Lehrstuhl für Numerische Mathematik
und Wissenschaftliches Rechnen
Prof. Dr. Folkmar Bornemann

Adaptive Methods for the Numerical Simulation of Transport Processes

Martin Andreas Käser

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. F. Bornemann
Prüfer der Dissertation:		
	1.	Priv.-Doz. Dr. A. Iske
	2.	Univ.-Prof. Dr. H. Igel Ludwig-Maximilians-Universität München
	3.	Prof. E. F. Toro, Ph.D. Univ. Trient / Italien

Die Dissertation wurde am 09.10.2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 16.12.2004 angenommen.

Acknowledgments

First of all I want to thank my supervisor Armin Iske, who always had the time to discuss any question and who taught me how to approach numerical problems effectively. I enjoyed working together with him not only because of his experience and encouraging support, but also because of his patience and good sense of humor. Many thanks to Jörn Behrens, whose knowledge about the semi-Lagrangian approach and the finite element method resulted in very important and helpful hints.
I am also deeply grateful to all my colleagues of the department for the comfortable and productive working atmosphere and the many discussions that helped to look at problems from a different view.
Special thanks to Heiner Igel, who enabled me to keep in close contact with geophysics through his various seminars and workshops. His interest in new numerical methods, especially for numerical seismology, always encouraged me to keep in mind the applicability and efficiency of numerical schemes with respect to elastic wave propagation.
In particular, I would like to thank Tito Toro, who inspired me to work on ADER methods. His invitation to the workshops and conferences in Barcelona and Cambridge offered me the great opportunity to get a better understanding of a variety of numerical methods for hyperbolic conservation laws. Furthermore, he gave me the chance to present and discuss my own results in front of an exclusive audience in Cambridge. Thanks, to Randy LeVeque, Chi-Wang Shu, and Knut-Andreas Lie for their helpful and encouraging comments during this workshop.
My collaborators within the European project NetAGES (Network for Automated Geometry Extraction from Seismic), Stine Kjersti Richardsen, Tobias Werther, Alexander Boukhgueim, and Erik Monsen should also be acknowledged for broadening my view on the different topics of their work related to the NetAGES project. Special thanks to the project coordinator Trygve Randen, Magne Lygren, and Brice Valles, who supported my work very actively and made my visits at Schlumberger Stavanger Research very productive and successful, in particular while using their commercial software to compare numerical results.

Ich bedanke mich auch herzlich bei meiner Familie und meiner Freundin, die mich bei so viele Dingen stets tatkräftig unterstützt haben und deren Rückhalt für mich immer von großer Bedeutung war.

München, Oktober 2003
Martin Käser

Zusammenfassung

Zunächst wird ein neues, gitterfreies Verfahren zur numerischen Lösung nichtlinearer Transportgleichungen entwickelt. Das Verfahren kombiniert eine adaptive semi-Lagrange Methode mit einer lokalen Interpolation unter Verwendung von radialen Basisfunktionen. Die vorgestellte Partikelmethode verwendet eine Menge beliebig verteilter Punkte, wodurch kein Gitter zur Diskretisierung benötigt wird. Im weiteren wird dieses semi-Lagrange Verfahren erweitert, so daß für lineare Advektionsprobleme exakte Massenerhaltung erlangt wird. Dazu werden adaptive Voronoi-Diagramme verwendet, um die Flexibilität einer Partikelmethode beizubehalten. Abschließend wird eine neuartige, gitterbasierte Methode (ADER-Methode) auf adaptive, unstrukturierte Triangulierungen erweitert, die exakte Massenerhaltung auch für nichtlineare Transportprobleme liefert. Insbesondere unterliegt die Konvergenzordnung solcher ADER-Methoden keiner Beschränkung und wird bis zu Verfahren vierter Ordnung untersucht.
Für alle behandelten Methoden werden sowohl die Fehlerschätzer und die Adaptionsroutinen zur Verfeinerung oder Vergröberung der Partikel bzw. des Gitters ausführlich diskutiert, als auch die Leistungsfähigkeit dieser adaptiven Verfahren anhand numerischer Beispiele unterstrichen. Insbesondere wird auf ein Anwendungsbeispiel aus dem Bereich der Reservoir Simulation eingegangen, das vor allem in der Erdölindustrie zur Modellierung von Flüssigkeitstransport in Lagerstätten von großem Interesse ist.

Abstract

At first a new meshfree advection scheme for numerically solving nonlinear transport equations is developed. The scheme, being a combination of an adaptive semi-Lagrangian method and local radial basis function interpolation, is essentially a method of backward characteristics. The proposed particle method works with an unstructured set of nodes, such that no mesh is necessary for the discretization. Furthermore, this semi-Lagrangian method is modified in order to achieve exact mass conservation for linear advection problems. Voronoi diagrams are used to retain the flexibility of the particle method. Finally, a new conservative scheme (ADER-scheme) is extended to adaptive unstructured triangulations in order to treat nonlinear transport problems. In particular, ADER schemes can be constructed up to an arbitrary high order of convergence and are investigated in detail up to order four.

For all methods the error estimation and the strategy of the adaption rules for the refinement or coarsening of particles or meshes are discussed in detail. The good performance of the resulting methods is confirmed by numerical examples. In particular, a test case from the oil industry is addressed, which plays an important role in the modelling of fluid flow in petroleum reservoirs.

Contents

Introduction

Hyperbolic conservation laws constitute the governing equations for many important and powerful mathematical models of a wide variety of physical phenomena. In particular, they are used to describe advective transport processes in gas dynamics, geophysical fluid dynamics, meteorology, astrophysics, multi-phase flow in porous media, convection dominated flow, elastodynamics and many others.
Historically, many of the fundamental numerical techniques were first developed for the special case of the Euler equations in gas dynamics, where the nonlinearity of the conservation law causes discontinuities in the solutions, the so-called *shocks*. The treatment of these discontinuities poses many of the computational challenges, that motivated the development of various numerical methods. To this end, the study of model equations, such as the popular Burgers equation, has played an important role in the development of such numerical methods. Usually, the design of new numerical schemes requires a close interplay between numerical analysis, physical modelling, numerical simulation and visualisation.
One century ago, in 1904, Vilhelm Bjerknes firstly suggested the possibility of deterministic weather prediction in [13], before Lewis Richardson in [69] actually attempted to produce such a forecast by manually integrating a finite difference approximation to the governing partial differential equations describing atmospheric motion. Unfortunately, his results were disappointing and the amount of human labour required to do the calculations were so immense, that deterministic weather prediction had to await the introduction of computers. However, his work marked the beginning of Computational Fluid Dynamics (CFD).
In 1950, researchers under the direction of Jule Charney and John von Neumann managed to produce the first computer-generated weather forecast and their surprisingly good results, reported in [18], led to the rapid growth in a new discipline, *numerical weather prediction*. Since that, computers have developed with tremendous speed, and the use of numerical models has subsequently expanded into almost all areas of current research and consequently often led to fruitful interdisciplinary collaborations between numerical analysts and scientists of various application fields.

During the past few years, the interest has grown especially to construct efficient, robust and high order accurate numerical schemes in order to treat conservation laws.
The problem considered is the homogeneous, scalar, first order time-dependent partial differential equation

$$\frac{\partial}{\partial t}u(t,x) + \nabla \cdot f\Big(u(t,x)\Big) = 0\,,$$

where $u : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$ is the unknown function depending on time t and the two space variables $x = (x_1, x_2)$, and $f(u) = (f_1(u), f_2(u))$ denotes the *flux vector*. Many physically relevant problems give rise to *nonlinear* conservation laws, in which the flux $f(u)$ is a nonlinear function of the conserved quantity u. A fundamental feature of these nonlinear conservation laws is that their solution can develop discontinuities, even from smooth initial data, which have to be dealt with mathematically and computationally.

This work proposes adaptive numerical schemes in order to solve both linear and nonlinear conservation laws. At first we introduce a new *meshless* semi-Lagrangian advection scheme, which is based on a set of particles in order to discretise the problem in the computational domain $\Omega \subset \mathbb{R}^2$. In the following, we construct a *conservative* semi-Lagrangian finite volume method that relies on Voronoi tessellations of Ω and the concept of backward characteristics. Finally, we propose an adaptive ADER scheme using **A**rbitrary high order **DER**ivatives of the approximated unknown function u in order to design finite volume methods of arbitrary high order of accuracy.
In this application-oriented work, we test the feasibility and performance of the resulting schemes by solving scalar conservation laws describing linear and nonlinear transport processes. Thereby, the main focus for all introduced methods lies on their *adaptivity*. In order to optimize the computing time as well as the use of storage, the design of self-adaptive numerical schemes based on locally refined discretisations is essential. Adaptivity, however, requires reliable *error estimators*.
For elliptic and parabolic problems there is already a well-known theory about *a posteriori error estimators*, e.g. [20, 27], and also for linear hyperbolic systems results for error estimators are available in [33, 80]. Unfortunately, there are only very few theoretical results for a posteriori error estimates for nonlinear conservation laws, e.g. [61]. Therefore, we use a new, more heuristic error indicator in order to identify regions, where shocks or large gradients occur. The spatial discretisation in these regions is then locally refined by new and robust adaption rules in order to improve the local resolution and enhance the approximation quality. The proposed error indicator is based on the idea of detecting discontinuities in scattered data [35] and relies on local interpolations using *radial basis functions*.

Equipped with this modern and effective error indicator, we assign a *significance value* to each node or cell (depending on the numerical method used) of a discretisation, which flags them for refinement or coarsening. The according adaption rules for the local refinement or coarsening of the discretisation, which are fundametal for the substantial progress of adaptive schemes, are simple, robust and efficient, as confirmed by various model applications in this work.
Our major interest lies in the solution of a nonlinear conservation law introduced by Buckley and Leverett in [16]. This so-called *Buckley-Leverett equation* describes the two-phase flow in a porous medium. Generally, the two different immiscible liquids move through the porous medium driven by a pressure gradient. The solution of this model problem typically develops a shock followed by a rarefaction wave, which is particularly challenging for high order accurate numerical methods.
Solving the Buckley-Leverett equation accurately and especially predict the propagation of the moving shock with high resolution is of major importance in oil reservoir simulation. In the area of petroleum reservoir simulation and engineering *waterflooding* is a well established technique of enhanced oil recovery, where the displacement of oil in the pores of a reservoir rock by injected water can be modelled by the Buckley-Leverett equation. As demonstrated by our numerical results our modern adaptive strategy, especially when combined with the new high order accurate ADER schemes, constitutes a very progressive approach compared to industrial standards. Furthermore, the past few years have shown, that results of large, detailed reservoir simulations have an increasing impact on reservoir management decisions.

The present work is subdivided into four major parts, which concentrate on the application-oriented discussion of the three new numerical approaches and the application to the real-world problem of the Buckley-Leverett equation.
Therefore, the work is arranged as follows:

In **Chapter 1** we introduce a new adaptive *meshfree* advection scheme, as proposed for numerically solving *linear* transport equations in previous work [9, 11], is extended to *nonlinear* transport equations. Meshless methods represent a very recent technology for solving partial differential equations and became very popular in structural mechanics as an alternative to mesh-based techniques as Finite Elements or Finite Volumes. Therefore, a general overview of state-of-the-art approaches is given primarily. Our proposed scheme, being a combination of an adaptive semi-Lagrangian method and local radial basis function interpolation, is essentially a *particle method* or a *method of backward characteristics*. Our aim is to transfer the advantages of a meshless method from mechanics to fluid flow problems. The adaptivity of the meshfree advection scheme relies on customized rules for the refinement and coarsening of scattered nodes. In order to be able to model shock propagation, an artificial viscosity term is added to the scheme. Moreover, the local interpolation method and the node adaption rules in [9, 11] are modified accordingly. The performance of the resulting method is finally shown in numerical examples by using two specific nonlinear model problems: *Burgers equation* and the *Buckley-Leverett equation.*

In **Chapter 2** we design a *conservative* semi-Lagrangian advection scheme overcoming the problems of our pure particle method in Chapter 1, which is not conservative. Considerable effort has been made recently in order to construct *conservative* semi-Lagrangian methods [47, 66, 67], in particular Phillips and Williams [64] developed an attractive conservative semi-Lagrangian Finite Volume scheme. Their formulation of the discrete problem is based on satisfying a physical conservation constraint in a way that conservation is satisfied by construction. However, the main weekness of their approach is, that they have to use a fixed Cartesian mesh. In contrast, our new semi-Lagrangian scheme works with finite volumes on an unstructured mesh, which is given by a Voronoi diagram. Moreover, in our modern approach the mesh is subject to adaptive modifications during the simulation, which serves to effectively combine good approximation quality with small computational costs. The required adaption rules for the refinement and the coarsening of the mesh rely on our introduced customized error indicator and the efficient adaption rules. Additionally, we develop a technology for the effective implementation of boundary conditions. Finally, numerical results confirm the good performance of the proposed conservative and adaptive advection scheme, especially when long simulation times are desired.

In **Chapter 3** the further development of modern ADER schemes is presented. The *ADER* approach introduced by Toro, Millington, and Nejad in [84], and advanced by Titarev in [81, 83] represents a finite volume scheme of **A**rbitrary high order using high order **DER**ivatives of piecewise polynomial reconstructions. We extend their recent approach in order to solve linear as well as nonlinear scalar conservation laws on adaptive unstructured triangulations. Firstly, a general overview of the development of the *weighted essentially non-oscillatory* (WENO) reconstruction technique and the development high order Finite Volume schemes is given. ADER schemes can be interpreted as high order generalizations of the classical Godunov scheme, which lead to an arbitrary order of accuracy in both, space and time. The proposed scheme is conservative and combines high order WENO reconstruction techniques with a high order flux evaluation method to update cell average values. To this end, we solve generalized Riemann problems across cell interfaces by transforming them into a series of conventional Riemann problems. Moreover, the underlying mesh can be unstructured and is adaptively modified during the simulation to effectively combine high order accuracy with high resolution at small computational costs. The required adaption rules for the refinement and coarsening of the triangular mesh rely again on the error indicator explained in the previous chapters. The implementation of inflow and outflow boundaries is addressed as well as the use of periodic boundaries. Finally, numerical experiments confirm the expected orders of accuracy and the good performance of the proposed scheme for linear and nonlinear problems.

In **Chapter 4** the newly developed adaptive ADER schemes are applied to a well-established standard test case [2] in the area of reservoir simulation, the five-spot problem. Here, one oil production well in the center of the computational domain is surrounded by four water injection wells. Due to the pressure gradient between the wells the water is forced to move towards the production well displacing part of the oil in the porous reservoir rock. The solution of the five-spot problem is computed with adaptive ADER schemes of different order and the results are compared to reference solutions, which are obtained by two of the standard reservoir simulation software-packages in the oil industry. The numerical results and the comparison with the reference solutions demonstrate that the combination of high order ADER schemes and adaptive mesh refinement can provide high accuracy and high resolution while dramatically reducing the required number of mesh cells.

As the numerical methods discussed in this work differ in their performances and demonstrate their individual advantages and disadvantages, each Chapter is summarized in a separate conclusion. Considerations concerning open problems and ideas for future research are finally addressed in the **Outlook**.

In order to keep this work widely self-contained we explain the fundamental ideas of O'Rourke's intersection algorithm for convex polygons [63] in **Appendix A**. Furthermore, a brief overview of the isotropic mesh refinement strategy of Hempel [39] is given in **Appendix B** together with preliminary results, that have been obtained by combining his approach with ADER schemes. Finally, **Appendix C** provides a list of Gaussian integration rules on triangles that are exact for polynomials of degree ≤ 7.

Chapter 1

Adaptive Meshfree Advection

In this chapter a new, adaptive *meshfree* advection scheme, as proposed for numerically solving *linear* transport equations in previous work [9, 11], is extended to *nonlinear* transport equations[1]. The scheme, being a combination of an adaptive semi-Lagrangian method and local radial basis function interpolation, is essentially a *method of backward characteristics*. The adaptivity of the meshfree advection scheme relies on customized rules for the refinement and coarsening of scattered nodes. In order to be able to model shock propagation, an artificial viscosity term is added to the scheme. Moreover, the local interpolation method and the node adaption rules in [9, 11] are modified accordingly. The performance of the resulting method is finally shown in numerical examples by using two specific nonlinear model problems: *Burgers equation* and the *Buckley-Leverett equation*.

1.1 General Overview

Currently, Finite Element Methods (FEM) [15, 74, 90] and Finite Volume Methods (FVM) [40, 85] are well-established numerical techniques to solve problems in computational fluid dynamics. Their main advantage is their capability of handling complicated domain geometries and their local approximation character. However, both methods are based on the decomposition of the computational domain into non-overlapping subdomains, called *elements* or *cells*, and therefore rely on a mesh. It is widely acknowledged that especially in 3-D or higher dimensions, mesh generation is still a big challenge and often is the bottleneck in large scale industrial applications. In fact, it can be more costly than the numerical solution of the discretized problem itself. Although complicated domains in 3-D can already be discretized automatically with tetrahedral elements, local mesh refinement and coarsening often require rather sophisticated strategies mainly in order to avoid hanging

[1]This is joined work with J. Behrens and A. Iske and is already published in [10].

nodes or other mesh degeneracies, which would have to be treated separately. During the last decade considerable effort was put into the investigation of so-called *meshless* or *grid-free* methods. These methods require a set of computational *nodes* distributed throughout the domain, but they do not necessarily require a specific connectivity of the nodes as in traditional meshing. Generally, the objective of meshless methods is to eliminate at least part of the mesh structure by constructing approximations entirely on nodes often referred to as *points* or *particles*. However, in many meshless techniques an auxiliary *background-mesh* is used in parts of the method. Nevertheless, it becomes possible to solve large classes of problems without remeshing and therefore with minor computational costs.
One of the first approaches was the generalization of Finite-Difference Methods (FDM) working on arbitrary, irregular grids [52]. Here, the *star*-concept was introduced to derive an approximation for each central node by using local, truncated Taylor series expansions. This usually leads to an overdetermined set of linear equations and the solution is obtained by a least square approximation. In recent work [59], weighted least squares (WLS) and moving least squares (MLS) interpolation methods with simple point collocation techniques have been used for the numerical solution of a wide range of problems in computational mechanics and are generally referred to as Finite Point Methods (FPM).
An alternative approach is the method of Smooth Particle Hydrodynamics (SPH), also called the Free Lagrange method, which only depends on a set of scattered particles and achieved considerable popularity in computational physics and astrophysics [56, 68]. The foundation of SPH is interpolation theory. Since there is no mesh involved, the method can handle large deformations of the computational domain and is particularly suited for problems with free or moving boundaries. On the other hand, its accuracy compared to other methods is rather low.
Later, a parallel path of constructing meshless methods has been investigated, where MLS approximations are used in a Galerkin method and called the Diffusive Element Method (DEM) [60]. An extension of this method has been proposed in [12], and named the Element-Free Galerkin (EFG) method. Both methods require a regular cell structure as auxiliary mesh and their computational cost is substantially more expensive than that for SPH.
A further approach of meshless methods is the Finite Mass Method (FMM) [31], a Lagrangian method based on a discretization of mass, not of space. Mass is subdivided into small packets of finite extension, which are moved and deformed under the influence of internal and external forces and the laws of thermodynamics.
A comparative study of many of these methods can be found in the overview article of Duarte [24] or the comprehensive book of Liu [53].

1.2 Introduction

Many physical phenomena in transport processes are described by time-dependent hyperbolic conservation laws. The governing scalar equation for multi-dimensional problems has the form

$$\frac{\partial u}{\partial t} + \nabla f(u) = 0 \tag{1.1}$$

where for some domain $\Omega \subset \mathbb{R}^d$, $d \geq 1$, and a compact time interval $I = [0, T]$, $T > 0$, the function $u : I \times \Omega \to \mathbb{R}$ is unknown. Moreover, $f(u) = (f_1(u), \ldots, f_d(u))^T$ denotes the *flux vector*. In this paper, we consider numerically solving (1.1) on given initial conditions

$$u(0, x) = u_0(x), \qquad \text{for } x \in \Omega = \mathbb{R}^d, \tag{1.2}$$

and for *nonlinear* flux functions f.
In previous work [9, 11], a new adaptive meshfree advection scheme has been proposed for numerically solving (1.1) for the special case, where

$$f(u) = a \cdot u, \tag{1.3}$$

in which case we obtain the *linear* (passive) advection equation

$$\frac{\partial u}{\partial t} + a \cdot \nabla u = 0 \tag{1.4}$$

provided that the given velocity field

$$a = a(t, x) \in \mathbb{R}^d, \qquad t \in I, \, x \in \Omega,$$

is divergence-free.
The method in [9, 11] is a combination of an adaptive semi-Lagrangian method (ASLM) [7, 8] and the meshfree radial basis function interpolation. The resulting advection scheme is used for the simulation of tracer transportation in the arctic stratosphere [11]. We remark that the scheme in [9, 11] is a *method of characteristics*, see [21, 34]. Indeed, the characteristic curves of (1.4) coincide with the trajectories of fluid particles, and the meshfree ASLM in [9, 11] captures the flow of particles along their characteristic curves. This is accomplished by computing backward trajectories for a finite set of current particles at each time step, whereas the node set is adaptively modified during the simulation.
Here an adaptive meshfree method of backward characteristics is designed for the purpose of numerically solving *nonlinear* equations of the form (1.1). In contrast to the linear case, a nonlinear flux function f usually leads to *discontinuities* in the solution u even if the initial condition (1.2) is smooth. These

shocks are observed in many relevant applications in fluid dynamics, meteorology, astrophysics, petroleum reservoir simulation, etc. The characteristics-based method in [9, 11] becomes unwieldy or impossible in nonlinear problems where the evolution of the flow along the characteristic curves may be much more complicated or characteristic curves may even be not defined, cf. [26], Subsection 6.3.1. Therefore, we apply a *vanishing viscosity* approach yielding the modified advection-diffusion equation

$$\frac{\partial u}{\partial t} + \nabla f(u) = \epsilon \cdot \Delta u, \tag{1.5}$$

with $\epsilon > 0$ being the artificial *diffusion coefficient.*
When it comes to extending the advection scheme of [9, 11], the local interpolation scheme is to be modified accordingly. The extension of the advection scheme is subject of the discussion in Section 1.3. The two remaining major ingredients, local *thin plate spline* interpolation, and the customized adaption rules, are then explained in the Sections 1.4 and 1.5.
Finally, the good performance of the resulting adaptive and meshfree method of backward characteristics is shown by numerical results in Section 1.6, where we consider using two different nonlinear model problems: *Burgers equation*, a standard test case, where

$$f(u) = \frac{1}{2}u^2 \cdot r, \tag{1.6}$$

with flow direction $r \in \mathbb{R}^d$, and the *Buckley-Leverett equation*, whose flux function has the form

$$f(u) = \frac{u^2}{u^2 + \mu(1-u)^2} \cdot r. \tag{1.7}$$

The Buckley-Leverett equation models the saturation of a two-phase flow in a porous medium when neglecting gravitational forces or capillary effects. In this case, the value of μ in (1.7) is the ratio of the two different fluid viscosities. This model problem is typically encountered in *oil reservoir modelling.* Details on this particular application are explained in Section 1.6 and the final Chapter 4.

1.3 Meshfree Method of Backward Characteristics

For the special case of passive advection (1.3), the scalar function u is constant along *trajectories* (*streamlines*) whose shapes are entirely and uniquely determined by the given velocity field $a \equiv a(t, x)$. Likewise, in the nonlinear

case the solution u is constant along trajectories of fluid particles, characteristic curves. In contrast to the linear case of passive advection, these characteristic curves do, however, depend on u.
In order to numerically solve the modified equation (1.5), the adaptive meshfree semi-Lagrangian method of [9, 11] is extended as follows. At each time step $t \longrightarrow t+\tau$, with $\tau > 0$ being the time step size, the values $u(t+\tau, \xi)$ at a current finite set Ξ of nodes, each of which corresponding to a flow particle, are computed from the previous values $u(t, \xi)$, $\xi \in \Xi$. Initially, the set Ξ^0 is randomly chosen in the computational domain Ω.
Starting point of the method is the Lagrangian form of (1.5),

$$\frac{du}{dt} = \epsilon \cdot \Delta u,$$

where $\frac{du}{dt} = \frac{\partial u}{\partial t} + \nabla f(u)$ is the material derivative. This leads us to the discretization

$$\frac{u(t+\tau, \xi) - u(t, x^-)}{\tau} = \epsilon \Delta u(t, x^-),$$

where $x^- \equiv x^-(\xi)$ is the *upstream location* of the node ξ. Note that a particle located at the upstream point x^- at time t moves along its trajectory and arrives at ξ at time $t+\tau$. Having computed x^- for any $\xi \in \Xi$, the desired approximation of $u(t+\tau, \xi)$ would thus be given explicitly by

$$u(t+\tau, \xi) = u(t, x^-) + \tau \cdot \epsilon \Delta u(t, x^-), \qquad \text{for } \xi \in \Xi. \tag{1.8}$$

But on given $\xi \in \Xi$, the *exact* location of the upstream point x^- is usually not known. A linearized approximation of x^- is given by

$$\tilde{x} = \xi - \beta,$$

where $\beta = \tau v$ and $v = \frac{\partial f(u)}{\partial u}$ is the advection velocity, i.e.

$$v(t, x) = \dot{x} = \frac{dx}{dt}. \tag{1.9}$$

In order to compute $\tilde{x}$, we need to solve the ordinary differential equation (ODE) in (1.9). Figure 1.1 displays the backward trajectory of a node $\xi \in \Xi$, its corresponding upstream point x^-, along with a linear approximation of the trajectory, leading to $\tilde{x} \approx x^-$.
For computing $\tilde{x}$, our implementation utilizes a fixed point iteration, based on the midpoint rule, already used in the seminal paper on semi-Lagrangian discretization by Robert [70]:

$$\beta^{(k+1)} = \tau \cdot v\left(t + \frac{\tau}{2}, \xi - \frac{\beta^{(k)}}{2}\right). \tag{1.10}$$

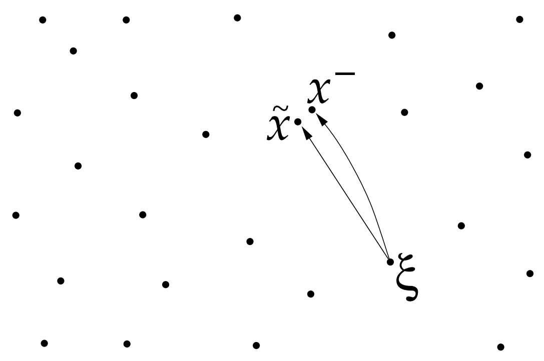


Figure 1.1: Upstream point x^- of the node $\xi \in \Xi$, and its linearized approximation $\tilde{x} \approx x^-$.

Note that the above iteration (1.10) relies on the evaluation of v at the intermediate time $t + \frac{\tau}{2}$. In the situation of passive advection, this can be accomplished by the evaluation of the given wind field $a \equiv v$. But in the nonlinear case, the velocity v does also depend on the solution u. In order to compute $v\left(t + \frac{\tau}{2}, \xi - \frac{\beta^{(k)}}{2}\right)$ we employ the following extrapolation scheme.

$$v\left(t + \frac{\tau}{2}, \cdot\right) = \frac{3}{2}\, v(t, \cdot) - \frac{1}{2}\, v(t - \tau, \cdot). \tag{1.11}$$

Initially, in order to obtain the required values of $u(\tau, \cdot)$ from the given initial conditions (1.2), we use a generalized *two-level* Lax-Friedrich scheme on $\Xi^0 \equiv \Xi^\tau$.
Having computed the values $u(t + \tau, \xi)$, for all $\xi \in \Xi$, via (1.8), the current node set $\Xi \equiv \Xi^t$ (at time t) is finally modified by the removal (coarsening), and the insertion (refinement) of nodes, yielding a new node set $\Xi \equiv \Xi^{t+\tau}$ (at time $t + \tau$). The adaption of the nodes relies on a customized a posteriori error indicator, to be explained in Section 1.5.
We finally remark that the characteristics-based discretization scheme, as introduced in this section, has been theoretically analyzed by Falcone and Ferretti [28]. Their concise convergence analysis shows that the semi-Lagrangian method is of second order in time and space, provided that the interpolation method is of second order.

1.4 Interpolation using Thin Plate Splines

In this section, we are concerned with computing approximations for the values $u(t, \cdot), \Delta u(t, \cdot)$ in the advection step (1.8), and $v(t, \cdot), v(t - \tau, \cdot)$ in the

extrapolation (1.11). To this end, we work with a *local* interpolation scheme, which first collects on given $x \in \Omega$ a set $\mathcal{N}_x \equiv \mathcal{N}_x^t$ of current neighbours (at time t) in the local neighbourhood of x, before the (known) function values of u at these neighbouring points are used for computing the approximations of $u(t,\cdot), \Delta u(t,\cdot), v(t,\cdot), v(t-\tau,\cdot)$. But this requires some preparations. Therefore, we defer details to later in this section.
As already observed in [9, 11], the interpolation is critical for the advection method's performance in terms of its efficiency and approximation quality. Indeed, the interpolation scheme does not only affect the evaluation of the model $u(x) \equiv u(t,x)$, but also the adaption rules, to be explained in the following Section 1.5, do heavily rely on the interpolation. Altogether, a reliable and robust interpolation scheme of good approximation quality is required.
As explained in the previous work [9, 11], *thin plate splines* provide suitable and powerful meshfree methods for scattered data interpolation. But the setting in [9, 11] needs to be extended here. According to the general framework of thin plate spline interpolation, dating back to Duchon [25], we work with interpolants of the form

$$s(y) = \sum_{\nu \in \mathcal{N}_x} \lambda_\nu \phi_k(\|y-\nu\|) + \sum_{|\alpha| \le k} \mu_\alpha y^\alpha, \tag{1.12}$$

in order to solve interpolation problems of the form

$$s(\nu) = u(\nu), \qquad \text{for all } \nu \in \mathcal{N}_x. \tag{1.13}$$

In (1.12), the *radial basis function* $\phi_k(r) = r^{2k}\log(r)$, $k \ge 1$, is referred to as *thin plate spline*, and $\|\cdot\|$ is the Euclidean norm on $\mathbb{R}^d$. Moreover, $\alpha = (\alpha_1, \ldots, \alpha_d)$ in (1.12) is a multi-index, a d-tuple of non-negative integers, with absolute value $|\alpha| = \alpha_1 + \ldots + \alpha_d$, and where

$$y^\alpha = y_1^{\alpha_1} \cdot \ldots \cdot y_d^{\alpha_d} \qquad \text{for } y = (y_1, \ldots, y_d)^T \in \mathbb{R}^d.$$

The above $N = \#\mathcal{N}_x$ interpolation conditions in (1.13) constitute a linear system of N equations with $N+Q$ unknowns in the coefficient vectors $\lambda = (\lambda_\nu)_{\nu \in \mathcal{N}_x} \in \mathbb{R}^N$ of the *major part* and $\mu = (\mu_\alpha)_{|\alpha| \le k} \in \mathbb{R}^Q$ of the *polynomial part* of s in (1.12), where $Q = \binom{k+d}{d}$ is the dimension of the linear space Π_k^d of all real-valued polynomials in d variables and of degree at most k.
In order to eliminate the Q additional degrees of freedom, the coefficients λ_ν in (1.12) are subject to the additional Q side conditions

$$\sum_{\nu \in \mathcal{N}_x} \lambda_\nu \nu^\alpha = 0, \qquad \text{for all } |\alpha| \le k. \tag{1.14}$$

Altogether, solving (1.13) under constraints (1.14) leads us to the linear system

$$\begin{bmatrix} A_{\mathcal{N}_x} & P_{\mathcal{N}_x} \\ P_{\mathcal{N}_x}^T & 0 \end{bmatrix} \cdot \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} (u(\nu))_{\nu \in \mathcal{N}_x} \\ 0 \end{bmatrix}, \tag{1.15}$$

where

$$A_{\mathcal{N}_x} = (\phi_k(\|\nu - \nu'\|))_{\nu,\nu' \in \mathcal{N}_x} \in \mathbb{R}^{N \times N} \text{ and } P_{\mathcal{N}_x} = (\nu^\alpha)_{\nu \in \mathcal{N}_x; |\alpha| \leq k} \in \mathbb{R}^{N \times Q}.$$

As to the well-posedness of thin plate spline interpolation, we remark that there always exists an interpolant s of the form (1.12) satisfying (1.13). Moreover, s is unique provided that the points in $\mathcal{N}_x$ are Π_k^d-unisolvent, i.e.

$$p(\nu) = 0 \quad \text{for all} \quad \nu \in \mathcal{N}_x \qquad \Longrightarrow \qquad p \equiv 0 \tag{1.16}$$

for $p \in \Pi_k^d$. In this case, the linear system (1.15) has a unique solution, and thus the thin plate spline interpolation scheme achieves to reconstruct polynomials in Π_k^d exactly. Note that (1.16) is a very mild side condition. E.g., for $k = 1$ this requires that the points in $\mathcal{N}_x$ must not all lie on a straight line.

For further details on thin plate spline interpolation, including their optimality properties, and alternative choices for *radial basis functions* ϕ, we refer to the recent tutorial paper [42].

Now let us finally turn to the approximation of the values $u(t, \cdot), \Delta u(t, \cdot)$, $v(t, \cdot), v(t - \tau, \cdot)$. In order to compute an approximation for $v(t, x)$, on given $x \in \Omega$, we first consider solving the interpolation problem

$$s(\nu) = v(t, \nu), \qquad \text{for all } \nu \in \mathcal{N}_x^t,$$

by using the ansatz (1.12) for s and with $k = 1$. This then gives us by $s(x)$ the desired approximation of $v(t, x)$, i.e. $s(x) \approx v(t, x)$. Likewise, the approximation of $v(t - \tau, \cdot)$ is computed by solving

$$s(\nu) = v(t - \tau, \nu), \qquad \text{for all } \nu \in \mathcal{N}_x^{t-\tau},$$

using exactly the same approach, but for the previous set $\mathcal{N}_x^{t-\tau}$ of neighbours. As to the approximation of $\Delta u(t, \tilde{x})$, this requires using a *smoother* instance of ϕ_k in (1.12), i.e. with $k > 1$, since $\Delta \phi_1(\|x\|)$ has a singularity at zero. We prefer to work with $\phi_2(r) = r^4 \log(r)$, whose Laplacian $\Delta \phi_2(\|x\|)$ is well-defined on all of $\mathbb{R}^d$. This amounts to first solving the interpolation problem

$$s(\nu) = u(t, \nu), \qquad \text{for all } \nu \in \mathcal{N}_{\tilde{x}}^t,$$

using the ansatz (1.12) with $k = 2$, which provides by

$$\Delta s(y) = \sum_{\nu \in \mathcal{N}_{\tilde{x}}^t} \lambda_\nu \Delta \phi_2(\|y - \nu\|) + 2 \left(\mu_{(2,0,\ldots,0)} + \mu_{(0,\ldots,0,2)} \right)$$

the desired approximation $\Delta s(\tilde{x}) \approx \Delta u(t, \tilde{x})$. Moreover, $s(\tilde{x}) \approx u(t, \tilde{x})$. In our numerical examples, however, we kept on using the basis function ϕ_1 for computing an approximation of $u(t, \tilde{x})$. This helps to avoid undesired oscillations near the shocks.

1.5 Adaption Rules

1.5.1 Error Indication

An effective strategy for the adaptive modification of the nodes requires well-motivated refinement and coarsening rules as well as a customized error indicator. We understand the error indicator $\eta : \Xi \to [0, \infty)$ as a function of the current node set $\Xi \equiv \Xi^t$ (at time t) which serves to assign a *significance* value $\eta(\xi)$ to each $\xi \in \Xi$. The value $\eta(\xi)$ is required to reflect the local approximation quality of the interpolation around $\xi \in \Xi$. The significances $\eta(\xi)$, $\xi \in \Xi$, are then used in order to flag single nodes $\xi \in \Xi$ as "to be refined" or "to be coarsened" according to the following criteria.

Definition 1 *Let $\eta^* = \max_{\xi \in \Xi} \eta(\xi)$, and let $\theta_{\text{crs}}, \theta_{\text{ref}}$ be two tolerance values satisfying $0 < \theta_{\text{crs}} < \theta_{\text{ref}} < 1$. We say that a node $\xi \in \Xi$ is* `to be refined`*, iff $\eta(\xi) > \theta_{\text{ref}} \cdot \eta^*$, and ξ is* `to be coarsened`*, iff $\eta(\xi) < \theta_{\text{crs}} \cdot \eta^*$.*

In our numerical examples typical choices for the relative tolerance values are $\theta_{\text{crs}} = 0.001$ and $\theta_{\text{ref}} = 0.2$. Note that a node ξ cannot be refined and be coarsened at the same time; in fact, it may neither be refined nor be coarsened.
Now let us turn to the definition of the error indicator η. We follow along the lines of [35], where a local scheme for the detection of discontinuities of a surface from scattered data was developed, and we let

$$\eta(\xi) = |u(\xi) - s(\xi)|,$$

where the thin plate spline interpolant $s \equiv s_{\mathcal{N}}$ matches current values of $u \equiv u(t, \cdot)$ at a neighbouring set $\mathcal{N} \equiv \mathcal{N}(\xi) \subset \Xi \setminus \{\xi\}$ of current nodes, i.e. $s(\nu) = u(\nu)$ for all $\nu \in \mathcal{N}$. In our numerical examples, we preferred to use the thin plate spline $\phi_1(r) = r^2 \log(r)$, and thus the ansatz (1.12) with $k = 1$. This particular interpolation scheme achieves to reconstruct linear polynomials. In this case, the value $\eta(\xi)$ vanishes whenever u is linear around ξ. Moreover, the value $\eta(\xi)$ is small whenever the reproduction quality of u by s around ξ is good. In contrast, a high value of $\eta(\xi)$ typically indicates that u is subject to strong variation locally around ξ.

1.5.2 Coarsening and Refinement

In order to balance the approximation quality of the model against the required computational complexity we insert new nodes into regions where the value of η is high (refinement), whereas we remove nodes from Ξ in regions where the value of η is small (coarsening).
To avoid additional computational overhead and complicated data structures, effective adaption rules are required to be as simple as possible. In particular,

these rules ought to be given by *local* operations on the current node set Ξ. The following coarsening rule is in fact very easy and, in combination with the refinement, it turned out to be very effective as well.

Coarsening. A node $\xi \in \Xi$ is *coarsened* by its removal from the current node set Ξ, i.e. Ξ is modified by replacing Ξ with $\Xi \setminus \{\xi\}$.

As to the refinement of a node $\xi \in \Xi$, we follow along the lines of the previous paper [11], where the effective refinement rules were motivated on the basis of available local error estimates for radial basis function interpolation. For the special case of thin plate spline interpolation, the local error estimate at $x \in \Omega$ is according to Wu and Schaback [87] of the form

$$|u(x) - s(x)| \leq C \cdot h^k_{\mathcal{N},\varrho}(x) \tag{1.17}$$

where $C > 0$ is a constant depending on u, and (for some radius $\varrho > 0$)

$$h_{\mathcal{N},\varrho}(x) = \sup_{\|y-x\|<\varrho} d_{\mathcal{N}}(y)$$

is the local *fill distance* of $\mathcal{N}$ around x, with

$$d_{\mathcal{N}}(y) = \min_{\nu \in \mathcal{N}} \|y - \nu\|$$

being the Euclidean distance between the point y and the set $\mathcal{N}$. We remark that for the special case $k = 1, d = 2$, the thin plate spline interpolation scheme is due to [35] locally of second order accuracy.
As suggested in [11], the reduction of the local error (1.17) around any $\xi \in \Xi$ is accomplished by reducing the distance function $d_{\mathcal{N}} = \min_{\nu \in \mathcal{N}} \| \cdot - \nu\|$ in a local neighbourhood of ξ.
Now recall that for a fixed node set $\Xi \subset \mathbb{R}^d$ and any $\xi \in \Xi$, the *Voronoi tile*

$$V_\xi = \left\{x \in \mathbb{R}^d \,:\, d_\Xi(x) = \|x - \xi\|\right\} \subset \mathbb{R}^d$$

of ξ w.r.t. Ξ contains all points in $\mathbb{R}^d$ whose nearest point in Ξ is ξ. The tile V_ξ is a convex polytope, whose vertices are referred to as the *Voronoi points*, forming a finite point set $\mathcal{V}_\xi$ in the neighbourhood of ξ. Figure 1.2 shows the Voronoi tile V_ξ of a point ξ along with the set $\mathcal{V}_\xi$ of its Voronoi points. For more details on Voronoi diagrams, we refer to [65]. Observe, that for $\xi \in \mathcal{N}$ the distance function $d_{\mathcal{N}}$ is convex on V_ξ. Moreover, it has local maxima at the Voronoi points in $\mathcal{V}_\xi$. Altogether, this gives rise to define the local refinement of nodes as follows.

Refinement. A node $\xi \in \Xi$ is *refined* by the insertion of its Voronoi points into the current node set Ξ, i.e. Ξ is modified by replacing Ξ with $\Xi \cup \mathcal{V}_\xi$.

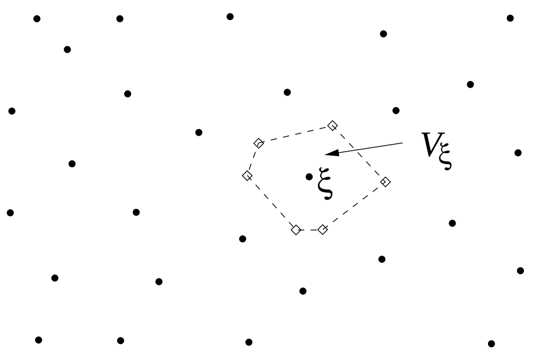


Figure 1.2: Refinement of the node ξ. The Voronoi points ($\diamond$) are inserted.

1.6 Numerical Results

We have implemented the proposed advection scheme for the special case of two dimensions, i.e. $d = 2$. In this section, the performance of the method on *nonlinear* equations (1.5) is shown. To this end, we considered using two model problems: *Burgers equation*, where the flux function is given by (1.6), and the *Buckley-Leverett equation*, whose flux function is (1.7).

1.6.1 Burgers Equation

Burgers [17] introduced the nonlinear flux function (1.6) in the hyperbolic conservation law (1.1) as a mathematical model of free turbulence in fluid dynamics. Burgers equation is nowadays a standard test case, popular mainly for the following reasons: (a) it contains the simplest form of a nonlinear advection term $u \cdot \nabla u$ simulating the physical phenomenon of wave motion, and (b) for its shock wave behaviour: As soon as the shock front occurs, there is no classical solution of the PDE and its weak solution becomes discontinuous.
In the test case, the following initial condition is used.

$$u_0(x) = \begin{cases} \exp\left(\frac{\|x-c\|^2}{\|x-c\|^2-R^2}\right) & \text{for } \|x-c\| < R \\ 0 & \text{otherwise} \end{cases}$$

with $R = 0.25$, $c = (0.3, 0.3)^T$, and we let the unit square $\tilde{\Omega} = [0,1]^2 \subset \Omega = \mathbb{R}^2$ be the computational domain, cf. [30]. Moreover, we selected the value $\epsilon = 2 * 10^{-3}$ for the diffusion coefficient in (1.5), and we let $r = (1,1)^T$ in (1.6), yielding a flow field along the diagonal of $\tilde{\Omega}$.

