinement/re-coarsening steps.

(a) If all nodes of a triangle/quadrilateral Ω_k have generation number zero, then $\Omega_k \in \mathcal{E}_0$, that is, the element belongs to the initial triangulation $\mathcal{T}_0$.

(b) If three nodes of a quadrilateral Ω_k have the same generation number, then Ω_k is a red element which results from $1Q \rhd 4Q$ refinement.

(c) If all nodes of a triangle Ω_k have the same generation number, then Ω_k is an inner red element resulting from $1T \rhd 3T$ refinement.

(d) If exactly two consecutive nodes of a quadrilateral Ω_k have largest (positive) generation number (in the cell), then the connecting edge was introduced by a green $1Q \rhd 2Q$ refinement and the adjacent element Ω_l constitutes the corresponding green neighbor.

(e) Let two nodes of a triangle Ω_k have largest generation number (in the cell).

  (e.1) If the neighbor along the edge connecting these two vertices is an inner red triangle, then Ω_k is an outer red triangle resulting from $1T \rhd 3T$ refinement. The third vertex which has smaller generation number coincides with one node of the macro element.

  (e.2) Otherwise (i.e. there is no adjacent inner red triangle), element Ω_k is a green triangle which results from $1Q \rhd 4T$ refinement of a quadrilateral into four triangles.

(f) Let one node, say v_i, of a triangle Ω_k possess largest generation number (in the cell).

  (f.1) If there exists a neighboring triangle Ω_l with two vertices with largest generation number, then Ω_k is one of the two outer green triangles resulting from $1Q \rhd 4T$ refinement.

  (f.2) If there exists exactly *one* triangle Ω_l meeting at the common node v_i which has largest generation number (in cell Ω_l), then both elements result from $1T \rhd 2T$ refinement of the original macro triangle $\Omega_k \cup \Omega_l$ into two green triangles.

  (f.3) If there exist exactly *two* triangles Ω_l meeting at the common node v_i which has largest generation number (in all cells), then the three elements result from $1Q \rhd 3T$ refinement of the original macro quadrilateral into three green triangles.

Proof. (a) The correctness of this proposition follows by construction of the nodal generation function (4.6.2) since all vertices that belong to the initial triangulation $\mathcal{T}_0$ have zero age.

(b) Let Ω_k be a quadrilateral of the initial grid whose nodes v_{i_a} with $a = 1,2,3,4$ belong to generation zero by construction, i.e. $\mathrm{v}_{i_a} \in \mathcal{G}_0$. If we subdivide Ω_k regularly into the four quadrilaterals $\Omega_{k_1}, \ldots, \Omega_{k_4}$, the five inserted nodes v_{i_b} with $b = 5, \ldots, 9$ belong to the first vertex generation $\mathcal{G}_1$ by definition of the generation function (4.6.2). Consequently, each red quadrilateral Ω_{k_a} with $a = 1,2,3,4$ has exactly three vertices with same positive age one (see Fig. D.1). Note

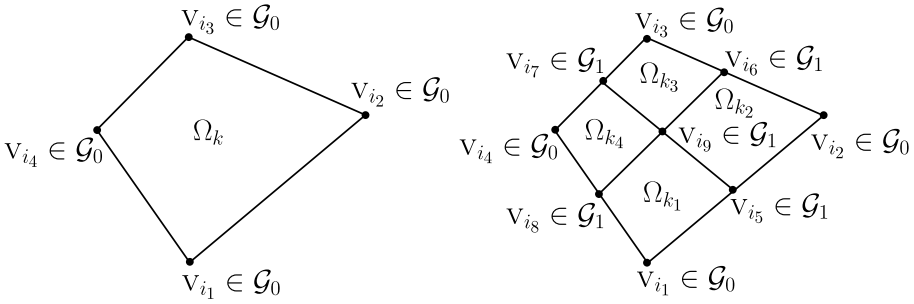


Fig. D.1. Regular subdivision of a quadrilateral from the initial triangulation.

that green quadrilaterals need to be converted to red ones prior to performing further refinement. It therefore suffices to start from a red quadrilateral and prove proposition (b) by induction. Without loss of generality let Ω_k be a red quadrilateral for which three nodes belong to $\mathcal{G}_q$. The remaining vertex coincides with a node of the macro elements and belongs to some previous generation, say, $\mathcal{G}_{q-r}$, where $r > 0$. Owing to the generation function (4.6.2) all five vertices satisfy $\mathrm{v}_{i_b} \in \mathcal{G}_{q+1}$ for $b = 5, \ldots, 9$ so that each red subelement has exactly three nodes with the same age (see Fig. D.2).

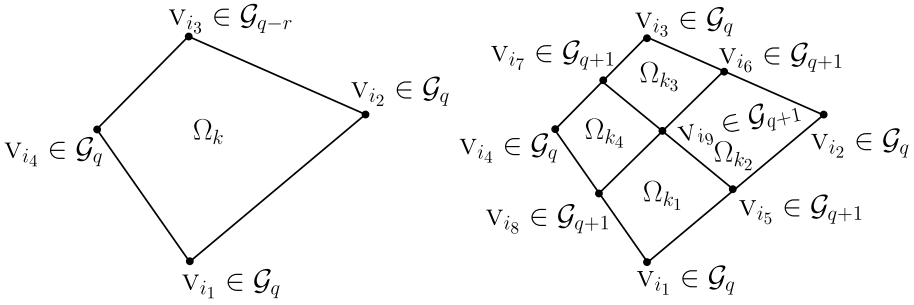


Fig. D.2. Regular subdivision of a quadrilateral from an arbitrary triangulation.

(d) Obviously, $1Q \triangleright 2Q$ refinement can only be applied to an element from the initial triangulation (cf. proposition (a)) or to a red quadrilateral which is characterized by the presence of exactly three vertices that exhibit the same age (cf. proposition (b)). Since green elements cannot be refined it suffices to consider all possible situations to prove proposition (d) which is trivial.

(c) Let Ω_k be a triangle of the initial grid whose nodes v_{i_a} with $a = 1,2,3$ belong to generation zero. If we subdivide Ω_k regularly into the four triangles $\Omega_{k_1}, \ldots, \Omega_{k_4}$ all newly introduced vertices v_{i_b} with $b = 4,5,6$ belong to the first generation so that all nodes of the interior triangle are of unit age. Let Ω_k be an *inner* red triangle for which all vertices have the same age $\mathcal{G}_q$. If $1T \triangleright 4T$ refinement is applied to Ω_k, three new vertices are inserted at the midpoints of the edges. Since all nodes of the macro element Ω_k belong to generation $\mathcal{G}_q$, the generation number of the new vertices

is given by $g(\mathrm{v}_{i_b}) = q+1$ for $b = 4,5,6$. Thus, all nodes of the inner red triangle satisfy $\mathrm{v}_{i_b} \in \mathcal{G}_{q+1}$ for $b = 4,5,6$ which proves proposition (c) for an inner triangle (see Fig. D.3).

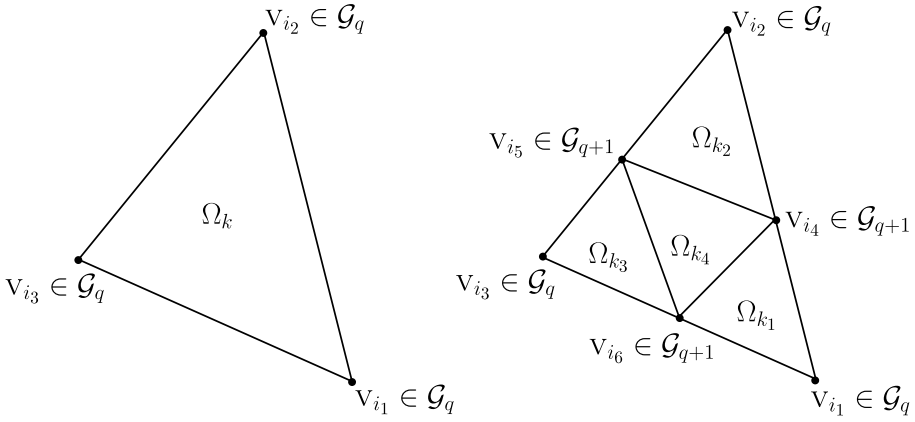


Fig. D.3. Regular subdivision of an inner red triangle.

