universität wien

DISSERTATION

Titel der Dissertation

Numerical Grids for Spherical Shells and Other Complex Domains

Verfasser

Mag. rer. nat. Hannes Grimm-Strele

angestrebter akademischer Grad

Doktor der Naturwissenschaften (Dr. rer. nat.)

Wien, im November 2013

Studienkennzahl lt. Studienblatt:	A 791 405
Studienrichtung lt. Studienblatt:	Mathematik
Betreuer:	Univ.-Prof. Dr. Herbert J. Muthsam

Abstract

In this thesis, we present several ways to design numerical grids covering spherical shells and other complex domains. Cartesian and spherical grids which by now are nearly exclusively used in numerical astrophysics have certain deficiencies when applied to these domains. When applying Cartesian grids, around 50 % of the grid cells lie outside of the star. For spherical coordinate systems, grid lines converge at the poles and at the core, requiring special treatment of these regions and making the simulation inefficient due to converging grid lines resulting in small time steps.

We can get rid of these problems using overlapping grids, but we either limit the accuracy of our solution or destroy the exact conservation of conserved properties by the interpolation between the grids. Noise is generated at the grid boundaries which adversely affects the stability of the simulation.

Instead, we suggest to use curvilinear grids. Using non-smooth mapping functions, we can create a structured grid which completely covers circles or spheres. We show how these grids are implemented in simulation codes designed for Cartesian coordinate systems. Since the mapping functions we intend to use are non-smooth, the numerical errors are large when using a finite difference formulation. For a WENO code we show both theoretically and by numerical experiment that numerical results using a finite volume formulation are reasonably accurate, whereas the numerical error with the finite difference formulation is unacceptably large. Furthermore, we show using a PPM code that split time integration leads to huge errors, too, and present ways to rewrite existing codes to unsplit time integration schemes. If these conditions are fulfilled, the numerical results are satisfactory on the proposed non-smooth grids.

As a side topic, we investigate the practical efficiency of several explicit Runge–Kutta scheme with differing order of accuracy and error constants for the problem of solar surface convection and with WENO schemes for spatial discretisation. We show that in some situations, higher order time integration schemes are less efficient than lower order ones due to smaller error constants.

Zusammenfassung

In der vorliegenden Dissertation werden verschiedene Gitter vorgestellt, mit denen man numerische Simulationen auf Kugelschalen und anderen kugelförmigen Gebieten durchführen kann. Kartesische und sphärische Gitter, wie sie nahezu ausschließlich in der numerischen Astrophysik verwendet werden, sind für dieses Problem ineffizient oder nicht direkt anwendbar. Wenn man einen Stern mit einer kartesischen Box umschreibt, liegen etwa 50 % des Volumens außerhalb der Kugel. Bei sphärischen Koordinatensystemen konvergieren Gitterlinien an den Polen und im Kern, was neben den analytischen noch schwer wiegende numerische Konsequenzen hat, da der Zeitschritt durch die kleinsten Gittermaschenweiten beschränkt wird.

Überlappende Gitter lösen das Problem der Singularitäten, jedoch wird durch die Interpolation zwischen den Gittern entweder die Genauigkeitsordnung der numerischen Lösung beschränkt oder die numerische Erhaltung der Erhaltungsgrößen geht verloren. An den Übergängen zwischen den Gittern entsteht durch die Interpolation Rauschen, das die Stabilität der Simulation verschlechtert.

Wir schlagen stattdessen die Verwendung von krummlinigen Koordinatensystemen vor. Mit Hilfe nichtglatter Abbildungsfunktionen können wir ein einziges strukturiertes Gitter erzeugen, das kreis- und kugelförmige Gebiete vollständig überdeckt. Wir legen dar, wie krummlinige Koordinaten in bestehende, für kartesische Koordinaten entwickelte Codes eingebaut werden können. Auf Grund der Nichtglattheit der Abbildungsfunktion sind die numerischen Fehler im Falle eines Finite Differenzen–Codes groß. Wir zeigen für einen WENO-Code sowohl theoretisch als auch durch numerische Experimente, dass eine Finite Volumen–Formulierung Ergebnisse mit guter Genauigkeit liefert, während eine Finite Differenzen–Formulierung zu inakzeptablen Fehlern führt. Ebenso zeigen wir für einen PPM-Code, dass die verschiedenen räumlichen Richtungen nicht nacheinander, sondern gleichzeitig zeitlich integriert werden müssen. Wenn diese Bedingungen erfüllt sind, liefern die verwendeten Codes zufriedenstellende Ergebnisse auf den vorgeschlagenen nichtglatten Gittern.

Darüber hinaus untersuchen wir die Effizienz mehrerer expliziter Runge–Kutta–Schemata, die sich in Genauigkeitsordnung und den Fehlerkonstanten unterscheiden, für die Simulation von solarer Oberflächenkonvektion mit WENO–Methoden. Unter bestimmten Umständen führen die unterschiedlichen Fehlerkonstanten dazu, dass Zeitintegrationsverfahren niedriger Ordnung effizienter sind als solche höherer Ordnung.

Contents

1 Introduction

The equations of fluid mechanics, i.e. the Navier–Stokes and the Euler equations, are analytically solvable only in few special cases. Since their solution is of great interest for many fields of science, the numerical solution of these equations is of great importance. The numerical approach to solving these equations is called *computational fluid dynamics (CFD)*. It can be seen as a "sub–field of either fluid dynamics or numerical analysis" (Ferziger and Perić, 2002), emphasizing the interdisciplinary character of this field.

In practice, when we speak about CFD, we have engineering applications in mind. But also in astrophysics there are many problems which require the solution of the equations of fluid mechanics as we will describe in Chapter 2. The exchange of methods and experiences can be very useful, since the computational problems are similar.

Nevertheless, there are certain differences between engineering and astrophysical applications. The parameter space in terms of length and time scales as well as viscosity and (heat) conduction differs by several orders of magnitude. This implies that astrophysicists will, for a long time from now, not be able to effectively resolve all scales in their simulations. In the terminology from Section 1.2, astrophysical simulations follow the *large eddy simulation* paradigm.

Whereas engineers often encounter complicated geometries in their applications as, e.g., airfoils, the geometry of most astrophysical applications is rather simple. In most cases, the simulation domain can be either chosen to be a Cartesian box or a spherical cone. Because of this, engineers tend to use complicated grids which limits the order of accuracy of the numerical method (Ferziger and Perić, 2002), whereas astrophysicists use high–order methods on regular grids.

However, in this thesis we will show that the standard grids used in numerical astrophysics are sometimes not sufficient and limit the applicability and physical relevance of the simulation code. Instead, techniques from engineering applications can be used to extend the grid capabilities of the existing astrophysical simulation codes.

We only mention that numerical astrophysics encomprises a wide variety of additional topics as, e.g., self–gravity, special and general relativity, magnetic fields, and radiation transport. We will shortly mention some of these topics in Chapter 2. All numerical methods in this thesis deal with the equations of fluid mechanics exclusively. We add that parts of this thesis, in particular from Chapter 5, are published in Grimm-Strele et al. (2013b).

1.1 Numerical Grids for Astrophysical Simulations

There are many astrophysical applications where the physical domain of interest is a sphere or a circle, e.g. the numerical simulation of core convection (Browning et al., 2004; Cai et al., 2011) or of convection in giant planets (Evonuk and Glatzmaier, 2006, 2007). The usability of spherical coordinate systems is restricted due to the grid singularity in the centre of the sphere as discussed, for instance, by Evonuk and Glatzmaier (2007). With a Cartesian grid, a huge part of the computational resources are wasted (Freytag et al., 2002) and to improve resolution along spheres, complex adaptive mesh refinements have to be used (Zingale et al., 2013) to keep the computational requirements manageable. Therefore, all default grids used in numerical astrophysics have some fatal deficiencies.

For the specific case of a sphere with grid singularity at the centre, Cai et al. (2011) proposed a different approach. They use spectral expansion methods on a spherical grid. To avoid the time step restriction due to converging grid lines at the centre, they lower the order of the harmonic expansion at the centre. The equations are recast in a form such that boundary conditions can easily be applied at the centre. In this way, the simulation domain can be extended to the full sphere. Anyway, their procedure still requires the specification of a boundary condition at the centre and applies only to spherical domains.

Mocz et al. (2013) implemented a discontinuous Galerkin method on arbitrary static and moving Voronoi meshes. In theory, their approach promises great flexibility and wide applicability. In applications, however, there are still numerical difficulties present, e.g. in the treatment of shocks, making the use of the method in astrophysical applications difficult at the moment.

In this paper, we present the methods used to extend the applicability of the simulation codes ANTARES (Muthsam et al., 2010a) and Prometheus (Müller et al., 1991) to more general geometries. Until now, ANTARES was exclusively applied to numerical simulations of solar and stellar surface convection and stellar interiors in Cartesian geometry (e.g., Muthsam et al., 2010a; Happenhofer et al., 2013) as well as convection in Cepheids in spherical geometry (e.g., Muthsam et al., 2010b; Mundprecht et al., 2013). For the inviscid part of the Navier–Stokes equations, the WENO finite difference scheme is employed (Shu and Osher, 1988; Shu, 2003; Merriman, 2003). The WENO scheme is a highly efficient shock-capturing scheme which can be implemented at several different orders of accuracy. In this paper, we consider the fifth order variant called WENO5. Its superiority compared to other high–order schemes was shown, e.g., in Muthsam et al. (2007). Nevertheless, its applicability is restricted by its specific requirements concerning the grid geometry (Merriman, 2003).

Prometheus, on the other hand, is a finite volume code using the PPM scheme (Colella and Woodward, 1984) for spatial discretisation. Its main application is the simulation of supernova explosions in spherical geometry. Recently, Wongwathanarat et al. (2010) implemented the "Yin–Yang" grid which is a combination of two spherical grids. Its advantage is that, in contrast to a standard spherical grid, the cell sizes are quasi–

uniform.

The Yin–Yang grid is a special case of a *composite grid* (Chesshire and Henshaw, 1990; Ferziger and Perić, 2002). In this approach, the computational domain is covered with several overlapping or patched grids. Usually, each of these grids has a rather simple geometry where efficient algorithms of high accuracy are easily available. Information at the grid boundary and in the overlap must be exchanged by some interpolation process. The numerical consequences are described and verified by numerical experiments in Chapter 4.

The technique of mapped grids is widely used in engineering (Wesseling, 2001; Ferziger and Perić, 2002; LeVeque, 2004), but until now was only seldomly applied in an astrophysical setting (Kifonidis and Müller, 2012). In principle, given a suitable mapping function, any problem defined on a general domain can be transformed to a problem in a computational space which is equidistant and Cartesian and where any standard numerical scheme, the applicability of which often is restricted to Cartesian and equidistant grids, can be used. The only requirement is that the grid in physical space is structured.

Shu (2003) applied the WENO finite difference scheme in a straightforward way to smooth grids[1]. In all numerical examples in the mentioned paper, the Mach number $\mathrm{Ma} = \frac{|u|}{v_{\mathrm{snd}}}$ was higher than 1.

The behaviour of the method in the low Mach number limit on Cartesian grids was investigated in Happenhofer et al. (2013). They showed that the WENO5 finite difference scheme does not perform well for Mach numbers smaller than 0.1 even on Cartesian grids.

In Chapter 5, we will show how well the WENO finite difference and finite volume scheme and the PPM scheme with several time integration methods performs on smooth and non–smooth grids for flows in the intermediate flow regime of $0.1 \leq \mathrm{Ma} \leq 1$.

Most of the findings of this thesis are not restricted to our specific code, but apply to any finite difference or finite volume code. From the numerical experiments in Chapter 5, we conclude in which situations and in which numerical setup the mapped grid technique gives reliable results. Thereby, we concentrate on the WENO and the PPM algorithm and on astrophysical simulations. We demonstrate the usefulness and applicability of the mapping functions for a sphere from Calhoun et al. (2008) in this setting.

1.2 Time and Length Scales of Astrophysical Simulations

The *Reynolds number* Re of a flow is a dimensionless quantity measuring the strength of inviscid terms in the Navier–Stokes equations compared to the viscous ones (Pope, 2000). It is defined as

$$\mathrm{Re} = \frac{L\,U}{\nu}, \tag{1.1}$$

where L and U are the macroscopic length scale and the velocity of the physical problem, and ν is the kinematic viscosity. Kupka (2009b) and Freytag et al. (2012) estimated Re

[1] In this thesis, we will call a grid (non)smooth if the associated mapping function is (non)smooth.

for the case of solar surface convection to be of order 10^{10} at least, varying throughout the whole star.

A turbulent flow contains a wide range of length scales. According to Pope (2000) and Canuto et al. (1988), the standard model of energy transport for turbulent flows is that energy is fed in at the largest length scale, the so–called "integral scale" L of the problem, and transported to the smallest scales l where the energy is dissipated by the molecular viscosity. Using the Reynolds number of the problem, the ratio of these two scales can be determined by

$$\frac{L}{l} = \mathrm{Re}^{\frac{3}{4}}. \tag{1.2}$$

For simulations of solar surface convection, L is several Mm. The flow at the surface is expected to be turbulent. Therefore, with the above formula the dissipation scale l is of magnitude cm.

This implies that a numerical simulation for solar surface convection can never resolve all relevant scales. This would require an enourmous amount of grid points which is not feasible with today's computational resources. Instead, these simulations follow the *large eddy simulation* (LES) paradigm. The idea of this approach is that whereas the large eddies depend on the geometry of the problem, the small eddies are self–similar (Pope, 2000). In an LES, the grid resolution corresponds to a sufficiently fine length scale where the eddies are already self–similar. All smaller scales are modelled either by an explicit *subgrid model* as the famous Smagorinsky model (Smagorinsky, 1963) or by the numerical viscosity of the scheme (Grinstein et al., 2007).

More generally, astrophysical simulations embrace several orders of magnitude of spatial scales. The length scales as well as the viscosity in astrophysical problems are such that they cannot be resolved in a numerical simulation. Therefore, the vast majority of astrophysical simulations are large eddy simulations. As a side remark, these conditions can also not be reached in terrestrial experiments. This limits the possibilities to validate the simulation data as outlined in Chapter 2. Finally, we note that similar to the spatial scales, the time scales of an astrophysical problem can differ by several orders of magnitude (e.g., Kupka, 2009b).

This has several implications for the design of the numerical grid. The grid size should not be dictated by the coordinate system but be flexible to be increased or decreased depending on the flow regime. Small cells as they occur near the poles of a spherical coordinate system limit the time step and increase the computational costs of a simulation (e.g., Washington et al., 2009). In Chapter 5, we will describe the design and implementation of grids for a sphere which fulfil these requirements.

2 Process of Numerical Modelling

As described in Ferziger and Perić (2002), the numerical modelling process consists of two steps. Given a physical problem, the first step is picking the set of appropriate mathematical equations which describe the problem in sufficient detail. In many cases these will be partial differential equations. These equations may contain analytical simplifications to decrease the complexity of the model. The second step is designing and implementing the numerical scheme solving these equations. Of course, the choice of the analytical model influences the design of the numerical method. There is no efficient numerical method which is suited for all types of equations. Finally, the correctness and the physical realism of the numerical model must be demonstrated.

In the following, we will illustrate the modelling process by an example. Then, we will outline the main difficulties of developing codes which provide realistic results for the problem at hand, and how their correctness can be checked.

2.1 Analytical Model

As an example for the first step, we consider the Sun. A schematic view of the inner structure of the Sun can be found in Figure 2.1.

The equations governing the flow of the plasma are the Navier–Stokes equations. Following Chorin and Marsden (1993) and Obertscheider (2007), they take the form

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{2.1a}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u} + p\,\mathrm{Id}) = \rho \mathbf{g} + \nabla \cdot \tau, \tag{2.1b}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot (\mathbf{u}\,(E + p)) = \rho\,(\mathbf{g} \cdot \mathbf{u}) + \nabla \cdot (\mathbf{u} \cdot \tau) + Q_{\mathrm{rad}}, \tag{2.1c}$$

neglecting the magnetic field terms and assuming the fluid to be Newtonian. The meaning and units of all variables is shown in Table 2.1. Id is the identity matrix. An equation of state must be specified to complete this set of equations. The viscous stress tensor $\tau = (\tau_{i,j})_{i,j=1,2,3}$ is given by

$$\tau_{i,j} = \eta \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{i,j} (\nabla \cdot \mathbf{u}) \right) + \zeta\, \delta_{i,j} (\nabla \cdot \mathbf{u}). \tag{2.2}$$

$\mathbf{g}$ is the gravity vector and Q_{rad} is the radiative heating rate describing the energy

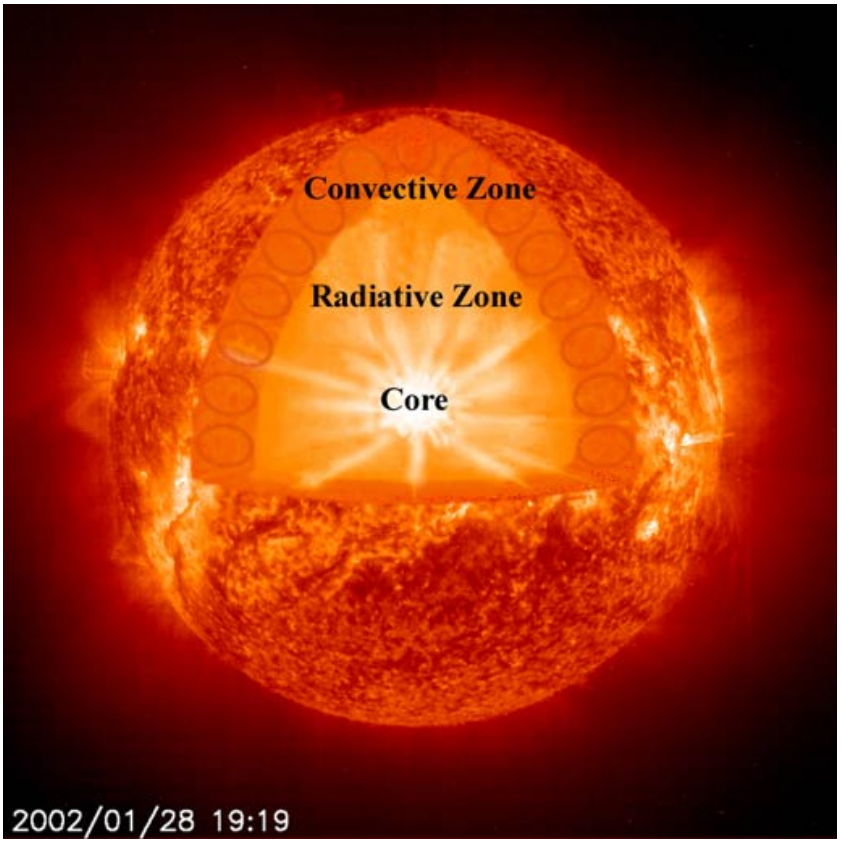

Figure 2.1: A schematic view of the Sun. Picture taken from `http://sunearthday.nasa.gov/`. In the core, energy is released by nuclear burning. In the inner two thirds, the energy is transported by radiation. In the outer third, it is transported by turbulent convection. Near the surface of the Sun, the scales get smaller, and the turbulence manifests in the solar granulation.

exchange between gas and radiation. $\delta_{i,j}$ is the Kronecker symbol. η and ζ are the first and second coefficients of viscosity.

We can rewrite equations (2.1) as

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F}_{\text{adv}} = \nabla \cdot \mathbf{F}_{\text{visc}} + \mathbf{S} \tag{2.3a}$$

with

variable	meaning	unit (CGS)
ρ	gas density	$\mathrm{g\,cm^{-3}}$
T	temperature	K
p	pressure	$\mathrm{dyn\,cm^{-2}}$
u	x velocity (vertical)	$\mathrm{cm\,s^{-1}}$
v	y velocity (horizontal)	$\mathrm{cm\,s^{-1}}$
w	z velocity (horizontal)	$\mathrm{cm\,s^{-1}}$
μ_x	x momentum density (vertical)	$\mathrm{g\,s^{-1}\,cm^{-2}}$
μ_y	y momentum density (horizontal)	$\mathrm{g\,s^{-1}\,cm^{-2}}$
μ_z	z momentum density (horizontal)	$\mathrm{g\,s^{-1}\,cm^{-2}}$
Q_{rad}	radiative heating rate	$\mathrm{erg\,s^{-1}\,cm^{-3}}$
v_{snd}	sound speed	$\mathrm{cm\,s^{-1}}$
E	total energy	$\mathrm{erg\,cm^{-3}}$
e	internal energy	$\mathrm{erg\,cm^{-3}}$
ϵ	specific internal energy	$\mathrm{erg\,g^{-1}}$
η	dynamic viscosity	$\mathrm{g\,cm^{-1}\,s^{-1}}$
ζ	second (bulk) viscosity	$\mathrm{g\,cm^{-1}\,s^{-1}}$

Table 2.1: Variable names, meaning and CGS units as used in this thesis. Please note that x denotes the vertical direction. Vectors are written in bold face. The velocity vector is $\mathbf{u} = (u, v, w)^T$.

$$\begin{aligned} \mathbf{Q} = \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ E \end{pmatrix}, \mathbf{F}_{\mathrm{adv}} = \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u}\mathbf{u} + p\,\mathrm{Id} \\ \mathbf{u}\,(E+p) \end{pmatrix}, \\ \mathbf{F}_{\mathrm{visc}} = \begin{pmatrix} 0 \\ \tau \\ \mathbf{u}\cdot\tau \end{pmatrix}, \mathbf{S} = \begin{pmatrix} 0 \\ \rho\mathbf{g} \\ \rho\,(\mathbf{g}\cdot\mathbf{u}) + Q_{\mathrm{rad}} \end{pmatrix}. \end{aligned} \tag{2.3b}$$

We call the terms collected in $\mathbf{F}_{\mathrm{adv}}$ the *advective* or *inertial* part and in $\mathbf{F}_{\mathrm{visc}}$ the *viscous* part of the Navier–Stokes equations. All first derivatives are contained in $\mathbf{F}_{\mathrm{adv}}$, all second order terms in $\mathbf{F}_{\mathrm{visc}}$.

However, due to the large time and length scales, it is impossible with today's computer resources to simulate the whole Sun in one simulation, solving the full set of equations (Freytag et al., 2012). Instead, every numerical simulation can only cover a small subproblem.

For the simulation of sunspots, the full magnetohydrodynamics (MHD) equations must be solved with realistic equation of state (e.g., Rempel et al., 2009). In the quiet Sun, the influence of the magnetic field is negligible, but the radiative heating rate must be modelled suitably. The frequency-dependent intensity I_ν can be calculated by the radiative transfer equation

$$\left(\frac{1}{c}\frac{\partial}{\partial t} + \mathbf{r}\cdot\nabla\right) I_\nu = \rho\chi_\nu \left(S_\nu - I_\nu\right), \tag{2.4}$$

where ν is the frequency, S_ν is the source function and χ_ν the opacity of the material (Obertscheider, 2007). Defining the radiative energy flux $\mathbf{F}_{\text{rad}}$ by

$$\mathbf{F}_{\text{rad}} = \int_\nu \oint I_\nu(\mathbf{r},\mathbf{n})\,\mathbf{n}\,d\omega\,d\nu, \tag{2.5}$$

where ω is the solid angle and $\mathbf{n}$ the surface normal, the heating rate Q_{rad} can be calculated by

$$Q_{\text{rad}} = -\nabla\cdot\mathbf{F}_{\text{rad}}. \tag{2.6}$$

For details, we refer to Mihalas (1978). In optically thick regions, the calculation of $\mathbf{F}_{\text{rad}}$ can be simplified by means of the diffusion approximation. Then,

$$\mathbf{F}_{\text{rad}} = \kappa\nabla T, \tag{2.7}$$

with the radiative conductivity κ. Due to the choice of our coordinate system with the x direction pointing inwards the star, $\mathbf{F}_{\text{rad}}$ is positive with this definition as long as T increases with depth.

In the Sun, most of the material is ionised for temperatures above around $10^5\,\text{K}$. Energy is mainly transported by radiation. In this region, the appropriate equation is the ideal gas equation of state for a fully ionized plasma,

$$p = \frac{r_{\text{gas}}}{\mu}\,\rho\,T, \tag{2.8}$$

where r_{gas} is the universal gas constant and μ is the mean molecular weight (Nordlund et al., 2009).

The deep interior of the solar convection zone is nearly adiabatic. Here, the anelastic approximation is valid. Its basic idea is to filter out fast sound waves analytically since they do not contribute much to the convective motions of the fluid (Lilly, 1996).

More precisely, in the anelastic approximation we assume that the deviations of thermodynamic variables from a horizontally uniform reference state are small. The continuity equation in (2.1) is replaced by

$$\nabla\cdot(\rho_0\mathbf{u}) = 0, \tag{2.9}$$

where ρ_0 is some reference density value, neglecting the time derivative of density. In a stratified atmosphere, ρ_0 usually refers to the hydrostatic density stratification fulfilling

$$\nabla p_0 = \rho_0\mathbf{g} \tag{2.10}$$

with the mean pressure p_0 (Brown et al., 2012).

There are several formulations of the anelastic approximation, differing, amongst oth-

ers, in terms of energy conservation. Their formulations and behaviour is compared in Brown et al. (2012). Due to its initial assumptions the anelastic approximation yields qualitatively wrong results in regions where these assumptions are violated, as e.g., near the stellar surface in simulations of stellar convection, but reduces the complexity in equations (2.1) considerably in regions where it is applicable.

Another, even more restrictive simplification is the Boussinesq approximation (Lilly, 1996). It neglects all density-related changes except in the buoyancy term, and is applicable only to low Mach number flows and very narrow simulation boxes (Zaussinger and Spruit, 2013).

The Boussinesq approximation is a special case of an *incompressible* flow. A fluid is called *incompressible*, if the density of a fluid particle is constant following the fluid (Chorin and Marsden, 1993). In precise terms, the *Lagrangian derivative* $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$ of the mass density ρ is 0. This implies $\nabla \cdot \mathbf{u} = 0$ by virtue of the equation of continuity. This condition eliminates the need for an energy conservation equation as the third equation in (2.1). The pressure is then obtained by reformulating the divergence constraint. The resulting equation is of elliptic type (Lilly, 1996). We note that for the anelastic approximation, the reference state stays compressible (Lilly, 1996; Brown et al., 2012).

In some situations, it is necessary to describe the fluid as a multi-component fluid, e.g, as a mixture of hydrogen and helium, or other heavier elements. For each component, a separate continuity equation is added to equations (2.1), and the momentum and energy equations are changed accordingly (Kupka, 2009b).

We conclude that depending on the problem of interest, the Navier–Stokes equations (2.1) can either be simplified considerably or additional terms must be added, increasing or decreasing the complexity of the analytical model considerably. The most general approach is often not affordable in terms of computation time, as outlined in Chapter 1 for the case of stellar surface convection. Details can be found in Kupka (2009b) and Freytag et al. (2012). Selecting simplifications of the model equations, however, limits the physical relevance of the simulation. This is the case, e.g., for the anelastic approximation. Its limitations are described in Brown et al. (2012). Anyway, analytical solutions are only available in the simplest cases being of little physical relevance. The only way to solve these equations is by performing numerical simulations. The analytical model must be complex enough to capture the essential physical properties of the system and, at the same time, allow its numerical solution with reasonable computational resources.

Alternatively, reformulating and appropriately discretising the Navier–Stokes equations (2.1) may lead to a system of discrete equations which avoid the problems of the full set of analytical equations. A successful example for this approach in the case of the Navier–Stokes equations in the low Mach number regime is the method from Kwatra et al. (2009) and Happenhofer et al. (2013) where the stability and accuracy of the numerical scheme could be improved considerably without losing the full compressibility of the system. In this approach, the discretisation of the equations is part of the modelling process. In the next section, we will focus on the numerical treatment of the equations.

2.2 Numerical Simulation

After the analytical model has been chosen, the numerical method must be designed and implemented. The model formulation and the expected flow regime impose specific requirements on the numerical method. For numerical hydrodynamical simulations, the continuous flow equations must be discretised on a suitable grid. We will exemplify this again by simulations of the Sun.

For simulations of solar surface convection, we expect strong shocks and Mach numbers of order 1. The Reynolds number, a parameter relating the strength of the advective and the viscous terms in the Navier–Stokes equations, is very high. Therefore, the numerical method must be shock-capturing and stable, without adding to much numerical dissipation. The time integration can be done explicitly, since the advection time scale is the most restrictive one. WENO methods as used, e.g., in Muthsam et al. (2007) and Muthsam et al. (2010a) are an efficient choice yielding very accurate results.

For simulations of the deeper parts of the solar convection zone, the flow is rather smooth, nearly incompressible and very subsonic. Most upwind schemes, which are very well suited for the simulation of surface convection, do not perform well in this regime (Guillard and Viozat, 1999; Guillard and Murrone, 2004). This stems from the fact that the advective part of the Navier–Stokes equations called $\mathbf{F}_{\mathrm{adv}}$ in equation (2.3) is stiff in the low Mach number limit (chapter 14 in Wesseling, 2001). In that case, explicit time integration is very inefficient and inaccurate since it must resolve the fast sound waves leading to very small time steps and large rounding errors. Instead, implicit time integration methods should be used (e.g., Euler backward; Viallet et al., 2011; Kupka et al., 2012) and the sound waves can be removed analytically or numerically. Another possibility is preconditioning of the equations (Wesseling, 2001; Miczek, 2013) or analytical reformulation of the equations as in Kwatra et al. (2009) and Happenhofer et al. (2013).

Conservation of mass, energy and momentum are the key concepts of most applications of fluid dynamics. The numerical method must be able to conserve these quantities. For this reason, we prefer to use finite volume or conservative finite difference schemes for discretisation of the equations since then, the conservation properties are fulfilled automatically. Nevertheless, conservation can also get lost due to the analytical model (Lilly, 1996; Brown et al., 2012).

2.3 Code Validation and Verification

After all, choosing the analytical model as well as the numerical method introduces errors into the simulation. These errors should be measured and controlled separately as well as together.

Following the description in Ferziger and Perić (2002), the accuracy of numerical solutions is limited by three types of systematic errors. Firstly, there are *modelling errors* which arise since the mathematical equations do not describe the actual flow perfectly, or since the geometry of the problem cannot be reflected by the numerical

simulation domain. Secondly, due to the transformation of the continuous equations into discrete form we get *discretisation errors*. Thirdly, the differences between the discretised (algebraic) equations and the numerical solution caused by the error of the numerical iterative schemes used to solve the equations are called *iteration errors*.

Adapting the terminology from Calder et al. (2002), in the process of *code validation and verification* these errors are controlled and quantified. The discretisation and iteration errors are checked in the verification process. In the validation step, the compliance of the model with reality is tested. Informally speaking, we understand by *code verification* the process of checking that we are correctly solving the equations, as they were chosen in the modelling process. *Code validation* means to check whether we solve the appropriate equations (Calder et al., 2002). Figure 2.2 provides a schematic view of the overall process.

Of course, validation is much more complicated, especially in astrophysics. There are only very limited experimental data for any astrophysical application with which the numerical data can be compared. The observational data have, in most cases, very low resolution and can only be used to compare with integral quantities of the whole simulation.

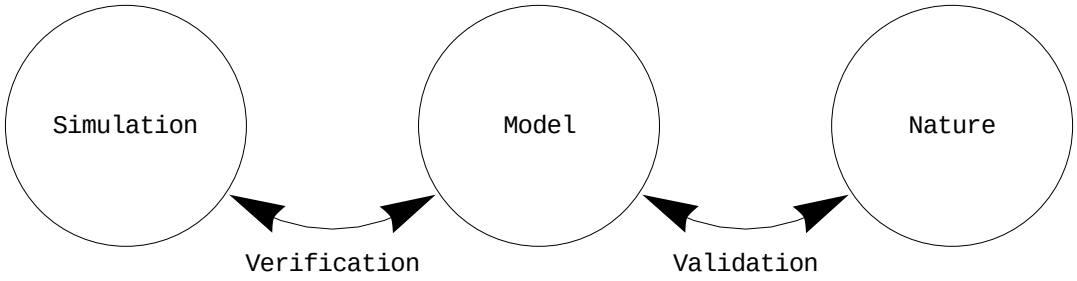

Figure 2.2: Code verification and validation. The process of checking the (mathematical) correctness of the numerical solution for the given set of partial differential equations (the model) is called verification. The process of checking if the model describes nature correctly is called validation.

In the following, we will introduce the main methods for code verification, and give some examples of code validation in the astrophysical context. We will demonstrate how the numerical method limits the validity of the simulation as well.

2.3.1 Verification Methods

In the code verification step we check whether there are errors in the numerical program solving the given set of differential equations. In some cases, we have analytical solutions of the equations we can compare. These and some more sophisticated methods are presented in the following paragraphs.

Test Problems

If the differential equations are given in a specific simplified form, we may be able to solve them analytically and then check the numerical solution by comparing with the analytical solution. Of course, this approach is limited to very simple and specific cases and it is not sufficient to prove the correctness of a code. In other cases, very accurate numerical solutions are known which were calculated with other simulation codes. A whole set of such test problems can be found, e.g., in Liska and Wendroff (2003).

Not every test problem is suited for any numerical code. If the code is designed, e.g., for low Mach number flows, it will probably fail for any test where the flow has a high Mach number. But this imposes no restrictions on the applicability of the code to the problems it was designed for.

Self-Convergence Studies

Testing self-convergence means increasing the spatial and temporal resolution of the simulation and test whether the numerical solution approaches a stable and unique solution. Following the description in Appendix A.6 in LeVeque (2007), we expect that the error ε of the numerical solution decreases with grid spacing h according to a power law of the form

$$\varepsilon(h) \approx Ch^p, \tag{2.11}$$

where p is the *(empirical) order of convergence* or *order of accuracy* and C is the *error constant* of the method. p and C, however, depend on the test problem as well. By comparing the numerical solution of several grid resolutions, the convergence rate p of a code for the given test problem can be calculated if the error can be determined by comparing with an analytical solution or a numerical reference solution. If, for example, the resolution is increased by a factor 2, the convergence rate p can be estimated by

$$p = \log_2 \left(\varepsilon(h)/\varepsilon(h/2)\right), \tag{2.12}$$

where $\varepsilon(h)$ is the numerical error at grid spacing h. Then, the error constant of the method can be calculated by

$$C = \varepsilon(h)/h^p. \tag{2.13}$$

Self-convergence studies show errors in the discretisation process. They do not prove that the numerical code solves the equations it was designed for, and in particular they do not demonstrate the physical relevance of the solution. Instead, they show that with decreasing grid spacing the code approaches some unique solution. Nevertheless, they are a valuable tool to measure the consistency of the code and the accuracy of the discretisation.

The order of accuracy and error constant may depend on the norm chosen for measurement of the error. For a grid function $f_i = f(x_i)$, where x_i are the discrete grid points

where the function value is defined, we define the following three norms (Appendix A.5 in LeVeque, 2007).

$$\|f\|_{L^1} = \sum_i h_i \left|f_i\right|, \tag{2.14}$$

$$\|f\|_{L^2} = \left(\sum_i h_i f_i^2\right)^{\frac{1}{2}}, \tag{2.15}$$

$$\|f\|_{L^\infty} = \max_i \left|f_i\right|, \tag{2.16}$$

where h_i is the width of the ith grid cell. The L^1 *norm* $\|f\|_{L^1}$ gives the arithmetic mean of the absolute function value in each grid point. The L^∞ *norm* $\|f\|_{L^\infty}$ finds the absolute maximum of the function on the whole domain. Finally, the L^2 *norm* $\|f\|_{L^2}$ defines an average error value where the extreme function values have higher weight. Depending on the application in mind, each of these norms can give valuable insights into the quality of the numerical solution.

Code–To–Code Comparisons

The most sophisticated method of code verification is comparison with other codes which should use different numerics and/or different analytical models for the same test problem. If the results are similar it is a strong indication that the codes are working properly. Having congruent data from several codes strengthens the trustworthiness of each of the codes.

Even if a numerical code is verified with the methods described above, there is still the possibility that the numerical solution inherits deficiencies from the chosen numerical method or even from the choice of the model equations. By comparing with different codes it can be checked whether the numerical solutions are method-independent, and if the analytical model behind the codes is different, these tests can already provide a certain validation of the results.

For the case of solar surface convection, extensive code comparisons were done by Kupka (2009a) and Beeck et al. (2012). They show a general agreement of the codes considered in the comparisons, even though the amplitude of the difference may depend on the variable you are looking at. E.g., the mean temperature stratification of all simulations is very similar whereas the standard deviation shows larger differences as depicted in Figure 2.3. Kupka (2009a) demonstrated that the higher moments of the temperature distribution exhibit increased sensitivity on the numerical method and the boundary condition.