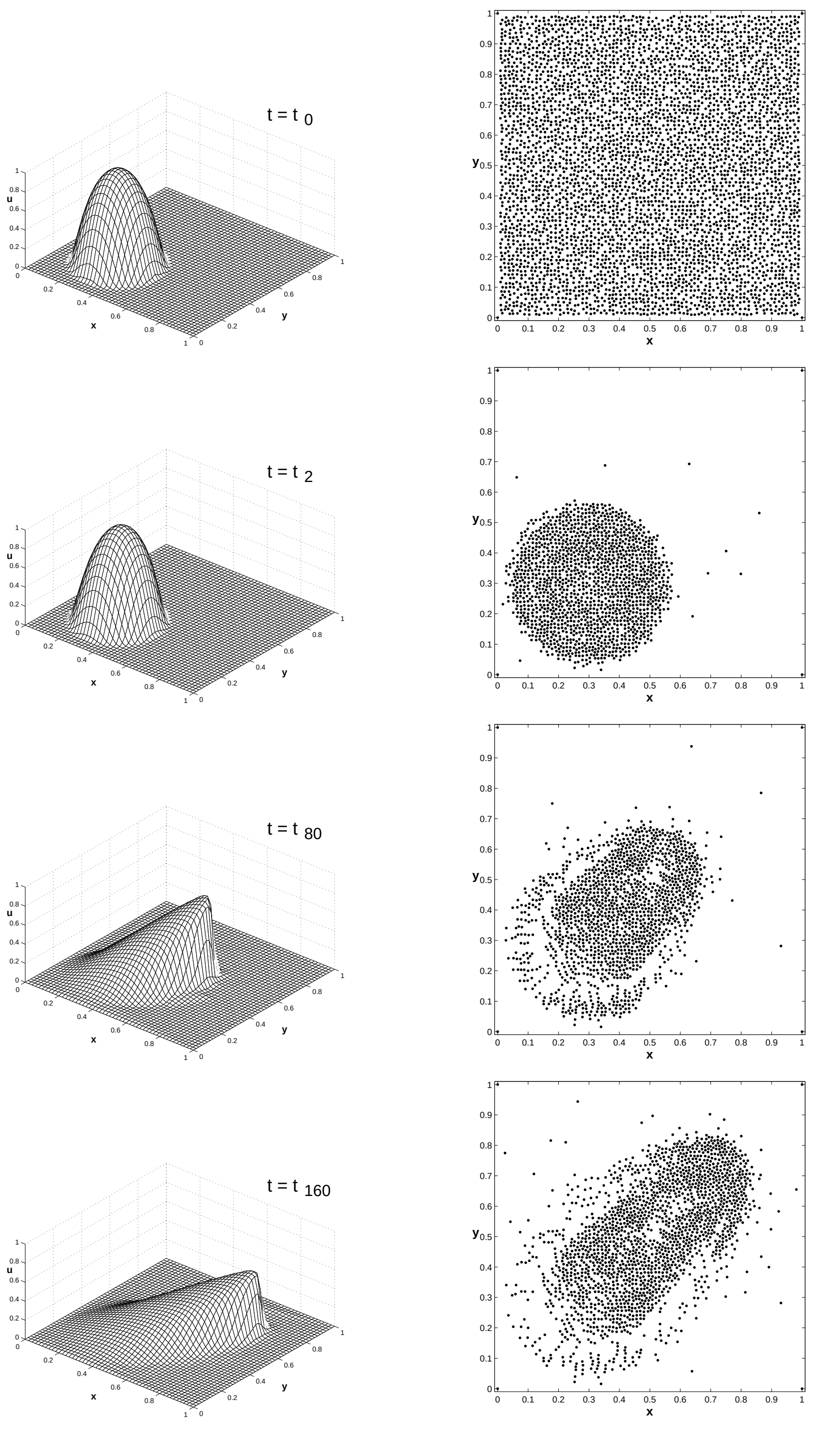


Figure 1.3: Burgers equation: Evolution of the solution u for four different time steps, $t = t_0, t_2, t_{80}, t_{160}$ (left), and the corresponding node distribution (right).

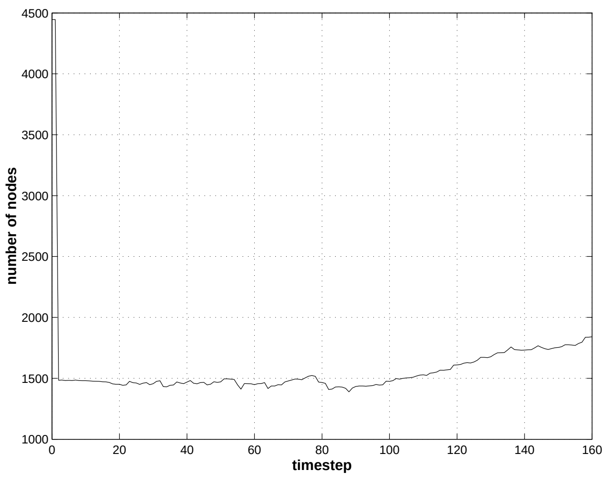


Figure 1.4: Burgers equation: Number of nodes per time step.

Initially, u_0 is sampled at 4446 randomly chosen points in the unit square, yielding the initial node set $\Xi^0 \subset \Omega$. Moreover, a constant time step size $\tau = 0.0075$ has been selected. A plot of the solution u at the three time steps $t_2 = 2 \cdot \tau, t_{80} = 80 \cdot \tau$, and $t_{160} = 160 \cdot \tau$ is shown in Figure 1.3.
Observe that the node adaption scheme achieves to localize the support of the solution u very effectively. Moreover, the shock front propagation is reasonably well resolved by the distribution of the current nodes, cf. the time steps $t = t_{80}, t_{160}$ in Figure 1.3. As already observed in the numerical examples of the previous paper [11], where we studied passive (linear) advection, this confirms the utility of the customized adaption rules yet once more.
Figure 1.4 shows a plot of the number of nodes per time step. The number of nodes, initially $\#\Xi^0 = \#\Xi^{t_1} = 4446$, immediately drops down to $\#\Xi^{t_2} = 1484$. This is due to *adaptivity*, starting at time $t = t_2$. Then, the number of nodes remains roughly constant for a while. Due to the growing support of u, a moderate increase of the number of nodes can be observed in the second half of the simulation, resulting in $\#\Xi^{t_{160}} = 1841$ after the final time step $t = t_{160}$.

1.6.2 Buckley-Leverett Equation

Buckley and Leverett [16] introduced the flux function (1.7) in hyperbolic conservation laws of the form (1.1) in order to describe the flow of two different liquids in a porous medium. This two-phase flow problem is typically encountered in applications of *oil reservoir modelling*, where specific enhanced oil recovery processes are simulated. When an oil reservoir is tapped, a certain amount of oil flows out on its own due to the high reservoir pressure. After the flow has stopped, there is usually still a large amount of oil in

the reservoir pores. A standard method like waterflooding can now be used, where a fluid (e.g. water) is injected into a well in a reservoir to displace the contained hydrocarbons (e.g. oil) and produce them from another well. If gravity effects and capillary forces are neglected and the *flux field* (*total velocity field*) is known, the two-phase flow can be approximated by the saturation equation only, namely the *Buckley-Leverett equation*. Further details are explained in the final Chapter 4.
In our test case, we assume a single water injection well in the center of a 100% oil saturated, homogeneous, porous medium (reservoir rock) defined on the unit square. We allow open boundaries, so that the displaced oil can leave the computational domain $\tilde{\Omega} = [0,1]^2 \subset \Omega = \mathbb{R}^2$. Here, the function u quantifies the saturation of water in the reservoir pores. The values of u lie between 0 and 1, where $u \equiv 1$ denotes pure water and $u \equiv 0$ pure oil. In this case, the initial condition (1.2) is given by

$$u_0(x) = \begin{cases} 1 & \text{for } \|x - c\| \leq R \\ 0 & \text{otherwise} \end{cases}$$

with the injection well centered at $c = (0.5, 0.5)^T$ and with radius $R = 0.05$. Thus, initially the water saturation u inside the injection well is 1 and outside the well it is 0. A radial total velocity field $r = (x - c)/\|x - c\|$ in (1.7) is assumed, so that the injected water should displace the oil radially. We remark, that any near borehole effects are neglected. We decided to select a constant time step size $\tau = 0.001$, and the simulation comprises 264 time steps. Moreover, we let $\epsilon = 4 * 10^{-3}$ for the diffusion coefficient in (1.5). Finally, we selected the value $\mu = 0.5$ for the viscosity ratio of water and oil, appearing in the flux function (1.7).
The evolution of the oil's displacement by water is shown in Figure 1.5 (3D view and top view) and Figure 1.6 (side view) for four different times $t = t_0, t_2, t_{132}, t_{264}$. Initially, the function u_0 is sampled at a set Ξ^0 of $\#\Xi^0 = 4446$ of randomly distributed nodes.
Already after the second time step, the nodes are adapted to the vicinity of the well, and the number of nodes immediately drops down to $\#\Xi^{t_2} = 128$, see Figure 1.7. As soon as water flows into the well, a shock wave is formed, and a discontinuity in u can be observed, cf. Figures 1.5 and 1.6. In this case, a certain amount of oil is displaced immediately, whereas beyond the shock front there is a mixture of oil and water, with less oil at proceeding time. This phenomenon is referred to as the *rarefaction wave*.
Figures 1.5 and 1.6 also show a comparison of the numerical and the analytic solution, the latter determined by Welge's tangent method [86]. It can be observed, that the adaptive distribution of nodes achieves to capture the propagating shock front (the solid line in Figure 1.6) well. This helps to reduce the required computational costs while maintaining the accuracy, due

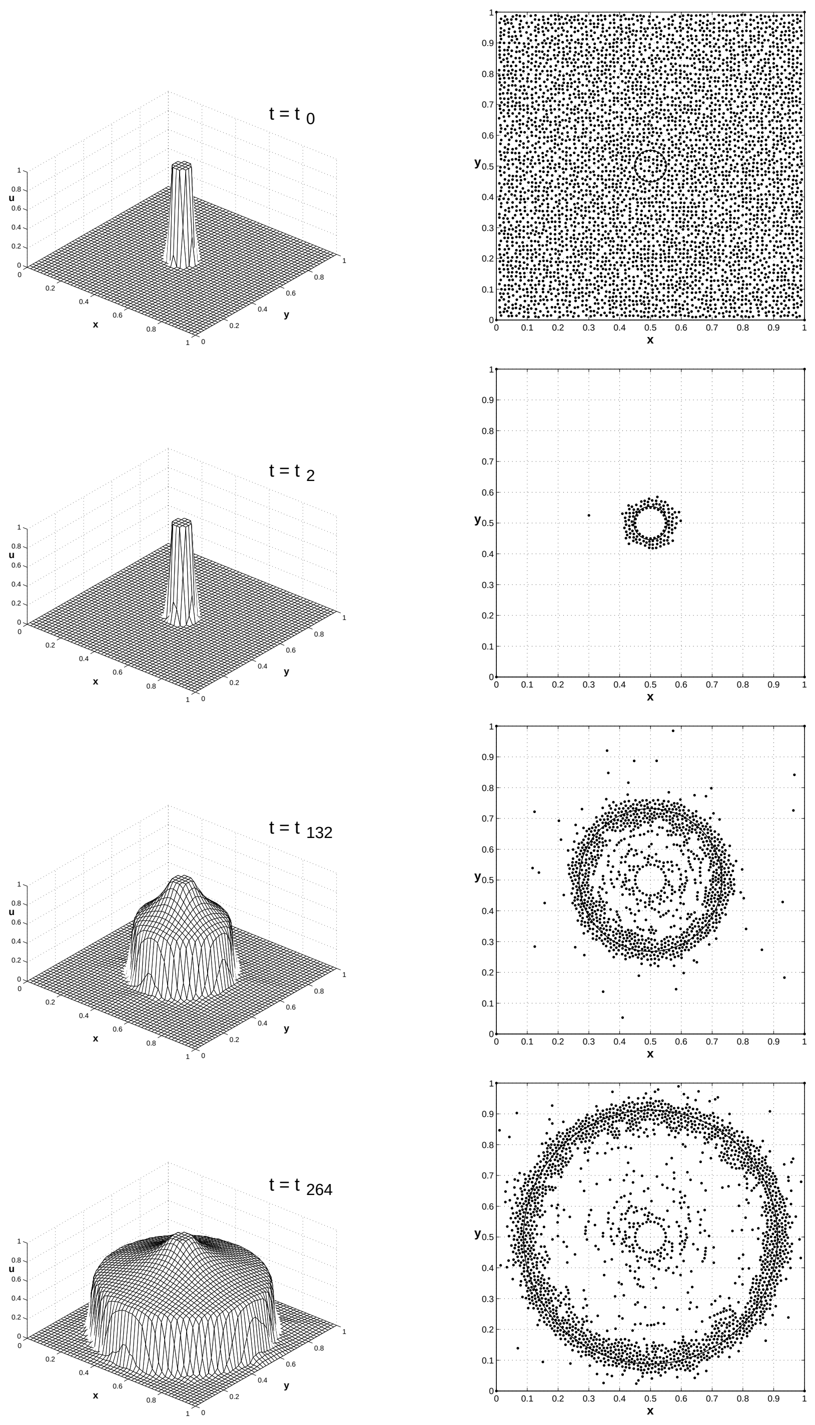


Figure 1.5: Buckley-Leverett equation: The saturation u (3D view, left column; top view, right column), and the analytic solution (solid line, right column) for four different time steps $t = t_0, t_2, t_{132}, t_{264}$ (left).

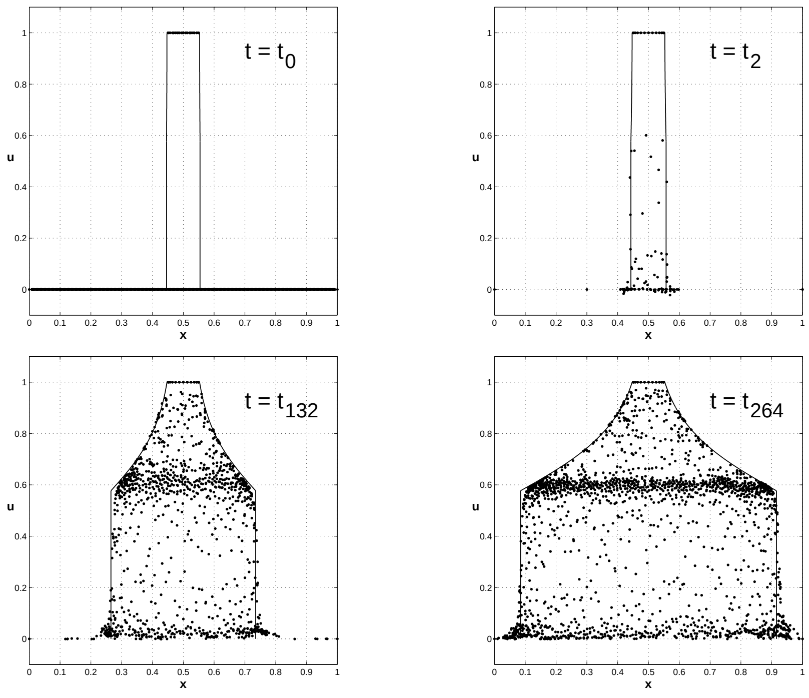


Figure 1.6: Buckley-Leverett equation: The saturation u (nodes), and the analytic solution (solid line) for four different time steps $t = t_0, t_2, t_{132}, t_{264}$, side view.

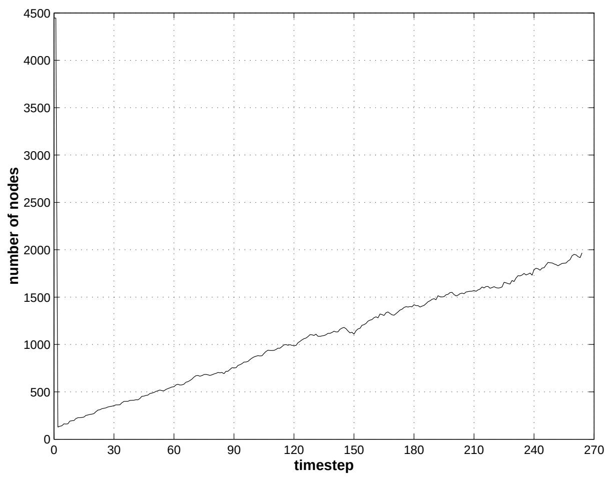


Figure 1.7: Buckley-Leverett equation: Number of nodes per time step.

to higher resolution around the shock. Note that since the radius of the propagating shock front increases linearly with time, the number of nodes increases linearly at proceeding time, ending up with $\#\Xi^{t_{264}} = 1967$ at the final time step $t = t_{264}$ of the simulation, see Figure 1.7 .

1.7 Conclusion

The meshfree method of backward characteristics for *linear* (passive) advection from the previous work [9, 11] has been extended for solving *nonlinear* transport problems. In order to avoid degeneration of characteristic curves, a vanishing viscosity approach has been added to the advection scheme. Moreover, the local interpolation scheme, using thin plate splines, has been modified accordingly. An error indicator derived from the interpolation scheme yields in combination with customized adaption rules an effective distribution of nodes during the simulation. Numerical examples on nonlinear advection-dominated PDEs confirm the good performance of the proposed method.
We finally remark that the proposed method is potentially useful for higher dimensional problems. Note, that our approach poses no principal restrictions to enhancements in higher dimensions. However, the implementation of the scheme is not yet extended to dimensions $d > 2$.
We also point out, that the vanishing viscosity approach, i.e. the introduction of the Laplacian Δu, imposes a time step restriction to the proposed explicit method due to stability requirements. An implicit version of the scheme would necessitate the solution of a large system of nonlinear equations. The implementation of an efficient technique solving this kind of problem was not considered, yet.

Furthermore, the proposed meshfree method is not *conservative* in the classical sense, cf. Section 4.1 in [50]. In fact, it is unclear how to measure the mass, as each particle carries a concentration value, which is representative for some region around it. The size of that region should depend on the density of nodes in the surrounding region.

As many relevant applications, however, require the conservation of mass, momentum, or energy, constituting the fundamental laws of physics, this property should also be satisfied by the numerical scheme. Therefore, the following Chapter 2 focuses on the development of a conservative, adaptive advection schemes.

Chapter 2

Adaptive Conservative Advection

In this Chapter a *conservative* semi-Lagrangian advection scheme is designed in order to solve *linear* advection equations in two space variables. The proposed method works with a combination of finite volumes on an unstructured mesh, which is given by a Voronoi diagram, and the semi-Lagrangian approach for time evolution as introduced in Chapter 1. Moreover, the mesh is subject to adaptive modifications during the simulation, which serves to effectively combine good approximation quality with small computational costs. The required adaption rules for the refinement and the coarsening of the mesh rely on a customized error indicator similar to the rules explained in Section 1.5 of Chapter 1. Additionally, the implementation of boundary conditions is addressed. Numerical results finally confirm the good performance of the proposed conservative and adaptive advection scheme.

2.1 General Overview

Finite volume schemes provide well-established conservative methods for solving the governing equations of advection problems. However, explicit, standard finite volume methods possess time step restrictions for the sake of their stability. In contrast to Eulerian schemes, Lagrangian particle methods require often less restrictive conditions on the time step.
Here, a set of particles is followed through space and time. However, the disadvantage of Lagrangian schemes is, that the set of initial distribution of particles may become greatly changed and rendered unsuitable to approximate the solution of the problem. To overcome this problem, a semi-Lagrangian approach can be used, in which the paths of particles are followed, that pass through given positions. In other words, the particles are traced back over one time step to determine their *upstream* locations at the previous time level. The solution values carried by the particles at the upstream loca-

tions are usually determined by some interpolation as shown for a meshless method in Chapter 1. The advantage of the semi-Lagrangian approach is, that it combines the less restrictive stability requirement of particle methods with the Eulerian idea of fixed nodes, where particles have to pass through. The main difficulty of semi-Lagrangian methods is the satisfaction of discrete conservation, i.e. Lagrangian methods are usually *not* conservative. A more comprehensive discussion on these and related aspects concerning Eulerian versus Lagrangian schemes for hyperbolic conservation laws is offered in the textbooks [26, 50, 57].

Considerable effort has been made recently in order to construct *conservative* semi-Lagrangian methods [47, 66, 67]. More recently, conservative semi-Lagrangian Finite Volume schemes have been developed in [64, 75]. Their formulation of the discrete problem is based on satisfying a physical conservation constraint in away that conservation is satisfied by construction. With their approach, higher order schemes can be developed in a straightforward manner. The main weaknesses of their advection schemes are, that they work on Cartesian meshes and they are not adaptive. In fact, these papers' methods work with a rectangular grid, respectively, which is fixed throughout a simulation. In our opinion, however, *adaptivity* is an essential requirement, especially when modelling multiscale phenomena, in order to effectively balance a methods approximation quality and its computational costs.

2.2 Introduction

A second order accurate semi-Lagrangian finite volume method is proposed for passive advection, which is both conservative and adaptive. To this end, the method works with control volumes of an unstructured adaptive mesh, which is given by the Voronoi diagram of current nodes (particles). The node set is corresponding to a set of moving particles. Moreover, the node set is subject to adaptive modifications during the simulation, and so is the Voronoi diagram to be updated accordingly. This requires customized adaption rules for the dynamic refinement and coarsening of the node set as discussed in Chapter 1. The adaption rules rely on available local error estimates at the current nodes as discussed in Section 1.5.

We remark that the proposed method can be viewed as an extension of the previous conservative but non-adaptive advection schemes by Scroggs & Semazzi [75] and by Phillips & Williams [64]. Indeed, adaptivity is one crucial feature of our method. It enhances the method's accuracy and allows us to capture discontinuities of the solution with high resolution.

Additionally, the proposed scheme is highly flexible by using the Voronoi diagram of the current nodes. Therefore, there is no restriction on the dis-

tribution of the nodes in the computational domain.
Consider the *passive advection* equations, given by a scalar time-dependent *hyperbolic conservation law* of the form

$$\frac{\partial u}{\partial t} + \nabla a u = 0, \tag{2.1}$$

where for a compact time interval $I = [0, T] \subset \mathbb{R}$, $T > 0$, and the computational domain $\Omega = \mathbb{R}^2$, the *velocity field*

$$a = a(t, x), \qquad t \in I,\ x \in \Omega,$$

is assumed to be given. In this case, the scalar solution $u : I \times \Omega \to \mathbb{R}$ of (2.1) usually corresponds to a physical quantity, such as density or concentration. Especially when modelling physical phenomena in relevant applications, such as chemical tracer transportation, *mass conservation*, i.e.,

$$\frac{d}{dt}\int_{\mathbb{R}^2} u(t, x)\, dx = 0,$$

is always an important requirement.
Here, we consider solving (2.1) numerically, on the given initial condition

$$u(0, x) = u_0(x), \quad \text{for } x \in \Omega. \tag{2.2}$$

To this end, we work with a *conservative* semi-Lagrangian adaptive advection scheme, whose construction is subject of the discussion in this Chapter. For the moment of the following discussion, we wish to avoid boundary conditions. To this end, we assume that the computational domain is the whole plane, i.e., $\Omega = \mathbb{R}^2$. But later in Subsection 2.4.3, where the implementation of boundary conditions is explained, we drop this assumption.
The following construction of our method combines the particle-based semi-Lagrangian approach of the previous papers [9, 10, 11] with a finite volume scheme on adaptive unstructured meshes. We remark that the particle-based scheme of [9] is applied in [11] on a real-world test case scenario concerning chemical tracer transportation over the arctic stratosphere. However, none of the *meshfree* particle schemes in [9, 10, 11] is conservative, which is a severe drawback in several relevant applications, such as the abovementioned tracer advection simulation.
Before we expand all relevant ingredients of our method in detail, let us briefly explain the basic ideas for the construction of the proposed *conservative* scheme. Similar to the scheme discussed in Chapter 1 the discretization works with a finite set $\Xi \subset \Omega$ of nodes, each of which corresponds to one flow particle at a time $t \in I$. Any *current* node set Ξ defines a unique Voronoi diagram $\mathcal{V}_\Xi$, which yields a partitioning of the computational domain $\Omega = \mathbb{R}^2$ into $\#\Xi$ finite control volumes, *Voronoi cells*, so that each cell in $\mathcal{V}_\Xi$ contains

exactly one node from Ξ. Given the *cell average values* over the current Voronoi cells at time t, initially given by using a suitable node set Ξ along with the initial condition (2.2), the advection step $t \to t + \tau$ in our scheme is accomplished as follows.
For each current Voronoi cell $V \in \mathcal{V}_\Xi$, a corresponding *upstream cell* $U \subset \Omega$ is constructed. The upstream cell U contains at time t the proportion of mass which is advected into the cell V during the time step $t \to t + \tau$. This duality relation between U and V is used in order to establish (local) mass conservation by

$$\int_U u_h(t,x)\,dx = \int_V u_h(t+\tau,x)\,dx, \tag{2.3}$$

where $u_h \approx u$ is an approximation to the solution u of the Cauchy problem (2.1), (2.2). Details on the construction of the upstream cells are discussed in Section 2.3.
Moreover, be it sufficient for the moment to remark that, in our second order advection scheme, u_h in (2.3) is a piecewise linear function over the current Voronoi diagram $\mathcal{V}_\Xi$. Details on the construction of u_h are explained in Subsections 2.4.1 and 2.4.2, whereas the implementation of boundary conditions is explained in Subsection 2.4.3. The adaption rules for the nodes in Ξ are similar to those discussed in Section 1.5 and are only reviewed briefly in Section 2.5. Finally, numerical results are shown for the two test case scenarios of Section 2.6, where the good performance of the proposed conservative advection scheme is illustrated.

2.3 Semi-Lagrangian Advection on Voronoi Cells

In this section, the construction of the abovementioned upstream cells is discussed. To this end, let us first recall some relevant ingredients from computational geometry, in particular Voronoi diagrams (see the textbook [65] for more details on Voronoi diagrams). For a fixed finite node set $\Xi \subset \mathbb{R}^2$, the *Voronoi diagram* $\mathcal{V}_\Xi = \{V_\xi\}_{\xi\in\Xi}$ of Ξ is a planar graph, which yields a partitioning

$$\mathbb{R}^2 = \bigcup_{\xi\in\Xi} V_\xi$$

of the plane into *Voronoi cells* of the form

$$V_\xi = \left\{ x \in \mathbb{R}^2 \, : \, \min_{\zeta\in\Xi} \|x-\zeta\| = \|x-\xi\| \right\} \subset \mathbb{R}^2.$$

Figure 2.1 shows the Voronoi diagram $\mathcal{V}_\Xi$ of a planar node set Ξ. For any $\xi \in \Xi$, its corresponding Voronoi cell V_ξ is a convex polytope containing all points in the plane whose nearest point in Ξ is ξ. The vertices of V_ξ are said to be the *Voronoi vertices* of V_ξ. The nodes in Ξ whose Voronoi cells are adjacent to the Voronoi cell V_ξ are said to be the *Voronoi neighbours* of ξ.

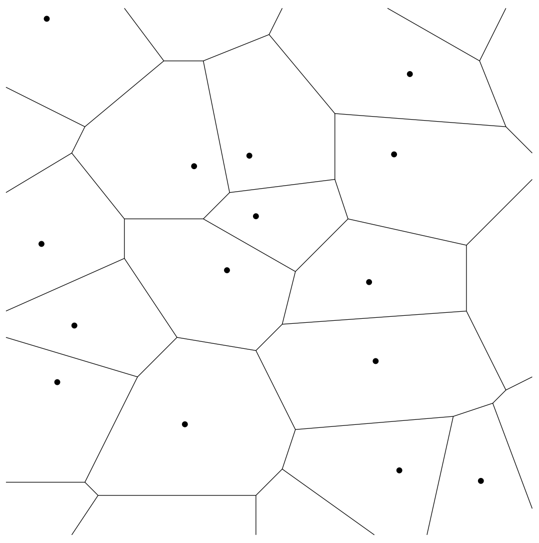


Figure 2.1: Voronoi diagram $\mathcal{V}_\Xi$ of a node set Ξ.

2.3.1 Computation of Upstream Vertices

The construction of any upstream cell U relies on the construction of its *upstream vertices.* In order to explain the construction of the upstream vertices, let $V \equiv V_\xi$ be the Voronoi cell of any node $\xi \in \Xi$. Moreover, let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ denote the n Voronoi vertices of V, labelled in a counter-clockwise ordering. For the purpose of constructing the upstream cell U corresponding to V, we first approximate the upstream positions of the Voronoi vertices $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of V. This is accomplished in a similar way explained in Section 1.3. However, instead of tracing backwards the node $\xi \in \Xi$ itself, we now trace backwards the n Voronoi vertices of the it's Voronoi cell V.

Let $\mathbf{u}^-$ be the exact *upstream position* of any vertex $\mathbf{v}$ of V satisfying $u(t, \mathbf{u}^-) = u(t + \tau, \mathbf{v})$. The upstream position $\mathbf{u}^-$ of $\mathbf{v}$ can be viewed as the location of a flow particle at time t, which by traversing along its trajectory arrives at the vertex $\mathbf{v}$ at time $t + \tau$, see Figure 2.2. In case of passive advection, the shape of the particles' flow trajectories are entirely and uniquely determined by the given velocity field $a = a(t, x)$. Note that the exact upstream position $\mathbf{u}^-$ of $\mathbf{v}$ is usually unknown. In order to compute an approximation $\mathbf{u}$ to the upstream point $\mathbf{u}^-$, this amounts to numerically

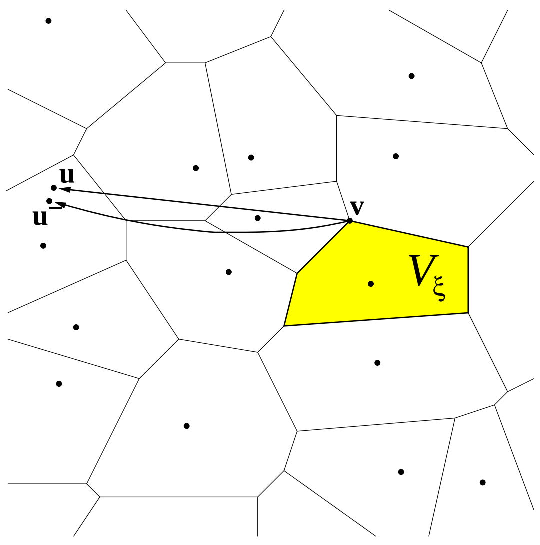


Figure 2.2: The backward trajectory from the Voronoi vertex $\mathbf{v} \in V_\xi$ to its upstream point $\mathbf{u}^-$, and its linear approximation, leading to $\mathbf{u}$.

solving the ODE

$$\dot{x} = \frac{dx}{dt} = a(t, x), \tag{2.4}$$

with initial condition $x(t + \tau) = \mathbf{v}$, so that the solution x of the initial value problem satisfies $x(t) = \mathbf{u}^-$, cf. (1.9).
Adopting some standard notation from dynamic systems [23], we express the upstream position $\mathbf{u}^-$ of the vertex $\mathbf{v}$ as

$$\mathbf{u}^- = \Phi^{t,t+\tau}\mathbf{v}, \tag{2.5}$$

where $\Phi^{t,t+\tau} : \Omega \to \Omega$ denotes the *continuous evolution* of the (backward) flow of (2.4). An equivalent formulation for (2.5) is given by $\mathbf{v} = \Phi^{t+\tau,t}\mathbf{u}^-$, since $\Phi^{t+\tau,t}$ is the inverse of $\Phi^{t,t+\tau}$.
Likewise, for the sake of notational simplicity in the following of this text, it is convenient to express the approximation $\mathbf{u}$ of $\mathbf{u}^-$ as

$$\mathbf{u} = \Psi^{t,t+\tau}\mathbf{v}, \tag{2.6}$$

where $\Psi^{t,t+\tau} : \Omega \to \Omega$ is the *discrete evolution* of the flow. Note that the operator $\Psi^{t,t+\tau}$ is usually given by any suitable numerical method for solving the above ODE (2.4). This, however, is only a generic definition for $\Psi^{t,t+\tau}$.
In order to be more concrete on the upstream point approximation, we remark that our implementation works with the fixed point iteration

$$\beta^{(k+1)} = \tau \cdot a\left(t + \frac{\tau}{2}, \mathbf{v} - \frac{\beta^{(k)}}{2}\right), \qquad k = 0, 1, 2, \ldots, \tag{2.7}$$

where we let $\beta^{(0)} = 0$, cf. (1.10). This yields after merely a few iterations a sufficiently accurate linear approximation β of the backward trajectory at $\mathbf{v}$. In this case, the desired approximation to the upstream point $\mathbf{u}^-$ is given by

$$\mathbf{u} = \mathbf{v} - \beta,$$

see Figure 2.2, cf. Figure 2.1. We remark that the iteration (2.7) has already been recommended in the seminal paper on semi-Lagrangian methods by Robert [70], see also [57, equation (7.66a)].

2.3.2 Construction of Upstream Cells

Recall the generic notation of the discrete evolution $\Psi^{t,t+\tau}$ in (2.6). Note that for any Voronoi vertex $\mathbf{v}_j$, $1 \leq j \leq n$, of a cell $V \in \mathcal{V}_\Xi$ we obtain an approximation to its corresponding upstream point by

$$\mathbf{u}_j = \Psi^{t,t+\tau}\mathbf{v}_j, \qquad 1 \leq j \leq n.$$

By connecting the sequence $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n, \mathbf{u}_1$ of consecutive upstream point approximations we obtain a closed polygon, denoted by U. This defines the *upstream cell* of the (bounded) Voronoi cell V. Again, for the sake of notational simplicity, it is convenient to use the notation

$$U = \Psi^{t,t+\tau}V, \qquad \text{for } V \in \mathcal{V}_\Xi. \tag{2.8}$$

Figure 2.3 shows an example for an upstream cell $U \equiv U_\xi$, corresponding to a Voronoi cell $V \equiv V_\xi$, $\xi \in \Xi$.
In order to be able to facilitate the following computations, we require that every upstream cell U is *convex* and *nondegenerate* (details on this are explained in Subsection 2.4.2). We achieve this by introducing a time step restriction on τ, similar to the one suggested in [64, 75]. To be more precise, we first select one initial value $\tau_0 > 0$, which is gradually reduced by letting $\tau_{k+1} = \tau_k/2$, $k \geq 0$, whenever at least one non-convex or degenerate upstream cell occurs. In our numerical experiments, where we let $\tau_0 = 0.1$, this iteration requires only at most four steps. Note that by using this time step restriction, the duality relation (2.8) is well-defined.
We remark that our condition on the time step τ is, when compared with Eulerian schemes, much less restrictive. Indeed, the stability of explicit Eulerian methods is typically dominated by rather restrictive CFL conditions, which in turn leads to very small time steps, especially when working with adaptive meshes containing very small finest control volumes [50].
According to the construction of the upstream cells, and in view of mass conservation, any upstream cell U is supposed to contain the proportion of mass, which is to be advected into its corresponding Voronoi cell $V = \Psi^{t+\tau,t}U$ at the

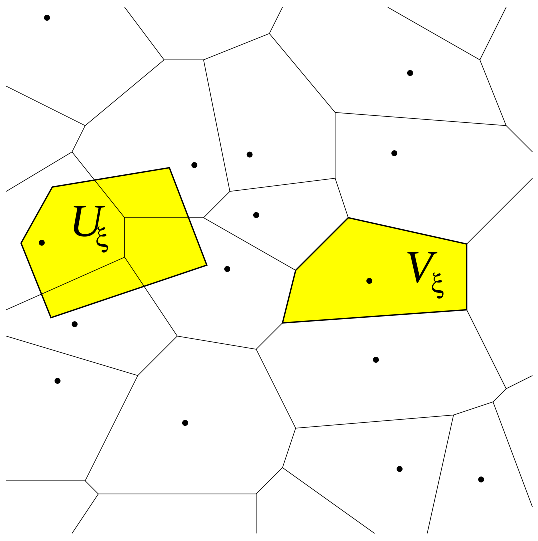


Figure 2.3: Voronoi cell V_ξ of the node $\xi \in \Xi$, and its upstream cell U_ξ.

(current) time step $t \to t + \tau$. This requirement is accommodated by (2.3). Yet it remains to determine the *total mass*

$$m_U \equiv m_U(t) = \int_U u_h(t,x)\,dx$$

contained in any single upstream cell U, at time t, from the given mass distribution in the current Voronoi diagram $\mathcal{V}_\Xi$. To this end, we compute the intersections of U with its overlapping Voronoi cells in $\mathcal{V}_\Xi$. Further details on this are discussed in the following section.

2.4 Mass Conservation by Construction

Given the current Voronoi diagram $\mathcal{V}_\Xi$, at time t, let $\mathcal{U}_\Xi = \{U_\xi\}_{\xi\in\Xi}$ denote the corresponding set of current upstream cells, each of which is given by the duality relation (2.8). Note that the collection $\mathcal{U}_\Xi$ of upstream cells yields (besides the Voronoi diagram $\mathcal{V}_\Xi$) yet another partitioning of the plane, so that we have

$$\mathbb{R}^2 = \bigcup_{\xi\in\Xi} U_\xi.$$

We make use of the duality (2.8) between the Voronoi cells and the upstream cells in order to design an advection scheme which is *mass conservative by construction.*

2.4.1 Reconstruction from Cell Average Values

The modelling taken in this approach works with piecewise linear functions over the Voronoi diagram $\mathcal{V}_\Xi$, so that we obtain a *second order* finite volume scheme. In order to be more precise on this, the current approximation u_h to the solution u of the Cauchy problem (2.1), (2.2) is, at any time t, an element of the linear function space

$$\mathcal{S}_1(\mathcal{V}_\Xi) = \{u_h : \Omega \to \mathbb{R} : u_h|_{V\cap\Omega} \text{ linear; for all } V \in \mathcal{V}_\Xi\}$$

containing all piecewise linear functions on $\mathcal{V}_\Xi \cap \Omega$.
Starting point for computing the numerical solution $u_h(t,\cdot) \in \mathcal{S}_1(\mathcal{V}_\Xi)$, at a time t, are known *cell average values*

$$\bar{u}_V(t) \approx \frac{1}{|V|}\int_V u(t,x)\,dx, \qquad \text{for all } V \in \mathcal{V}_\Xi,$$

over the current Voronoi diagram $\mathcal{V}_\Xi$, where $|V|$ denotes the volume of V in $\mathbb{R}^2$. Initially, for a *suitable* set Ξ of nodes, we let $\bar{u}_{V_\xi}(0) = u_0(\xi)$ for all $\xi \in \Xi$, by using the initial condition (2.2).
Given $\{\bar{u}_V(t)\}_{V\in\mathcal{V}_\Xi}$, we wish to determine $u_h(t,\cdot) \in \mathcal{S}_1(\mathcal{V}_\Xi)$ from the cell averages, such that

$$\bar{u}_V(t) = \frac{1}{|V|}\int_V u_h(t,x)\,dx, \qquad \text{for all } V \in \mathcal{V}_\Xi, \tag{2.9}$$

holds. This is accomplished as follows.
For any Voronoi cell $V_\xi \in \mathcal{V}_\Xi$, we first determine the best approximation $u^* \in \mathcal{P}_1$ satisfying

$$\min_{u\in\mathcal{P}_1} \sum_{\nu\in\mathcal{N}} |\bar{u}_\nu - u(\nu)|^2 = \sum_{\nu\in\mathcal{N}} |\bar{u}_\nu - u^*(\nu)|^2.$$

Here, $\mathcal{P}_1$ is the space of all linear bivariate polynomials, and $\mathcal{N}$ denotes the set of Voronoi neighbours of ξ. Hence, the function u^* is the *least squares fit* of the cell average values $\bar{u}_\nu$, $\nu \in \mathcal{N}$, in the neighbourhood of V_ξ.
Next, we determine a constant c such that the function $u_h = u^* + c \in \mathcal{P}_1$ satisfies (2.9). This is achieved by letting

$$c = \frac{1}{|V|}\cdot\left(m_V - \int_V u^*\,dx\right), \tag{2.10}$$

where $m_V = |V|\cdot\bar{u}_V(t)$ is the total mass in the cell V. In other words, the constant c in (2.10) adjusts the linear least square fit u^* in order to guarantee mass conservation (2.9) locally.

Additionally, we constrain the slopes of the linear approximation u_h by requiring the two conditions

$$\begin{array}{rcl} \min_{x\in V} u_h(x) & \geq & \min_{\nu\in\mathcal{N}} \bar{u}_\nu \\ \max_{x\in V} u_h(x) & \leq & \max_{\nu\in\mathcal{N}} \bar{u}_\nu, \end{array} \tag{2.11}$$

which can be viewed as a *slope limiter.* Slope limiters are typically employed in TVD (Total Variation Diminishing) schemes in order to avoid spurious oscillations of the solution. In order to match the restrictions (2.11), this needs merely a local correction of u_h, so that the resulting modification of u_h continues to satisfy (2.9). To this end, we follow along the lines of the construction in [82].

2.4.2 Conservative Advection of Cell Average Values

Having computed $u_h(t,\cdot)$ from the current cell averages $\{\bar{u}_V(t)\}_{V\in\mathcal{V}_\Xi}$, we are in a position to compute, for any upstream cell U, its total mass

$$m_U = \int_U u_h(t,x)\,dx$$

which is advected into the corresponding Voronoi cell $V = \Psi^{t+\tau,t}U$. To this end, we first decompose U into smaller tiles by computing the intersections between U and its overlapping Voronoi cells in $\mathcal{V}_\Xi$ (see Figure 2.4).
In order to be able to facilitate the computation of these intersections, the upstream cell U is supposed to be *convex*, which explains the time step restriction suggested in the previous Section 2.3.2. With assuming convexity for U, this namely allows us to use the efficient intersection algorithm proposed in [63]. The algorithm in [63] requires merely $\mathcal{O}(p+q)$ operations for the intersection of two *convex* polygons with p and q vertices, respectively, and is explained in Appendix A. In contrast, computing the intersection of two *non-convex* polygons would cost $\mathcal{O}(p\cdot q)$ operations. For further details on this, see [63].
Now, the intersections between U and the cells in the Voronoi diagram $\mathcal{V}_\Xi$ yield by

$$U = \bigcup_{V\in\mathcal{V}_\Xi} (U\cap V)$$

a partitioning of U into merely a *small* number of tiles, which requires only local computations. Indeed, this tiling for U is given by all *non-empty* intersections between U and Voronoi cells in $\mathcal{V}_\Xi$. Figure 2.4 shows one example, where one upstream cell U is decomposed into four tiles, U_1, U_2, U_3, U_4, so that $U = U_1 \cup U_2 \cup U_3 \cup U_4$.
The current approximation $u_h(t,\cdot) \in \mathcal{S}_1(\mathcal{V}_\Xi)$ to the solution u in (2.1) is now used in order to determine the total mass m_U in U. This is done as follows.

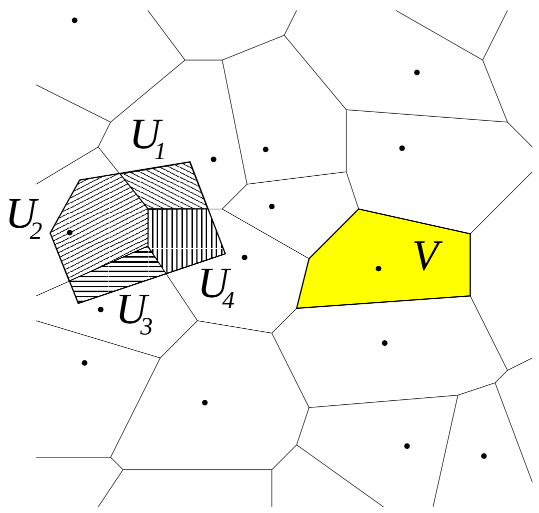


Figure 2.4: The decomposition of an upstream cell U into four tiles, U_1, U_2, U_3 and U_4.

Note that the restriction of $u_h(t, \cdot)$ to any Voronoi cell $V \in \mathcal{V}_\Xi$ is a linear function by definition. So, in particular the restriction of $u_h(t, \cdot)$ to any non-empty tile $U \cap V$ is linear. This allows us to compute, for any $V \in \mathcal{V}_\Xi$, at time t, the current mass

$$m_{U\cap V} = \int_{U\cap V} u_h(t, x)\, dx, \qquad \text{for } V \in \mathcal{V}_\Xi, \tag{2.12}$$

of the linear function u_h over the tile $U \cap V$ exactly. If $U \cap V$ is empty, then we have $m_{U\cap V} = 0$. This in turn yields the total mass

$$m_U = \sum_{V \in \mathcal{V}_\Xi} m_{U\cap V} \tag{2.13}$$

in the upstream cell U at time t. In the advection step $t \to t + \tau$, the total mass m_U is advected from the upstream cell U into the Voronoi cell V. We establish this *local mass conservation* by replacing the current cell average values $\bar{u}_V(t)$ in (2.9) by the updates

$$\bar{u}_V(t + \tau) = \frac{1}{|V|}\, m_U, \qquad \text{for all } V \in \mathcal{V}_\Xi. \tag{2.14}$$

Altogether, we obtain a *conservative* semi-Lagrangian advection scheme, each of whose time steps $t \to t + \tau$ is given by the following algorithm.