In order to show that proposition (c) is also true if regular refinement is applied to an *outer* red triangle assume that proposition (e.1) holds for the time being. That is, exactly two vertices of the outer red triangle Ω_k have the same age, i.e. $g(\mathrm{v}_{i_1}) = g(\mathrm{v}_{i_2}) = q$, and the third node v_{i_3} coincides with some vertex of the macro element, and hence, it belongs to $\mathcal{G}_{q-r}$, where $r > 0$. By construction of the generation function (4.6.2), the age of the new vertices v_{i_b} with $b = 4,5,6$ equals $q+1$ so that all nodes of an inner red triangle feature the same generation (see Fig. D.4).

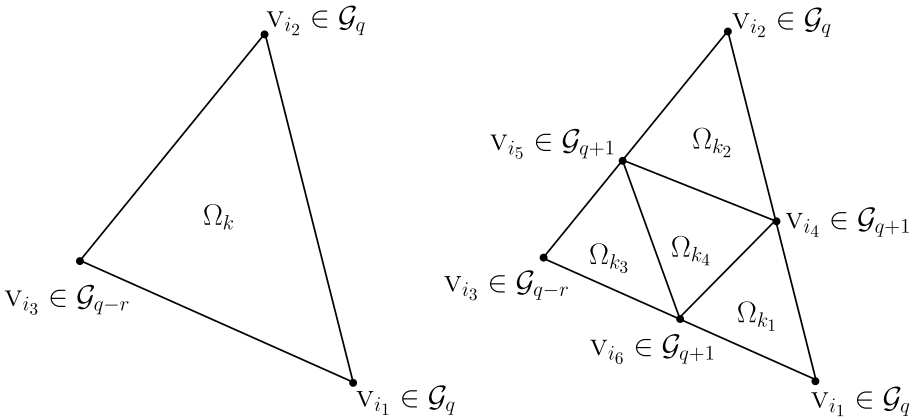


Fig. D.4. Regular subdivision of an outer red triangle.

(e.1) Following the details presented above, it is clear that all outer red triangles resulting from regular refinement of the initial triangulation possess two nodes belonging to the first generation, whereas one vertex has age zero. To prove proposition (e.1) by induction, let us consider an *inner* red triangle Ω_k and perform regular subdivision into four triangles $\Omega_{k_1}, \ldots, \Omega_{k_4}$. It follows from proposition (c) that the three vertices v_{i_a} for $a = 1,2,3$ of the macro element Ω_k belong to the same generation $\mathcal{G}_q$, so that all new nodes v_{i_b} for $b = 4,5,6$ inserted at the midpoints of edges possess age $q+1$. In other words, each outer triangle $\Omega_{k_1}, \Omega_{k_2}, \Omega_{k_3}$, is connected to one vertex v_{i_a} with $a \in \{1,2,3\}$ of the macro element Ω_k and exactly two new nodes v_{i_b} with $b \in \{4,5,6\}$. The latter ones represent the corners of the interior red triangle Ω_{k_4} (see Fig. D.3).

Conversely, let Ω_k be an *outer* red triangle that is refined regularly (see Fig. D.4). Without loss of generality let $\mathrm{v}_{i_1}, \mathrm{v}_{i_2} \in \mathcal{G}_q$ so that the third vertex $\mathrm{v}_{i_3} \in \mathcal{G}_{q-r}$, where $r > 0$ belongs to the macro element. By construction of the generation function (4.6.2), all new vertices v_{i_b} with $b = 4,5,6$ are assigned age $q+1$. Hence, each outer triangle $\Omega_{k_1}, \Omega_{k_2}, \Omega_{k_3}$ shares one common vertex v_{i_a} for $a \in \{1,2,3\}$ with the macro element Ω_k, whereas the remaining nodes satisfy $\mathrm{v}_{i_b} \in \mathcal{G}_{q+1}$, where $b \in \{4,5,6\}$. The latter ones constitute the corners of the interior triangle Ω_{k_4} so that (e.1) holds.

Let us emphasize the fact that green elements cannot be refined further so that the remaining propositions can be shown step-by-step without the need to adopt induction arguments.

(f.2) Let the element Ω_k be subdivided into two green triangles. Note that the bisected edge must be shared by two red quadrilaterals or red triangles since green refinement cannot take place by its own. All possible cases are depicted in Figure D.5. The left sketch corresponds to an element of the initial triangulation, whereas the right figure illustrates the situation for an inner ($r = 0$) and outer ($r > 0$) triangle, respectively. There exist exactly two triangles sharing the unique vertex v_{i_4} with largest generation number. By construction, all other elements meeting at node v_{i_4} have two (outer red triangle) and three (inner red triangle/quadrilateral) vertices belonging to generation $\mathcal{G}_{q+1}$, respectively. Thus, cells resulting from $1T \rhd 2T$ refinement can be identified by (f.2).

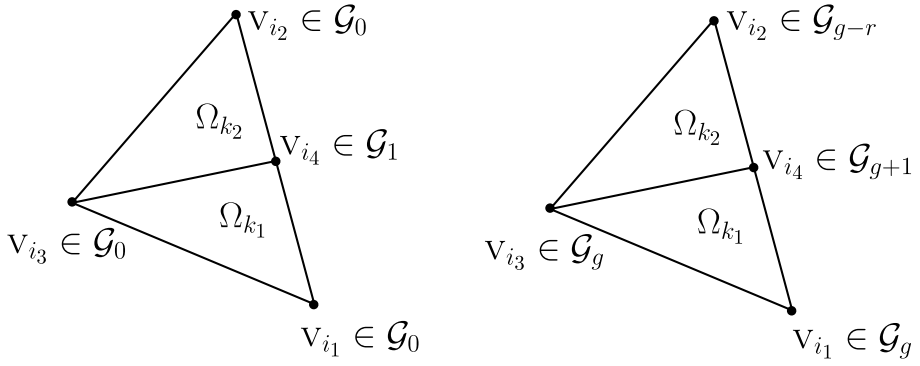


Fig. D.5. Bisection of a triangle into two green elements.

(f.3) Let the quadrilateral Ω_k be subdivided into three green triangles as depicted in Figure D.6. Obviously, the common vertex v_{i_4} is the most recent one. It is therefore the unique node that features largest generation number $\mathcal{G}_{q+1}$ in each subelement $\Omega_{k_1}, \Omega_{k_2}, \Omega_{k_3}$. Arguing as above, i.e. there are only red elements adjacent to the bisected edge), all other cells meeting at node v_{i_4} must have at least two vertices of age $q+1$ which completes the proof of proposition (f.3).

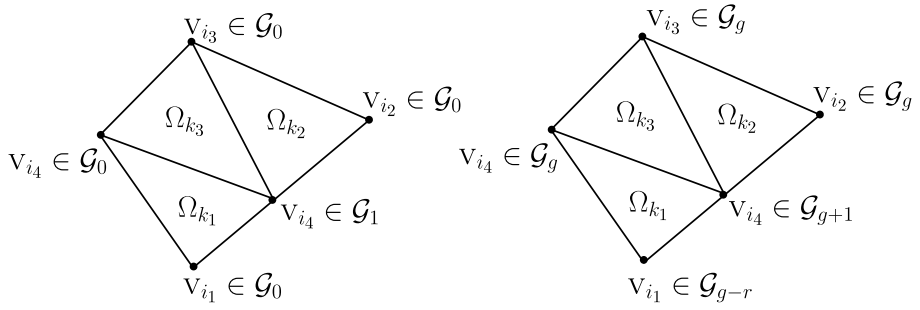


Fig. D.6. Subdivision of a quadrilateral into three green triangles.

(e.2) and (f.1) Consider the quadrilateral Ω_k and perform $1Q \triangleright 4T$ refinement into four green triangles $\Omega_{k_1}, \ldots \Omega_{k_4}$ as illustrated in Figure D.7. By definition of the generation function (4.6.2), element Ω_{k_1} has two nodes which feature largest age $\mathcal{G}_{q+1}$ ($q = 0$ left and $q > 0$ right) but the elements adjacent to the corresponding edge does not satisfy property (c) of an inner red triangle. Hence, proposition (e.1) does not hold for the element under consideration so that Ω_{k_1} is a green triangle resulting from a $1Q \triangleright 4T$ refinement. The same argumentation can be applied to Ω_{k_3}.