2.3.2 Examples of Code Validation

Validation is the second step in proving the physical relevance of the simulation results, and usually it is the much harder one. There is no standard procedure how one can

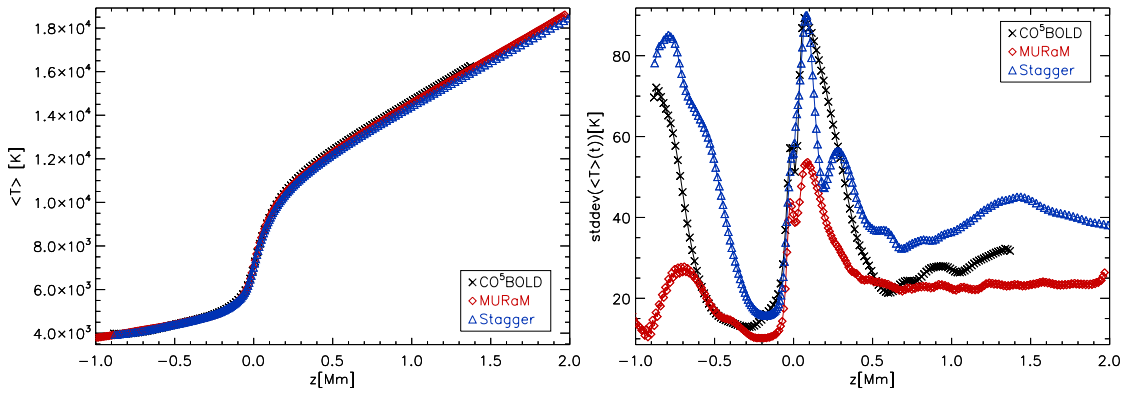

Figure 2.3: Mean temperature stratification and standard deviation of the temperature distribution for three simulations of solar surface granulation. Figure taken from Beeck et al. (2012).

show that the output of the simulation code actually is a good model for the application one is interested in.

In the wide field of computational fluid dynamics, the most effective and easiest way of validation is comparison of the numerical data with experiments or direct measurements of the actual flow. Another possibility is performing simulations where all scales are resolved and no simplifications of the basic analytical model are used. Even though these experiments and numerical simulations are very demanding in terms of human and computer time as well as resources, they can be performed for several test cases, and the numerical code can be benchmarked and validated using this data.

The parameter range of nearly all astrophysical applications covers a regime which cannot be reached in terrestrial conditions (e.g., Kupka, 2009b). All fluid experiments which can be performed in a laboratory differ by several orders of magnitude in length and time scales as well as viscosity. Similarly, it is not affordable to perform simulations where all scales are resolved. Therefore, the two main possibilities of validating a simulation code are not available in astrophysics.

In the next paragraphs, we will give examples of code validation for several astrophysical applications. Thereby, we will emphasize the limitations to code validity due to the numerical method.

Stellar Surface Convection

In astrophysics, most observational data have a very low resolution such that it cannot be used to gain detailed information about the dynamics of the plasma on the stellar surface or even inside of the star. The only case where truely high-resolution observations exist is the Sun and that is why it has become an important benchmark for any stellar simulation code.

On the surface of the quiet Sun, we observe a granular pattern which is the manifestation of the turbulent convection in the outer regions of the Sun, as shown in Figure 2.1

(e.g., Nordlund et al., 2009; Kupka, 2009b). By choosing a small simulation box of only several Mm in size around the optical surface of the Sun, Nordlund (1982) started to perform detailed and accurate simulations of the convection on the solar surface. Since then, several other groups have developed their own simulation codes for stellar surface convection (e.g. Freytag et al., 2012; Vögler et al., 2005; Muthsam et al., 2007, 2010a).

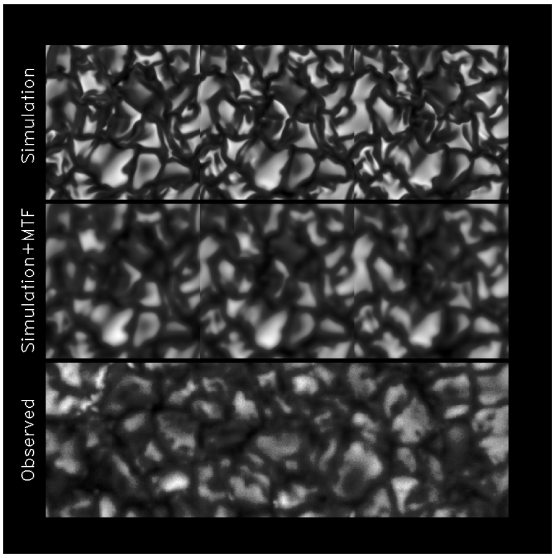

Figure 2.4: Simulated (top) and observed (bottom) granulation patterns at the surface of the Sun. The middle picture shows artificially blurred simulation data. Due to the high resolution of the simulation, the simulated picture is actually sharper than the observed one. Figure taken from Stein and Nordlund (1998).

These simulations have proven to be very accurate. The first test is direct comparison with the observational data. As shown in Figure 2.4, the simulated and observed snapshots cannot be distinguished except for the even higher resolution of the simulation. At the same time, this test is already the starting point where we can learn something from the simulation: it is impossible to get as accurate and highly resolved data from the observations as we get from the simulation. The simulations deepen our understanding of the convective motions on the surface of the Sun and even in the upper part of the solar convection zone. See Lemmerer et al. (2013) for details.

In Asplund et al. (2000a) and Asplund et al. (2000b), the observed spectral line formation of Fe lines is compared to the synthetic one calculated from the three-dimensional simulation data. Its line shapes, shifts and asymmetries are very similar, showing that the simulation produces realistic results also for these derived quantities. Pereira et al. (2013) showed that the predictions from the three-dimensional models for the continuum centre–to–limb variations are closer to the observations than any one-dimensional model which has been used in the past.

Another method of validation is comparison with models and observation from aster-

oseismology. Georgobiani et al. (2000) showed that the numerical simulation reproduced the basic observed properties of solar oscillations.

We conclude that the simulation of solar surface convection is very well tested and validated. After the validation process has been successful, one can start to gain information from the simulation where no observational data is available. The surface of many main-sequence stars is assumed to behave similarly as the Sun. But, due to their larger distance to the earth, we do not possess as detailed and accurate observational data as for the Sun. But one can use the simulation codes which have been verified by the solar benchmark to predict the dynamic properties of other main-sequence stars provided they behave similarly to the Sun. Trampedach et al. (2013) and Magic et al. (2013) have started to calculate grids of stars distributed over a wide range of stellar parameters, spending a huge amount of computation time and producing terabytes of promising output data.

Nevertheless, we must be aware of the limitations of this approach. The assumption that we can model the granulation in Cartesian boxes which are small in comparison to the stellar radius implies that the integral length scale of the problem is small. The size of a granule scales roughly with the inverse of the gravity acceleration g (Robinson et al., 2004). More precisely, Freytag et al. (2002) used the empirical relation

$$x_{\mathrm{gran}} \approx 10\, r_{\mathrm{gas}} \frac{T_{\mathrm{eff}}}{g} \tag{2.17}$$

where x_{gran} is the typical size of a granule and r_{gas} is the universal gas constant. In terms of the solar radius $R_\odot$, mass $M_\odot$ and effective temperature $T_{\mathrm{eff},\odot}$, this can be rewritten as

$$\frac{x_{\mathrm{gran}}}{R_\star} \approx 0.0025\, \frac{R_\star}{R_\odot} \frac{T_{\mathrm{eff},\star}}{T_{\mathrm{eff},\odot}} \frac{M_\odot}{M_\star}. \tag{2.18}$$

Therefore, the assumption that the granule size is small compared to the stellar radius is not true any more when the mass of the star is small or the radius and the effective temperature are high compared to the solar values.

Typically, the simulation box is placed near the stellar surface such that the top boundary is in the upper photosphere, whereas the bottom boundary is situated deeply inside the convective envelope. There is no obvious choice for the boundary conditions. Ideally, they should allow free in– and outflow and not influence the solution in the inner regions of the simulation box. The design of such boundary conditions is not trivial (Grimm-Strele et al., 2013a).

Star In A Box

Substituting the stellar parameters of the red supergiant Betelgeuse into formula (2.17), Freytag et al. (2002) suggested that there are only a few hundred convection cells on the whole surface of the star. In this case, the local approach — approximating a small representative piece of the star near the surface by a Cartesian box — will not work. Instead, the simulation domain must include the whole star.

A direct way to apply the Cartesian codes designed for surface convection simulations to these stars is the *star–in–a–box* approach (Freytag et al., 2002). The Cartesian simulation domain is chosen large enough to contain the whole star.

The numerical difficulties connected to this approach are obvious and manifold. By fitting a sphere inside a cube, about 50 % of the computational cells are located outside of the star and therefore do not provide any information. This makes parallelisation by domain decomposition much more wasteful and limits the efficiency of the code. Furthermore, the resolution is uniform everywhere even though we would like to have higher resolution near the optical surface of the star. Imposing boundary conditions is very difficult as every approximation of a spherical shell by Cartesian grid cells is rather crude.

Nevertheless, this approach has the advantage of easy implementation. All efficient algorithms designed for Cartesian grids can be used. Chiavassa et al. (2009) showed that the results obtained by these simulations fit existing interferometric observations reasonably well, and in the absence of better alternatives, they provide a valuable tool for numerical astrophysicists.

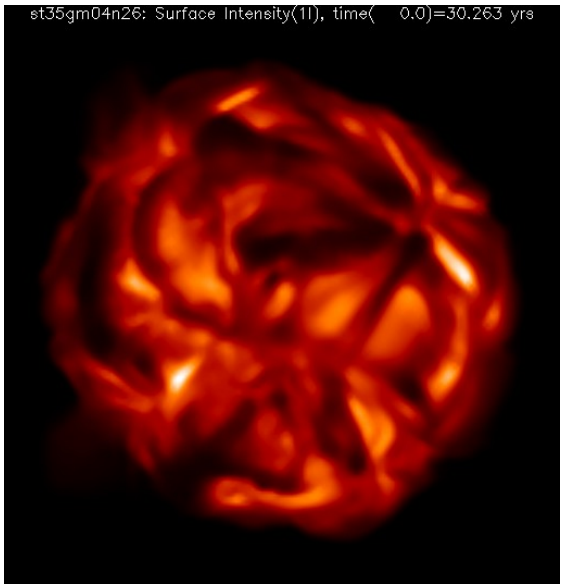

Figure 2.5: Emergent surface intensity from a numerical simulation of Betelgeuse. Picture taken from `http://www.astro.uu.se/~bf/`.

Core and Envelope Convection with ASH

Deeper in the convection zone of the Sun, the convective motions act on much larger scales such that they cannot be captured in small Cartesian boxes. Instead, Glatzmaier (1984) developed the code ASH which solves the three-dimensional anelastic magnetohydrodynamic equations in a spherical coordinate system. In the horizontal directions,

all variables are expanded in spherical harmonics, whereas in the vertical direction, Chebychev polynomials are used.

The spherical coordinate system defined by the transformation

$$\begin{aligned} x &= r\sin\theta\cos\phi, \\ y &= r\sin\theta\sin\phi, \qquad \text{with } \phi \in [0, 2\pi], \theta \in [0, \pi], \\ z &= r\cos\theta, \end{aligned} \tag{2.19}$$

has several obvious advantages compared to the star–in–a–box approach. It is the natural choice for the simulation of spherical shells and cones. The radial grid layering can be adapted to the structure of the star, and resolution can be increased near the tachocline (the transition from the radiative interior to the convective outer envelope) or the spherical surface. With the anelastic approximation, the sound waves are filtered out explicitly, and the numerical time step can be large enough to cover the dynamical time scales of core and envelope convection in main sequence stars.

The code was applied to a variety of astrophysical problems, including, e.g., core convection of an A–type star (Browning et al., 2004), the transition between the radiative and the convective zone of the Sun (Browning et al., 2007) and super-granulation (DeRosa et al., 2002). Toomre et al. (2012) give a comprehensive overview of the astrophysical problems tackled with ASH and how these simulations match observations from asteroseismology.

Despite its success in applications, the numerical approach of ASH has severe drawbacks. The anelastic approximation is qualitatively wrong near the surface of the star. Therefore, the outer 2 to 5 % percent of the star are usually excluded from their simulations, even though the turbulent convection at the surface is assumed to drive the convection in the convective envelope (e.g., Kupka, 2009b). This influence must be modelled by an artificial boundary condition.

We explain the basic idea of spectral methods for the case of Fourier expansions. The set of functions

$$\phi_k(x) = \mathrm{e}^{\mathrm{i}kx} \tag{2.20}$$

is an orthonormal system on $[0, 2\pi]$. i is the imaginary unit. For a given function f, its *Fourier coefficients* $\hat{f}_k$ are given by

$$\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)\mathrm{e}^{-\mathrm{i}kx}\,dx. \tag{2.21}$$

The *Fourier expansion* of f is defined by the infinite series

$$\mathcal{F}f = \sum_{k=-\infty}^{\infty} \hat{f}_k\phi_k \tag{2.22}$$

(Canuto et al., 1988). Other expansions can be obtained by choosing a different orthonor-

mal set of expansion functions. When this approximation is implemented numerically, two types of errors arise: *truncation errors* when only a finite number of terms are considered in the expansion (2.22). Then, the error arising due to approximate calculation of the coefficients (2.21) is called *aliasing error*.

There are "de-aliasing" methods to remove the aliasing error from the numerical solution. But, since the aliasing and truncation errors decay at the same rate, the accuracy of the aliased solution will be well enough as soon as the resolution is high (Canuto et al., 1988). It is doubtable that any simulation in an astrophysical context can reach sufficient high resolution, and the effect on the accuracy of the numerical solution is unclear.

The convergence speed of spectral methods is, in principle, superior to any finite difference or finite volume method (Canuto et al., 1988). But there are mainly two restrictions to this theoretical advantage. Firstly, coupling the spectral method with the Crank–Nicolson or the second order Adams–Bashforth time integration scheme scheme as done in Glatzmaier (1984) leads to dominance of temporal errors when the resolution is increased, and the overall order is restricted again. Secondly, if discontinuities are present in the numerical solution, the Gibbs phenomenon, an oscillatory overshoot of — in the limit of vanishing grid spacing — finite amplitude, leads to further reduction of the convergence order (p. 45, Canuto et al., 1988). Table 2.4-1. in Fornberg (1998) shows the order of the errors in the maximum norm caused by irregularities of a function. Variable and non-smooth coefficient functions are problematic for spectral methods, expecially when the spherical harmonics expansion is used (Fornberg, 1998).

Finally, we note that it is a challenging task to efficiently implement the Legendre transform on parallel computers (Clune et al., 1999). The transform, which is needed due to the spherical harmonics expansion, dominates the total computation time of ASH and any other code employing spherical harmonics expansions (Cai et al., 2011).

The spherical coordinate system itself has several disadvantages. At $r = 0$ and $\theta = 0, \pi$, there are grid singularities. The singularity in θ is taken care of when using spherical harmonics expansions, but for a finite difference or finite volume method, grid cells get very small and impose strong restrictions on the time step if explicit time integration methods are used. At the inner core, an artificial boundary must be introduced. This might not be problematic for simulations of the outer convective envelope, but it certainly changes the velocity field if a flow through the centre of the grid exists. This probably is the case for the simulations of core convection as in Browning et al. (2004). By using a two-dimensional Cartesian code, Evonuk and Glatzmaier (2007) demonstrated that, depending on the rotation rate, the velocity field of a fully convective planet is changed completely by the introduction of a small artificial core, no matter how small the core is. Figure 2.6 taken from Evonuk and Glatzmaier (2007) illustrates this behaviour.

2.4 The Four Main Restrictions of Validity

In the previous section we demonstrated that the numerical method limits the validity and applicability of an astrophysical simulation code. We have identified the four main

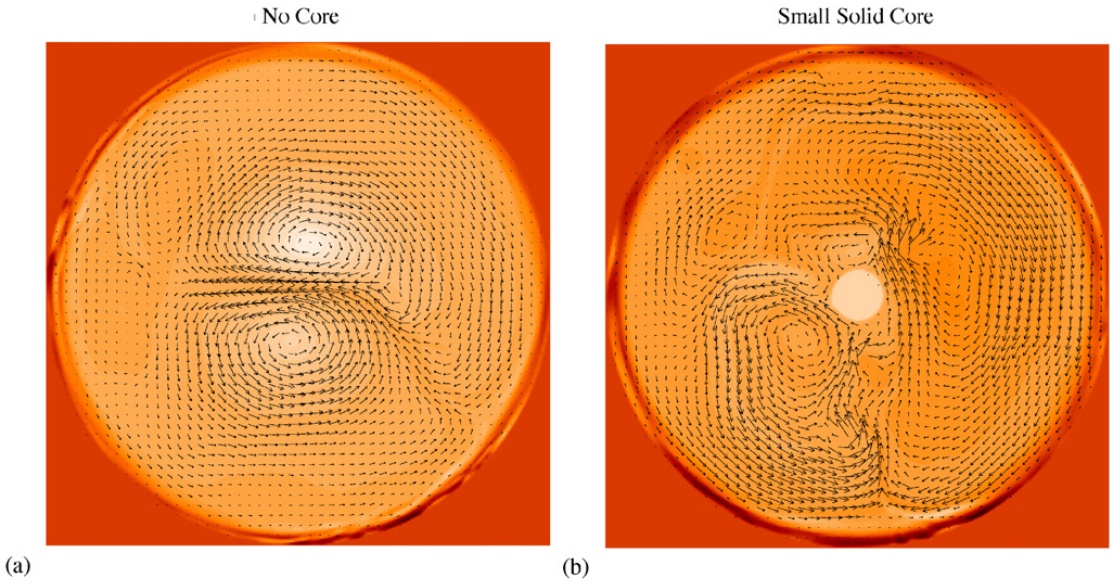

Figure 2.6: Flow pattern in the interior of a giant gaseous planet with and without artificial inner core. Figure taken from Evonuk and Glatzmaier (2007).

restrictions as the limitations due to the numerical grid, the problem of resolving a wide variety of time scales, the problem of resolving the parameter space and viscosity, and the design of appropriate boundary conditions. We will describe each of these points in detail.

2.4.1 Grid Geometry

The geometry of many astrophysical computational problems is rather simple. Either the domain can be approximated by a Cartesian box, or it has spherical shape. However, for certain applications as, e.g., core convection, the classical Cartesian and spherical coordinate systems are not sufficient.

In other cases, a non-uniform grid resolution throughout the domain would be desirable. In surface convection simulations, the flow inside the granules is expected to be rather laminar. Only in the fast downdrafts between the granules, turbulence is expected to develop (Kupka, 2009b). There, it would be desirable to have a higher resolution to better resolve the turbulent motions. The technique of grid refinement as described in Muthsam et al. (2010a) is a valuable tool to get to very high resolutions without making the computational requirements unbearable.

When going deeper into the convection zone, spatial scales increase such that the resolution requirements decrease compared to the surface. A non-equidistant grid in the radial direction as used in Magic et al. (2013) and Glatzmaier (1984) helps lowering the computational requirements of the simulation.

The choice of the numerical grid limits the applicability of a numerical code considerably. Many numerical schemes are only usable for equidistant Cartesian or other simple coordinate systems (e.g. WENO as described in Merriman, 2003). Generalisations to unstructured grids are very complicated and expensive both in terms of programming work and computation time (p. 36 in Ferziger and Perić, 2002). Therefore, as long as it is possible to use the simple and efficient high-order schemes for Cartesian grids, one should continue to use them. We summarise the advantages and disadvantages of the spectral approach from Glatzmaier (1984), the WENO approach for Cartesian domains from Muthsam et al. (2010a) and for spherical domains from Mundprecht et al. (2013) in Table 2.2.

The topic of the PhD thesis will be to extend the grid capabilites of two existing codes working in Cartesian coordinates, keeping the changes in code and numerical method as small as possible.

2.4.2 Time Scales

In the right panel of Figure 2.7, the mean Mach number $\mathrm{Ma} = \frac{|\mathbf{u}|}{v_{\mathrm{snd}}}$ of the convective flow is shown (dashed line). It denotes the ratio of the convective velocity to the speed of the sound waves. Near the surface, Ma is around unity, but going deeper in the convection zone, it decreases by several orders of magnitude.

As described in Kupka (2009b), explicit time integration schemes use the information at the old time step to calculate an extrapolation of the new time step. More precisely, if we denote the differential operator by L and the solution at time step n by u^n,

$$u^{n+1} = u^n + \delta t \, \mathrm{L}\left(u^n\right). \tag{2.23}$$

The time step size δt is limited by the Courant–Friedrichs–Levy (CFL) condition (e.g., Strikwerda, 1989), and is constrained by the local speed of sound. Therefore, explicit time integration schemes are very efficient near the surface, but allow only very small time steps in the deeper convection zone where the Mach number is low.

Famous examples of explicit time integration schemes in the context of Runge–Kutta methods are the Euler forward scheme (Strikwerda, 1989) and the TVD2 and TVD3 schemes from Jiang and Shu (1996). In this thesis, we also use the SSP RK(3,2) scheme from Kraaijevanger (1991) and Kupka et al. (2012). These methods differ, amongst others, in number of stages, order of accuracy and error constants.

On the contrary, implicit time integration schemes require the solution of a (non)linear system to get the solution at the new time level, i.e.

$$u^{n+1} = u^n + \delta t \, \mathrm{L}\left(u^{n+1}\right). \tag{2.24}$$

Therefore, one step with an implicit time integration scheme is much more expensive than with an explicit one. The use of implicit schemes pays off when the time step can be chosen much larger. On the other hand, large time steps may decrease the accuracy of the solution (Kupka, 2009b).

Guillard and Viozat (1999) and Guillard and Murrone (2004) demonstrated that the accuracy of certain upwind schemes is very bad in low Mach number flows. This was confirmed in Happenhofer et al. (2013) for the WENO5 scheme with explicit time integration. For low Mach numbers, the numerical scheme must be adapted suitably both for reasons of efficiency and accuracy. The disadvantage of analytical approaches like the anelastic approximation is that their basic assumptions limit their domain of applicability to specific regions in the star, and the use of the approaches in other regimes will yield wrong results (Brown et al., 2012).

Solving the Navier–Stokes equations without any simplifications with implicit time integration schemes is possible in theory, but numerically very complicated and hard to implement efficiently (Miczek, 2013; Viallet et al., 2011). Happenhofer et al. (2013) suggested the use of Kwatra's method (Kwatra et al., 2009) instead. There, the Navier–Stokes equations are reformulated with very tiny simplifications to yield an elliptic equation for a pressure prediction. Using this prediction of the pressure in a modified update step increases the stability and accuracy of the method in the low Mach number limit considerably without destroying its ability to handle shocks and high Mach number flows (Happenhofer et al., 2013).

Besides the time scales of advection and sound waves there exist a variety of other time scales describing various physical processes. Thermal relaxation, e.g., acts on the Kelvin–Helmholtz time scale. For a numerical simulation this is the time it takes for the whole energy of the box to be transported away by radiation only. For solar granulation simulations, this time scale is approximately several hundred hours (Grimm-Strele et al., 2013a) and therefore out of scope for any multidimensional simulation — even though its importance for the relaxation and rearrangement of a model is huge.

If other terms like, e.g., heat diffusion or viscosity, impose a rigid limit to the time step, implicit–explicit (IMEX) methods may be a efficient alternative. The basic idea is to split the differential operator in two parts,

$$\mathrm{L}\left(u(t)\right) = \mathrm{L}_{\mathrm{exp}}\left(u(t)\right) + \mathrm{L}_{\mathrm{imp}}\left(u(t)\right), \tag{2.25}$$

and integrate the one part $\mathrm{L}_{\mathrm{exp}}$ with an explicit scheme and the other $\mathrm{L}_{\mathrm{imp}}$ with an implicit one (Kupka et al., 2012). The terms in the equations corresponding to the stiff processes are put in the implicit part $\mathrm{L}_{\mathrm{imp}}$. Nevertheless, the integration of both parts is simpler than performing a fully implicit integration step of the differential operator.

Due to the large computational requirements of these simulations, interesting results can only be obtained by means of parallelisation, in particular domain decomposition (e.g. Obertscheider, 2007). Whereas this is rather trivial for explicit schemes, good scaling is very difficult to obtain for implicit time integration schemes. Therefore, Hotta et al. (2012) modified the differential equations in such a way that their explicit schemes can be used efficiently throughout the solar convection zone. They sacrificed the essential property of mass conservation for that. However, Grimm-Strele (2010) describes a solution algorithm based on the Schur Complement method (Saad, 2003) which has been proven to scale strongly up to several thousand MPI processes (Happenhofer et al., 2013).

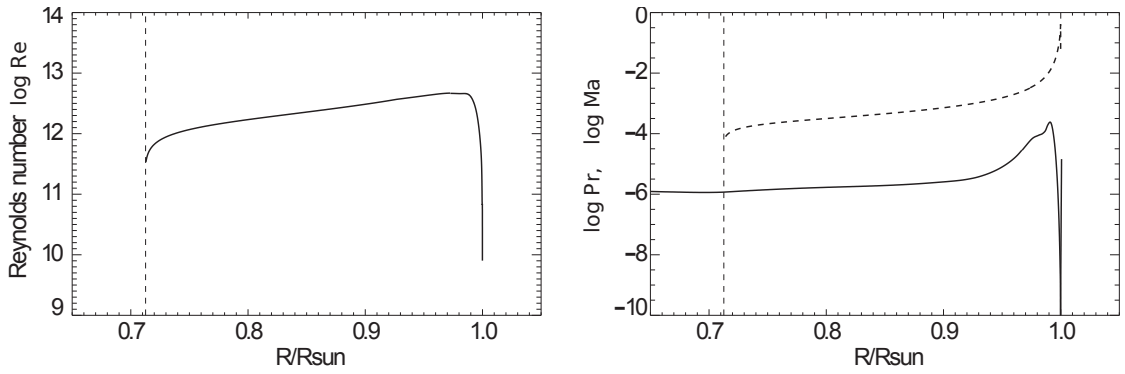

Figure 2.7: Reynolds number (left), Prandtl and Mach number (right, solid and dashed lines, respectively) throughout the solar convection zone. Figure taken from Freytag et al. (2012).

2.4.3 Parameter Space

The Reynolds number Re is a measure of the magnitude of the viscous forces compared to the advective ones. High Reynolds numbers imply small viscosity. Freytag et al. (2012) evaluated Re for the case of solar convection by

$$\mathrm{Re} = \frac{v_c H_p}{\nu}, \tag{2.26}$$

with the convective velocity v_c, the local pressure scale height H_p and the microscopic kinematic viscosity ν evaluated from one-dimensional stellar evolution models. Figure 2.7 shows that the Reynolds number is of the order 10^{10} to 10^{13} throughout the convection zone. Similarly, the Prandtl number Pr, defined as the ratio of coefficients of momentum and heat transfer, was found to be of order 10^{-6} and 10^{-10} near the surface of the Sun.

But instead of the realistic values for Re and Pr, the concept of large eddy simulations implies that the *effective viscosity* of a numerical simulation is much higher, and therefore the *effective* values of Reynolds and Prandtl number are much lower resp. higher than the realistic ones (e.g. Kupka, 2009b; Freytag et al., 2012). The spatial resolution needed to reach realistic effective values of these parameters is not feasible with today's computers. This limits the physical realism of the simulations in an unpredictable way. Since this is not the topic of this thesis, we will not go into more detail, but refer the reader to Kupka (2009b) and Freytag et al. (2012), instead.

2.4.4 Boundary Conditions

Finally, in most astrophysical applications there is no obvious way how the boundary conditions should be set. Grimm-Strele et al. (2013a) describe the problem for the case of stellar surface convection in much detail. As already mentioned in paragraph 2.3.2, the domain of surface convection simulations typically embraces a small Cartesian box at the surface of the star. The top boundary is located in the upper photosphere, whereas

the lower boundary is somewhere in the convection zone. Although it is obvious that it is desirable to put as much distance between the boundaries and the surface, the exact vertical position of both boundaries is dictated by the computational resources available. In any case, the boundary conditions should be designed such that they disturb the flow as little as possible which is not a trivial task. Again, this topic will not be covered in this thesis. An extensive discussion can be found in Grimm-Strele et al. (2013a).

	Spectral Approach Glatzmaier (1984)		Cartesian WENO Muthsam et al. (2010a)		Spherical WENO Mundprecht et al. (2013)	
order of accuracy	++	theoretically high, but limited by solution and time integration accuracy	+	usually 5, but limited by time integration accuracy	−	limited to 2
pole problem in latitude	+	avoided due to spherical harmonics	+	not present	−	solvable with Yin–Yang grid
core singularity	−	not solvable	+	not present	−	not solvable
smoothness requirements	−	high, unpredictable behaviour for non-smooth flow	+	none, shock-capturing	+	none, shock-capturing
effect of low resolution	−	aliasing errors make results untrustworthy	+	LES: smaller scales are modelled by subgrid scale model	+	LES: smaller scales are modelled by subgrid scale model
conservation property	−	not conservative	+	fully conservative except for gravity	−	not conservative due to geometry terms
computational cost	−	Legendre transform	+−	intermediate	+−	intermediate
outer boundary	+	spherical	−	Cartesian box	+	spherical
parallelisation	−	difficult	+	straightforward	+	straightforward

Table 2.2: Comparison of advantages and disadvantages of standard spectral and WENO approaches for the numerical simulation of the full sphere.

3 Numerical Methods

In this chapter, we will describe the numerical methods used in this thesis and their important properties concerning the structure of the grid.

As a side topic, we will investigate the efficiency of several explicit Runge–Kutta schemes combined with the fifth order WENO scheme as described in Section 3.1 for some idealised cases and the case of solar convection.

3.1 The Weighted Essentially Non Oscillatory (WENO) Scheme

In this section, we consider the Euler equations without any viscous, gravity or diffusion terms. The Euler equations are a system of partial differential equations. In three spatial dimensions and in a Cartesian coordinate system, their differential form is

$$\frac{\partial}{\partial t}\mathbf{Q} + \frac{\partial}{\partial x}\mathbf{F} + \frac{\partial}{\partial y}\mathbf{G} + \frac{\partial}{\partial z}\mathbf{H} = 0, \tag{3.1}$$

with the state vector $\mathbf{Q}$ and the flux functions $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ given by

$$\begin{aligned} \mathbf{Q} &= \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{pmatrix}, \mathbf{F}(\mathbf{Q}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (p+E)u \end{pmatrix}, \\ \mathbf{G}(\mathbf{Q}) &= \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ (p+E)v \end{pmatrix}, \mathbf{H}(\mathbf{Q}) = \begin{pmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ (p+E)w \end{pmatrix}, \end{aligned} \tag{3.2}$$

where the pressure $p = p(\rho, e)$ is given by an equation of state and $e = E - \frac{u^2+v^2+w^2}{2\rho}$ is the internal energy. In the following, we will write $\mathbf{u} := (u, v, w)^{\mathrm{T}}$ for the velocity vector.

3.1.1 Finite difference and finite volume discretisation

First, we derive the finite difference and finite volume discretisation of the Euler equations for the one-dimensional case. Thereby, we follow the *method of lines* approach

of discretising space and time derivatives separately (Toro, 2009; LeVeque, 2007). The differential form of the Euler equations in one spatial dimension is

$$\frac{\partial}{\partial t}\mathbf{Q} + \frac{\partial}{\partial x}\mathbf{F} = 0, \tag{3.3}$$

$$\mathbf{Q} = \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}, \mathbf{F}(\mathbf{Q}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (p+E)u \end{pmatrix}, \tag{3.4}$$

where $p = p(\rho, e)$ and the velocity u. Let $\Omega = [a, b]$ and let a grid on Ω be defined on the half-integer nodes, i.e.

$$x_{i+\frac{1}{2}} = x_{i-\frac{1}{2}} + \delta x_i,\ i = 1, \ldots, n,\ x_{\frac{1}{2}} = a, x_{n+\frac{1}{2}} = b. \tag{3.5}$$

The grid is not necessarily equidistant. The index i refers to the centre of the cell $C_i = \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$.

Definition 1. *The Cell Average Operator* A *on a grid* (3.5) *is defined by*

$$\mathrm{A}(h)_i := \frac{1}{\delta x_i} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} h(\tilde{x}) d\tilde{x}. \tag{3.6}$$

Applying A to (3.3) gives

$$\frac{\partial}{\partial t}(\mathrm{A}\mathbf{Q})_i + \frac{\delta \mathbf{F}_i}{\delta x_i} = 0 \text{ with } \frac{\delta \mathbf{F}_i}{\delta x_i} = \frac{\mathbf{F}_{i+\frac{1}{2}} - \mathbf{F}_{i-\frac{1}{2}}}{\delta x_i} \tag{3.7}$$

with $\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}\left(\mathbf{Q}(x_{i+\frac{1}{2}})\right)$. $(\mathrm{A}\mathbf{Q})_i =: \overline{\mathbf{Q}}_i$ is the *Cell Average* of $\mathbf{Q}$ in the cell $C_i = \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$.

Applying now A^{-1} as in Merriman (2003), we get

$$\frac{\partial}{\partial t}\mathbf{Q}_i + \mathrm{A}^{-1}\frac{\delta \mathbf{F}_i}{\delta x_i} = 0. \tag{3.8}$$

Merriman (2003) showed that

$$\mathrm{A}^{-1}\frac{\delta \mathbf{F}_i}{\delta x_i} = \frac{\delta\left(\mathrm{A}^{-1}\mathbf{F}_i\right)}{\delta x_i} \text{ if the grid is equidistant, i.e. } \delta x_i = \delta x\ \forall i. \tag{3.9}$$

In this case, (3.3) is equivalent to

$$\frac{\partial}{\partial t}\mathbf{Q}_i + \frac{\delta \mathbf{f}_i}{\delta x_i} = 0 \text{ with } \mathbf{F}_i = (\mathrm{A}\mathbf{f})_i. \tag{3.10}$$

(3.10) is called the **Shu–Osher form** of (3.3). It is equivalent to (3.1) if the grid is equidistant (Merriman, 2003).

A finite volume scheme starts with equation (3.7) and computes approximations for $\mathbf{Q}_{i+\frac{1}{2}}$ from the given cell averaged $\overline{\mathbf{Q}}_i$. The numerical flux $\hat{\mathbf{F}}_{i+\frac{1}{2}}$ is calculated by

$$\hat{\mathbf{F}}_{i+\frac{1}{2}} = \mathbf{F}\left(\mathbf{Q}_{i+\frac{1}{2}}\right). \tag{3.11}$$

In contrast, the finite difference scheme starts with equation (3.10) and computes approximations for $\mathbf{f}_{i+\frac{1}{2}}$ from the values of the analytical flux function $\mathbf{F}$. Since $\mathbf{F}_i = (\mathrm{A}\mathbf{f})_i$, both approaches are computationally equivalent and require the numerical solution of the following

Problem 1 (Reconstruction problem)**.** *Given a set of cell averages, compute the value of the underlying function at the half-integer nodes.*

Definition 2. *The Reconstruction Operator* R *acts on cell averages and reconstructs the point value of the underlying function*

$$\mathrm{R}(\mathrm{A}h)_{i+\frac{1}{2}} = h_{i+\frac{1}{2}} \tag{3.12}$$

on a given grid (3.5)*.*

If the grid is equidistant, the same algorithm can be used to compute $\mathbf{Q}_{i+\frac{1}{2}}$ and $\mathbf{f}_{i+\frac{1}{2}}$. The only difference lies in the input for the reconstruction process: in the case of a finite volume scheme, the inputs are given by the cell averages $\overline{\mathbf{Q}}_i$ of the state vector, whereas in the case of a finite difference scheme, they are given by the analytical flux function $\mathbf{F}$ evaluated at the cell centre.

Given a reconstruction operator R, Problem 1 can be solved. The WENO reconstruction operator is described in detail in Shu (2003) and also in 3.1.2, following the cited reference. In general, finite volume methods can be designed on non-equidistant grids. However, since the WENO reconstruction operator as it is described in 3.1.2 works only on equidistant grids, the applicability of the finite volume method using this reconstruction method is restricted to equidistant grids, too.

The advantage of the finite difference scheme lies in its easy extension to higher dimensions. Let a region (i.e., a non-empty, open, and connected set) $\Omega \subset \mathbb{R}^3$ be given. We examine the Euler equations (3.1) on Ω.

In the finite volume approach, applying $\mathrm{A}_x\mathrm{A}_y$ to the three-dimensional Euler equations (3.1) results in

$$\begin{aligned} \frac{\partial}{\partial t}\left(\mathrm{A}_{x,y,z}\mathbf{Q}\right)_{i,j,k} + \frac{\mathrm{A}_{y,z}(\mathbf{F}_{i+\frac{1}{2}})_{j,k} - \mathrm{A}_{y,z}(\mathbf{F}_{i-\frac{1}{2}})_{j,k}}{\delta x} \\ + \frac{\mathrm{A}_{x,z}(\mathbf{G}_{j+\frac{1}{2}})_{i,k} - \mathrm{A}_{x,z}(\mathbf{G}_{j-\frac{1}{2}})_{i,k}}{\delta y} + \frac{\mathrm{A}_{x,y}(\mathbf{H}_{k+\frac{1}{2}})_{i,j} - \mathrm{A}_{x,y}(\mathbf{H}_{k-\frac{1}{2}})_{i,j}}{\delta z} = 0. \end{aligned} \tag{3.13}$$

We wrote $\mathrm{A}_{x,y}$ for $\mathrm{A}_x\mathrm{A}_y$ here. The fluxes are now surface integrals over the cell boundaries. With the midpoint rule,

$$\mathrm{A}_{y,z}(\mathbf{F}_{i+\frac{1}{2}})_{j,k} = \mathbf{F}_{i+\frac{1}{2},j,k} + O(h^2). \tag{3.14}$$

With only one evaluation of the fluxes, the overall order of the method is restricted to two. We notice that if $\mathbf{F}$ is linear, this integration is exact.

Contrary, in the finite difference approach, (3.1) is transformed to

$$\frac{\partial}{\partial t}\mathbf{Q}_{i,j,k} + \frac{\mathbf{f}_{i+\frac{1}{2},j,k} - \mathbf{f}_{i-\frac{1}{2},j,k}}{\delta x} + \frac{\mathbf{g}_{i,j+\frac{1}{2},k} - \mathbf{g}_{i,j-\frac{1}{2},k}}{\delta y} + \frac{\mathbf{h}_{i,j,k+\frac{1}{2}} - \mathbf{h}_{i,j,k-\frac{1}{2}}}{\delta z} = 0 \tag{3.15}$$

with $\mathbf{F} = \mathrm{A}_x(\mathbf{f})$, $\mathbf{G} = \mathrm{A}_y(\mathbf{g})$ and $\mathbf{H} = \mathrm{A}_z(\mathbf{h})$. The overall order of the method only depends on the order of the reconstruction of $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$, which are one-dimensional reconstruction problems. The one-dimensional algorithms can be directly applied without loss of order of accuracy.