Algorithm 1 (Conservative Semi-Lagrangian Advection).

```
INPUT: Time step size $\tau > 0$, node set $\Xi \subset \Omega$, Voronoi diagram $\mathcal{V}_\Xi$, and
cell average values $\bar{u}_V(t)$ for all $V \in \mathcal{V}_\Xi$.

  • Compute piecewise linear $u_h(t, \cdot) \in \mathcal{S}_1(\mathcal{V}_\Xi)$ satisfying (2.9) and (2.11).
  • FOR each $V \in \mathcal{V}_\Xi$ DO
      – Compute upstream cell $U = \Psi^{t,t+\tau} V$;
      – Compute the total mass $m_U$ over the upstream cell via (2.12),
        (2.13);
      – Update cell average values $\bar{u}_V(t+\tau)$ using (2.14);

OUTPUT: Updated cell average values $\bar{u}_V(t+\tau)$ for all $V \in \mathcal{V}_\Xi$.
```

2.4.3 Implementation of Boundary Conditions

Now let us finally turn to the implementation of boundary conditions. In the above discussion until now, we have considered the special case where the computational domain Ω is the whole plane, i.e., $\Omega = \mathbb{R}^2$. However, in specific applications of interest, Ω is bounded, and, moreover, boundary conditions are of relevance. Therefore, suppose from now that Ω is bounded. Recall that our proposed scheme works with a partitioning of the plane by using Voronoi cells, given by the Voronoi diagram $\mathcal{V}_\Xi$ of the current node set Ξ. In the situation of a bounded domain $\Omega \subset \mathbb{R}^2$, we work with restricted Voronoi cells of the form $\tilde{V} = V \cap \Omega$, which yields a decomposition of Ω by

$$\Omega = \bigcup_{V \in \mathcal{V}_\Xi} \tilde{V}.$$

Note that $\tilde{V} = V$ for $V \subset \Omega$, so that this restriction is only relevant for Voronoi cells which intersect the boundary $\partial\Omega$ of the domain Ω, in which case $\tilde{V} \neq V$ (see Figure 2.5 for illustration). Now the collection $\{\tilde{U}_\xi\}_{\xi \in \Xi}$ of all upstream cells, $\tilde{U}_\xi = \Psi^{t,t+\tau} \tilde{V}_\xi$, yields by their union

$$\Omega^- = \bigcup_{\xi \in \Xi} \tilde{U}_\xi$$

an *upstream domain* $\Omega^- \subset \mathbb{R}^2$. This gives rise to define the two sets

$$\Omega_{\text{in}} = \Omega^- \setminus \Omega \quad \text{and} \quad \Omega_{\text{out}} = \Omega \setminus \Omega^-.$$

The set Ω_{in} corresponds to a region outside of Ω through which mass is advected into Ω across its boundary $\partial\Omega$, whereas Ω_{out} contains the mass

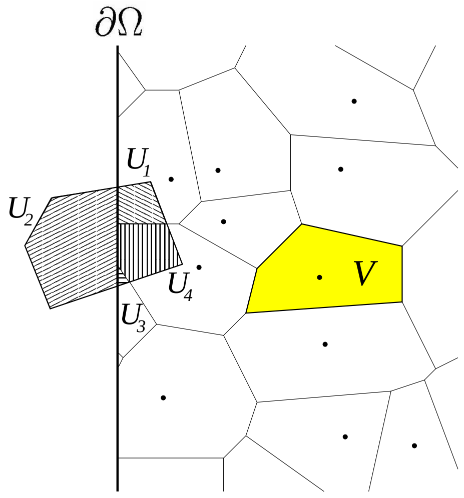


Figure 2.5: The upstream cell U of the Voronoi cell V intersects the boundary $\partial\Omega$ of the domain Ω. A boundary value m_{U_2} is assigned to the tile U_2.

which is advected from Ω to the exterior of Ω across $\partial\Omega$. More precisely, we can explain this as follows.
Note that an upstream cell U may partly or entirely lie outside the domain Ω. Figure 2.5 shows an upstream cell U, which intersects the boundary $\partial\Omega$. This leads to a tiling of U as before, but where at least one tile lies entirely outside of the domain Ω. In the situation of Figure 2.5, this is the tile U_2. Now in order to implement boundary conditions concerning the *incoming flow*, we assign a boundary value to each such tile of Ω_{in}, such as $U_2 \in \Omega_{\text{in}}$. This boundary value, say m_{U_2}, determines the mass which is advected through the tile U_2 into the domain Ω.
As regards *outgoing flow*, we remark that the upstream domain Ω^- may not cover the entire domain Ω, in which case the set Ω_{out} is non-empty. The mass contained in Ω_{out} is advected into regions outside of the domain Ω.

2.5 Adaption Rules

One important feature of our advection schemes is its adaptivity. Adaptivity requires the modification of the node set Ξ after each time step $t \to t+\tau$ of the above Algorithm 1. This is in order to be able to balance the two conflicting requirements of good approximation quality and small computational costs. In fact, for the sake of reducing the computational complexity we wish to reduce the size of the node set Ξ, whereas for the sake of good approximation quality we prefer to increase the density (and thus the size) of the node set Ξ in Ω.

We have combined the proposed conservative advection scheme with the ideas of the adaption strategy discussed in Section 1.5 and in [9, 10, 11], in order to obtain an *adaptive* and *conservative* semi-Lagrangian advection method. In the following, we briefly review the basic ideas of the adaption strategy and point out the modifications required to guarantee a conservative scheme.

2.5.1 Error Indication

As described in Section 1.5 we use a customized error indicator in order to adaptively modify the node set Ξ. A *significance* value $\eta(\xi)$ for each $\xi \in \Xi$ is required to reflect the local approximation quality of the cell average $\bar{u}_{V_\xi}(t)$ around $\xi \in \Xi$. These significances $\eta(\xi)$, $\xi \in \Xi$, are again used in order to flag single nodes $\xi \in \Xi$ as "to be refined" or "to be coarsened".
Definition 1 in Section 1.5 can be carried forward to our *conservative* advection scheme, whereas the relative *tolerance values* θ_{crs} and θ_{ref} are slightly modified. To be precise, we let $\theta_{\text{crs}} = 0.05$ and $\theta_{\text{ref}} = 0.2$ in our numerical experiments.
Following along the lines of [35], the error indicator is now given by

$$\eta(\xi) = |\bar{u}_{V_\xi}(t) - s(\xi)|, \tag{2.15}$$

where for a set $\mathcal{N} \subset \Xi \setminus \{\xi\}$ of neighbouring nodes of ξ, the *thin plate spline* interpolant $s \equiv s_{\mathcal{N}}$ in (2.15), satisfying the interpolation conditions $s(\nu) = \bar{u}_{V_\nu}(t)$ for all $\nu \in \mathcal{N}$, is of the form

$$s = \sum_{\nu \in \mathcal{N}} c_\nu \| \cdot - \nu \|^2 \log(\| \cdot - \nu \|) + p$$

(cf. Section 1.4). Here, p is a linear polynomial in two variables and $\| \cdot \|$ denotes the Euclidean norm. For more details concerning thin plate spline interpolation, due to Duchon [25], and related interpolation methods, the reader is referred to the recent tutorial [42].
Hence, the thin plate spline interpolant s in (2.15) matches current cell average values of $\bar{u}_{V_\nu}(t)$ in the neighbourhood of the Voronoi cell V_ξ, but not at V_ξ itself, i.e., we have $\bar{u}_{V_\xi}(t) \neq s(\xi)$ in general. Now the error indication $\eta(\xi)$ for the node ξ is small whenever the reproduction quality of $\bar{u}$ by s around ξ is good. In contrast, a high value of $\eta(\xi)$ typically indicates that $\bar{u}$ is subject to strong variation locally around ξ. Indeed, this observation relies on available local error estimates for thin plate spline interpolation (see the corresponding discussion on this in [9, 11, 10]). We remark that the error indicator allows us to locate discontinuities of the solution u quite effectively. This is supported by the numerical results in the following Section 2.6.

2.5.2 Coarsening and Refinement

Similarly to the adaption rules of the meshless advection scheme discussed in Section 1.5.2, we can balance the approximation quality of the solution against the required computational complexity by inserting new nodes into regions where the value of η is high (refinement), whereas we remove nodes from Ξ in regions where the value of η is small (coarsening).

Coarsening. A node $\xi \in \Xi$ is *coarsened* by its removal from the current node set Ξ. I.e., in this case we let $\Xi = \Xi \setminus \xi$, and the Voronoi diagram $\mathcal{V}_\Xi$ is updated accordingly in order to obtain the modified Voronoi diagram $\mathcal{V}_{\Xi\setminus\xi}$.

Refinement. A node $\xi \in \Xi$ is *refined* by the insertion of the n Voronoi vertices $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of its corresponding Voronoi cell V_ξ. I.e., in this case we let $\Xi = \Xi \cup \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, and the current Voronoi diagram $\mathcal{V}_\Xi$ is updated accordingly.

2.6 Numerical Results

In this section, the performance of our advection scheme is evaluated by using two numerical experiments. In the first experiment, the accuracy and convergence is analyzed. This is done by considering the test case suggested by Phillips & Williams in [64], which allows us to compare our numerical results with those in [64] directly. In the second experiment, we apply our advection scheme to the *slotted cylinder*, a well-known test case suggested by Zalesak [89]. This illustrates the efficacy of the chosen adaption strategy, and, moreover, it confirms that our method is conservative.
The numerical experiments were prepared on a personal computer, model `IBM 236623G` (Genuintel Pentium(R) 4 1600MHz processor, 256MB physical memory). The algorithms were implemented by using `MATLAB`, Version 6.5, Release 13.

2.6.1 Experiment 1

According to the numerical experiment suggested in [64], we consider solving the hyperbolic equation (2.1) on the computational domain $\Omega = [1,2]^2 \subset \mathbb{R}^2$. As shown in Figure 2.6 **(a)**, we let $a(x) = (x_1, -x_2)$, $x = (x_1, x_2)$, for the velocity field. The initial condition suggested in [64] is given by

$$u(0,x) = 0, \qquad \text{for } x \in (1,2] \times [1,2).$$

For the boundary conditions at the two inflow boundaries, we let

$$\begin{aligned} u(t,x) &= 1 + x_2^2, && \text{for } x_1 = 1,\ x_2 \in [1,2],\ t \geq 0, \\ u(t,x) &= 1 + 4x_1^2, && \text{for } x_2 = 2,\ x_1 \in [1,2],\ t \geq 0. \end{aligned} \tag{2.16}$$

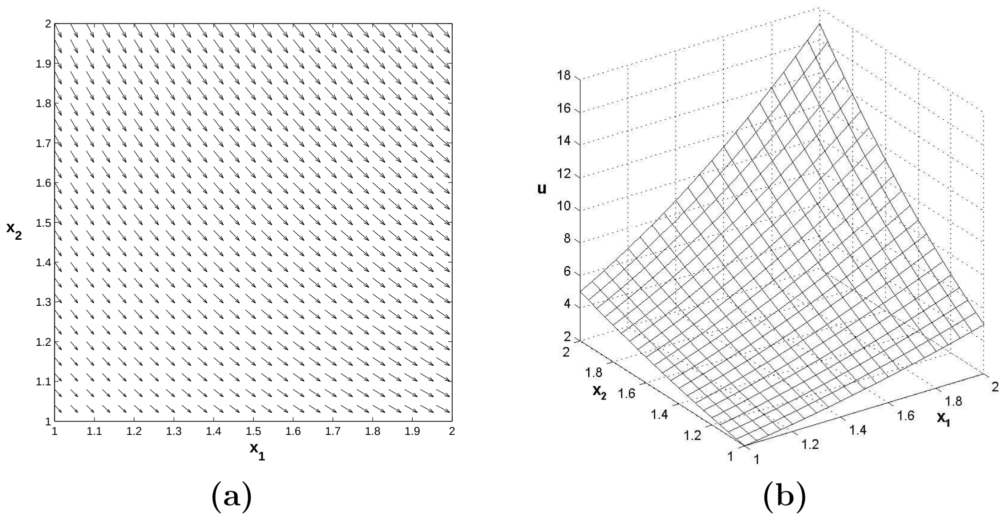


Figure 2.6: **(a)** Velocity field and **(b)** steady state solution u on Ω.

This condition (2.16) determines the boundary values of the incoming flow. We remark that in this model problem, the solution of (1.4) converges to the *steady state* solution

$$u(x) = 1 + (x_1 x_2)^2, \qquad x \in \Omega, \tag{2.17}$$

as displayed in Figure 2.6 **(b)**. In order to compare our results with those of Phillips & Williams [64], we proceed as follows.
We first create a nested sequence of four different node sets $\Xi_h \subset \Omega$, such that each corresponding Voronoi diagram $\mathcal{V}_{\Xi_h}$ yields a regular mesh with mesh size h. The sequence of these four meshes is displayed in Figure 2.7. In order to make a fair comparison with the results in [64], we keep the node set Ξ_h fixed throughout each simulation. Moreover, we also let $\tau = 0.01$ fixed, which leads to convex and nondegenerate upstream cells during the simulation.
According to [64], the simulation terminates, as soon as the stopping criterion

$$\frac{\|\bar{u}_V(t+\tau) - \bar{u}_V(t)\|_\infty}{\tau} \leq 10^{-5}, \qquad \text{for all } V \in \mathcal{V}_{\Xi_h}, \tag{2.18}$$

is satisfied. In this case, the numerical solution has reached the *steady state* approximatively.
In order to measure accuracy and convergence order, let $\tilde{u}_h \approx u$ denote this final approximation to the *steady state* u in (2.17), on a mesh of width h. The accuracy of $\tilde{u}_h$ is measured by using the relative error

$$E_p(h) = \frac{\|u - \tilde{u}_h\|_p}{\|\tilde{u}_h\|_p}, \tag{2.19}$$

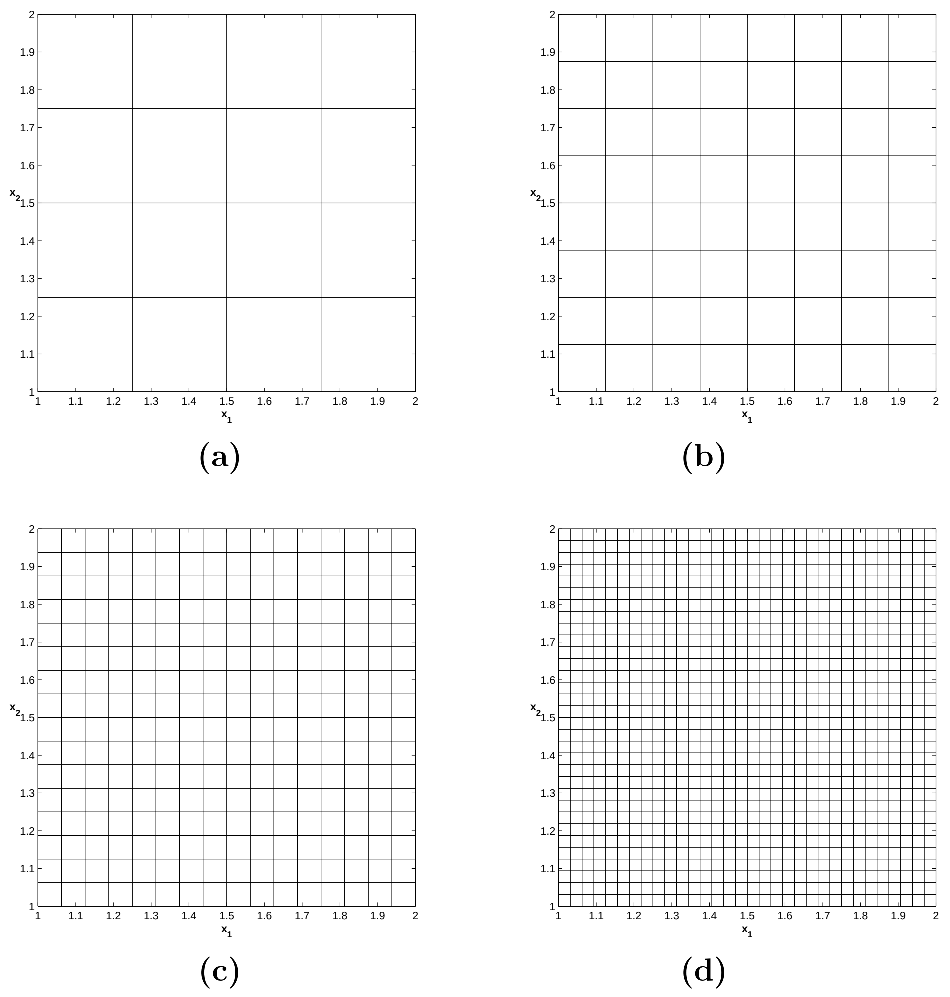


Figure 2.7: A sequence of four regular Voronoi diagrams $\mathcal{V}_{\Xi_h}$ with mesh widths **(a)** $h = 0.25$, **(b)** $h = 0.125$, **(c)** $h = 0.0625$, **(d)** $h = 0.03125$.

where we let $p = 1, 2$ or $p = \infty$ for the corresponding norm. Furthermore, the expression

$$k_p = \frac{\log\Big(E_p(h)\,/\,E_p(h/2)\Big)}{\log(2)}\,, \tag{2.20}$$

yields an estimate for the convergence order, where $E_p(h)$ and $E_p(h/2)$ are the errors (w.r.t. the selected norm $p = 1, 2, \infty$) observed on two subsequent meshes (see Figure 2.7).

Table 2.1 shows the dependence of the error $E_p(h)$ in (2.19) on the mesh width h for the norms $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$, together with the corresponding convergence orders k_p in (2.20). The results agree very well with those of Phillips & Williams in [64]. In fact, our scheme reaches the expected order of 2 in all three norms. In [64] four different schemes, denoted as **A**, **B**, **C**,

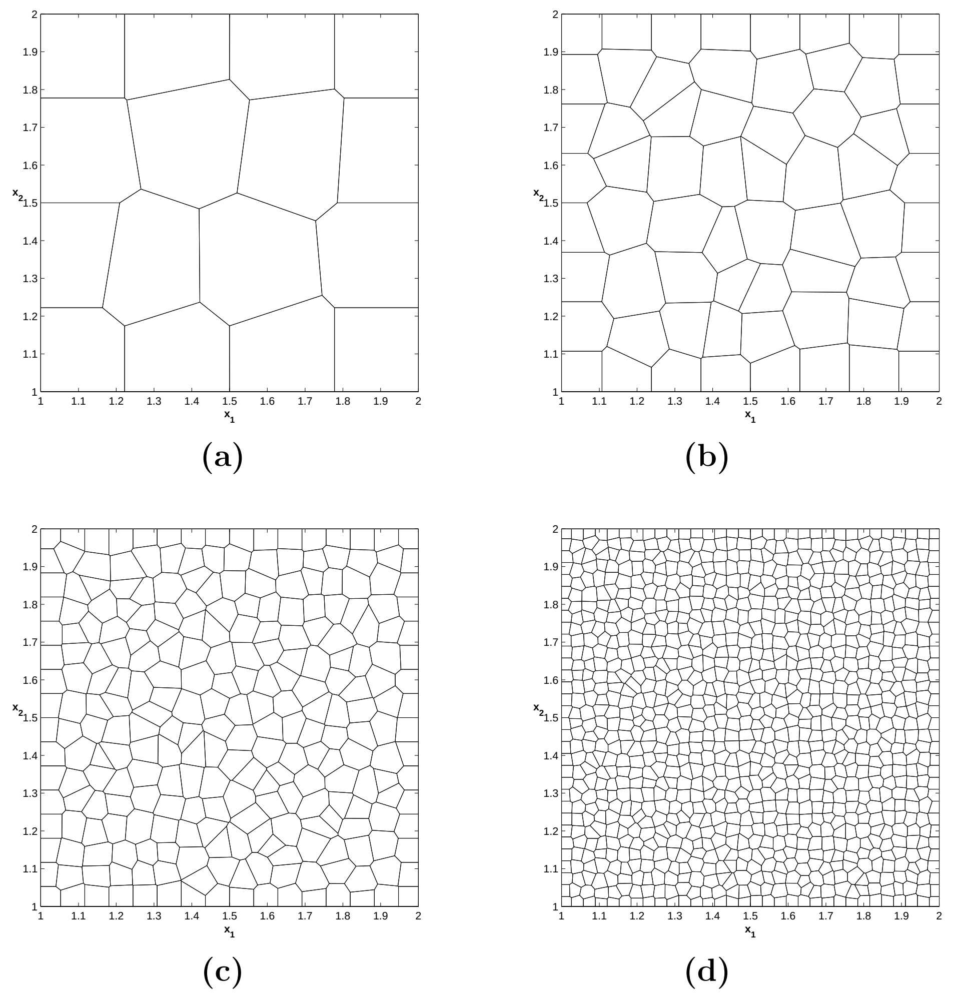


Figure 2.8: A sequence of four irregular Voronoi diagrams $\mathcal{V}_{\Xi_h}$ with mesh widths **(a)** $h = 0.25$, **(b)** $h = 0.125$, **(c)** $h = 0.0625$, **(d)** $h = 0.03125$.

and **D**, are introduced. In comparison with the schemes **A**, **C**, and **D**, our method is more accurate. We can explain this as follows. Firstly, in contrast to the schemes **A** and **C** in [64], our scheme is based on a *centered* reconstruction stencil, i.e., on each cell V_ξ we use all *Voronoi neighbours* of the node ξ in order to reconstruct the piecewise linear u_h. Secondly, our slope limiter in (2.11) is less restrictive than the one used in scheme **D** of [64].
But the scheme **B** in [64] yields more accurate results in terms of approximation errors and convergence orders. This is because centered differences without any slope limiter are used in the scheme **B** of [64]. The absence of the slope limiter, however, often leads to an oscillatory solution u_h, which is considered to be a severe drawback in many relevant applications.
In the following of this experiment, we now investigate the influence of mesh

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
0.25	$3.962 \cdot 10^{-2}$	-	$4.653 \cdot 10^{-2}$	-	$7.230 \cdot 10^{-2}$	-
0.125	$9.598 \cdot 10^{-3}$	2.05	$1.178 \cdot 10^{-2}$	1.98	$2.441 \cdot 10^{-2}$	1.57
0.0625	$1.872 \cdot 10^{-3}$	2.36	$2.228 \cdot 10^{-3}$	2.37	$6.471 \cdot 10^{-3}$	1.91
0.03125	$2.789 \cdot 10^{-4}$	2.74	$3.391 \cdot 10^{-4}$	2.75	$1.452 \cdot 10^{-3}$	2.16

Table 2.1: Convergence results for *regular* Voronoi diagrams $\mathcal{V}_{\Xi_h}$ (Figure 2.7).

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
0.25	$1.999 \cdot 10^{-2}$	-	$2.749 \cdot 10^{-2}$	-	$4.734 \cdot 10^{-2}$	-
0.125	$7.308 \cdot 10^{-3}$	1.45	$9.569 \cdot 10^{-3}$	1.52	$1.706 \cdot 10^{-2}$	1.47
0.0625	$2.065 \cdot 10^{-3}$	1.82	$2.483 \cdot 10^{-3}$	1.95	$5.362 \cdot 10^{-3}$	1.67
0.03125	$5.491 \cdot 10^{-4}$	1.91	$6.432 \cdot 10^{-4}$	1.95	$1.469 \cdot 10^{-3}$	1.87

Table 2.2: Convergence results for *irregular* Voronoi diagrams $\mathcal{V}_{\Xi_h}$ (Figure 2.8).

irregularity on the accuracy of our scheme. To this end, we work with a sequence of four irregular meshes $\mathcal{V}_{\Xi_h}$ (displayed in Figure 2.8) comprising the same number $\#\Xi_h$ of cells as in the regular case (see Table 2.3). In this irregular case, h is considered as a measure of an average mesh width. Table 2.2 shows the approximation errors and convergence orders which were obtained by using the same model problem as discussed above, but with using the mesh sequence in Figure 2.8 instead of the one in Figure 2.7. We observe that the errors $E_p(h)$ are of the same magnitude as in the regular case, but the convergence orders k_p are slightly lower.

We remark that a higher number of time steps, and thus more CPU seconds, are necessary for the irregular meshes in order to reach the steady state $\tilde{u}_h$ satisfying the stopping criterion (2.18). This is shown in Table 2.3. Note also from Table 2.3 that the advection scheme converges with a fewer number of time steps (for both cases, regular and irregular), as the mesh width h is reduced. This confirms the utility of semi-Lagrangian schemes. In contrast, Eulerian schemes typically require smaller time steps when the mesh is refined, which in turn leads to an increasing number of necessary time steps.

In order to investigate, how much CPU time is used by the different parts of the proposed semi-Lagrangian scheme (see Algorithm 1 in Section 2.4.2), we register the CPU seconds needed to compute one time step on the finest regular and irregular mesh ($h = 0.03125$, $\#\Xi = 1024$). The results are presented in Table 2.4, where t_r is the average time used to compute the piecewise linear reconstruction function u_h, t_u is the average time used to compute the upstream cells U_ξ, $\xi \in \Xi$, and t_m is the time used to compute

		regular mesh		irregular mesh	
h	$\#\Xi_h$	time steps	CPU seconds	time steps	CPU seconds
0.25	16	189	1.442	202	1.926
0.125	64	129	4.356	157	6.520
0.0625	256	108	12.016	125	17.245
0.03125	1024	95	30.424	99	39.314

Table 2.3: Time steps and CPU seconds required to reach the *steady state* $\tilde{u}_h$.

the total mass in each upstream cell and to update the cell averages $\bar{u}_V$. We also give the percentage for each step in Algorithm 1 in Table 2.4, which clearly demonstrates, that the computation of the total mass based on the intersection of overlapping Voronoi cells uses 72% to 73% of the total CPU time t_{tot} in each time step. Table 2.4 also shows, that the CPU times for the different steps are independent of the irregularity of the mesh.

	regular mesh		irregular mesh	
	CPU seconds	%	CPU seconds	%
t_r	0.049	15	0.059	15
t_u	0.040	13	0.047	12
t_m	0.231	72	0.293	73
t_{tot}	0.320	100	0.399	100

Table 2.4: CPU seconds for the different computational steps of the conservative semi-Lagrangian scheme.

In conclusion, this numerical experiment shows that our semi-Lagrangian scheme reaches second order accuracy on both structured and unstructured Voronoi diagrams (see Tables 2.1 and 2.2). This provides high flexibility, especially in view of more complicated geometries. In the following experiment, we investigate the conservation properties of our advection scheme. Moreover, we combine the enhanced flexibility of unstructured meshes with local mesh adaption.

2.6.2 Experiment 2

In this experiment we consider the rotating *slotted cylinder*, a popular test case suggested by Zalesak [89]. Here, $\Omega = [-0.5, 0.5]^2 \subset \mathbb{R}^2$ and the initial condition is given by

$$u(0,x) = \begin{cases} 1 & \text{for } x \in D, \\ 0 & \text{otherwise,} \end{cases} \tag{2.21}$$

where $D \subset \Omega$ is the slotted disc of radius $r = 0.15$, centered at $(0, 0.25)$ with slot width 0.06 and length 0.22, see Figure 2.9 **(a)**.

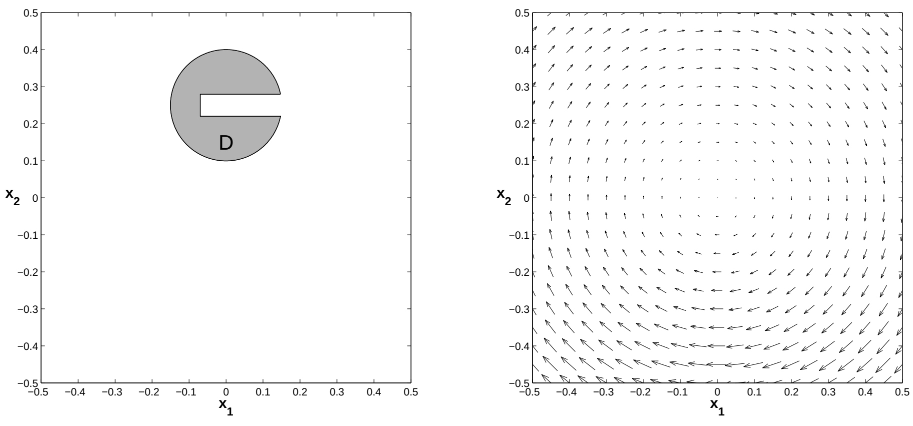


Figure 2.9: The slotted cylinder. **(a)** Initial condition and **(b)** velocity field.

In the original test case of Zalesak, the slotted cylinder is rotated by a steady flow field $a(x) \sim (x_2, -x_1)$, where $x = (x_1, x_2)$. We decided to replace this velocity field by

$$a(x) = (x_2, -x_1) \begin{cases} \frac{1}{2}\sin(2\varphi(x) - \frac{\pi}{2}) + \frac{3}{2} & \text{for } x_2 < 0, \\ 1 & \text{for } x_2 \geq 0, \end{cases} \tag{2.22}$$

whose azimuth angle is given by

$$\varphi(x) = \begin{cases} \arctan(-x_2/x_1) & \text{for } x_1 > 0, \\ \arctan(x_1/x_2) + \frac{\pi}{2} & \text{for } x_1 \leq 0. \end{cases}$$

This velocity field rotates the slotted cylinder clockwise with constant angular velocity in the first and second quadrant, whereas the cylinder is accelerated in the fourth quadrant, and decelerated in the third quadrant, see Figure 2.9 **(b)**. The maximum angular velocity $\omega = 2$ is attained in the lower half of the coordinate system, namely at the points on the vertical line

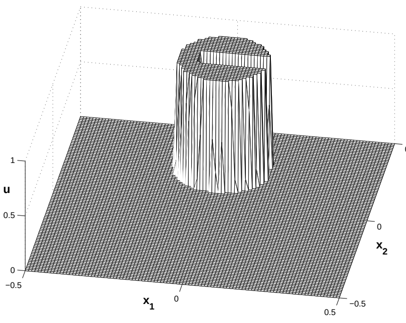
$$\{x = (x_1, x_2) \colon x_1 = 0, x_2 < 0\}.$$

The slotted cylinder is *stretched* when passing through the *acceleration part* of the velocity field in the fourth quadrant, whereas it is *squashed* in the *deceleration part* of the third quadrant in order to recover its original shape of the initial condition at each full revolution.
Initially, a set $\Xi \subset \Omega$ of 1500 randomly distributed nodes is chosen. The initial condition (2.21) is used in order to assign a cell average value $\bar{u}_V(0)$ in (2.9) to each Voronoi cell $V \in \mathcal{V}_\Xi$ of the initial Voronoi diagram $\mathcal{V}_\Xi$. The initial nodes are automatically adapted to the discontinuities of the initial condition u_0, by using the adaption strategy discussed in the previous Section 2.5, see Figures 2.10 **(b)**,**(c)**.

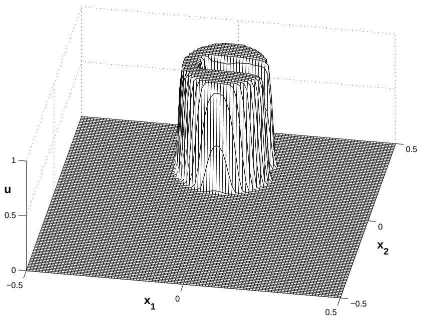
At each revolution of the slotted cylinder, the cell average values $\bar{u}_V$ are decreasing, as soon as the cylinder enters the acceleration part of the velocity field, see Figure 2.11. This behaviour is due to the mass conservation of the scheme. In contrast to this, in the deceleration part, the cell average values $\bar{u}_V$ are increasing. Moreover, in this region, the initial shape of slotted cylinder is gradually recovered, see the Figures 2.11, 2.12, and 2.13. Our simulation of this model problem comprises six full revolutions of the slotted cylinder. During the simulation, we have recorded the number of current nodes, the variation of the time step size τ, and the ratio of the first mass moment

$$\mathrm{RFM}(t) = \frac{\displaystyle\sum_{V \in \mathcal{V}_\Xi} \bar{u}_V(t) - m_{\mathrm{in}}(t) + m_{\mathrm{out}}(t)}{\displaystyle\sum_{V \in \mathcal{V}_\Xi} \bar{u}_V(0)}, \tag{2.23}$$

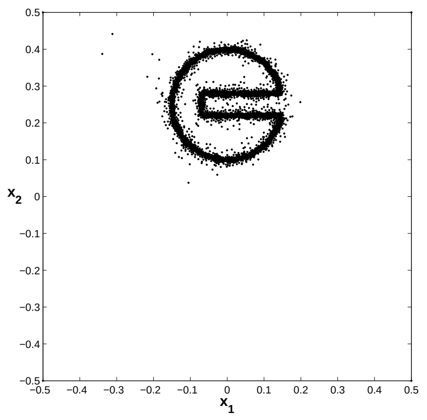
where $m_{\mathrm{in}}(t)$ is the total mass of the incoming flow and $m_{\mathrm{out}}(t)$ is the total mass of the outgoing flow, during time $[0, t]$, respectively. Our numerical results are reflected by Figure 2.14. Let us provide a few comments on the three graphs of this figure.
The number of nodes is increasing, whenever the slotted cylinder passes through the accelerating part of the velocity field (2.22). In this case, the cylinder is stretched, and so more nodes are needed in order to adaptively resolve the elongated edges of the cylinder. We remark that the moderate increase in numbers of nodes at the beginning of the simulation is due to numerical diffusion. In contrast to this, the number of nodes is decreasing, whenever the cylinder enters the decelerating part of the velocity field. In this case, the cylinder is gradually squashed back to its original shape, and so fewer nodes are needed in order to adaptively resolve the cylinder's edges. Altogether, this explains the periodic behaviour of the graph concerning the number of nodes in Figure 2.14, first row.
Figure 2.14 shows also the variation of the time step τ in its second row. As mentioned in Subsection 2.3.2, the time step size is determined, such that all upstream cells are convex. Not surprisingly, this leads to an acceleration



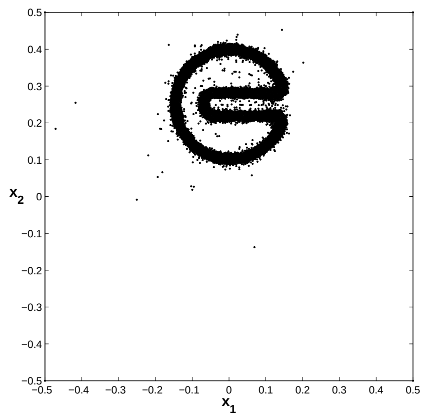
(a)



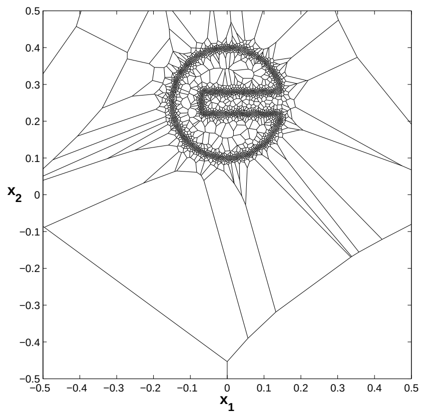
(d)



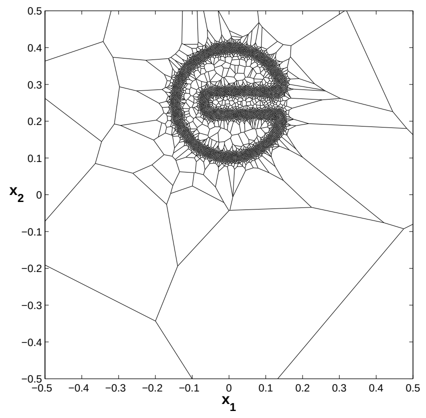
(b)



(e)

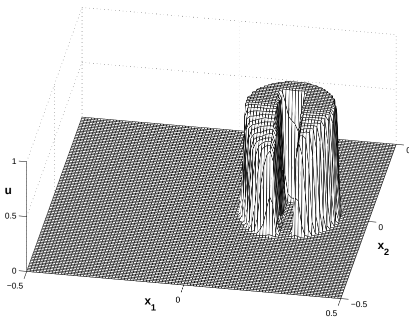


(c)

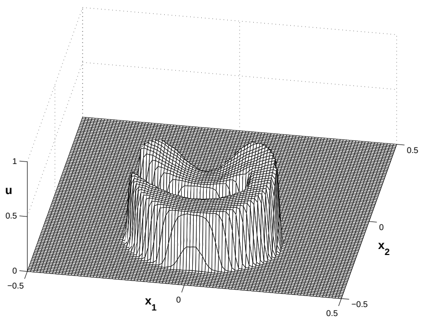


(f)

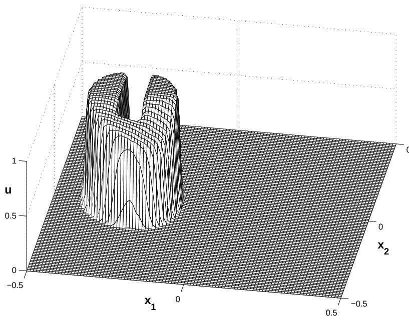
Figure 2.10: The slotted cylinder. **(a)** 3D view, **(b)** node distribution, and **(c)** Voronoi diagram of the initial condition (left column), and after six revolutions (right column), **(d)**,**(e)**,**(f)**.



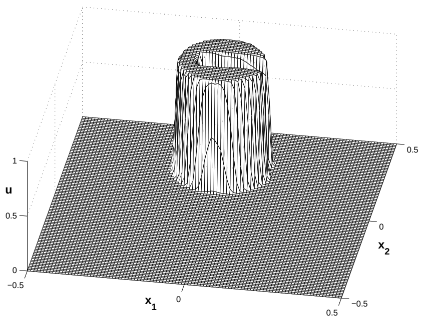
(a)



(b)



(c)



(d)

Figure 2.11: The slotted cylinder. 3D view on the evolution of $\bar{u}_V(t)$ at four different times, **(a)** $t = t_{53}$; **(b)** $t = t_{70}$; **(c)** $t = t_{102}$; **(d)** $t = t_{180}$, during the first revolution.

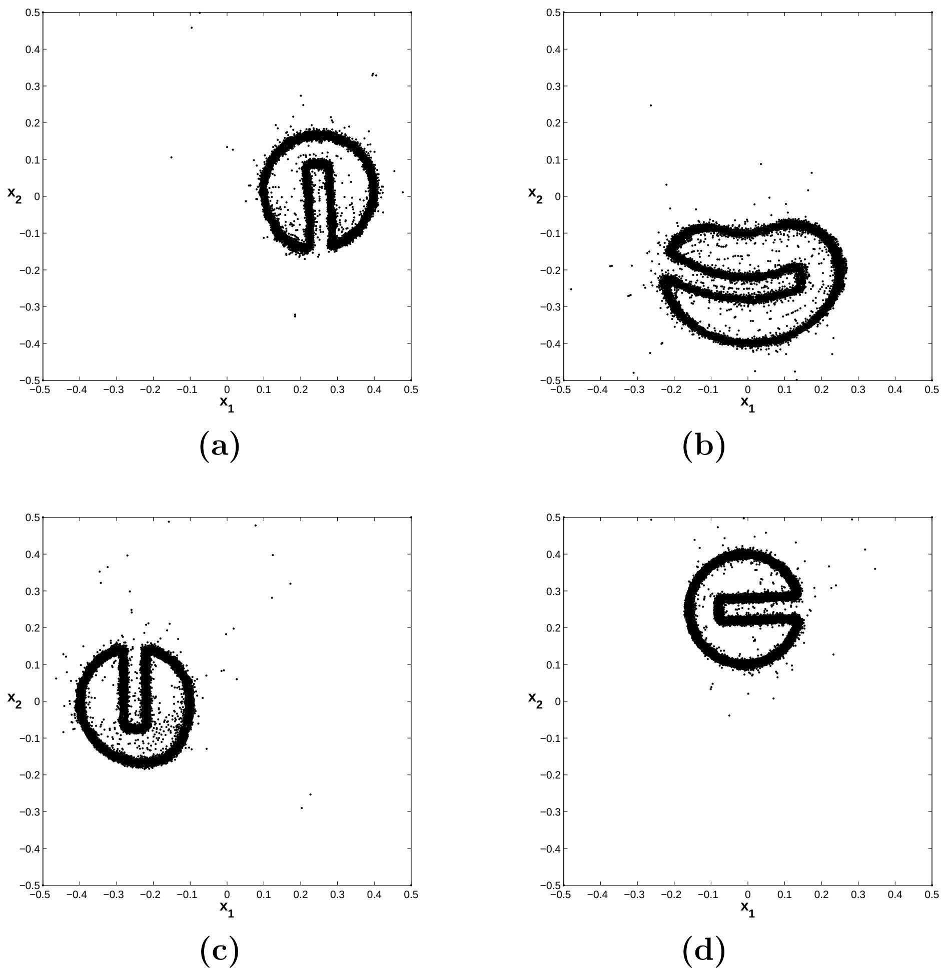


Figure 2.12: The slotted cylinder. Node distribution during the simulation at four different times, **(a)** $t = t_{53}$; **(b)** $t = t_{70}$; **(c)** $t = t_{102}$; **(d)** $t = t_{180}$, during the first revolution.

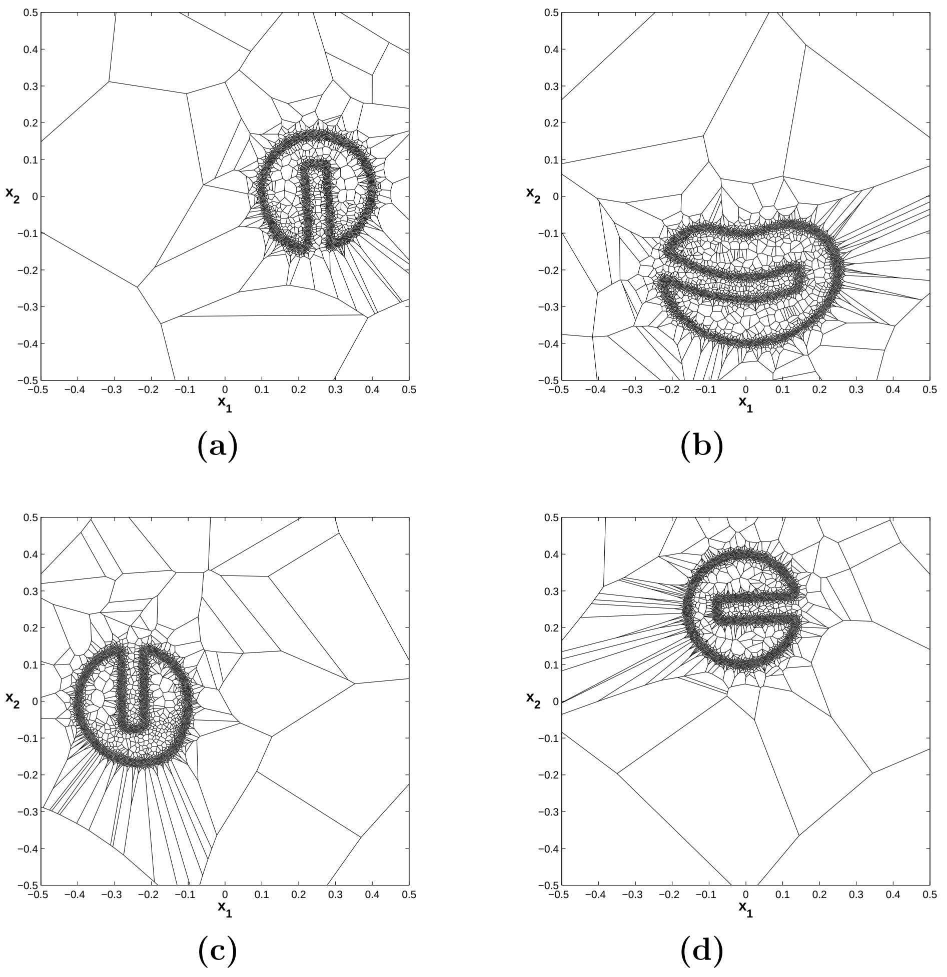


Figure 2.13: The slotted cylinder. Voronoi diagram $\mathcal{V}_\Xi$ during the simulation at four different times, **(a)** $t = t_{53}$; **(b)** $t = t_{70}$; **(c)** $t = t_{102}$; **(d)** $t = t_{180}$, during the first revolution.