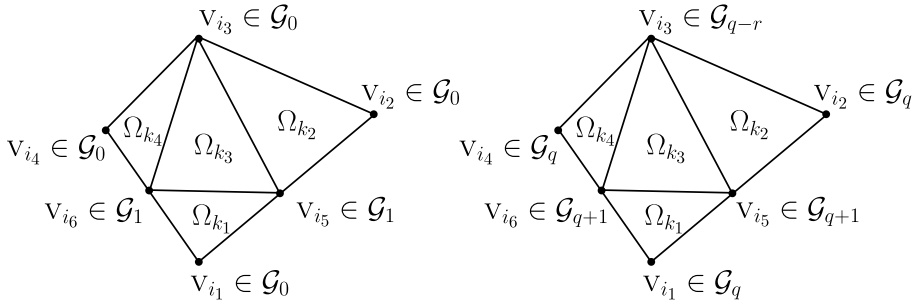


Fig. D.7. Subdivision of a quadrilateral into four green triangles.

It follows from Figure D.7 that vertex v_{i_5} is the unique node of element Ω_{k_2} that has largest age. All other cells meeting at Ω_{k_2} possess one or more vertices of the same generation. For the green triangles resulting from the $1Q \triangleright 4T$ refinement this property can be readily seen from the above illustration. Moreover, elements which are adjacent to the bisected edges must be either red triangles or quadrilaterals, and therefore, two or three vertices possess largest age (cf. propositions (b), (c) and (e.1) above). In fact, the same arguments can be applied to the triangle Ω_{k_4}. □

Bibliography

[1] S. Adjerid, J.E. Flaherty, P.K. Moore, and Y.J. Wang. High-order adaptive methods for parabolic systems. *Physica D*, 60(1–4):94–111, 1992.

[2] M. Aftosmis and N. Kroll. A quadrilateral based second-order TVD method for unstructured adaptive methods. *AIAA Paper 91-0124*, 1991. 29th Aerospace Sciences Meeting, Reno, Nevada, USA, 7–10 January 1991.

[3] M. Ainsworth and J.T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. Pure and Applied Mathematics: A Wiley-Interscience Publication. John Wiley and Sons, New York, 2000.

[4] M. Ainsworth, J.Z. Zhu, A.W. Craig, and O.C. Zienkiewicz. Analysis of the Zienkiewicz-Zhu a-posteriori error estimator in the finite element method. *Internat. J. Numer. Methods Engrg.*, 28(9):2161–2174, 1989.

[5] J.E. Akin. *Finite Element Analysis With Error Estimators: An Introduction to the FEM and Adaptive Error Analysis for Engineering Students*. Elsevier, Amsterdam, 2005.

[6] D.A. Anderson, J.C. Tannenhill, and R.H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Hemisphere Publishing Corporation, 1984.

[7] Jr. J.D. Anderson. *Modern Compressible Flow With Historical Perspective*. Series in Aeronautical and Aerospace Engineering. McGraw-Hill Publishing Company, 1990.

[8] J.D. Andersson. *Computational Fluid Dynamics – The Basics with Applications*. McGraw-Hill, 1995.

[9] P. Arminjon and A. Dervieux. Construction of TVD-like artificial viscosities on two-dimensional arbitrary FEM grids. *J. Comput. Phys.*, 106(1):176–198, 1993.

[10] Athena test suite. `http://www.astro.virginia.edu/VITA/ATHENA/dmr.html`. Virginia Institute of Theoretical Astronomy.

[11] K. Baba and M. Tabata. On a conservative upwind finite element scheme for convection diffusion equations. *R.A.I.R.O. Anal. Numér.*, 15:3–25, 1981.

[12] I. Babuška and W.C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15(4):736–354, 1978.

[13] I. Babuška and W.C. Rheinboldt. A posteriori error estimates for the finite element method. *Internat. J. Numer. Methods Engrg.*, 12(10):1597–1615, 1978.

[14] I. Babuška and T. Strouboulis. *The Finite Element Method and its Reliability*. Oxford University Press, 2001.

[15] I. Babuška, O.C. Zienkiewicz, and J. Gago. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*. John Wiley and Sons, New York, 1986.

[16] N. Balakrishnan and G. Fernandez. Wall boundary conditions for inviscid compressible flows on unstructured meshes. *Internat. J. Numer. Methods Fluids*, 28:1481–1501, 1998.

[17] W. Bangerth and R. Rannacher. *Adaptive Finite Element Methods for Differential Equations*. Lectures in Mathematics. Birkhäuser, Basel, 2003.

[18] E.B. Bank and C.C. Douglas. SMMP: Sparse matrix multiplication package. *Adv. Comput. Math.*, 1(1):127–137, 1993.

[19] R.E. Bank, A.H. Sherman, and A. Weiser. Some refinement algorithms and data structures for regular local mesh refinement. In R. Stepleman, editor, *Scientific Computing, Applications of Mathematics and Computing to the Physical Sciences*, volume I of *IMACS Transactions on Scientific Computation*, pages 3–17. North-Holland, Amsterdam, 1983.

[20] R.E. Bank and A. Weiser. Some a posteriori error estimates for elliptic partial differential equations. *Math. Comp.*, 44(170):283–301, 1985.

[21] T.J. Barth. Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes. *AIAA Paper 91-0721*, 1991.

[22] T.J. Barth, editor. *Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations*, 1992.

[23] T.J. Barth and H. Deconinck, editors. *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*. Lecture Notes in Computational Science and Engineering. Springer Verlag, Berlin, 2003.

[24] J.D. Baum and R. Löhner. Numerical simulation of pilot/seat ejection from an F-16. *AIAA Paper 93-0783*, 1993.

[25] J.D. Baum, H. Luo, and R. Löhner. Validation of a new ALE adaptive unstructured moving body methodology for multi-store ejection simulations. *AIAA Paper 95-1792*, 1995.

[26] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–101, 2001.

[27] F. Benkhaldoun, T. Fernandez, B. Larrouturou, and P. Leyland. A dynamical adaptive method based on local refinement and unrefinement for triangular finite-element meshes: Preliminary results. Technical Report RR-1270, INRIA, 1990.

[28] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, 1989.

[29] M.J. Berger and R.J. LeVeque. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. *AIAA Paper 89-1930*, 1989.

[30] M.J. Berger and R.J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35(6):1439–1461, 1998.

[31] M.J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53(3):484–512, 1984.

[32] M. Berzins. Mesh quality: A function of geometry, error estimates or both? *Engrg. Comput.*, 15(3):236–247, 1999.

[33] M. Berzins. Modified mass matrices and positivity preservation for hyperbolic and parabolic PDEs. *Comm. Numer. Methods Engrg.*, 17(9):659–666, 2001.

[34] M. Berzins, P.K. Jimack, M. Walkley, and L.J.K. Durbeck. Mesh quality and moving meshes for 2D and 3D unstructured mesh flow solvers. In N.P. Weatherill and H. Deconinck, editors, *VKI Lecture Series 2000-05, 31st Computational Fluid Dynamics*, 2000.

[35] K.S. Bey and J.T. Oden. hp-version discontinuous Galerkin methods for hyperbolic conservation laws. *Comput. Methods Appl. Mech. Engrg.*, 133:259–286, 1996.

[36] M.B. Bieterman, J.E. Bussoletti, C.L. Hilmers, F.T. Johnson, R.G. Melvin, and D.P. Young. An adaptive grid method for analysis of 3D aircraft configurations. *Comput. Methods Appl. Mech. Engrg.*, 101(1–3):225–249, 1992.

[37] H. Blum, J. Harig, S. Müller, and S. Turek. *FEAT2D. Finite Element Analysis Tools. User Manual. Release 1.3*. Universität Heidelberg, Institut für Angewandte Mathematik, 1995.

[38] J.P. Boris. Flux-corrected transport modules for solving generalized continuity equations. *N.R.L. Memorandum Report*, 1976.

[39] J.P. Boris and D.L. Book. Flux-corrected transport. I. SHASTA, A fluid transport algorithm that works. *J. Comput. Phys.*, 11:38–69, 1973.

[40] A. Bowyer. Computing Dirichlet tessellations. *Comput. J.*, 24(2):162–166, 1981.

[41] D. Braess. *Finite elements - Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 2001.

[42] A. Brandt. Multi-level adaptive computations in fluid dynamics. *AIAA Journal*, 18(10):1165–1172, 1980.