Even if the high order one-dimensional WENO reconstruction algorithm is applied to evaluate the flux functions in (3.13), the resulting finite volume method is only second order accurate. To obtain a truly high-order multidimensional finite volume method, more complicated integrations for the line integrals in (3.13) must be performed, increasing the computational costs of the finite volume scheme tremendously (Shu, 2003; Colella et al., 2011; Zhang et al., 2011). Therefore, finite difference schemes are clearly preferable on Cartesian grids.

The procedure is described in pseudocode in Algorithms 1 and 2.

Algorithm 1 Finite difference scheme for the three-dimensional Euler equations.

1: $\mathbf{Q}_{i,j,k}$ is given as point value at the cell centre.
2: $\mathrm{A}(\mathbf{f})_{i,j,k} = \mathbf{F}_{i,j,k}$, $\mathrm{A}(\mathbf{g})_{i,j,k} = \mathbf{G}_{i,j,k}$, $\mathrm{A}(\mathbf{h})_{i,j,k} = \mathbf{H}_{i,j,k}$
3: $\mathbf{f}_{i\pm\frac{1}{2},j,k} = \mathrm{R}_x\left(\mathbf{F}_{i,j,k}\right)$, $\mathbf{g}_{i,j\pm\frac{1}{2},k} = \mathrm{R}_y\left(\mathbf{G}_{i,j,k}\right)$, $\mathbf{h}_{i,j,k\pm\frac{1}{2}} = \mathrm{R}_z\left(\mathbf{H}_{i,j,k}\right)$
4:

$$\begin{aligned}\frac{\partial \mathbf{Q}_{i,j,k}}{\partial t} = &-\frac{1}{\delta x}\left(\mathbf{f}_{i+\frac{1}{2},j,k} - \mathbf{f}_{i-\frac{1}{2},j,k}\right) - \frac{1}{\delta y}\left(\mathbf{g}_{i,j+\frac{1}{2},k} - \mathbf{g}_{i,j-\frac{1}{2},k}\right)\\ &-\frac{1}{\delta z}\left(\mathbf{h}_{i,j,k+\frac{1}{2}} - \mathbf{h}_{i,j,k-\frac{1}{2}}\right)\end{aligned}$$

Both algorithms need the specification of a reconstruction operator. A reconstruction operator calculates the point value of a function given its cell averages. The WENO reconstruction operator is described, e.g., in Shu (2003) and 3.1.2.

3.1.2 Reconstruction Algorithm

In the following, the WENO reconstruction operator is derived following Shu (2003). The purpose of the reconstruction operator is to solve Problem 1, i.e. reconstruct the value of the underlying function at a certain position given a set of cell averages.

Algorithm 2 Finite volume scheme for the three-dimensional Euler equations.

1: $\mathbf{Q}_{i,j,k} = \overline{\mathbf{Q}}_{i,j,k}$ is given as cell average.
2: $\mathbf{Q}_{i\pm\frac{1}{2},j,k} = \mathrm{R}_x\left(\overline{\mathbf{Q}}_{i,j,k}\right)$, $\mathbf{Q}_{i,j\pm\frac{1}{2},k} = \mathrm{R}_y\left(\overline{\mathbf{Q}}_{i,j,k}\right)$. $\mathbf{Q}_{i,j,k\pm\frac{1}{2}} = \mathrm{R}_z\left(\overline{\mathbf{Q}}_{i,j,k}\right)$
3: $\mathbf{F}_{i\pm\frac{1}{2},j,k} = \mathbf{F}\left(\mathbf{Q}_{i\pm\frac{1}{2},j,k}\right)$, $\mathbf{G}_{i,j\pm\frac{1}{2},k} = \mathbf{G}\left(\mathbf{Q}_{i,j\pm\frac{1}{2},k}\right)$, $\mathbf{H}_{i,j,k\pm\frac{1}{2}} = \mathbf{H}\left(\mathbf{Q}_{i,j,k\pm\frac{1}{2}}\right)$
4:

$$\begin{aligned}\frac{\partial\overline{\mathbf{Q}}_{i,j,k}}{\partial t} = &-\frac{1}{\delta x}\left(\mathbf{F}_{i+\frac{1}{2},j,k} - \mathbf{F}_{i-\frac{1}{2},j,k}\right) - \frac{1}{\delta y}\left(\mathbf{G}_{i,j+\frac{1}{2},k} - \mathbf{G}_{i,j-\frac{1}{2},k}\right)\\ &-\frac{1}{\delta z}\left(\mathbf{H}_{i,j,k+\frac{1}{2}} - \mathbf{H}_{i,j,k-\frac{1}{2}}\right)\end{aligned}$$

We will only consider equidistant one-dimensional grids and reconstruction of the value at the half-integer node $i+\frac{1}{2}$. This is sufficient for the finite difference and finite volume algorithms used in this paper.

The idea of the WENO reconstruction process is to use several stencils in the neighbourhood of $i+\frac{1}{2}$. On each of the stencils, an interpolating polynomial of high order is defined. The interpolated value is obtained by a convex combination of these polynomials weighted according to their smoothness. If a discontinuity is contained in the stencil of a polynomial, its weight will be very small thereby avoiding oscillatory behaviour known from high order interpolation.

Assume that the cell averages $\overline{\phi}_i = (\mathrm{A}\phi)_i$ of the function ϕ are given and $\phi_{i+\frac{1}{2}}$ should be reconstructed. We consider k stencils

$$S_r(i) = \{x_{i-r}, \ldots, x_{i-r+k-1}\},\ r = 0, \ldots, k-1. \tag{3.16}$$

On each stencil $S_r(i)$ a polynomial p_r of degree $k-1$ is defined such that the approximation $\phi^{(r)}_{i+\frac{1}{2}}$ to $\phi_{i+\frac{1}{2}}$ is given by

$$\phi^{(r)}_{i+\frac{1}{2}} = p_r(x_{i+\frac{1}{2}}) + O((\delta x)^k) \tag{3.17}$$

and

$$\mathrm{A}(p_r) = \mathrm{A}(\phi) = \overline{\phi} \text{ on } S_r(i). \tag{3.18}$$

Solving the resulting linear system for the case $k = 3$ gives the interpolation polynomials

$$p_0(x_{i+\frac{1}{2}}) = \frac{1}{3}\overline{\phi}_{i-2} - \frac{7}{6}\overline{\phi}_{i-1} + \frac{11}{6}\overline{\phi}_i, \tag{3.19}$$

$$p_1(x_{i+\frac{1}{2}}) = -\frac{1}{6}\overline{\phi}_{i-1} + \frac{5}{6}\overline{\phi}_i + \frac{1}{3}\overline{\phi}_{i+1}, \tag{3.20}$$

$$p_2(x_{i+\frac{1}{2}}) = \frac{1}{3}\overline{\phi}_i + \frac{5}{6}\overline{\phi}_{i+1} - \frac{1}{6}\overline{\phi}_{i+2}, \tag{3.21}$$

The interpolating polynomial of fifth order can be obtained by combination of the three polynomials of third order. If we define the weights

$$d_0 = \frac{3}{10}, d_1 = \frac{3}{5}, d_2 = \frac{1}{10}, \tag{3.22}$$

the fifth order interpolation polynomial $p^{(5)}$ can be obtained by

$$p^{(5)}(x) = d_0 p_0(x) + d_1 p_1(x) + d_2 p_2(x). \tag{3.23}$$

High-order polynomial interpolation is known to produce oscillatory results. To avoid oscillations in the WENO approach, a convex combination of all candidate stencils p_r is used to compute $\phi_{i+\frac{1}{2}}$. This procedure leads to non-oscillatory approximations of order $2k-1$, where k is the width of each of the stencils $S_r(i)$.

Therefore, the approximation to $\phi_{i+\frac{1}{2}}$ is calculated by

$$\phi_{i+\frac{1}{2}} = \omega_0 p_0(x_{i+\frac{1}{2}}) + \omega_1 p_1(x_{i+\frac{1}{2}}) + \omega_2 p_2(x_{i+\frac{1}{2}}), \tag{3.24}$$

where ω_0, ω_1 and ω_2 are nonlinear weights comparing the smoothness of the interpolation polynomials. Defining

$$\begin{aligned}
\beta_0 &= \frac{13}{12}\left(\overline{\phi}_i - 2\overline{\phi}_{i+1} + \overline{\phi}_{i+2}\right)^2 + \frac{1}{4}\left(3\overline{\phi}_i - 4\overline{\phi}_{i+1} + \overline{\phi}_{i+2}\right)^2, \\
\beta_1 &= \frac{13}{12}\left(\overline{\phi}_{i-1} - 2\overline{\phi}_i + \overline{\phi}_{i+1}\right)^2 + \frac{1}{4}\left(\overline{\phi}_{i-1} - \overline{\phi}_{i+1}\right)^2, \\
\beta_2 &= \frac{13}{12}\left(\overline{\phi}_{i-2} - 2\overline{\phi}_{i-1} + \overline{\phi}_i\right)^2 + \frac{1}{4}\left(\overline{\phi}_{i-2} - 4\overline{\phi}_{i-1}\right)^2,
\end{aligned} \tag{3.25}$$

we calculate

$$\tilde{\omega}_0 = \frac{d_0}{(\beta_0+\epsilon)^2},\ \tilde{\omega}_1 = \frac{d_1}{(\beta_1+\epsilon)^2},\ \tilde{\omega}_2 = \frac{d_2}{(\beta_2+\epsilon)^2}, \tag{3.26}$$

and finally

$$\omega_0 = \frac{\tilde{\omega}_0}{\tilde{\omega}_0+\tilde{\omega}_1+\tilde{\omega}_2},\ \omega_1 = \frac{\tilde{\omega}_1}{\tilde{\omega}_0+\tilde{\omega}_1+\tilde{\omega}_2},\ \omega_2 = \frac{\tilde{\omega}_2}{\tilde{\omega}_0+\tilde{\omega}_1+\tilde{\omega}_2}. \tag{3.27}$$

ϵ is a small parameter which is used to avoid division by zero.

3.2 Explicit Runge–Kutta Time Integration Schemes

In Section 3.1, we followed the *method of lines* approach of discretising space and time separately (Toro, 2009; LeVeque, 2007). The continuous problem is converted to

$$\frac{\partial \mathbf{Q}}{\partial t} = \mathrm{L}\left(\mathbf{Q}^n\right), \tag{3.28}$$

where L is the operator resulting from the spatial (WENO) discretisation procedure. The integration of this equation can be performed with any numerical method for solving ordinary differential equations, in particular Runge–Kutta methods.

We follow Gottlieb et al. (2001) in defining some basic properties of Runge–Kutta schemes.

Definition 3. *Let an initial value problem of the form*

$$\phi'(t) = \mathrm{L}\left(\phi(t)\right), \ \phi(0) = \phi_0, \tag{3.29}$$

be given. An explicit s–stage Runge–Kutta scheme is an integration scheme of the form

$$\begin{aligned} \phi^{(0)} &= \phi^n, \\ \phi^{(i)} &= \sum_{k=0}^{i-1} \left(\alpha_{i,k}\, \phi^{(k)} + \delta t\, \beta_{i,k}\, \mathrm{L}(\phi^{(k)}) \right), \ \alpha_{i,k} \geq 0, \ i = 1, \ldots, s, \\ \phi^{n+1} &= \phi^{(s)}, \end{aligned} \tag{3.30}$$

where $\phi^n = \phi(t_n)$ and the time step δt is given by the CFL condition.

Definition 4. *Assume that* L *results from the discretisation of a spatial operator and let a seminorm $\|\cdot\|$ be given. Following Wang and Spiteri (2007), a Runge–Kutta method of the form* (3.30) *is called strong stability preserving (SSP) if for all stages i, $i = 1, 2, \ldots s$,*

$$\left\| \phi^{(i)} \right\| \leq \|\phi^n\| \tag{3.31}$$

with a CFL restriction on the time step δt.

The *total variation diminishing (TVD)* property (Shu and Osher, 1988) is a special case of this definition. It results from inserting the *total variation* norm of ϕ at time t_n,

$$\mathrm{TV}(\phi^n) = \sum_j \left| \phi^n_{j+1} - \phi^n_j \right|, \tag{3.32}$$

in (3.31).

In this paragraph, we consider four explicit time integration schemes: the first-order Euler forward method, the second-order two-stage TVD2 and the third-order three-stage TVD3 scheme from Shu and Osher (1988). The fourth explicit scheme is the second-order three-stage scheme from Kraaijevanger (1991), further studied in Ketcheson et al. (2009) and Kupka et al. (2012), called SSP RK(3,2).

The TVD2 and TVD3 (total variation diminishing) schemes were also analysed with respect to their SSP (strong stability preserving) properties by Kraaijevanger (1991). Their coefficients were first derived by Heun (1900) and Fehlberg (1970) from a different viewpoint. They are the explicit Runge–Kutta schemes of second order with two stages (TVD2) and of third order with three stages (TVD3) which have the largest domain for which the SSP property holds among all schemes of such order and such number of stages, i.e. they are the optimum SSP RK(2,2) and SSP RK(3,3) schemes. The SSP RK(3,2) scheme is the optimum one among all three-stage explicit Runge–Kutta schemes with SSP property, if the approximate order is required to be only two instead of three (see Kraaijevanger (1991) for proofs of these results). Their Butcher arrays (e.g., Kraaijevanger, 1991; LeVeque, 2007) and their Shu–Osher arrays (Shu and Osher, 1988) are given in Table 3.1 resp. Table 3.2.

We note that all schemes are explicit schemes. According to Wang and Spiteri (2007), they are all linearly unstable in theory when coupled with the WENO5 scheme. But the Courant numbers we use are small enough in terms of Motamed et al. (2011) to make the combination with WENO5 stable in practical applications.

0		
1	1	
A_{TVD2}	$\frac{1}{2}$	$\frac{1}{2}$

0			
$\frac{1}{2}$	$\frac{1}{2}$		
1	$\frac{1}{2}$	$\frac{1}{2}$	
$A_{\text{SSP RK(3,2)}}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

0			
1	1		
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	
A_{TVD3}	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{2}{3}$

Table 3.1: The Butcher arrays of the explicit schemes considered in Section 3.2. From left to right: TVD2, SSP RK(3,2), TVD3.

scheme	order	stages	α_i			β_i		
Euler	1	1	1			1		
TVD2	2	2	1			1		
			$\frac{1}{2}$	$\frac{1}{2}$		0	$\frac{1}{2}$	
SSP RK(3,2)	2	3	1			$\frac{1}{2}$		
			0	1		0	$\frac{1}{2}$	
			$\frac{1}{3}$	0	$\frac{2}{3}$	0	0	$\frac{1}{3}$
TVD3	3	3	1			1		
			$\frac{3}{4}$	$\frac{1}{4}$		0	$\frac{1}{4}$	
			$\frac{1}{3}$	0	$\frac{2}{3}$	0	0	$\frac{2}{3}$

Table 3.2: The Shu–Osher arrays (Shu and Osher, 1988) of the explicit schemes considered in Section 3.2.

3.2.1 Some Thoughts on Efficiency

In practice, the order of accuracy is not sufficient to describe the efficiency of a Runge–Kutta method. As already mentioned in Chapter 2 and described in Appendix A.6 in LeVeque (2007), we assume that the error ε of a method decays with the step size h as

$$\varepsilon(h) \approx Ch^p, \tag{3.33}$$

where p is the *(empirical) order of convergence* or *order of accuracy* and C is the *error constant* of the method. $\varepsilon(h)$ is the numerical error at grid spacing h. A higher order method may, for a given grid, deliver worse results than a lower order scheme due to its high error constant C (p. 35, Ferziger and Perić, 2002).

p and C can be estimated from a numerical solution if the exact (or at least, very accurate) solution is known by comparing the error for several values of h. If, for example, the resolution is increased by a factor 2, the convergence rate p can be estimated by

$$p = \log_2 \left(\varepsilon(h)/\varepsilon(h/2)\right). \tag{3.34}$$

Then, the error constant of the method can be calculated by

$$C = \varepsilon(h)/h^p. \tag{3.35}$$

The obtained values depend on the test problem and on the norm chosen to measure the error size.

Analytical Test Cases

We compare the efficiency and accuracy of several time integration schemes by solving the advection equation

$$\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x} = 0 \tag{3.36}$$

for $t > 0$ and $x \in [0, 1]$ with periodic boundary conditions, and the WENO5 scheme for spatial discretisation. With the initial condition

$$\phi(x, 0) = 1 + 0.1 \sin(2\pi x), \tag{3.37}$$

the analytical solution stays smooth for all times. Therefore, this is an appropriate test case for determining the empirical order of accuracy and the error constants of a method.

Given discontinuous initial data,

$$\phi(x, 0) = \begin{cases} 1, & \text{if } 0.1 < x < 0.3, \\ 0, & \text{else,} \end{cases} \tag{3.38}$$

the convergence order is restricted by the smoothness of the solution.

The analytical solution of the advection equation (3.36) at time t is $\phi(x, t) = \phi(x-t, 0)$. By comparing the numerical solution to the analytical one, we calculate the mean L^2

error after 2 s (cf. Appendix A.5 in LeVeque, 2007) for a set of spatial and temporal resolutions. The results are given for the smooth initial condition (3.37) in Tables 3.10, 3.11, 3.13 and 3.12 for the Euler forward, the TVD2, the TVD3, and the SSP RK(3,2) scheme, respectively. For the discontinuous initial condition, they can be found in Tables 3.14, 3.15, 3.17 and 3.16. In each row, the spatial resolution is fixed, whereas in the columns, the temporal resolution is constant. Since ϕ is of magnitude 1, the absolute errors shown are also relative errors.

For the advection equation, the (advective) Courant number σ is defined by

$$\sigma = u\frac{\delta t}{\delta x}, \tag{3.39}$$

where u is the advection velocity. In these tests, $u = 1$. On the diagonal of each of the error tables, we see the error for a fixed Courant number. If the solution was not stable, we do not give a number for the error size. In most cases, the algorithm is stable only if $\sigma < 1$.

From these data we can deduce the size of the temporal and spatial error for each scheme, and its dependence on the Courant number. For the smooth initial condition (3.37), we observe that the error $\varepsilon(\delta x, \delta t)$ of the Euler scheme is never smaller than 10^{-4}. It shows approximately first order convergence in time. For many combinations of δt and δx, decreasing δx does not lead to a decrease in the error, since the error is dominated by the error of the time integration scheme. We conclude that the Euler forward scheme is not efficient unless the spatial resolution is very coarse. Then, the maximum allowed Courant numbers are rather small.

For the other schemes, the error reaches much smaller magnitudes down to approximately machine precision. In most cases, temporal and spatial error are balanced or the spatial error dominates except for the regions where both resolutions are either very coarse or very fine. The TVD3 scheme shows the smallest errors, but these are only reached with very high spatial and temporal resolution.

For the discontinuous problem, the errors are much larger. For a large range of combinations of δt and δx, the error is nearly independent of the temporal scheme. It does decrease with spatial resolution, but at a much slower rate determined by the smoothness of the solution. Nevertheless, the stability properties are different. The Euler forward scheme is always unstable for $\sigma \geq 1$, except for very coarse resolution. All other schemes give stable solutions with $\sigma = 1$, but often they are very inaccurate. Only the SSP RK(3,2) scheme seems to allow $\sigma > 1$, at least for coarse resolutions.

In Tables 3.3, 3.4, and 3.5, we give the empirical order of accuracy and error constant calculated for fixed values of σ when solving the advection equation (3.36) with smooth initial data (3.37). We clearly see second order convergence for TVD2 and SSP RK(3,2) and third order convergence for TVD3 as soon as the spatial resolution is high. For small Courant numbers, the convergence rate of TVD3 is even higher, but always in combination with a huge error constant. For lower resolution, the spatial error dominates and we see the fifth order convergence of the WENO5 scheme used for the spatial discretisation.

It is not reasonable to compare the error constants if the order of accuracy of the methods are different. This is also the reason why we did not give data for the first-order Euler scheme. Nevertheless we observe that the error constant of TVD2 is twice as large as the one of SSP RK(3,2). Therefore, SSP RK(3,2) is the more efficient scheme even though it has three stages whereas TVD2 has two stages. The smaller the Courant number is the smaller is the influence of the time integration scheme on the overall accuracy.

For the discontinuous initial condition (3.38) we give the mean orders and error constants for all integration schemes in Table 3.26, averaged over all spatial resolutions. They do not differ much from method to method, since the accuracy of the numerical solution is limited by the smoothness of the analytical solution. Only the Euler forward method shows other convergence rates with much higher error constants, indicating very irregular error convergence. This means that the additional effort of using a three-stage scheme does not pay off in terms of accuracy compared to TVD2. In terms of stability, SSP RK(3,2) is the best scheme.

Finally, we solve the one-dimensional diffusion equation

$$\frac{\partial \phi}{\partial t} - D\frac{\partial^2 \phi}{\partial x^2} = 0 \tag{3.40}$$

for $t > 0$ and $x \in [0, 2]$ with periodic boundary conditions and with initial data

$$\phi\left(x, 0\right) = 1.1 + 0.1 \sin\left(\pi x\right). \tag{3.41}$$

The analytical solution is

$$\phi\left(x, t\right) = 1.1 + 0.1 \exp\left(-D\pi^2 t\right) \sin\left(\pi x\right). \tag{3.42}$$

$D > 0$ is the (constant) diffusion coefficient which we choose as 10^{-3}. For the conservative discretisation of $\frac{\partial^2 \phi}{\partial x^2}$, the fourth-order order stencils from Happenhofer et al. (2013) are used. On an equidistant grid, the first derivative of a function ϕ is approximated by

$$\frac{\partial \phi}{\partial x}\left(x_{i+\frac{1}{2}}\right) \approx \frac{\phi_{i-1} + 15\phi_i - 15\phi_{i+1} - \phi_{i-2}}{12\,\delta x}. \tag{3.43a}$$

We write $\phi_i = \phi(x_i)$. Then, the conservative discretisation of the outer derivative leads to a fourth order accurate approximation of the second derivative. More precisely,

$$\frac{\partial^2 \phi}{\partial x^2}\left(x_i\right) \approx \frac{\frac{\partial \phi}{\partial x}(x_{i+\frac{1}{2}}) - \frac{\partial \phi}{\partial x}(x_{i-\frac{1}{2}})}{\delta x} \tag{3.43b}$$

is a fourth order approximation to the analytical value if the first derivatives are calculated as defined in equation (3.43a).

For the diffusion equation, we define the (diffusive) Courant number σ by

$$\sigma = D\frac{\delta t}{\delta x^2}, \tag{3.44}$$

The errors of the numerical solutions calculated after 50 s are shown in Tables 3.18, 3.19, 3.21 and 3.20. Keeping the Courant number σ fixed and decreasing the grid spacing means decreasing the time step size quadratically. Therefore, the empirical convergence rates and error constants shown in Tables 3.6, 3.7, 3.8 and 3.9 exhibit second to fourth order convergence since the convergence rate of the time integration is doubled. The overall convergence is then restricted by the fourth order spatial discretisation as defined in equations (3.43). The increase in error for very small grid spacings is due to rounding errors, and is larger the more stages the time integration scheme has. TVD3 yields stable results even with $\sigma = 0.5$, but this is only due to the simplicity of our test problem. In practice, TVD3 can not be used with this Courant number increasing the inefficiency of the method.

From Table 10 in Kupka et al. (2012) we deduce that the maximum Courant number σ as defined in equation (3.44) for diffusive terms is 0.375 for TVD2, 0.299 for TVD3 and 0.672 for SSP RK(3,2). From looking at the error decay in Tables 3.7, 3.8 and 3.9 we deduce that the error does not change much when going from $\sigma = 0.25$ to $\sigma = 0.5$, depending on the spatial resolution. We conclude that, depending on the spatial resolution, the high Courant number of SSP RK(3,2) makes it to the most efficient scheme for diffusion–type equations even though its theoretical order of accuracy is only 2.

One could argue that formally, it is not a consistent way of measuring convergence orders by using the spatial resolution δx as h in formula (3.33) since the number of degrees of freedom increases quadratically for the advection equation or even cubic for the diffusion equation due to the smaller time steps induced by the CFL condition. But from a practical point of view, modifying the spatial resolution and choosing a Courant number is the only way to control the accuracy of an existing simulation. Therefore, measuring the order of error decay when decreasing the spatial resolution while keeping the Courant number fixed gives the type of "convergence order" which you will encounter in applications.

We conclude that the efficiency of the time integration scheme depends on the expected smoothness and the required accuracy of the numerical solution. Therefore, in the next paragraph we try to estimate the typical accuracy and smoothness of a simulation of solar surface convection.

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
δx	p	C	p	C	p	C	p	C
0.2500	1.6042	1.11e-1	3.1336	7.25e-1	3.8831	2.16e0	3.9609	2.49e0
0.1250	1.9672	2.35e-1	2.1068	8.57e-2	3.3880	7.72e-1	4.6528	1.05e1
0.0625	1.9963	2.55e-1	1.9867	6.14e-2	2.0373	1.82e-2	2.7312	5.09e-2
0.0312	1.9998	2.58e-1	1.9964	6.35e-2	1.9942	1.57e-2	2.0064	4.13e-3
0.0156	2.0000	2.58e-1	1.9985	6.41e-2	1.9977	1.59e-2	1.9972	3.98e-3
0.0078	2.0000	2.58e-1	1.9993	6.43e-2	1.9989	1.60e-2	1.9987	4.01e-3
0.0039	2.0000	2.58e-1	1.9996	6.44e-2	1.9995	1.61e-2	1.9995	4.02e-3

Table 3.3: Empirical order of accuracy p and error constant C for WENO with TVD2 scheme for several fixed Courant numbers σ when solving (3.36) & (3.37).

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
δx	p	C	p	C	p	C	p	C
0.2500	2.2595	2.19e-1	3.7177	1.72e0	3.9350	2.36e0	3.9611	2.50e0
0.1250	1.9893	1.25e-1	2.5914	1.65e-1	4.1633	3.80e0	4.8362	1.54e1
0.0625	1.9945	1.27e-1	1.9934	3.14e-2	2.2324	1.80e-2	3.4997	3.79e-1
0.0312	1.9997	1.29e-1	1.9960	3.17e-2	1.9958	7.91e-3	2.0598	2.58e-3
0.0156	2.0000	1.29e-1	1.9985	3.20e-2	1.9976	7.97e-3	1.9976	1.99e-3
0.0078	2.0000	1.29e-1	1.9993	3.22e-2	1.9989	8.02e-3	1.9987	2.00e-3
0.0039	2.0000	1.29e-1	1.9996	3.22e-2	1.9995	8.04e-3	1.9995	2.01e-3

Table 3.4: Empirical order of accuracy p and error constant C for WENO with SSP RK(3,2) scheme for several fixed Courant numbers σ when solving (3.36) & (3.37).

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
δx	p	C	p	C	p	C	p	C
0.2500	3.8447	2.45e0	3.9783	2.66e0	3.9397	2.42e0	3.9584	2.50e0
0.1250	4.1773	4.90e0	4.7464	1.31e1	4.8706	1.68e1	4.8857	1.72e1
0.0625	3.6070	1.01e0	4.5749	8.17e0	4.9584	2.14e1	5.0160	2.47e1
0.0312	3.2028	2.49e-1	3.9409	9.08e-1	4.7613	1.08e1	4.9858	2.22e1
0.0156	3.0561	1.35e-1	3.3692	8.42e-2	4.2681	1.39e0	4.8700	1.37e1
0.0078	3.0144	1.10e-1	3.1087	2.38e-2	3.6170	5.90e-2	4.5418	2.79e0
0.0039	3.0036	1.04e-1	3.0283	1.52e-2	3.2035	5.95e-3	3.6701	2.22e-2

Table 3.5: Empirical order of accuracy p and error constant C for WENO with TVD3 scheme for several fixed Courant numbers σ when solving (3.36) & (3.37).

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
δx	p	C	p	C	p	C	p	C
0.25000	1.8938	2.56e-2	1.3826	2.07e-3	-1.3608	1.37e-6	9.5049	4.51e1
0.12500	1.8806	2.49e-2	1.8231	5.18e-3	1.5568	5.91e-4	-3.7697	4.64e-11
0.06250	1.9535	3.04e-2	1.9262	6.90e-3	1.8760	1.43e-3	1.6425	1.53e-4
0.03125	1.9715	3.24e-2	1.9683	7.98e-3	1.9557	1.89e-3	1.9074	3.82e-4
0.01562	1.9865	3.45e-2	1.9848	8.55e-3	1.9820	2.11e-3	1.9705	4.97e-4
0.00781	1.9929	3.56e-2	1.9927	8.88e-3	1.9920	2.21e-3	1.9891	5.44e-4

Table 3.6: Empirical order of accuracy p and error constant C for solving (3.40) & (3.41) with Euler forward and fixed Courant numbers.

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
δx	p	C	p	C	p	C	p	C
0.25000	3.9394	4.83e-2	3.8610	2.45e-2	3.8850	2.45e-2	3.8729	2.38e-2
0.12500	3.8917	4.37e-2	3.9606	3.01e-2	3.9562	2.84e-2	3.9574	2.84e-2
0.06250	3.9570	5.24e-2	3.9790	3.17e-2	3.9838	3.07e-2	3.9834	3.05e-2
0.03125	3.9727	5.54e-2	3.9913	3.31e-2	3.9928	3.16e-2	3.9925	3.15e-2
0.01562	3.9870	5.87e-2	3.9959	3.37e-2	3.9889	3.11e-2	3.9974	3.21e-2
0.00781	3.9938	6.07e-2	3.8854	1.97e-2	4.0487	4.16e-2	4.0310	3.78e-2

Table 3.7: Empirical order of accuracy p and error constant C for solving (3.40) & (3.41) with TVD2 and fixed Courant numbers.

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
δx	p	C	p	C	p	C	p	C
0.25000	3.9722	4.05e-2	3.8673	2.43e-2	3.8855	2.45e-2	3.8729	2.38e-2
0.12500	3.9128	3.58e-2	3.9644	2.97e-2	3.9564	2.84e-2	3.9574	2.84e-2
0.06250	3.9694	4.19e-2	3.9811	3.11e-2	3.9839	3.06e-2	3.9834	3.05e-2
0.03125	3.9795	4.34e-2	3.9924	3.24e-2	3.9929	3.16e-2	3.9930	3.15e-2
0.01562	3.9905	4.54e-2	3.9962	3.29e-2	3.9956	3.19e-2	3.9776	2.96e-2
0.00781	3.9877	4.48e-2	3.7622	1.06e-2	2.5241	2.53e-5	0.6311	2.63e-9

Table 3.8: Empirical order of accuracy p and error constant C for solving (3.40) & (3.41) with SSP RK(3,2) and fixed Courant numbers.

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
δx	p	C	p	C	p	C	p	C
0.25000	4.0284	3.30e-2	3.8739	2.41e-2	3.8859	2.45e-2	3.8729	2.38e-2
0.12500	3.9508	2.80e-2	3.9684	2.93e-2	3.9567	2.83e-2	3.9574	2.84e-2
0.06250	3.9926	3.15e-2	3.9834	3.05e-2	3.9841	3.06e-2	3.9834	3.05e-2
0.03125	3.9924	3.15e-2	3.9936	3.16e-2	3.9930	3.15e-2	3.9930	3.15e-2
0.01562	3.9973	3.21e-2	3.9968	3.20e-2	3.9960	3.19e-2	3.9816	3.01e-2
0.00781	3.9832	3.00e-2	3.7839	1.14e-2	2.6530	4.72e-5	0.7838	5.49e-9

Table 3.9: Empirical order of accuracy p and error constant C for solving (3.40) & (3.41) with TVD3 and fixed Courant numbers.

$\delta x \setminus \delta t$	0.2500	0.1250	0.0625	0.0312	0.0156	0.0078	0.0039	0.0020	0.0010	0.0005	0.0002	0.0001
0.2500	6.00e-1	1.81e-1	6.24e-2	2.17e-2	7.12e-3	5.40e-3	7.49e-3	8.84e-3	9.54e-3	9.90e-3	1.01e-2	1.02e-2
0.1250	6.54e-1	2.26e-1	8.23e-2	3.39e-2	1.56e-2	7.25e-3	3.32e-3	1.44e-3	6.10e-4	4.34e-4	5.10e-4	5.80e-4
0.0625	7.23e-1		5.58e-1	3.55e-2	1.65e-2	7.89e-3	3.85e-3	1.90e-3	9.34e-4	4.56e-4	2.18e-4	9.98e-5
0.0312					1.41e-2	7.98e-3	3.91e-3	1.93e-3	9.61e-4	4.79e-4	2.39e-4	1.19e-4
0.0156						6.99e-3	4.12e-3	1.94e-3	9.65e-4	4.81e-4	2.40e-4	1.20e-4
0.0078							4.43e-3	1.76e-3	9.67e-4	4.82e-4	2.41e-4	1.20e-4
0.0039								3.13e-3	1.13e-3	5.54e-4	2.41e-4	1.20e-4
0.0020									2.08e-3	1.14e-3	3.52e-4	1.20e-4

Table 3.10: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with Euler forward when solving (3.36) & (3.37).

$\delta x \setminus \delta t$	0.2500	0.1250	0.0625	0.0312	0.0156	0.0078	0.0039	0.0020	0.0010	0.0005	0.0002	0.0001
0.2500	5.58e-2	1.20e-2	9.41e-3	9.93e-3	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2
0.1250	9.73e-1	1.71e-2	3.94e-3	1.07e-3	6.73e-4	6.59e-4	6.62e-4	6.63e-4	6.63e-4	6.63e-4	6.63e-4	6.63e-4
0.0625			4.31e-3	1.01e-3	2.49e-4	6.43e-5	2.62e-5	2.25e-5	2.24e-5	2.25e-5	2.25e-5	2.25e-5
0.0312				2.28e-3	2.52e-4	6.28e-5	1.57e-5	3.95e-6	1.17e-6	7.22e-7	6.90e-7	6.89e-7
0.0156					2.42e-3	6.31e-5	1.57e-5	3.93e-6	9.82e-7	2.46e-7	6.45e-8	2.59e-8
0.0078						1.36e-3	1.58e-5	3.94e-6	9.84e-7	2.46e-7	6.15e-8	1.54e-8
0.0039							7.22e-4	3.94e-6	9.85e-7	2.46e-7	6.16e-8	1.54e-8
0.0020								4.05e-4	9.86e-7	2.46e-7	6.16e-8	1.54e-8

Table 3.11: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with TVD2 when solving (3.36) & (3.37).

$\delta x \setminus \delta t$	0.2500	0.1250	0.0625	0.0312	0.0156	0.0078	0.0039	0.0020	0.0010	0.0005	0.0002	0.0001
0.2500	2.63e-2	9.56e-3	9.91e-3	1.01e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2
0.1250	7.50e-2	8.27e-3	2.00e-3	7.53e-4	6.60e-4	6.62e-4	6.63e-4	6.63e-4	6.63e-4	6.63e-4	6.63e-4	6.63e-4
0.0625		2.22e-1	2.05e-3	5.03e-4	1.25e-4	3.68e-5	2.32e-5	2.24e-5	2.25e-5	2.25e-5	2.25e-5	2.25e-5
0.0312				5.09e-4	1.26e-4	3.14e-5	7.84e-6	2.05e-6	8.29e-7	6.95e-7	6.89e-7	6.89e-7
0.0156				4.58e-1	1.27e-4	3.15e-5	7.87e-6	1.97e-6	4.91e-7	1.24e-7	3.69e-8	2.24e-8
0.0078						3.16e-5	7.89e-6	1.97e-6	4.92e-7	1.23e-7	3.07e-8	7.71e-9
0.0039							1.20e-4	1.97e-6	4.93e-7	1.23e-7	3.08e-8	7.69e-9
0.0020								1.47e-4	4.93e-7	1.23e-7	3.08e-8	7.70e-9

Table 3.12: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with SSP RK(3,2) when solving (3.36) & (3.37).

$\delta x \setminus \delta t$	0.2500	0.1250	0.0625	0.0312	0.0156	0.0078	0.0039	0.0020	0.0010	0.0005	0.0002	0.0001
0.2500	1.81e-2	1.19e-2	1.07e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2	1.03e-2
0.1250	9.28e-2	2.14e-3	8.28e-4	6.80e-4	6.70e-4	6.66e-4	6.64e-4	6.63e-4	6.64e-4	6.63e-4	6.63e-4	6.63e-4
0.0625		7.83e-2	2.21e-4	4.58e-5	2.53e-5	2.29e-5	2.25e-5	2.25e-5	2.25e-5	2.25e-5	2.25e-5	2.25e-5
0.0312				2.56e-5	3.76e-6	1.06e-6	7.36e-7	6.96e-7	6.91e-7	6.90e-7	6.90e-7	6.90e-7
0.0156					3.13e-6	4.08e-7	6.92e-8	2.71e-8	2.20e-8	2.13e-8	2.12e-8	2.12e-8
0.0078						3.88e-7	4.90e-8	6.69e-9	1.41e-9	7.51e-10	6.69e-10	6.59e-10
0.0039							4.85e-8	6.07e-9	7.76e-10	1.15e-10	3.22e-11	2.20e-11
0.0020								6.05e-9	7.57e-10	9.51e-11	1.25e-11	2.53e-12

Table 3.13: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with TVD3 when solving (3.36) & (3.37).