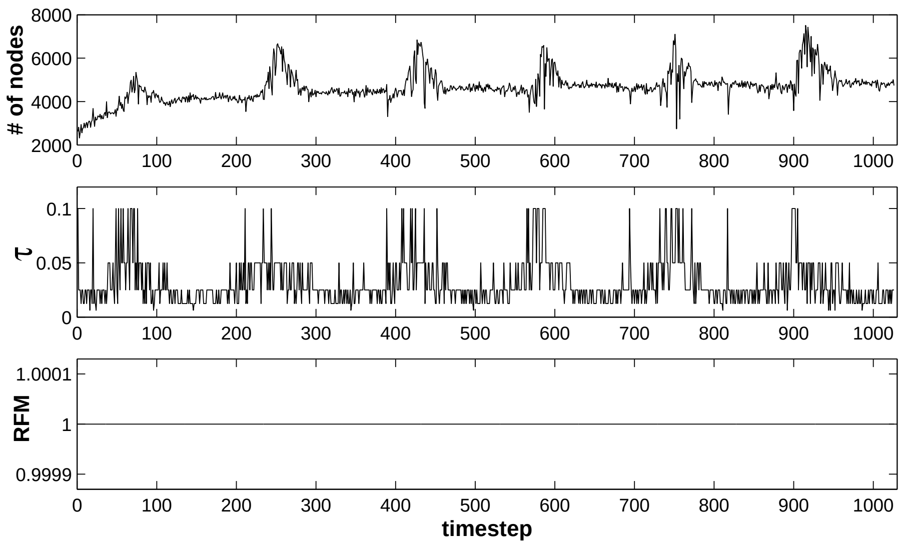


Figure 2.14: The slotted cylinder. Number of nodes, time step size τ, and ratio of the first mass moment (RFM).

(long time steps), whenever the cylinder passes through the accelerating part of the velocity field, whereas a slow down of the cylinder (short time steps) is observed in its decelerating part. Altogether, this explains the correlation between the time step size τ and the number of nodes, as shown in Figure 2.14.

As to mass conservation, we have implemented boundary conditions, as explained in Subsection 2.4.3. We let $m_U = 0$ for all tiles $U \in \Omega_{\text{in}}$, and so we have $m_{\text{in}}(t) = 0$, for all $t \in [\tau, T]$, for the total mass of the incoming flow. Figure 2.14, third row, shows the ratio of the first mass moment, RFM(t) in (2.23). Note that

$$\text{RFM}(t) \equiv 1, \qquad \text{for all } t \in [0, T],$$

and so this confirms that our proposed advection scheme is *conservative.*

Figure 2.10 shows the 3D view of the cell averages $\bar{u}_V$, the node distribution, and the Voronoi diagram of the initial condition (2.21) in the left column, in comparison to the corresponding numerical result after six full revolutions (right column). Observe that the shape of the cylinder is accurately maintained during the simulation, and numerical diffusion is widely suppressed. Finally, the employed TVD slope limiter helps to avoid spurious oscillations of the numerical solution u_h. In fact, the slope limiter serves to guarantee the non-negativity of the cell averages throughout the entire simulation, i.e., $\bar{u}_V(t) \geq 0$ for all $t \in I, V \in \mathcal{V}_\Xi$.

2.7 Conclusion

We have proposed a conservative and adaptive advection scheme for linear hyperbolic conservation laws. This semi-Lagrangian method works with finite volumes of an unstructured mesh, given by the Voronoi diagram of a current node set. The nodes, and so the Voronoi diagram, are subject to adaptive modifications during the simulation. These modifications are done according to customized adaption rules, which rely on available error estimates based on scattered data interpolation. This adaptive approach helps to reduce the required computational costs while maintaining the accuracy, due to higher resolution around discontinuities in the solution. The implementation of boundary conditions is considered in order to control mass flow into or out of the computational domain. As confirmed in two different numerical experiments, the proposed advection scheme is of second order. The scheme avoids spurious oscillations of the solution by using a TVD slope limiter in the reconstruction of linear polynomials. Altogether, the numerical results confirm the good performance of the proposed conservative advection scheme.

However, we remark that the proposed conservative semi-Lagrangian approach cannot be extended to nonlinear advection equations in a straight forward manner. This is mainly due to the computation of the upstream cells using backward trajectories, as they cannot be determined by the algorithms suggested in this Chapter. In fact, the simulation of nonlinear transport processes requires additional sophisticated techniques, especially for modelling shock front propagation. The construction of conservative methods for nonlinear transport equations is currently an active research area. Promising approaches include high order accurate **ADER** schemes (**A**rbitrary high order schemes using high order **DER**ivatives), which were recently introduced in [81, 84]. However, they belong to the class of finite volume schemes and therefore are subject to more severe time step restrictions due to stability requirements. Therefore, the idea of a semi-Lagrangian approach is dropped in the following Chapter 3, where we combine conservative ADER schemes on unstructured meshes with our powerful and robust adaption strategy in order to accurately model moving discontinuities.

Chapter 3

Adaptive ADER Schemes

In this work an extension of ADER schemes is presented in order to solve both linear and nonlinear scalar conservation laws on unstructured triangulations. The proposed scheme is conservative and belongs to the class of finite volume schemes. It combines high order reconstruction techniques with a high order flux evaluation method to update cell average values through fluxes across cell interfaces. The ADER approach results in an explicit, one-step scheme based on the solution of generalized Riemann problems via Taylor series expansion of the solution and the solution of conventional Riemann problems for the state and the derivatives. Moreover, the triangulation is adaptively modified during the simulation to effectively combine high order accuracy with locally refined meshes and therefore reduce the computational costs. The required adaption rules for the refinement and coarsening of the triangular mesh rely on a customized error indicator. Numerical experiments confirm the expected orders of accuracy and show the good performance of the proposed scheme for linear and nonlinear problems. Finally, the adaptive ADER schemes are applied to a test case from the oil industry, which plays an important role in the modelling of fluid flow in petroleum reservoirs.

3.1 General Overview

Modern approaches of constructing conservative, very high order numerical methods for hyperbolic conservation laws are typically based on the Finite Volume approach combined with *essentially non-oscillatory* (ENO) or *weighted essentially non-oscillatory* (WENO) techniques. Harten, Engquist, Osher, and Chakravarthy [38] introduced a one-dimensional cell average version of the original ENO schemes. Later, Harten and Chakravarthy [37], Abgrall [1], and Sonar [79] extended the finite volume formulation of ENO schemes to unstructured triangular meshes. The central idea of ENO schemes is to select the *smoothest* stencil out of several possible ones and then reconstruct the solution from cell averages with high order accuracy, e.g. by using

high order polynomials. This way, the growth of spurious oscillations can successfully be avoided. The more advanced WENO schemes were first suggested by Liu, Osher, and Chan [54] and Jiang and Shu [44]. Very recently, Friedrich [29] constructed WENO schemes on unstructured meshes based on the dual mesh of a triangulation. In the WENO approach, a set of different possible reconstruction polynomials is used in order to construct a specific weighted sum of polynomials. In contrast to ENO schemes the major advantages of WENO schemes are the better convergence to steady-state solutions and the increased accuracy, especially in smooth regions of the solution.
High order accuracy in time is typically achieved through multi-stage Runge-Kutta methods. However, to retain the monotonicity properties of the space discretisation the chosen time discretisation requires to be *total variation diminishing* (TVD) as observed by Shu [77] and Shu and Osher [78]. Ruuth and Spiteri [71] recently showed, that using such TVD Runge-Kutta methods constitutes barriers to the order of time accuracy and consequently to the order of the entire scheme. Thus, in most practical implementations a third order TVD Runge-Kutta method is used even for very high order WENO schemes, i.e. the spatial order of accuracy is much larger than three.
A new approach introduced by Toro, Millington, and Nejad in [84] and further developed by Titarev in [81, 83] is the so-called ADER approach, which is an explicit one-step finite volume scheme of **A**rbitrary high order using high order **DER**ivatives of piecewise polynomial reconstructions. In [81, 84, 83] a very high order version of the classical Godunov scheme [32] is constructed, which leads to an arbitrary high order of accuracy in both, space and time. In fact, ADER schemes can be interpreted as high order generalizations of the classical Godunov scheme. In the last few years, the use of ADER schemes has gained considerable popularity in the field of gas and aerodynamics, e.g. [72, 73], especially for linear advection problems, and currently constitutes a very active research area, also for nonlinear problems and systems of hyperbolic equations.

3.2 Introduction

In this paper, high order WENO schemes on unstructured triangulations are combined with the *ADER* approach introduced in [84] to solve scalar, *linear* and *nonlinear* conservation laws of the form

$$\frac{\partial u}{\partial t} + \nabla f(u) = 0\,, \tag{3.1}$$

where for some domain $\Omega \subset \mathbb{R}^2$, and a compact time interval $I = [0, T]$, $T > 0$, the function $u : I \times \Omega \to \mathbb{R}$ is unknown and $f(u) = (f_1(u), f_2(u))^T$ denotes the *flux tensor*. Furthermore, *adaptive mesh refinement* is included to balance computational cost and approximation quality. This is important, in

particular, in the vicinity of discontinuities, that typically occur in solutions of hyperbolic problems and can be accurately resolved by locally refined meshes. Therefore, our aim is to combine the ADER approach with adaptive mesh refinement in order to obtain highly accurate results at reasonable computational costs.
In general, a high order extension of the classical scheme of Godunov [32] consists of the three basic operations:

- polynomial reconstruction of the solution from cell average values,
- evaluation of fluxes across interfaces between adjacent cells,
- update (evolution) of cell average values in each cell.

The present paper is arranged by following these main steps. An introduction to the reconstruction of high order polynomials from cell average values on unstructured triangulations is given in Section 3.3. The corresponding WENO reconstruction is addressed and followed by a detailed description of the stencil selection algorithm. The finite volume formulation of the governing equation (3.1) on unstructured triangulations is outlined in Section 3.4 together with the presentation of the high order flux evaluation technique using the ADER approach. The remaining update of the cell average values is discussed in Section 3.5. The treatment of boundary conditions is addressed in Section 3.6 before the accuracy of the proposed scheme is evaluated in Section 3.7. The brief discussion of the error estimation in Section 3.8 is based on the ideas presented in Sections 1.5 and 2.5, but the adaption rules and the corresponding refinement and coarsening strategy have to be modified accordingly. Finally, the good performance of an adaptive high order ADER scheme is confirmed by numerical experiments in Section 3.9.

3.3 Reconstruction of High Order Polynomials

The reconstruction of high order polynomials on unstructured triangulations is much more difficult than the reconstruction on one-dimensional intervals or multi-dimensional Cartesian grids. In fact, polynomial reconstruction on scattered data requires the solution of multi-dimensional interpolation problems, which typically tend to be ill-conditioned. This problem becomes even more critical with increasing order of the reconstruction.
To keep the notation short we use multi-indices, i.e. $\alpha = (\alpha_1, \alpha_2)$ with $\alpha_i \in \{0, 1, 2, ...\}$, $i = 1, 2$, and $x = (x_1, x_2) \in \mathbb{R}^2$. Moreover, we let $|\alpha| = \alpha_1 + \alpha_2$ and $x^\alpha = x^{(\alpha_1,\alpha_2)} = x_1^{\alpha_1} x_2^{\alpha_2}$. In the following, let $\mathcal{P}_n$ denote

the set of bivariate polynomials of degree at most n. Then, the standard expansion of any polynomial $p \in \mathcal{P}_n$ is given by

$$p(x) = \sum_{|\alpha| \leq n} a_\alpha (x-b)^\alpha \,, \tag{3.2}$$

where the $a_\alpha \in \mathbb{R}$ are the coefficients of the Taylor series expansion of p around b and $b = (b_1, b_2)$ is any point in $\mathbb{R}^2$. The set $\mathcal{P}_n$ is a vector space of dimension $N(n) = \frac{1}{2}(n+1)(n+2)$, and $\{(\,\cdot\, - b)^\alpha\}_{|\alpha| \leq n}$ constitutes a basis of $\mathcal{P}_n$. We remark, that the expansion (3.2) is not a suitable basis for practical computations. However, we want to keep the notation short for the moment. Let us assume, that the computational domain $\Omega \subset \mathbb{R}^2$ is discretized by a conforming triangulation $\mathcal{T}$ (cf. Section 3.3.1 in [55]), formed by the set $\mathcal{T} = \{T_\ell\}_\ell$ of triangles $T_\ell \subset \Omega$, $\ell = 1, ..., \#\mathcal{T}$. In the finite volume framework each triangle (cell, control volume) T_ℓ carries a *cell average value*

$$\bar{u}_\ell = \frac{1}{|T_\ell|} \int_{T_\ell} u(x)\, dx \,, \tag{3.3}$$

where $|T_\ell|$ is the area of triangle T_ℓ and u is the solution of (3.1).
Now, in the reconstruction we consider solving the following problem:

Given the polynomial degree n and cell average values $\bar{u}_{\ell_k}$, $k = 1, ..., N$, $N = \dim \mathcal{P}_n$, of the function u on each control volume T_{ℓ_k}, find a polynomial $p \in \mathcal{P}_n$, that satisfies

$$\begin{array}{rcl} \bar{p}_{\ell_1} & = & \bar{u}_{\ell_1} \,, \\ \bar{p}_{\ell_2} & = & \bar{u}_{\ell_2} \,, \\ \vdots & & \vdots \\ \bar{p}_{\ell_N} & = & \bar{u}_{\ell_N} \,. \end{array} \tag{3.4}$$

The linear system (3.4) has a unique solution, iff the Vandermonde matrix

$$M_\ell = \left(\overline{(\,\cdot\, - b)^\alpha}_{\ell_k} \right)_{1 \leq k \leq N,\, |\alpha| \leq n} \tag{3.5}$$

is non-singular. In this case, we call the set $\mathcal{S} = \{T_{\ell_k}\}_{1 \leq k \leq N}$ of triangles a $\mathcal{P}_n$-unisolvent or *admissible* stencil . Note that the matrix M_ℓ in (3.5) may be ill-conditioned due to the following reasons:

- Firstly, as Abgrall has shown in [1], the condition number of the above system matrix is $\mathcal{O}(h^{-n})$, where $h = \sqrt{|T_\ell|}$ is a measure for the local mesh width and n the degree of the polynomial space.

- Secondly, the matrix can be badly conditioned due to the geometry of the chosen stencil.

The first problem has been considered by Friedrich in [29], who introduces a scaling factor $s = (\sqrt{|T_\ell|})^{-1}$ to obtain a condition number independent from h.

Therefore, the standard representation (3.2) of a polynomial is changed to

$$p(x) = \sum_{|\alpha| \le n} s^{|\alpha|} \tilde{a}_\alpha (x-b)^\alpha \, , \tag{3.6}$$

where $\tilde{a}_\alpha$ are the scaled coefficients of p. Therefore, by using the scaled polynomial expansion (3.6), the condition number of the system matrix M_ℓ does not depend on the mesh width h. However, it is still unclear, if this expansion is sufficient to provide a robust reconstruction procedure for strongly distorted, unisotropic meshes. For this reason, Abgrall suggests in [1] to use a polynomial expansion based on barycentric coordinates . For simplicity, let $\mathcal{S}_n = \{T_1, T_2, ..., T_N\}, n \ge 3$, be an admissible stencil. Then at least one subset of three elements, say $\{T_1, T_2, T_3\}$, is an admissible stencil for $n = 1$. Then the three linear polynomials Λ_i, $i = 1, 2, 3$, defined by

$$\bar{\Lambda}_{i_j} = \delta_i^j \, , \qquad 1 \le i, j \le 3, \qquad \text{with} \quad \sum_{i=1}^{3} \Lambda_i = 1 \tag{3.7}$$

are the *barycentric coordinates* of the triangle constructed on the barycenters of T_1, T_2, and T_3. The polynomial expansion is now given by

$$p(x) = \sum_{|\alpha| \le n} \hat{a}_\alpha \Lambda_2^{\alpha_1} \Lambda_3^{\alpha_2} \, , \tag{3.8}$$

where $\hat{a}_\alpha$ are the new coefficients and Λ_2, Λ_3 are two barycentric coordinates from (3.7). Especially with respect to adaptive mesh refinement, we use the polynomial expansion (3.8), which is independent of the local mesh width h as shown in [1] and turns out to be very robust even for strongly distorted meshes.

The second problem was already considered in previous work [4, 62], and can be overcome using overdetermined systems. Instead of using exactly N neighbouring cells, we work with a slightly larger stencil to enhance the robustness of our reconstruction procedure. In our computations we typically use 4 cells for linear, 8 cells for quadratic, and 13 cells for cubic reconstruction. To guarantee a conservative scheme we need to satisfy

$$\bar{p}_\ell = \bar{u}_\ell \tag{3.9}$$

on the cell T_ℓ, on which the polynomial p is computed. Therefore, the overdetermined system (3.4) states a linear least-squares problem with the linear equality constraint (3.9), which can be solved (cf. Chapter 21 in [48]). Finally, we remark, that the entries of the system matrix M_ℓ, $\ell = 1, ..., \#\mathcal{T}$, in (3.5) can be computed by using quadrature rules for triangles, that are exact for the desired polynomial degree n. A detailed list of quadrature rules for triangles is given in Appendix C.

3.3.1 WENO Reconstruction

WENO methods have been extensively used for one-dimensional problems in the last decade and also have gained popularity for problems on multi-dimensional Cartesian grids. The general idea of ENO and WENO schemes is to chose several stencils S_i, $i = 1, ..., k$, where k denotes the number of stencils, and to compute the corresponding reconstruction polynomials p_i. The ENO approach only uses the one polynomial with the least oscillating behaviour. In contrast, WENO methods work with a weighted sum

$$p(x) = \sum_{i=1}^{k} \omega_i p_i(x) , \tag{3.10}$$

where the ω_i are positive, data-dependent, and normalized weights, such that $\sum_{i=1}^{k} \omega_i = 1$. Originally introduced in [44, 54], the WENO approach was extended to unstructured meshes in [29, 19]. In order to compute the weights ω_i in (3.10), we have to clarify how the oscillation of the corresponding polynomial p_i is measured. Numerical tests in [29, 19] have demonstrated, that a suitable *oscillation indicator* for a polynomial p on a triangular cell T is given by

$$\mathcal{I}_i = \sum_{1 \le |\alpha| \le n} \int_T |T|^{|\alpha|-1} \Big(D^\alpha p_i(x) \Big)^2 \, dx , \tag{3.11}$$

where $D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2}}$ is the α-th partial derivative operator with respect to x_1 and x_2. The factor $|T|^{|\alpha|-1}$ eliminates effects due to the local mesh width. Then, the weights ω_i can be calculated through

$$\omega_i = \frac{\tilde{\omega}_i}{\sum_{i=1}^{k} \tilde{\omega}_i} \qquad \text{with} \qquad \tilde{\omega}_i = (\epsilon + \mathcal{I}_i)^{-r} . \tag{3.12}$$

Here ϵ is a small positive number to avoid the division by zero. Usually, numerical results are not very sensitive to the choice of ϵ. In general, however, larger ϵ are better suited for smooth problems but may generate small oscillations near shocks, whereas smaller ϵ are better suited for discontinuous problems. In the literature, typically values of $\epsilon \in [10^{-6}, 10^{-2}]$ are chosen. In our computations, we let $\epsilon = 10^{-5}$. The positive integer r in (3.12) is a measure of the sensitivity of the weights with respect to the oscillation indicator of (3.11). As r tends to infinity, the WENO scheme behaves like a classical ENO scheme. On the other hand, if r tends to zero, the oscillation indicator has almost no effect on the weights, which means that the scheme becomes an oscillatory, or even unstable scheme. In most applications we find $r \in [2, 8]$, and in our implementation we chose $r = 4$, which turns out to be large enough to essentially avoid oscillations near discontinuities, but small enough to improve upon the classical ENO scheme.

3.3.2 Stencil Selection Algorithm

So far, we have assumed to have admissible stencils. However, as shown in Section 3.6.1 of [55], selecting a small number of reasonable, admissible stencils on unstructured triangulations is not a trivial task as there is a large number of possible stencils to choose from. Furthermore, for polynomials of degree greater than 1 it is unknown, if there is a geometrical property on unstructured meshes indicating if a chosen stencil is admissible or not.
However, some major aspects for selecting a stencil have to be taken into account:

- the stencils should be local,
- the number of stencils should be small to keep computational costs small,
- in smooth regions it is necessary that the stencil is well centered with respect to the cell T_ℓ to obtain a good approximation quality,
- in non-smooth regions one-sided stencils have to be selected to avoid interpolation across discontinuities leading to oscillations.

Considering the construction of stencils on unstructured triangulations, it is convenient to define the neighbourhood of adjacent triangles as introduced in [79].

Definition 2 *Let $\mathcal{T}$ be a conforming triangulation. Then for any triangle $T_\ell \in \mathcal{T}$ the set*

$$\mathcal{K}_N^0(T_\ell) = \{T \in \mathcal{T} \; : \; T \cap T_\ell \;\; \textit{is edge of} \;\; T_\ell \;\; \textit{and} \;\; T \neq T_\ell\}$$

is called the von Neumann neighbourhood *of T_ℓ and all triangles $T \in \mathcal{K}_N^0(T_\ell)$ are* level-0 von Neumann neighbours *of T_ℓ.*

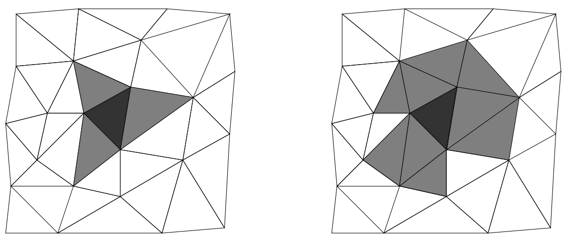


Figure 3.1: A triangle (dark shaded) with **(a)** level-0 von Neumann neighbours (light shaded) and **(b)** additional level-1 von Neumann neighbours.

An extended von Neumann neighbourhood of level-1 can be constructed by merging von Neumann neighbourhoods of the original level-0 von Neumann neighbours, i.e.

$$\mathcal{K}_N^1(T_\ell) = \left(\bigcup_{T \in \mathcal{K}_N^0(T_\ell)} \mathcal{K}_N^0(T) \right) \setminus T_\ell \,,$$

as shown in Figure 3.1**(a)** and **(b)**. This way, we can extend the von Neumann neighbourhoods level by level, until a desired number of cells, i.e. a desired size of a stencil for the polynomial reconstruction is reached. Note, that this procedure typically leads to centered stencils as shown in the top row of Figure 3.4. In order to construct one-sided stencils in the vicinity of discontinuous data, we follow the idea of Harten and Chakravarthy in [37] and use a sectoral search algorithm. Their basic idea is to include only von Neumann neighbours, whose barycenters lie in specified sectors.

Definition 3 *Let $T_\ell \in \mathcal{T}$ be a triangle with counter-clockwise ordered vertices $v_1, v_2, v_3 \in \mathbb{R}^2$ and $f_{11} = v_2 - v_1$, $f_{12} = v_3 - v_1$, $f_{21} = v_3 - v_2$, $f_{22} = v_1 - v_2$ and $f_{31} = v_1 - v_3$, $f_{32} = v_2 - v_3$ the vector pairs representing the oriented triangle edges. Then the sets*

$$\mathcal{F}_j = \{x = v_j + \gamma_1 f_{j1} + \gamma_2 f_{j2} \,:\, \gamma_1, \gamma_2 \geq 0\}\,, \quad j = 1, 2, 3 \tag{3.13}$$

are called the forward sectors *of triangle T_ℓ* (see Figure 3.2).

However, our numerical tests have shown, that the three sectors defined by (3.13) not always provide stencils with smooth data. Therefore, we introduce additional sectors in order to to cover neighbouring regions of a triangular cell T_ℓ, that are not covered by the three forward sectors $\mathcal{F}_j$, $j = 1, 2, 3$.

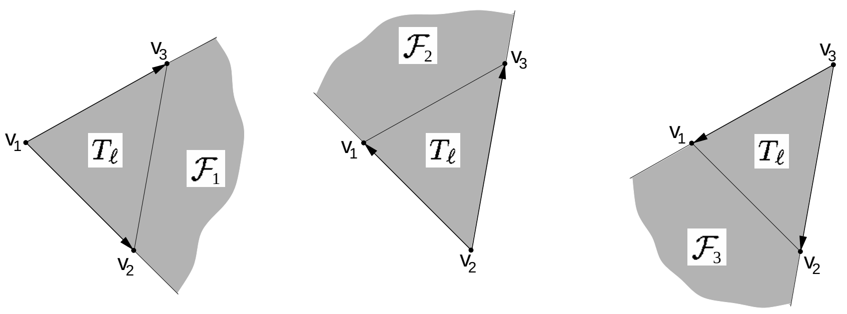


Figure 3.2: The three forward sectors of triangle T_ℓ.

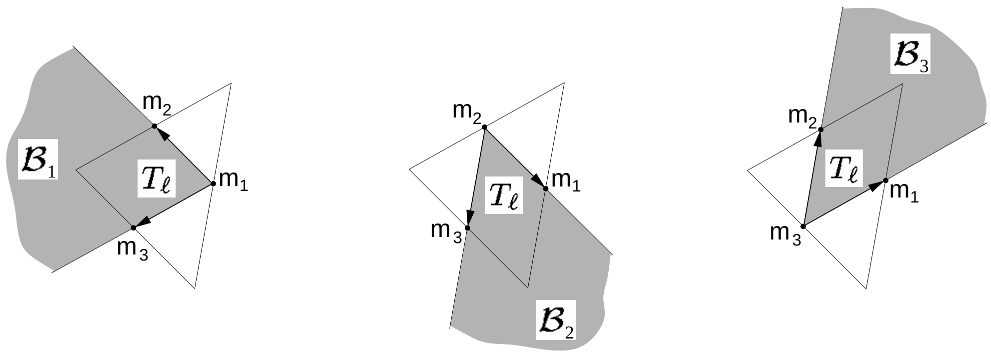


Figure 3.3: The three backward sectors of triangle T_ℓ.

Definition 4 *Let $T_\ell \in \mathcal{T}$ be a triangle with $m_1, m_2, m_3 \in \mathbb{R}^2$ denoting the midpoints of the triangle edges and $b_{11} = m_2 - m_1$, $b_{12} = m_3 - m_1$, $b_{21} = m_3 - m_2$, $b_{22} = m_1 - m_2$ and $b_{31} = m_1 - m_3$, $b_{32} = m_2 - m_3$ the vector pairs representing the oriented lines parallel to the triangle edges. Then the sets*

$$\mathcal{B}_j = \{x = m_j + \gamma_1 b_{j1} + \gamma_2 b_{j2} \, : \, \gamma_1, \gamma_2 \geq 0\}\,, \quad j = 1, 2, 3 \tag{3.14}$$

are called the backward sectors *of triangle T_ℓ* (see Figure 3.3).

The second and third row in Figure 3.4 show stencils of size 6 constructed by successive von Neumann neighbours with barycenters inside the three different forward sectors $\mathcal{F}_j$ from (3.13) and backward sectors $\mathcal{B}_j$ from (3.14). We remark, that the shape of the selected stencils strongly depends on the local geometry of the mesh. Especially for high order reconstruction, as mentioned above, a chosen stencil might turn out to be *non-admissible* in the sense, that the resulting reconstruction problem has no unique solution. In this case, such stencils are detected and ignored. However, we remark, that we never encountered an inadmissible stencil in our computations when we use the approach in [4, 62], where slightly larger stencils lead to overdetermined systems.

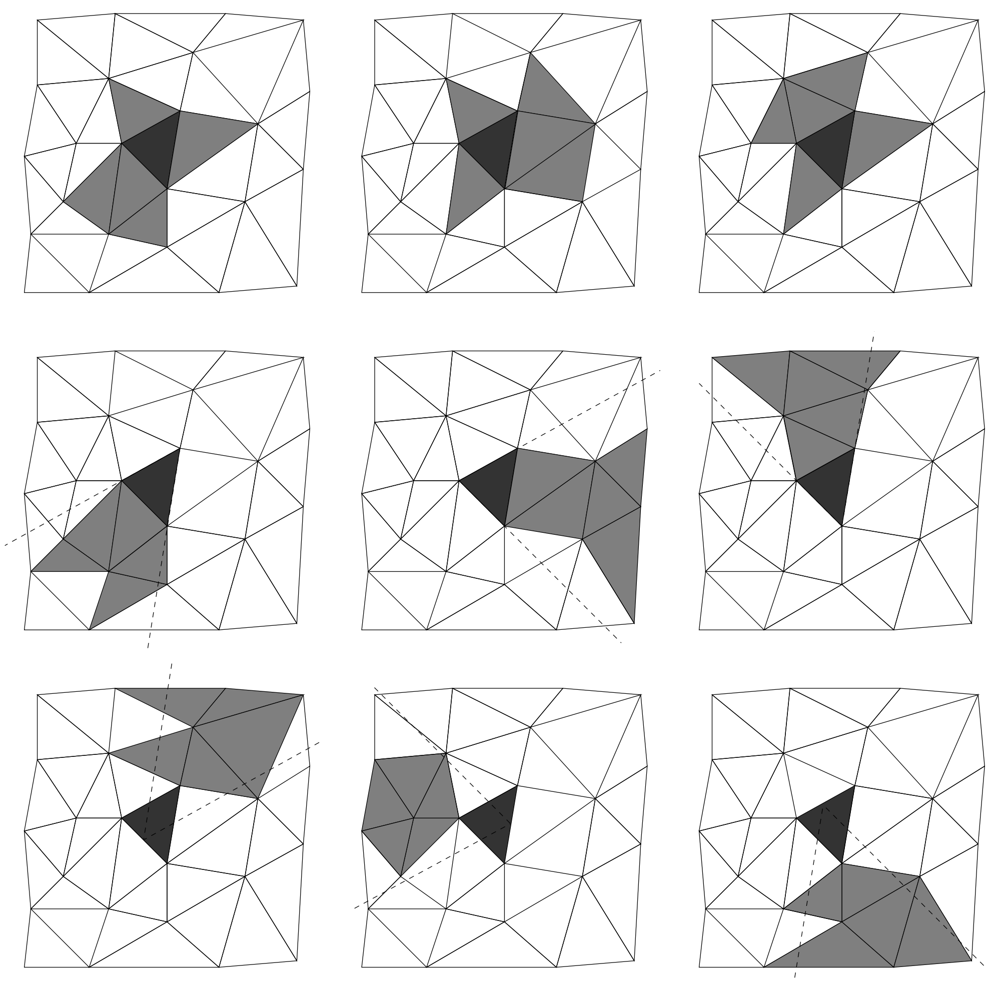


Figure 3.4: Example of nine stencils of size 6 constructed by combining successive von Neumann neighbours and a sectoral search.

3.4 Finite Volume Formulation

Consider a two-dimensional, scalar conservation law of the form (3.1) with solution $u(t,x)$. Within the finite volume framework , each discrete value of the function u is viewed as a cell average $\bar{u}_\ell$ over a cell T_ℓ . The advantage of the finite volume approach is, that any kind of mesh can be used, i.e. the shape of the control volume can be chosen arbitrarily. Here, we work with a conforming triangulation $\mathcal{T}$ with cells $T_\ell \in \mathcal{T}$, $\ell = 1, ..., \#\mathcal{T}$, for which the integral form of the conservation law in (3.1) has the form

$$\frac{d}{dt}\int_T u(t,x)\,dx + \int_{\partial T} \vec{F}(t,s)\cdot\vec{n}(s)\,ds = 0\,, \tag{3.15}$$

with the outer normal vector $\vec{n}(s)$ and the flux

$$\vec{F}(t,s) = \begin{pmatrix} f_1(u(t,x(s))) \\ f_2(u(t,x(s))) \end{pmatrix}$$

where the boundary ∂T of the triangle T is parameterized by arclength s (cf. Chapter 23 in [50]). Integrating (3.15) over the time interval $[t^n, t^{n+1}]$, where $\tau = t^{n+1} - t^n$ is the time step length, and using the definition of cell averages in (3.3), we derive a finite volume scheme of the form

$$\bar{u}_\ell^{n+1} = \bar{u}_\ell^n - \frac{\tau}{|T_\ell|}\sum_{j=1}^{3} \hat{F}_{\ell,j}^n\,, \tag{3.16}$$

where the *numerical flux* $\hat{F}_{\ell,j}^n$ across each cell boundary $\partial T_{\ell,j}$, $j = 1,2,3$, of the cell T_ℓ during the time interval $[t^n, t^{n+1}]$ is the time-averaged physical flux given by

$$\hat{F}_{\ell,j}^n = \frac{1}{\tau}\int_{t^n}^{t^{n+1}} \left(\int_{\partial T_{\ell,j}} \vec{F}(t^n,s)\cdot\vec{n}_{\ell,j}\,ds\right)dt\,. \tag{3.17}$$

The time integral and the integral along the j-th edge of triangle T_ℓ in (3.17) can be computed exactly by using a suitable Gaussian quadrature rule. Therefore, the numerical flux can be computed through the weighted sum

$$\hat{F}_{\ell,j}^n = \sum_{k=1}^{N_t} \alpha_k |\partial T_{\ell,j}| \sum_{h=1}^{N_x} \beta_h \vec{F}(u(t_{G_k}, x_{G_h}))\cdot\vec{n}_{\ell,j}\,, \tag{3.18}$$

where α_k and β_h are the weights of the Gaussian quadrature rule and t_{G_k} and x_{G_h} are the corresponding integration points with respect to time and space. N_t and N_x are the numbers of integration points. The situation for a third order approximation using two Gaussian integration points in time

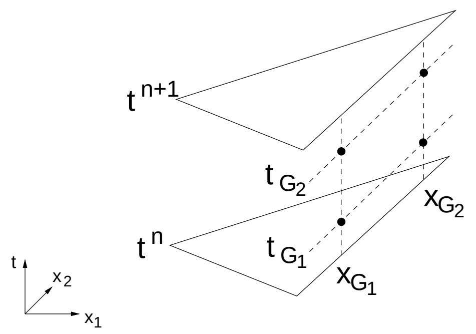


Figure 3.5: Example of a third order accurate flux evaluation across a triangle edge from time t^n to t^{n+1} using two Gaussian integration point in space and time each.

and space is illustrated in Figure 3.5. It is clear, that in order to evaluate the flux function $\vec{F}$ at a particular Gaussian integration point we have to find the function value $u(t_{G_k}, x_{G_h})$, the so-called *state* of the solution by solving a Riemann problem at this point. This is accomplished through the ADER approach, which is extended to unstructured meshes and discussed in detail in the following section.

3.4.1 Flux Evaluation via ADER

Originally, Toro, Millington, and Nejad introduced a method in [84] termed ADER to construct arbitrary high order finite volume schemes for scalar, *linear* conservation laws utilizing high order derivatives. Just very recently, these schemes were extended to scalar, *nonlinear* conservation laws in one dimension by Titarev and Toro in [81, 83] and were applied to problems on multi-dimensional, Cartesian grids by Schwartzkopff, Munz and Toro in [72, 73]. A first attempt to extend ADER schemes for linear conservation laws from structured grids to unstructured triangulations was taken in a preliminary, unpublished note by Munz and Schneider [58].
The main ingredients of the proposed ADER scheme are:

- a WENO technique to reconstruct high order polynomials without creating spurious oscillations,
- a high order flux evaluation based on a Taylor series expansion in time leading to an explicit one-step method to evolve cell averages,
- a Lax-Wendroff procedure to replace time derivatives by spatial derivatives through extensively using the information of the governing PDE,
- the solution of generalized Riemann problems across cell interfaces.

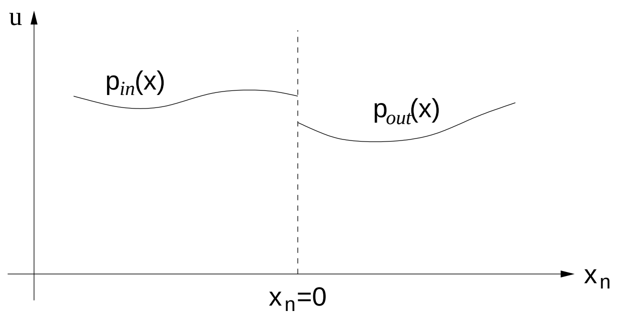


Figure 3.6: The generalized Riemann problem along the outward pointing unit vector with reconstructed polynomials $p_{in}(x)$ and $p_{out}(x)$ approximating the solution u.

In the following, we discuss in detail the construction of ADER schemes for *linear* and *nonlinear* problems. According to the WENO reconstruction procedure in Section 3.3, the solution $u(t, x)$ at the discrete time $t = t^n$ is represented by polynomials p_ℓ, $\ell = 1, ..., \#\mathcal{T}$, on each triangular cell $T_\ell \in \mathcal{T}$. In general, these polynomials are different on each cell, leading to a piecewise polynomial approximation of $u(t, x)$ with discontinuities across the cell interfaces. The situation of having two constant functions separated by a discontinuity is usually called a conventional *Riemann problem* (RP). The solution of a RP is a fundamental tool in the development of finite volume methods and is discussed in many textbooks, e.g. [50, 82]. In general, a RP is defined by the governing hyperbolic equation together with a particular initial condition (IC). As mentioned above this IC is usually given by two constant functions and the solution of the RP can be computed through various techniques [50, 82]. However, as our functions p_ℓ, which are separated by the cell interfaces, are not necessarily constant, the situation is more difficult and is called the *generalized Riemann problem* (GRP). Depending on the order m of the designed ADER scheme we will call these schemes ADERm schemes . Therefore, the ADER1 scheme is the classical Godunov scheme [32] of first order (m=1). We remark, that the degree of the reconstruction polynomials for an ADERm scheme is $m - 1$. Denoting the corresponding GRP more precisely, we have to solve a GRP$_{m-1}$ at the cells interfaces when using an ADERm scheme[1].

In order to apply the ideas in [81, 83], we reduce the multi-dimensional GRP at the Gaussian integration points x_{G_h} of a cell interface (see Figure 3.5) to a one-dimensional GRP oriented perpendicularly to the interface, i.e. along the outer normal $\vec{n}$ as displayed in Figure 3.6. The GRP is described by the

[1]GRP$_0$ denotes the conventional Riemann Problem (RP) with two constant functions as initial condition.

governing PDE and the IC of $u(t,x)$ at the local time $t = 0$ by

$$\text{PDE:} \qquad \frac{\partial u}{\partial t} + \nabla f(u) = 0\,, \tag{3.19}$$

$$\text{IC:} \qquad u(0,x) = \begin{cases} p_{in}(x)\,, & \text{for} \quad x_n < 0\,, \\ \\ p_{out}(x)\,, & \text{for} \quad x_n > 0\,, \end{cases} \tag{3.20}$$

where x_n is a local coordinate oriented along the outer normal $\vec{n}$ with the origin at the Gaussian integration point x_{G_h}. The two polynomials belonging to the actual cell and the adjacent cell are indicated by p_{in} and p_{out}, respectively.
Recalling equation (3.18) and Figure 3.5, we are looking for the solution of the one-dimensional GRP at a Gaussian integration point t_{G_k} in time, i.e. for $u(t,\cdot)$ at an intermediate time $t \in [t^n, t^{n+1}]$. Now, one of the central ideas of the ADER approach is to approximate this solution at m-th order accuracy at the cell interface $x_n = 0$ via the Taylor series expansion in time around the initial time $t = 0$ given through

$$u(t,0) \approx u(0,0) + \sum_{k=1}^{m-1} \frac{t^k}{k!} \frac{\partial^k}{\partial t^k} u(0,0)\,. \tag{3.21}$$

The time derivatives in (3.21) can be replaced by space derivatives by applying the *Lax-Wendroff (Cauchy-Kowalewski) procedure* in order to make the maximum use of the information given by the governing PDE in (3.19).

3.4.2 The Lax-Wendroff Procedure

Originally, this technique of substituting time by space derivatives using the governing PDE (3.19) itself was considered in [49]. We remark that for nonlinear problems this procedure can become quite tedious in contrast to the linear case, as the number of terms required to express the time derivatives grows rapidly with the order of the derivative. However, as shown in [83], these terms are necessary in order to guarantee the desired order of accuracy of the designed ADERm scheme. To be more precise, let the characteristic speeds with respect to the space dimensions x_1 and x_2 be given by

$$\lambda_1(u) = \frac{\partial f_1(u)}{\partial u} \quad \text{and} \quad \lambda_2(u) = \frac{\partial f_2(u)}{\partial u}\,. \tag{3.22}$$

Using the notation $\lambda_i'(u) = \frac{\partial \lambda_i(u)}{\partial u}$, $\lambda_i''(u) = \frac{\partial^2 \lambda_i(u)}{\partial u^2}$ etc., $i = 1,2$, and $u_t = \frac{\partial u}{\partial t}$, $u_{tt} = \frac{\partial^2 u}{\partial t^2}$, $u_{x_1} = \frac{\partial u}{\partial x_1}$, etc., we can write the governing PDE (3.19) in two dimensions as

$$u_t + \lambda_1(u) u_{x_1} + \lambda_2(u) u_{x_2} = 0\,. \tag{3.23}$$

Therefore, to replace the first order time derivative in (3.21) by space derivatives using the Lax-Wendroff procedure, we simply have to solve (3.23) for u_t leading to

$$u_t = -\lambda_1(u)u_{x_1} - \lambda_2(u)u_{x_2}\,. \tag{3.24}$$

The higher order time derivatives of u can now be computed successively by partial differentiation of (3.24) with respect to t. For example, an ADER4 scheme requires time derivatives up to order 3 in (3.21) given by

$$u_{tt} = -\lambda_1'(u)u_t u_{x_1} - \lambda_1(u)u_{tx_1} - \lambda_2'(u)u_t u_{x_2} - \lambda_2(u)u_{tx_2}\,, \tag{3.25}$$

$$\begin{aligned} u_{ttt} &= -\lambda_1''(u)u_t^2 u_{x_1} - \lambda_1'(u)\Big(u_{tt}u_{x_1} + 2u_t u_{tx_1}\Big) - \lambda_1(u)u_{ttx_1} \\ &\quad -\lambda_2''(u)u_t^2 u_{x_2} - \lambda_2'(u)\Big(u_{tt}u_{x_2} + 2u_t u_{tx_2}\Big) - \lambda_2(u)u_{ttx_2}\,. \end{aligned} \tag{3.26}$$

Note, that the above expressions for the time derivatives include mixed derivatives with respect to time and space. These can also be expressed by space derivatives by successively differentiating (3.24) with respect to x_1 and x_2. For example, for the ADER4 scheme we get

$$\begin{aligned}
u_{tx_1} &= -\lambda_1'(u)u_{x_1}^2 - \lambda_1(u)u_{x_1x_1} - \lambda_2'(u)u_{x_1}u_{x_2} - \lambda_2(u)u_{x_1x_2}\,, \\
u_{tx_2} &= -\lambda_1'(u)u_{x_1}u_{x_2} - \lambda_1(u)u_{x_1x_2} - \lambda_2'(u)u_{x_2}^2 - \lambda_2(u)u_{x_2x_2}\,, \\
u_{tx_1x_1} &= -\lambda_1''(u)u_{x_1}^3 - 3\lambda_1'(u)u_{x_1}u_{x_1x_1} - \lambda_1(u)u_{x_1x_1x_1} \\
&\quad -\lambda_2''(u)u_{x_1}^2 u_{x_2} - \lambda_2'(u)\Big(u_{x_1x_1}u_{x_2} + 2u_{x_1}u_{x_1x_2}\Big) - \lambda_2(u)u_{x_1x_1x_2}\,, \\
u_{tx_1x_2} &= -\lambda_1''(u)u_{x_1}^2 u_{x_2} - \lambda_1'(u)\Big(u_{x_1x_1}u_{x_2} + 2u_{x_1}u_{x_1x_2}\Big) - \lambda_1(u)u_{x_1x_1x_2} \\
&\quad -\lambda_2''(u)u_{x_1}u_{x_2}^2 - \lambda_2'(u)\Big(u_{x_1}u_{x_2x_2} + 2u_{x_2}u_{x_1x_2}\Big) - \lambda_2(u)u_{x_1x_2x_2}\,, \\
u_{tx_2x_2} &= -\lambda_1''(u)u_{x_1}u_{x_2}^2 - \lambda_1'(u)\Big(u_{x_1}u_{x_2x_2} + 2u_{x_2}u_{x_1x_2}\Big) - \lambda_1(u)u_{x_1x_2x_2} \\
&\quad -\lambda_2''(u)u_{x_2}^3 - 3\lambda_2'(u)u_{x_2}u_{x_2x_2} - \lambda_2(u)u_{x_2x_2x_2}\,, \\
u_{ttx_1} &= -\lambda_1''(u)u_t u_{x_1}^2 - \lambda_1'(u)\Big(u_t u_{x_1x_1} + 2u_{x_1}u_{tx_1}\Big) - \lambda_1(u)u_{tx_1x_1} \\
&\quad -\lambda_2''(u)u_t u_{x_1}u_{x_2} - \lambda_2'(u)\Big(u_{tx_1}u_{x_2} + u_{x_1}u_{tx_2} + u_t u_{x_1x_2}\Big) - \lambda_2(u)u_{tx_1x_2}\,, \\
u_{ttx_2} &= -\lambda_1''(u)u_t u_{x_1}u_{x_2} - \lambda_1'(u)\Big(u_{tx_1}u_{x_2} + u_{x_1}u_{tx_2} + u_t u_{x_1x_2}\Big) - \lambda_1(u)u_{tx_1x_2} \\
&\quad -\lambda_2''(u)u_t u_{x_2}^2 - \lambda_2'(u)\Big(u_t u_{x_2x_2} + 2u_{x_2}u_{tx_2}\Big) - \lambda_2(u)u_{tx_2x_2}\,.
\end{aligned}$$

The problem remaining is to determine the space derivatives at the quadrature points at the cell interface, i.e. to solve the one-dimensional GRP illustrated in Figure 3.6. In [83], Toro and Titarev suggest to use the *boundary extrapolated values*

$$u_l = \lim_{x \to x_{G_h}^-} p_{in}(x)\,, \tag{3.27}$$

$$u_r = \lim_{x \to x_{G_h}^+} p_{out}(x)\,, \tag{3.28}$$

which represent the values obtained by evaluating the polynomials reconstructed inside and outside the actual cell interface at a Gaussian quadrature point x_{G_h}. We remark, that according to the one-dimensional representation in Figure 3.6 the inside and outside of a cell are referred to as *left* and *right* respectively. Therefore, we obtain a conventional Riemann problem GRP$_0$ with constant functions u_l and u_r of the form

$$\text{PDE:} \qquad \frac{\partial u}{\partial t} + \nabla f(u) = 0\,, \tag{3.29}$$

$$\text{IC:} \qquad u(0,x) = \begin{cases} u_l & \text{for} \quad x_n < 0 \\ \\ u_r & \text{for} \quad x_n > 0\,. \end{cases} \tag{3.30}$$

The solution of the above GRP$_0$ (3.29),(3.30) is described in many text books, e.g. [50, 82], and its solution u^* is usually called the *Godunov state* . With this state u^* the characteristic speeds in (3.22) can be evaluated and used to linearize the governing equation (3.23). As shown in [84], the linearized equation (3.23) also holds for all space derivatives $q^\alpha = D^\alpha u$, $|\alpha| \leq m-1$, where $D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2}}$ is the α-th partial derivative operator. Similar to (3.27) and (3.28), boundary extrapolated values for the derivatives can be defined by

$$q_l^\alpha = \lim_{x \to x_{G_h}^-} D^\alpha p_{in}(x) \tag{3.31}$$

$$q_r^\alpha = \lim_{x \to x_{G_h}^+} D^\alpha p_{out}(x) \tag{3.32}$$

and we can formulate a series of *linear* conventional Riemann problems of the form

$$\text{PDE:} \qquad \frac{\partial}{\partial t} q^\alpha + \lambda_1(u^*) \frac{\partial}{\partial x_1} q^\alpha + \lambda_2(u^*) \frac{\partial}{\partial x_2} q^\alpha = 0\,, \tag{3.33}$$

$$\text{IC:} \qquad q^\alpha(0,x) = \begin{cases} q_l^\alpha & \text{for} \quad x_n < 0 \\ \\ q_r^\alpha & \text{for} \quad x_n > 0\,, \end{cases} \tag{3.34}$$

with constant functions q_l^α and q_r^α given by (3.31) and (3.32). The solution of these linear Riemann problems (3.33),(3.34) is obvious and thus all terms on the right hand side of (3.21) are determined. Therefore, the state $u(t_{G_k}, x_{G_h})$ can be computed through the expansion (3.21) for any local integration point (t_{G_k}, x_{G_h}) in space and time as indicated in Figure 3.5. Note, that one of the fundamental ideas of using an ADERm scheme is to solve an GRP$_{m-1}$ by solving one RP, which is linear or nonlinear depending on the governing equation, and a series of *linear* RPs. The number of these linear RPs is $m - 1$ for one-dimensional problems and $\frac{1}{2}m(m + 1) - 1$ for two-dimensional problems. We remark, that the leading term in the computation of the ADER state in (3.21) is the classical Godunov state u^* itself. The remaining terms in (3.21) are the corrective terms to enhance the approximation quality. Therefore, ADER schemes can be interpreted as high order generalizations of the classical Godunov scheme in [32]. In fact, ADER schemes enable us to evaluate the flux in (3.18) with arbitrary high order accuracy by using an arbitrary high order accurate state u given by (3.21) at the cell interface. Finally, this flux is the information we need in order to update the cell average values as summarized in the following Section.