[43] E. Burman and A. Ern. Stabilized Galerkin approximation of convection-diffusion-reaction equations: Discrete maximum principle and convergence. *Math. Comp.*, 74(252):1637–1652, 2005.

[44] E. Burman and P. Hansbo. Edge stabilization for Galerkin approximations of convection-diffusion-reaction problems. *Comput. Methods Appl. Mech. Engrg.*, 193(15–16):1437–1453, 2005.

[45] G. Cantin, C. Loubignac, and C. Touzot. An iterative scheme to build continuous stress and displacement solutions. *Internat. J. Numer. Methods Engrg.*, 12:1493–1506, 1978.

[46] P.J. Capon and P.K. Jimack. An inexact Newton method for systems arising from the finite element method. *Appl. Math. Lett.*, 10(3):9–12, 1997.

[47] J.-C. Carette, H. Deconinck, H. Paillère, and P.L. Roe. Multidimensional upwinding: Its relation to finite elements. *Internat. J. Numer. Methods Fluids*, 20(8–9):935–955, 1995.

[48] G.F. Carey. *Computational Grids: Generation, Adaptation, and Solution Strategies*. Taylor and Francis, Washington D.C., 1997.

[49] M.J. Castro-Díaz, F. Hecht, and B. Mohammadi. New progress in anisotropic grid adaptation for inviscid and viscous flow simulation. In *Proceedings of the 4th Annual International Meshing Roundtable, Sandia National Laboratories*, pages 73–85, 1995.

[50] R. Choquet. A matrix-free preconditioner applied to CFD. Technical Report RR-2605, INRIA, 1995.

[51] B. Cockburn and H. Gau. A posteriori error estimates for general numerical methods for scalar conservation laws. *J. Comput. Appl. Math.*, 14:37–47, 1995.

[52] B. Cockburn, G.E. Karniadakis, and C.-W. Shu. The development of discontinuous Galerkin methods. In *Discontinuous Galerkin methods. Theory, computation and applications*, volume 11 of *Lecture Notes in Computational Science and Engineering*, pages 3–50. Springer Verlag, 2000.

[53] B. Cockburn and C.-W. Shu. TVD Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework. *Math. Comp.*, 52:411–435, 1989.

[54] B. Cockburn and C.-W. Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws. V. Multidimensional systems. *J. Comput. Phys.*, 141:199–224, 1998.

[55] R. Codina and M. Cervera. Block-iterative algorithms for nonlinear coupled problems. In M. Papadrakakis and G. Bugeda, editors, *Advanced Computational Methods in Structural Mechanics*, Theory and Engineering Applications of Computational Methods CIMNE, pages 114–134, 1996.

[56] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.

[57] M. Crouzeix and P.A. Raviart. Conforming and non-conforming finite elements for solving the stationary Stokes equations. *R.A.I.R.O. Anal. Numér.*, 7:33–76, 1973.

[58] C. Cuvelier, A. Segal, and A.A. van Steenhoven. *Introduction to the Finite Element Method*. D. Reidel Publishing Company, 1986.

[59] J.F. Dannenhoffer. *Grid adaptation for complex two-dimensional transonic flows*. PhD thesis, M.I.T., 1987.

[60] J.F. Dannenhoffer and J.R. Baron. Robust grid adaptation for complex transonic flows. *AIAA Paper 86-0495*, 1986.

[61] H. Deconinck, H. Paillère, R. Struijs, and P.L. Roe. Multidimensional upwind schemes based on fluctuation-splitting for systems of conservation laws. *Comput. Mech.*, 11(5–6):323–340, 1993.

[62] R. Deiterding. AMROC: A generic framework for blockstructured adaptive mesh refinement in object-oriented C++. `http://amroc.sourceforge.net/examples/euler/2d/html/ramp_n.htm`.

[63] R.S. Dembo, S.C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19(2):400–408, 1982.

[64] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems*. John Wiley and Sons, New York, 2003.

[65] J. Donea, L. Quartapelle, and V. Selmin. An analysis of time discretization in the finite element solution of hyperbolic problems. *Internat. J. Numer. Methods Engrg.*, 70(2):463–499, 1987.

[66] J. Donea, V. Selmin, and L. Quartapelle. Recent developments of the Taylor-Galerkin method for the numerical solution of hyperbolic problems. In K.W. Morton and M.J. Baines, editors, *Numerical methods for fluids dynamics III*, pages 171–185. Oxford University Press, 1988.

[67] J. Dongarra. Sparse matrix storage formats. In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, pages 315–337. SIAM, 2000.

[68] I.S. Duff, R.G. Grimes, and J.G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Software*, 15:1–14, 1989.

[69] P.R. Eiseman. Adaptive grid generation. *Comput. Methods Appl. Mech. Engrg.*, 64(1–3):321–376, 1986.

[70] S.C. Eisenstat and H.F. Walker. Globally convergent inexact Newton methods. *SIAM J. Optim.*, 4(2):393–422, 1994.

[71] S.C. Eisenstat and H.F. Walker. Choosing the forcing term in an inexact Newton method. *SIAM J. Sci. Comput.*, 17(1):16–32, 1996.

[72] E. Emmrich. Hyperbolische Erhaltungsgleichungen – Aspekte der analytischen und numerischen Lösung. Master's thesis, Technische Universität "Otto von Guericke", Magdeburg, 1993. (in German).

[73] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. Introduction to adaptive methods for differential equations. *Acta Numerica*, pages 105–158, 1995.

[74] M. Feistauer, J. Felcman, and M. Lukáčová-Medvidóvá. Combined finite element-finite volume solutions of compressible flow. *J. Comput. Appl. Math.*, 63(1–2):179–199, 1995.

[75] M. Feistauer, J. Felcman, and I. Straškraba. *Mathematical and Computational Methods for Compressible Flow*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2003.

[76] J.H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer Verlag, Berlin, 1996.

[77] J.H. Ferziger and M. Perić. Further discussion of numerical errors in CFD. *Internat. J. Numer. Methods Fluids*, 23:1–12, 1996.

[78] C.A.J. Fletcher. The group finite element formulation. *Comput. Methods Appl. Mech. Engrg.*, 37:225–243, 1983.

[79] C.A.J. Fletcher. *Computational Techniques for Fluid Dynamics, Vol. I*. Springer Verlag, 1988.

[80] L.A. Freitag. On combining Laplacian and optimization-based mesh smoothing techniques. In *Proceedings, Symp. Trends in Unstructured Mesh Generation, Amer. Soc. Mechanical Engineers*, pages 37–43, 1997.

[81] P.A. Gnoffo. A finite-volume, adaptive grid algorithm applied to planetary entry flowfields. *AIAA Journal*, 21(9):1249–1254, 1983.

[82] E. Godlewski and P.A. Raviart. *Numerical Approximation of Hyperbolic Systems of Conservation Laws*. Springer Verlag, 1995.

[83] S.K. Godunov. Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sbornic*, 47:271–306, 1959. (in Russian).

[84] J.B. Goodman and R.J. LeVeque. On the accuracy of stable schemes for 2D scalar conservation laws. *Math. Comp.*, 45:15–21, 1985.

[85] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, PA, 2000.

[86] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer Verlag, Berlin–Heidelberg, 1985.

[87] W. Hackbusch. *Elliptic Differential Equations. Theory and Numerical Treatment*. Series in Computational Mathematics. Springer Verlag, Berlin, 1992.

[88] W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*, volume 95 of *Applied Mathematical Sciences*. Springer Verlag, New-York, 1994.

[89] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Series in Computational Mathematics. Springer Verlag, 1993.

[90] P. Hansbo. Aspects of conservation in finite element flow computations. *Comput. Methods Appl. Mech. Engrg.*, 117:423–437, 1994.

[91] P. Hansbo. A free-Lagrange finite element method using space-time elements. *Comput. Methods Appl. Mech. Engrg.*, 188(1–3):347–361, 2000.

[92] A. Harten. High resolution schemes for hyperbolic conservation laws. *J. Comput. Phys.*, 49:357–393, 1983.

[93] A. Harten. On a class of high resolution total-variation-stable finite-difference schemes. *SIAM J. Numer. Anal.*, 21(1):1–23, 1984.

[94] R. Hartmann and P. Houston. Adaptive discontinuous Galerkin finite element methods for nonlinear hyperbolic conservation laws. *SIAM J. Sci. Comput.*, 24(3):979–1004, 2002.