$\delta x \setminus \delta t$	0.2500	0.1250	0.0625	0.0312	0.0156	0.0078	0.0039	0.0020	0.0010	0.0005	0.0002	0.0001
0.2500	8.15e-1	2.95e-1	4.16e-1	4.49e-1	4.48e-1	4.51e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1
0.1250			2.31e-1	3.57e-1	3.18e-1	3.31e-1	3.39e-1	3.44e-1	3.45e-1	3.46e-1	3.46e-1	3.46e-1
0.0625				3.77e-1	3.49e-1	2.31e-1	2.30e-1	2.34e-1	2.40e-1	2.43e-1	2.45e-1	2.45e-1
0.0312					1.23e-1	3.41e-1	2.22e-1	1.53e-1	1.39e-1	1.43e-1	1.47e-1	1.49e-1
0.0156						8.39e-2	7.10e-2	8.18e-2	9.40e-2	1.04e-1	1.12e-1	1.16e-1
0.0078							6.43e-2	6.84e-2	5.78e-2	6.61e-2	7.38e-2	8.03e-2
0.0039								5.34e-2	3.50e-2	4.08e-2	4.67e-2	5.22e-2
0.0020									1.43e-1		2.88e-2	3.30e-2

Table 3.14: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with Euler forward when solving (3.36) & (3.38).

$\delta x \setminus \delta t$	0.2500	0.1250	0.0625	0.0312	0.0156	0.0078	0.0039	0.0020	0.0010	0.0005	0.0002	0.0001
0.2500	4.80e-1	4.20e-1	4.38e-1	4.53e-1	4.51e-1	4.53e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1
0.1250		4.35e-1	3.05e-1	3.38e-1	3.38e-1	3.43e-1	3.45e-1	3.47e-1	3.46e-1	3.46e-1	3.46e-1	3.47e-1
0.0625			3.58e-1	2.42e-1	2.35e-1	2.40e-1	2.46e-1	2.45e-1	2.46e-1	2.46e-1	2.46e-1	2.46e-1
0.0312				3.12e-1	1.98e-1	1.57e-1	1.52e-1	1.51e-1	1.51e-1	1.51e-1	1.51e-1	1.51e-1
0.0156					2.55e-1	1.45e-1	1.23e-1	1.21e-1	1.20e-1	1.20e-1	1.20e-1	1.20e-1
0.0078						1.91e-1	1.06e-1	9.34e-2	9.01e-2	8.99e-2	8.99e-2	8.99e-2
0.0039							1.58e-1	7.91e-2	6.98e-2	6.77e-2	6.75e-2	6.75e-2
0.0020								1.32e-1	6.52e-2	5.36e-2	5.08e-2	5.06e-2

Table 3.15: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with TVD2 when solving (3.36) & (3.38).

$\delta x \setminus \delta t$	0.2500	0.1250	0.0625	0.0312	0.0156	0.0078	0.0039	0.0020	0.0010	0.0005	0.0002	0.0001
0.2500	4.40e-1	4.25e-1	4.40e-1	4.54e-1	4.51e-1	4.53e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1
0.1250	5.66e-1	3.66e-1	3.09e-1	3.42e-1	3.39e-1	3.43e-1	3.45e-1	3.47e-1	3.46e-1	3.46e-1	3.46e-1	3.47e-1
0.0625		3.81e-1	2.97e-1	2.49e-1	2.37e-1	2.41e-1	2.46e-1	2.45e-1	2.46e-1	2.46e-1	2.46e-1	2.46e-1
0.0312			5.03e-1	1.98e-1	1.76e-1	1.55e-1	1.52e-1	1.51e-1	1.51e-1	1.51e-1	1.51e-1	1.51e-1
0.0156					1.55e-1	1.33e-1	1.22e-1	1.21e-1	1.20e-1	1.20e-1	1.20e-1	1.20e-1
0.0078						1.17e-1	9.56e-2	9.38e-2	9.01e-2	8.99e-2	8.99e-2	8.99e-2
0.0039							9.32e-2	6.98e-2	6.85e-2	6.77e-2	6.76e-2	6.76e-2
0.0020								7.42e-2	5.80e-2	5.17e-2	5.07e-2	5.06e-2

Table 3.16: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with SSP RK(3,2) when solving (3.36) & (3.38).

$\delta x \setminus \delta t$	0.2500	0.1250	0.0625	0.0312	0.0156	0.0078	0.0039	0.0020	0.0010	0.0005	0.0002	0.0001
0.2500	4.56e-1	4.32e-1	4.42e-1	4.54e-1	4.51e-1	4.53e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1	4.54e-1
0.1250		4.01e-1	3.19e-1	3.46e-1	3.40e-1	3.43e-1	3.45e-1	3.47e-1	3.46e-1	3.46e-1	3.46e-1	3.47e-1
0.0625			3.13e-1	2.70e-1	2.40e-1	2.42e-1	2.46e-1	2.45e-1	2.46e-1	2.46e-1	2.46e-1	2.46e-1
0.0312				2.44e-1	1.64e-1	1.54e-1	1.52e-1	1.51e-1	1.51e-1	1.51e-1	1.51e-1	1.51e-1
0.0156					1.95e-1	1.27e-1	1.22e-1	1.21e-1	1.20e-1	1.20e-1	1.20e-1	1.20e-1
0.0078						1.73e-1	9.44e-2	9.47e-2	9.01e-2	8.99e-2	8.99e-2	8.99e-2
0.0039							1.48e-1	7.05e-2	6.82e-2	6.77e-2	6.76e-2	6.76e-2
0.0020								1.21e-1	5.28e-2	5.11e-2	5.08e-2	5.07e-2

Table 3.17: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with TVD3 when solving (3.36) & (3.38).

$\delta x \setminus \delta t$	15.62500	3.90625	0.97656	0.24414	0.06104	0.01526	0.00381	0.00095	0.00024	0.00006	0.00001
0.25000	1.85e-3	3.05e-4	9.04e-6	8.55e-5	1.05e-4	1.09e-4	1.10e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4
0.12500	2.37e-3	4.98e-4	1.17e-4	2.32e-5	1.18e-7	5.65e-6	7.09e-6	7.45e-6	7.54e-6	7.56e-6	7.56e-6
0.06250	2.58e-3	5.51e-4	1.35e-4	3.31e-5	7.89e-6	1.61e-6	3.58e-8	3.57e-7	4.55e-7	4.79e-7	4.85e-7
0.03125	2.69e-3			3.49e-5	8.70e-6	2.15e-6	5.14e-7	1.05e-7	3.25e-9	2.23e-8	2.87e-8
0.01562	2.74e-3				8.90e-6	2.22e-6	5.54e-7	1.37e-7	3.28e-8	6.75e-9	2.37e-10
0.00781	2.77e-3					2.25e-6	5.61e-7	1.40e-7	3.50e-8	8.65e-9	2.07e-9
0.00391	4.16e-2						5.64e-7	1.41e-7	3.53e-8	8.81e-9	2.20e-9
0.00195								1.41e-7	3.54e-8	8.84e-9	2.21e-9

Table 3.18: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with Euler forward when solving (3.40) & (3.41).

$\delta x \setminus \delta t$	15.62500	3.90625	0.97656	0.24414	0.06104	0.01526	0.00381	0.00095	0.00024	0.00006	0.00001
0.25000	2.05e-4	1.16e-4	1.12e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4
0.12500	1.19e-4	1.34e-5	7.99e-6	7.59e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6
0.06250	1.29e-4	2.84e-5	9.01e-7	5.13e-7	4.89e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7
0.03125	3.00e-3			5.80e-8	3.25e-8	3.09e-8	3.08e-8	3.08e-8	3.08e-8	3.08e-8	3.08e-8
0.01562					3.70e-9	2.05e-9	1.94e-9	1.94e-9	1.94e-9	1.94e-9	1.94e-9
0.00781						2.33e-10	1.28e-10	1.22e-10	1.21e-10	1.21e-10	1.21e-10
0.00391							1.46e-11	8.67e-12	7.39e-12	7.42e-12	7.43e-12
0.00195								1.00e-12	6.24e-13	5.67e-13	6.24e-13

Table 3.19: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with TVD2 when solving (3.40) & (3.41).

$\delta x \setminus \delta t$	15.62500	3.90625	0.97656	0.24414	0.06104	0.01526	0.00381	0.00095	0.00024	0.00006	0.00001
0.25000	1.64e-4	1.14e-4	1.12e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4
0.12500	6.18e-5	1.05e-5	7.81e-6	7.58e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6
0.06250	6.24e-5	3.81e-6	6.96e-7	5.00e-7	4.88e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7
0.03125				4.44e-8	3.17e-8	3.09e-8	3.08e-8	3.08e-8	3.08e-8	3.08e-8	3.08e-8
0.01562					2.82e-9	1.99e-9	1.94e-9	1.94e-9	1.94e-9	1.94e-9	1.96e-9
0.00781						1.77e-10	1.25e-10	1.22e-10	1.23e-10	1.44e-10	3.30e-10
0.00391							1.12e-11	9.19e-12	2.11e-11	7.93e-11	3.12e-10
0.00195								4.71e-12	1.97e-11	7.94e-11	3.14e-10

Table 3.20: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with SSP RK(3,2) when solving (3.40) & (3.41).

$\delta x \setminus \delta t$	15.62500	3.90625	0.97656	0.24414	0.06104	0.01526	0.00381	0.00095	0.00024	0.00006	0.00001
0.25000	1.24e-4	1.12e-4	1.12e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4	1.11e-4
0.12500	4.72e-6	7.58e-6	7.63e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6	7.57e-6
0.06250	3.36e-5	3.88e-3	4.90e-7	4.88e-7	4.88e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7	4.87e-7
0.03125				3.08e-8	3.08e-8	3.08e-8	3.08e-8	3.08e-8	3.08e-8	3.08e-8	3.08e-8
0.01562					1.94e-9	1.94e-9	1.94e-9	1.94e-9	1.94e-9	1.94e-9	1.96e-9
0.00781						1.21e-10	1.21e-10	1.21e-10	1.23e-10	1.40e-10	3.01e-10
0.00391							7.67e-12	8.80e-12	1.93e-11	7.12e-11	2.78e-10
0.00195								4.56e-12	1.80e-11	7.13e-11	2.80e-10

Table 3.21: $\varepsilon(\delta x, \delta t)$ for the combination of WENO5 with TVD3 when solving (3.40) & (3.41).

Error Size in Simulations of Solar Surface Convection

To measure the typical error in simulations of solar surface convection with ANTARES, we performed two two–dimensional simulations which only differ in the numerical resolution. Their specifications are summarised in Table 3.22.

	resolution [km]	grid points	box size [Mm]	binning
Model 1	19.5×40.0	195×150	3.8×6.0	grey
Model 2	9.74×20.0	389×300	3.8×6.0	grey

Table 3.22: Basic parameters of the two two-dimensional models from Section 3.2.1. Model 2 was started from Model 1. The data was mapped to the finer grid by interpolation, and both models were run for 31 s. Both models use the Smagorinsky subgrid model (Smagorinsky, 1963) to resolve motions with scales smaller than the grid resolution and the integration rule A4 by Carlson (1963) for the angular integration in the radiative transfer solver.

The error was calculated according to LeVeque (2007, Appendix A.6) by comparing the solution on the finer grid with the solution on the coarser grid on coinciding grid points. By assuming that the error on the finer grid is much smaller than the error on the coarser grid, the difference of the two solutions is a good estimate for the total error. The resulting error estimates in several variables and several norms are shown in Table 3.23.

variable	type	L^1	L^2	L^∞
density $\left[\mathrm{g\,cm^{-3}}\right]$	absolute	6.03e-9	1.14e-8	1.47e-7
temperature [K]	absolute	32.3	103.8	1413.4
density	relative	0.7 %	2.2 %	38.1 %
temperature	relative	0.4 %	1.4 %	29.0 %
mean temperature [K]	absolute	18.1	49.9	385.4
mean temperature	relative	0.3 %	0.7 %	5.0 %

Table 3.23: The error sizes calculated by comparing the models from Table 3.22 according to the procedure from LeVeque (2007). The norms are calculated as described in the cited reference. The relative errors are calculated by dividing the absolute difference by the value of Model 1.

We remark that these simulations are Large Eddy Simulations (LES), i.e., they do not resolve all scales of motion. All motions with length scales smaller than the grid resolution are modelled by the Smagorinsky subgrid model and by the numerical viscosity of the numerical scheme. Therefore, the measurement of "the error" is not trivial. Changing the resolution will change the results and monotonic convergence of the solution with decreasing grid spacing can not be expected due to the chaotic nature of turbulence.

We observe that the L^∞ error is much larger than the L^1 and L^2 error. This stems from the fact that near the optical surface, the motion of the fluid is turbulent and changes in the numerical parameters as grid resolution produces arbitrarily large differences. Here, the pointwise changes due to increased resolution are large compared to the rest of the simulation domain as can be seen in Figure 3.2.

Therefore, we refrain from measuring the error pointwise. Instead, we suggest to use the mean temperature profile for error measurement. In Figure 3.1 we observe that the mean temperature is much more stable but still sensitive enough to changes in the resolution. The standard deviation of the temperature profile is even more sensitive, but since it approaches 0 it is not suited for calculating relative errors. Furthermore, its behaviour near the top boundary is strongly influenced by the boundary condition (Grimm-Strele et al., 2013a). The typical mean error of the temperature profile, i.e. the relative error in the L^1 norm, is around 0.1 % to 0.5 %.

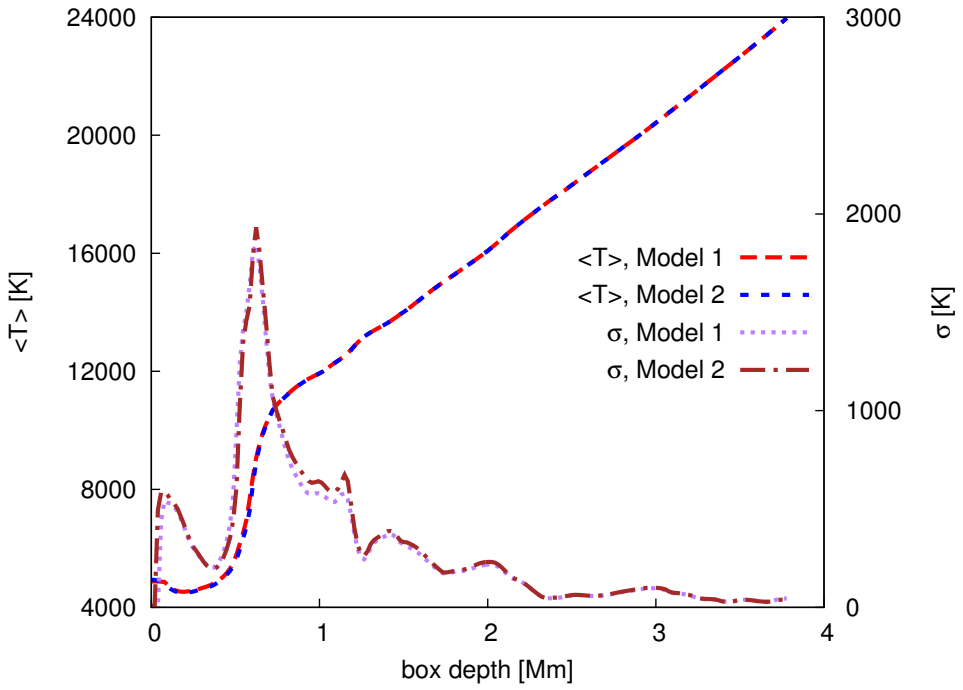

Figure 3.1: Mean temperature profile and standard deviation of Models 1 and 2 as described in Table 3.22.

In simulations where all scales of motion are resolved on the grid scale (i.e., DNS), the magnitude of the error typically is of the size 0.1 % (cf. Fig. 7 in Happenhofer et al., 2013).

Finally, we want to determine the area ratio of smooth to non-smooth regions. For this purpose, we calculate the nonlinearity index NI as defined in equation (8) of Taylor et al. (2007). Therein, the nonlinear weights ω_j of the interpolating polynomials in

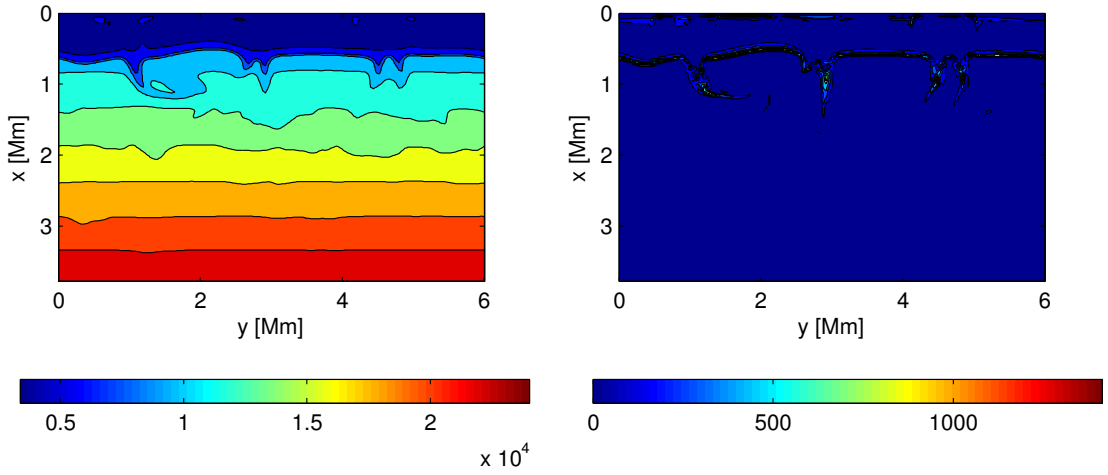

Figure 3.2: Left: snapshot of the temperature distribution [K] of Model 1; right: absolute difference of the temperature distribution [K] of Model 1 and Model 2. The parameters of the models are described in Table 3.22.

the WENO reconstruction scheme as described in Paragraph 3.1.2 are compared to the optimal linear weights d_j. In smooth regions, they should be of the same size, whereas in non-smooth regions the weight of one of the parabolae should be much higher. Then, the nonlinearity index NI defined by

$$\mathrm{NI} = \frac{1}{\sqrt{k(k+1)}} \left(\sum_{j=0}^{k} \left(1 - \frac{(k+1)\omega_j/d_j}{\sum_{l=0}^{k} \omega_l/d_l} \right)^2 \right)^{\frac{1}{2}}, \tag{3.45}$$

will be close to 1. Here, k is, as in Paragraph 3.1.2, the width of the stencil of each interpolation polynomial such that the order of the reconstruction process is $2k-1$.

We plot NI as calculated in the WENO reconstruction procedure in the first characteristic variable for reconstruction in the vertical (x) direction, for a three-dimensional standard model of solar surface convection. Its vertical resolution is 19.5 km and the horizontal 40 km for a box size of 4 Mm × 6 Mm × 6 Mm. In Figure 3.3, we show NI together with the entropy at a fixed geometrical depth near the optical surface. Actually, NI is located at the half-integer node, but we ignore this small visualisation error. In Figure 3.4, the mean value, the standard deviation, the minimum and the maximum error in each vertical layer is plotted.

We conclude that NI captures the dynamics of surface convection very well. In regions where the flow is turbulent — mainly the intergranular lanes near the optical surface (which is located at a geometrical depth of around 800 km) —, its value is large whereas it is reasonably small in smooth regions of the flow. We remark that the size of the minimum value of NI depends on the design of the nonlinear weights in the WENO reconstruction (Taylor et al., 2007). In our tests, ϵ as defined in Section 3.1.2 is fixed to

10^{-40}.

We conclude that even though NI is a purely numerical parameter, it also has a physical meaning and is a good indicator of whether a solution is smooth or not. Counting the number of points where NI < 0.25 and where NI > 0.5, we get a good estimate of the area ratio of smooth to non-smooth regions. In this particular simulation, the fraction of non-smooth regions never exceeds 8 % except for the uppermost layers which are strongly influenced by the boundary conditions. Over the whole simulation box, we can estimate the ratio to be

$$\frac{\text{volume where the flow is non-smooth}}{\text{volume where the flow is smooth}} \approx 0.05. \tag{3.46}$$

Therefore, even though the fraction of non-smooth regions is not negligible, the flow in the simulation box is mostly smooth.

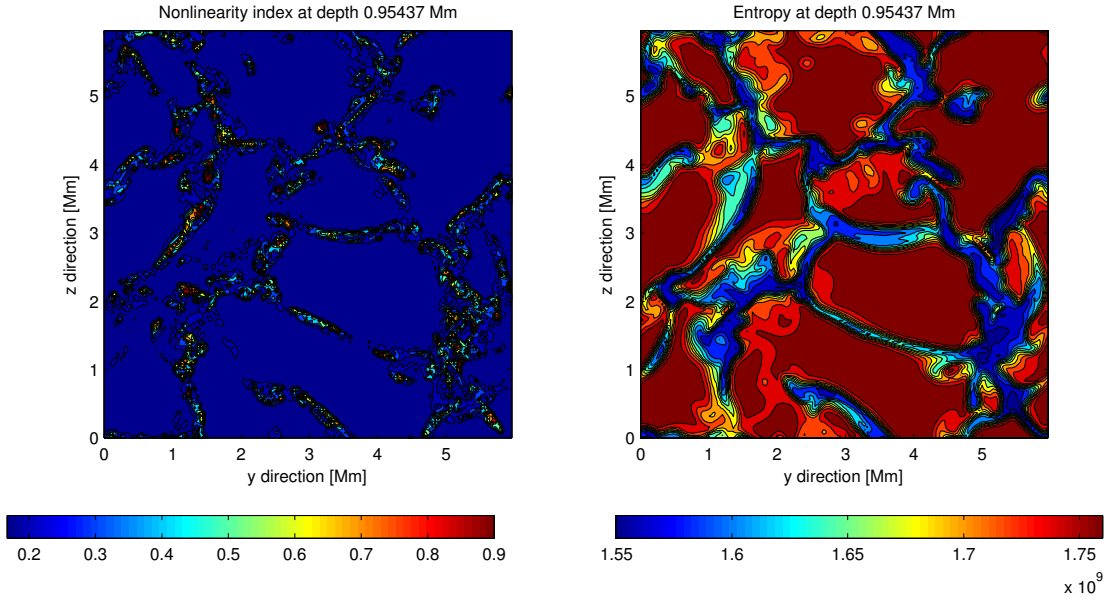

Figure 3.3: Snapshots of the nonlinearity index NI and the entropy of the three-dimensional model at a geometrical depth of around 1 Mm. The optical surface is at a geometrical depth of around 800 km. We observe that NI is largest in the intergranular lanes where the fluid motion is most turbulent (Kupka, 2009b). On the top of each granule, the flow is rather smooth such that NI is small.

Calculation of Computational Costs

The reason to use higher-order time integration schemes is that we expect more accurate results with less computation time than with the first-order Euler method. Clearly, all

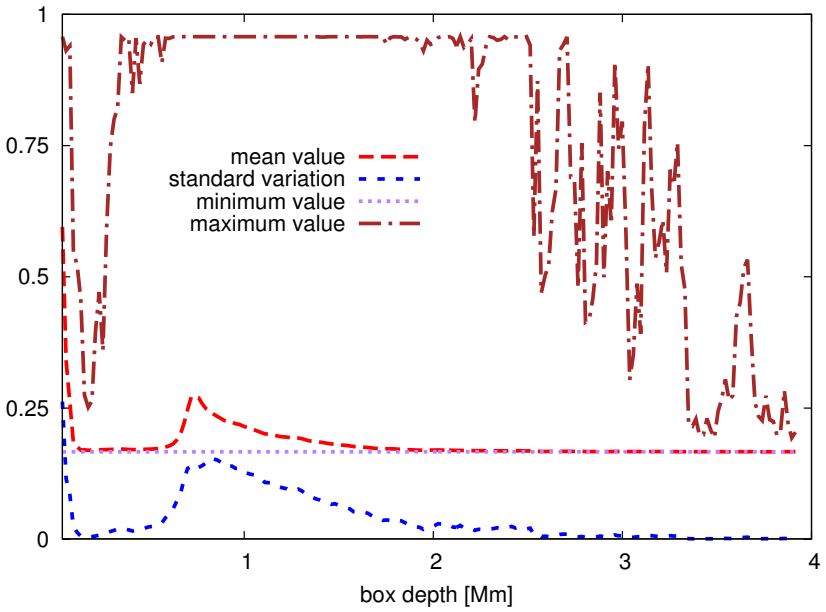

Figure 3.4: Mean value, standard deviation, minimum and maximum of the nonlinearity index NI at a specific vertical depth. We observe that NI reaches both its maximum values and its maximum average just below the optical surface. Deeper in the convection zone, the flow is smoother but NI never falls below a value of around 0.17.

of the higher order schemes from Section 3.2.1 fulfil this for all grid resolutions and for both the advection and the diffusion equation.

It is more difficult to say which one of the three higher order methods is the best for our purposes. Our approach to this problem is determining the required grid spacing to reach a typical accuracy given the error of the numerical method as calculated in the previous section at a given Courant number. In precise terms, we solve the following problem:

Problem 2. *Given a relative accuracy ε_{rel} and a Courant number, which grid spacing is necessary for a computational cube of unit size, and how many time steps are needed for a time interval of unit length?*

Given an empirical order of accuracy p and error constant C of a method, the required grid spacing can be calculated by inversion of (3.35),

$$h = \left(\frac{\varepsilon_{\text{rel}}}{C}\right)^{\frac{1}{p}}. \tag{3.47}$$

In three dimensions, we need $N = h^{-3}$ grid points for a unit cube with side length of 1 cm. For an advection equation,

$$T = n_{\text{stages}} \frac{|u|}{\sigma h} \tag{3.48}$$

integration steps are needed to cover a time interval of 1 s length. We set the advection velocity u to 1 in the following. High speeds will require small time steps and increase the importance of the time integration method. For a diffusion equation,

$$T = n_{\text{stages}} \frac{1}{\sigma h^2}, \tag{3.49}$$

assuming a diffusion coefficient D of size 1. n_{stages} is the number of stages of the Runge–Kutta method, and σ is the Courant number. Finally, the computational costs are given by $T \cdot N$ corresponding to the number of evaluations of the differential operator.

We start with considering the advection equation (3.36) with smooth initial data. From Tables 3.3, 3.4 and 3.5, we deduce the mean orders and error constants summarised in Table 3.24.

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
scheme	p	C	p	C	p	C	p	C
Euler	0.92	5.76e-1	0.82	2.49e-1	0.85	8.84e-2	0.84	5.19e-2
TVD2	1.94	2.33e-1	2.17	1.61e-1	2.47	4.31e-1	2.76	1.86e0
SSP RK(3,2)	2.03	1.41e-1	2.39	2.91e-1	2.62	8.87e-1	2.91	2.62e0
TVD3	3.42	1.28e0	3.82	3.57e0	4.23	7.55e0	4.56	1.19e1

Table 3.24: Empirical order of accuracy p and error constants C for WENO with several time integration schemes and fixed Courant numbers σ when solving (3.36) & (3.37).

scheme	Euler	TVD2	SSP RK(3,2)		TVD3	
σ	0.25	0.25	0.25	0.5	0.25	0.5
ε_{rel}	$5 \cdot 10^{-3}$					
h	0.009	0.202	0.183	0.193	0.179	0.198
N	1.6e6	121	164	139	174	130
T	469.7	39.6	65.7	31.1	67.0	30.4
overall costs	7.6e8	4.8e3	1.1e4	4.3e3	1.2e4	3.9e3

Table 3.25: Computational costs for model simulation when solving (3.36) & (3.37).

In Table 3.25, the computational costs with each scheme are calculated. According to the data from Table 3.23, we choose a relative accuracy of $5 \cdot 10^{-3}$. The Euler forward scheme is by far the most expensive one. It needs a grid spacing of around 0.01 to reach this error size. For the higher-order schemes, much larger grid spacings can be chosen. Since the total error is dominated by the spatial error in this regime, increasing

the time step by increasing the Courant number does not lead to considerably larger errors. The Courant number can be chosen as large as the stability of the method allows. Consequently, SSP RK(3,2) and TVD3 are the most efficient schemes since they allow Courant numbers of 0.5, as indicated by the data in Tables 3.12 and 3.13 and confirmed by numerical experiments with solar surface convection simulations. TVD2 is most efficient when comparing all schemes with fixed Courant number of 0.25, but it is not stable with higher Courant numbers.

With discontinuous initial data (3.38), the error size is much larger. Using the orders and constants from Table 3.26 and a relative error size of $2.5 \cdot 10^{-1}$ results in computational costs as summarised in Table 3.27.

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
scheme	p	C	p	C	p	C	p	C
Euler	0.15	5.09e0	0.60	1.47e2	0.57	5.17e0	0.54	1.00e0
TVD2	0.38	7.18e-1	0.43	7.70e-1	0.45	8.25e-1	0.45	8.47e-1
SSP RK(3,2)	0.41	7.46e-1	0.44	8.08e-1	0.45	8.30e-1	0.45	8.48e-1
TVD3	0.43	8.73e-1	0.45	8.35e-1	0.45	8.34e-1	0.45	8.49e-1

Table 3.26: Empirical order of accuracy p and error constants C for WENO with several time integration schemes and fixed Courant numbers σ when solving (3.36) & (3.38).

scheme	Euler	TVD2	SSP RK(3,2)		TVD3	
σ	0.25	0.25	0.25	0.5	0.25	0.5
ε_{rel}	$2.5 \cdot 10^{-1}$					
h	0.001	0.073	0.070	0.070	0.070	0.055
N	7.0e8	2561	2976	2979	3102	6150
T	3556.5	109.5	172.6	86.3	175.0	109.9
overall costs	2.5e12	2.8e5	5.1e5	2.6e5	5.4e5	6.8e5

Table 3.27: Computational costs for model simulation when solving (3.36) & (3.38).

We deduce from Table 3.27 that again Euler forward is very ineffective whereas for all higher-order schemes, the required grid spacing is similar. Since the error is dominated by the spatial one and the smoothness of the solution, increasing the Courant number does not considerably decrease the accuracy. Once more, SSP RK(3,2) with $\sigma = 0.5$ turns out to be more efficient than TVD2 with $\sigma = 0.25$.

For pure diffusion, we calculate the mean order of accuracy and error constant by averaging over all resolutions in Tables 3.6, 3.7, 3.8 and 3.9. Of course, these values are only rough estimates. The resulting values are summarised in Table 3.28. Due to equation (3.44), the convergence speed in time is doubled, such that the overall order for all schemes is restricted by the fourth order spatial discretisation as defined in

equations (3.43).

	$\sigma = 0.5$		$\sigma = 0.25$		$\sigma = 0.125$		$\sigma = 0.0625$	
scheme	p	C	p	C	p	C	p	C
Euler	1.94	2.95e-2	1.82	6.13e-3	1.20	1.20e-3	2.25	9.03e0
TVD2	3.95	5.17e-2	3.96	3.06e-2	3.96	2.93e-2	3.96	2.93e-2
SSP RK(3,2)	3.96	4.14e-2	3.96	3.01e-2	3.96	2.94e-2	3.96	2.88e-2
TVD3	3.99	3.12e-2	3.96	2.95e-2	3.96	2.94e-2	3.96	2.89e-2

Table 3.28: Empirical order of accuracy p and error constants C for WENO with several time integration schemes and fixed Courant numbers σ when solving (3.40) & (3.41).

scheme	Euler		TVD2		SSP RK(3,2)		TVD3	
σ	0.25	0.5	0.25	0.5	0.25	0.5	0.25	0.5
ε_{rel}	$1.0 \cdot 10^{-6}$							
h	0.008	0.005	0.074	0.064	0.074	0.068	0.074	0.075
N	1.7e6	8.2e6	2502	3801	2471	3146	2433	2394
T	5.8e4	8.1e4	1474	974	2193	1288	2171	1074
overall costs	1.0e11	6.6e11	3.7e6	3.7e6	5.4e6	4.1e6	5.2e6	2.6e6

Table 3.29: Computational costs for model simulation when solving (3.40) & (3.41).

For the diffusion equation, we expect much smaller errors due to the smoothing properties of the diffusion equation. Therefore, we calculate the computational costs for a relative accuracy of $1.0 \cdot 10^{-6}$. The computational costs of the Euler forward scheme exceed the costs of the higher order schemes by several orders of magnitude. As long as the spatial error is dominating, increasing the Courant number leads to more efficient schemes.

In contrast to non-linear methods like WENO for hyperbolic equations where stability limits for σ must be found by experiment, there are analytical methods to determine the maximum admissible Courant number for each time integration scheme. From Table 10 in Kupka et al. (2012) we deduce that the maximum Courant number σ as defined in equation (3.44) for diffusive terms is 0.375 for TVD2 and Euler forward, 0.299 for TVD3 and 0.672 for SSP RK(3,2).

Taking this into account, the efficiency of the TVD2 and the SSP RK(3,2) scheme is the highest one. Since the costs are of similar size and all numbers are only rough estimates, no clear conclusion can be drawn which scheme is the most efficient one. The fact the TVD3 is still stable with $\sigma = 0.5$ comes from the fact that the test problem (3.40) with initial condition given by equation (3.41) is rather simple and the total integration time is not very long. In realistic applications, TVD3 will not be stable with this Courant number.

In conclusion, for both advection and diffusion equations the WENO5 method combined with SSP RK(3,2) time integration is more efficient and more accurate than any other time integration schemes tested, both for smooth and non-smooth flows. We benefit from the high stability of SSP RK(3,2) and from the fact that the spatial error usually is much larger than the temporal one. This justifies the additional efforts required for the implementation of SSP RK(3,2), even though its theoretical order of accuracy is lower than TVD3 and it has more stages than TVD2.

We note that these results may change with advection velocity and diffusion coefficients since the specific choice will affect the influence of the time and spatial integration method on the overall accuracy and efficiency. Nevertheless, we assume that this influence is rather small, considering, in particular, the highly idealised nature of our estimates.

3.3 The Piecewise Parabolic Method (PPM)

We will not describe the Piecewise Parabolic Method in full detail. A precise description of the reconstruction process can be found in Colella and Woodward (1984). With the reconstructed states at the cell boundaries, a Riemann problem must be solved. A Riemann solver for real gases is described in Colella and Glaz (1985). Toro (2009) gives an extensive introduction to Riemann solvers.

The basic idea of the reconstruction process, however, is similar to the WENO procedure. We will outline the procedure for the one-dimensional case on an equidistant grid as described in Colella and Woodward (1984) and Colella and Sekora (2008). We assume that all variables are given as cell averages. In Colella and Woodward (1984), a parabolic interpolant is constructed which takes the form

$$\phi_{i+\frac{1}{2}} = \frac{-\overline{\phi}_{i-1} + 7\overline{\phi}_i + 7\overline{\phi}_{i+1} - \overline{\phi}_{i+2}}{12}, \tag{3.50}$$

where $\overline{\phi}_i$ is the cell average of ϕ in the cell $\left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$, $\overline{\phi}_i = \mathrm{A}(\phi)_i = \frac{1}{\delta x_i}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \phi(\tilde{x})d\tilde{x}$ (Colella and Woodward, 1984). To increase the order of accuracy in smooth flows, Colella and Sekora (2008) suggested to use instead the sixth-order accurate formula

$$\phi_{i+\frac{1}{2}} = \frac{\overline{\phi}_{i-2} - 8\overline{\phi}_{i-1} + 37\overline{\phi}_i + 37\overline{\phi}_{i+1} - 8\overline{\phi}_{i+2} + \overline{\phi}_{i+3}}{60}. \tag{3.51}$$

This value must be limited to avoid oscillations near discontinuities in the solution. Colella and Sekora (2008) suggested the following procedure which preserves smooth extrema as well, which had not been the case for the original methods presented in Colella and Woodward (1984). The three approximations to the second derivative in $i + \frac{1}{2}$ are

$$\left(D^2\phi\right)_{i+\frac{1}{2},C} = \frac{3}{h^2}\left(\overline{\phi}_i - 2\phi_{i+\frac{1}{2}} + \overline{\phi}_{i+1}\right), \tag{3.52a}$$

$$\left(D^2\phi\right)_{i+\frac{1}{2},L} = \frac{1}{h^2}\left(\overline{\phi}_{i-1} - 2\overline{\phi}_i + \overline{\phi}_{i+1}\right), \tag{3.52b}$$

$$\left(D^2\phi\right)_{i+\frac{1}{2},R} = \frac{1}{h^2}\left(\overline{\phi}_i - 2\overline{\phi}_{i+1} + \overline{\phi}_{i+2}\right), \tag{3.52c}$$

where h is the (constant) grid spacing. $\phi_{i+\frac{1}{2}}$ is taken from equation (3.51). If the sign of all approximations coincides, we define

$$\begin{aligned}\left(D^2\phi\right)_{i+\frac{1}{2},\lim} = &\operatorname{sign}\left(\left(D^2\phi\right)_{i+\frac{1}{2},C}\right)\cdot \\ &\min\left(C_{\lim}\left|\left(D^2\phi\right)_{i+\frac{1}{2},L}\right|, C_{\lim}\left|\left(D^2\phi\right)_{i+\frac{1}{2},R}\right|, \left|\left(D^2\phi\right)_{i+\frac{1}{2},C}\right|\right).\end{aligned} \tag{3.53}$$

$C_{\lim} > 1$ is a constant which should be independent of the grid spacing. Colella and Sekora (2008) suggested to set it to 1.25. If the signs of $\left(D^2\phi\right)_{i+\frac{1}{2},C}$, $\left(D^2\phi\right)_{i+\frac{1}{2},L}$ and $\left(D^2\phi\right)_{i+\frac{1}{2},R}$ differ, $\left(D^2\phi\right)_{i+\frac{1}{2},\lim}$ is set to 0. The final approximation to $\phi_{i+\frac{1}{2}}$ is obtained by

$$\phi_{i+\frac{1}{2}} = \frac{1}{2}\left(\overline{\phi}_i + \overline{\phi}_{i+1} - \frac{h^2}{3}\left(D^2\phi\right)_{i+\frac{1}{2},\lim}\right). \tag{3.54}$$

The whole procedure is summarised in pseudo code in Algorithm 3.