3.5 Update of Cell Average Values

Recalling the numerical scheme in (3.16) we see, that given the fluxes across all cell boundaries, we can update the cell average values $\bar{u}_\ell^{n+1}$ for time $t = t^{n+1}$ via a one step explicit scheme. In contrast to multi-stage TVD Runge-Kutta schemes, typically used in combination with high order WENO techniques as presented in [19], we do not need the intermediate stages. Therefore, we can reduce the required computational costs quite significantly, as we have to go through the reconstruction procedure only once. We remark that in multi-stage TVD Runge-Kutta schemes one reconstruction step is necessary for each intermediate stage. Furthermore, Ruuth and Spiteri have just shown recently in [71], that TVD Runge-Kutta of arbitrary high order cannot be constructed in a straight forward manner. In contrast, ADER schemes can be extended to arbitrary high order by simply adding higher order terms in (3.21) and therefore the order of accuracy is basically limited by the available computing power.
It is well known, that explicit time discretization schemes, such as the proposed ADER scheme, have to satisfy rather severe restrictions on the time step τ due to the Courant-Friedrich-Levy (CFL) condition[2]. Generally speaking, information from one cell must not interact with information coming from other cells.

[2]CFL conditions on TVD Runge-Kutta schemes can even be more severe [71].

Let ρ_ℓ be the radius of the inscribed circle of a triangular cell T_ℓ serving as a measure of its diameter (see Figure 3.7). And let

$$\lambda^{(max)} = \max_{1 \le j \le 3N_x} |\lambda_{1,j}(u) \cdot n_{1,j} + \lambda_{2,j}(u) \cdot n_{2,j}|$$

be the maximum normal characteristic speed appearing at the $3N_x$ Gaussian integration points along the cell interfaces. As shown in previous work [81, 84, 83] ADER schemes are stable up to a CFL-number of 1 for structured, Cartesian grids. Therefore, we restrict the time step size τ in our computations by a similar CFL-condition

$$\tau \le \min_{1 \le \ell \le \# \mathcal{T}} \frac{\rho_\ell}{\lambda_\ell^{(max)}}, \tag{3.35}$$

for unstructured triangulations. This is similar to the idea in Section 3.4.1 of [55].
Altogether, the different computational steps discussed in the previous Sections 3.3 and 3.4 are combined in order to construct a conservative ADERm scheme, that can be used to solve linear and nonlinear conservation laws. For each time step $t \to t + \tau$ we can formulate the following algorithm:

Algorithm 2 (Conservative ADERm scheme).

INPUT: *Triangulation $\mathcal{T}$ with cell average values $\bar{u}_\ell(t)$, $\ell = 1, ..., \#\mathcal{T}$, time step size $\tau > 0$ satisfying (3.35), and the desired order m.*

- *Compute polynomial functions p_ℓ of degree m–1 from cell average values $\bar{u}_\ell(t)$ satisfying (3.4) by the WENO approach (3.10), (3.12).*
- **FOR** *each $T_\ell \in \mathcal{T}$* **DO**
  - *Use Lax-Wendroff procedure (3.24) - (3.26) to replace time derivatives by space derivatives in (3.21).*
  - *Solve one-dimensional GRP_{m-1} in (3.19), (3.20) at Gaussian integration points x_{G_h} at cell edges.*
  - *Evaluate the solution u at Gaussian integration points t_{G_k} in time via (3.21).*
  - *Compute numerical fluxes $\hat{F}_{\ell,j}$, $j = 1, 2, 3$, via (3.18).*
  - *Update cell average values $\bar{u}_\ell(t + \tau)$ using (3.16).*

OUTPUT: *Updated cell average values $\bar{u}_\ell(t + \tau)$ for all $T_\ell \in \mathcal{T}$.*

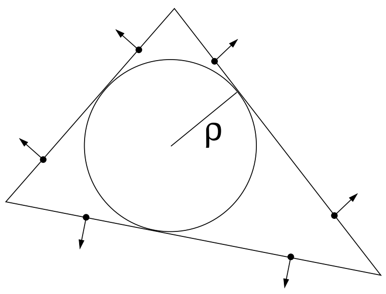


Figure 3.7: The radius ρ of the inscribed circle of a triangular cell together with its $N_x = 2$ integration points per edge, where the maximum characteristic speed normal to each edge is evaluated.

3.6 Boundary Conditions

Boundary conditions are an important issue, when designing a numerical method. In practice, the computations are carried out on a finite set of cells T_ℓ, $\ell = 1, ..., \#\mathcal{T}$, covering the bounded domain $\Omega \subset \mathbb{R}^2$. Therefore, cells at the boundary of Ω may not have the required neighbours in order to reconstruct the required polynomials for the computation of the intercell fluxes. In general, there are various possibilities to handle this problem.
Due to some physical boundary condition, the fluxes at the Gaussian integration points x_{G_h} can be specified directly. However, a more common approach is to extend the computational domain Ω and include a few additional cells, so-called *ghost cells*, outside the original boundary, whose cell average values have to be set at the beginning of each time step. This way, the reconstruction and updating of cell averages is exactly the same for all interior cells and there is no need to develop special methods for boundary data. Instead we only have to decide, how to set the cell average values of the ghost cells, which is entirely independent of the choice of the applied numerical method.

3.6.1 Periodic Boundaries

Periodic boundary conditions usually are easy to apply by copying cell average data from one side of the computational mesh to the cells on the opposite side. The number of necessary ghost cells with their corresponding average values from the opposite side depends on the size of the stencil. The required stencil size, in turn depends on the desired order of the scheme. We remark, that periodic boundary conditions on unstructured triangulations must be handled with particular care, as the same number of cell edges must occur on opposite sides of the triangulation. In our examples, we usually work with a computational domain Ω, that is bounded by a rectangular. However, we point out that this is not necessarily required and boundaries with any

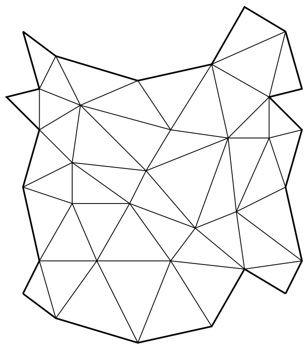


Figure 3.8: Domains with polygonal boundaries, that periodically match on opposite sides.

polygonal shape can be used as long as the opposite sides of the mesh fit together in a periodic manner as schematically displayed in Figure 3.8.

3.6.2 Inflow Boundaries

The treatment of inflow boundary conditions for one-dimensional problems is discussed in detail in Chapter 7 of [50]. In order to handle inflow boundary conditions for unstructured triangulations, the use of ghost cells can become very sophisticated. In fact, standard methods of evaluating cell average values on these ghost cells through dimensional splitting as normally used on Cartesian meshes are not carried over in a straight forward manner. Therefore, for a given boundary condition

$$u(s,t) = b(s,t)\,,$$

where s parameterizes the boundary, we suggest to evaluate the function b at the Gaussian integration points along the boundary s and compute the numerical flux in (3.16) directly, i.e. $\hat{F}^n_{\ell,j} = f(b(s,t^n))$.

3.6.3 Outflow Boundaries

As problems have to be solved on a bounded domain $\Omega \subset \mathbb{R}^2$, we create artificial computational boundaries, that do not exist in physical space. Therefore, we often want to have no incoming flow, though there may be outgoing flow that should leave the domain cleanly without any disturbance at these boundaries. As discussed in [50, 82], ghost cells, whose cell average values can be obtained by extrapolation from the interior solution, are again used in order to implement outflow boundary conditions for Cartesian meshes.
In the case of unstructured triangular meshes, we accomplish similar outflow

boundaries in a slightly different way. First we detect all Gaussian integration points x_{G_h}, where flow is leaving the domain. At these integration points we set the boundary extrapolated value u_r, that cannot be obtained from a polynomial of the outside of the cell at the boundary, equal to u_l, the boundary extrapolated value from the inside. Therefore, we have to solve a trivial Riemann problem with $u_l = u_r$, which does not produce any flow inside the domain. In the literature, such outflow boundaries are often called *transmissive* or *open* boundaries and work quite effectively. For further details on different boundary conditions see for example [6, 26].

3.7 Performance of ADER schemes

In the last few years, ADER schemes were developed and analysed mainly for one-dimensional linear and nonlinear problems [81, 84, 83] and applied to multi-dimensional problems on fixed rectangular Cartesian meshes, e.g. in [73, 72]. Here we investigate the performance of the proposed ADER schemes for linear and nonlinear problems on unstructured triangulations by determining their convergence properties numerically. Furthermore, we consider their efficiency with respect to computational cost depending on the order of the scheme.

3.7.1 Experimental Orders of Convergence

In this section, the experimental order of convergence of the proposed ADER schemes on two-dimensional linear and nonlinear advection problems are determined numerically in order to compare them with the theoretically expected orders.

Linear Advection: For the linear problem we solve the two-dimensional equation

$$u_t + u_{x_1} + u_{x_2} = 0\,, \tag{3.36}$$

a linear example of the general equation (3.1) with the initial condition

$$u_0(x) = u(0,x) = \sin\Big(2\pi(x_1 + x_2)\Big) \tag{3.37}$$

on the computational domain $\Omega = [-0.5, 0.5] \times [-0.5, 0.5]$. The computations are carried out for the time interval $I = [0, 1]$. We remark, that we use periodic boundary conditions, such that the reference solution $\tilde{u}(1, x)$ at the end of the simulation time $t = 1$, is identical to the initial condition (3.37), i.e. $u_0(x) \equiv u(1, x)$.

In order to study the influence of the mesh irregularity, i.e. the distortion of the mesh on the accuracy of the numerical results, we compute the solution

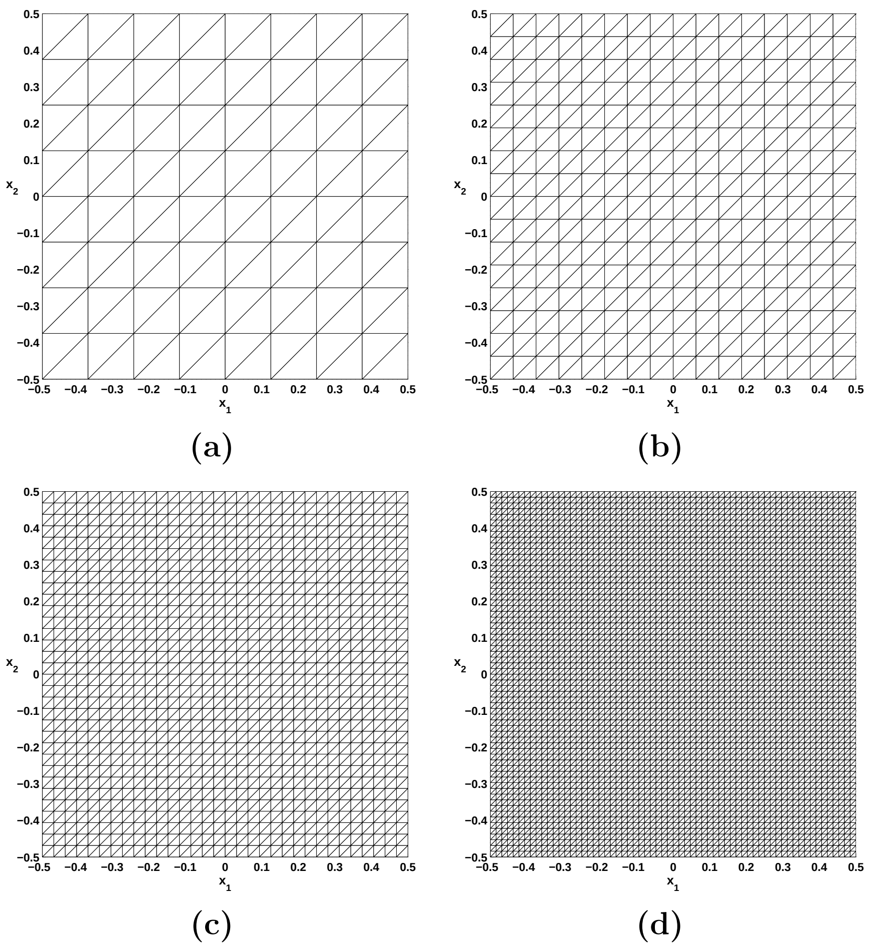


Figure 3.9: The sequence of the four regular meshes with their mesh widths **(a)** $\mathbf{A}_0$ ($h = 0.125$), **(b)** $\mathbf{A}_1$ ($h = 0.0625$), **(c)** $\mathbf{A}_2$ ($h = 0.03125$), **(d)** $\mathbf{A}_3$ ($h = 0.015625$).

of (3.36) on sequences of three different triangular meshes. Mesh **A** is a regular mesh obtained by adding the diagonal line in each square (see Figure 3.9), mesh **B** is an irregular mesh (see Figure 3.10) obtained by slightly distorting mesh **A**, and mesh **C** is a strongly distorted irregular mesh (see Figure 3.11). All sequences of the three meshes consist of five successive refinement levels and are constructed by uniformly refining the coarsest mesh, namely by subdividing each triangular cells into four similar smaller ones. The refinement level of a particular mesh is indicated by subscripts, e.g. $\mathbf{A}_0$ denotes the original mesh **A**, whereas $\mathbf{C}_3$ indicates the third refinement of mesh **C**, as displayed in Figures 3.9, 3.10, and 3.11. for the first four refinement levels.

Note, that only for the regular mesh **A** the mesh spacing h is representative for the entire mesh, whereas h can only be a rough indicator of the mesh

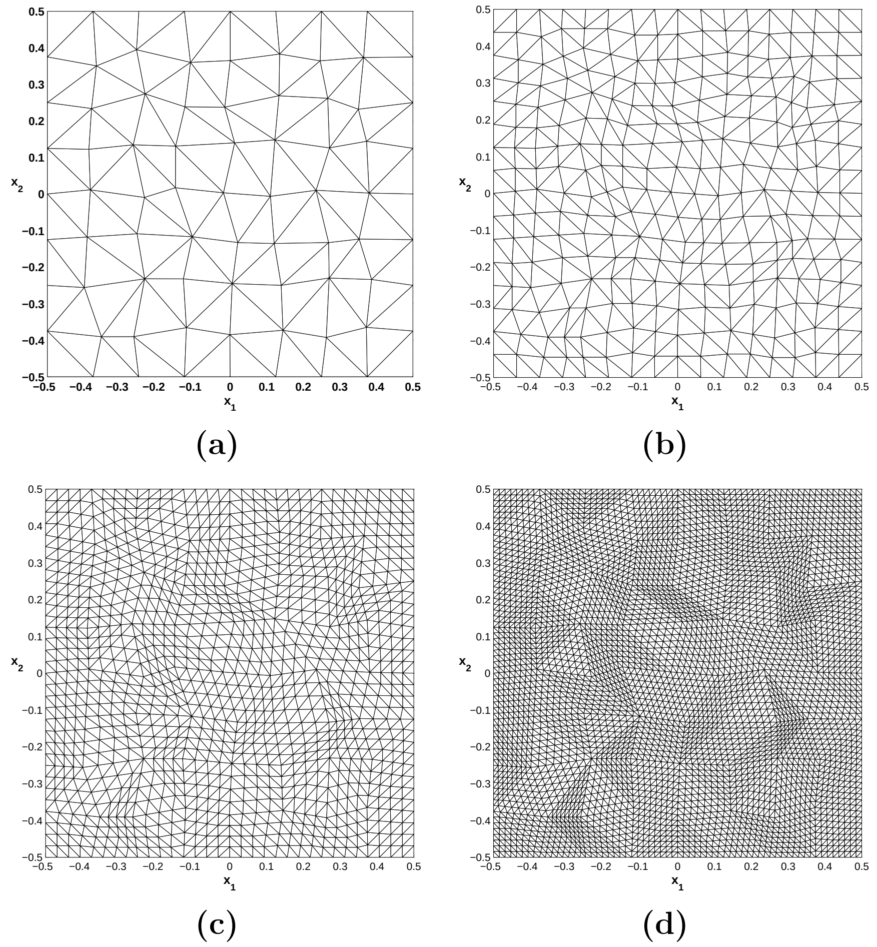


Figure 3.10: The sequence of the four slightly irregular meshes with their mesh widths **(a)** $\mathbf{B}_0$ ($h = 0.125$), **(b)** $\mathbf{B}_1$ ($h = 0.0625$), **(c)** $\mathbf{B}_2$ ($h = 0.03125$), **(d)** $\mathbf{B}_3$ ($h = 0.015625$).

width for meshes **B** and **C**. However, all meshes consist of the same number of cells in the corresponding refinement level in order to keep the computational cost independent of the the mesh irregularity.
All computations are carried out for ADER2, ADER3, and ADER4 schemes, where we use nine stencils as constructed by the WENO reconstruction procedure of Section 3.3.2 , i.e. three *centered* stencils, three stencils in the forwards sectors $\mathcal{F}_j$ (see Figure 3.2), and three in the backwards sectors $\mathcal{B}_j$ (see Figure 3.3). The stencils consist of 4, 8, or 13 cells for the ADER2, ADER3, or ADER4 schemes, respectively. The time step τ is set to $\tau = 0.025$ for the computations on meshes $\mathbf{A}_0$ and $\mathbf{B}_0$ and to $\tau = 0.0125$ on mesh $\mathbf{C}_0$. With successive refinement levels the time step τ is halfed accordingly.

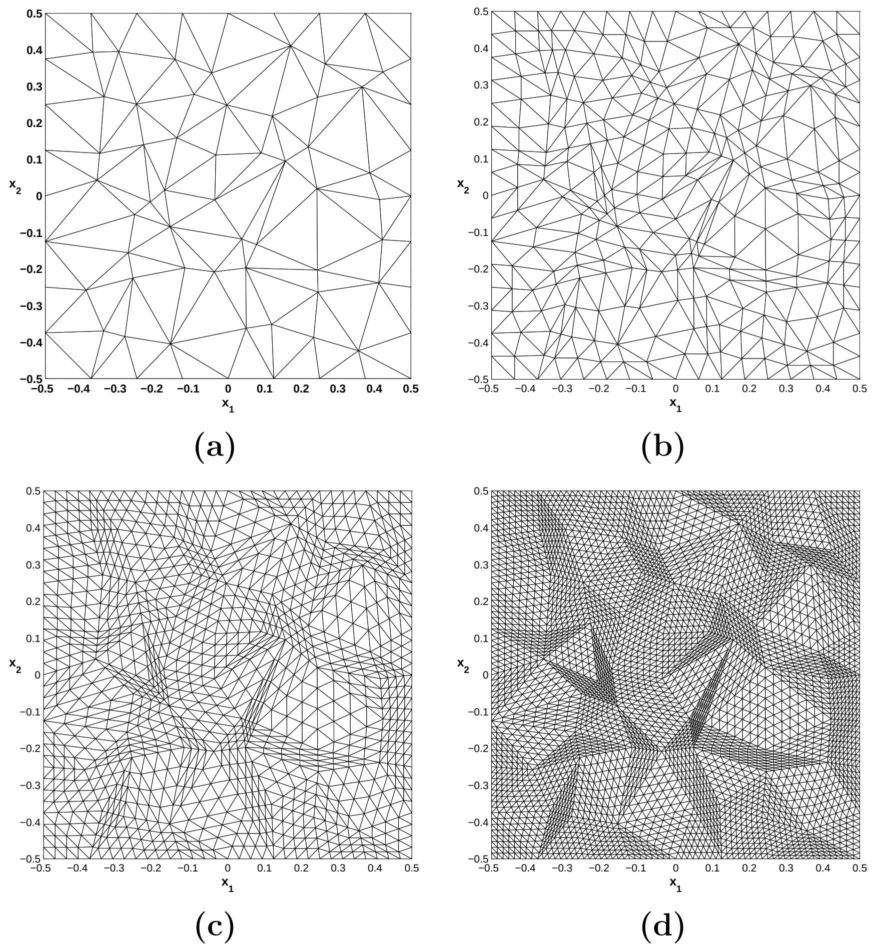


Figure 3.11: The sequence of the four strongly irregular meshes with their mesh widths **(a)** $\mathbf{C}_0$ ($h = 0.125$), **(b)** $\mathbf{C}_1$ ($h = 0.0625$), **(c)** $\mathbf{C}_2$ ($h = 0.03125$), **(d)** $\mathbf{C}_3$ ($h = 0.015625$).

We remark, that the errors presented are those of the cell averages $\bar{u}$ of the solution u and the reference solution $\tilde{u}$, which are computed by a 7-points quadrature rule for triangles (see Appendix C).
Table 3.1 shows the results for the errors

$$E_p(h) = \|u - \tilde{u}\|_p \,, \tag{3.38}$$

for the norms $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$, where $\tilde{u}$ is the reference solution, together with the corresponding convergence orders k_p

$$k_p = \frac{\log\Big(E_p(h) \,/\, E_p(h/2)\Big)}{\log(2)} \,, \tag{3.39}$$

obtained by ADER2, ADER3, and ADER4 schemes on the sequence of regular meshes $\mathbf{A}_0$ to $\mathbf{A}_4$.

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$3.1024 \cdot 10^{-1}$	–	$3.5613 \cdot 10^{-1}$	–	$5.0293 \cdot 10^{-1}$	–
1/16	$8.9463 \cdot 10^{-2}$	1.79	$9.9535 \cdot 10^{-2}$	1.84	$1.4043 \cdot 10^{-1}$	1.84
1/32	$2.2632 \cdot 10^{-2}$	1.98	$2.5127 \cdot 10^{-2}$	1.99	$3.5492 \cdot 10^{-2}$	1.98
1/64	$5.6576 \cdot 10^{-3}$	2.00	$6.2828 \cdot 10^{-3}$	2.00	$8.8815 \cdot 10^{-3}$	2.00
1/128	$1.4139 \cdot 10^{-3}$	2.00	$1.5703 \cdot 10^{-3}$	2.00	$2.2205 \cdot 10^{-3}$	2.00

(a)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$3.4715 \cdot 10^{-1}$	–	$4.0503 \cdot 10^{-1}$	–	$5.8055 \cdot 10^{-1}$	–
1/16	$5.0290 \cdot 10^{-2}$	2.79	$5.7626 \cdot 10^{-2}$	2.81	$8.1474 \cdot 10^{-2}$	2.83
1/32	$6.1105 \cdot 10^{-3}$	3.04	$6.7352 \cdot 10^{-3}$	3.10	$9.1247 \cdot 10^{-3}$	3.16
1/64	$6.1757 \cdot 10^{-4}$	3.31	$6.8527 \cdot 10^{-4}$	3.30	$9.5297 \cdot 10^{-4}$	3.26
1/128	$7.1885 \cdot 10^{-5}$	3.10	$7.9826 \cdot 10^{-5}$	3.10	$1.1223 \cdot 10^{-4}$	3.09

(b)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$1.1148 \cdot 10^{-1}$	–	$1.3002 \cdot 10^{-1}$	–	$1.9584 \cdot 10^{-1}$	–
1/16	$3.9053 \cdot 10^{-3}$	4.84	$4.4563 \cdot 10^{-3}$	4.87	$8.8605 \cdot 10^{-3}$	4.47
1/32	$2.2444 \cdot 10^{-4}$	4.12	$2.4633 \cdot 10^{-4}$	4.18	$3.4535 \cdot 10^{-4}$	4.68
1/64	$1.4011 \cdot 10^{-5}$	4.00	$1.5064 \cdot 10^{-5}$	4.03	$2.0771 \cdot 10^{-5}$	4.06
1/128	$8.2222 \cdot 10^{-7}$	4.09	$9.1268 \cdot 10^{-7}$	4.04	$1.3338 \cdot 10^{-6}$	3.96

(c)

Table 3.1: Results for the linear advection obtained by **(a)** ADER2, **(b)** ADER3, and **(c)** ADER4 schemes on the regular meshes $\mathbf{A}_0$ to $\mathbf{A}_4$.

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$1.1265 \cdot 10^{-1}$	–	$1.2826 \cdot 10^{-1}$	–	$2.7656 \cdot 10^{-1}$	–
1/16	$4.2780 \cdot 10^{-2}$	1.40	$4.8948 \cdot 10^{-2}$	1.39	$1.0326 \cdot 10^{-1}$	1.42
1/32	$1.1288 \cdot 10^{-2}$	1.92	$1.2915 \cdot 10^{-2}$	1.92	$2.6589 \cdot 10^{-2}$	1.96
1/64	$2.6513 \cdot 10^{-3}$	2.42	$3.0153 \cdot 10^{-3}$	2.43	$1.1444 \cdot 10^{-2}$	1.41
1/128	$6.3234 \cdot 10^{-4}$	2.13	$7.1838 \cdot 10^{-4}$	2.14	$3.7882 \cdot 10^{-3}$	1.65

(a)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$1.4226 \cdot 10^{-1}$	–	$1.6078 \cdot 10^{-1}$	–	$2.7919 \cdot 10^{-1}$	–
1/16	$1.6160 \cdot 10^{-2}$	3.14	$1.8617 \cdot 10^{-2}$	3.11	$3.9276 \cdot 10^{-2}$	2.83
1/32	$1.5446 \cdot 10^{-3}$	3.39	$1.8346 \cdot 10^{-3}$	3.34	$4.2469 \cdot 10^{-3}$	3.21
1/64	$2.0259 \cdot 10^{-4}$	3.40	$2.2524 \cdot 10^{-4}$	3.51	$4.2128 \cdot 10^{-4}$	3.87
1/128	$2.4139 \cdot 10^{-5}$	3.17	$2.6835 \cdot 10^{-5}$	3.17	$5.1008 \cdot 10^{-5}$	3.14

(b)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$2.9912 \cdot 10^{-2}$	–	$3.4907 \cdot 10^{-2}$	–	$7.2935 \cdot 10^{-2}$	–
1/16	$1.1801 \cdot 10^{-3}$	4.66	$1.5787 \cdot 10^{-3}$	4.47	$5.2470 \cdot 10^{-3}$	3.80
1/32	$6.9519 \cdot 10^{-5}$	4.09	$8.9930 \cdot 10^{-5}$	4.13	$3.2150 \cdot 10^{-4}$	4.03
1/64	$6.4714 \cdot 10^{-6}$	3.97	$8.0984 \cdot 10^{-6}$	4.03	$3.1137 \cdot 10^{-5}$	3.91
1/128	$4.4070 \cdot 10^{-7}$	4.00	$5.5669 \cdot 10^{-7}$	3.99	$2.2974 \cdot 10^{-6}$	3.88

(c)

Table 3.2: Results for the linear advection obtained by **(a)** ADER2, **(b)** ADER3, and **(c)** ADER4 schemes on the slightly irregular meshes $\mathbf{B}_0$ to $\mathbf{B}_4$.

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$1.3924 \cdot 10^{-1}$	–	$1.6233 \cdot 10^{-1}$	–	$3.9986 \cdot 10^{-1}$	–
1/16	$3.2158 \cdot 10^{-2}$	2.11	$3.8800 \cdot 10^{-2}$	2.06	$1.4476 \cdot 10^{-1}$	1.47
1/32	$6.8809 \cdot 10^{-3}$	2.22	$8.3858 \cdot 10^{-3}$	2.21	$3.9424 \cdot 10^{-2}$	1.88
1/64	$1.6080 \cdot 10^{-3}$	2.10	$1.9787 \cdot 10^{-3}$	2.08	$1.0345 \cdot 10^{-2}$	1.93
1/128	$3.8924 \cdot 10^{-4}$	2.05	$4.8469 \cdot 10^{-4}$	2.03	$3.1769 \cdot 10^{-3}$	1.70

(a)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$2.7500 \cdot 10^{-1}$	–	$3.0955 \cdot 10^{-1}$	–	$4.9177 \cdot 10^{-1}$	–
1/16	$3.8493 \cdot 10^{-2}$	2.84	$4.4821 \cdot 10^{-2}$	2.79	$9.5172 \cdot 10^{-2}$	2.37
1/32	$4.5424 \cdot 10^{-3}$	3.08	$5.3011 \cdot 10^{-3}$	3.08	$1.1456 \cdot 10^{-2}$	3.05
1/64	$5.2333 \cdot 10^{-4}$	3.12	$6.0649 \cdot 10^{-4}$	3.13	$1.2106 \cdot 10^{-3}$	3.24
1/128	$6.1609 \cdot 10^{-5}$	3.09	$7.1088 \cdot 10^{-5}$	3.09	$1.4629 \cdot 10^{-4}$	3.05

(b)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$6.6326 \cdot 10^{-2}$	–	$7.9679 \cdot 10^{-2}$	–	$1.5932 \cdot 10^{-1}$	–
1/16	$3.9170 \cdot 10^{-3}$	4.08	$5.2793 \cdot 10^{-3}$	3.92	$1.3527 \cdot 10^{-2}$	3.56
1/32	$2.0676 \cdot 10^{-4}$	4.24	$2.7034 \cdot 10^{-4}$	4.29	$8.8686 \cdot 10^{-4}$	3.93
1/64	$1.3002 \cdot 10^{-5}$	3.99	$1.5726 \cdot 10^{-5}$	4.10	$5.3229 \cdot 10^{-5}$	4.06
1/128	$7.7907 \cdot 10^{-7}$	4.06	$9.5160 \cdot 10^{-7}$	4.05	$3.7559 \cdot 10^{-6}$	3.82

(c)

Table 3.3: Results for the linear advection obtained by **(a)** ADER2, **(b)** ADER3, and **(c)** ADER4 schemes on the strongly irregular meshes $\mathbf{C}_0$ to $\mathbf{C}_4$.

Tables 3.2 and 3.3 show the corresponding results obtained on the sequences of the slightly and strongly irregular meshes **B** and **C**.
Note, that the ADER schemes reach the expected orders of convergence k_p in (3.39) in all norms and on all meshes. However, a closer look at the errors $E_p(h)$ from (3.38) shows, that the slightly irregular meshes in Table 3.2 give the best results. We think, that this is due to the fact, that many of the triangles of the mesh sequence $\mathbf{B}_0$ to $\mathbf{B}_4$ are closer to being equilateral than in the other cases of meshes **A** and **C**. As shown in [5, 45], the shape of triangular cells plays an important role for the accuracy in the sense that simulation results obtained on triangular meshes of equilateral triangles are more accurate than those obtained on other non-equilateral cells. We remark, that these results are confirmed by our tests on ADER schemes.
On the other hand, it is obvious that the even for the strongly distorted mesh **C** in Figure 3.11 we still get very satisfying results, indicating that the proposed ADER scheme combined with the discussed WENO reconstruction technique seems to be very useful and robust approach applicable to strongly distorted unisotropic meshes.

Nonlinear Advection: The nonlinear problem is constituted by the two-dimensional Burgers equation

$$u_t + \left(\frac{1}{2}u^2\right)_{x_1} + \left(\frac{1}{2}u^2\right)_{x_2} = 0\,, \tag{3.40}$$

a nonlinear example of equation (3.1) with the initial condition

$$u_0(x) = u(0,x) = 0.3 + 0.7\sin\Big(2\pi(x_1 + x_2)\Big) \tag{3.41}$$

on the computational domain $\Omega = [-0.5, 0.5] \times [-0.5, 0.5]$. The computations are carried out for the time interval $I = [0, \frac{1}{4\pi}]$, such that no discontinuity has developed yet, i.e. the solution is still smooth at the end of the simulation time. Again, periodic boundary conditions are used. Note, that the initial condition (3.41) leads to a transonic rarefaction.
Then, the same sequences of meshes **A**, **B**, and **C** as shown in Figure 3.9, 3.10, and 3.11 are used similar to the linear case. The cell averages of reference solution $\tilde{u}$ are calculated via the 7-points quadrature rule (see Appendix C), where the value at each quadrature point is calculated via Newton's method.

Analogously to the linear case, Tables 3.4, 3.5, and 3.6 show the errors $E_p(h)$ from (3.38) of the cell averages at the end of the simulation together with the experimental orders of convergence k_p in (3.39) for the ADER2, ADER3, and ADER4 schemes on the meshes $\mathbf{A}_0$ to $\mathbf{A}_4$, $\mathbf{B}_0$ to $\mathbf{B}_4$, and $\mathbf{C}_0$ to $\mathbf{C}_4$, respectively.

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$1.3523 \cdot 10^{-2}$	–	$1.5406 \cdot 10^{-2}$	–	$4.5874 \cdot 10^{-2}$	–
1/16	$3.4015 \cdot 10^{-3}$	1.99	$4.2354 \cdot 10^{-3}$	1.86	$9.9464 \cdot 10^{-3}$	2.21
1/32	$8.1563 \cdot 10^{-4}$	2.06	$1.0681 \cdot 10^{-3}$	1.99	$2.6907 \cdot 10^{-3}$	1.89
1/64	$1.9851 \cdot 10^{-4}$	2.04	$2.6442 \cdot 10^{-4}$	2.01	$6.7492 \cdot 10^{-4}$	2.00
1/128	$4.8844 \cdot 10^{-5}$	2.02	$6.5561 \cdot 10^{-5}$	2.01	$1.6728 \cdot 10^{-4}$	2.01

(a)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$2.4460 \cdot 10^{-2}$	–	$2.6815 \cdot 10^{-2}$	–	$4.6138 \cdot 10^{-2}$	–
1/16	$3.2812 \cdot 10^{-3}$	2.90	$3.9929 \cdot 10^{-3}$	2.75	$8.3484 \cdot 10^{-3}$	2.47
1/32	$3.2445 \cdot 10^{-4}$	3.34	$4.4015 \cdot 10^{-4}$	3.18	$1.0869 \cdot 10^{-3}$	2.94
1/64	$3.3403 \cdot 10^{-5}$	3.28	$4.6285 \cdot 10^{-5}$	3.25	$1.2003 \cdot 10^{-4}$	3.18
1/128	$3.9009 \cdot 10^{-6}$	3.10	$5.4216 \cdot 10^{-6}$	3.09	$1.4117 \cdot 10^{-5}$	3.09

(b)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$6.4417 \cdot 10^{-3}$	–	$8.5263 \cdot 10^{-3}$	–	$2.1361 \cdot 10^{-2}$	–
1/16	$3.2915 \cdot 10^{-4}$	4.29	$4.9585 \cdot 10^{-4}$	4.10	$1.3426 \cdot 10^{-3}$	3.99
1/32	$2.0740 \cdot 10^{-5}$	3.99	$3.5896 \cdot 10^{-5}$	3.79	$1.2353 \cdot 10^{-4}$	3.44
1/64	$1.6025 \cdot 10^{-6}$	3.69	$2.6907 \cdot 10^{-6}$	3.74	$9.0486 \cdot 10^{-6}$	3.77
1/128	$1.1391 \cdot 10^{-7}$	3.81	$1.8737 \cdot 10^{-7}$	3.84	$5.7273 \cdot 10^{-7}$	3.98

(c)

Table 3.4: Results for Burgers equation obtained by **(a)** ADER2, **(b)** ADER3, and **(c)** ADER4 schemes on the regular meshes $\mathbf{A}_0$ to $\mathbf{A}_4$.

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$1.4816 \cdot 10^{-2}$	–	$2.1592 \cdot 10^{-2}$	–	$8.9534 \cdot 10^{-2}$	–
1/16	$5.0152 \cdot 10^{-3}$	1.56	$6.8720 \cdot 10^{-3}$	1.65	$3.2865 \cdot 10^{-2}$	1.45
1/32	$1.3421 \cdot 10^{-3}$	1.90	$1.8877 \cdot 10^{-3}$	1.86	$1.0561 \cdot 10^{-2}$	1.64
1/64	$3.4067 \cdot 10^{-4}$	1.98	$4.8618 \cdot 10^{-4}$	1.96	$2.7014 \cdot 10^{-3}$	1.97
1/128	$8.3667 \cdot 10^{-5}$	2.03	$1.2018 \cdot 10^{-4}$	2.02	$7.0141 \cdot 10^{-4}$	1.95

(a)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$1.2429 \cdot 10^{-2}$	–	$1.5481 \cdot 10^{-2}$	–	$4.7784 \cdot 10^{-2}$	–
1/16	$1.6329 \cdot 10^{-3}$	2.93	$2.2922 \cdot 10^{-3}$	2.76	$1.0174 \cdot 10^{-2}$	2.23
1/32	$1.9838 \cdot 10^{-4}$	3.04	$3.0528 \cdot 10^{-4}$	2.91	$2.1328 \cdot 10^{-3}$	2.25
1/64	$2.7484 \cdot 10^{-5}$	3.31	$4.0679 \cdot 10^{-5}$	3.37	$2.8764 \cdot 10^{-4}$	3.35
1/128	$3.5762 \cdot 10^{-6}$	3.04	$5.1999 \cdot 10^{-6}$	3.06	$4.9262 \cdot 10^{-5}$	2.63

(b)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$2.9430 \cdot 10^{-3}$	–	$3.9772 \cdot 10^{-3}$	–	$1.6612 \cdot 10^{-2}$	–
1/16	$2.2322 \cdot 10^{-4}$	3.72	$3.5916 \cdot 10^{-4}$	3.47	$1.5177 \cdot 10^{-3}$	3.45
1/32	$1.9599 \cdot 10^{-5}$	3.51	$3.7513 \cdot 10^{-5}$	3.26	$2.7872 \cdot 10^{-4}$	2.44
1/64	$1.7003 \cdot 10^{-6}$	4.09	$2.9834 \cdot 10^{-6}$	4.24	$2.9170 \cdot 10^{-5}$	3.78
1/128	$1.3478 \cdot 10^{-7}$	3.78	$2.4466 \cdot 10^{-7}$	3.72	$2.6691 \cdot 10^{-6}$	3.56

(c)

Table 3.5: Results for Burgers equation obtained by **(a)** ADER2, **(b)** ADER3, and **(c)** ADER4 schemes on the regular meshes $\mathbf{B}_0$ to $\mathbf{B}_4$.

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$2.4789 \cdot 10^{-2}$	–	$3.5598 \cdot 10^{-2}$	–	$1.2987 \cdot 10^{-1}$	–
1/16	$8.1998 \cdot 10^{-3}$	1.60	$1.1486 \cdot 10^{-2}$	1.63	$6.5593 \cdot 10^{-2}$	0.99
1/32	$2.2506 \cdot 10^{-3}$	1.87	$3.2835 \cdot 10^{-3}$	1.81	$2.7181 \cdot 10^{-2}$	1.27
1/64	$5.5952 \cdot 10^{-4}$	2.01	$8.4517 \cdot 10^{-4}$	1.96	$9.1484 \cdot 10^{-3}$	1.57
1/128	$1.3480 \cdot 10^{-4}$	2.05	$2.0520 \cdot 10^{-4}$	2.04	$2.5284 \cdot 10^{-3}$	1.86

(a)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$2.1345 \cdot 10^{-2}$	–	$2.7487 \cdot 10^{-2}$	–	$8.6973 \cdot 10^{-2}$	–
1/16	$3.0335 \cdot 10^{-3}$	2.81	$4.4508 \cdot 10^{-3}$	2.63	$1.9878 \cdot 10^{-2}$	2.13
1/32	$3.8506 \cdot 10^{-4}$	2.98	$6.4792 \cdot 10^{-4}$	2.78	$5.4981 \cdot 10^{-3}$	1.85
1/64	$4.5916 \cdot 10^{-5}$	3.07	$7.6192 \cdot 10^{-5}$	3.09	$6.2541 \cdot 10^{-4}$	3.14
1/128	$5.5909 \cdot 10^{-6}$	3.04	$9.2328 \cdot 10^{-6}$	3.04	$8.5906 \cdot 10^{-5}$	2.86

(b)

h	$E_1(h)$	k_1	$E_2(h)$	k_2	$E_\infty(h)$	k_∞
1/8	$5.6973 \cdot 10^{-3}$	–	$8.1636 \cdot 10^{-3}$	–	$4.3144 \cdot 10^{-2}$	–
1/16	$5.1513 \cdot 10^{-4}$	3.47	$9.2607 \cdot 10^{-4}$	3.14	$5.0294 \cdot 10^{-3}$	3.10
1/32	$3.9238 \cdot 10^{-5}$	3.71	$7.8427 \cdot 10^{-5}$	3.56	$6.1687 \cdot 10^{-4}$	3.03
1/64	$2.7966 \cdot 10^{-6}$	3.81	$6.0176 \cdot 10^{-6}$	3.70	$5.2142 \cdot 10^{-5}$	3.56
1/128	$1.8851 \cdot 10^{-7}$	3.89	$4.4105 \cdot 10^{-7}$	3.77	$5.6319 \cdot 10^{-6}$	3.21

(c)

Table 3.6: Results for Burgers equation obtained by **(a)** ADER2, **(b)** ADER3, and **(c)** ADER4 schemes on the regular meshes $\mathbf{C}_0$ to $\mathbf{C}_4$.