[95] R. Hartmann and P. Houston. Symmetric interior penalty DG methods for the compressible Navier-Stokes equations I: Method formulation. *Int. J. Numer. Anal. Modeling*, 3(1):1–20, 2006.

[96] R. Hartmann and P. Houston. Symmetric interior penalty DG methods for the compressible Navier-Stokes equations II: Goal-oriented a posteriori error estimation. *Int. J. Numer. Anal. Modeling*, 3(2):141–162, 2006.

[97] A.D. Harvey, S. Acharya, and S.L. Lawrence. Prediction of complex three-dimensional flowfields using a solution-adaptive mesh algorithm. *AIAA Paper 91-3237*, 1991.

[98] A.D. Harvey, S. Acharya, and S.L. Lawrence. A solution-adaptive grid procedure for the three-dimensional parabolized Navier-Stokes equations. *AIAA Paper 91-0104*, 1991.

[99] A.D. Harvey, S. Acharya, S.L. Lawrence, and S. Cheung. A solution adaptive grid procedure for an upwind parabolized flow solver. *AIAA Paper 90-1576*, 1990.

[100] P.W. Hemker and B. Koren. Defect correction and nonlinear multigrid for steady Euler equations. Technical Report NM-R9007, Centre for Mathematics and Computer Science, Amsterdam (Netherlands), 1990.

[101] D. Hempel. Isotropic refinement and recoarsening in two dimensions. *Numer. Algorithms*, 13(1):33–43, 1996.

[102] D. Hempel. *Rekonstruktionsverfahren auf unstrukturierten Gittern zur numerischen Simulation von Erhaltungsprinzipien*. PhD thesis, Universität Hamburg, 1999. (in German).

[103] E. Hinton and J. Campbell. Local and global smoothing of discontinuous finite element functions using least squares method. *Internat. J. Numer. Methods Engrg.*, 8(3):461–480, 1974.

[104] C. Hirsch. *Numerical Computation of Internal and External Flows. Vol. I: Fundamentals of Numerical Discretization*. John Wiley and Sons, New York, 1990.

[105] C. Hirsch. *Numerical Computation of Internal and External Flows. Vol. II: Computational Methods for Inviscid and Viscous Flows*. John Wiley and Sons, New York, 1990.

[106] R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[107] C. Houston, P. Schwab and E. Süli. Discontinuous hp-finite element methods for advection-diffusion-reaction problems. *SIAM J. Numer. Anal.*, 39(6):2133–2163, 2001.

[108] P. Houston, E. Süli, J.A. Mackenzie, and G. Warnecke. A posteriori error analysis for numerical approximations of Friedrichs systems. *Numer. Math.*, 82(3):433–470, 1999.

[109] J. Hron, A. Ouazzi, and S. Turek. A computational comparison of two FEM solvers for nonlinear incompressible flow. In E. Bänsch, editor, *Challenges in Scientific Computing – CISC 2002*, volume 35 of *Lecture Notes in Computational Science and Engineering*. Springer Verlag, 2003.

[110] T.J.R. Hughes, M. Mallet, and A. Mizukami. A new finite element formulation for computational fluid dynamics: II. Beyond SUPG. *Comput. Methods Appl. Mech. Engrg.*, 54(3):341–355, 1986.

[111] C. Ilinca, X.D. Zhang, J.Y. Trépanier, and R. Camarero. A comparison of three error estimation techniques for finite-volume solutions of compressible flows. *Comput. Methods Appl. Engrg.*, 189(4):1277–1294, 2000.

[112] O.P. Jacquotte and J. Cabello. Three-dimensional grid generation method based on a variational principle. *Rech. Aérosp.*, 4:7–19, 1990.

[113] A. Jameson. Computational algorithms for aerodynamic analysis and design. *Appl. Numer. Math.*, 13(5):383–422, 1993.

[114] A. Jameson. Analysis and design of numerical schemes for gas dynamics 1. Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence. *Int. J. Comput. Fluid Dyn.*, 4:171–218, 1995.

[115] A. Jameson. Positive schemes and shock modelling for compressible flows. *Internat. J. Numer. Methods Fluids*, 20(8–9):743–776, 1995.

[116] A. Jameson, T.J. Baker, and N.P. Weatherill. Calculation of inviscid transonic flow over a complete aircraft. *AIAA Paper 86-0103*, 1986. 24th Aerospace Sciences Meeting, Reno, Nevada.

[117] V. John and P. Knobloch. On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations: Part I - A review. *Comput. Methods Appl. Mech. Engrg.*, 196(17–20):2197–2215, 2007.

[118] V. John and P. Knobloch. On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations: Part II - Analysis for P_1 and Q_1 finite elements. *Comput. Methods Appl. Mech. Engrg.*, 197(21–24):1997–2014, 2008.

[119] V. John and E. Schmeyer. Finite element methods for time-dependent convection-diffusion-reaction equations with small diffusion. Technical report, Universität des Saarlandes, 2008.

[120] C. Johnson. *Numerical solution of partial differential equations by the finite element method*. Studentlitteratur, 1987.

[121] C. Johnson. Adaptive finite element methods for conservation laws. In *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, Lecture Notes in Mathematics, pages 269–323. Springer Verlag, Berlin, 1998.

[122] T. Jongen and Y.P. Marx. Design of an unconditionally stable, positive scheme for the $k-\varepsilon$ and two-layer turbulence models. *Comput. & Fluids*, 26(5):469–487, 1997.

[123] D.A. Knoll and D.E. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *J. Comput. Phys.*, 193(3):357–397, 2004.

[124] S. Korotov and M. Křížek. Global and local refinement techniques yielding nonobtuse tetrahedral partitions. *Int. J. Comput. Math.*, 50:1105–1113, 2005.

[125] M. Köster. Institut für Angewandte Mathematik und Numerik, Technische Universität Dortmund, Germany. Private communication.

[126] L. Krivodonova and M. Berger. High-order accuracte implementation of solid wall boundary conditions in curved geometries. *J. Comput. Phys.*, 211(2):492–512, 2006.

[127] T. Kröger. *Multidimensional systems of hyperbolic conservation laws, numerical schemes, and characteristic theory – Connections, differences, and numerical comparison*. PhD thesis, RWTH-Aachen, 2004.

[128] D. Kröner. *Numerical Schemes for Conservation Laws*. Wiley, Teubner, 1997.

[129] D. Kröner, M. Küther, M. Ohlberger, and C. Rohde. A posteriori error estimates and adaptive methods for hyperbolic and convection dominated parabolic conservation laws. In M. Kirkilionis, editor, *Trends in nonlinear Analysis*. Springer Verlag, Berlin, 2003.

[130] D. Kröner and M. Ohlberger. A-posteriori error estimates for upwind finite volume schemes for nonlinear conservation laws in multidimensions. *Math. Comp.*, 69:25–39, 2000.

[131] D. Kuzmin. Institut für Angewandte Mathematik und Numerik, Technische Universität Dortmund, Germany. Private communication.

[132] D. Kuzmin. Algebraic Flux Correction: High-Resolution Schemes and Their Applications in CFD. In preparation.

[133] D. Kuzmin. Positive finite element schemes based on the flux-corrected transport procedure. In K.J. Bathe, editor, *Computational Fluid and Solid Mechanics*, pages 887–888. Elsevier, 2001.

[134] D. Kuzmin. On the design of general-purpose flux limiters for finite element schemes. I. Scalar convection. *J. Comput. Phys.*, 219(2):513–531, 2006.

[135] D. Kuzmin. Algebraic flux correction for finite element discretizations of coupled systems. In E. Oñate, M. Papadrakakis, and B. Schrefler, editors, *Computational Methods for Coupled Problems in Science and Engineering II*, CIMNE, pages 653–656, Barcelona, 2007.

[136] D. Kuzmin. Explicit and implicit FEM-FCT algorithms with flux linearization. *Submitted to: Internat. J. Numer. Methods Fluids*, 2008.

[137] D. Kuzmin and D. Kourounis. A semi-implicit FEM-FCT algorithm for efficient treatment of time-dependent problems. Technical Report 302, Institut für Angewandte Mathematik und Numerik, Technische Universität Dortmund, Germany, 2005.