Algorithm 3 Interpolation procedure from Colella and Sekora (2008).

$\phi_{i+\frac{1}{2}} = \frac{1}{60}\left(\overline{\phi}_{i-2} - 8\overline{\phi}_{i-1} + 37\overline{\phi}_i + 37\overline{\phi}_{i+1} - 8\overline{\phi}_{i+2} + \overline{\phi}_{i+3}\right)$
$\left(D^2\phi\right)_C = 3\left(\overline{\phi}_{i+1} - 2\phi_{i+\frac{1}{2}} + \overline{\phi}_i\right)$,
$\left(D^2\phi\right)_L = \left(\overline{\phi}_{i-1} - 2\overline{\phi}_i + \overline{\phi}_{i+1}\right)$, $\left(D^2\phi\right)_R = \left(\overline{\phi}_i - 2\overline{\phi}_{i+1} + \overline{\phi}_{i+2}\right)$
$\left(D^2\phi\right)_{\lim,L/R} = C_{\lim}\min\left(\left|\left(D^2\phi\right)_L\right|, \left|\left(D^2\phi\right)_R\right|\right)$
$\left(D^2\phi\right)_{\lim} = 0$
if $\left(D^2\phi\right)_C > 0$ & $\left(D^2\phi\right)_R > 0$ & $\left(D^2\phi\right)_L > 0$ **then**
    $\left(D^2\phi\right)_{\lim} = \min\left(\left|\left(D^2\phi\right)_C\right|, \left(D^2\phi\right)_{\lim,L/R}\right)$
end if
if $\left(D^2\phi\right)_C < 0$ & $\left(D^2\phi\right)_R < 0$ & $\left(D^2\phi\right)_L < 0$ **then**
    $\left(D^2\phi\right)_{\lim} = -\min\left(\left|\left(D^2\phi\right)_C\right|, \left(D^2\phi\right)_{\lim,L/R}\right)$
end if
$\phi_{i+\frac{1}{2}} = \frac{1}{2}\left(\overline{\phi}_i + \overline{\phi}_{i+1} - \frac{1}{3}\left(D^2\phi\right)_{\lim}\right)$

In the next step, we obtain left and right cell edge values $\phi_{i,-}$ and $\phi_{i,+}$ as input for the Riemann solver from the interpolated value $\phi_{i+\frac{1}{2}}$. If

$$\left(\phi_{i+\frac{1}{2}} - \overline{\phi}_i\right)\left(\overline{\phi}_i - \phi_{i-\frac{1}{2}}\right) \leq 0 \text{ and } \left(\overline{\phi}_{i+1} - \overline{\phi}_i\right)\left(\overline{\phi}_i - \overline{\phi}_{i-1}\right) \leq 0, \tag{3.55}$$

there is a local extremum at $i + \frac{1}{2}$. Defining the approximations

$$\left(D^2\phi\right)_{i+\frac{1}{2},P} = -\frac{2}{h^2}\left(6\overline{\phi}_i - 3\left(\phi_{j-\frac{1}{2}} + \phi_{j+\frac{1}{2}}\right)\right), \tag{3.56a}$$

$$\left(D^2\phi\right)_{i+\frac{1}{2},L} = \frac{1}{h^2}\left(\overline{\phi}_{i-2} - 2\overline{\phi}_{i-1} + \overline{\phi}_i\right), \tag{3.56b}$$

$$\left(D^2\phi\right)_{i+\frac{1}{2},C} = \frac{1}{h^2}\left(\overline{\phi}_{i-1} - 2\overline{\phi}_i + \overline{\phi}_{i+1}\right), \tag{3.56c}$$

$$\left(D^2\phi\right)_{i+\frac{1}{2},R} = \frac{1}{h^2}\left(\overline{\phi}_i - 2\overline{\phi}_{i+1} + \overline{\phi}_{i+2}\right), \tag{3.56d}$$

we proceed similarly as before. If the sign of all approximations coincides, we define

$$\begin{aligned}\left(D^2\phi\right)_{i+\frac{1}{2},\lim} = &\operatorname{sign}\left(\left(D^2\phi\right)_{i+\frac{1}{2},P}\right) \cdot \min\Big(\left|\left(D^2\phi\right)_{i+\frac{1}{2},P}\right|, \\ &C_{\lim}\left|\left(D^2\phi\right)_{i+\frac{1}{2},L}\right|, C_{\lim}\left|\left(D^2\phi\right)_{i+\frac{1}{2},R}\right|, C_{\lim}\left|\left(D^2\phi\right)_{i+\frac{1}{2},C}\right|\Big).\end{aligned} \tag{3.57}$$

$C_{\lim} > 1$ is set to 1.25 again. If the signs differ, $\left(D^2\phi\right)_{i+\frac{1}{2},\lim}$ is set to 0. Finally, $\phi_{i,\pm}$ is obtained by

$$\phi_{i,\pm} = \overline{\phi}_i + \left(\phi_{j\pm\frac{1}{2}} - \overline{\phi}_i\right)\frac{\left(D^2\phi\right)_{i+\frac{1}{2},\lim}}{\left(D^2\phi\right)_{i+\frac{1}{2},P}}. \tag{3.58}$$

If (3.55) is not fulfilled and

$$\left(\phi_{i+\frac{1}{2}} - \overline{\phi}_i\right)\left(\overline{\phi}_i - \phi_{i-\frac{1}{2}}\right) < 0, \tag{3.59}$$

we set

$$\phi_{i,+} = \phi_{i,-} = \overline{\phi}_i. \tag{3.60}$$

Otherwise, if $\left|\phi_{i\pm\frac{1}{2}} - \overline{\phi}_i\right| \geq 2\left|\phi_{i\mp\frac{1}{2}} - \overline{\phi}_i\right|$, we set

$$\phi_{i,\pm} = \overline{\phi}_i - 2\left(\phi_{i,\mp} - \overline{\phi}_i\right). \tag{3.61}$$

The whole procedure is summarised in pseudo code in Algorithm 4.

3.3.1 Time Integration

Definition 5. *Let a multdimensional problem be given. We call a time integration scheme split if the spatial integration is performed dimension-wise (in "sweeps") using*

Algorithm 4 Construction of cell edge values as in Colella and Sekora (2008).

if $\left(\phi_{i+\frac{1}{2}} - \overline{\phi}_i\right)\left(\overline{\phi}_i - \phi_{i-\frac{1}{2}}\right) \leq 0$ & $\left(\overline{\phi}_{i+1} - \overline{\phi}_i\right)\left(\overline{\phi}_i - \overline{\phi}_{i-1}\right) \leq 0$ **then**
  $\left(D^2\phi\right)_P = -2\left(6\overline{\phi}_i - 3\left(\phi_{j-\frac{1}{2}} + \phi_{j+\frac{1}{2}}\right)\right)$, $\left(D^2\phi\right)_C = \left(\overline{\phi}_{i-1} - 2\overline{\phi}_i + \overline{\phi}_{i+1}\right)$,
  $\left(D^2\phi\right)_L = \left(\overline{\phi}_{i-2} - 2\overline{\phi}_{i-1} + \overline{\phi}_i\right)$, $\left(D^2\phi\right)_R = \left(\overline{\phi}_i - 2\overline{\phi}_{i+1} + \overline{\phi}_{i+2}\right)$
  $\left(D^2\phi\right)_{\lim,C/L/R} = C_{\lim} \min\left(\left|\left(D^2\phi\right)_C\right|, \left|\left(D^2\phi\right)_L\right|, \left|\left(D^2\phi\right)_R\right|\right)$
  $\left(D^2\phi\right)_{\lim} = 0$
  if $\left(D^2\phi\right)_C > 0$ & $\left(D^2\phi\right)_R > 0$ & $\left(D^2\phi\right)_L > 0$ & $\left(D^2\phi\right)_P > 0$ **then**
    $\left(D^2\phi\right)_{\lim} = \min\left(\left|\left(D^2\phi\right)_P\right|, \left(D^2\phi\right)_{\lim,C/L/R}\right)$
  end if
  if $\left(D^2\phi\right)_C < 0$ & $\left(D^2\phi\right)_R < 0$ & $\left(D^2\phi\right)_L < 0$ & $\left(D^2\phi\right)_P < 0$ **then**
    $\left(D^2\phi\right)_{\lim} = -\min\left(\left|\left(D^2\phi\right)_P\right|, \left(D^2\phi\right)_{\lim,C/L/R}\right)$
  end if
  if $\left(D^2\phi\right)_P = 0$ **then**
    $\phi_{i,+} = \phi_{i,-} = \overline{\phi}_i$
  else
    $\phi_{i,\pm} = \overline{\phi}_i + \left(\phi_{j\pm\frac{1}{2}} - \overline{\phi}_i\right)\frac{\left(D^2\phi\right)_{\lim}}{\left(D^2\phi\right)_P}$
  end if
else
  if $\left(\phi_{i+\frac{1}{2}} - \overline{\phi}_i\right)\left(\overline{\phi}_i - \phi_{i-\frac{1}{2}}\right) < 0$ **then**
    $\phi_{i,+} = \phi_{i,-} = \overline{\phi}_i$
  else if $\left|\phi_{i\pm\frac{1}{2}} - \overline{\phi}_i\right| \geq 2\left|\phi_{i\mp\frac{1}{2}} - \overline{\phi}_i\right|$ **then**
    $\phi_{i,\pm} = \overline{\phi}_i - 2\left(\phi_{i,\mp} - \overline{\phi}_i\right)$
  end if
end if

the information from the previous sweep in the current time step. In an unsplit scheme, all fluxes are updated simultaneously.

Until now, we only investigated unsplit time integration schemes. On Cartesian grids, split schemes can lead to more stable and accurate schemes with the same computational costs as its unsplit counterparts.

Split Time Integration

To extend any one-dimensional integration scheme to higher dimensions, Warming and Beam (1976) suggested the first-order scheme

$$\mathbf{Q}^{n+1} = \mathrm{L}_x\mathrm{L}_y\mathbf{Q}^n, \tag{3.62}$$

and the second-order scheme

$$\mathbf{Q}^{n+2} = \mathrm{L}_x \mathrm{L}_y \mathrm{L}_y \mathrm{L}_x \mathbf{Q}^n. \tag{3.63}$$

Here, L_x means application of the one-dimensional scheme in x direction, and L_y application in y direction. When the latter scheme is combined with a suitable spatial reconstruction scheme, Courant numbers of 0.9 can be used, and the time integration is second-order accurate. The accuracy of the first scheme is restricted to first order no matter which spatial scheme is used. Nevertheless, it might be superior to unsplit schemes as we will show in the following paragraph.

We will exemplify the advantage of split time integration schemes for the two-dimensional advection equation

$$\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x} + \frac{\partial \phi}{\partial y} = 0 \tag{3.64}$$

using von Neumann analysis (Wesseling, 2001; LeVeque, 2007). The computational domain is $[0,1]^2$ with periodic boundary conditions. The equation is discretised on an equidistant grid with grid spacings $\delta x = \delta y$. We consider the first order accurate upwind scheme

$$\phi_{i,j}^{n+1} = \phi_{i,j}^n - \sigma \left(2\phi_{i,j}^n - \phi_{i-1,j}^n - \phi_{i,j-1}^n\right), \tag{3.65}$$

We denote the Courant number by $\sigma = \frac{\delta t}{\delta x}$. This scheme results from replacing the derivatives in (3.64) by one-sided finite differences,

$$\frac{\partial \phi}{\partial t} \approx \frac{\phi^{n+1} - \phi^n}{\delta t}, \quad \frac{\partial \phi}{\partial x} \approx \frac{\phi_{i,j} - \phi_{i-1,j}}{\delta x}, \quad \frac{\partial \phi}{\partial y} \approx \frac{\phi_{i,j} - \phi_{i,j-1}}{\delta y}. \tag{3.66}$$

Discretising equation (3.64) with upwind differences in space and the first order split scheme from Warming and Beam (1976) in time yields

$$\phi_{i,j}^\star = \phi_{i,j}^n - \sigma \left(\phi_{i,j}^n - \phi_{i,j-1}^n\right), \tag{3.67a}$$

$$\phi_{i,j}^{n+1} = \phi_{i,j}^\star - \sigma \left(\phi_{i,j}^\star - \phi_{i-1,j}^\star\right). \tag{3.67b}$$

Since the equation (3.64) is linear, the scheme is symmetric, and by rearranging terms, it can be brought in the one-step form

$$\begin{aligned} \phi_{i,j}^{n+1} = \phi_{i,j}^n &- \sigma \left(2\phi_{i,j}^n - \phi_{i-1,j}^n - \phi_{i,j-1}^n\right) \\ &+ \sigma^2 \left(\phi_{i,j}^n + \phi_{i-1,j-1}^n - \phi_{i,j-1}^n - \phi_{i-1,j}^n\right). \end{aligned} \tag{3.68}$$

In the von Neumann analysis, the Fourier transform is applied to the discretised equation (LeVeque, 2007). Writing $\hat{\phi}(\xi)$ for the Fourier transform at wave number ξ, the discretised equation is brought in the form

$$\hat{\phi}^{n+1}(\xi) = g(\xi)\hat{\phi}^{n}(\xi). \tag{3.69}$$

For the numerical solution to be stable, it is sufficient that the *amplification factor* $g(\xi)$ fulfils

$$|g(\xi)| \leq 1 + \alpha\,\delta t \tag{3.70}$$

for some α independent of ξ. Since

$$\frac{\partial \mathrm{e}^{\imath(x+y)\xi}}{\partial x} = \frac{\partial \mathrm{e}^{\imath(x+y)\xi}}{\partial y} = \imath\,\xi\,\mathrm{e}^{\imath(x+y)\xi}, \tag{3.71}$$

where $\imath$ is the imaginary unit, $\mathrm{e}^{\imath(x+y)\xi}$ is an Eigenfunction with Eigenvalue $\imath$ for both $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$.

We investigate how the schemes (3.65) and (3.68) work on the single wave number ξ by setting $\phi^n_{i,j} = \mathrm{e}^{\imath(i\,\delta x + j\,\delta y)\xi}$, $\delta x = \delta y$. Since we expect that

$$\phi^{n+1}_{i,j} = g(\xi)\phi^{n}_{i,j}, \tag{3.72}$$

we can calculate $g(\xi)$ for both schemes. For the upwind scheme,

$$g(\xi) = 1 - 2\sigma\left(1 - \mathrm{e}^{-\imath\,\xi\,\delta x}\right), \tag{3.73}$$

the absolute value of which is bound by 1 for all ξ as long as $\sigma \leq \frac{1}{2}$.

For the case of the split scheme (3.68), we calculate

$$\begin{aligned} g(\xi) &= 1 - \sigma\left(2 - 2\mathrm{e}^{-\imath\,\xi\,\delta x}\right) + \sigma^2\left(1 - 2\mathrm{e}^{-\imath\,\xi\,\delta x} + \mathrm{e}^{-2\imath\,\xi\,\delta x}\right) \\ &= 1 - 2\sigma\left(1 - \mathrm{e}^{-\imath\,\xi\,\delta x}\right) + \sigma^2\left(1 - \mathrm{e}^{-\imath\,\xi\,\delta x}\right)^2 \\ &= \left(1 - \sigma\left(1 - \mathrm{e}^{-\imath\,\xi\,\delta x}\right)\right)^2. \end{aligned}$$

Therefore, the absolute value of $g(\xi)$ is smaller than 1 for all ξ if $\sigma \leq 1$. We conclude even though both schemes are first order accurate, the stability of the split scheme is twice as large as the one of the unsplit scheme in terms of maximum allowed Courant numbers.

The split scheme (3.68) is identical to the first order *corner transport upwind* (CTU) scheme for the advection equation defined in Colella (1990). There, a second-order extension of the scheme to systems of conservation laws is introduced which we will present in the following paragraph.

The CTU Scheme

Even though the stability and accuracy of split time integration schemes is high, they restrict the applicability of the code. On the other hand, simple unsplit schemes like the first order upwind scheme (3.65) are not very stable. The *corner transport upwind* (CTU) scheme from Colella (1990) is an extension of simple unsplit schemes to second order with improved stability and direct applicability to systems of conservation laws. We will describe the scheme for the two-dimensional Euler equations

$$\frac{\partial}{\partial t}\mathbf{Q} + \frac{\partial}{\partial x}\mathbf{F} + \frac{\partial}{\partial y}\mathbf{G} = 0, \tag{3.74}$$

with the state vector $\mathbf{Q}$ and the flux functions $\mathbf{F}$ and $\mathbf{G}$ given by

$$\mathbf{Q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \ \mathbf{F}(\mathbf{Q}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (p+E)u \end{pmatrix}, \ \mathbf{G}(\mathbf{Q}) = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ (p+E)v \end{pmatrix}, \tag{3.75}$$

where the pressure $p = p(\rho, e)$ is given by an equation of state and $e = E - \frac{u^2+v^2}{2\rho}$ is the internal energy. In the following, we will write $\mathbf{u} := (u, v)^{\mathrm{T}}$ for the velocity vector. Furthermore, we define the vector of primitive variables

$$\mathbf{S} = \begin{pmatrix} \rho \\ u \\ v \\ p \end{pmatrix}, \text{ resp., in three dimensions, } \mathbf{S} = \begin{pmatrix} \rho \\ u \\ v \\ w \\ p \end{pmatrix}. \tag{3.76}$$

The CTU algorithm is a predictor–corrector scheme. Once predictions $\mathbf{Q}^{n+\frac{1}{2}}_{i\pm\frac{1}{2},j}$ and $\mathbf{Q}^{n+\frac{1}{2}}_{i,j\pm\frac{1}{2}}$ are constructed (either in the conservative variables $\mathbf{Q}$ or in the primitive variables $\mathbf{S}$), the values at the new time step for the Euler equations (3.74) are obtained by

$$\mathbf{Q}^{n+1}_{i,j} = \mathbf{Q}^{n}_{i,j} - \frac{\delta t}{\delta x}\left(\mathbf{F}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} - \mathbf{F}^{n+\frac{1}{2}}_{i-\frac{1}{2},j}\right) - \frac{\delta t}{\delta y}\left(\mathbf{G}^{n+\frac{1}{2}}_{i,j+\frac{1}{2}} - \mathbf{G}^{n+\frac{1}{2}}_{i,j-\frac{1}{2}}\right), \tag{3.77}$$

where $\mathbf{F}^{n+\frac{1}{2}}_{i\pm\frac{1}{2},j} = \mathbf{F}\left(\mathbf{Q}^{n+\frac{1}{2}}_{i\pm\frac{1}{2},j}\right)$ and $\mathbf{G}^{n+\frac{1}{2}}_{i,j\pm\frac{1}{2}} = \mathbf{G}\left(\mathbf{Q}^{n+\frac{1}{2}}_{i,j\pm\frac{1}{2}}\right)$. The construction of the predictive variables at time step $n+\frac{1}{2}$ starts with the extrapolation formula

$$\mathbf{Q}^{n+\frac{1}{2}}_{i\pm\frac{1}{2},j} = \mathbf{Q}^{n}_{i,j} \pm \frac{\delta x}{2}\frac{\partial \mathbf{Q}}{\partial x} + \frac{\delta t}{2}\frac{\partial \mathbf{Q}}{\partial t}. \tag{3.78}$$

From the Euler equations (3.74) we get

$$\frac{\partial \mathbf{Q}}{\partial t} = -\left(\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y}\right). \tag{3.79}$$

Introducing the linearisation of $\mathbf{F}$ in $\mathbf{Q}$

$$\mathbf{A}_x = \frac{\partial \mathbf{F}}{\partial \mathbf{Q}} \text{ such that } \frac{\partial \mathbf{F}}{\partial x} = \mathbf{A}_x \frac{\partial \mathbf{Q}}{\partial x}, \tag{3.80}$$

the equation (3.78) transforms to

$$\mathbf{Q}^{n+\frac{1}{2}}_{i\pm\frac{1}{2},j} = \mathbf{Q}^{n}_{i,j} + \left(\pm\frac{\delta x}{2} - \frac{\delta t}{2}\mathbf{A}_x \right) \frac{\partial \mathbf{Q}}{\partial x} - \frac{\delta t}{2}\frac{\partial \mathbf{G}}{\partial y}. \tag{3.81}$$

In Colella (1990), all terms on the right-hand side of (3.81) are evaluated at the cell centre in an appropriate manner. Instead of linearising in $\mathbf{Q}$, one could as well linearise in $\mathbf{S}$. In this way, Colella (1990) constructed a two-dimensional unsplit scheme of second order with high stability.

Saltzman (1994) presented the first extension of the CTU algorithm to three dimensions in the context of Riemann solvers (Toro, 2009). To reach the high stability of the two-dimensional algorithm, twelve Riemann problems had to be solved instead of three as required by a split or a simple unsplit scheme as, e.g., the upwind scheme. Gardiner and Stone (2008) modified the algorithm slightly and reduced the number of Riemann problems to six. The stability of the algorithm is still superior to any other unsplit scheme, but only half as good as the scheme by Saltzman (1994). In the following, we outline their algorithm for the three-dimensional Euler equations (3.1). Here, `PPM` denotes application of the PPM algorithm to reconstruct the primitive variables at the cell boundary as needed by the Riemann solver. `RIEMANN` means solution of the Riemann problem with the given boundary data.

Step 1 Calculate the left and right states and the associated interface flux with the standard PPM scheme.

$$\begin{aligned} \mathbf{S}^{\star}_{L/R,x,i+\frac{1}{2},j,k} &= \texttt{PPM}(\mathbf{Q}^{(n)}), \\ \mathbf{F}^{\star}_{i+\frac{1}{2},j,k} &= \texttt{RIEMANN}(\mathbf{S}^{\star}_{L,x,i+\frac{1}{2},j,k}, \mathbf{S}^{\star}_{R,x,i+\frac{1}{2},j,k}) \end{aligned} \tag{3.82}$$

Step 2 Update the states with the transversal fluxes to the intermediate time level.

$$\begin{aligned} \mathbf{Q}^{(n+\frac{1}{2})}_{L,x,i+\frac{1}{2},j,k} = \mathbf{Q}^{\star}_{L,x,i+\frac{1}{2},j,k} &- \frac{\delta t}{2\delta y}\left(\mathbf{G}^{\star}_{i,j+\frac{1}{2},k} - \mathbf{G}^{\star}_{i,j-\frac{1}{2},k} \right) \\ &- \frac{\delta t}{2\delta z}\left(\mathbf{H}^{\star}_{i,j,k+\frac{1}{2}} - \mathbf{H}^{\star}_{i,j,k-\frac{1}{2}} \right), \\ \mathbf{Q}^{(n+\frac{1}{2})}_{R,x,i+\frac{1}{2},j,k} = \mathbf{Q}^{\star}_{R,x,i+\frac{1}{2},j,k} &- \frac{\delta t}{2\delta y}\left(\mathbf{G}^{\star}_{i+1,j+\frac{1}{2},k} - \mathbf{G}^{\star}_{i+1,j-\frac{1}{2},k} \right) \\ &- \frac{\delta t}{2\delta z}\left(\mathbf{H}^{\star}_{i+1,j,k+\frac{1}{2}} - \mathbf{H}^{\star}_{i+1,j,k-\frac{1}{2}} \right) \end{aligned} \tag{3.83}$$

Here, $\mathbf{Q}^{\star}_{L/R,x,i+\frac{1}{2},j,k}$ and $\mathbf{Q}^{(n+\frac{1}{2})}_{L/R,x,i+\frac{1}{2},j,k}$ are the variables in conservation form corresponding to the primitive variables $\mathbf{S}^{\star}_{L/R,x,i+\frac{1}{2},j,k}$ and $\mathbf{S}^{(n+\frac{1}{2})}_{L/R,x,i+\frac{1}{2},j,k}$.

Step 3 Calculate the flux according to these updated states.

$$\mathbf{F}^{(n+\frac{1}{2})}_{i+\frac{1}{2},j,k} = \texttt{RIEMANN}\left(\mathbf{S}^{(n+\frac{1}{2})}_{L,x,i+\frac{1}{2},j,k}, \mathbf{S}^{(n+\frac{1}{2})}_{R,x,i+\frac{1}{2},j,k}\right) \tag{3.84}$$

Step 4 Update the conserved variables with the corrected fluxes.

$$\begin{aligned}\mathbf{Q}^{(n+1)}_{i,j,k} = \mathbf{Q}^{(n)}_{i,j,k} &- \frac{\delta t}{\delta x}\left(\mathbf{F}^{(n+\frac{1}{2})}_{i+\frac{1}{2},j,k} - \mathbf{F}^{(n+\frac{1}{2})}_{i-\frac{1}{2},j,k}\right) \\ &- \frac{\delta t}{\delta y}\left(\mathbf{G}^{(n+\frac{1}{2})}_{i,j+\frac{1}{2},k} - \mathbf{G}^{(n+\frac{1}{2})}_{i,j-\frac{1}{2},k}\right) - \frac{\delta t}{\delta z}\left(\mathbf{H}^{(n+\frac{1}{2})}_{i,j,k+\frac{1}{2}} - \mathbf{H}^{(n+\frac{1}{2})}_{i,j,k-\frac{1}{2}}\right)\end{aligned} \tag{3.85}$$

We immediately see that twice as many Riemann problems have to be solved than in the unsplit scheme. Furthermore, as any unsplit scheme the algorithm leads to an increase in memory consumption compared to a split scheme since the fluxes must be saved in several three-dimensional arrays for the simultaneous update. In step (3.83), the cell boundary states must either be recalculated or saved in step (3.82), leading to either an increase in total computation time or an increase in memory requirements.

4 Composite Grids

We have seen in Section 2 that Cartesian and spherical coordinate systems are the most common choices in numerical astrophysics. Most algorithms work only on these types of grids. Nevertheless, these coordinate systems have several disadvantages which restrict their applicability.

For the case that the simulation domain is a full sphere, spherical coordinate systems cannot cover the whole domain due to the grid singularities at the centre and at the poles. With a Cartesian grid, a huge part of the computational resources is wasted.

A straightforward way to extend the capabilities of a code written for spherical or Cartesian coordinate systems is to use several grids and cover the computational domain in this way. This is called the *composite grid* approach. The grids can either overlap (Chesshire and Henshaw, 1990) or be patched together (Rai, 1986). Allowing overlapping grids makes the use of standard grids for rather complex domains possible while increasing computational costs and complicating the interpolation at the boundary. On the other hand, patched grids decrease the flexibility of the code again. E.g., a Cartesian and a spherical grid can only be patched together in special situations.

In this section, we will investigate the advantages and disadvantages of the overlapping grid or *Chimera grid* approach (Ferziger and Perić, 2002). We start with a famous example, the *Yin–Yang grid*, before commenting on the numerical difficulties of this method.

4.1 The Yin–Yang Grid

The domain of interest of many meteorological computational problems, but also of mantle convection in geophysics and supernova explosions, is a spherical shell (e.g., Washington et al., 2009; Wongwathanarat et al., 2010). A spherical coordinate system implies very small time steps due to convergence of the grid lines near the poles when finite difference or finite volume schemes with explicit time integration are used. Nevertheless, the geometry of the problem is quite regular such that unstructured grids are not suited as well. Instead, the *cubed sphere* approach (Sadourny, 1972; Ronchi et al., 1996) and the *Yin–Yang* grid (Kageyama and Sato, 2004) were introduced. We will focus on the second approach in this section.

The basic idea of the Yin–Yang grid is to use two spherical grids with low latitudes. By rotating one of them, their union covers the whole shell, but each grid has quasi-uniform grid spacings. Wongwathanarat et al. (2010) report that a factor 26 for 3° to 80 for 1° angular resolution in computation time can be saved when using the Yin–Yang grid instead of a conventional spherical coordinate system.

The computational efficiency of the Yin–Yang grid is much better than that of any spherical coordinate system, and it is a good choice as long as the centre of the grid is not of interest. The singularity at the centre is not removed by this approach. Furthermore, at the boundary some kind of interpolation must be performed. This brings new difficulties in the algorithm as described in the next paragraph.

4.1.1 Conservation Problem

At the boundary of the two grids, some interpolation procedure must be implemented. Kageyama and Sato (2004) used bilinear interpolation of the variables from the one grid as boundary conditions for the other. But Peng et al. (2006) show that polynomial interpolation of variables does not lead to conservation at the grid boundaries. Instead, they suggest a simple flux interpolation formula which leads to conservation for scalar variables. Wongwathanarat et al. (2010) describe in much detail that this approach does not work for vector variables, such as the velocity vector, which change their orientation in every grid cell. This is the case, e.g., for the velocity vector $\mathbf{u} = (u_r, u_\phi, u_\theta)^T$ in spherical coordinate systems. Anyway, after measuring the conservation error they found it to be small and to decrease with grid spacing.

Pärt-Enander and Sjögreen (1994) describe why no interpolation scheme based on interpolation of variables at grid boundaries can be conservative. Following their description, we show that any method based on interpolation or reconstruction of variables is not conservative. Simplifying as much as possible, we assume that two one-dimensional grids with constant grid spacings δx_1 and δx_2 are given. Slightly modifying the notation from Pärt-Enander and Sjögreen (1994), the left grid G_1 is defined on $(-\infty, b]$ and the right grid G_2 on $[c, \infty)$. The cell boundaries $y_{j+1/2}$ on the left grid are given by $y_{j+1/2} = b-(N_1-j)\,\delta x_1$, $j = N_1, N_1-1, \ldots$, where N_1 is the number of grid points on the left grid. The cell boundaries $x_{j-1/2}$ on the right grid are defined by $x_{j-1/2} = c + j\,\delta x_2$, $j = 1, 2, \ldots$ The overlap is defined by

$$\begin{aligned} d &= x_{1/2} - y_{q-1/2} > 0 \\ \text{where } q &= \max\{j:\ x_{1/2} - y_{j-1/2} > 0,\ y_{j-1/2} \in G_1,\ x_{1/2} \in G_2\}. \end{aligned} \tag{4.1}$$

We choose b and c such that $q < N_1$. We want to solve a one-dimensional conservation law of the form

$$\frac{\partial \phi}{\partial t} + \frac{\partial F(\phi)}{\partial x} = 0,\ \phi(x,0) = \phi_0(x),\ x \in (-\infty, \infty). \tag{4.2}$$

At time step t_n, the numerical solutions v_j^n on G_1 and u_j^n on G_2 are defined at the cell centres $y_j = \frac{1}{2}(y_{j-1/2} + y_{j+1/2})$ and $x_j = \frac{1}{2}(x_{j-1/2} + x_{j+1/2})$. They are advanced by a conservative finite difference scheme of the form

$$\begin{aligned} v_j^{n+1} &= v_j^n - \frac{\delta t}{\delta x_1}\left(f_{j+1/2} - f_{j-1/2}\right), \\ u_j^{n+1} &= u_j^n - \frac{\delta t}{\delta x_2}\left(g_{j+1/2} - g_{j-1/2}\right), \end{aligned} \tag{4.3}$$

where f and g are the numerical fluxes given by the scheme used for spatial discretisation. We assume that at time step n, sufficient boundary data is given to compute all fluxes to update v_j^{n+1}, $j = N_1, N_1 - 1, \ldots$, and u_j^{n+1}, $j = 1, 2, \ldots$. Boundary conditions must be specified at $N_1 + 1$ for G_1 and at 0 for G_2. The *total mass* of ϕ at time t_n is defined by

$$I^n = \sum_{j=-\infty}^{q-1} \delta x_1 v_j^n + d v_q^n + \sum_{j=1}^{\infty} \delta x_2 u_j^n. \tag{4.4}$$

We call a method *conservative*, if $I^{n+1} = I^n$. Inserting (4.3) into I^{n+1}, we arrive at

$$\begin{aligned} \frac{I^{n+1} - I^n}{\delta t} &= -\sum_{j=-\infty}^{q-1}\left(f_{j+1/2} - f_{j-1/2}\right) - \frac{d}{\delta x_1}\left(f_{q+1/2} - f_{q-1/2}\right) \\ &\quad - \sum_{j=1}^{\infty}\left(g_{j+1/2} - g_{j-1/2}\right) = g_{1/2} - \left(\left(1 - \frac{d}{\delta x_1}\right) f_{q-1/2} + \frac{d}{\delta x_1} f_{q+1/2}\right). \end{aligned} \tag{4.5}$$

We conclude that the method is conservative if and only if the leftmost numerical flux $g_{1/2}$ on G_2 is set by

$$g_{1/2} = \left(1 - \frac{d}{\delta x_1}\right) f_{q-1/2} + \frac{d}{\delta x_1} f_{q+1/2}. \tag{4.6}$$

This is exactly the flux interpolation formula by Berger (1987). Boundary conditions on the variables, no matter whether they use interpolation or reconstruction principles, do not lead to conservation at the grid boundary (Pärt-Enander and Sjögreen, 1994).

4.2 Boundary Interpolation Methods

As outlined in the previous paragraph, Pärt-Enander and Sjögreen (1994) showed that only interpolation of numerical fluxes at the boundary leads to conservation at the grid boundary. Other approaches based either on polynomial interpolation or on integration of a reconstruction of the cell values do not conserve mass since the numerical fluxes obtained in this way do not cancel out exactly. Non-conservation leads to displacements when shock fronts cross the grid interface. On the other hand, the conservative formula of Berger (1987) is significantly less stable. Both methods produce numerical noise at the interface, in particular at low speeds.

To remove the noise Pärt-Enander and Sjögreen (1994) suggested a non-linear filtering procedure, and to increase the stability of the flux interpolation methods they recommend switching to the characteristic variables at an outflow boundary. All of these methods are necessary, but complicated to implement.

4.2.1 Conservative Approach

The idea of interpolating numerical fluxes at grid interfaces was introduced by Berger (1987) for a variety of one- and two-dimensional special cases as, e.g., abrupt change in spatial or temporal grid resolution and overlapping grids. Whereas the method is rather simple to implement in one spatial dimension, it gets much more complicated when going to two or three spatial dimensions. The method is based on linear interpolation and therefore limits the overall order of the scheme to two. Furthermore, the method decreases the stability of the numerical scheme (Pärt-Enander and Sjögreen, 1994) and application to vector variables, which change their orientation from cell to cell, is unclear (Wongwathanarat et al., 2010).

4.2.2 High-Order Approach

In the context of high-order finite difference schemes, Sebastian and Shu (2003) presented a different approach based on high order interpolation of the variables. They claimed that by using high order Lagrangian or WENO–type interpolation, the conservation errors are very small and decrease with grid spacing. In contrast to the method of Berger (1987), the overall order of the scheme is not restricted by the grid interpolation procedure.

We outline the approach of Sebastian and Shu (2003) in the following. The domain of interest is covered with several overlapping grids. On each of these grids, the WENO finite difference scheme (Shu and Osher, 1988; Shu, 2003) is used. The overlap of the grids is large enough such that enough boundary points can be obtained from the other grid by some method solving the following

Problem 3 (interpolation problem)**.** *Given a set of point values, approximate the value of the underlying function at a position between these cells to high order of accuracy.*

Problem 3 is closely related to Problem 1. Therefore, Sebastian and Shu (2003) suggested a procedure similar to the WENO reconstruction algorithm as described in paragraph 3.1.2 to solve the interpolation problem. We describe the WENO interpolation method in the following paragraph. Of course, Problem 3 can also be solved with standard interpolation methods as Lagrange interpolation. Sebastian and Shu (2003) found that both Lagrange and WENO–type interpolation succeed in obtaining high order of accuracy and keeping the conservation errors small. Due to the simplicity of the Lagrange procedure and since it did not show any adverse effects in the presence of shocks, they prefer Lagrange interpolation.

As described in Carpenter et al. (1995) and Fornberg (1998), time-dependent boundary conditions can limit the order of accuracy of a Runge–Kutta scheme when not

implemented correctly. In Table G.1-4. in Fornberg (1998), an overview can be found how the method of implementing time-dependent boundary conditions influences the order of accuracy of the solution. Interpolation at grid boundaries can be understood as using time-dependent boundary conditions on every domain, and limits the order of accuracy to two. Nevertheless, as long as the error at the boundary is small compared to the error in the inner part of each domain, the overall order is not affected adversely. However, this theoretical limitation should be kept in mind when designing high-order interpolation methods at grid interfaces.

WENO–type Interpolation Algorithm

In the following, the WENO–type interpolation operator is derived following Sebastian and Shu (2003). The purpose of the interpolation operator is to solve Problem 3, i.e. interpolate the value of the underlying function at a certain position given a set of point values.

We will only consider equidistant one-dimensional grids and interpolation of the value in the cell $\left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$. The idea of the WENO interpolation process is to use several stencils in the neighbourhood of the point to be interpolated to. On each of the stencils, an interpolating polynomial of high order is defined. The interpolated value is obtained by summing these polynomials weighting them according to their smoothness. If a discontinuity is contained in the stencil of a polynomial, its weight will be very small thereby avoiding oscillatory behaviour as it is common when using high order interpolation.