In analogy to the results for the linear problem the considered ADER schemes achieve the expected orders of convergence. Only the orders k_∞ of the ADER4 scheme on the strongly distorted mesh sequence $\mathbf{C}_0$ to $\mathbf{C}_4$ seem not to reach the expected order of 4 (see Table 3.6(**c**)).
In our opinion, the reason for that might be, that the appearance of thin, stretched triangles can lead to reconstruction polynomials of rather poor approximation quality due to degenerated one-sided stencils. The stencil construction discussed in Section 3.3.2 uses a sectoral search of subsequent von Neumann neighbours of increasing levels. Therefore, a very thin triangular cell with a very small angle leads to a very narrow sectors $\mathcal{F}_j$ and $\mathcal{B}_j$, which in turn result in very elongated stencils with shapes preferring a particular direction. The resulting reconstruction of rather low approximation quality then influences the computation of the fluxes and finally causes errors that appear especially in the $\|\cdot\|_\infty$ norm.
This effect, however, was not seen in the linear case. We believe, that in the linear case, the applied mesh sequences were fine enough to discretise the smooth solution u to reach the expected convergence orders. In the nonlinear example, the simulation time $T = \frac{1}{4\pi}$ was chosen to keep the solution u smooth, however, gradients have already steepened. We think, that these gradients are not sufficiently fine resolved by the mesh sequence $\mathbf{C}_0$ to $\mathbf{C}_4$ and therefore, the expected order is not quite reached. Further mesh refinement should help to eliminate this effect.

3.7.2 Computational Efficiency

An important consideration, when applying numerical schemes to particular problems is their *computational efficiency*, which is a measure of reaching a desired accuracy in a particular computing time. In general, there are two possibilities to enhance the accuracy of a given scheme.
First, the same scheme can be used on a finer discretisation, i.e. on smaller cells, or secondly a higher order version of the scheme can be used. Either possibility leads to an increase in computation time. Therefore, it is important to investigate which option provides the desired accuracy in less computational time. In other words the following question has to be answered: Is it more efficient to use a simple and fast low-order scheme on fine meshes or to use a more sophisticated and slower scheme of higher order on rather coarse meshes?
In this section we numerically evaluate the proposed ADER schemes with respect to computing time and achieved accuracy which should help to answer the above question. Therefore, we record the CPU time used by the different ADER schemes in order to compute the various steps necessary to complete one time step. As an example we choose the computation of the solution of the nonlinear advection equation (3.40) with the four ADER schemes on the

slightly irregular mesh $\mathbf{B}_2$ shown in Figure 3.10(**c**), which consists of 2048 fixed cells. We remark, that all computations have been carried out with MATLAB 6 Release 13 on a PC (model: IBM 236623G) with processor type Intel Pentium(R) 4 1600MHz.

Table 3.7 shows the CPU times in seconds needed to complete the different steps of each ADERm scheme, $m = 1, ..., 4$. Here, t_s denotes the time in CPU seconds required to construct the stencils, t_r is the time to compute the reconstruction polynomials , and t_o is the time required for all other computations, such as the flux evaluation and the update of cell averages. Note, that t_o also includes the Lax-Wendroff procedure described in Section 3.4.2 to replace the time derivatives by space derivatives. The total time t_{tot} indicates the required CPU seconds in order to complete one time step. It

	ADER1		ADER2		ADER3		ADER4	
	CPU sec	%	CPU sec	%	CPU sec	%	CPU sec	%
t_s	0	0	0.058	2	0.208	6	0.467	9
t_r	0	0	2.429	83	2.948	82	4.001	81
t_o	0.426	100	0.430	15	0.448	12	0.481	10
t_{tot}	0.426	100	2.917	100	3.604	100	4.949	100

Table 3.7: CPU seconds for the different computational step of ADER schemes.

ADER1	ADER2	ADER3	ADER4
1	6.8	8.5	11.6
	1	1.2	1.7
		1	1.4

Table 3.8: Factors indicating the slowdown of ADER schemes.

is obvious, that with increasing order of accuracy m an ADERm scheme becomes more expensive. Note, that the main contribution to the increasing total CPU times t_{tot} is caused by t_s for the stencil construction and t_r for the reconstruction, whereas is increase in t_o is almost negligible.
Table 3.8 shows the factors the indicate the slowdown of an ADER scheme, when increasing the order of accuracy. The factors represent the ratios of the times t_{tot} normalized to ADER1 (first row), ADER2 (second row), or ADER3 (third row). For example, we can read from Table 3.8 that the CPU time for an ADER4 scheme is 11.6 times than that of an ADER1 scheme, or 1.7 times larger than that of an ADER2 scheme, etc.

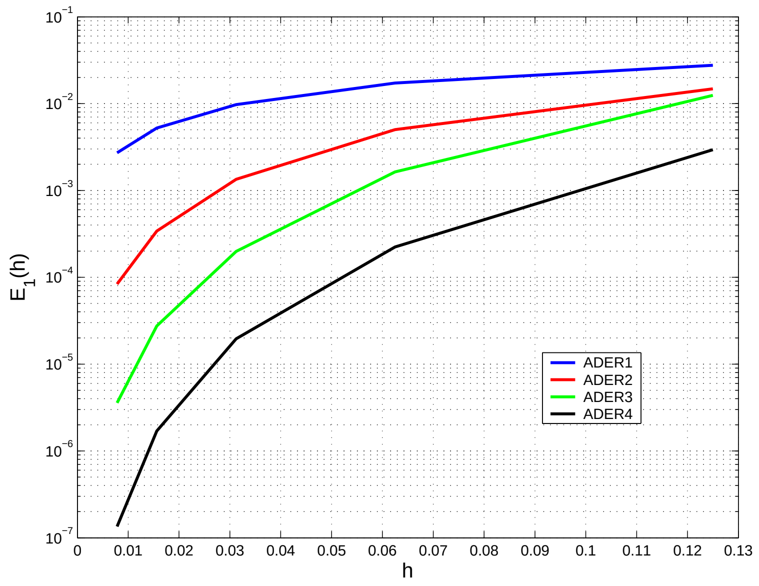


Figure 3.12: Accuracy of the ADER schemes for different mesh width h.

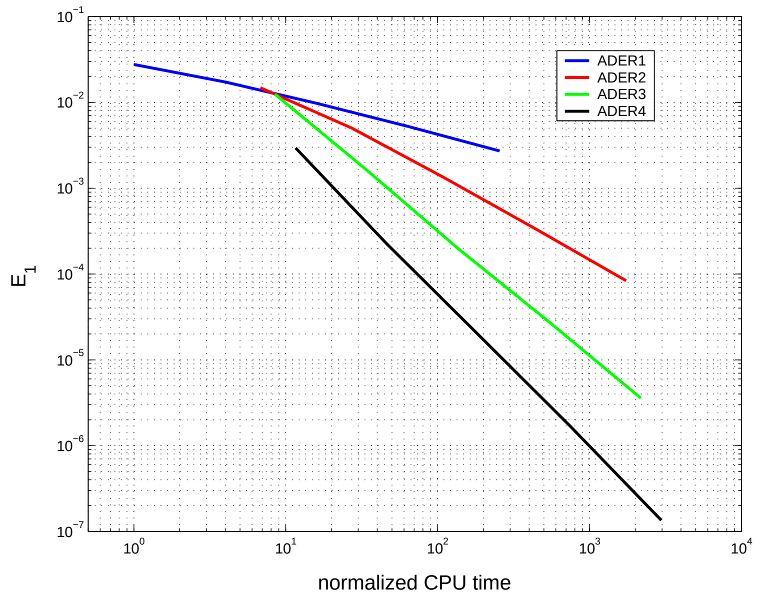


Figure 3.13: The accuracy reached by the different ADER schemes in the corresponding relative CPU time.

However, due to the higher order of accuracy, the higher order ADER schemes can be applied to much coarser meshes, which reduces the computational time as discussed below.

Figure 3.12 displays, how the error $E_1(h)$ obtained by the four different ADER schemes decreases with decreasing mesh width h. The plot also shows, that for ADER schemes of higher order the errors decrease very rapidly, if the mesh is refined. Recall, that the number N of required mesh cells for a two-dimensional problem is $N \sim h^{-2}$. The time to compute new cell averages at the next time step depends linearly on N, i.e. $t_{tot} \sim N$.

To give an example, let us look at an error of $E_1(h) = 10^{-3}$ in Figure 3.12, where the ADER2, ADER3, and ADER4 schemes require $h \approx 0.028$, $h \approx 0.055$ and $h \approx 0.1$ to reach the desired accuracy. Therefore, the ADER2 scheme needs about $(0.1/0.028)^2 \approx 12$ times more cells and the ADER3 requires about $(0.1/0.055)^2 \approx 3$ times more cells as ADER4. Now, as the CPU time for one time step for the ADER4 scheme is only 1.7 times larger than for ADER2 and only 1.4 times larger than for ADER3 (see Table 3.8), the ADER4 turn out to be much more efficient.

In other words, we have to combine the results of Figure 3.12 with those in Table 3.7 to decide, which scheme gives the desired accuracy at the lowest overall computational time.

The corresponding results are shown in Figure 3.13. The CPU time is normalized by the CPU time of the ADER1 scheme on the coarsest mesh $\mathbf{B}_0$. Coming back to the above example of $E_1 = 10^{-3}$ Figure 3.13 indicates, that the ADER2, ADER3, and ADER4 schemes require CPU times of about 150, 50, and 20, respectively. Note, that extrapolating the results the ADER1 scheme would require a CPU time larger than 10^3 to reach the desired error of $E_1 = 10^{-3}$.

We remark, that for results of low accuracy, i.e. of a rather large error $E_1 \approx 10^{-2}$, ADER1, ADER2, and ADER3 require roughly the same CPU time. There, the saving of mesh cells just balances the additional time used per time step.

In general, our results in Figure 3.13 show, that for problems with smooth solutions higher order schemes are more efficient than low order schemes with globally refined meshes, especially when highly accurate results are desired. In order to enhance the accuracy of the proposed ADER schemes even more, especially with respect to solutions with discontinuities, we combine the ADER schemes with the ideas of adaptive mesh refinement in order to reduce numerical smearing. Details of our adaptive mesh strategy are discussed below.

3.8 Adaption Rules

One important feature of our ADER schemes on unstructured triangulations is the time dependent adaptive mesh. Adaptivity requires the modification of the triangulation $\mathcal{T}$ during the simulation in order to be able to balance the two conflicting requirements of good approximation quality and small computational costs. In fact, for the sake of reducing the computational complexity we wish to reduce the number of cells, whereas for the sake of good approximation quality we prefer to use a fine mesh and therefore increase the number of cells.
We have combined the proposed ADER schemes with the ideas of the adaption strategy, that has been discussed in previous work [10, 11, 43] and has proved to be efficient and robust. In analogy to Sections 1.5 and 2.5 we work with an error indicator based on local interpolation with radial basis functions. However, the adaption rules for a triangular mesh are modified accordingly.

3.8.1 Error Indication

As described in [10, 11] we use a customized error indicator in order to adaptively modify the triangulation $\mathcal{T}$. A *significance* value η_ℓ for each cell $T \in \mathcal{T}$ is required to reflect the local approximation quality of the cell average $\bar{u}_\ell$. These significances η_ℓ, $\ell = 1, ..., \#\mathcal{T}$, are used in order to flag single triangles as “to be refined” or “to be coarsened”.

Definition 5 *Let $\eta^* = \max_{1\leq\ell\leq\#\mathcal{T}} \eta_\ell$, and let $\theta_{\text{crs}}, \theta_{\text{ref}}$ be two tolerance values satisfying $0 < \theta_{\text{crs}} < \theta_{\text{ref}} < 1$. We say that a cell $T \in \mathcal{T}$ is* `to be refined`, *iff $\eta_\ell > \theta_{\text{ref}} \cdot \eta^*$, and T is* `to be coarsened`, *iff $\eta_\ell < \theta_{\text{crs}} \cdot \eta^*$.*

To be precise, we let $\theta_{\text{crs}} = 0.01$ and $\theta_{\text{ref}} = 0.05$ in our numerical experiments. Note that a node ξ cannot be refined and be coarsened at the same time; in fact, it may neither be refined nor be coarsened.
In order to define the error indicator η_ℓ we first need to specify a set of neighbouring cells for each triangular cell $T \in \mathcal{T}$.

Definition 6 *Let $\mathcal{T}$ be a conforming triangulation. Then for any triangle $T_\ell \in \mathcal{T}$ the set*

$$K_M(T_\ell) = \{T \in \mathcal{T} \,:\, T \cap T_\ell \ \text{ is edge of } \ T_\ell \ \text{ or node of } \ T_\ell \text{ and } \ T \neq T_\ell\}$$

is called Moore neighbourhood *of T_ℓ and all triangles $T \in \mathcal{K}_M(T_\ell)$ are* Moore neighbours *of T_ℓ.*

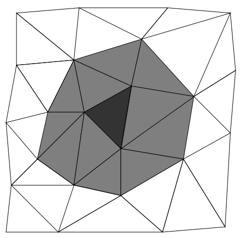


Figure 3.14: A triangle T (dark shaded) with it's Moore neighbours (light shaded).

An example of a Moore neighbourhood is displayed in Figure 3.14.
Following along the lines of [35], and assuming that each cell average value $\bar{u}_\ell$ is assigned to the barycenter ξ_ℓ of cell T_ℓ, i.e. $\bar{u}_\ell \equiv \bar{u}(\xi_\ell)$, the error indicator is then given by

$$\eta_\ell = |\bar{u}(\xi_\ell) - s(\xi_\ell)|, \tag{3.42}$$

where for the Moore neighbourhood $\mathcal{K}_M(T_\ell)$ of T_ℓ the *thin plate spline* interpolant $s \equiv s_{\mathcal{K}_M}$ in (3.42), satisfying the interpolation conditions $s(\xi_\nu) = \bar{u}(\xi_\nu)$ for all $T_\nu \in \mathcal{K}_M(T_\ell)$, is of the form

$$s = \sum_{T_\nu \in \mathcal{K}_M} c_\nu \| \cdot - \xi_\nu \|^2 \log(\| \cdot - \xi_\nu \|) + p\,.$$

Here, p is a linear polynomial in two variables and $\|\cdot\|$ denotes the Euclidean norm. For more details concerning thin plate spline interpolation, due to Duchon [25], and related interpolation methods, the reader is referred to the recent tutorial [42].
Hence, the thin plate spline interpolant s in (3.42) matches current cell average values of $\bar{u}_\nu$ in the Moore neighbourhood of the cell T_ℓ, but not at T_ℓ itself, i.e. we have $\bar{u}(\xi_\ell) \neq s(\xi_\ell)$ in general. Now the error indication η_ℓ for the cell T_ℓ is small whenever the reproduction quality of $\bar{u}_\ell$ by s around cell T_ℓ is good. In contrast, a high value of η_ℓ typically indicates that $\bar{u}_\ell$ is subject to strong variation locally around T_ℓ. Indeed, this observation relies on available local error estimates for thin plate spline interpolation (see the corresponding discussion on this in [10, 11]). We remark that the error indicator allows us to locate discontinuities of the solution u quite effectively. This is supported by the numerical results in the following Section 3.9.

3.8.2 Coarsening and Refinement

We can balance the approximation quality of the solution against the required computational complexity by inserting the barycenter of the triangle T_ℓ as a new node in the triangulation $\mathcal{T}$ where the value of η is high (refinement), whereas we remove nodes from the triangulation $\mathcal{T}$ in regions where the value of η is small (coarsening).

Coarsening. A cell $T_\ell \in \mathcal{T}$ is *coarsened* by the removal of its vertices (nodes) from the current triangulation $\mathcal{T}$. All cells sharing a node in $\mathcal{T}$ have to be flagged as `to be coarsened` to actually remove that particular node from the triangulation $\mathcal{T}$. Afterwards, the triangulation is updated in the sense of a local Delaunay re-triangulation.

Refinement. A cell $T_\ell \in \mathcal{T}$ is *refined* by the insertion of its barycenter ξ_ℓ and a local re-triangulation according to the Delaunay criterion, which means that the number of nodes in the triangulation $\mathcal{T}$ is increased by 1 and the triangulation is updated accordingly.

Recall the results of Section 3.7 and previous work [5, 45], where a dependency of the accuracy of numerical schemes on the degree of the mesh distorting was observed.
We remark, that our refinement strategy may lead to rather long and thin triangles especially in transition zones between regions of very fine and very coarse meshes as shown by our numerical examples in the following Section 3.9. Even though our algorithm is capable of handling such triangular meshes, we suggest to consider a refinement strategy introduced by Hempel in [39] in order to avoid these mesh degeneracies, such as very long and thin triangles. A brief discussion of this isotropic mesh refinement technique together with some preliminary results is given in Appendix B.

3.8.3 Conservativity

The conservative ADER schemes belong to the class of finite volume methods and therefore physical data, e.g. density or saturation, is represented by cell averages in each control volume as given by (3.3).
The adaptive algorithm, where data is transferred from an input triangulation $\mathcal{T}^n$ onto a modified output triangulation $\mathcal{T}^{n+1}$, must respect the conservation property of the underlying scheme (3.16). To this end, we have to redistribute new cell average values to the new cells $T \in \mathcal{T}^{n+1}$. This is accomplished by using the intersection algorithm of O'Rourke [63], which is outlined in Appendix A. Knowing the cell average values on $\mathcal{T}^n$, we get the new cell averages on $\mathcal{T}^{n+1}$ by computing the intersections of triangles similar to the technique discussed in Section 2.4.2.
In order to redistribute new cell average values on $\mathcal{T}^{n+1}$ with m-th order accuracy, when using an ADERm scheme, we need to use an appropriate Gaussian quadrature rule for triangles (see Appendix C), which is exact for polynomials of degree $m-1$.

3.9 Numerical Results

In this section, we apply the proposed adaptive ADER4 schemes to two numerical model problems. The first problem is the rotating slotted cylinder, a linear problem with variable coefficients as suggested by Zalesak in [89] and already discussed in Section 2.6. The second problem is the Burgers equation [17] a popular nonlinear test problem as used in Section 1.6. The results illustrate the performance of the adaptive scheme, in particular the essentially non-oscillatory behaviour of the solution with discontinuities.

3.9.1 Slotted Cylinder

During the last few years, various numerical methods have been developed to treat shocks, that arise in many nonlinear problems, with satisfactory success. This is mainly due to the *self-sharpening* properties of these nonlinear shocks. In our opinion, the accurate representation and sharp resolution of the evolution of discontinuities in linear problems, still seems to be a challenge.

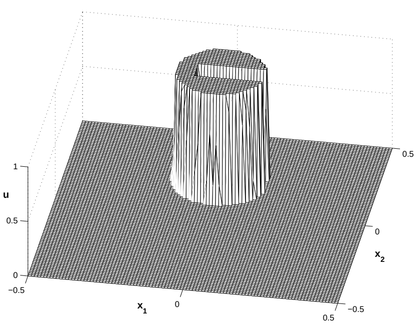
Therefore, we choose the linear problem of the rotating slotted cylinder suggested by Zalesak [89]. Similar to Experiment 2 in Section 2.6.2, we let the computational domain $\Omega = [-0.5, 0.5]^2 \subset \mathbb{R}^2$ and use the initial condition (2.21), where $D \subset \Omega$ is the slotted disc as shown in Figure 2.9**(a)**.

The simulation time is set to $I = [0, 6]$. The rotational flow field of constant angular velocity $\omega = 1$ is given by $a(x) = (x_2, -x_1)$, with $x = (x_1, x_2)$. Therefore, the disc D completes six full rotations around the center $(0, 0)$ of the domain Ω.

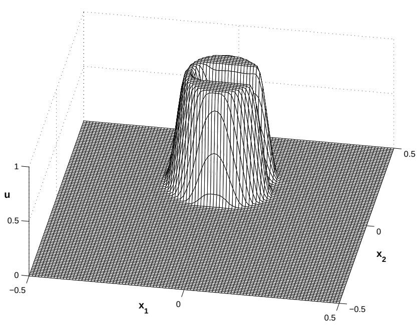
Initially, the computational domain is discretised by a slightly irregular mesh of 200 triangular cells and the initial condition (2.21) is applied. The adaption algorithm then refines the triangles in the vicinity of the discontinuities of the initial function $\bar{u}(0, x)$, such that after only a few iterations, the disc D is resolved sharply as shown in Figure 3.15**(a)**, **(b)**.

Figure 3.16 displays the solution $\bar{u}(t, x)$ at four different times during the first revolution. The corresponding triangular mesh is adaptively modified during the simulation in order to capture the moving discontinuity by locally refined cells as shown in Figure 3.17.

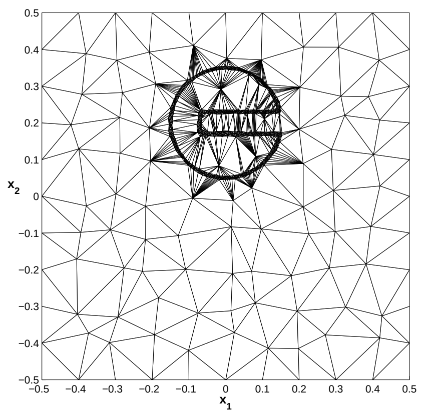
Due to the combination of the adaptive mesh refinement and the highly accurate ADER4 scheme the shape of the slotted cylinder can be preserved even after six full revolutions (see Figure 3.15**(c)**,**(d)**). Comparing our results with those in [73] shows a significant improvement of the method due to the successful reduction of the numerical diffusion by an adaptive ADER4 scheme. However, numerical diffusion still remains a problem for very long simulation times. The solution in Figure 3.15**(c)** and the corresponding mesh in Figure 3.17**(d)** already show the smearing of the discontinuity.



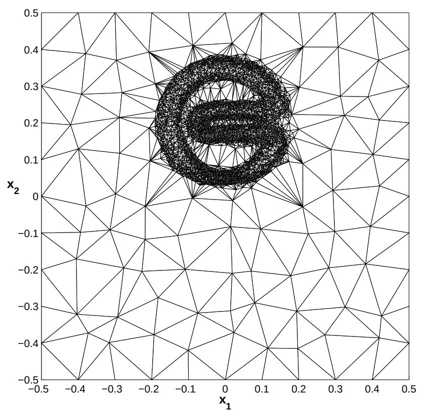
(a)



(c)



(b)



(d)

Figure 3.15: The slotted cylinder. **(a)** 3D view and **(b)** cell distribution, of the initial condition (left column), and after six revolutions (right column), **(c)**,**(d)**.

Instead of increasing the order of the applied ADER scheme or increasing the resolution of the mesh by using even finer locally refined meshes an alternative is to use the method of artificial compression as introduced in [36, 88]. However, artificial compression still is a field of active research and is susceptible to various technical pitfalls as shown in [51], especially for reconstructions of polynomials of degree $d > 1$.
Similar to the numerical Experiment 2 in Section 2.6 we analyse the performance of the adaptive ADER4 scheme for the slotted cylinder problem by recording the number of cells, the variation of the time step size τ, and the ratio of the first mass moment

$$RFM(t) = \sum_{\ell=1}^{\#\mathcal{T}} \bar{u}_\ell(t) \ / \ \sum_{\ell=1}^{\#\mathcal{T}} \bar{u}_\ell(0) \tag{3.43}$$

with cell averages $\bar{u}_\ell$ of the triangular cells $T_\ell \in \mathcal{T}$, during the simulation.

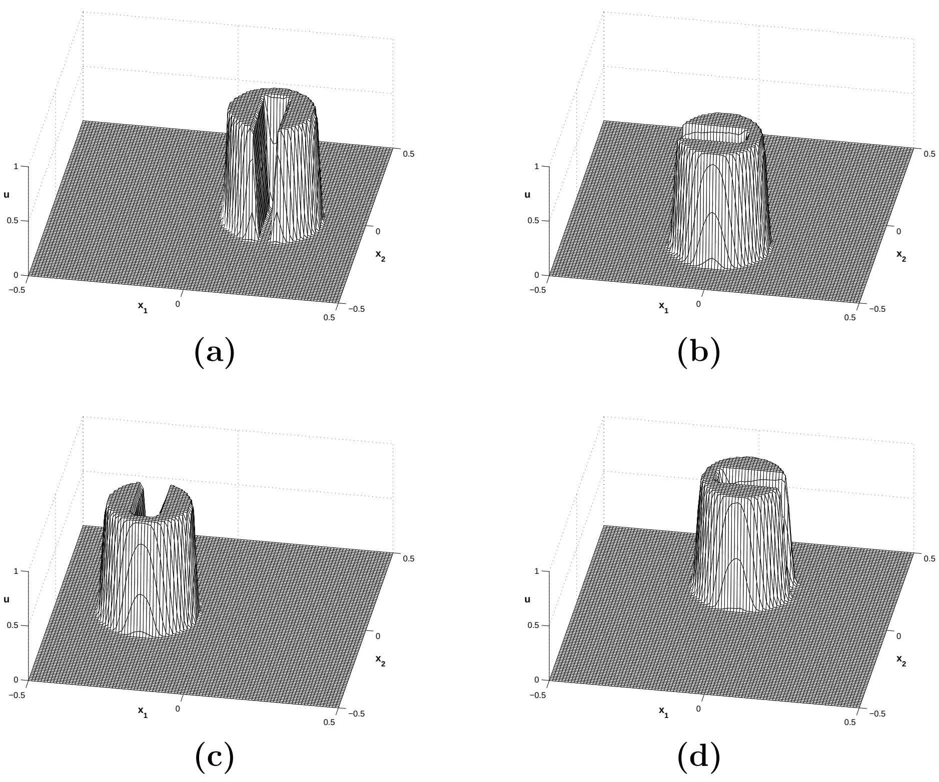


Figure 3.16: The slotted cylinder. 3D view on the evolution of $\bar{u}(t)$ at four different times, **(a)** $t = t_{460}$; **(b)** $t = t_{920}$; **(c)** $t = t_{1380}$; **(d)** $t = t_{1840}$, during the first revolution.

The results are reflected by Figure 3.18. In the first graph the number of cells is displayed for the six revolutions. The increasing number of cells at the beginning of the simulation is due to numerical diffusion, but levels off at about 8000 cells. The very slight decrease in cells towards the end of the simulation is also an effect caused by numerical diffusion, that gradually smears the initial discontinuity. In fact, for later times the discontinuity is represented by a very step gradient of the solution.
Recall that our *a posteriori* error estimator is based on the local interpolation using Thin Plate Splines as radial basis functions (see Sections 1.4 and 1.5). Therefore, linear functions can be approximated exactly leading to very small errors, i.e. very small significance values.
Now, the steep gradients representing the smeared discontinuities especially towards the end of the simulation, can be considered as linear functions and therefore result in rather small significance values. This in turn lead to the coarsening and slight reduction of cells, where the gradients are close to linear functions.

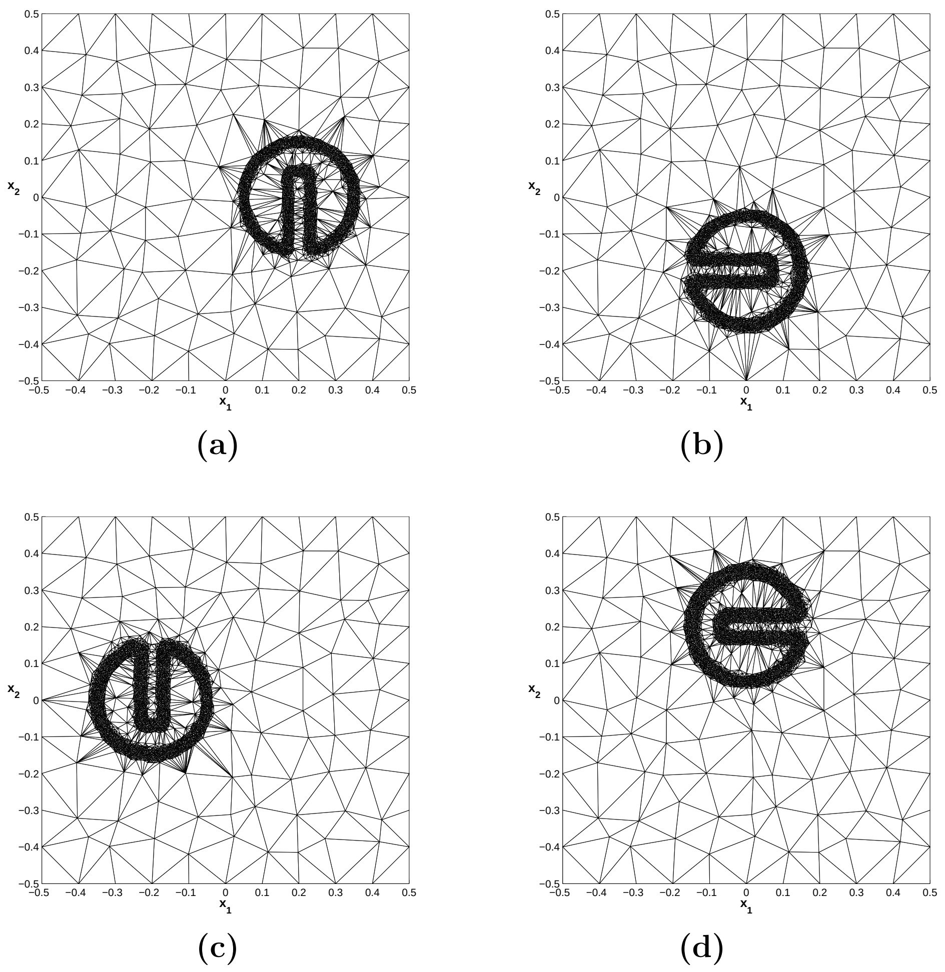


Figure 3.17: The slotted cylinder. Adaptive triangulation during the simulation at four different times, **(a)** $t = t_{460}$; **(b)** $t = t_{920}$; **(c)** $t = t_{1380}$; **(d)** $t = t_{1840}$, during the first revolution.

This effect can also be seen in Figure 3.15**(c)** and **(d)**, where the coarsening of the locally refined mesh becomes obvious in areas of steep gradients.
The second graph in Figure 3.18 shows the variation of the time step τ. As discussed in Section 3.5 the proposed ADER schemes are explicit and, for stability reasons, the time step τ is restricted by a CFL condition given in (3.35). Here, τ depends on the radius ρ_ℓ of the inscribed circle of the cells T_ℓ, $\ell = 1, ..., \#\mathcal{T}$, and therefore is chosen adaptively in each time step. The graph indicates, that during the simulation smallest accepted cells can occur, that reduce the time step size τ accordingly. Choosing a global τ small enough to satisfy (3.35) for the entire simulation would necessitate a dramatically increased number of time steps to compute.
The third graph in Figure 3.18 displays the *RFM* in (3.43). As we avoid

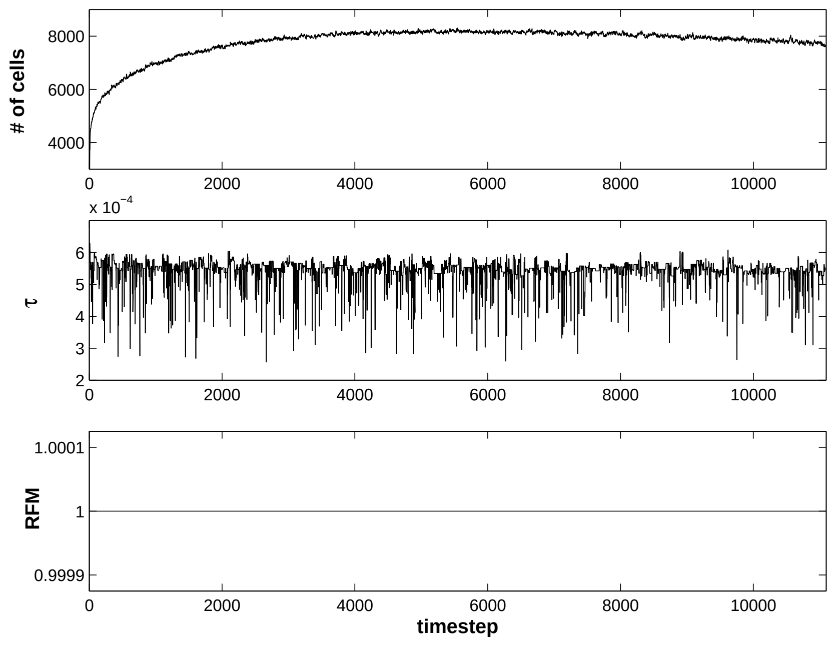


Figure 3.18: The slotted cylinder. Number of nodes, time step size τ, and ratio of the first mass moment (RFM).

fluxes across the boundaries of Ω for the slotted cylinder example and use the conservative formulation (3.16) for the proposed ADER schemes, it is no surprise, that mass conservation is satisfied exactly.

3.9.2 Burgers equation

As discussed in Section 1.6 Burgers equation (3.40) represents a standard test case of a nonlinear conservation law because of it shock wave behaviour. Even for smooth initial data the solutions typically develop shock fronts. Similar to the example in Section 1.6, we solve (3.40) with the initial condition

$$u_0(x) = \begin{cases} \exp\left(\frac{\|x-c\|^2}{\|x-c\|^2-R^2}\right) & \text{for } \|x-c\| < R \\ 0 & \text{otherwise} \end{cases} \tag{3.44}$$

with $R = 0.15$, $c = (-0.2, -0.2)^T$ on the two-dimensional computational domain $\Omega = [-0.5, 0.5]^2 \subset \mathbb{R}^2$ (cf. [30]).
Initially, Ω is discretised by a triangular mesh of 200 cells, which are successively adapted to the initial condition within a few iterations using our customized adaption rules of Section 3.8.
Plots of the solution u at four different time steps $t = t_0$, $t = t_{100}$, $t = t_{300}$, and $t = t_{700}$ are shown in Figure 3.19. Note, that the error indicator localizes the support of the initial function u_0 effectively and locally refines the

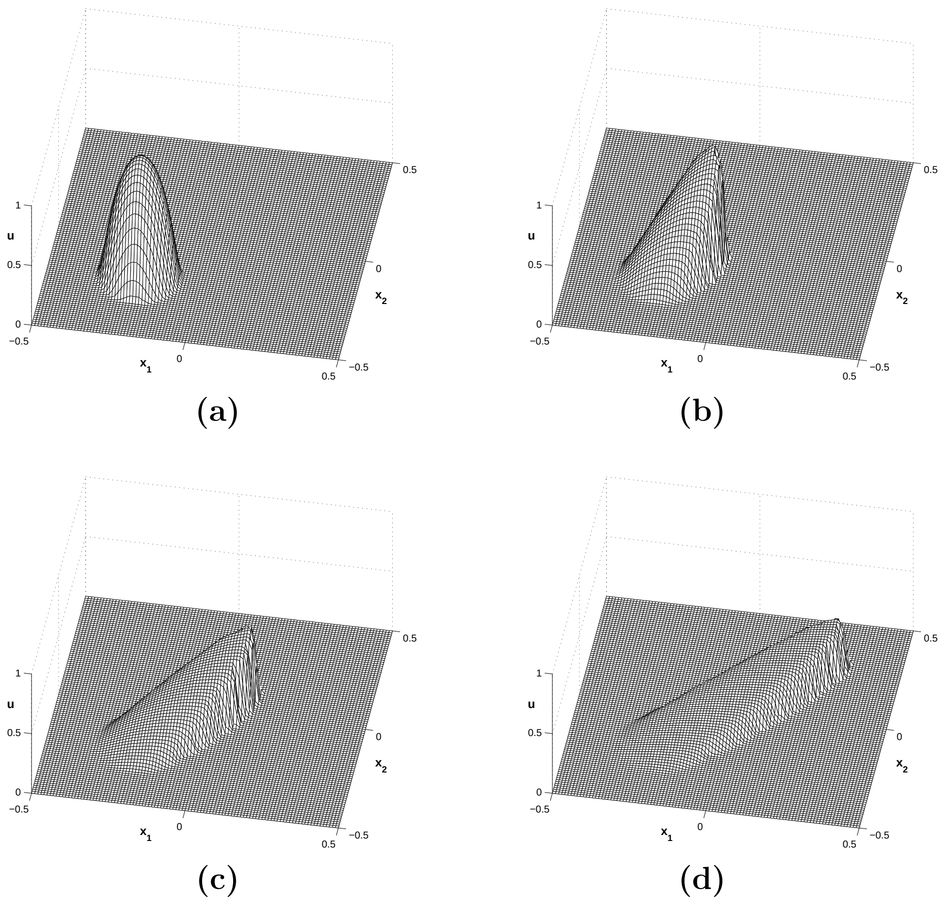
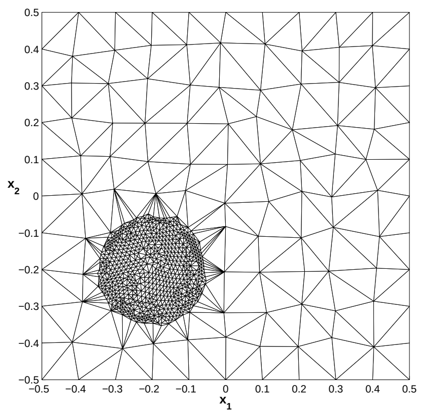
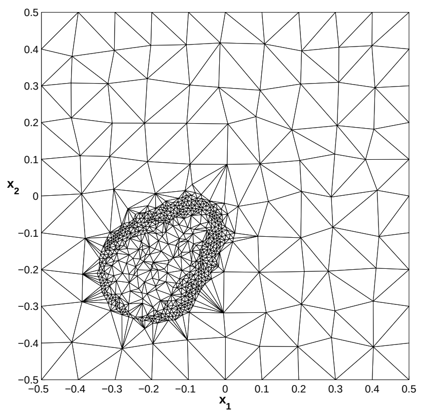


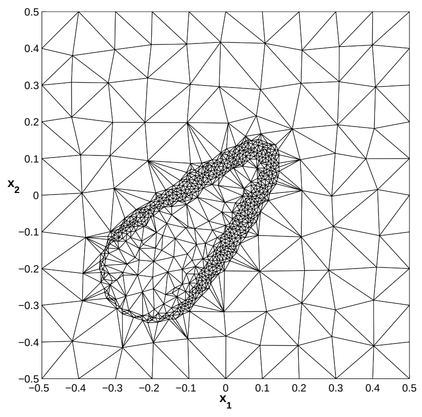
Figure 3.19: Burgers equation. 3D view on the evolution of $\bar{u}(t)$ at four different times, **(a)** $t = t_0$; **(b)** $t = t_{100}$; **(c)** $t = t_{300}$; **(d)** $t = t_{700}$.



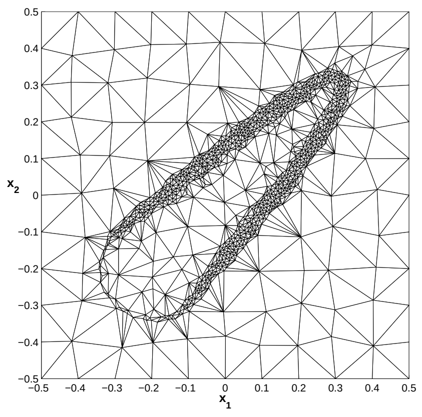
(a)



(b)



(c)



(d)

Figure 3.20: Burgers equation. Adaptive triangulation during the simulation at four different times, **(a)** $t = t_0$; **(b)** $t = t_{100}$; **(c)** $t = t_{300}$; **(d)** $t = t_{700}$.

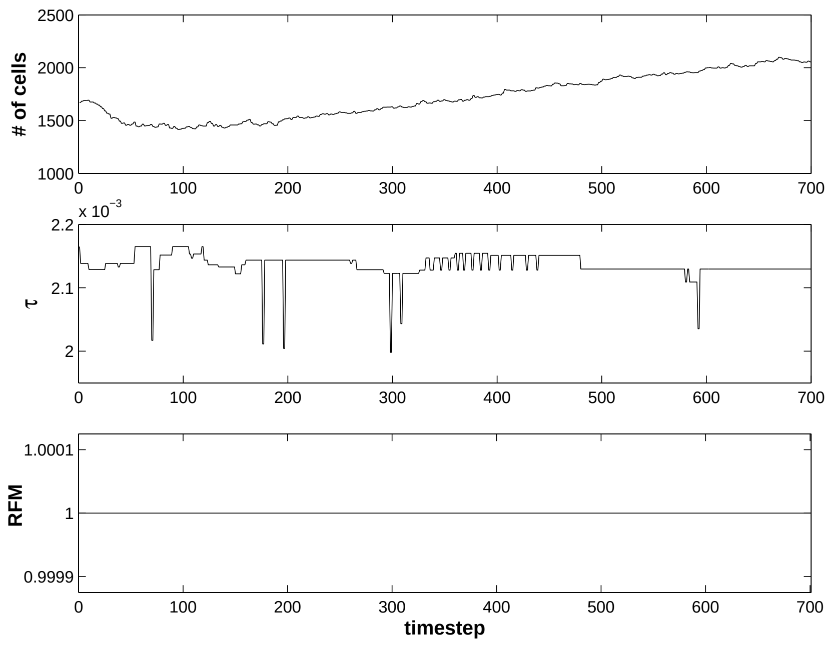


Figure 3.21: Burgers equation. Number of nodes, time step size τ, and ratio of the first mass moment (RFM).

mesh. Furthermore, after the shock formation its propagation through the computational domain Ω is well resolved by the adaptively modified mesh during the simulation. We also remark, that in regions, where the solution is rather smooth, the mesh is coarsened to keep the computational cost low (see Figure 3.20). Therefore, the utility of the customized adaption rules is confirmed once more.

We remark, that we solve the same model problem of Burgers equation with an alternative, isotropic mesh adaption strategy addressed in Section 3.8.2. See Appendix B for further details.

The graphs of Figure 3.21 display the results recorded during the simulation. The first plot shows the evolution of the number of cells. At the beginning, we see a slight decrease of cells. The initial function u_0 is *tilting* in the direction of the velocity field $a(x) = (1, 1)$ leading to a shock formation in the front followed by a rarefaction. As the solution becomes rather smooth in the region of the rarefaction, the mesh is gradually coarsened due to small significance values of the error estimator. As this happens before the shock is formed and to propagates along the diagonal of Ω, the number of cells is initially reduced.

The effects of coarsening the mesh in the region of the rarefaction and refining the mesh at the propagating shock balance each other until time step 200. Afterwards, the refinement of the mesh around the propagating shock domi-

nates and the number of mesh cells increases. Due to the growing support of the solution u and the corresponding growth of the length of the shock front causes a moderate but steady increase of the number of cells.
The second graph of Figure 3.21 displays the variation of the time step τ during the computations. Again through the CFL condition (3.35) the time step τ depends on the smallest triangular cell occurring in the current mesh. However, in contrast to the previous example of the slotted cylinder, periods of constant time step sizes can be observed. This indicates that during these periods the smallest triangle remains in the adaptive mesh and determines the time step τ.
In the third graph of Figure 3.21 we again plot the RFM of (3.43) to check the conservation property of our scheme. This time, periodic boundary conditions are applied and we observe the expected exact mass conservation of the proposed adaptive ADER scheme.