[138] D. Kuzmin, R. Löhner, and S. Turek, editors. *Flux-Corrected Transport, Principles, Algorithms, and Applications*. Scientific Computation. Springer Verlag, 2005.

[139] D. Kuzmin and M. Möller. Algebraic flux correction I. Scalar conservation laws. In D. Kuzmin, R. Löhner, and S. Turek, editors, *Flux-Corrected Transport, Principles, Algorithms, and Applications*, pages 155–206. Springer Verlag, 2005.

[140] D. Kuzmin and M. Möller. Algebraic flux correction II. Compressible Euler equations. In D. Kuzmin, R. Löhner, and S. Turek, editors, *Flux-Corrected Transport, Principles, Algorithms, and Applications*, pages 207–250. Springer Verlag, 2005.

[141] D. Kuzmin, M. Möller, and S. Turek. Multidimensional FEM-FCT schemes for arbitrary time-stepping. *Internat. J. Numer. Methods Fluids*, 42(3):265–295, 2003.

[142] D. Kuzmin, M. Möller, and S. Turek. High-resolution FEM-FCT schemes for multidimensional conservation laws. *Comput. Methods Appl. Mech. Engrg.*, 193(45–47):4915–4946, 2004.

[143] D. Kuzmin, M. Möller, and S. Turek. Implicit flux-corrected transport algorithm for finite element simulation of the compressible Euler equations. In M. Křížek, P. Neittaanmäki, R. Glowinski, and S. Korotov, editors, *Conjugate Gradient Algorithms and Finite Element Methods*, Scientific Computation. Springer Verlag, 2004. Jyväskylä, Finland, June 11-12, 2002.

[144] D. Kuzmin and S. Turek. Flux correction tools for finite elements. *J. Comput. Phys.*, 175(2):525–558, 2002.

[145] D. Kuzmin and S. Turek. High-resolution FEM-TVD schemes based on a fully multidimensional flux limiter. *J. Comput. Phys.*, 198(1):131–158, 2004.

[146] P. Labbé and A. Garon. A robust implementation of Zienkiewicz and Zhu's local patch recovery method. *Comm. Numer. Methods Engrg.*, 11(5):427–434, 1995.

[147] M. Laforest, M.A. Christon, and T.E. Voth. A survey of error indicators and error estimators for hyperbolic problems. Available online at `http://www.mgi.polymtl.ca/marc.laforest/pages/sand_rep.ps`.

[148] B. Laney and D.A. Caughey. Extremum control II: Semi-discrete approximations to conservation laws. *AIAA Paper 91-0632*, 1991. 29th Aerospace Sciences Meeting, Reno, Nevada.

[149] C.B. Laney. *Computational Gasdynamics*. Cambridge University Press, Cambridge, 1998.

[150] P.D. Lax. *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*. SIAM, Philadelphia, 1990.

[151] P.D. Lax and B. Wendroff. Systems of conservation laws. *Comm. Pure Appl. Math.*, 13:217–237, 1960.

[152] E. Leitner and S. Selberherr. Mixed-element decomposition method for three-dimensional grid adaptation. *IEEE Trans. Circuits and Systems*, 17(7):561–572, 1998.

[153] P. Lesaint and P.A. Raviart. On a finite element method for solving the neutron transport equation. In *Mathematical Aspects of Finite Elements Methods in Partial Differential Equations*, pages 89–123. Academic Press, New York, 1974.

[154] R.J. LeVeque. *Numerical Methods for Conservation Laws*. Lectures in mathematics: ETH Zürich. Birkhäuser, 1992.

[155] R.J. LeVeque. CLAWPACK – a software package for solving multi-dimensional conservation laws. In *Proceedings of the 5th International Conference on Hyperbolic Problems*, `http://www.amath.washington.edu/~claw/`, 1994.

[156] R.J. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numer. Anal.*, 33:627–665, 1996.

[157] R.J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[158] H. Liepmann and A. Roshko. *Elements of Gasdynamics*. John Wiley and Sons, New York, 1957.

[159] W.K. Liu and A. Jameson. Multigrid Euler calculations for three-dimensional cascades. *AIAA Journal*, 31(10):1785–1791, 1993.

[160] R. Löhner. Center for Computational Fluid Dynamics, George Mason University, USA. Private communication.

[161] R. Löhner. An adaptive finite element scheme for transient problems in CFD. *Comput. Methods Appl. Mech. Engrg.*, 61:323–338, 1987.

[162] R. Löhner. Adaptive remeshing for transient problems. *Comput. Methods Appl. Mech. Engrg.*, 75:195–214, 1989.

[163] R. Löhner. *Applied CFD Techniques: An Introduction based on Finite Element Methods*. John Wiley and Sons, 2008.

[164] R. Löhner and J.D. Baum. Adaptive h-refinement on 3-D unstructured grids for transient problems. *Internat. J. Numer. Methods Fluids*, 14:1407–1419, 1992.

[165] R. Löhner and J.D. Baum. 30 years of FCT: Status and directions. In D. Kuzmin, R. Löhner, and S. Turek, editors, *Flux-Corrected Transport: Principles, Algorithms, and Applications*, pages 251–296. Springer Verlag, 2005.

[166] R. Löhner, H. Luo, and J.D. Baum. Selective edge removal for unstructured grids with Cartesian cores. *J. Comput. Phys.*, 206(1):208–226, 2005.

[167] R. Löhner, K. Morgan, J. Peraire, and M. Vahdati. Finite element flux-corrected transport (FEM-FCT) for the Euler and Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, 7(10):1093–1109, 1987.

[168] R. Löhner, K. Morgan, and O.C. Zienkiewicz. An adaptive finite element procedure for compressible high speed flows. *Comput. Methods Appl. Mech. Engrg.*, 51:441–465, 1984.

[169] R. Löhner, K. Morgan, and O.C. Zienkiewicz. Adaptive grid refinement for the Euler and compressible Navier-Stokes equations. In I. Babuška, editor, *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, pages 281–297. John Wiley and Sons, Chichester, 1986.

[170] R. Löhner and P. Parikh. Generation of three-dimensional unstructured grids by the advancing-front method. *Internat. J. Numer. Methods Fluids*, 8(10):1135–1149, 1988.

[171] F. Lörcher, G. Gassner, and C.D. Munz. The space-time expansion DG method. In C. Tropea, S. Jakirlic, H.J. Heinemann, R. Henke, and H. Hönlinger, editors, *New Results in Numerical and Experimental Fluid Mechanics VI*, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, pages 154–161. Springer Verlag, Berlin–Heidelberg, 2007.

[172] P.R.M. Lyra. *Unstructured Grid Adaptive Algorithms for Fluid Dynamics and Heat Conduction*. PhD thesis, University of Wales, Swansea, 1994.

[173] P.R.M. Lyra and K. Morgan. A review and comparative study of upwind biased schemes for compressible flow computations. III: Multidimensional extension on unstructured grids. *Arch. Comput. Methods Engrg.*, 9(3):207–256, 2002.

[174] P.R.M. Lyra, K. Morgan, J. Peraire, and J. Peiro. TVD algorithms for the solution of the compressible Euler equations on unstructured meshes. *Internat. J. Numer. Methods Fluids*, 19(9):827–847, 1994.

[175] J.A. Mackenzie, E. Süli, and G. Warnecke. A posteriori error estimates for the cell-vertex finite volume method. In W. Hackbusch and G. Wittum, editors, *Proceedings of the 9th GAMM Seminar on Adaptive Methods*, pages 221–235. Vieweg, Braunschweig-Wiesbaden, 1994.

[176] R.C. Martineau and R.A. Berry. Characteristic boundary conditions for the two-step Taylor-Galerkin FEM. *Comput. Methods Appl. Mech. Engrg.*, 195(7–8):742–762, 2006.

[177] C. Mavriplis. Adaptive mesh strategies for the spectral element method. *Comput. Methods Appl. Mech. Engrg.*, 116:77–86, 1994.

[178] D.J. Mavriplis. Adaptive mesh generation for viscous flows using Delaunay triangulation. *J. Comput. Phys.*, 90(2):271–291, 1990.

[179] D.J. Mavriplis. *Computational Fluid Dynamics Techniques*, chapter Euler and Navier-Stokes Solutions Using Unstructured Meshes, pages 779–798. Gordon and Breach Publishing Group, 1995.