Assume that the point values ϕ_i of the function ϕ are given and we want to interpolate the value of ϕ at $x_0 \in \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$. We consider k stencils

$$S_r(i) = \{x_{i-r}, \ldots, x_{i-r+k-1}\},\ r = 0, \ldots, k-1. \tag{4.7}$$

On each stencil $S_r(i)$ a polynomial p_r of degree $k-1$ is defined by Lagrange interpolation,

$$p_r(x) = \sum_{j=0}^{k-1} \phi_{i-r+j} c_{rj}(x),\ c_{rj}(x) = \prod_{l=0, l\neq j}^{k-1} \frac{x - x_{i-r+l}}{x_{i-r+j} - x_{i-r+l}}. \tag{4.8}$$

The interpolation polynomial P of degree $2k-1$ on the $2k-1$ points $x_{i-k+1}, \ldots, x_{i+k-1}$ can be obtained in the same way. The *linear weights* $d_r(x)$ are defined uniquely by the relation

$$P(x) = \sum_{r=0}^{k-1} d_r(x) p_r(x),\ \text{with}\ \sum_{r=0}^{k-1} d_r(x) = 1. \tag{4.9}$$

The values of c_{rj} and d_r for interpolation at $x_{i-\frac{1}{2}}$ are given in Table 4.1 for the case of an equidistant grid and $k = 3$. We note that for this case, the linear weights can be obtained by

$$d_0(x) = c^5_{2,4}(x),\ d_2(x) = c^5_{2,0}(x),\ d_1(x) = 1 - d_0(x) - d_2(x), \tag{4.10}$$

where $c^5_{rj}(x)$ are the coefficients of the Lagrange interpolation polynomial of degree $k = 5$. Similar formulae are available (but more complicated) for any other value of k.

c_{rj}	$j=0$	$j=1$	$j=2$	d_r
$r=0$	$\frac{15}{8}$	$-\frac{5}{4}$	$\frac{3}{8}$	$\frac{5}{16}$
$r=1$	$\frac{3}{8}$	$\frac{3}{4}$	$-\frac{1}{8}$	$\frac{1}{16}$
$r=2$	$-\frac{1}{8}$	$\frac{3}{4}$	$\frac{3}{8}$	$\frac{5}{8}$

Table 4.1: The interpolation constants $c_{rj}(x)$ and $d_r(x)$ as defined in (4.8) for $x = x_{i-\frac{1}{2}}$ and $k = 3$ on an equidistant grid.

High-order polynomial interpolation is known to produce oscillatory results. To avoid oscillations in the WENO approach, a convex combination of all candidate stencils p_r is used to compute $\phi(x_0)$. This procedure leads to non-oscillatory approximations of order $2k-1$, where k is the width of each of the stencils $S_r(i)$. We set $k = 3$ in the following.

Therefore, the approximation to $\phi(x_0)$, $x_0 \in \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$, is calculated by

$$\phi(x_0) = \omega_0 p_0(x_0) + \omega_1 p_1(x_0) + \omega_2 p_2(x_0), \tag{4.11}$$

where ω_0, ω_1 and ω_2 are nonlinear weights comparing the smoothness of the interpolation polynomials. To measure the smoothness of each interpolation polynomial p_r, we calculate

$$\beta_r(x) = \sum_{l=1}^{2} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \delta x^{2l-1} \left(\frac{d^l\, p_r(x)}{dx^l}\right)^2 dx,\ x \in \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]. \tag{4.12}$$

This results in the smoothness indicators

$$\begin{aligned}
\beta_0 &= \frac{1}{3}\left(10\phi_i^2 - 31\phi_i\phi_{i+1} + 25\phi_{i+1}^2 + 11\phi_i\phi_{i+2} - 19\phi_{i+1}\phi_{i+2} + 4\phi_{i+2}^2\right), \\
\beta_1 &= \frac{1}{3}\left(4\phi_{i-1}^2 - 13\phi_{i-1}\phi_i + 13\phi_i^2 + 5\phi_{i-1}\phi_{i+1} - 13\phi_i\phi_{i+1} + 4\phi_{i+1}^2\right), \\
\beta_2 &= \frac{1}{3}\left(4\phi_{i-2}^2 - 19\phi_{i-2}\phi_{i-1} + 25\phi_{i-1}^2 + 11\phi_{i-2}\phi_i - 31\phi_{i-1}\phi_i + 10\phi_i^2\right).
\end{aligned} \tag{4.13}$$

Then,

$$\tilde{\omega}_0 = \frac{d_0(x)}{(\beta_0+\epsilon)^2},\ \tilde{\omega}_1 = \frac{d_1(x)}{(\beta_1+\epsilon)^2},\ \tilde{\omega}_2 = \frac{d_2(x)}{(\beta_2+\epsilon)^2}, \tag{4.14}$$

with the linear weights d_r defined by (4.9). ϵ is a small parameter which is used to avoid division by zero. Finally,

$$\omega_0 = \frac{\tilde{\omega}_0}{\tilde{\omega}_0 + \tilde{\omega}_1 + \tilde{\omega}_2},\ \omega_1 = \frac{\tilde{\omega}_1}{\tilde{\omega}_0 + \tilde{\omega}_1 + \tilde{\omega}_2},\ \omega_2 = \frac{\tilde{\omega}_2}{\tilde{\omega}_0 + \tilde{\omega}_1 + \tilde{\omega}_2}. \tag{4.15}$$

Macdonald and Ruuth (2008) remark that with the choice of interpolation stencils from Sebastian and Shu (2003), one of the candidate stencils corresponds to an extrapolation rather than an interpolation. Since the choice of stencils is quite arbitrary, they suggested a different scheme where all stencils correspond to interpolations. The effect is a secondary one and not important for our purposes.

We emphasize that the difference of this algorithm compared to the reconstruction algorithm presented in paragraph 3.1.2 is that the interpolation polynomials agree with the point values of ϕ in each grid point, and not the cell averages over each computational cell.

A straightforward alternative approach to discretising the conservation law (4.2) in space is replacing the space derivative by a central difference, i.e.

$$\frac{\partial F(\phi)}{\partial x} \approx \frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{\delta x_i}. \tag{4.16}$$

For an equidistant grid, this approximation is of second order. Interpreting all values of ϕ and F as point values allows to apply the WENO interpolation procedure to calculate $F(\phi(x_{i+\frac{1}{2}})) = F_{i+\frac{1}{2}}$ from the point values $F(\phi(x_i)) = F_i$ to high order. Conservation is obtained by using the flux-based form (4.16). Nevertheless, the overall method is only of second order for equidistant grids, but extension to non-equidistant grids is straightforward.

Finally, we show that WENO–type interpolation is non-linear.

Definition 6. *An interpolation operator I is* **linear**, *if*

$$I(\lambda f) = \lambda I(f) \tag{4.17}$$

for a constant λ and given data f.

Collorary 1. *Polynomial interpolation is linear.*

Proof. For a given set of data points (x_i, f_i), the Lagrange form of the interpolation polynomial p is

$$p(x) = \sum_{i=0}^{n} f_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \tag{4.18}$$

For the data $(x_i, \lambda\, f_i + \mu\, g_i)$, $\lambda, \mu \in \mathbb{R}$,

$$
\begin{aligned}
p_{\lambda f+\mu g}(x) &= \sum_{i=0}^{n} (\lambda f_i + \mu g_i) \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \\
&= \lambda \sum_{i=0}^{n} f_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} + \mu \sum_{i=0}^{n} g_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \\
&= \lambda\, p_f(x) + \mu\, p_g(x).
\end{aligned}
\tag{4.19}
$$

□

Collorary 2. *WENO interpolation is non-linear.*

Proof. The WENO interpolation polynomial of order k is of the form

$$
p_{\mathrm{WENO}}(x) = \sum_{r=0}^{k} \omega_r(x) p^{(r)}(x). \tag{4.20}
$$

The weights $\omega_r(x)$ do not depend linearly on the interpolation data. Therefore, the whole interpolation procedure is non-linear. □

4.2.3 Numerical Experiments

We test the procedures presented in the previous section for several conservative finite difference methods. We solve the one-dimensional conservation law as defined in (4.2) on a finite interval $[a, b]$ with periodic boundary conditions. For the first-order *upwind method* (Strikwerda, 1989), the discretised version of (4.2) is then

$$
\phi_j^{n+1} = \phi_j^n - \frac{\delta t}{\delta x} \left(F(\phi)_j^n - F(\phi)_{j-1}^n \right). \tag{4.21}
$$

assuming $\frac{\partial F}{\partial \phi} > 0$. The *Lax-Wendroff method* (Strikwerda, 1989) is a second-order method using the derivative $\frac{\partial F}{\partial \phi}$ in calculating the spatial derivative. It takes the form

$$
\begin{aligned}
\phi_j^{n+1} = \phi_j^n &- \frac{\delta t}{2\,\delta x} \left(F(\phi)_{j+1}^n - F(\phi)_{j-1}^{n+1} \right) \\
&+ \frac{\delta t^2}{2\,\delta x^2} \left(A_{j+\frac{1}{2}} \left(F(\phi)_{j+1}^n - F(\phi)_j^n \right) - A_{j-\frac{1}{2}} \left(F(\phi)_j^n - F(\phi)_{j-1}^n \right) \right),
\end{aligned}
\tag{4.22}
$$

where $A_{j\pm\frac{1}{2}}$ is the derivative of F evaluated at $\frac{\phi_j + \phi_{j\pm1}}{2}$.

Furthermore, we solve the equation with the fifth order finite difference WENO scheme (Shu and Osher, 1988; Shu, 2003) and the variant where we replaced the reconstruction by an interpolation as described in paragraph 4.2.2. For the spatial discretisation, we use the third-order three-stage TVD3 scheme from Shu and Osher (1988). We expect the scheme using the reconstruction algorithm of fifth order as outlined in paragraph 3.1.2

to be of third order since the time integration is third order, and the scheme using the WENO–type interpolation algorithm to be second order accurate. The methods used in this section are summarised in Table 4.2.

scheme	order	$\frac{\partial \phi}{\partial t}$	$H_{j+\frac{1}{2}}$
upwind	1	$\frac{\phi_j^{n+1}-\phi_j^n}{\delta t}$	ϕ_j
Lax–Wendroff	2	$\frac{\phi_j^{n+1}-\phi_j^n}{\delta t}$	$\frac{F_{j+1}+F_j}{2} - \frac{\delta t}{2\,\delta x}\left(A_{j+\frac{1}{2}} \cdot (F_{j+1} - F_j)\right)$
WENO5 interpolation	2	TVD3	WENO5 interpolation of $F(\phi)$
WENO5 reconstruction	3	TVD3	WENO5 reconstruction of $F(\phi)$

Table 4.2: The schemes used in Section 4.2.3 to solve the one-dimensional conservation law (4.2). The equation is first semi-discretised in space and brought in conservation form $\frac{\partial \phi}{\partial t} + \frac{1}{\delta x}(H_{j+\frac{1}{2}} - H_{j-\frac{1}{2}}) = 0$. The table shows how the time derivative $\frac{\partial \phi}{\partial t}$ and the value of the numerical flux function at the cell boundary $H_{j+\frac{1}{2}}$ is obtained. The derivative $A_{j\pm\frac{1}{2}}$ is defined by $A_{j\pm\frac{1}{2}} = \frac{\partial F}{\partial \phi}|_{\frac{1}{2}(\phi_j+\phi_{j\pm1})}$. The WENO reconstruction scheme is described in paragraph 3.1.2, and the interpolation scheme in paragraph 4.2.2. The coefficients for the third-order three-stage TVD3 scheme from Shu and Osher (1988) can be found in Tables 3.1 and 3.2. The order corresponds to the order of the overall scheme, i.e. the minimum of the order of spatial and temporal discretisation.

In all of the following tests, we choose $F(\phi) = \phi$ such that the conservation law (4.2) is just the one-dimensional advection equation, and $\frac{\partial F}{\partial \phi} = 1 > 0$. Then, we consider the three initial conditions

$$\phi_0(x) = \begin{cases} 1 + 5\exp\left(-\frac{1}{(2-4x)^2} - \frac{1}{(4x)^2}\right), & 0.0 < x < 0.5 \\ 1, & \text{else} \end{cases}, \tag{4.23a}$$

$$\phi_0(x) = 1.1 + 0.1\sin(\pi x), \tag{4.23b}$$

$$\phi_0(x) = \begin{cases} 1, & 0.1 < x < 0.3 \\ 0, & \text{else} \end{cases}. \tag{4.23c}$$

We call these tests the "hill", the "sine wave" and the "step" initial condition in the following. They are shown in Figure 4.1. The computational domain is $[0, 2]$ with periodic boundary conditions. We discretise the domain on the two grids G_1 and G_2 with constant grid spacing δx_1 and δx_2. This implies that there are two grid interfaces, at 0 and at 1.1. The number of grid points N_1 on G_1 and N_2 on G_2 as well as the constant grid spacings δx_1 and δx_2 are given by

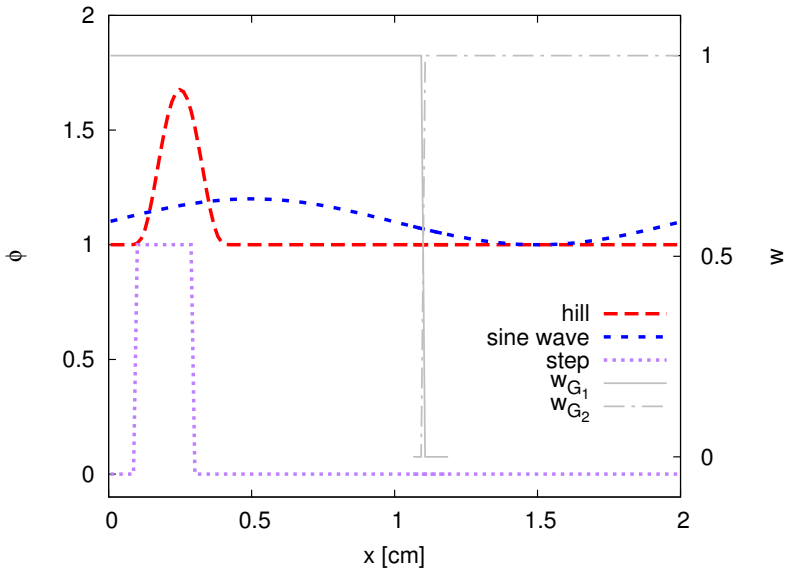

Figure 4.1: Initial conditions as defined in equations (4.23), and cell weights w on both grids G_1 and G_2 used in the calculation of conserved quantities. 82 cells are used on G_1 and 64 cells on G_2.

$$N_1 = 2^i + 3 \cdot i,\ \delta x_1 = \frac{1.1}{N_1},\ N_2 = 2^i,\ \delta x_2 = \frac{0.9}{N_2}. \tag{4.24}$$

Then, we add grid points to both grids such that they overlap at 0 and at 1.1 sufficiently wide to provide boundary information to the other grid by each of the interpolation methods described before.

We evaluate the error of the numerical solution by comparing to the analytical solution given by

$$\phi(x,t) = \phi_0(x-t). \tag{4.25}$$

We choose $t = 2\,\mathrm{s}$ for the sine wave and the step test, and $t = 1\,\mathrm{s}$ for the hill test as final point in time. The Courant number $\sigma = \frac{\delta t}{\delta x}$ is set to 0.1 in all calculations in this section.

In order to calculate the sum of ϕ over the whole computational domain, we have to assign a weight w to each grid cell. In regions which are not overlapped, the weight is 1. We denote by $q_1 < N_1$ the first node of G_1 at the grid interface at 1.1 the computational cell of which overlaps with G_2, and $q_2 < N_2$ the corresponding point of G_2 at the grid interface at 0. The weight of q_1 and q_2 is given by the ratio of the non-overlapped region to the total cell length. We assign $w = 0$ to all other cells. For the case of 82 cells on G_1 and 64 cells on G_2, the weights are shown in Figure 4.1.

Hill Initial Condition

We show results for the case of the hill test on 47 and 32 grid points in Figure 4.2. Linear interpolation is used at the grid interface, but the outcome does not change significantly with different interpolation methods. We observe that the upwind method damps the initial conditions strongly. The Lax–Wendroff method, on the other hand, produces oscillatory results. Only the WENO5 scheme gives an accurate solution at this resolution.

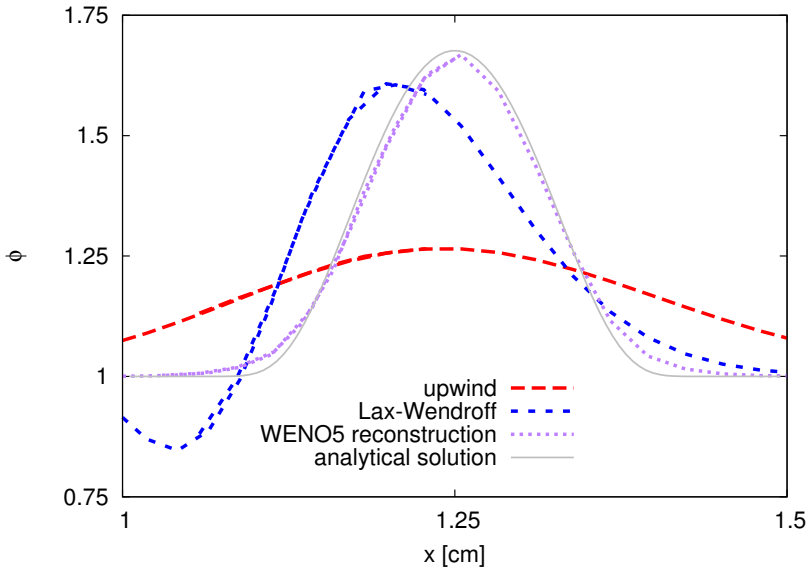

Figure 4.2: Numerical solutions for the hill test after 1 s calculated with the methods described in Table 4.2. 47 grid points are used on G_1 and 32 on G_1. Grid interfaces are located at 0 and 1.1. Linear interpolation without flux correction was used for boundary interpolation, but the influence of the interpolation method is small compared to the effect of the numerical scheme. The solution on each grid is drawn with its own line, therefore there is some overlap.

In the following we discuss the error decay in the L^2 norm for the initial conditions defined in equations (4.23). The grid spacing shown on the x axis is the grid spacing on G_2. We test linear interpolation, parabolic interpolation with and without the flux correction by Berger (1987) and Pärt-Enander and Sjögreen (1994), and WENO–type interpolation as described in paragraph 4.2.2 and Sebastian and Shu (2003).

In Figure 4.3, the error decay for the hill test is shown. We observe that the convergence of the first-order upwind, the second-order Lax–Wendroff method and the second-order WENO–interpolation method is not affected by the boundary interpolation method. Only for the high order accurate WENO–reconstruction method, the error decay changes depending on the boundary interpolation method. We observe that the decay is reduced to second order for the linear interpolation and when the flux correction

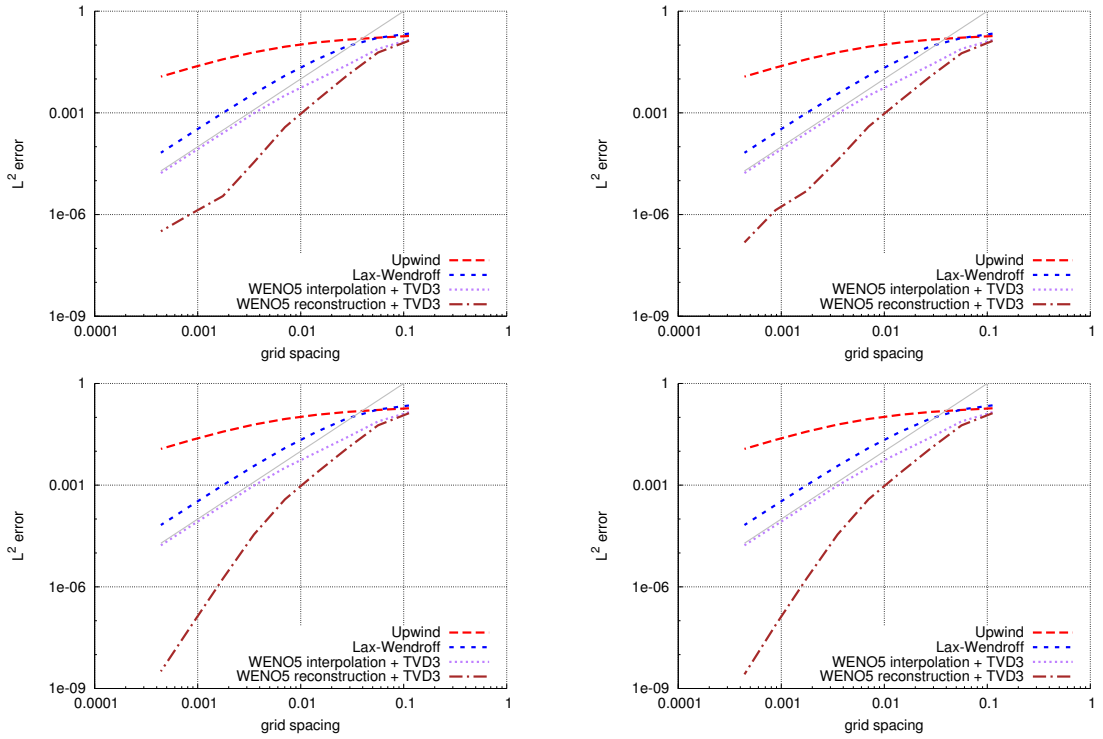

Figure 4.3: Error decay for the hill test with several boundary interpolation methods and numerical schemes as defined in Table 4.2. From top left to bottom right: linear interpolation, parabolic interpolation with flux correction, parabolic interpolation without flux correction, WENO5 interpolation. The grey line indicates second-order convergence.

is used. Much smaller errors can be reached with the parabolic and the WENO–type interpolation.

Comparing the effect of boundary interpolation for the WENO5 reconstruction scheme isolated in Figure 4.4, we conclude that the higher complexity and computational costs of the WENO–type interpolation do not pay off in higher accuracy. On the contrary, for very fine resolutions the error of the parabolic interpolation is slightly smaller. For coarser resolutions, the differences are much smaller. The overall error is dominated by the errors due to the spatial and temporal discretisation, and the influence of the boundary interpolation method is small.

Next we turn to the conservation error. As shown in Figure 4.5, the total "mass" of ϕ is conserved exactly with the flux correction mechanism by Berger (1987). The size of the conservation error is of similar size for all other boundary interpolation methods. Its absolute size depends on the numerical scheme and the test problem, therefore we refrain from giving absolute numbers. Instead, we observe that the conservation error is smaller the higher the accuracy of the method is, and that it decreases with grid

spacing for all tests and numerical schemes. With a highly accurate scheme such as the WENO5 scheme by Shu and Osher (1988) its magnitude will be very small, but it will never vanish completely unless the method by Berger (1987) is used which decreases the accuracy and stability of the method again.

Finally, we note that even though the WENO–interpolation method is only second-order accurate, its errors are smaller by about one magnitude compared to the second-order Lax–Wendroff scheme. This indicates that the interpolation method can yield accurate results efficiently as long as the resolution is rather coarse.

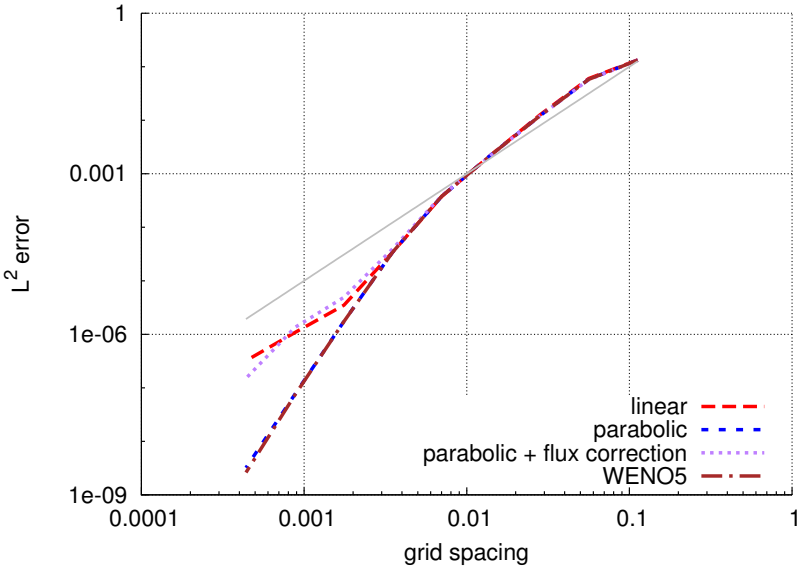

Figure 4.4: Error decay for the hill test with several boundary interpolation methods and the fifth order WENO scheme. Time integration is done with TVD3. The grey line indicates second-order convergence.

Sine Wave Initial Condition

Figures 4.6 and 4.7 for the sine wave test lead to similar conclusions. With the linear interpolation and when using the flux correction, the overall order is restricted to two, which mainly affects the accuracy of the WENO–reconstruction method. We loose several order of magnitude in accuracy compared to WENO–type and parabolic interpolation. The errors when using the WENO–type interpolation at the boundary and when using parabolic interpolation are of comparable size. For very high resolutions, they are limited by the accumulation of rounding errors. Again, the WENO–interpolation method is superior compared to Lax–Wendroff.

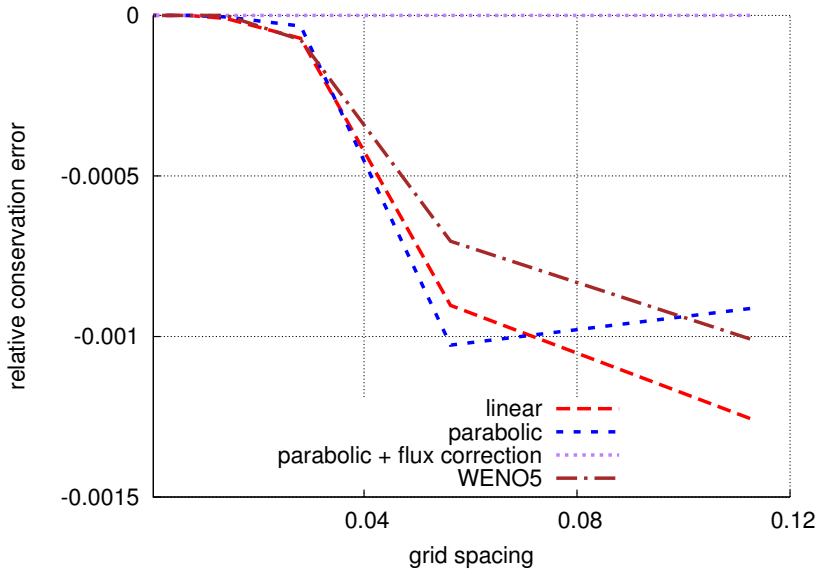

Figure 4.5: Mass conservation error for the hill test with several boundary interpolation methods and the fifth order WENO scheme. Time integration is done with TVD3.

Step Initial Condition

Finally, we investigate the step test. Here, the analytical solution is not differentiable. All numerical methods and all boundary interpolation schemes lead to convergent solutions, but the order is always reduced to linear convergence in the L^1 norm and square-root convergence in the L^2 norm. This stems from the non-smoothness of the analytical solution and the action of the numerical integration scheme, as we demonstrate in the next section. Here, the grid interpolation method does not degrade the accuracy of the solution, but the stability.

Since we observe that the method of interpolation at the grid interface does not have any noticeable influence on the accuracy of the overall scheme, we only show results with linear interpolation in Figure 4.8. With several combinations of numerical scheme and boundary interpolation method, the numerical solutions get unstable even though the Courant number in these tests was only 0.1. This results in very high errors and slow convergence of the solutions, and happens mostly for the low-order methods such as the Upwind and the Lax–Wendroff scheme, as shown in Figure 4.9 for several grid resolutions. Higher numerical resolution decreases the error, which obviously is created when the step passes the grid interface marked by a vertical line. We do not observe these instabilities when using WENO methods.

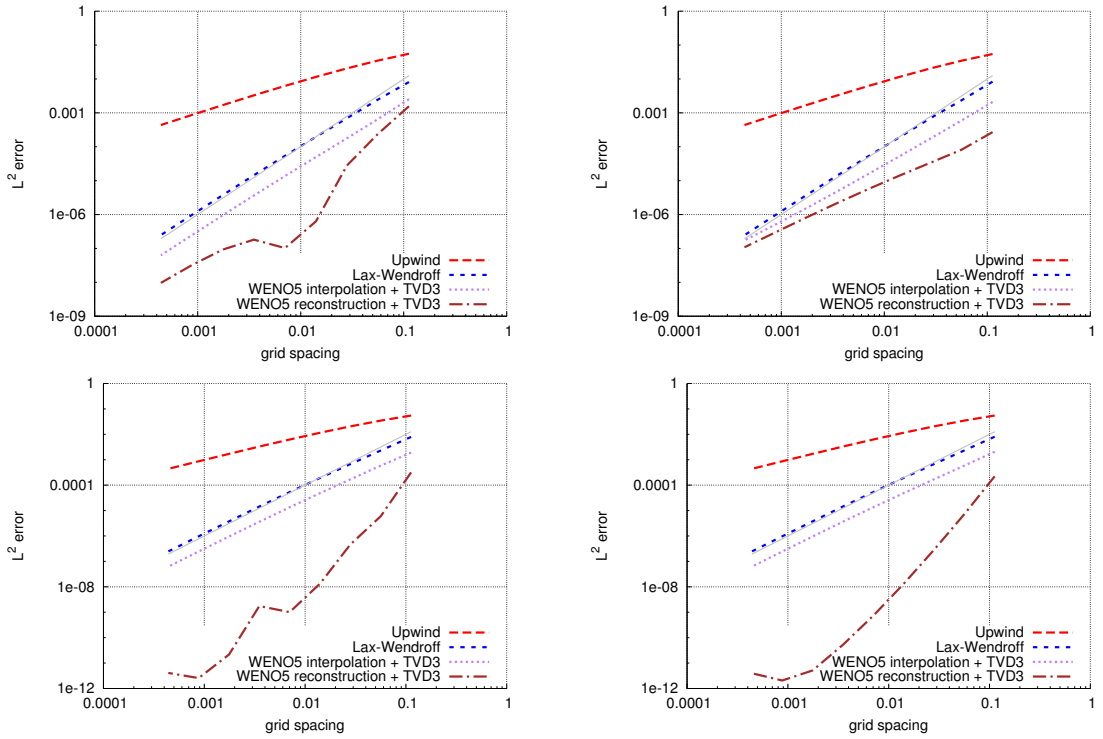

Figure 4.6: Error decay for the sine wave test with several boundary interpolation methods and numerical schemes as defined in Table 4.2. From top left to bottom right: linear interpolation, parabolic interpolation with flux correction, parabolic interpolation without flux correction, WENO5 interpolation. The grey line indicates second-order convergence.

Heaviside Initial Condition

We confirm that the slow convergence in the step test is not due to the grid interpolation method but stems from the non-smoothness of the analytical solution and the numerical viscosity of the WENO scheme by using the following test. Again, we consider the advection equation

$$\frac{\partial \phi}{\partial t} + a \frac{\partial \phi}{\partial x} = 0 \tag{4.26}$$

where a is the (constant) advection speed, with the Heaviside initial condition

$$\phi(x, 0) = H(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}. \tag{4.27}$$

The Heaviside function does not possess any strong or weak derivatives. The analytical solution of (4.26) is

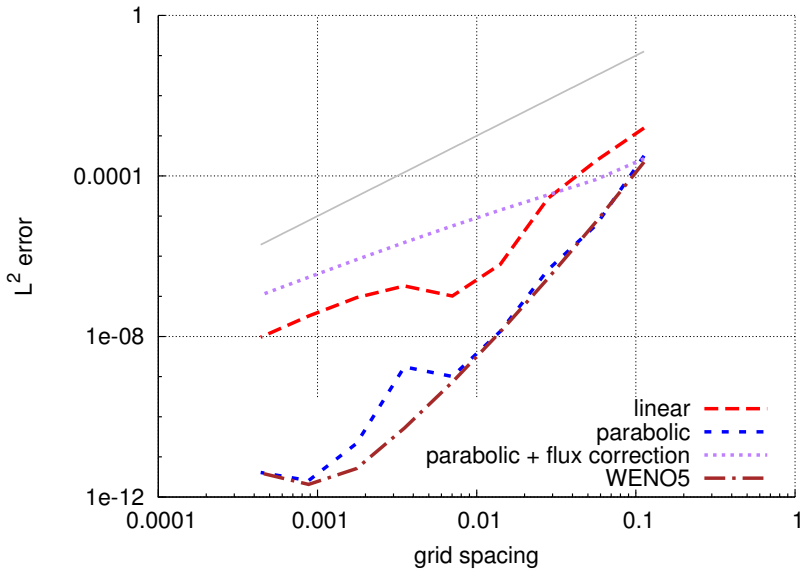

Figure 4.7: Error decay for the sine wave test with several boundary interpolation methods and the fifth order WENO scheme. Time integration is done with TVD3. The grey line indicates second-order convergence.

$$\phi\left(x,t\right)=\phi\left(x-a\,t,0\right), \tag{4.28}$$

i.e. a shift of the initial condition (4.27) by the distance $a\,t$.

We fix the Courant number $\sigma=\frac{a\,\delta t}{\delta x}=0.1$ and the advection speed $a=1$. When we solve the boundary value problem

$$\begin{gathered}\frac{\partial\phi}{\partial t}+\frac{\partial\phi}{\partial x}=0,\\ \phi\left(x,0\right)=H(x)=\begin{cases}0, & x<0\\ 1, & x\geq 0\end{cases},\ \phi\left(0,t\right)=0,\ \phi\left(10,t\right)=1,\end{gathered} \tag{4.29}$$

on the domain $x\in[0,10]$, $t\in[0,5]$ with the fifth order WENO scheme and TVD3 time integration as implemented in ANTARES on one equidistant grid, we get approximately first order convergence in the L^1 norm, but square root convergence in the L^2 norm at time $t=5\,\mathrm{s}$, as shown in Figure 4.10.

When looking at the numerical solution after 5 s for several numbers of grid points as given in Figure 4.11, we observe that besides the advection of the initial condition, the numerical viscosity of the WENO scheme leads to smoothing near the discontinuity. The width of the region where the solution is smoothed decreases with grid spacing. Furthermore, the position of the jump is shifted for all grid resolutions leading to large

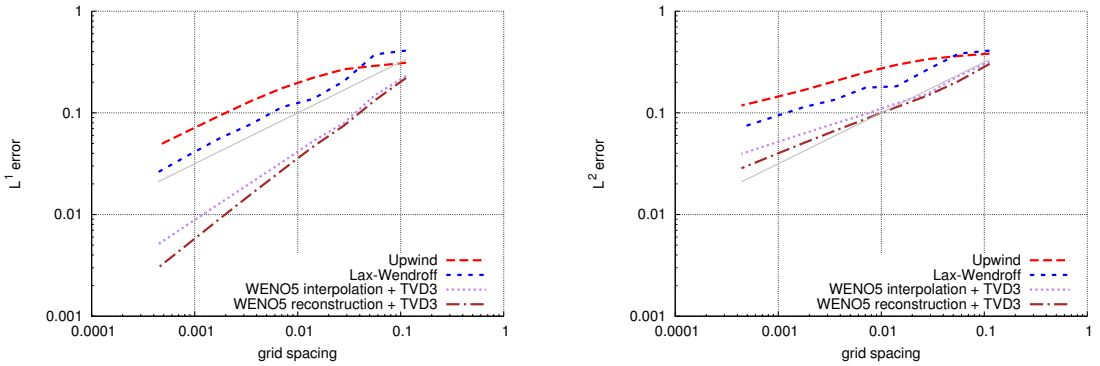

Figure 4.8: Error decay in the L^1 and the L^2 norm for the step test with linear boundary interpolation of variables and with several numerical schemes as defined in Table 4.2. The results do not differ considerably depending on the boundary interpolation method. On both pictures, the grey line indicates square-root convergence.

local errors, and there is a small overshoot for coarse grid resolutions.