3.10 Conclusion

We presented an extension of the new ADER schemes on adaptive, unstructured triangulations in order to solve linear and nonlinear scalar conservation laws. Originally, the ADER approach based on Arbitrary high order DERivatives was introduced by Toro, Millington and Nejad in [84] for linear problems on Cartesian meshes and further expanded by Toro and Titarev to nonlinear problems in [81, 83]. As ADER schemes belong to the class of finite volume methods, we discuss in detail the reconstruction of high order polynomials from cell average values on unstructured triangular meshes. We apply the WENO reconstruction technique in order to achieve a high order approximation quality while avoiding spurious oscillations of the solution. To this end an elaborate stencil selection algorithm is introduced that relies on the idea of a sectoral search. The resulting piecewise polynomial approximation of the solution provides generalized Riemann problems at the cells interfaces, which can be solved by reducing them to a series of conventional Riemann problems. Furthermore, the Lax-Wendroff procedure uses the solutions of these Riemann problems in order to determine high order fluxes across the cell interfaces and update the cell average values for each time step.
The performance of the proposed ADER schemes is evaluated with respect to experimental orders of convergence and their computational efficiency. The results show, that using higher order ADER schemes, although increasing the computational costs per time step, finally pay off due to the tremendous reduction in mesh cells required to reach a desired accuracy. We then optimize the computational efficiency by combining the high order ADER approach with the strategy of adaptive mesh refinement. Therefore, the according error estimator and the customized adaption rules for refining and coarsening

the triangular mesh are explained in detail. Finally, we demonstrate the good performance of the proposed adaptive ADER schemes in two numerical experiments by solving linear and nonlinear conservation laws.
Our main interest, however, is the solution of the Buckley-Leverett equation, which describes the flow of two fluids through a porous medium. In particular, such problems are relevant for the oil industry. Therefore, we apply the introduced adaptive ADER schemes on a well-established model problem from reservoir simulation in the following Chapter 4.

Chapter 4

Two-phase Flow in Porous Media

4.1 Introduction

The exploration and production of hydrocarbon reservoirs is still the most important technology to deploy natural energy sources. Thereby, fluid flow simulators play a key role in order to help oil companies to make effective use of expensive data collected through field measurements, data processing and interpretation. In fact, simulation is one of the few tools available for modelling changes in a reservoir over time. Combined with other measurements it improves the degree of confidence in the understanding of reservoirs and heavily influences reservoir management decisions.
A central problem in petroleum reservoir simulation is to model the displacement of one fluid by another within a porous medium. A typical problem is characterized by the injection of a wetting fluid (e.g. water) into the reservoir at a particular location displacing the non-wetting fluid (e.g. oil), which is extracted or *produced* at another location. The nature of the front between the water and the oil is of primary importance and the goal is to withdraw as much oil as possible before water reaches the production location.
The physical phenomena that govern these enhanced oil recovery processes typically have important local properties. Thus, numerical schemes used to simulate these effects must be able to resolve such critical local features with high accuracy. In addition, to be useful for large scale simulations, the schemes should be efficient and therefore adaptive. The interaction between the two fluids water and oil typically results in a moving shock front, whose shape and movement is required to be predicted numerically. In our opinion, the use of adaptive mesh refinement in combination with ADER schemes on arbitrary triangulations seems to be a promising approach to accurately capture such shock fronts.

4.1.1 Reservoir Flow Formulation

Some general aspects of the reservoir flow problem and the corresponding fundamental equations of reservoir flow are reviewed briefly. A detailed discussion of the governing equations and their derivation from physical constraints is given in the renowned text book of Aziz and Settari [3]. Petroleum reservoirs consist of hydrocarbons and other chemicals trapped in the pores of a rock. If the rock permits and if the fluid is sufficiently forced, the fluid can flow from one location to another inside a reservoir. By the injection of additional fluids and the release of pressure during the production phase the flow rates and even the mixture of chemicals can be modified by petroleum engineers.
A simplified, but realistic model problem is given by the two-phase Buckley-Leverett model [16]. This model considers reservoirs containing some mixture of water and oil, which both are incompressible. Diffusive effects, such as capillary pressure or the physical mixing of fluids (as a result of flow through a large number of randomly connected rock pores) are ignored. Furthermore, gravitational forces are neglected.
As each fluid phase is conserved, their behaviour can be modelled be the following equations:
Mass conservation of water:

$$\phi(x)\frac{\partial}{\partial t}u_w(t,x) + \nabla a_w(t,x) = 0\,. \tag{4.1}$$

Mass conservation of oil:

$$\phi(x)\frac{\partial}{\partial t}u_o(t,x) + \nabla a_o(t,x) = 0\,. \tag{4.2}$$

Here, the scalar field $\phi(x)$ describes the porosity of the rock, the vector fields $a_w(t,x)$ and $a_o(t,x)$ are the phase velocities, and $u_w(t,x)$ and $u_o(t,x)$ are the saturations of water and oil, respectively. Note, that u_w and u_o are the fractions of the pore space, that are filled with water or oil, i.e. $0 \le u_{w,o} \le 1$. Equations (4.1) and (4.2) indicate, that a change of mass for each phase in a given region of a reservoir is equal to the net flux of the phase across the boundary of that region. Therefore, the class of finite volume schemes, such as the proposed ADER schemes, obviously are a natural choice of available numerical methods to solve such problems.
By definition, the saturations u_w and u_o must fulfill the condition

$$u_w(t,x) + u_o(t,x) = 1\,, \tag{4.3}$$

as the pore space is assumed to be entirely filled with a mixture of water and oil.

The phase velocities are determined by Darcy's law

$$\begin{aligned} a_w(t,x) &= -\mathbf{K}(\mathbf{x})\frac{kr_w(u_w)}{\mu_w}\nabla p(t,x)\,, \\ a_o(t,x) &= -\mathbf{K}(\mathbf{x})\frac{kr_o(u_o)}{\mu_o}\nabla p(t,x)\,, \end{aligned}$$

where $\mathbf{K}(x)$ is the permeability tensor of the porous rock and $kr_w(u_w)$ and $kr_o(u_o)$ are the relative permeabilities of the water and oil phase. The permeability tensor $\mathbf{K}(x)$ of a rock describes its ability to transmit fluids, whereas the relative permeabilities depend on the actual saturation of the rock of the according phase. The variables μ_w and μ_o denote the viscosities of the two fluids and $p(t,x)$ is the reservoir pressure. The required values for permeabilities and viscosities are typically determined by laboratory measurements of core samples, whereas the reservoir pressure can be estimated from in situ down-hole measurements in the field. Here, the ratios

$$\frac{kr_w(u_w)}{\mu_w} = M_w(u_w) \quad \text{and} \quad \frac{kr_o(u_o)}{\mu_o} = M_o(u_o)$$

usually are termed the phase mobilities providing the total mobility $M = M_w + M_o$. Adding equations (4.1) and (4.2) and use the relation (4.3), leads to

$$\nabla\cdot\Big(a_w(t,x) + a_o(t,x)\Big) = \nabla\cdot a(t,x) = 0\,,$$

stating that the total fluid velocity $a(t,x)$ is divergence free. Now, the phase velocity, e.g. of water, can be expressed by

$$a_w(t,x) = a(t,x)\cdot f_w(u_w)\,,$$

where $f_w(u_w)$ is the flux tensor and is defined by

$$f_w(u_w) = \frac{M_w(u_w)}{M(u_w)}\,, \tag{4.4}$$

the ratio of the phase mobility $M_w(u_w)$ and the total mobility M. We remark, that in the field of reservoir simulation and engineering, the flux tensor $f_w(u_w)$ usually is called the *fractional flow* of the water phase. Therefore, we let $u = u_w$ and $f(u) = f_w(u_w)$ in order to simplify the notation.

Then the equations describing two-phase flow in a reservoir are

$$\begin{aligned} a(t,x) &= -\mathbf{K}(x)M(u)\nabla p(t,x)\,, && (4.5)\\ \nabla\cdot a(t,x) &= 0\,, && (4.6)\\ \frac{\partial}{\partial t}u + a\cdot\nabla f(u) &= 0\,, && (4.7)\end{aligned}$$

where $0 \leq u \leq 1$ has to be satisfied. The above equations are known as Darcy's law (4.5), the incompressibility relation (4.6), and the Buckley-Leverett equation (4.7).
In reservoir modelling the function $f:\ u \rightarrow f(u)$ is monotonic increasing and satisfies $0 \leq f(u) \leq 1$ for all $u \in [0,1]$. In the following applications, all computations are based on the Corey model (cf. [3]) with quadratic relative permeabilities of the form

$$kr_w(u) = u^2\,, \qquad kr_o(u_o) = (1-u)^2\,,$$

which yield the total mobility

$$M(u) = \frac{u^2}{\mu_w} + \frac{(1-u)^2}{\mu_o}\,,$$

and a fractional flow in (4.4) of the form

$$f(u) = \frac{u^2}{u^2 + \frac{\mu_w}{\mu_o}(1-u)^2}\,. \qquad (4.8)$$

Note, that f in (4.8) is also called the Buckley-Leverett flux already introduced by (1.7) in Chapter 1 and is used to model the displacement of oil by water.

4.1.2 The Five-Spot-Problem

In the following, the computational domain $\Omega = [-0.5, 0.5] \times [-0.5, 0.5]$ represents a homogeneous medium with $\phi(x) = \phi = 1$ and $\mathbf{K}(x) = 1$ for $x \in \Omega$. Furthermore, we assume $M(u) = 1$ in (4.5) and keep the reservoir pressure constant in time, i.e. $p(t,x) = p(x)$. Therefore, the total velocity field $a(t,x)$ in (4.5) has to be computed only once at the beginning of a simulation and is then independent of time, i.e. $a(t,x) = a(x)$.
In general, the total velocity field will change during the simulation as the mobility $M(u)$ depends on the changes in the saturation u. Substituting equation (4.5) in (4.6) yields an elliptic equations, that would have to be solved for the pressure, which in turn provides an updated total velocity field $a(t,x)$ through (4.5).

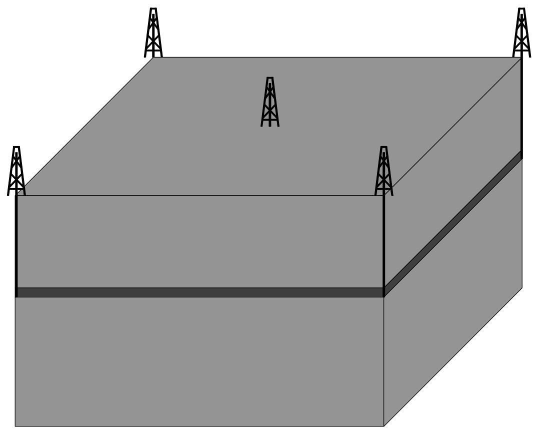


Figure 4.1: A model of the five-spot-problem with one production well in the center and four injection wells at the corners.

However, as we are focussing on solving the nonlinear conservation law (4.7) we separate the coupled differential equations for saturation and pressure and neglect the pressure equation. This can be justified by the fact, that for the chosen five-spot-problem the pressure changes have rather small effects on the solution [1]. The main idea of using this approximation is to test the performance of the proposed ADER schemes of Chapter 3 as adaptive unstructured saturation solvers. Our aim is to obtain qualitatively reasonable results to investigate how the adaptive ADER schemes cope with velocity fields that typically arise in reservoir simulations.

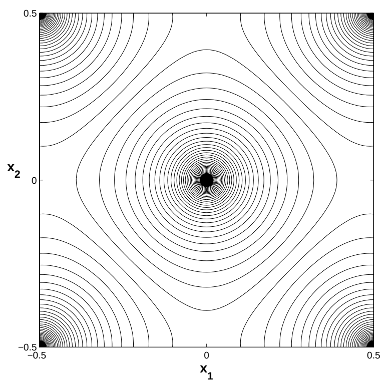
Now, let us consider the standard test case of the five-spot-problem. Here, one production well P in the center of the computational domain Ω is surrounded by four injection wells I_j, $j = 1, ..., 4$, located at the corners of the model as displayed in Fig. 4.1. Assume a thin oil bearing layer trapped between two sealing layers as shown in Fig. 4.1, which allows us to reduce the problem to two-dimensions by taking a (x_1-x_2)-slice (see Figure 4.2) of the model at the depth of the oil bearing layer.
The distances between a particular location $x = (x_1, x_2)$ in the computational domain Ω and the wells are given by

$$r_{I_j}(x) = \sqrt{(x_1 - x_{1,I_j})^2 + (x_2 - x_{2,I_j})^2}\,, \quad j = 1, ..., 4 \tag{4.9}$$

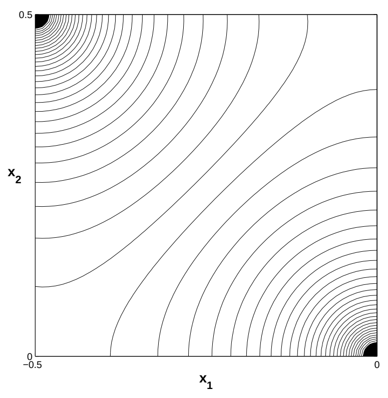
$$r_P(x) = \sqrt{(x_1 - x_{1,P})^2 + (x_2 - x_{2,P})^2}\,, \tag{4.10}$$

for the four injection wells I_j and the production well P.

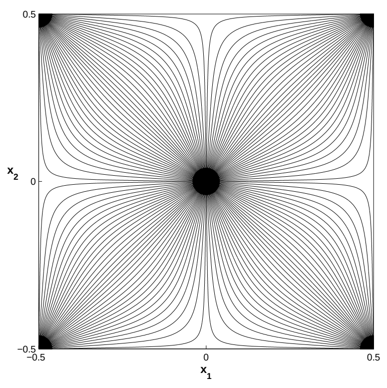
[1]Even in sophisticated, full reservoir simulators the pressure field and therefore the velocity field are recomputed infrequently compared to the saturation.



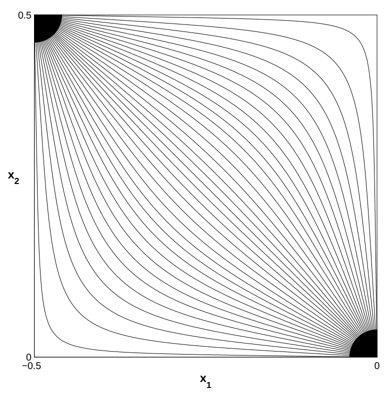
(a)



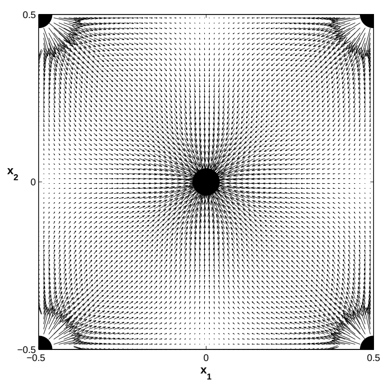
(b)



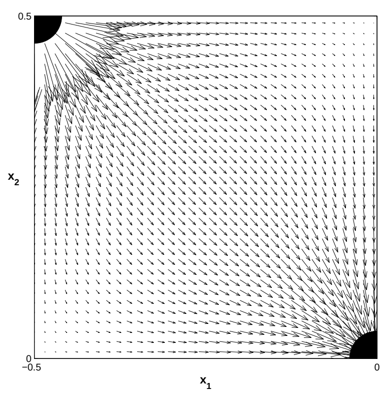
(c)



(d)



(e)



(f)

Figure 4.2: **(a)** Contour plot of the pressure field, **(c)** the total velocity field represented by streamlines, and as **(e)** velocity vectors. The plots in **(b)**,**(d)**,**(f)** show the zoomed section of the top left quarter.

With the distances (4.9), (4.10) we define a scalar reservoir pressure field through

$$p(x) = \log\Big(r_P(x)\Big) - \sum_{j=1}^{4} \log\Big(r_{I_j}(x)\Big)\,, \tag{4.11}$$

as displayed in Figures 4.2**(a)**, **(b)**. We then use Darcy's law (4.5) and the assumptions discussed above in order to derive the total velocity field from the pressure (4.11). Figures 4.2 represents the velocity field via streamlines in **(c)**, **(d)** and via velocity vectors in **(e)**, **(f)**. Note, that the direction of flow is always along the pressure gradient and therefore perpendicular to the pressure contours. To be precise, the components of the velocity field $a(x) = a(x_1, x_2) = (a_1(x_1, x_2), a_2(x_1, x_2))$ are given by

$$\begin{aligned} a_1(x_1,x_2) &= -\tfrac{x_1 - x_{1,P}}{(x_1-x_{1,P})^2+(x_2-x_{2,P})^2} + \textstyle\sum_{j=1}^{4} \tfrac{x_1 - x_{1,I_j}}{(x_1-x_{1,I_j})^2+(x_2-x_{2,I_j})^2} \\ a_2(x_1,x_2) &= -\tfrac{x_2 - x_{2,P}}{(x_1-x_{1,P})^2+(x_2-x_{2,P})^2} + \textstyle\sum_{j=1}^{4} \tfrac{x_2 - x_{2,I_j}}{(x_1-x_{1,I_j})^2+(x_2-x_{2,I_j})^2}\,. \end{aligned} \tag{4.12}$$

It can be shown by differentiation, that this velocity field $a(x)$ in (4.12) is divergence free as required by equation (4.6). Note, that it is not uncommon to have orders of magnitudes difference in the absolute values of the total velocity a, with high velocities near the wells and lower velocities in places between the wells.

4.2 Numerical Results

In the following, we solve the Buckley-Leverett equation (4.7) with the total velocity field a given in (4.12) and the fractional flow specified in (4.8) by using the proposed ADERm schemes, $m = 1, ..., 4$, of Chapter 3. Then we solve the same model problem with two standard reservoir simulator well-established in the oil industry.

4.2.1 Adaptive ADER Schemes

Using an arbitrary triangulation, we can easily adapt the triangular cells to the injection wells. We set the radius of the injection wells to $R = 0.04$ and use the initial condition

$$u_0(x) = \begin{cases} 1 & \text{for } \|x - c_j\| \le R, \quad j = 1, ..., 4 \\ 0 & \text{otherwise,} \end{cases} \tag{4.13}$$

where the c_j are the centers of the four injection wells. This way, the initial condition (4.13) mimics the situation of pure water injection into an initially 100% oil saturated reservoir.

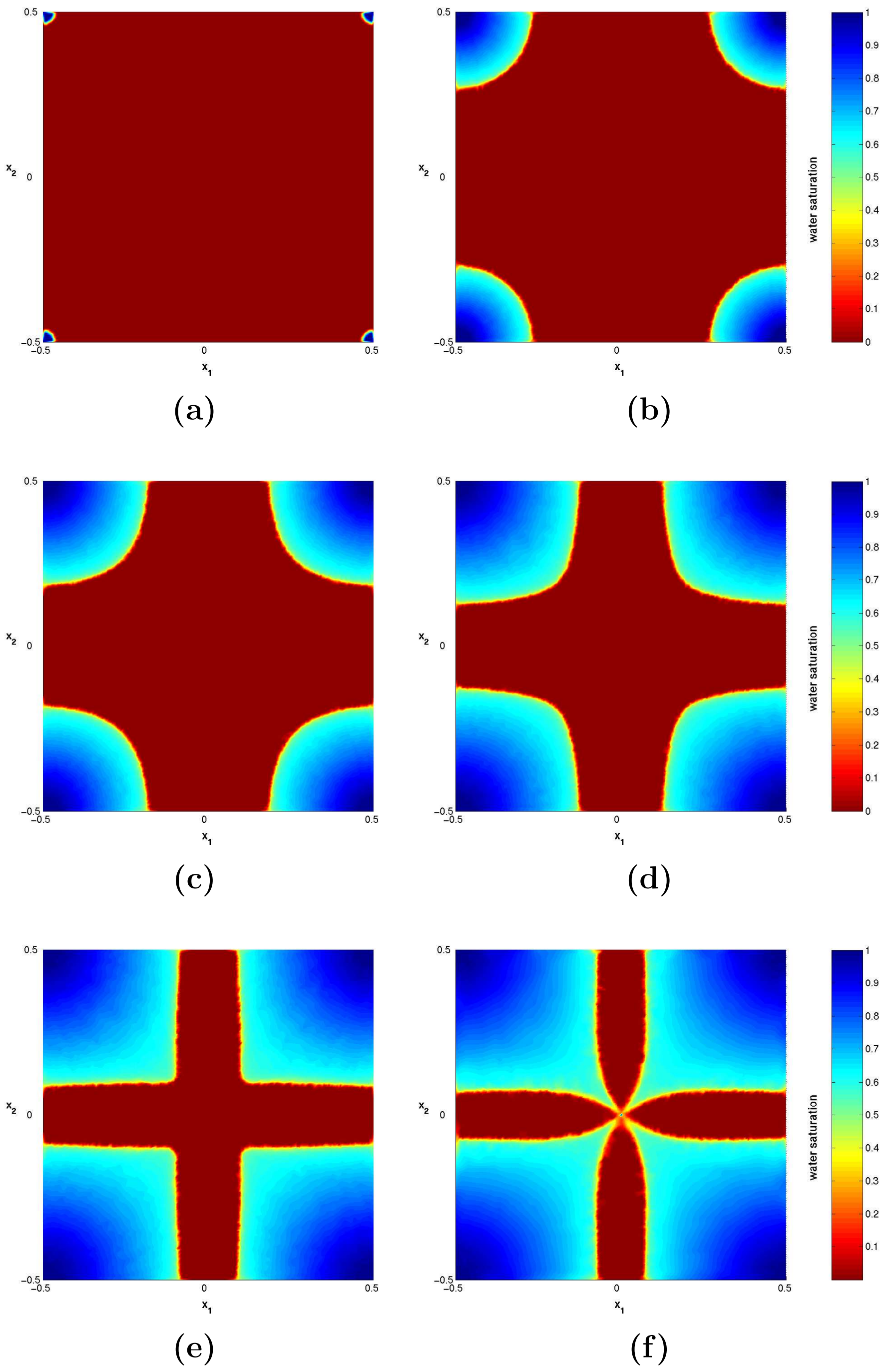


Figure 4.3: Five spot problem computed with ADER1. Color plots indicating the injection of water during the simulation at six different times, **(a)** $t = t_0$; **(b)** $t = t_{120}$; **(c)** $t = t_{240}$; **(d)** $t = t_{360}$; **(e)** $t = t_{480}$; and **(f)** $t = t_{600}$; .

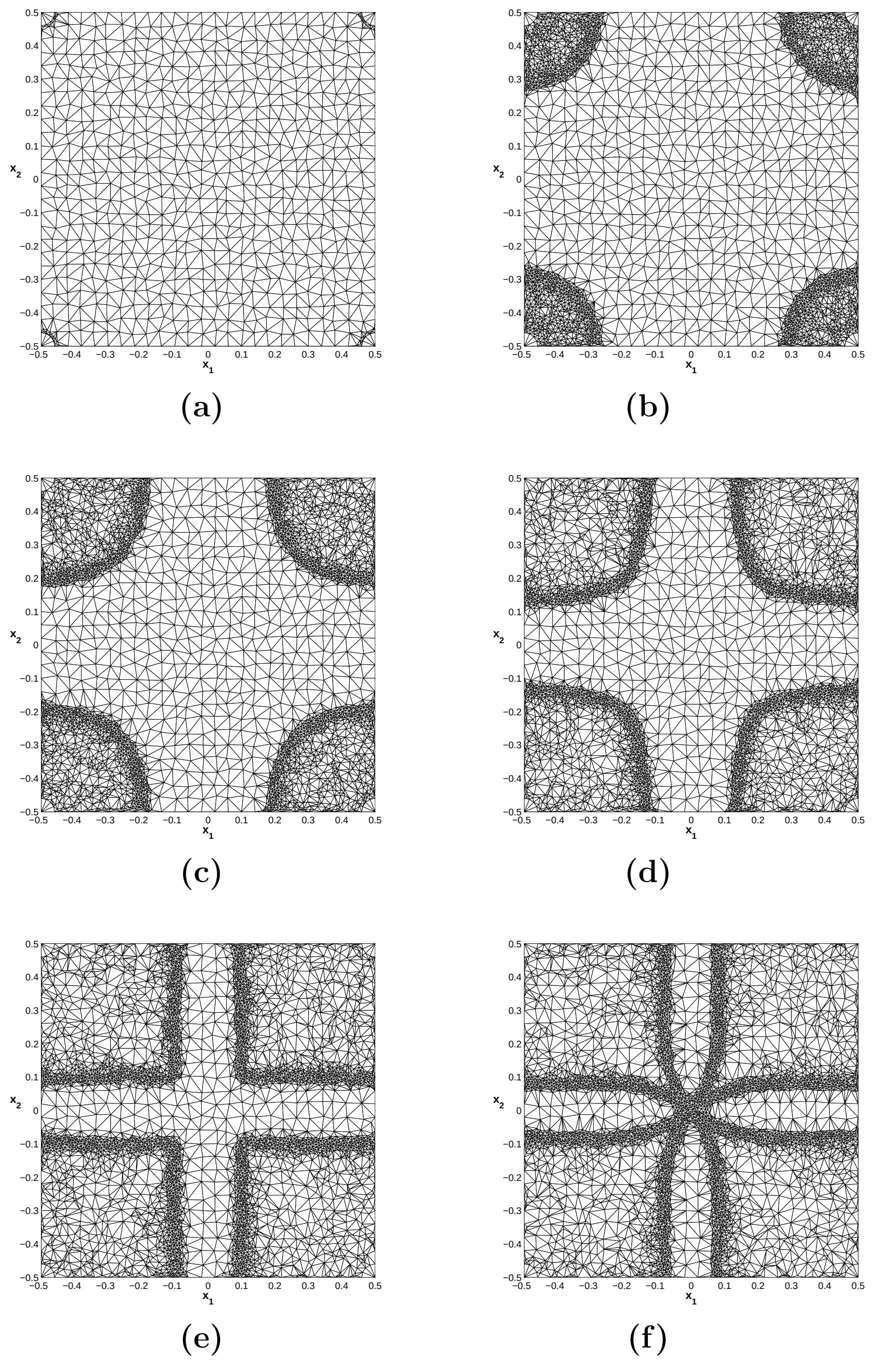


Figure 4.4: Five spot problem computed with ADER1. Adaptive triangulation during the simulation at six different times, **(a)** $t = t_0$; **(b)** $t = t_{120}$; **(c)** $t = t_{240}$; **(d)** $t = t_{360}$; **(e)** $t = t_{480}$; and **(f)** $t = t_{600}$; .

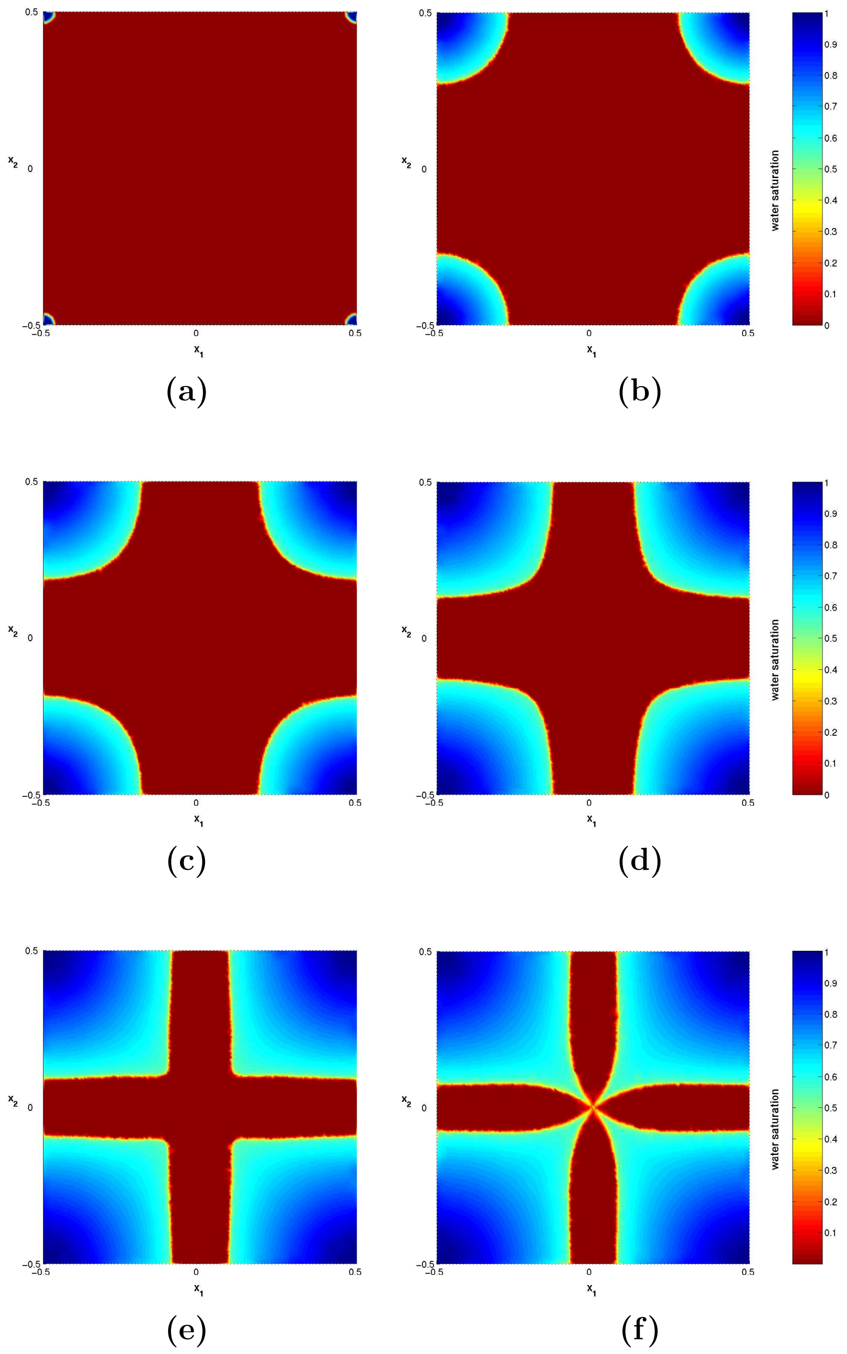
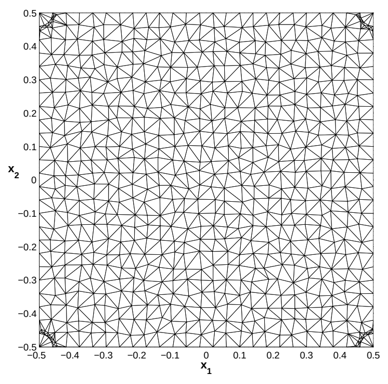
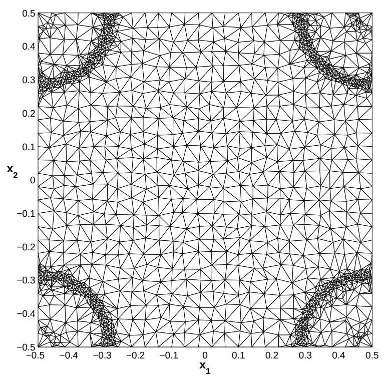


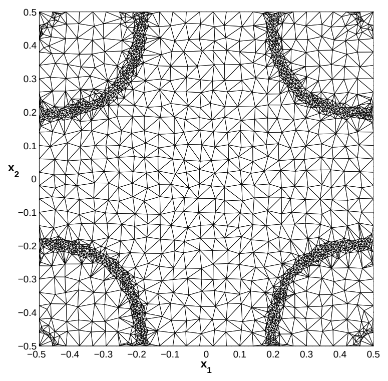
Figure 4.5: Five spot problem computed with ADER4. Color plots indicating the injection of water during the simulation at six different times, **(a)** $t = t_0$; **(b)** $t = t_{120}$; **(c)** $t = t_{240}$; **(d)** $t = t_{360}$; **(e)** $t = t_{480}$; and **(f)** $t = t_{600}$; .



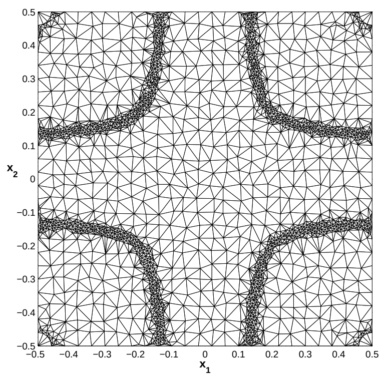
(a)



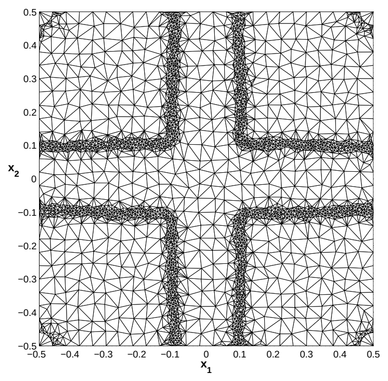
(b)



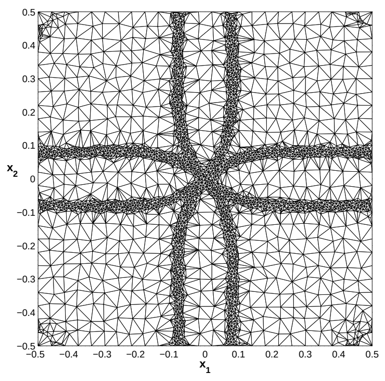
(c)



(d)



(e)



(f)

Figure 4.6: Five spot problem computed with ADER4. Adaptive triangulation during the simulation at six different times, **(a)** $t = t_0$; **(b)** $t = t_{120}$; **(c)** $t = t_{240}$; **(d)** $t = t_{360}$;**(e)** $t = t_{480}$; and **(f)** $t = t_{600}$; .

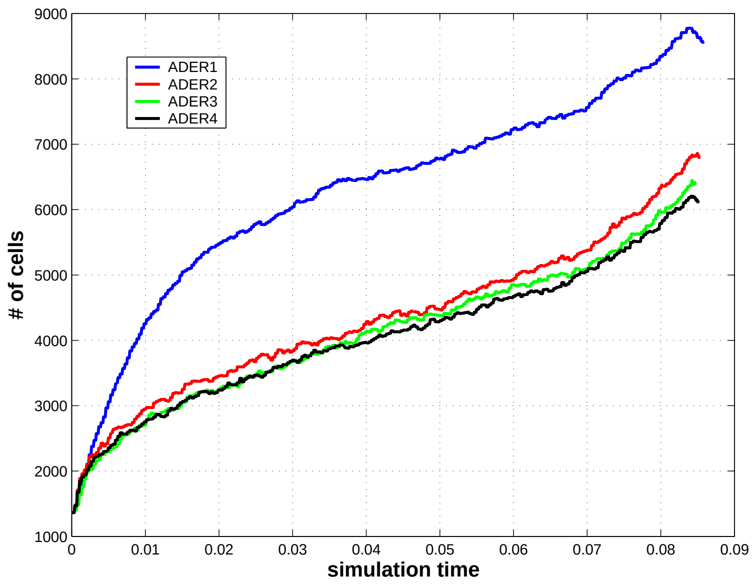


Figure 4.7: The number of cells during the simulation for four different ADER schemes.

In order to see the differences between low and high order ADER schemes, we first show the color-coded water saturation obtained with an adaptive ADER1 scheme in Fig. 4.3. The shocks representing the interface between pure oil and a mixture of oil and water are moving from the corners of the model reservoir towards its center. This way, oil in the porous medium is displaced by water, i.e. it is effectively pushed towards the production well. Before the shocks actually reach the production well at the center, the sucking effect of the production well becomes obvious, which is due to the increasing total velocity resulting from the pressure drop at the production well. We remark, that the event, when the shock front arrives at the production well, is called the *breakthrough*.

The underlying adaptive mesh is displayed in Fig. 4.4 and clearly shows, how the locally refined mesh adaptively captures the shocks. After a shock has passed a particular locations, the mesh is coarsened, if the error indicator allows, in order to reduce the computational costs. However, the mesh behind the shocks remains finer than it was originally, which is due to the rarefaction following the moving shock, where the saturation of water slowly increases. In fact, in the zones of the rarefactions in Fig. 4.3 one can recognise the shapes of some triangular cells emerged from the recoarsening.

In Fig. 4.5 we show the results of the same model problem obtained by an adaptive ADER4 scheme. The shocks appear sharper in this case compared to the ADER1 results of Fig. 4.3. Also the rarefaction is much smoother, as the structure of an underlying triangular mesh disappears.

However, if we look at the corresponding adaptive mesh in Fig. 4.6, we see that there is even a coarser mesh in the areas of the rarefactions. In fact, behind the shocks the error indicator allows to recoarsen the mesh to is original coarsest level. This is due to the increased approximation quality of the higher order ADER4 scheme, which uses piecewise cubic polynomials instead of piecewise constant functions to reconstruct the water saturation function u. Therefore, coarser meshes can be used without losing accuracy.

In Fig. 4.7 the number of cells of the adaptive mesh is plotted versus time for the four different ADERm schemes, $m = 1, ..., 4$. Clearly, if the order of the ADERm scheme is increased, the number of required cells is reduced. Note, that the reduction in mesh cells going from an ADER1 to an ADER2 scheme is significant, whereas by going to higher order ADER schemes this reduction is not very remarkable. This is due to the fact, that the shock is always resolved with a very fine mesh, whereas in the the regions of the rarefaction wave the error estimator already allows an ADER2 scheme to recoarsen the mesh almost to its initial state.

4.2.2 Comparison with Reference Solutions

In order to confirm the performance of the proposed ADER schemes and the obtained results, we compute the solution of the discussed five-spot-problem with two different reservoir simulators and use these results as reference solutions. The chosen simulators ECLIPSE and FRONTSIM are two commercial software packages used by the majority of reservoir simulations groups in the oil industry. We remark, that these simulators solve the coupled system of the pressure and saturation equations (4.5)-(4.7) and therefore consider the effect of pressure changes due to changes in saturation.
This in turn typically causes changes in the total velocity field, but as confirmed by the following results, these effects can be neglected for the homogeneous five-spot-problem. The reference solutions are both computed on a two-dimensional Cartesian mesh consisting of 100×100 rectangular cells, which are fixed throughout the simulation.
Figure 4.8 shows the water saturation as obtained with the simulator ECLIPSE. It is obvious, that the moving shocks are much more smeared than in the case of the ADER schemes. On the other hand, the general behaviour of the solutions is very similar as well as the size of the discontinuity.
The results obtained by the simulator FRONTSIM are displayed in Fig. 4.9. Here, one can see, that the numerical diffusion is extremely small and the interface at the oil-water contact is resolved very sharply.
In order to get a better comparison of results of the different ADER schemes and the reference solutions of ECLIPSE and FRONTSIM, we look at a cross section of the water saturation u.

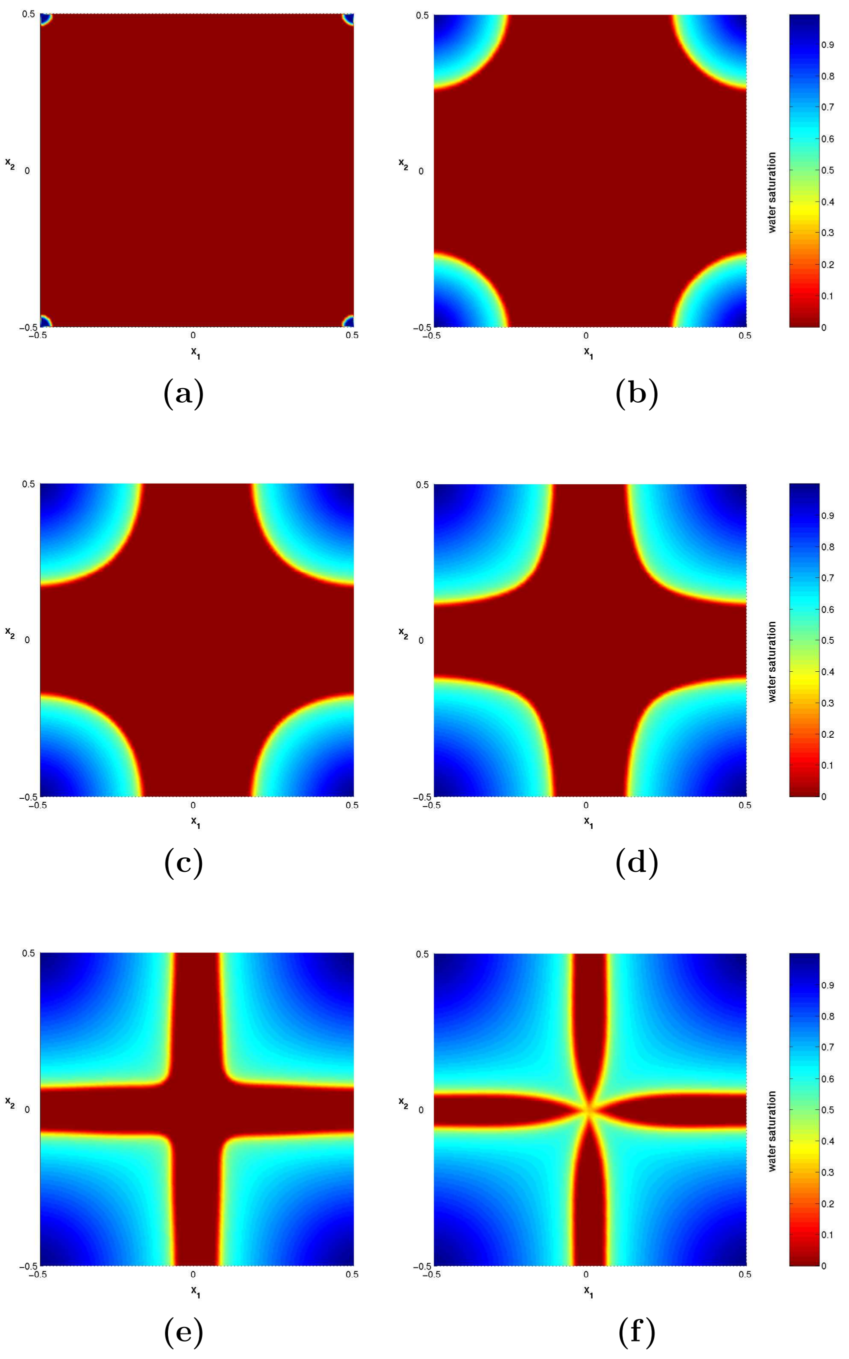


Figure 4.8: Five spot problem computed with ECLIPSE. Color plots indicating the injection of water during the simulation at six different times, **(a)** $t = t_0$; **(b)** $t = t_{120}$; **(c)** $t = t_{240}$; **(d)** $t = t_{360}$; **(e)** $t = t_{480}$; and **(f)** $t = t_{600}$; .