[180] D.J. Mavriplis. Adaptive meshing techniques for viscous flow calculations on mixed-element unstructured meshes. *AIAA Paper 97-0857*, 1997.

[181] J.A Meijerink and H.A. van der Vorst. Iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comput.*, 31:148–162, 1977.

[182] K. Miller and R.N. Miller. Moving finite elements. *SIAM J. Numer. Anal.*, 18:1019–1032, 1981.

[183] M. Möller. Efficient solution techniques for implicit finite element schemes with flux limiters. *Internat. J. Numer. Methods Fluids*, 55(7):611–635, 2007.

[184] M. Möller. On an efficient solution strategy of Newton type for implicit finite element schemes based on algebraic flux correction. *Internat. J. Numer. Methods Fluids*, 56(8):1085–1091, 2008.

[185] M. Möller and D. Kuzmin. Adaptive mesh refinement for high-resolution finite element schemes. *Internat. J. Numer. Methods Fluids*, 52(5):545–569, 2006.

[186] M. Möller and D. Kuzmin. On the use of slope limiters for the design of recovery based error indicators. In A. Bermudez de Castro, D. Gomez, P. Quintela, and P. Salgado, editors, *Proceedings of ENUMATH 2005, the 6th European Conference on Numerical Mathematics and Advanced Applications Santiago de Compostela, Spain, July 2005*, Numerical Mathematics and Advanced Applications, pages 233–240, 2006.

[187] M. Möller, D. Kuzmin, and D. Kourounis. Implicit FEM-FCT algorithms and discrete Newton methods for transient convection problems. *Internat. J. Numer. Methods Fluids*, 57(6):761–792, 2008.

[188] J.J. Moré and D.J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Software*, 20:286–307, 1984.

[189] K. Morgan and J. Peraire. Unstructured grid finite element methods for fluid mechanics. *Rep. Progr. Phys.*, 61(6):569–638, 1998.

[190] W. Mulder and B.V. Leer. Experiments with implicit upwind methods for the Euler equations. *Internat. J. Numer. Methods Engrg.*, 59:232–246, 1985.

[191] E.J. Nielsen, W.K. Anderson, R.W. Walters, and D.E. Keyes. Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code. *AIAA Paper 95-0221*, 1995.

[192] W.L. Oberkampf and F.G. Blottner. Issues in computational fluid dynamics code verification and validation. *AIAA Journal*, 36(5):687–695, 1998.

[193] J.T. Oden and H.J. Brauchli. On the calculation of consistent stress distributions in finite element approximations. *Internat. J. Numer. Methods Engrg.*, 3:317–325, 1971.

[194] J.T. Oden and S. Prudhomme. Goal-oriented error estimation and adaptivity for the finite element methods. *Comput. Math. Appl.*, 41:735–756, 2000.

[195] J.T. Oden and J.N. Reddy. Note on an approximate method for computing consistent conjugate stresses in elastic finite elements. *Internat. J. Numer. Methods Engrg.*, 6:55–61, 1973.

[196] J.T. Oden, T. Strouboulis, and P. Devloo. Adaptive finite element methods for the analysis of inviscid compressible flow: Part I. Fast refinement/unrefinement and moving mesh methods for unstructured meshes. *Comput. Methods Appl. Mech. Engrg.*, 59:327–362, 1986.

[197] M. Ohlberger. *A posteriori error estimates and adaptive methods for convection dominated transport processes*. PhD thesis, Universität Freiburg, 2001.

[198] O. Oleĭnik. Discontinuous solutions of non-linear differential equations. *Usp. Mat. Nauk (N.S.)*, 12:3–73, 1957. (in Russian), translated in *Am. Math. Soc. Transl. (Ser. 2)* **26**, 95–172.

[199] A. Ouazzi and S. Turek. Unified edge-oriented stabilization of nonconforming FEM for incompressible flow problems: Numerical investigations. *J. Numer. Math.*, 15(4):299–322, 2007.

[200] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill, 1980.

[201] R. P. Pawlowski, J.N. Shadid, J.P Simonis, and H.F. Walker. Globalization techniques for Newton-Krylov methods and applications to the fully-coupled solution of the Navier-Stokes equations. Report SAND2004-1777, Sandia National Laboratories, 2004.

[202] J. Peraire, K. Morgan, J. Peiro, and O.C. Zienkiewicz. An adaptive finite element method for high speed flows. *AIAA Paper 87-0558*, 1987.

[203] J. Peraire, M. Vahdati, K. Morgan, and O.C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *J. Comput. Phys.*, 72:449–466, 1987.

[204] J. Peraire, M. Vahdati, J. Peiro, and K. Morgan. The construction and behaviour of some unstructured grid algorithms for compressible flows. In *Proc. ICFD Conf. on Numerical Methods for Fluid Dynamics*, volume VI, pages 221–239. Oxford University Press, 1993.

[205] J. Peraire, J. White, A. Patera, and B.C. Khoo. Numerical methods for partial differential equations (sma 5212). Available at: `http://ocw.mit.edu/OcwWeb/Aeronautics-and-Astronautics/`, 2003.

[206] M. Pernice and H.F. Walker. NITSOL: A Newton iterative solver for nonlinear systems. *SIAM J. Sci. Comput.*, 19:302–318, 1998.

[207] A. van der Ploeg. *Preconditioning for sparse matrices with applications*. PhD thesis, University of Groningen, 1994.

[208] A. van der Ploeg, R. Keppens, and G. Tóth. Block incomplete LU-preconditioners for implicit solution of advection dominated problems. In B. Hertzberger and P.M.A. Sloot, editors, *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking*, volume 1225 of *Lecture Notes In Computer Science*, pages 421–430, 1997.

[209] E.J. Probert, O. Hassan, and K. Morgan. An adaptive finite element method for transient compressible flows with moving boundaries. *Internat. J. Numer. Methods Engrg.*, 32:751–765, 1991.

[210] S. Prudhomme. Adaptive finite element simulation of a supersonic underexpanded jet. Master's thesis, University of Virginia, 1992.

[211] S. Prudhomme and J.T. Oden. Computable error estimators. In T.J. Barth and H. Deconinck, editors, *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*, Lecture Notes in Computational Science and Engineering. Springer Verlag, Berlin, 2003.

[212] R. Rannacher and S. Turek. A simple nonconforming quadrilateral Stokes element. *Numer. Methods Partial Differential Equations*, 8:97–111, 1992.

[213] M. C. Rivara. Using longest-side bisection techniques for the automatic refinement of Delaunay triangulations. *Internat. J. Numer. Methods Engrg.*, 40:581–597, 1997.

[214] M.C. Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *Internat. J. Numer. Methods Engrg.*, 20:745–756, 1984.

[215] M.C. Rivara. Design and data structure of fully adaptive, multigrid, finite-element software. *ACM Trans. Math. Software*, 10(3):242–264, 1984.

[216] M.C. Rivara. New mathematical tools and techniques for the refinement and/or improvement of unstructured triangulations. In *Proceedings, 5th International Meshing Roundtable, Sandia National Laboratories*, pages 77–86, Pittsburgh, 1996.

[217] M.C. Rivara. New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations. *Internat. J. Numer. Methods Engrg.*, 40:3313–3324, 1997.

[218] M.C. Rivara and G. Iribarren. The 4-triangles longest-side partition of triangles and linear refinement algorithms. *Math. Comp.*, 65(216):1485–1502, 1996.

[219] M.C. Rivara and C. Levin. A 3D refinement algorithm for adaptive and multigrid techniques. *Comm. Appl. Numer. Methods*, pages 281–290, 1992.

[220] M.C. Rivara and M. Venere. Cost analysis of the longest-side (triangle bisection) refinement algorithm for triangulations. *Engrg. Comput.*, 12:224–234, 1996.

[221] A. Rizzi and J. Vos. Towards establishing credibility in computational fluid dynamics simulatiuons. *AIAA Journal*, 36:668–675, 1998.

[222] P.J. Roache. A method for uniform reporting of grid refinement studies. *J. Fluids Engrg.*, 116:405–413, 1994.

[223] P.J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, New Mexico, 1998.

[224] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, 43(2):357–372, 1981.

[225] A. Rohde. Eigenvalues and eigenvectors of the Euler equations in general geometries. *AIAA Paper 2001-2609*, 2001.

[226] C.J. Roy. Grid convergence error analysis for mixed-order numerical schemes. *AIAA Journal*, 41(4):595–604, 2003.