Next, we try to find an analytical equation of the form

$$\frac{\partial \phi}{\partial t} - \kappa \frac{\partial^\beta \phi}{\partial x^\beta} = 0, \;\; \beta = 2, 3, \ldots, \tag{4.30}$$

which describes the action of the numerical viscosity of the WENO scheme in the Heaviside example. The analytical solution for $\beta = 2$ is given by (cf. p. 47 in Evans, 2002)

$$\phi(x,t) = \int_{-\infty}^{\infty} G(x-y,t) H(y)\, dy \text{ where } G(x,t) = \frac{1}{\sqrt{4\pi\kappa t}} \exp\left(-\frac{x^2}{4\kappa t}\right). \tag{4.31}$$

For the initial condition (4.27), we obtain

$$\phi(x,t) = \int_{0}^{\infty} G(x-y,t)\, dy = \int_{-\infty}^{x} G(z,t)\, dz, \tag{4.32}$$

by substituting $z = x - y$. Finally, by

$$\phi(x,t) = \frac{1}{\sqrt{4\pi\kappa t}} \int_{-\infty}^{x} \exp\left(-\frac{z^2}{4\kappa t}\right) dz, \tag{4.33}$$

we conclude that $\phi(x,t)$ is the distribution function of the Gaussian distribution with mean value 0 and standard deviation $\sqrt{2\kappa t}$. Therefore, for fixed x and t, the deviation from the initial condition (4.27) is proportional to $\sqrt{\kappa}$. The analytical solution for $\beta = 2$ and several values of κ given by (4.33) and calculated with *Mathematica* is shown on the left panel in Figure 4.12. On the right panel, these solutions are compared to the numerical solution of problem (4.29). We observe that the numerical solutions are not

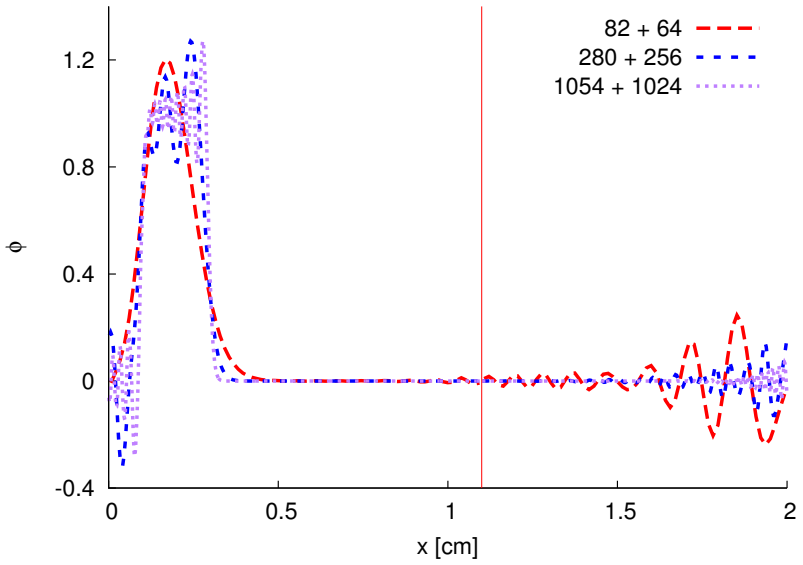

Figure 4.9: Numerical solution of the step test with the Lax–Wendroff scheme and linear boundary interpolation of variables. The results do not differ considerably depending on the boundary interpolation method. In the caption, we state the number of grid points on each of the grids. The vertical line is at the interface between the two grids.

perfectly symmetric as their analytical counterparts, but there is a rough correlation between numerical resolution and the value of κ in (4.30).

Given a numerical solution of (4.29) at time t, we define $x_{\text{jump}} := a\,t$, the analytically correct position of the jump for problem (4.29). For solutions of equation (4.30), we set $x_{\text{jump}} = 0$. Then, we define the left and right widths w_l and w_r for both the numerical solutions of (4.29) and the analytical solutions of (4.30) with initial condition (4.27) by

$$w_l = \underset{x<x_{\text{jump}}}{\operatorname{argmin}} \, ||\phi(x,t)| - \epsilon| - x_{\text{jump}}, \;\; w_r = \underset{x>x_{\text{jump}}}{\operatorname{argmin}} \, ||\phi(x,t) - 1| - \epsilon| - x_{\text{jump}}, \tag{4.34}$$

where ϵ is some small number, usually 0.01. We interpret w_l and w_r as the smoothing widths of the solution, and plot w_l and w_r for the numerical solution of (4.29) and the analytical solution of (4.30) for $\beta = 2$, in Figure 4.13. Note that for the analytical solution, $w_l = w_r$ since the solution is symmetric around x_{jump}. We observe that w_l and w_r of the numerical solution of (4.29) agree very well with the values for the analytical solution of (4.30), at least for intermediate and high resolutions. This result seems to be independent of advection speed and simulation time. We remark that the smoothing width w of the analytical solution of (4.30) scales with $\kappa^{1/\beta}$. Since we observe that the smoothing width of the numerical solution of (4.29) roughly scales with $\sqrt{\delta x}$, we

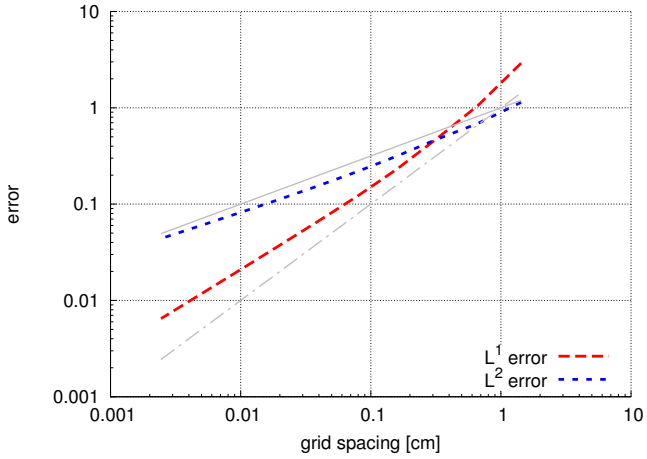

Figure 4.10: Empirical order of accuracy of the WENO5 scheme with TVD3 time integration when solving equation (4.29) with fixed $\sigma = 0.1$ in the L^1 and the L^2 norm. The grey lines indicate square root (solid line) and linear convergence (dash-dotted line).

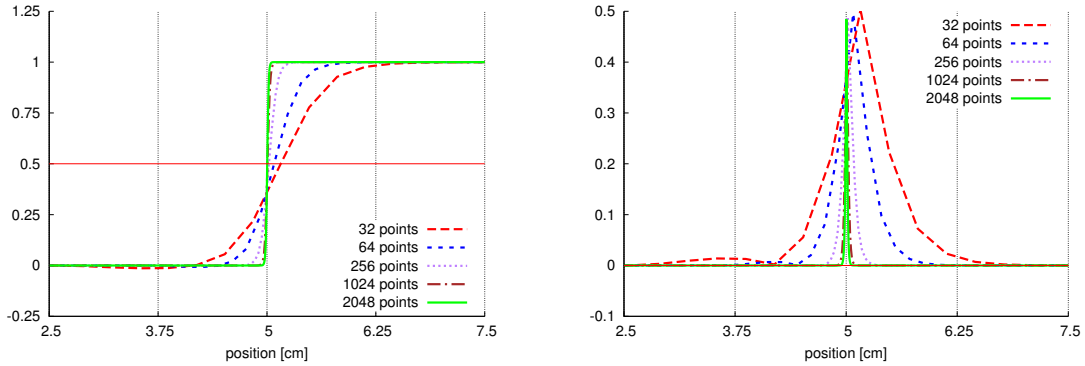

Figure 4.11: Numerical solution (left) and error of the numerical solution compared to the analytical solution (right) of problem (4.29) at $t \approx 5\,\mathrm{s}$ (end time differs slightly depending on time step size) with fixed $\sigma = 0.1$ for several grid resolutions. x ranges from 0 to 10 cm while t evolves over 0 to 5 s.

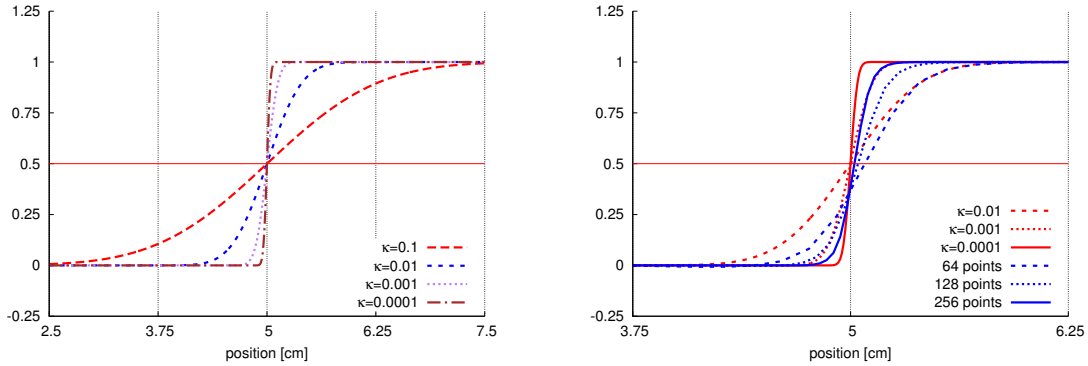

Figure 4.12: Left panel: analytical solution of equation (4.30) with $\beta = 2$ and several values of κ as calculated by *Mathematica*. The solution was shifted to be centred at $x = 5$. Right panel: comparison of the analytical solution of equation (4.30) with $\beta = 2$ (red lines) and the numerical solution of problem (4.29) (blue lines).

conclude that $\beta = 2$ describes the numerical viscosity of the WENO scheme.

We deduce that the numerical viscosity acts similarly to a diffusive process with $\beta = 2$ when we describe κ as a function of a and the grid resolution δx. From Figure 4.13, we obtain the approximate relation

$$w\left(\log \delta x\right) \approx w\left(\frac{3}{4}\log\left(\frac{\kappa}{a}\right) + \frac{3}{4}\right). \tag{4.35}$$

Combining (4.30) and (4.35), we conclude that for the Heaviside initial condition (4.27), the numerical viscosity of the WENO5 scheme acts like a diffusive process of the form

$$\frac{\partial \phi}{\partial t} - \left(0.1\, a\, (\delta x)^{\frac{4}{3}}\right) \frac{\partial^2 \phi}{\partial x^2} = 0. \tag{4.36}$$

The error arising from the numerical viscosity dominates the overall error in this case. Therefore, the numerical error is proportional to $\sqrt{0.1\, a\, (\delta x)^{\frac{4}{3}}} \sim (\delta x)^{\frac{2}{3}}$, which is similar to the L^1 and L^2 error convergence order we observe in Figure 4.10.

The fact that the L^2 error has a slower convergence order than the L^1 error can be explained by the displacement of the numerical solution compared to the analytical solution of (4.29) as shown in Figure 4.11. Since the L^2 error assesses a higher weight to outliers than the L^1 norm, the region where the errors are high due to the displacement affects the L^2 error stronger.

We conclude that the slow convergence of the WENO5 scheme when applied to the Heaviside initial condition (4.27) can be explained by the action of the numerical viscosity which dominates the overall error. The difference between the L^1 and the L^2 convergence is due to the displacement of the position of the jump in the numerical solution.

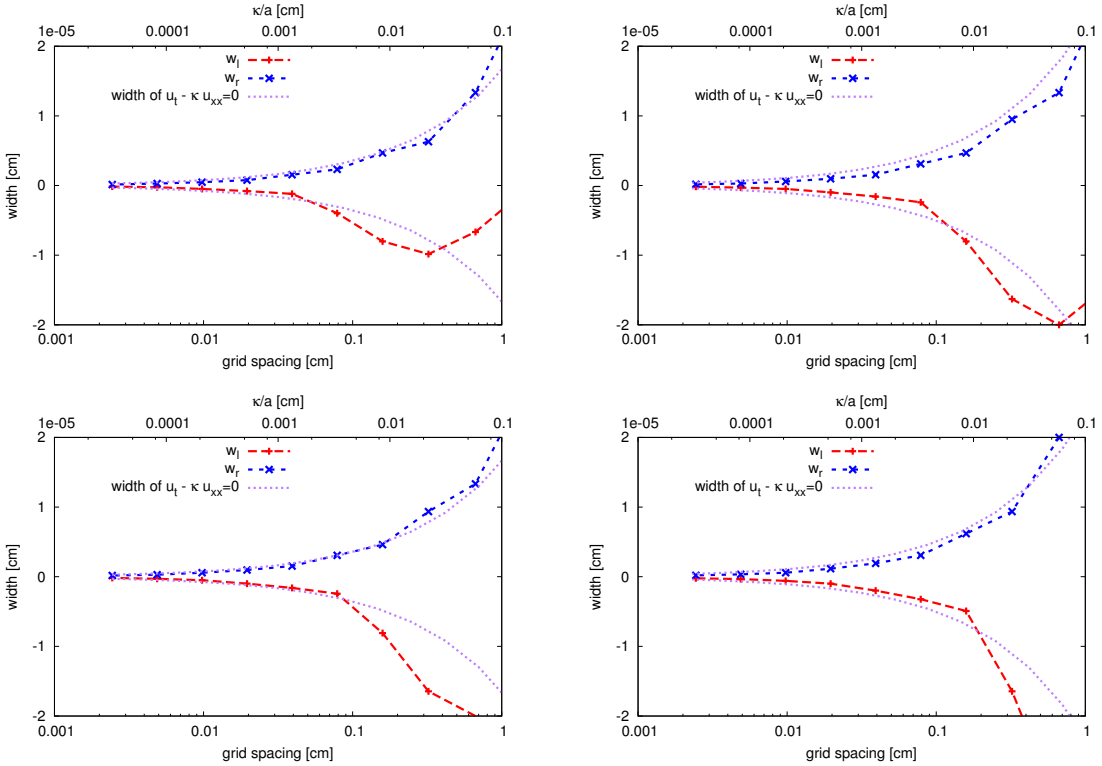

Figure 4.13: Smoothing widths w_l and w_r for the numerical solution of (4.29) and the analytical solution of (4.30) with $\beta = 2$. The widths of the numerical solution are plotted over grid spacing whereas the widths of the analytical solutions are plotted over values of κ/a. On the top for $a = 0.5\,\mathrm{cm\,s^{-1}}$, on the bottom for $a = 1\,\mathrm{cm\,s^{-1}}$. On the left for $t = 2.5\,\mathrm{s}$, on the right for $t = 5\,\mathrm{s}$.

Two-dimensional Euler Equations

We extend the composite approach to two dimensions and the Euler equations (3.74). As computational grid, we choose a Cartesian grid with side length 1.2 centred at the origin, and a spherical grid given by

$$x = r\cos\phi,\ y = r\sin\phi,\ r \in [0.5, 1], \phi \in [0, 2\pi]. \tag{4.37}$$

Therefore, the computational domain is the circle with radius 1. At the grid interface, we employ bilinear interpolation of the conservative variables as described in Peng et al. (2006) and Wongwathanarat et al. (2010).

In Figure 4.14, the results for the two-dimensional extension of the Sod shock tube (Sod, 1978) are shown, similar to the setup in Wongwathanarat et al. (2010). Outflow boundary conditions are used at the physical boudaries. We use 400 grid points in each direction on both grids.

We observe that the overall results look rather fine. The shock front stays straight and is only distorted at the physical boundaries by the outflow boundary conditions. The grid interface, on the other hand, does not seem to hinder the motion of the shock, but to transmit it without difficulties. Nevertheless, numerical noise is generated moving in shock direction, similar to what was seen in Pärt-Enander and Sjögreen (1994). We note that the Mach number of the SOD shock is around 1, whereas Pärt-Enander and Sjögreen (1994) suggest that slowly moving shocks are more problematic.

4.2.4 Conclusions

Our numerical experiments confirm the theoretical results of Chapter 4. Standard low-order interpolation methods at grid boundaries limit the accuracy of the overall solution and are not conservative. On the other hand, flux-based correction mechanisms like the one proposed in Berger (1987) fix the problem of conservation but impose a new limit on the overall convergence order. High-order approaches like the one from Sebastian and Shu (2003) allow high order convergence when high order accurate numerical schemes are used, but they induce conservation errors at the grid boundary. This error might be small and can be neglected in specific cases (Wongwathanarat et al., 2010), but for long-term numerical simulations with quasi-stationary flows as, e.g., the simulation of stellar convection, they are not suited. Especially when discontinuities are present in the numerical solution, the stability of all grid interpolation methods is very low.

Moreover, it is not trivial to solve elliptic equations on composite grids. For elliptic equations, the solution in each grid point is coupled at each instant of time to the whole computational domain. When the equation is solved on each grid separately, the algorithm must be iterated in a Schwartz–type procedure increasing the computational costs tremendously (Saad, 2003). Furthermore, the convergence of the iteration can be very slow, or the algorithm can fail to converge at all (e.g., Gander, 2005). Other approaches are to use an auxilliary grid where the elliptic equation is solved using standard techniques (Wongwathanarat et al., 2010).

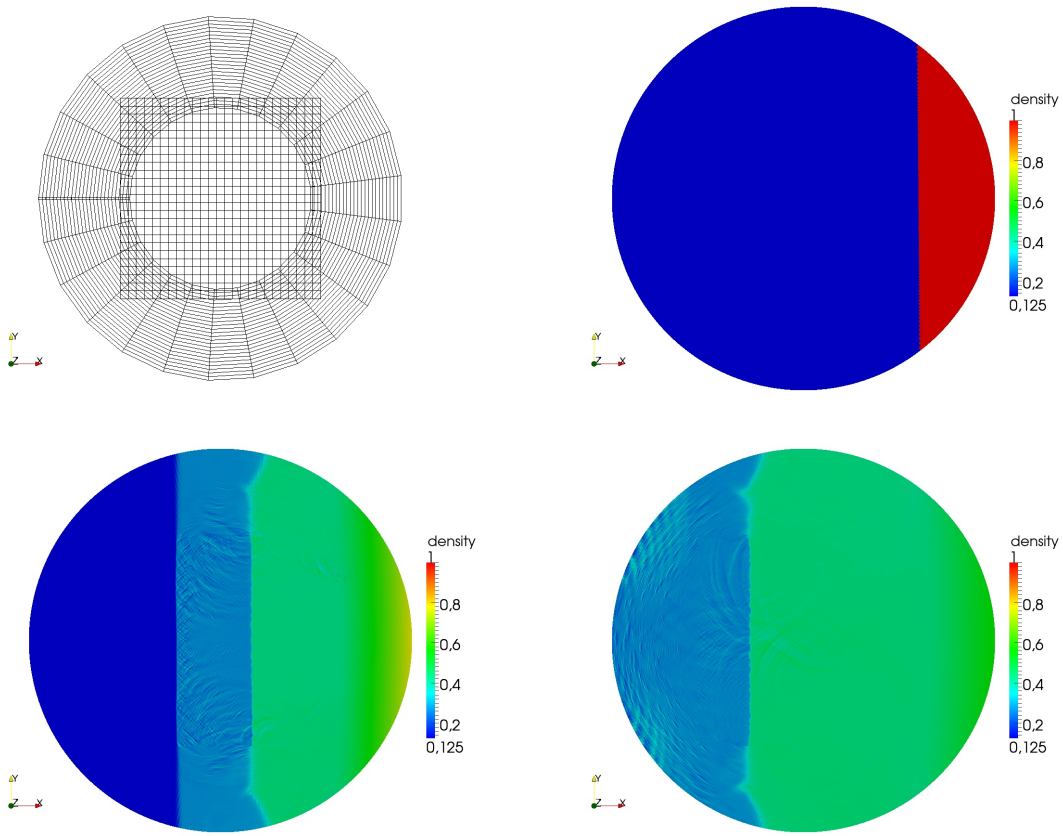

Figure 4.14: Time evolution of the two-dimensional SOD shock tube test on a composite grid. Top left: grids with only every 16th grid point plotted, top right: $t = 0\,\mathrm{s}$, bottom left: $t = 0.47\,\mathrm{s}$, bottom right: $t = 0.94\,\mathrm{s}$.

In higher dimensions, the calculation of total sums of conserved quantities as well as the identification of neighbouring cells on the other grid gets much more complicated. If overlap of grids is allowed, this can lead to a considerable waste of computation time, and the problem of merging the solutions together in the overlapping region must be solved. Finally, numerical noise is generated when a shock is passing a grid interface.

Keeping the concepts described in section 1.2 in mind, we note that it is unclear what happens to a turbulent flow at a grid interface, especially if the resolution is different between the grids. The results of an LES typically depend on the numerical resolution and could differ on each grid.

We therefore refrain from using the composite grid approach and present the alternative of curvilinear grids in the following chapter, which allows the design of high order, globally conservative schemes on reasonably complex geometries.

As a side remark, the interpolation-based WENO method (4.16) is only second-order accurate, but its error constants are much smaller than standard second-order accurate

schemes, and it has the advantage of non-oscillatory and shock-capturing numerical solutions. In case a reconstruction-based scheme cannot be used due to, e.g., the numerical grid, this scheme is a reasonable alternative.

5 Curvilinear Grids

Parts of this chapter, in particular the sections concerning WENO methods, are published in Grimm-Strele et al. (2013b).

Numerical schemes which are designed for Cartesian, equidistant grids can be generalised to more complicated domains with the technique of mapped grids. There, a mapping function

$$M: \ [-1,1]^3 \to \Omega, \ M(\xi,\eta,\zeta) = (x,y,z)^{\mathrm{T}}, \tag{5.1}$$

is defined which maps the Cartesian and equidistant computational space into the physical space. The information about the geometry of the physical space is then contained in the transformed partial differential equations. The Euler equations in strong conservation form in physical space (3.1) are transformed into strong conservation form in computational space. In three dimensions, they take the form

$$\frac{\partial}{\partial t} J^{-1}\mathbf{Q} + \frac{\partial}{\partial \xi}\hat{\mathbf{F}} + \frac{\partial}{\partial \eta}\hat{\mathbf{G}} + \frac{\partial}{\partial \zeta}\hat{\mathbf{H}} = 0 \tag{5.2a}$$

with

$$\hat{\mathbf{F}} = n_1\mathbf{F} + n_2\mathbf{G} + n_3\mathbf{H}, \tag{5.2b}$$

$$\hat{\mathbf{G}} = m_1\mathbf{F} + m_2\mathbf{G} + m_3\mathbf{H}, \tag{5.2c}$$

$$\hat{\mathbf{H}} = o_1\mathbf{F} + o_2\mathbf{G} + o_3\mathbf{H}, \tag{5.2d}$$

where J^{-1} is the determinant of the inverse Jacobian of the mapping function M (see, e.g., Kifonidis and Müller, 2012). The precise form of the factors n_1, n_2, n_3 and so on can be found in section 5.6. They consist of metric derivatives of the mapping function M. ξ, η, and ζ are the computational variables defined by M.

We present two derivations of the strong conservation form of the two-dimensional Euler equations in computational space which differ in their differentiability assumptions concerning the mapping function M. The classical first approach assuming strong differentiability of the mapping function can be found in section 5.1. A more general derivation is sketched out in section 5.2.

We work in two dimensions in the following, but the approach can be generalised directly to three dimensions. The results for three dimensions can be found in section 5.6.

5.1 Strong Derivation

If there is a differentiable function

$$M: \ [-1,1]^2 \to \Omega, \ M(\xi,\eta) = (x,y)^{\mathrm{T}}, \tag{5.3}$$

we can transform the two-dimensional Euler equations in differential form (3.74) to

$$\frac{\partial}{\partial t} J^{-1}\mathbf{Q} + \frac{\partial}{\partial \xi}\hat{\mathbf{F}} + \frac{\partial}{\partial \eta}\hat{\mathbf{G}} = 0 \tag{5.4a}$$

with

$$\hat{\mathbf{F}} = \frac{\partial y}{\partial \eta}\mathbf{F} - \frac{\partial x}{\partial \eta}\mathbf{G}, \tag{5.4b}$$

$$\hat{\mathbf{G}} = -\frac{\partial y}{\partial \xi}\mathbf{F} + \frac{\partial x}{\partial \xi}\mathbf{G}, \tag{5.4c}$$

and the determinant of the inverse Jacobian of the transformation

$$J^{-1} = \left|\frac{\partial(x,y)}{\partial(\xi,\eta)}\right| = \frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}. \tag{5.4d}$$

In this way, the conservation law (3.74) defined in the physical space is transformed into a conservation law in the computational space. If M is at least in C^1, all derivatives and the Jacobian are well-defined (Vinokur, 1974; Thompson et al., 1985; Kifonidis and Müller, 2012).

Following the description in Tannehill et al. (1997), this form can be derived by multiplying (3.74) with J^{-1} and rearranging terms. First we look at $\frac{\partial \mathbf{F}}{\partial x}J^{-1}$. With the chain rule of differentiation,

$$\begin{aligned}
\frac{\partial \mathbf{F}}{\partial x}J^{-1} &= \left(\frac{\partial \xi}{\partial x}\frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial \mathbf{F}}{\partial \eta}\right) J^{-1} \\
&= \frac{\partial}{\partial \xi}\left(\mathbf{F}\frac{\partial \xi}{\partial x}J^{-1}\right) + \frac{\partial}{\partial \eta}\left(\mathbf{F}\frac{\partial \eta}{\partial x}J^{-1}\right) \\
&\quad - \mathbf{F}\frac{\partial}{\partial \xi}\left(\frac{\partial \xi}{\partial x}J^{-1}\right) - \mathbf{F}\frac{\partial}{\partial \eta}\left(\frac{\partial \eta}{\partial x}J^{-1}\right).
\end{aligned}$$

In two dimensions,

$$\begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix}^{-1} = J \begin{pmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial x}{\partial \eta} \\ -\frac{\partial y}{\partial \xi} & \frac{\partial x}{\partial \xi} \end{pmatrix}, \tag{5.5}$$

and as a direct consequence,

$$\frac{\partial \xi}{\partial x} J^{-1} = \frac{\partial y}{\partial \eta},\ \frac{\partial \xi}{\partial y} J^{-1} = -\frac{\partial x}{\partial \eta},\ \frac{\partial \eta}{\partial x} J^{-1} = -\frac{\partial y}{\partial \xi},\ \frac{\partial \eta}{\partial y} J^{-1} = \frac{\partial x}{\partial \xi}. \tag{5.6}$$

We can further write

$$\frac{\partial \mathbf{F}}{\partial x} J^{-1} = \frac{\partial}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\mathbf{F}\right) + \frac{\partial}{\partial \eta}\left(-\frac{\partial y}{\partial \xi}\mathbf{F}\right) - \mathbf{F}\underbrace{\left(\frac{\partial^2 y}{\partial \xi \partial \eta} - \frac{\partial^2 y}{\partial \eta \partial \xi}\right)}_{=0}.$$

Here we assumed the mapping function to be at least in C^2. The same procedure leads to similar expressions for $\frac{\partial \mathbf{G}}{\partial y} J^{-1}$. Plugging all these expressions into (3.74) leads to (5.4).

5.2 Derivation Assuming Weak Differentiability

In applications, the mapping function M from the physical domain Ω to the computational domain is not known in closed form or does not possess an analytical form. We therefore seek for a derivation of the transformed equations (5.4) which does not require any differentiability of the mapping function M. Following the description in Wesseling (2001), we assume that a structured set of nodes $(x_{i\pm\frac{1}{2},j\pm\frac{1}{2}}, y_{i\pm\frac{1}{2},j\pm\frac{1}{2}})$, $1 \le i \le n$, $1 \le j \le m$, is given (e.g., by an external grid generation program, see Thompson et al., 1985), instead of an analytical expression for M. Ω is the convex hull of $\{(x_{i\pm\frac{1}{2},j\pm\frac{1}{2}}, y_{i\pm\frac{1}{2},j\pm\frac{1}{2}}) : 1 \le i \le n,\ 1 \le j \le m\}$. Then we define the discrete mapping function $\tilde{M}$ point-wisely by

$$\tilde{M}(\xi_{i-\frac{1}{2}}, \eta_{j-\frac{1}{2}}) = (x_{i-\frac{1}{2},j-\frac{1}{2}}, y_{i-\frac{1}{2},j-\frac{1}{2}}), i = 1, \ldots, n+1, j = 1, \ldots, m+1. \tag{5.7}$$

For $(\xi, \eta) \in C_{i,j} := \left[\xi_{i-\frac{1}{2}}, \xi_{i+\frac{1}{2}}\right] \times \left[\eta_{j-\frac{1}{2}}, \eta_{j+\frac{1}{2}}\right]$, we define $M(\xi, \eta)$ by bilinear interpolation of the physical coordinates of the edges of the cell. In this way, we continue $\tilde{M}$ to $M : [-1,1] \times [-1,1] \to \Omega$. M is continuous, linear in each $C_{i,j}$, but not differentiable on the boundary of each cell $C_{i,j}$. It follows that $M \notin C^1([-1,1]^2)$, but $M \in H^1([-1,1]^2)$.

If the mapping function is defined in this way, the image of each cell $C_{i,j}$ is a quadrilateral $D_{i,j}$ with edges $\left(x_{i-\frac{1}{2},j-\frac{1}{2}}, y_{i-\frac{1}{2},j-\frac{1}{2}}\right)$, $\left(x_{i-\frac{1}{2},j+\frac{1}{2}}, y_{i-\frac{1}{2},j+\frac{1}{2}}\right)$, $\left(x_{i+\frac{1}{2},j-\frac{1}{2}}, y_{i+\frac{1}{2},j-\frac{1}{2}}\right)$, and $\left(x_{i+\frac{1}{2},j+\frac{1}{2}}, y_{i+\frac{1}{2},j+\frac{1}{2}}\right)$ in physical space. All cell sides are straight line segments. We can choose the quadrilateral $D_{i,j}$ as the (arbitrary) control volume $W \subset \Omega$ of the integral formulation of the Euler equations

$$\frac{\partial}{\partial t}\int_W \rho\, dV = -\int_{\partial W} \mathbf{n}\cdot\rho\mathbf{u}\, dA, \tag{5.8a}$$

$$\frac{\partial}{\partial t}\int_W \rho\mathbf{u}\, dV = -\int_{\partial W} \left[\mathbf{n}\cdot\rho\mathbf{u}\otimes\mathbf{u} + \mathbf{n}p\right]\, dA, \tag{5.8b}$$

$$\frac{\partial}{\partial t}\int_W E\, dV = -\int_{\partial W} \mathbf{n}\cdot(p+E)\,\mathbf{u}\, dA, \tag{5.8c}$$

where $\mathbf{n}$ is the unit outward normal on ∂W, the boundary of W. Due to Gauss' Theorem (p. 627, Evans, 2002), equations (5.8) are equivalent to the differential form (3.74) for sufficiently smooth functions (Chorin and Marsden, 1993). But $\partial W = \partial D_{i,j}$ consists of the four line segments

$$S_{i\pm\frac{1}{2}} = \begin{pmatrix} x_{i\pm\frac{1}{2},j+\frac{1}{2}} \\ y_{i\pm\frac{1}{2},j+\frac{1}{2}} \end{pmatrix} - \begin{pmatrix} x_{i\pm\frac{1}{2},j-\frac{1}{2}} \\ y_{i\pm\frac{1}{2},j-\frac{1}{2}} \end{pmatrix}, \tag{5.9a}$$

$$S_{j\pm\frac{1}{2}} = \begin{pmatrix} x_{i+\frac{1}{2},j\pm\frac{1}{2}} \\ y_{i+\frac{1}{2},j\pm\frac{1}{2}} \end{pmatrix} - \begin{pmatrix} x_{i-\frac{1}{2},j\pm\frac{1}{2}} \\ y_{i-\frac{1}{2},j\pm\frac{1}{2}} \end{pmatrix}. \tag{5.9b}$$

Their length is given by

$$\left|S_{i\pm\frac{1}{2}}\right| = \sqrt{\left(x_{i\pm\frac{1}{2},j+\frac{1}{2}} - x_{i\pm\frac{1}{2},j-\frac{1}{2}}\right)^2 + \left(y_{i\pm\frac{1}{2},j+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2}}\right)^2}, \tag{5.10a}$$

$$\left|S_{j\pm\frac{1}{2}}\right| = \sqrt{\left(x_{i+\frac{1}{2},j\pm\frac{1}{2}} - x_{i-\frac{1}{2},j\pm\frac{1}{2}}\right)^2 + \left(y_{i+\frac{1}{2},j\pm\frac{1}{2}} - y_{i-\frac{1}{2},j\pm\frac{1}{2}}\right)^2}, \tag{5.10b}$$

and the normal vectors $\mathbf{n} = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}$ in equations (5.8) are exactly

$$n_1 = \left(y_{i\pm\frac{1}{2},j+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2}}\right) / \left|S_{i\pm\frac{1}{2}}\right|, \tag{5.10c}$$

$$n_2 = -\left(x_{i\pm\frac{1}{2},j+\frac{1}{2}} - x_{i\pm\frac{1}{2},j-\frac{1}{2}}\right) / \left|S_{i\pm\frac{1}{2}}\right| \tag{5.10d}$$

on $S_{i\pm\frac{1}{2}}$ and

$$n_1 = -\left(y_{i+\frac{1}{2},j\pm\frac{1}{2}} - y_{i-\frac{1}{2},j\pm\frac{1}{2}}\right) / \left|S_{j\pm\frac{1}{2}}\right|, \tag{5.10e}$$

$$n_2 = \left(x_{i+\frac{1}{2},j\pm\frac{1}{2}} - x_{i-\frac{1}{2},j\pm\frac{1}{2}}\right) / \left|S_{j\pm\frac{1}{2}}\right|. \tag{5.10f}$$

on $S_{j\pm\frac{1}{2}}$. From now on, we will write $\mathbf{n} = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}$ for the normal vector on $S_{i\pm\frac{1}{2}}$ and $\mathbf{m} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}$ for the normal vector on $S_{j\pm\frac{1}{2}}$. The surface area of the quadrilateral $D_{i,j}$ is

$$\begin{aligned} |D_{i,j}| =& \frac{1}{2}\Big|(y_{i-\frac{1}{2},j-\frac{1}{2}} - y_{i+\frac{1}{2},j+\frac{1}{2}})(x_{i-\frac{1}{2},j+\frac{1}{2}} - x_{i+\frac{1}{2},j-\frac{1}{2}}) \\ &+ (y_{i+\frac{1}{2},j-\frac{1}{2}} - y_{i-\frac{1}{2},j+\frac{1}{2}})(x_{i-\frac{1}{2},j-\frac{1}{2}} - x_{i+\frac{1}{2},j+\frac{1}{2}})\Big|. \end{aligned}$$

Since $|D_{i,j}| > 0$, we define the cell average of the state vector $\mathbf{Q}_{i,j}$ in the cell (i,j) by

$$\overline{\mathbf{Q}}_{i,j} = |D_{i,j}|^{-1} \int_{D_{i,j}} \mathbf{Q}\, dV. \tag{5.11}$$

With $U := n_1 u + n_2 v$, $V := m_1 u + m_2 v$, we can write equations (5.8) as

$$\frac{\partial}{\partial t}\overline{\mathbf{Q}}_{i,j} = -\left(\int_{S_{i+\frac{1}{2}}} \hat{\mathbf{F}}\, dS - \int_{S_{i-\frac{1}{2}}} \hat{\mathbf{F}}\, dS + \int_{S_{j+\frac{1}{2}}} \hat{\mathbf{G}}\, dS - \int_{S_{j-\frac{1}{2}}} \hat{\mathbf{G}}\, dS\right), \tag{5.12}$$

where

$$\mathbf{Q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \hat{\mathbf{F}} = \begin{pmatrix} \rho U \\ \rho U u + n_1 p \\ \rho U v + n_2 p \\ (p+E)U \end{pmatrix}, \hat{\mathbf{G}} = \begin{pmatrix} \rho V \\ \rho V u + m_1 p \\ \rho V u + m_2 p \\ (p+E)V \end{pmatrix}. \tag{5.13}$$

Evaluating the line integrals in (5.12) with the midpoint rule, we get

$$\frac{\partial}{\partial t}|D_{i,j}|\,\overline{\mathbf{Q}}_{i,j} = -\left(\hat{\mathbf{F}}_{i+\frac{1}{2},j} - \hat{\mathbf{F}}_{i-\frac{1}{2},j} + \hat{\mathbf{G}}_{i,j+\frac{1}{2}} - \hat{\mathbf{G}}_{i,j-\frac{1}{2}}\right). \tag{5.14}$$

In the computational space, all standard numerical methods can be used to calculate the value of the numerical flux functions $\hat{\mathbf{F}}$ and $\hat{\mathbf{G}}$ since the computational space is equidistant and Cartesian. In particular, both the finite difference and the finite volume WENO scheme as described in Algorithms 1 and 2 can be applied to solve (5.14). The specific form of the algorithms for curvilinear coordinates can be found in Algorithms 5 and 6.

The advantage of the weak derivation, besides that it does not require M to be differentiable, is that one gets formulae for all metric terms occurring in the transformation process. The Jacobian of the transformation is given by the area of the corresponding quadrilateral and therefore is positive by definition. The mapping function is not required to fulfil any smoothness properties. We therefore prefer to define our mapping function in this way.

Algorithm 5 Finite difference scheme for curvilinear coordinates.

1: $\mathbf{Q}_{i,j}$ is given as point value at the cell centre.
2: $\mathrm{A}(\hat{\mathbf{f}})_{i,j} = \hat{\mathbf{F}}_{i,j} = n_1|_{i,j}\mathbf{F}_{i,j} + n_2|_{i,j}\mathbf{G}_{i,j}$,
$\mathrm{A}(\hat{\mathbf{g}})_{i,j} = \hat{\mathbf{G}}_{i,j} = m_1|_{i,j}\mathbf{F}_{i,j} + m_2|_{i,j}\mathbf{G}_{i,j}$
3: $\hat{\mathbf{f}}_{i\pm\frac{1}{2},j} = \mathrm{R}_\xi\left(\hat{\mathbf{F}}_{i,j}\right)$, $\hat{\mathbf{g}}_{i,j\pm\frac{1}{2}} = \mathrm{R}_\eta\left(\hat{\mathbf{G}}_{i,j}\right)$
4: $\frac{\partial \mathbf{Q}_{i,j}}{\partial t} = -\frac{1}{\delta\xi}\left(\hat{\mathbf{f}}_{i+\frac{1}{2},j} - \hat{\mathbf{f}}_{i-\frac{1}{2},j}\right) - \frac{1}{\delta\eta}\left(\hat{\mathbf{g}}_{i,j+\frac{1}{2}} - \hat{\mathbf{g}}_{i,j-\frac{1}{2}}\right)$

Algorithm 6 Finite volume scheme for curvilinear coordinates.