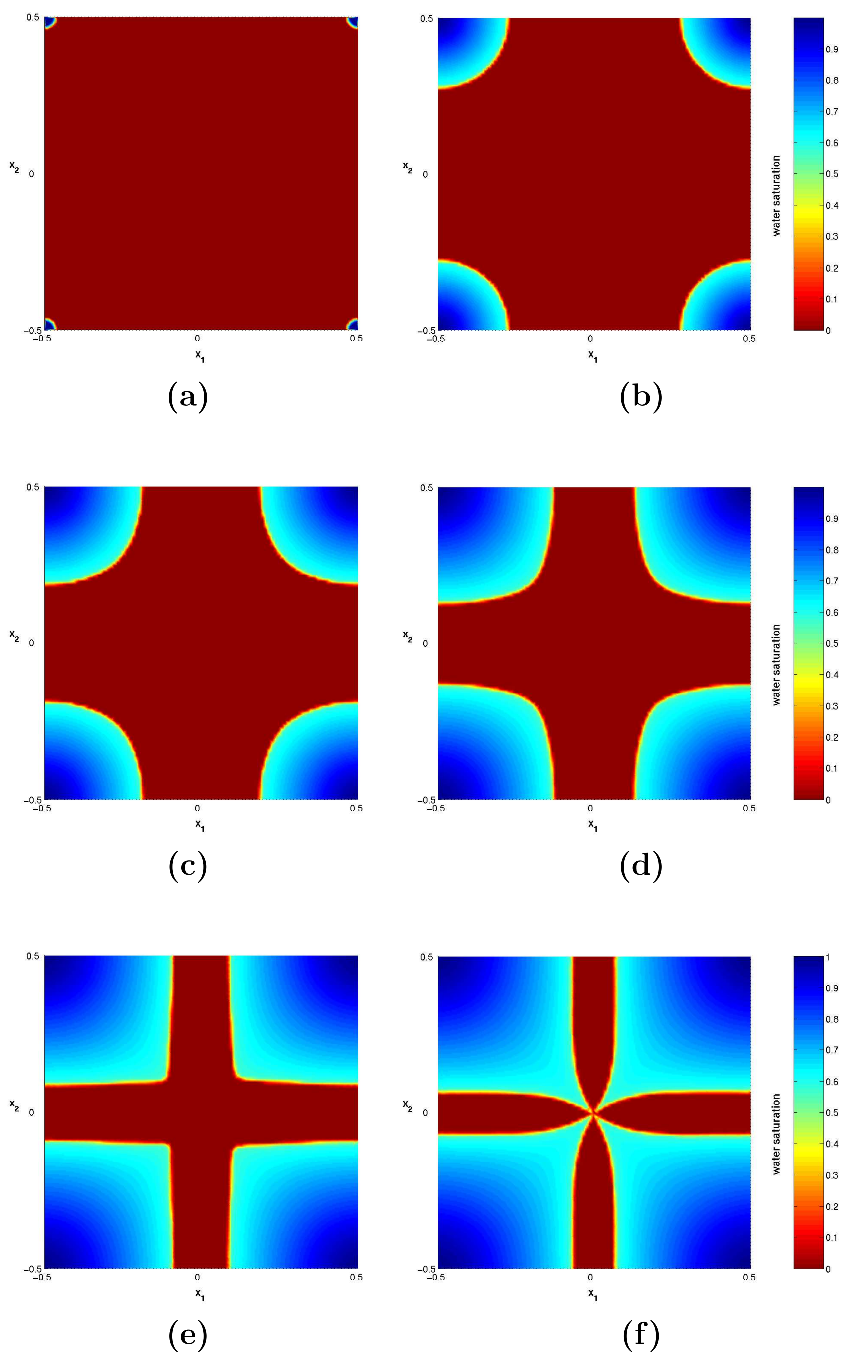
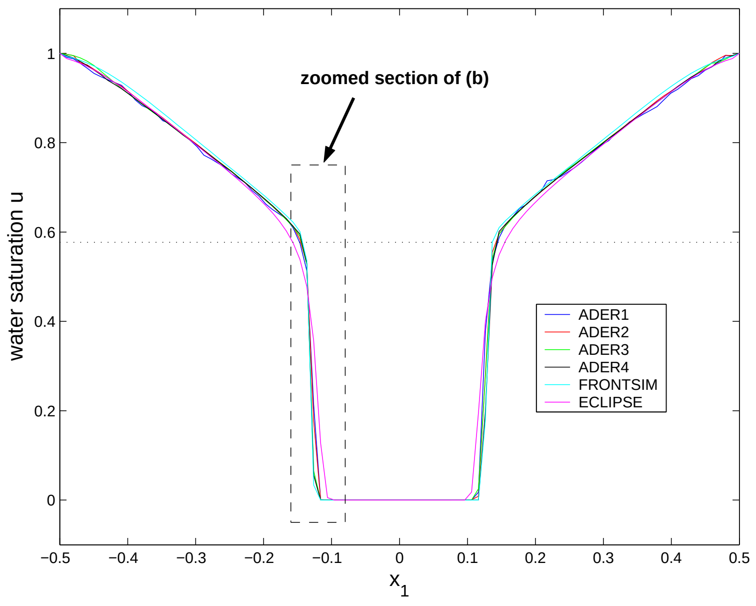
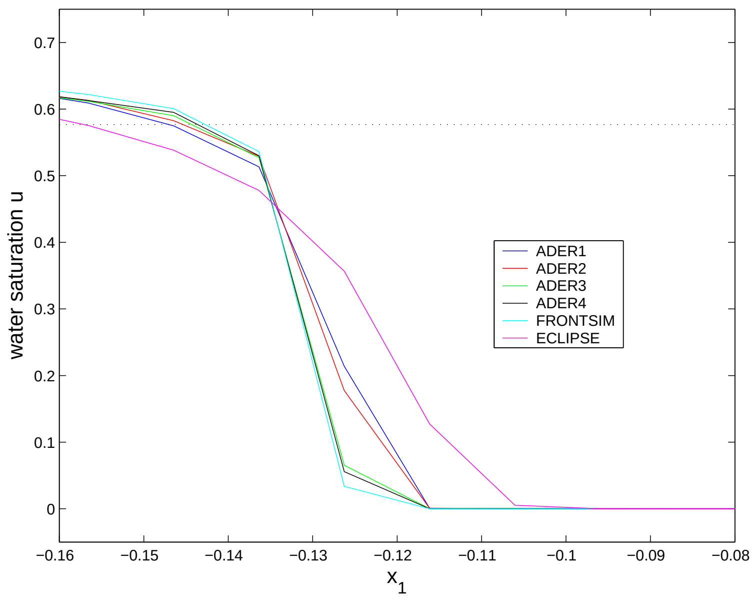


Figure 4.9: Five spot problem computed with FRONTSIM. Color plots indicating the injection of water during the simulation at six different times, **(a)** $t = t_0$; **(b)** $t = t_{120}$; **(c)** $t = t_{240}$; **(d)** $t = t_{360}$; **(e)** $t = t_{480}$; and **(f)** $t = t_{600}$; .



(a)



(b)

Figure 4.10: A comparison of saturation profiles obtained by different ADER schemes together with the reference solutions computed with ECLIPSE and FRONTSIM.

Figure 4.10 shows the water saturation profile taken at the time step $t = t_{360}$ along the line $x_1 \in [-0.5, 0.5]$ cutting through the computational domain Ω at $x_2 = 0.45$. The plot in Fig. 4.10**(a)** displays an overlay of all saturation profiles obtained by the schemes for ADERm schemes, $m = 1, ..., 4$, together with the two reference solutions. In addition, the theoretical expected size of the shock is drawn as dotted line.
We observe, that all methods lead to very similar results and the expected jump at the oil-water contact is more or less reproduced by all schemes. To see the differences between the chosen methods in more detail, we zoom in at an area around the left shock as indicated in Fig. 4.10**(a)** and show the magnified plot in Fig. 4.10**(b)**. Here, it becomes obvious, that the reference solution computed by ECLIPSE heavily smeared by numerical diffusion. The reason is, that ECLIPSE is based on a finite volume scheme of order 1.
A clear improvement is achieved by using the adaptive ADER1 scheme, which is also only first order, but resolves the shock much better due to the local mesh refinement in the vicinity of the shock. Furthermore, the number of mesh cells used by the ADER schemes (see Figure 4.7) can dramatically be reduced compared to the ECLIPSE and FRONTSIM, which both work with 100×100 Cartesian mesh. Increasing the order of the ADER schemes, the shock becomes sharper (see ADER2, ADER3 and ADER4 in Fig. 4.10**(b)**), whereas the number of required cells is even lower (see Figure 4.7).
However, the highest resolution of the shock is obtained by FRONTSIM, which is based on a *front tracking* scheme (see [41]) that are well known for their small numerical diffusion. Nevertheless, the saturation profile obtained by ADER4, which belongs to the class of *front capturing* schemes, is already very close to the results of FRONTSIM.
A fair comparison with respect to CPU times between our ADER schemes and the ECLIPSE or FRONTSIM software is difficult, as our simulations are carried out with MATLAB 6 Release 13 on a PC (model: IBM 236623G) with processor type Intel Pentium(R) 4 1600MHz, whereas ECLIPSE and FRONTSIM run as an optimized FORTRAN code on a UNIX system.

4.3 Conclusion

We applied the new ADER schemes on adaptive, unstructured triangulations on the nonlinear Buckley-Leverett equation. To this end, we choose the five-spot problem, that serves as a standard model problem for reservoir simulators in the oil industry. Simplifying assumptions justify the neglect of pressure changes during the simulation and therefore do not require a pressure solver for adaptive triangulations. Comparisons with reference solutions obtained by two commercial reservoir simulators also justify our simplifications.

The results demonstrate, that the adaptive ADER schemes capture the moving discontinuities with high resolution and produce sharp shock fronts due to the local mesh refinement. The robustness of the proposed high order ADER schemes in combination with WENO reconstruction techniques is confirmed through the fact, that no spurious oscillations are observed. Furthermore, the adaptive ADER schemes allow us to reduce the number of required cells dramatically compared to the commercial software based on fixed Cartesian meshes.
Further research and development is, of course, necessary in order to design a new adaptive reservoir simulator, which is able to handle real-world reservoirs. However, our encouraging results may serve as a trend-setting approach in order to construct such adaptive and therefore highly accurate and conservative simulators, that are able to handle complicated geometries easily.

Outlook

This work has shown, that adaptivity can constitute a crucial property of a numerical scheme. Especially for hyperbolic conservation laws, whose solutions may develop discontinuities, adaptive methods can effectively capture these critical features with high resolution and accuracy. We have also shown, that the number of mesh cells is hereby kept to a minimum, as regions with smooth parts of the solution are discretised with a rather coarse mesh.
However, each of our proposed methods have their particular weaknesses independent of their adaptivity.
The meshless approach of Chapter 1 turns out to be problematic, as it is not conservative. In fact, it is difficult to measure the mass represented by one particle as the particles carry concentration or density values. Therefore, the size of the region, that is represented by this particle plays an important role in the evaluation of the corresponding mass. We remark, that this problem is subject of current research.
Furthermore, the use of the artificial viscosity adds diffusion, which avoids the development of *real* shocks. Instead, we obtain steep gradients and the amount of added diffusion seems to be problem-dependent. We have shown, that it is possible to choose a viscosity term, that produces very promising results. However, the use of artificial viscosity also influences the stability of the numerical scheme. Therefore, the advantage of the semi-Lagrangian approach, namely the use of large time steps, is partly lost. A detailed investigation of how to determine the sufficient and necessary amount of artificial viscosity for various problems and defining the according time step criterion for arbitrary point clouds is still an open problem.
Our conservative semi-Lagrangian approach of Chapter 2 has shown, that for linear problems with variable coefficients, the proposed adaptive advection scheme is a very powerful tool. Due to its adaptivity and the capability of using rather large time steps, the method is very attractive for problems that require long simulation times. The numerical diffusion of the scheme is very small, such that initial discontinuities in the solution remain sharp even after long simulation times.
We remark, that there is a potential of increasing the order of the proposed second order scheme by replacing the linear reconstruction polynomials on the Voronoi cells by polynomials of higher order. Additionally, the compu-

tation of backward trajectories for the construction of upstream cells can be replaced by a more accurate ODE solver. However, either approach to enhance the accuracy of the proposed scheme will consequently increase the computational costs.
A further extension of the proposed advection scheme can be achieved by allowing for non-convex upstream cells. This way, ever larger time steps could be used. However, a more costly algorithm for the computation of the intersection of non-convex polygons is then necessary.
The major problem, in our opinion, is the extension of the proposed conservative semi-Lagrangian scheme from linear to nonlinear problems.
The ADER schemes of Chapter 3 turned out to be more suitable for the treatment of nonlinear problems. As ADER schemes belong to the class of explicit finite volume methods, the time step is restricted by a CFL criterion. Due to the local mesh refinement, the allowed time step may be quite small according to the smallest mesh cell. The large number of time steps and therefore updates of cell average values usually causes numerical diffusion, which reduces the high resolution properties of the adaptive high order ADER scheme. Therefore, artificial compression techniques [51, 88] could be incorporated in order to reduce numerical smearing to a minimum.
A significant improvement in accuracy and resolution may also be achieved by using elaborate mesh adaption methods, that work with isotropic refinement strategies as demonstrated in the preliminary results of Appendix B. Furthermore, one could think of mesh alignment methods in combination with mesh adaption, where edges of particular cells are aligned parallel to the orientation of a shock front. Encouraging results of such mesh alignment approaches are shown in [46].
Coupling the mesh adaptivity with time adaptivity may be a further issue to use small time steps only for small cells and allow for larger time steps in larger cells in order to reduce numerical diffusion.
Especially with respect to multi-phase flow in porous media and petroleum reservoir simulation, the modern adaptive ADER schemes have to be coupled with a pressure solver for adaptive unstructured meshes in order to handle more sophisticated real-world reservoirs. To this end, gravitational and capillary effects should also be considered.

Finally, we conclude that since the first successful attempts in computational fluid dynamics in 1950 a wide variety of numerical schemes has been developed. However, the past few years have shown, that the interest in efficient, robust and high order accurate numerical schemes has grown in many application fields. In our opinion, combining high order methods with customized adaption strategies definitely is a trend-setting approach for future research.

Appendix A

Intersection Algorithm

A.1 Intersection of Convex Polygons

It is know from [76], that the intersection of convex polygons with n and m vertices has linear complexity $\mathcal{O}(n+m)$. Computing the intersection of convex polygons is a key component in the numerical schemes developed in this work. In the following, the linear algorithm presented in [63] is outlined briefly.
Assume the boundaries of two convex polygons P and Q. The algorithm in [63] advances the edges of P and Q around the polygons such that they *chase* one another searching for an intersection point. In [63] it is proved, that all such intersection points can be found within two cycles around the polygons and therefore the algorithm achieves linear complexity.

Notation and Definitions. Let P and Q be two convex polygons with vertices p and q oriented counterclockwise. The next and previous vertices with respect to p are denoted by p_+ and p_-, respectively. Furthermore, let $\vec{p}$ and $\vec{q}$ be the directed edges (vectors) on each polygon, such that the inside of the polygon is always on the left. Vertex p is called *head* of $\vec{p}$ and similarly q is called *head* of $\vec{q}$.
The half-plane, including the half-plane edge, determined by the edge vector p is defined as

$$H(\vec{p}) = \{x : \vec{p} \times (x - p_-) \geq 0\}$$

where x is an arbitrary points in the plane.

Advance Rules. The algorithm is mainly based on a set of advance rules, discussed below. If $\vec{q}$ points towards the line containing $\vec{p}$ but does not cross it, then advance $\vec{q}$ to approach a possible intersection with $\vec{p}$. If both vectors point towards each other, either one may be advanced. If neither $\vec{p}$ nor $\vec{q}$ point towards the other, advance whichever is outside the half-plane of the other, or advance either of them, if they are both outside. This, in fact, is

$\vec{p} \times \vec{q}$	half-plane condition	advance
> 0	$q \in H(\vec{p})$	$\vec{p}$
> 0	$q \notin H(\vec{p})$	$\vec{q}$
< 0	$p \in H(\vec{q})$	$\vec{q}$
< 0	$p \notin H(\vec{q})$	$\vec{p}$

Table A.1: Advance rules.

the essence of the advance rules. In the following, $\vec{p} \times \vec{q} > 0$ means, that the z-coordinate of the cross product of $\vec{p}$ and $\vec{q}$ is > 0. All possible cases can be reduced to the four cases as displayed in Table A.1.

A.2 The Algorithm

These advance rules are embedded in a loop, as indicated in Algorithm 3. Here the flag *inside* is used to denote of the head of a particular edge vector lies inside the other polygon. In each iteration, two edges (of P and Q respectively) are checked for intersection and a vertex is output. After the loop has finished, all the vertices of the polygon $P \cap Q$ have been output in counterclockwise order. Of course, if P and Q do not intersect no vertex is delivered and the special case has to be treated separately.
Also note, that Algorithm 3 uses a subroutine, called *advance* of the form:

```
advance $p$:
    output $p$
    IF $inside$ ==“P”
        $p \leftarrow p_+$

advance $q$:
    output $q$
    IF $inside$ ==“Q”
        $q \leftarrow q_+$
```

This subroutine outputs a specified vertex of the corresponding polygon. Additionally, the current vertex is advanced along the edge of the polygon, if the current vertex is flagged as an *inside* vertex.
Note, that there are three types of *degenerate* intersections, that may occur: (a) a vertex of P may lie on an edge of Q, (b) a vertex of P may coincide with a vertex of Q, and (c) an edge of P may be parallel to and overlap an edge of Q. For details on these cases the reader is referred to [63], where a more comprehensive investigation of the algorithm is provided.

Algorithm 3 (Convex Polygon Intersection).

INPUT: *n vertices p and m vertices q of convex polygons P and Q.*

Choose p and q arbitrarily.

```
DO
  IF $\vec{p}$ and $\vec{q}$ intersect
    IF this intersection is the same as the first one
      BREAK
    ELSE
      output point of intersection
      IF $p \in H(\vec{q})$
        inside ← "P"
      ELSE
        inside ← "Q"
  IF $\vec{p} \times \vec{q} < 0$
    IF $p \in H(\vec{q})$
      advance $q$
    ELSE
      advance $p$
  ELSE
    IF $q \in H(\vec{p})$
      advance $p$
    ELSE
      advance $q$
WHILE loop has executed $\leq 2(n+m)$ times
```

Treat special cases, i.e. if $P \cap Q = \emptyset$, or $P \subseteq Q$, or $Q \subseteq P$.

Choose p and q arbitrarily.

```
IF $p \in Q$
  output vertices of $P$
ELSE IF $q \in P$
  output vertices of $Q$
ELSE
  output $\emptyset$
```

OUTPUT: *Vertices of $P \cap Q$.*

Appendix B

Isotropic Mesh Adaption

Here we briefly describe a mesh adaption strategy in order to refine and coarsen a triangular mesh isotropically. The ideas for the adaption procedure are essentially taken from the report of Hempel [39]. The aim is to refine and recoarsen conforming triangulations by dividing triangles into subtriangles, so-called *daughters*, of a common *mother*, or we unify daughters of a common mother, the so-called *sisters* and restore their mother. Therefore, it is possible to recoarsen a refined triangulation up to its initial state. We remark, that this type of recoarsening can also be extended to tetrahedral meshes, which are refined by the isotropic refinement algorithm in [14].

B.1 Input Parameters

The adaption procedure need three input parameters.

1. A conforming triangulation containing a set of triangles and a set of vertices,
2. a vector of integers (flags), that specifies for each triangle whether it has `to be refined` or `to be coarsened` or neither of both,
3. an array, that describes for each triangle the history of refinements that led to this triangle.

B.2 Red-Green-Refinement

This refinement procedure subdivides triangles in two different ways. The *red refinement* inserts the three midpoints of the triangle's edges as new vertices and creates four similar subtriangles as shown in Figure B.1. To avoid triangles with non-conforming nodes the *green refinement* is used. To this

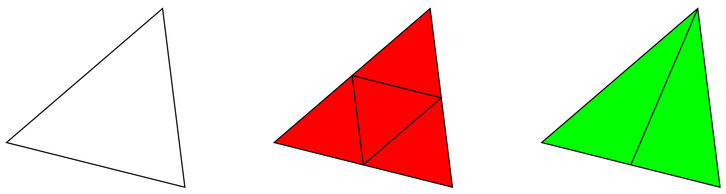


Figure B.1: An initial triangle and its red refinement and green refinement.

end, we divide a triangle with one non-conforming node into two subtriangles, the so-called *green triangles* along the median of the edge with the non-conforming node (see Figure B.1). All triangles, that have more than one non-conforming node are refined by red refinement. If a green triangle has to be refined, its mother is restored and then refined by red refinement.
Therefore, the resulting triangulation will always be conforming. Furthermore, the inner angles of the triangles are always limited from below for each adaption step and only depend on the initial triangulation.

B.3 Data Structure

In order to restore mothers of triangles, we need the information, which triangles are children of a particular mother. A suitable data structure for the book-keeping of triangle refinements can be created as described below.
Each triangle is assigned a data structure called `history` and has the following members:

`bool is_green` specifies, if a triangle is the result of a green refinement,

`integer green_sister` specifies the location of the sister, e.g. 0, 1, or 2 to determine, which of the neighbours the sister is,

`integer red_refinements` specifies the number of red refinements that led to a triangle

`stack red_history` contains for each red refinement the number of the child

For a red refined triangle the children are enumerated by 0, 1, 2, and 3 where child 0 is located at vertex 0 of the mother, child 1 at vertex 1 of the mother, etc. Child 3 is the inner triangle of a red refinement. The stack `red_history` contains for each red refinement the number of the child, where each daughter inherits this stack from her mother and puts her own number on top of this stack. With this information all ancestors of triangles can be restored. For full details see [39, 55].

B.4 Recoarsening

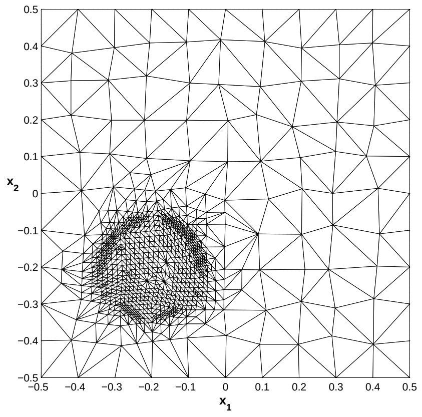
The recoarsening for this type of adaption algorithm is more involved than the refinement. In fact, a detailed description of the recoarsening algorithm requires several definition of a *simply red* triangle, a *thread* or a *resolvable patch*. However, this is beyond the scope of that brief appendix and the reader is referred to [39, 55] for details. In summary, the recoarsening strategy produces a conforming triangulation by restoring previously refined triangles. With a sequence of recoarsening steps it is possible to restore the state of the initial triangulation. The results of the following Section B.5 demonstrate the feasibility and performance of the isotropic refinement strategy.

B.5 Preliminary Results

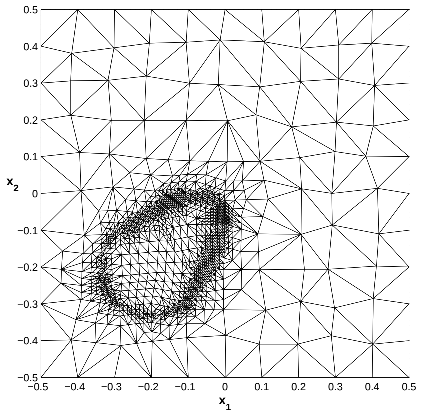
In Section 3.9 we solve the Burgers equation (3.40) with initial condition 3.44 with an adaptive ADER4 scheme. The refinement rule used in this simulation inserts the barycenter of a triangle as a new node into the current triangulation for each triangle, that is flagged `to be refined`. At the end of this refinement procedure, the nodes are retriangulated according to the Delaunay criterion. As this refinement strategy may produce degenerate, i.e. long and thin, triangles (see Figure 3.20), which affect the local accuracy, we here solve the same problem with the discussed isotropic mesh refinement strategy.

Initially, Ω is discretised by the same triangular mesh of 200 cells, which are successively adapted to the initial condition within a few iterations as shown in Figure B.2**(a)**. Here, the long and thin triangles at the transition zone between the coarse and fine discretisation are avoided compared to Figure 3.20**(a)**. In fact, the mesh gradually changes from coarse to fine triangular cells. As shown in Figure B.2**(b)**,**(c)**,**(d)** the discontinuity in the solution is well captured by the locally refined triangulation while degenerate triangles are avoided. The smallest inner angle is bounded from below due to the red-green refinement and is determined by the initial triangulation. However, the locally refined parts of the mesh in Figure B.2 are not as uniformly refined as in Figure 3.20, where Delaunay triangulations are used.

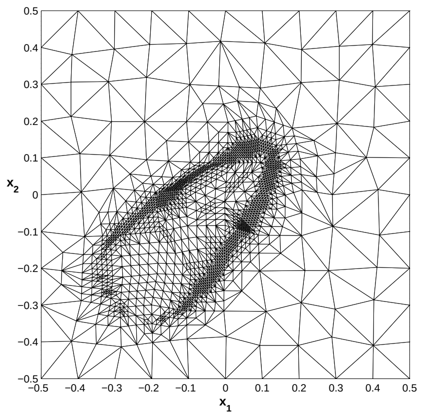
Plots of the solution u at the four different time steps $t = t_0$, $t = t_{100}$, $t = t_{300}$, and $t = t_{700}$ interpolated onto a regular 100×100 mesh are shown in Figure B.3. Especially at the discontinuities, we observe a major improvement (compare Figure 3.19). Encouraged by these promising results, the implementation of a fast and robust isotropic adaption algorithm is subject to current work.



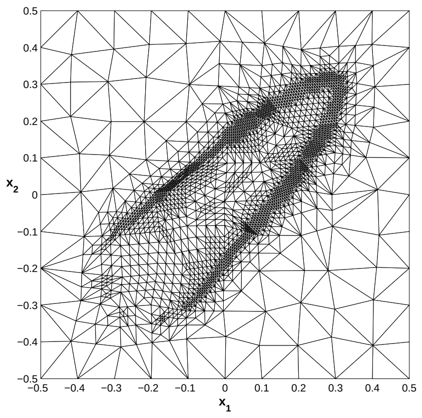
(a)



(b)

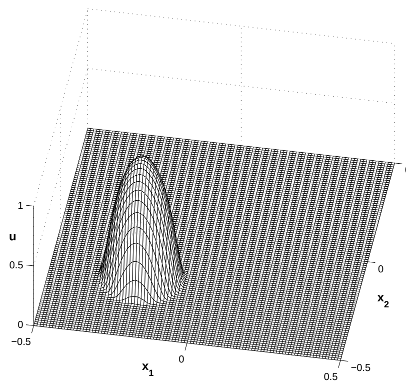


(c)

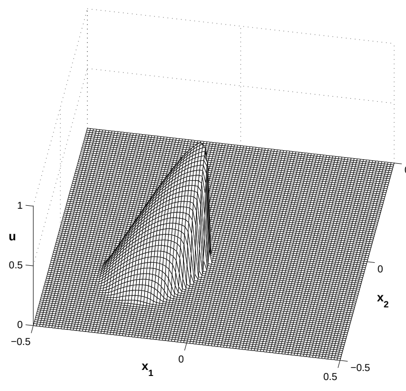


(d)

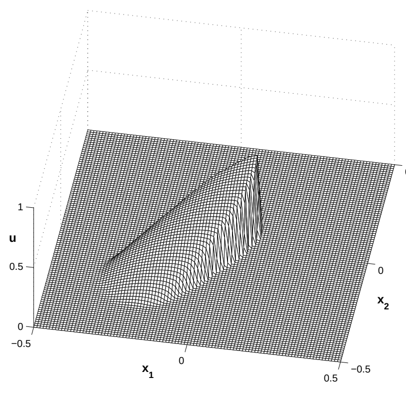
Figure B.2: Burgers equation. Adaptive triangulation during the simulation at four different times, **(a)** $t = t_0$; **(b)** $t = t_{100}$; **(c)** $t = t_{300}$; **(d)** $t = t_{700}$.



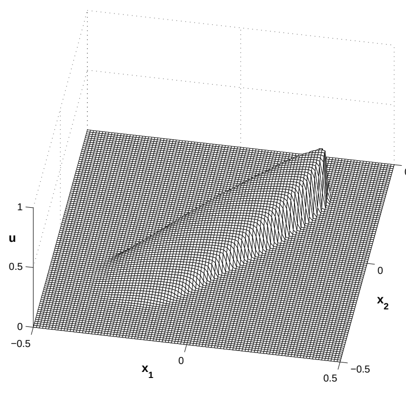
(a)



(b)



(c)



(d)

Figure B.3: Burgers equation. 3D view on the evolution of $\bar{u}(t)$ at four different times, **(a)** $t = t_0$; **(b)** $t = t_{100}$; **(c)** $t = t_{300}$; **(d)** $t = t_{700}$.

Appendix C

Numerical Integration

C.1 Quadrature Rules for Triangles

In this work, triangular and polygonal cells (Voronoi cells), which can always be broken down into a set of triangular subcells, are used extensively. In order to numerically compute the integral of a polynomial $p(x)$ over a triangle T with area $|T|$, there are exact quadrature formulas, such that

$$\int_T p(x)\,dx = \sum_{j=1}^{k} \omega_j\, p(\xi_j)\,,$$

where ω_j, $j = 1, ..., k$ are certain weights and ξ_j are the corresponding points inside the triangle T.
A list of the fundamental quadrature rules is given, where q indicates the maximum degree of the polynomial for which the formula is exact. We let $\mathbf{v}_j, j = 1, 2, 3$ be the vertices of the triangle T, then the locations ξ_j of the quadrature points in the triangle are given by $\xi_j = \alpha_j \mathbf{v}_1 + \beta_j \mathbf{v}_2 + \gamma_j \mathbf{v}_3$, where α, β, and γ are the so-called triangular coordinates. Because of the triangular symmetry of the quadrature formulas all quadrature points occur in groups of either one, three, or six. Thus, if a quadrature point has triangular coordinates $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ a single quadrature point occurs at the centroid of the triangle. If two triangular coordinates are equal, for example (a, b, b), then two further members of the same group occur with triangular coordinates (b, a, b) and (b, b, a) having the same weight. If none of the triangular coordinates is equal, i.e. (a, b, c), then there are five additional quadrature points of this group with triangular coordinates (a, c, b), (b, a, c), (b, c, a), (c, a, b), (c, b, a) with the same weight.
In Tables C.1 and C.2 only one point per group is given and the column *multiplicity* indicates, whether the point belongs to a group of one, three or six points. Details on the computation of the weights ω_j and the triangular coordinates α, β, γ are given in [22].

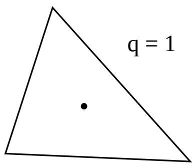
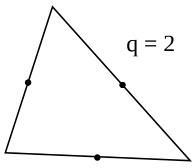
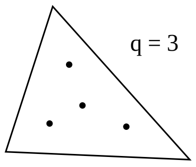
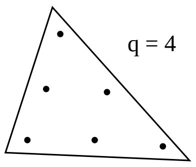
Figure	α β γ	ω	multiplicity
	$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$	1	1
	$\frac{1}{2}$ $\frac{1}{2}$ 0	$\frac{1}{3}$	3
	$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$	$-\frac{27}{48}$	1
	$\frac{3}{5}$ $\frac{1}{5}$ $\frac{1}{5}$	$\frac{25}{48}$	3
	0.816847572980459 0.091576213509771 0.091576213509771	0.109951743655322	3
	0.108103018168070 0.445948490915965 0.445948490915965	0.223381589678011	3

Table C.1: Quadrature rules for triangles

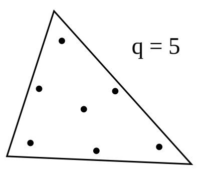
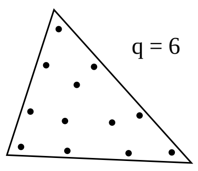
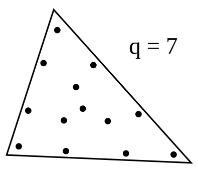
	$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$	0.225	1
	0.797426985353087 0.101286507323456 0.101286507323456	0.125939180544827	3
	0.470142064105115 0.470142064105115 0.059715871789770	0.132394152788506	3
	0.873821971016996 0.063089014491502 0.063089014491502	0.050844906370207	3
	0.501426509658179 0.249286745170910 0.249286745170910	0.116786275726379	3
	0.636502499121399 0.310352451033785 0.053145049844816	0.082851075618374	6
	$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$	−0.149570044467670	1
	0.479308067841923 0.260345966079038 0.260345966079038	0.175615257433204	3
	0.869739794195568 0.065130102902216 0.065130102902216	0.053347235608839	3
	0.638444188569809 0.312865496004875 0.048690315425316	0.077113760890257	6

Table C.2: Quadrature rules for triangles (continued)

Bibliography

[1] R. Abgrall. On essentially non-oscillatory schemes on unstructured meshes: Analysis and implementation. *J. Comput. Phys.*, 144:45–58, 1994.

[2] N. Albright, P. Concus, and W. Proskurowski. Numerical solution of the multi-dimensional Buckley-Leverett equation by a sampling method. *SPE 7681*, 1979.

[3] K. Aziz and A. Settari. *Petroleum Reservoir Simulation.* Applied Science, 1979.

[4] T. Barth and P. Frederickson. Higher order solution of the euler equations on unstructured grids using quadratic reconstruction. Technical Report No. 90-0013, AIAA Paper, 1990.

[5] M. Batdorf, L. A. Freitag, and C. F. Ollivier-Gooch. Computational study of the effect of unstructured mesh quality on solution efficiency. In *Proc. 13th AIAA Computational Fluid Dynamics Conference*, Snowmass Village, Colorado, 1997.

[6] A. Bayliss and E. Turkel. Radiation-boundary conditions for wave-like equations. *Comm. Pure and Appl. Maths.*, 33:707–725, 1980.

[7] J. Behrens. An adaptive semi-Lagrangian advection scheme and its parallelization. *Mon. Wea. Rev.*, 124:2386–2395, 1996.

[8] J. Behrens. Atmospheric and ocean modeling with an adaptive finite element solver for the shallow-water equations. *Appl. Numer. Math.*, 26:217–226, 1998.

[9] J. Behrens and A. Iske. Grid-free adaptive semi-lagrangian advection using radial basis functions. *Comput. Math. Appl.*, 43:319–327, 2002.

[10] J. Behrens, A. Iske, and M. Käser. Adaptive meshfree method of backward characteristics for nonlinear transport equations. In M. Griebel and M. A. Schweitzer, editors, *Meshfree Methods for Partial Differential*

Equations, volume 26 of *Lecture Notes in Computational Science and Engineering*, pages 21–36. Springer, Berlin, 2002.

[11] J. Behrens, A. Iske, and S. Pöhn. Effective node adaption for grid-free semi-lagrangian advection. In T. Sonar and I. Thomas, editors, *Discrete Modelling and Discrete Algorithms in Continuum Mechanics*, pages 110–119. Logos, Berlin, 2001.

[12] T. Belytschko, Y. Lu, and L. Gu. Element free galerkin methods. *Int. J. Numer. Meth. Eng.*, 37:229–256, 1994.

[13] V. Bjerknes. Das Problem der Wettervorhersage, betrachtet vom Standpunkt der Mechanik und der Physik. *Meteor. Zeits.*, 21:1–7, 1904.

[14] F. Bornemann, B. Erdmann, and R. Kronhuber. Adaptive Multilevel-Methods in Three Space Dimensions. *Int. J. Numer. Meth. Eng.*, 36:3187–3203, 1993.

[15] D. Braess. *Finite Elements.* Cambridge University Press, Cambridge, 2001.

[16] J. M. Buckley and M. C. Leverett. Mechanism of fluid displacement in sands. *Trans. AIME*, 146:107–116, 1942.

[17] J. M. Burgers. Application of a model system to illustrate some points of the statistical theory of free turbulence. In *Proc. Acad. Sci. Amsterdam*, volume 43, pages 2–12, 1940.

[18] J. Charney, R. Fjörtoft, and J. von Neumann. Numerical integration of the barotropic vorticity equation. *Tellus*, 2:237–254, 1950.

[19] C.Hu and C. Shu. Weighted essentially non-oscillatory schemes on triangular meshes. *J. Comput. Phys.*, 150:97–127, 1999.

[20] P. G. Ciarlet. *The finite element methods for elliptic problems.* North-Holland, Amsterdam, 1987.

[21] R. Courant, E. Isaacson, and M. Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Comm. Pure Appl. Math.*, 5:243–255, 1952.

[22] G. Cowper. Gaussian quadrature formulas for triangles. *Int. J. Numer. Meth. Eng.*, 7:405–408, 1973.

[23] P. Deuflhard and F. Bornemann. *Scientific Computing with Ordinary Differential Equations.* Springer, New York, 2002.

[24] C. A. Duarte. A review of some meshless methods to solve partial differential equations. Technical Report 95-06, Texas Institute for Computational and Applied Mathematics, University of Texas, Austin, 1995.

[25] J. Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In W. Schempp and K. Zeller, editors, *Constructive Theory of Functions of Several Variables*, pages 85–100. Springer, Berlin, 1977.

[26] D. Durran. *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics.* Springer, New York, 1999.

[27] K. Eriksson and C. Johnson. Adaptive finite element methods for parabolic problems I, a linear model problem. *SIAM J. Numer. Anal.*, 28:43–77, 1991.

[28] M. Falcone and R. Ferretti. Convergence analysis for a class of high-order semi-lagrangian advection schemes. *SIAM J. Numer. Anal.*, 35(3):909–940, 1998.

[29] O. Friedrich. Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids. *J. Comput. Phys.*, 144:194–212, 1998.

[30] J. Fürst and T. Sonar. On meshless collocation approximations of conservation laws: positive schemes and dissipation models. *ZAMM*, 81:403–415, 2001.

[31] C. Gauger, P. Leinen, and H. Yserentant. The finite mass method. *SIAM J. Numer. Anal.*, 37:1768–1799, 2000.

[32] S. K. Godunov. A finite difference method for the computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sb.*, 47:271–306, 1959.

[33] U. Göhner and G. Warnecke. A second order finite difference error indicator for adaptive transonic flow computations. *Numer. Math.*, 70:129–161, 1995.

[34] B. Gustafsson, H.-O. Kreiss, and J. Oliger. *Time Dependent Problems and Difference Methods.* John Wiley & Sons, New York, 1995.

[35] T. Gutzmer and A. Iske. Detection of discontinuities in scattered data approximation. *Numerical Algorithms*, 16(2):155–170, 1997.

[36] A. Harten. The artificial compression method for computation of shocks and contact discontinuities. iii. self-adjusting hybrid schemes. *Math. Comp.*, 32:363–389, 1978.

[37] A. Harten and S. Chakravarthy. Multi-dimensional ENO schemes for general geometries. Technical Report 91-76, ICASE, 1991.

[38] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy. Uniformly high order essentially non-oscillatory schemes, iii. *J. Comput. Phys.*, 71:231–303, 1987.

[39] D. Hempel. Isotropic refinement and recoarsening in 2 dimensions. Technical Report IB 223-95 A 35, DLR, 1995.

[40] C. Hirsch. *Numerical Computation of Internal and External Flows.* John Wiley & Sons, New York, 1990.

[41] H. Holden and N. H. Risebro. *Front Tracking for Hyperbolic Conservation Laws.* Springer, 2002.

[42] A. Iske. Scattered data modelling using radial basis functions. In A. Iske, E. Quak, and M. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, pages 205–242. Springer, Heidelberg, 2002.

[43] A. Iske and M. Käser. Conservative semi-Lagrangian advection on adaptive unstructured meshes. to appear in *Numer. Meth. Part. Diff. Eq.*

[44] G. S. Jiang and C. W. Shu. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.*, 126:202–228, 1996.

[45] M. Käser, H. Igel, M. Sambridge, and J. Brown. A comparative study of explicit differential operators on arbitrary grids. *J. Comput. Acoustics*, 9:1111–1125, 2001.

[46] D. Kröner. *Numerical Schemes for Conservation Laws.* Wiley & Teubner, 1997.

[47] J. Laprise and A. Plante. A class of semi-lagrangian integrated-mass (slim) numerical transport algorithms. *Mon. Wea. Rev.*, 123:553–565, 1995.

[48] C. Lawson and R. Hanson. *Solving Least Squares Problems*, volume 15 of *Classics in Applied Mathematics.* SIAM, Philadelphia, 1995.

[49] P. Lax and B. Wendroff. Systems of conservation laws. *Comm. Pure Appl. Math.*, 13:217–237, 1960.

[50] R. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge University Press, Cambridge, 2002.

[51] K.-A. Lie and S. Noelle. On the artificial compression method for second-order nonoscillatory central difference schemes for systems of conservation laws. *SIAM J. Sci. Comput.*, 24:1157–1174, 2003.

[52] T. Liszka. An interpolation method for an irregular set of nodes. *Int. J. Numer. Meth. Eng.*, 20:1599–1612, 1984.

[53] G. R. Liu. *Mesh Free Methods. Moving beyond the Finite Element Method.* CRC Press, New York, 2003.

[54] X. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.*, 115:200–212, 1994.

[55] A. Meister and J. Struckmeier. *Hyperbolic Partial Differential Equations: Theory, Numerics and Applications.* Vieweg, Braunschweig, 2002.

[56] J. J. Monaghan. An introduction to sph. *Comput. Phys. Comm.*, 48:89–96, 1988.

[57] K. W. Morton. *Numerical Solution of Convection-Diffusion Problems.* Chapman & Hall, London, 1996.

[58] C. D. Munz and R. Schneider. An arbitrary high order accurate finite volume scheme for the maxwell equations in two dimensions on unstructured meshes. Report.

[59] E. O. nate, S. Idelsohn, O. C. Zienkiewicz, and R. L. Taylor. A finite point method in computational mechanics. applications to convective transport and fluid flow. *Int. J. Numer. Meth. Eng.*, 39:3839–3866, 1996.

[60] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: diffuse approximation and diffuse elements. *Comput. Mech.*, 10:307–318, 1992.

[61] H. Nessyahu, T. Tassa, and E. Tadmor. The convergence rate of godunov type schemes. *SIAM J. Numer. Anal.*, 31:1–16, 1994.

[62] C. Ollivier-Gooch. Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction. *J. Comput. Phys.*, 133:6–17, 1997.

[63] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, Cambridge, 1993.

[64] T. Phillips and A. J. Williams. Conservative semi-lagrangian finite volume schemes. *Numer. Meth. Part. Diff. Eq.*, 17:384–391, 2001.

[65] F. Preparata and M. I. Shamos. *Computational Geometry.* Springer, New York, 1985.

[66] A. Priestley. A quasi-conservative version of the semi-lagrangian advection scheme. *Mon. Wea. Rev.*, 121:621–629, 1993.

[67] M. Rancic. Semi-lagrangian piecewise biparabolic scheme for two-dimensional horizontal advection of a passive scalar. *Mon. Wea. Rev.*, 120:1394–1406, 1992.

[68] P. W. Randles and L. D. Libersky. Smoothed particle hydrodynamics: Some recent improvements and applications. *Comput. Methods Appl. Mech. Engrg.*, 139:375–408, 1996.

[69] L. Richardson. *Weather Prediction by Numerical Process.* Cambridge University Press, 1922.

[70] A. Robert. A stable numerical integration scheme for the primitive meteorological equations. *Atmosphere-Ocean*, 19:35–46, 1981.

[71] T. J. Ruuth and R. J. Spiteri. Two barriers on strong-stability-preserving time discretization methods. *J. Sci. Comput.*, 17:211–220, 2002.

[72] T. Schwartzkopff, C. D. Munz, and E. F. Toro. Ader: A high-order approach for linear hyperbolic systems in 2d. *J. Sci. Comput.*, 17:231–240, 2002.

[73] T. Schwartzkopff, C. D. Munz, E. F. Toro, and R. C. Millington. The ader approach in 2d. In T. Sonar and I. Thomas, editors, *Discrete Modelling and Discrete Algorithms on Continuum Mechanics*, pages 207–216. Logos, Berlin, 2001.

[74] H. R. Schwarz. *Finite Element Methods.* Academic Press, London, 1988.

[75] J. Scroggs and F. Semazzi. A conservative semi-lagrangian method for multidimensional fluid dynamics applications. *Numer. Meth. Part. Diff. Eq.*, 11:445–452, 1995.

[76] M. I. Shamos. Geometric complexity. In *Proc. of 7th Annual ACM Symposium on Theory of Computing*, pages 224–233, 1975.

[77] C. W. Shu. Total-variation-diminishing time discretizations. *SIAM J. Sci. Stat. Comput.*, 9:1073–1084, 1988.

[78] C. W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

[79] T. Sonar. On the construction of essentially non-oscillatory finite volume approximations to hyperbolic conservation laws on general triangulations: Polynomial recovery, accuracy and stencil selection. *Comput. Methods Appl. Mech. Engrg.*, 140:157–181, 1997.

[80] E. Süli. Finite volume methods on distorted meshes: stability, accuracy, adaptivity. Technical report, Oxford Univ. Comput. Lab., 1989.

[81] V. A. Titarev and E. F. Toro. Ader: Arbitrary high order Godunov approach. *J. Sci. Comput.*, 17:609–618, 2002.

[82] E. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer, 2 edition, 1999.

[83] E. Toro and V. Titarev. Very high order Godunov-type schemes for nonlinear scalar conservation laws. In *ECCOMAS Computational Fluid Dynamics Conference*, Swansea, Wales, September 2001. European Congress on Computational Methods in Applied Sciences and Engineering.

[84] E. F. Toro, R. C. Millington, and L. A. M. Nejad. Towards very high order godunov schemes. In E. F. Toro, editor, *Godunov methods; Theory and applications*, pages 907–940, Oxford, 2001. Kluwer Academic Plenum Publishers. International Conference.

[85] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics. The Finite Volume Method.* Longman Scientific & Technical, New York, 1995.

[86] H. J. Welge. A simplified method for computing oil recovery by gas or water drive. *Trans. AIME*, 195:97–108, 1952.

[87] Z. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. *IMA J. Numer. Anal.*, 13:13–27, 1993.

[88] H. Yang. An artificial compression method for ENO schemes: The slope modification method. *J. Comput. Phys.*, 89:125–160, 1989.

[89] S. T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, 31:335–362, 1979.

[90] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method*, volume 1. McGraw Hill, New York, 1989.