[227] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[228] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[229] R. Sanders. On convergence of monotone finite difference schemes with variable spatial differencing. *Math. Comp.*, 40:91–106, 1983.

[230] E. Schall, D. Leservoisier, A. Dervieux, and B. Koobus. Mesh adaptation as a tool for certified computational aerodynamics. *Internat. J. Numer. Methods Fluids*, 45:179–196, 2004.

[231] V. Selmin. Finite element solution of hyperbolic equations. II. Two-dimensional case. Technical Report RR-708, INRIA, 1987.

[232] V. Selmin. Finite element solutions of hyperbolic equations. I. One-dimensional case. Technical Report RR-655, INRIA, 1987.

[233] V. Selmin. The node-centred finite volume appoach: Bridge between finite differences and finite elements. *Comput. Methods Appl. Mech. Engrg.*, 102:107–138, 1993.

[234] A.S. Sens and G.D. Mortchelewicz. Implicit schemes for unsteady Euler equations on triangular meshes. *Internat. J. Numer. Methods Fluids*, 18:647–668, 1994.

[235] J.N. Shadid, R.S. Tuminaro, and H.F. Walker. An inexact Newton method for fully coupled solution of the Navier-Stokes equations with heat and mass transport. *J. Comput. Phys.*, 137:155–185, 1997.

[236] R.A. Shapiro. *Adaptive Finite Element Solution Algorithm for the Euler Equations*, volume 32 of *Notes on Numerical Fluid Mechanics*. Vieweg, Braunschweig-Wiesbaden, 1991.

[237] M.S. Shepard and M.K Georges. Automatic three-dimensional mesh generation by the finite octree technique. *Internat. J. Numer. Methods Engrg.*, 1991.

[238] J.P. Simonis. A numerical study of globalizations of Newtons-GMRES methods. Master's thesis, Worcester Polytechnic Institute, 2003.

[239] S.P. Spekreijse. *Multigrid Solution of the Steady Euler Equations*. PhD thesis, Center for Mathematics and Computer Science, Amsterdam, 1988.

[240] J.L. Steger and R.F. Warming. Flux vector splitting of the inviscid gasdynamic equations with applications to finite difference methods. *J. Comput. Phys.*, 40:263–293, 1981.

[241] G. Strang and G. Fix. *An Analysis of the Finite Element Method*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.

[242] E. Süli. A-posteriori error analysis and adaptivity for finite element approximations of hyperbolic problems. In M. Ohlberger, D. Kröner, and C. Rohde, editors, *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, volume 5 of *Lecture Notes in Computational Science and Engineering*, pages 123–194. Springer Verlag, 1998.

[243] E. Süli and P. Houston. Adaptive finite element approximation of hyperbolic problems. In T.J. Barth and H. Deconinck, editors, *Error estimation and adaptive discretization methods in computational fluid dynamics*. Springer Verlag, 2003.

[244] E. Süli, P. Houston, and B. Senior. hp-discontinuous Galerkin finite element methods for nonlinear hyperbolic problems. *Internat. J. Numer. Methods Fluids*, 40(1–2):153–169, 2001.

[245] P.K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM J. Numer. Anal.*, 21:995–1011, 1984.

[246] E. Tadmor. Convenient total variation diminishing conditions for nonlinear difference schemes. *SIAM J. Numer. Anal.*, 25:1002–1014, 1988.

[247] H. Tang and T. Tang. Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. *SIAM J. Numer. Anal.*, 41(2):487–515, 2003.

[248] H.Z. Tang. A moving mesh method for the Euler flow calculations using a directional monitor function. *Comput. Phys. Comm.*, 1(4):656–676, 2006.

[249] J.F. Thompson. A survey of dynamically-adaptive grids in numerical solution to partial differential equations. *AIAA Paper 84-1606*, 1984.

[250] J.F. Thompson. A reflection on grid generation in the 90s: Trends, needs, and influences. In *5 th International Conference on Numerical Grid Generation in Computational Field Simulations, Mississippi State University*, pages 1029–1110, 1996.

[251] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics, A Practical Introduction*. Springer Verlag, 1999.

[252] F.T. Tracy, B.P. Donnell, S.E. Howington, and J.L. Hensley. Application of the pseudo-transient technique to a real-world unsaturated flow groundwater problem. In *Proc. 5th Int. Conf. on Computational Science, Part I*, volume 3514 of *Lecture Notes in Computer Science*, pages 66–73, Atlanta, GA, USA, 2005. Springer Verlag.

[253] R.S. Tuminaro, H.F. Walker, and J.N. Shadid. On backtracking failure in Newton-GMRES methods with a demonstration for the Navier-Stokes equations. *J. Comput. Phys.*, 180:549–558, 2002.

[254] S. Turek. On ordering strategies in a multigrid algorithm. In *Proc. 8th GAMM-Seminar*, volume 41 of *Notes on Numerical Fluid Mechanics*, Kiel, 1992. Vieweg.

[255] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*. Number 6 in Lecture Notes in Computational Science and Engineering. Springer Verlag, 1999.

[256] A.M.P. Valli, G.F. Carey, and A.L.G.A. Coutinho. Finite element simulation and control of nonlinear flow and reactive transport. In *Proc. 10th Int. Conf. Finite Element in Fluids*, Tucson, Arisona, 1998.

[257] R.S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1962.

[258] D.A. Venditti and D.L. Darmofal. A grid adaptive methodology for functional outputs of compressible flow simulations. *AIAA Paper 2001-2659*, 2001.

[259] R. Verfürth. *A Review of a posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Wiley-Teubner, 1996.

[260] D.F. Watson. Computing the n-dimensional Delaunay tessellation with applicaion to Voronoi polytopes. *Comput. J.*, 24(2):167–172, 1981.

[261] N.P. Weatherill. A strategy for the use of hybrid structured-unstructured meshes in CFD. In K.W. Morton and M.J. Baines, editors, *Numerical Methods for Fluid Dynamics*. Oxford University Press, 1998.

[262] P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer Verlag, 2001.

[263] G.B. Whitam. *Linear and Non-linear Waves*. John Wiley and Sons, 1974.

[264] P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shock. *J. Comput. Phys.*, 54:115–173, 1983.

[265] H.C. Yee. Numerical approximations of boundary conditions with applications to inviscid gas dynamics. *NASA report TM-81265*, 1981.

[266] H.C. Yee. Construction of explicit and implicit symmetric TVD schemes and their applications. *Internat. J. Numer. Methods Engrg.*, 68(1):151–179, 1987.

[267] H.C. Yee, R.F. Warming, and A. Harten. Implicit total variation diminishing (TVD) schemes for steady-state calculations. *Internat. J. Numer. Methods Engrg.*, 57(3):327–360, 1985.

[268] S.T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, 31:335–362, 1979.

[269] S.T. Zalesak. The design of flux-corrected transport (FCT) algorithms for structured grids. In D. Kuzmin, R. Löhner, and S. Turek, editors, *Flux-Corrected Transport: Principles, Algorithms, and Applications*, pages 29–78. Springer Verlag, 2005.

[270] X.D. Zhang, D. Pelletier, J.-Y. Trépanier, and R. Camarero. Numerical assessment of error estimators for Euler equations. *AIAA Journal*, 39(9):1706–1715, 2001.

[271] Z. Zhang and A. Naga. A posteriori error estimates based on polynomial preserving recovery. *SIAM J. Numer. Anal.*, 42(4):1780–1800, 2004.

[272] Z. Zhang and A. Naga. A new finite element gradient recovery method: Superconvergence property. *SIAM J. Sci. Comput.*, 26(4):1192–1213, 2005.

[273] O.C. Zienkiewicz, J.P. De, S.R. Gago, and D.W. Kelly. The hierarchical concept in finite element analysis. *Comput. & Structures*, 16:53–65, 1983.

[274] O.C. Zienkiewicz and J.Z. Zhu. A simple error estimator and adaptive procedure for practical engineering analysis. *Internat. J. Numer. Methods Engrg.*, 24(2):337–357, 1987.

[275] O.C. Zienkiewicz and J.Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery techniques. *Internat. J. Numer. Methods Engrg.*, 33(7):1331–1364, 1992.

[276] O.C. Zienkiewicz and J.Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity. *Internat. J. Numer. Methods Engrg.*, 33(7):1365–1382, 1992.