1: $\mathbf{Q}_{i,j} = \overline{\mathbf{Q}}_{i,j}$ is given as cell average.
2: $\mathbf{Q}_{i\pm\frac{1}{2},j} = \mathrm{R}_\xi\left(\overline{\mathbf{Q}}_{i,j}\right)$, $\mathbf{Q}_{i,j\pm\frac{1}{2}} = \mathrm{R}_\eta\left(\overline{\mathbf{Q}}_{i,j}\right)$
3: $\hat{\mathbf{F}}_{i\pm\frac{1}{2},j} = n_1|_{i\pm\frac{1}{2},j}\mathbf{F}\left(\mathbf{Q}_{i\pm\frac{1}{2},j}\right) + n_2|_{i\pm\frac{1}{2},j}\mathbf{G}\left(\mathbf{Q}_{i\pm\frac{1}{2},j}\right)$,
$\hat{\mathbf{G}}_{i,j\pm\frac{1}{2}} = m_1|_{i,j\pm\frac{1}{2}}\mathbf{F}\left(\mathbf{Q}_{i,j\pm\frac{1}{2}}\right) + m_2|_{i,j\pm\frac{1}{2}}\mathbf{G}\left(\mathbf{Q}_{i,j\pm\frac{1}{2}}\right)$
4: $\frac{\partial \overline{\mathbf{Q}}_{i,j}}{\partial t} = -\frac{1}{\delta\xi}\left(\hat{\mathbf{F}}_{i+\frac{1}{2},j} - \hat{\mathbf{F}}_{i-\frac{1}{2},j}\right) - \frac{1}{\delta\eta}\left(\hat{\mathbf{G}}_{i,j+\frac{1}{2}} - \hat{\mathbf{G}}_{i,j-\frac{1}{2}}\right)$

We note that similar formulae exist for the three-dimensional case. The precise formulation can be found, e.g., in section 5.6 and in Visbal and Gaitonde (2002).

5.3 Parabolic Terms

We call terms in time-dependent partial differential equations like (2.1) containing second derivatives *parabolic terms*. In contrast, the Euler equations (3.74) do not contain any parabolic terms. A precise definition of the term in two dimensions is given in the following

Definition 7. *In terms of Evans (2002, p. 350), a partial differential equation in two space dimensions of the form*

$$\frac{\partial u}{\partial t} - b_{11}\frac{\partial^2 u}{\partial \xi^2} - 2b_{12}\frac{\partial^2 u}{\partial \xi \partial \eta} - b_{22}\frac{\partial^2 u}{\partial \eta^2} = 0 \tag{5.15}$$

is parabolic if there exists a constant $\alpha > 0$ such that

$$b_{11}\zeta_1^2 + 2b_{12}\zeta_1\zeta_2 + b_{22}\zeta_2^2 \geq \alpha\,|\zeta| \tag{5.16}$$

for each $\zeta = (\zeta_1, \zeta_2)^{\mathrm{T}} \in \mathbb{R}^2$.

As a starting point, we consider the one-dimensional diffusion equation

$$\frac{\partial \phi}{\partial t} - D\frac{\partial^2 \phi}{\partial x^2} = \frac{\partial \phi}{\partial t} - D\frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) = 0 \tag{5.17}$$

with the constant coefficient of diffusion D. In one spatial dimension and on an equidistant Cartesian grid, the equation can be discretised with fourth order accuracy by the method described in Happenhofer et al. (2013) and in section 3.2.1. Therein, the outer derivative is approximated by

$$\frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right)(x_i) = \frac{\frac{\partial \phi}{\partial x}\left(x_{i+\frac{1}{2}}\right) - \frac{\partial \phi}{\partial x}\left(x_{i-\frac{1}{2}}\right)}{\delta x} \tag{5.18}$$

with constant grid spacing δx. Then, the inner derivative is calculated by

$$\frac{\partial \phi}{\partial x}\left(x_{i-\frac{1}{2}}\right) = \frac{\phi_{i-2} - 15\phi_{i-1} + 15\phi_i - \phi_{i+1}}{12\,\delta x}, \tag{5.19}$$

leading to a fourth-order accurate approximation both in the finite volume and the finite difference context. Here, $\phi_i = \phi(x_i)$.

Similar procedures can be applied to any parabolic term, in particular the viscous stress tensor in the Navier–Stokes equations (2.1). Special care has to be taken for mixed derivatives. In the two-dimensional case, neglecting all but the $\nabla \cdot \tau$ term in (2.1), we arrive at

$$\frac{\partial}{\partial t}(\mu_x) = \frac{\partial}{\partial x}\left(\left(\zeta + \frac{4}{3}\eta\right)\frac{\partial u}{\partial x} + \left(\zeta - \frac{2}{3}\eta\right)\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial y}\left(\eta\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right) \tag{5.20}$$

by virtue of equations (2.1) and (2.2). Both in the finite volume and in the finite difference approach, the outer derivatives are replaced by a finite difference, evaluating the inner function at the half-integer nodes. Therefore, we need the terms inside the spatial derivatives in (5.20) at $(i-\frac{1}{2}, j)$ and at $(i, j-\frac{1}{2})$. $\frac{\partial u}{\partial x}$ at $(i-\frac{1}{2}, j)$ and $\frac{\partial v}{\partial y}$ at $(i, j-\frac{1}{2})$ can be calculated directly by formula (5.19). Then, the coefficient functions must be interpolated to the half-integer grid. To fourth order accuracy,

$$\eta_{i-\frac{1}{2},j} = \frac{-\eta_{i-2,j} + 7\eta_{i-1,j} + 7\eta_{i,j} - \eta_{i+1,j}}{12}, \tag{5.21}$$

assuming that the variable is given as a cell average. To calculate $\frac{\partial v}{\partial y}$ at the half integer index, we calculate the derivative at the cell centre by

$$\frac{\partial v}{\partial y}|_{i,j} = \frac{v_{i,j-2} - 8v_{i,j-1} + 8v_{i,j+1} - v_{i,j+2}}{12\,\delta y}, \tag{5.22}$$

and then interpolate the result to $\left(i-\frac{1}{2}, j\right)$ according to formula (5.21). The resulting procedure is fourth-order accurate.

5.3.1 Numerical Experiments

We demonstrate the accuracy of the scheme by solving the diffusion equation with mixed terms

$$\frac{\partial \phi}{\partial t} - D\left(\frac{\partial^2 \phi}{\partial x^2} + c_{\text{mix}}\frac{\partial^2 \phi}{\partial x \partial y} + \frac{\partial^2 \phi}{\partial y^2}\right) = 0 \tag{5.23}$$

on a Cartesian grid. With the initial condition

$$\phi(x, y, t = 0) = 2 + \sin\left(\pi\left(2x + y\right)\right), \tag{5.24}$$

the analytical solution on $[0, 2]^2$ is given by

$$\phi(x, y, t) = 2 + \exp\left(-(5 + 2c_{\text{mix}})\pi^2 D t\right) \sin\left(\pi\left(2x + y\right)\right). \tag{5.25}$$

The parameter c_{mix} regulates the strength of the mixed term in (5.23). With $c_{\text{mix}} = 0$, the mixed term disappears and we arrive at the classical diffusion equation. We write the equation in the conservative formulation

$$\frac{\partial \phi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \tag{5.26}$$

by defining the flux functions

$$F = -D\left(\frac{\partial \phi}{\partial x} + \frac{c_{\text{mix}}}{2}\frac{\partial \phi}{\partial y}\right), \; G = -D\left(\frac{\partial \phi}{\partial y} + \frac{c_{\text{mix}}}{2}\frac{\partial \phi}{\partial x}\right). \tag{5.27}$$

To calculate the first derivative in the flux functions F and G, we can directly apply the formula (5.19), whereas we have to use (5.22) and (5.21) to calculate the mixed terms at the half-integer grid. Writing `DXB` for the application of formula (5.19), `DXC` for (5.22), and `INTX` for (5.21) in x direction and corresponding expressions for the y direction,

$$F_{i-\frac{1}{2},j} = -D\left(\texttt{DXB}(\phi) + \frac{c_{\text{mix}}}{2}\texttt{INTX}(\texttt{DYC}(\phi))\right), \tag{5.28a}$$

$$G_{i,j-\frac{1}{2}} = -D\left(\texttt{DYB}(\phi) + \frac{c_{\text{mix}}}{2}\texttt{INTY}(\texttt{DXC}(\phi))\right). \tag{5.28b}$$

If the coefficients D and c_{mix} were spatially varying, we would have to interpolate them by formula (5.21) to the boundary grid as well.

From Figure 5.1 we deduce that even for large values of c_{mix}, the scheme proves to be fourth order accurate, even though the error constants slightly increase. Only for $c_{\text{mix}} = 8$ corresponding to a situation where the mixed derivative strongly dominates the equation, the simulation gets unstable. In this case, the equation is not parabolic any more (p. 142, Strikwerda, 1989), and we cannot except a stable solution from our scheme. For all stable cases, the empirical values of the error constant C are given in Table 5.1. In all of these simulations, the Courant number was fixed to 0.1 and the diffusion coefficient D was 10^{-3}.

5.3.2 Transformation to Curvilinear Coordinates

In this section, we consider the two-dimensional diffusion equation without mixed terms

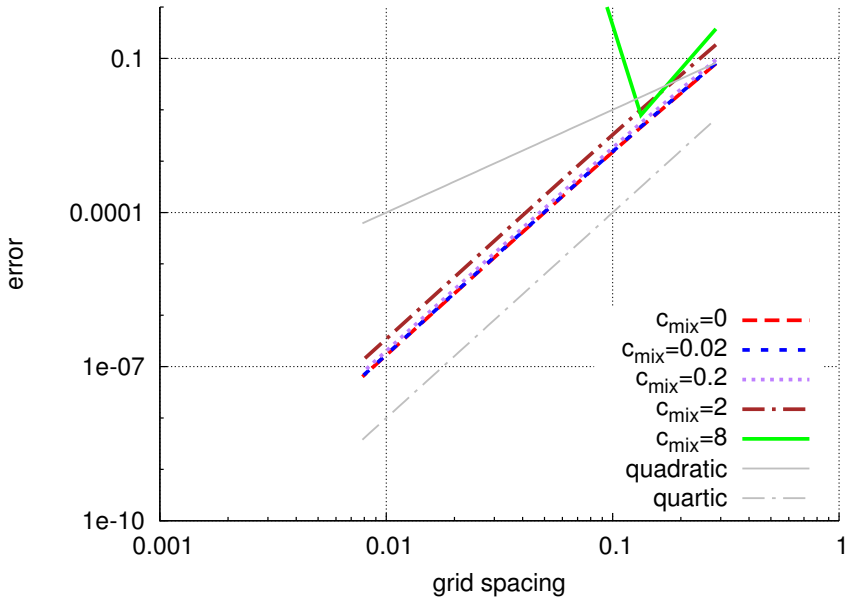

Figure 5.1: Empirical order of accuracy of the ANTARES scheme for diffusive terms when solving equation (5.23) with different values of c_{mix}. All simulations use the TVD2 Runge–Kutta scheme (Shu and Osher, 1988) for temporal discretisation, and a fixed Courant number of 0.1. Grey lines indicate second and fourth order convergence, respectively.

	$c_{\text{mix}} = 0$		$c_{\text{mix}} = 0.02$		$c_{\text{mix}} = 0.2$		$c_{\text{mix}} = 2.0$	
δx	p	C	p	C	p	C	p	C
0.286	3.735	8.53e0	3.733	8.69e0	3.725	1.01e1	3.815	2.19e1
0.133	3.885	1.16e1	3.885	1.18e1	3.885	1.39e1	3.924	2.73e1
0.065	3.948	1.37e1	3.948	1.40e1	3.949	1.66e1	3.968	3.08e1
0.032	3.975	1.51e1	3.975	1.54e1	3.977	1.83e1	3.991	3.34e1
0.016	3.988	1.59e1	3.988	1.63e1	3.989	1.92e1	3.997	3.42e1

Table 5.1: Empirical order of accuracy p and error constants C for the numerical solution of (5.23) in dependence of the strength of the mixed derivatives. We observe that the order of accuracy is about 4 in all cases, but the error constant for c_{mix} is more than twice as large for $c_{\text{mix}} = 2.0$ than without mixed terms. c_{mix} is a borderline case for the parabolic nature of (5.23). Strictly speaking, (5.23) is parabolic only if $|c_{\text{mix}}| < 2$.

$$\frac{\partial \phi}{\partial t} - D\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) = 0 \tag{5.29}$$

on a general structured grid. We assume that a mapping function M is given which maps the physical space in a Cartesian equidistant computational space. There, we can apply the standard procedure from section 5.3 for a Cartesian grid to the transformed equation.

Writing $F = -D\frac{\partial \phi}{\partial x}$ and $G = -D\frac{\partial \phi}{\partial y}$, (5.29) can be transformed to the "conservation law"

$$\frac{\partial \phi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0. \tag{5.30}$$

Similar to what was done in sections 5.1 and 5.2, by applying the chain rule of differentiation, this equation can be brought into strong conservation form in the computational space under the assumption that M is in H^1. As before, we write ξ and η for the independent variables in computational space, and $J^{-1} = \left|\frac{\partial(x,y)}{\partial(\xi,\eta)}\right|$ for the inverse Jacobian of the transformation M. With the transformed fluxes

$$\hat{F} = \frac{\partial y}{\partial \eta}F - \frac{\partial x}{\partial \eta}G,\ \hat{G} = -\frac{\partial y}{\partial \xi}F + \frac{\partial x}{\partial \xi}G, \tag{5.31}$$

the diffusion equation (5.29) transforms to

$$\frac{\partial J^{-1}\phi}{\partial t} + \frac{\partial \hat{F}}{\partial \xi} + \frac{\partial \hat{G}}{\partial \eta} = 0. \tag{5.32}$$

But the fluxes $\hat{F}$ and $\hat{G}$ still contain x and y derivatives. In the following, we write

$$n_1 := \frac{\partial y}{\partial \eta},\ n_2 := -\frac{\partial x}{\partial \eta},\ m_1 := -\frac{\partial y}{\partial \xi},\ m_2 := \frac{\partial x}{\partial \xi}, \tag{5.33}$$

and calculate the inverse Jacobian J^{-1} by

$$J^{-1} := \left| \frac{\partial(x,y)}{\partial(\xi,\eta)} \right| = n_1 m_2 - n_2 m_1. \tag{5.34}$$

The relations

$$\frac{\partial \xi}{\partial x} J^{-1} = n_1,\ \frac{\partial \eta}{\partial x} J^{-1} = m_1,\ \frac{\partial \xi}{\partial y} J^{-1} = n_2,\ \frac{\partial \eta}{\partial y} J^{-1} = m_2 \tag{5.35}$$

are direct consequences of these definitions. With the chain rule,

$$FJ^{-1} = -D\frac{\partial \phi}{\partial x} J^{-1} = -D\left(n_1 \frac{\partial \phi}{\partial \xi} + m_1 \frac{\partial \phi}{\partial \eta} \right), \tag{5.36a}$$

$$GJ^{-1} = -D\frac{\partial \phi}{\partial y} J^{-1} = -D\left(n_2 \frac{\partial \phi}{\partial \xi} + m_2 \frac{\partial \phi}{\partial \eta} \right). \tag{5.36b}$$

assuming again that M is at least in H^1. Therefore, the transformed fluxes $\hat{F}$ and $\hat{G}$ can be written as

$$\hat{F} = n_1 F + n_2 G = -DJ\left(\left(n_1^2 + n_2^2\right) \frac{\partial \phi}{\partial \xi} + \left(n_1 m_1 + n_2 m_2\right) \frac{\partial \phi}{\partial \eta} \right), \tag{5.37a}$$

$$\hat{G} = m_1 F + m_2 G = -DJ\left(\left(n_1 m_1 + n_2 m_2\right) \frac{\partial \phi}{\partial \xi} + \left(m_1^2 + m_2^2\right) \frac{\partial \phi}{\partial \eta} \right). \tag{5.37b}$$

Even though we started with the classical diffusion equation (5.29) without any mixed terms, the transformed equation, where all x and y derivatives have been replaced by ξ and η derivatives, has a more complicated form, containing coefficient functions and mixed derivatives.

We check if the equation is still parabolic in terms of definition 7. Definition 7 is equivalent to the requirement that

$$A = \begin{pmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{pmatrix} \tag{5.38}$$

has only positive eigenvalues. The eigenvalues of A are given by

$$\lambda_{1,2} = \frac{b_{11} + b_{22}}{2} \pm \sqrt{\left(\frac{b_{11} + b_{22}}{2} \right)^2 + \left(b_{12}^2 - b_{11} b_{22} \right)}. \tag{5.39}$$

Therefore, the above differential equation is parabolic if (p. 142, Strikwerda, 1989)

$$b_{11},\, b_{22} > 0 \text{ and } b_{12}^2 < b_{11} b_{22}. \tag{5.40}$$

In our case (dropping the positive factor DJ in front of all terms),

$$b_{11} = n_1^2 + n_2^2 > 0,\; b_{12} = n_1 m_1 + m_2 n_2,\; b_{22} = m_1^2 + m_2^2 > 0. \tag{5.41}$$

We calculate

$$\begin{aligned} b_{11} b_{22} - b_{12}^2 &= \left((n_1 m_1)^2 + (n_1 m_2)^2 + (n_2 m_1)^2 + (n_2 m_2)^2 \right) \\ &\quad - \left((n_1 m_1)^2 + 2 n_1 n_2 m_1 m_2 + (n_2 m_2)^2 \right) \\ &= (n_1 m_2 - n_2 m_1)^2 = \left(J^{-1} \right)^2 > 0, \end{aligned}$$

since $J^{-1} > 0$. Therefore, we showed that the conditions (5.40) are fulfilled. The transformed equation is parabolic again.

The transformed equation (5.32) with the flux functions defined by (5.37) can be discretised in the computational space. Since the grid in the computational space is equidistant and Cartesian, we can apply the methods from section 5.3 to solve the equation there.

Following Calhoun et al. (2008), we discretise equation (5.32) in space in the finite volume setting by

$$\frac{\partial J^{-1} \phi}{\partial t} = - \left(\hat{F}_{i+\frac{1}{2},j} - \hat{F}_{i-\frac{1}{2},j} + \hat{G}_{i,j+\frac{1}{2}} - \hat{G}_{i,j-\frac{1}{2}} \right). \tag{5.42}$$

The fluxes are calculated by

$$\hat{F}_{i\pm\frac{1}{2},j} == -DJ_{i\pm\frac{1}{2},j} \left(\left| S_{i\pm\frac{1}{2}} \right|^2 \frac{\partial \phi}{\partial \xi} |_{i\pm\frac{1}{2},j} + T_{i\pm\frac{1}{2}} \frac{\partial \phi}{\partial \eta} |_{i\pm\frac{1}{2},j} \right), \tag{5.43a}$$

$$\hat{G}_{i,j\pm\frac{1}{2}} == -DJ_{i,j\pm\frac{1}{2}} \left(T_{j\pm\frac{1}{2}} \frac{\partial \phi}{\partial \xi} |_{i,j\pm\frac{1}{2}} + \left| S_{j\pm\frac{1}{2}} \right|^2 \frac{\partial \phi}{\partial \eta} |_{i,j\pm\frac{1}{2}} \right). \tag{5.43b}$$

with the cell surfaces $S_{i\pm\frac{1}{2}}$ and $S_{j\pm\frac{1}{2}}$ as defined in equations (5.9), and $T_{i\pm\frac{1}{2}}$ and $T_{j\pm\frac{1}{2}}$ defined by

$$T_{i\pm\frac{1}{2}} = (n_1 m_1 + n_2 m_2) \,|_{i\pm\frac{1}{2},j},\; T_{j\pm\frac{1}{2}} = (n_1 m_1 + n_2 m_2) \,|_{i,j\pm\frac{1}{2}}. \tag{5.44}$$

We note that compared to the terms needed for hyperbolic equations, additional metric terms must be stored, i.e. $J_{i\pm\frac{1}{2},j}$, $J_{i,j\pm\frac{1}{2}}$, $T_{i\pm\frac{1}{2}}$ and $T_{j\pm\frac{1}{2}}$. They should not be obtained by interpolation, but be calculated using the mapping function M itself (cf. p. 118 in Thompson et al., 1985). For the discretisation of the derivates contained in $\hat{F}_{i\pm\frac{1}{2},j}$ and $\hat{G}_{i,j\pm\frac{1}{2}}$, Calhoun et al. (2008) used second order finite differences and interpolations, i.e.

$$\frac{\partial \phi}{\partial \xi}|_{i+\frac{1}{2},j} \approx \phi_{i+1,j} - \phi_{i,j}, \tag{5.45a}$$

$$\frac{\partial \phi}{\partial \eta}|_{i+\frac{1}{2},j} \approx \frac{\phi_{i+\frac{1}{2},j+1} - \phi_{i+\frac{1}{2},j-1}}{2} \approx \frac{\phi_{i,j+1} + \phi_{i+1,j+1}}{4} - \frac{\phi_{i,j-1} + \phi_{i+1,j-1}}{4}, \tag{5.45b}$$

$$\frac{\partial \phi}{\partial \xi}|_{i,j+\frac{1}{2}} \approx \frac{\phi_{i+1,j+\frac{1}{2}} - \phi_{i-1,j+\frac{1}{2}}}{2} \approx \frac{\phi_{i+1,j} + \phi_{i+1,j+1}}{4} - \frac{\phi_{i-1,j} + \phi_{i-1,j+1}}{4}, \tag{5.45c}$$

$$\frac{\partial \phi}{\partial \eta}|_{i,j+\frac{1}{2}} \approx \phi_{i,j+1} - \phi_{i,j}. \tag{5.45d}$$

Alternatively, one can use combinations of (5.19), (5.21) and (5.22) to obtain higher order approximations.

In general, for all kinds of partial differential equations, one can either discretise the transformed equations in computational space, or one can discretise in the physical space (Calhoun et al., 2008). It may depend on the application at hand which way is the more accurate and stable one (e.g., Happenhofer, 2013).

5.3.3 WENO–type Scheme to Calculate Derivatives

In the following, we present an alternative approach to calculate derivatives on a one-dimensional equidistant grid. It relies on the WENO idea to use several interpolation polynomials in the neighbourhood of the point where the derivative should be calculated. This approach can be used in the numerical solution of parabolic terms as an alternative to the method from Happenhofer et al. (2013), or to calculate the metric derivatives $\frac{\partial x}{\partial \xi}$, $\frac{\partial y}{\partial \eta}$, $\frac{\partial x}{\partial \eta}$ and $\frac{\partial y}{\partial \xi}$.

Problem 4 (Numerical Differentiation)**.** *Given the point values of a function ϕ on an equidistant grid $\{\xi_i, i = 1, \ldots, n\}$, calculate the derivative $\phi'(\xi_i)$.*

Remark 1. *Problem 4 can be formulated in a much more general way, but we only need it in the special case stated here. See Shu (2001) for a more general treatment.*

Remark 2. *We assume in the following that the variable ϕ is given by point values. Alternatively, it could be given by cell averages as well. The algorithmic changes are then similar to the differences between the WENO interpolation and reconstruction scheme, as they are described in paragraphs 4.2.2 and 3.1.2, respectively.*

The basic idea of this approach is outlined in Shu (2001). On the stencil $S_1 = \{\xi_{i-2}, \xi_{i-1}, \xi_i\}$, a unique approximation of $\phi'(\xi_i)$ can be obtained by differentiating the third order interpolation polynomial to ϕ on S_1. This yields the approximation

$$\phi_1'(\xi_i) = \frac{1}{2}\left(\phi_{i-2} - 4\phi_{i-1} + 3\phi_i\right). \tag{5.46}$$

Analogously for $S_2 = \{\xi_{i-1}, \xi_i, \xi_{i+1}\}$

$$\phi_2'(\xi_i) = \frac{1}{2}\left(-\phi_{i-1} + \phi_{i+1}\right), \tag{5.47}$$

and for $S_3 = \{\xi_i, \xi_{i+1}, \xi_{i+2}\}$

$$\phi_3'(\xi_i) = \frac{1}{2}\left(-3\phi_i + 4\phi_{i+1} - \phi_{i+2}\right). \tag{5.48}$$

Comparison with the fourth order interpolation polynomial,

$$\phi'(\xi_i) = \frac{1}{12}\left(\phi_{i-2} - 8\phi_{i-1} + 8\phi_{i+1} - \phi_{i+2}\right), \tag{5.49}$$

gives the linear weights d_1, d_2 and d_3 via

$$\phi'(\xi_i) = d_1\phi_1'(\xi_i) + d_2\phi_2'(\xi_i) + d_3\phi_3'(\xi_i). \tag{5.50}$$

We calculate

$$d_1 = \frac{1}{6}, d_2 = \frac{2}{3} \text{ and } d_3 = \frac{1}{6}. \tag{5.51}$$

As in the procedure described in section 4.2.2, these linear weights are replaced by nonlinear ones measuring the smoothness of the interpolation polynomial. According to Shu (2001), we set

$$\beta_j = (\delta\xi)^3 \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i-\frac{1}{2}}} \left(\frac{d^2}{d\xi^2}\phi(\xi)\right)^2 d\xi \tag{5.52}$$

which for $\xi = \xi_i$ leads to

$$\beta_1 = \phi_{i-2}^2 - 4\phi_{i-2}\phi_{i-1} + 4\phi_{i-1}^2 + 2\phi_{i-2}\phi_i - 4\phi_{i-1}\phi_i + \phi_i^2, \tag{5.53}$$

$$\beta_2 = \phi_{i-1}^2 - 4\phi_{i-1}\phi_i + 4\phi_i^2 + 2\phi_{i-1}\phi_{i+1} - 4\phi_i\phi_{i+1} + \phi_{i+1}^2, \tag{5.54}$$

$$\beta_3 = \phi_i^2 - 4\phi_i\phi_{i+1} + 4\phi_{i+1}^2 + 2\phi_i\phi_{i+2} - 4\phi_{i+1}\phi_{i+2} + \phi_{i+2}^2. \tag{5.55}$$

The final numerical approximation for $\phi'(\xi_i)$ is then obtained via

$$\phi'(\xi_i) = \omega_1\phi_1'(\xi_i) + \omega_2\phi_2'(\xi_i) + \omega_3\phi_3'(\xi_i), \tag{5.56}$$

with

$$\omega_j = \frac{\tilde{\omega}_j}{\tilde{\omega}_1 + \tilde{\omega}_2 + \tilde{\omega}_3}, \ \tilde{\omega}_j = \frac{d_j}{(\epsilon + \beta_j)^2}. \tag{5.57}$$

Usually ϵ is set to 10^{-6}.

Derivatives at the Half-Integer Nodes

We can extend this procedure to calculate derivatives at the half-integer nodes.

Problem 5 (Numerical Differentiation at the Half-Integer Nodes)**.** *Given the point values of a function ϕ on an equidistant grid $\{\xi_i, i = 1, \ldots, n\}$, calculate the derivative $\phi'(\xi_{i+\frac{1}{2}})$.*

The finite difference approximation to $\phi'(\xi_{i+\frac{1}{2}})$ is

$$\text{to second order: } \phi'(\xi_{i+\frac{1}{2}}) = \frac{1}{\delta\xi}\left(-\phi_i + \phi_{i+1}\right), \tag{5.58}$$

$$\text{to fourth order: } \phi'(\xi_{i+\frac{1}{2}}) = \frac{1}{24\,\delta\xi}\left(\phi_{i-1} - 27\phi_i + 27\phi_{i+1} - \phi_{i+2}\right). \tag{5.59}$$

In the WENO context, this problem can be handled in a similar way as Problem 4. First we consider only the two stencils $S_1 = \{\xi_{i-1}, \xi_i, \xi_{i+1}\}$ and $S_2 = \{\xi_i, \xi_{i+1}, \xi_{i+2}\}$. But for both stencils,

$$\phi'_{1/2}(\xi_{i+\frac{1}{2}}) = -\phi_i + \phi_{i+1}. \tag{5.60}$$

Therefore, this approach is equivalent to using central differences of second order.

If the two stencils $S_0 = \{\xi_{i-2}, \xi_{i-1}, \xi_i\}$ and $S_3 = \{\xi_{i+1}, \xi_{i+2}, \xi_{i+3}\}$ are added, we get the additional polynomials

$$\phi'_0(\xi_{i+\frac{1}{2}}) = \phi_{i-2} - 3\phi_{i-1} + 2\phi_i, \tag{5.61}$$

$$\phi'_3(\xi_{i+\frac{1}{2}}) = -2\phi_{i+1} + 3\phi_{i+2} - \phi_{i+3}. \tag{5.62}$$

On the large stencil

$$S = \{\xi_{i-2}, \xi_{i-1}, \xi_i, \xi_{i+1}, \xi_{i+2}, \xi_{i+3}\}, \tag{5.63}$$

the numerical derivative is

$$\phi'(\xi_{i+\frac{1}{2}}) = \frac{-9\phi_{i-2} + 125\phi_{i-1} - 2250\phi_i + 2250\phi_{i+1} - 125\phi_{i+2} + 9\phi_{i+3}}{1920}. \tag{5.64}$$

But the linear weights cannot be deduced from comparison with the interpolation polynomial on S since ξ_{i-1} and ξ_{i+2} are not used in $\phi'_{1,2}(\xi_{i+\frac{1}{2}})$.

If we consider stencils of width four, we get three interpolation polynomials on the stencils $S_1 = \{\xi_{i-2}, \xi_{i-1}, \xi_i, \xi_{i+1}\}$, $S_2 = \{\xi_{i-1}, \xi_i, \xi_{i+1}, \xi_{i+2}\}$ and $S_3 = \{\xi_i, \xi_{i+1}, \xi_{i+2}, \xi_{i+3}\}$. These are

$$\phi_1'(\xi_{i+\frac{1}{2}}) = \frac{\phi_{i-2} - 3\phi_{i-1} - 21\phi_i + 23\phi_{i+1}}{24}, \tag{5.65}$$

$$\phi_2'(\xi_{i+\frac{1}{2}}) = \frac{\phi_{i-1} - 27\phi_i + 27\phi_{i+1} - \phi_{i+2}}{24}, \tag{5.66}$$

$$\phi_3'(\xi_{i+\frac{1}{2}}) = \frac{-23\phi_i + 21\phi_{i+1} + 3\phi_{i+2} - \phi_{i+3}}{24}. \tag{5.67}$$

From the numerical approximation on the broad stencil

$$S = \{\xi_{i-2}, \xi_{i-1}, \xi_i, \xi_{i+1}, \xi_{i+2}, \xi_{i+3}\} \tag{5.68}$$

we deduce the linear weights

$$d_1 = d_3 = \frac{3}{16}, d_2 = \frac{5}{8}. \tag{5.69}$$

The smoothness indicators are defined by

$$\beta_j = \sum_{l=2}^{3} (\delta\xi)^{2l-1} \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i-\frac{1}{2}}} \left(\frac{d^l}{d\xi^l} \phi(\xi) \right)^2 d\xi. \tag{5.70}$$

We calculate

$$\begin{aligned} \beta_1 = \frac{1}{12}\big(&13\phi_{i-2}^2 - 78\phi_{i-2}\phi_{i-1} + 129\phi_{i-1}^2 + 78\phi_{i-2}\phi_i - 282\phi_{i-1}\phi_i \\ &+165\phi_i^2 - 26\phi_{i-2}\phi_{i+1} + 102\phi_{i-1}\phi_{i+1} - 126\phi_i\phi_{i+1} + 25\phi_{i+1}^2\big), \end{aligned} \tag{5.71}$$

$$\begin{aligned} \beta_2 = \frac{1}{12}\big(&25\phi_{i-1}^2 - 126\phi_{i-1}\phi_i + 165\phi_i^2 + 102\phi_{i-1}\phi_{i+1} - 282\phi_i\phi_{i+1} \\ &+129\phi_{i+1}^2 - 26\phi_{i-1}\phi_{i+2} + 78\phi_i\phi_{i+2} - 78\phi_{i+1}\phi_{i+2} + 13\phi_{i+2}^2\big), \end{aligned} \tag{5.72}$$

$$\begin{aligned} \beta_3 = \frac{1}{12}\big(&61\phi_i^2 - 318\phi_i\phi_{i+1} + 417\phi_{i+1}^2 + 270\phi_i\phi_{i+2} - 714\phi_{i+1}\phi_{i+2} \\ &+309\phi_{i+2}^2 - 74\phi_i\phi_{i+3} + 198\phi_{i+1}\phi_{i+3} - 174\phi_{i+2}\phi_{i+3} + 25\phi_{i+3}^2\big). \end{aligned} \tag{5.73}$$

This approach has the advantage of yielding non-oscillatory results near discontinuities, whereas the finite difference formulae (5.59) will produce overshoot. In the numerical simulations in this thesis, however, the procedure by Happenhofer et al. (2013) is used for discretisation of parabolic terms. If all coefficient functions are sufficiently smooth, the analytical solution of parabolic equations is smooth even if the initial data is non-smooth (Evans, 2002), and therefore high-order finite difference approaches without any limiting procedure are a suitable method to integrate these terms. For calculating the metric derivatives, the formulae (5.81) should be used, as we will demonstrate in section 5.4.

5.4 Numerical Consequences

When implementing curvilinear coordinates in an existing code written for Cartesian coordinates, the accuracy and stability of the numerical scheme can be deteriorated if the implementation is not done suitably. We show the theoretical background for some of these problems in the following, and illustrate them by numerical data in the next section.

5.4.1 The Freestream Problem

Nonomura et al. (2010) and Colella et al. (2011) emphasize the importance of the following, seemingly simple, test problem.

Problem 6 (Freestream preservation for the Euler equations)**.** *Given the initial conditions*

$$(\rho, \rho u, \rho v, E) = (\rho_0, 0, 0, e_0)\,, \tag{5.74}$$

with constant density ρ_0 and internal energy e_0, the numerical solution of the transformed system (5.4) *should stay close to the analytical solution*

$$(\rho, \rho u, \rho v, E) = (\rho_0, 0, 0, e_0) \tag{5.75}$$

for all times. $p_0 = p(\rho_0, e_0)$ is the pressure given by the equation of state.

Plugging these initial conditions into the transformed Euler equations (5.4), we get

$$0 = p_0 \left(\frac{\partial n_1}{\partial \xi} + \frac{\partial m_1}{\partial \eta} \right),\ 0 = p_0 \left(\frac{\partial n_2}{\partial \xi} + \frac{\partial m_2}{\partial \eta} \right), \tag{5.76}$$

since $\frac{\partial}{\partial t}(\rho u) = \frac{\partial}{\partial t}(\rho v) = 0$. This condition must be fulfilled numerically in order to prevent numerical errors. Since

$$n_1 = \frac{\partial y}{\partial \eta}, n_2 = -\frac{\partial x}{\partial \eta}, m_1 = -\frac{\partial y}{\partial \xi}, m_2 = \frac{\partial x}{\partial \xi}, \tag{5.77}$$

this is equivalent to the requirement that the second derivatives of the mapping function M are symmetric. Every function which is twice (weakly) differentiable fulfils this property (i.e. all mapping functions for which the strong derivation in section 5.1 holds, definitely do so as well).

Finite volume and finite difference discretisation

Next, we investigate if the freestream is preserved by the finite difference and the finite volume discretisation of the Euler equations. The WENO finite difference scheme as summarised in Algorithm 5 corresponds to the scheme `WENO-G` in Nonomura et al. (2010). In the cited reference, they describe precisely why the finite difference scheme does not fulfil the freestream preservation property. The fluxes $\hat{\mathbf{F}}$ and $\hat{\mathbf{G}}$ are reconstructed

directly from their value at the cell centre, including the metric terms evaluated at the cell centre. These fluxes are not constant for the freestream initial conditions. The reconstructed fluxes at the cell boundary will not be constant, too, and be different on any cell boundary. Therefore, they will not cancel out, and steadily, a numerical error is introduced.

We note that Nonomura et al. (2010) introduced the `WENO-C` scheme as a different WENO finite difference scheme which fulfils the freestream property by calculating $\frac{\partial \mathbf{Q}}{\partial t}$ via

$$\frac{\partial \mathbf{Q}}{\partial t} = \frac{\partial \xi}{\partial x}\frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial \mathbf{F}}{\partial \eta} + \frac{\partial \xi}{\partial y}\frac{\partial \mathbf{G}}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial \mathbf{G}}{\partial \eta}. \tag{5.78}$$

We did not consider this scheme here since it is not conservative and its computational costs are three times higher than `WENO-G` in three dimensions, making the scheme useless for our purposes.

On the contrary, for the WENO finite volume scheme as summarised in Algorithm 6, the state vector $\mathbf{Q}$ is reconstructed at the cell boundaries from its cell averages. Therefore, for Problem 6 the reconstruction process will yield constant approximations for the state vector at the cell interface, i.e.

$$\mathbf{Q}_{i\pm\frac{1}{2},j} = \mathbf{Q}_{i,j\pm\frac{1}{2}} = (\rho_0, 0, 0, e_0)^{\mathrm{T}}. \tag{5.79}$$

In the update step (5.14), only the metric terms will be non-constant. In precise terms, the conditions

$$\frac{n_1|_{i+\frac{1}{2},j} - n_1|_{i-\frac{1}{2},j}}{\delta\xi} + \frac{m_1|_{i,j+\frac{1}{2}} - m_1|_{i,j-\frac{1}{2}}}{\delta\eta} = 0, \tag{5.80a}$$

$$\frac{n_2|_{i+\frac{1}{2},j} - n_2|_{i-\frac{1}{2},j}}{\delta\xi} + \frac{m_2|_{i,j+\frac{1}{2}} - m_2|_{i,j-\frac{1}{2}}}{\delta\eta} = 0, \tag{5.80b}$$

are equivalent to preserving the freestream. If the metric terms are calculated, as described in section 5.2, by

$$n_1|_{i\pm\frac{1}{2},j} = \frac{\partial y}{\partial \eta}|_{i\pm\frac{1}{2},j} = \frac{y_{i\pm\frac{1}{2},j+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2}}}{\delta\eta}, \tag{5.81a}$$

$$n_2|_{i\pm\frac{1}{2},j} = -\frac{\partial x}{\partial \eta}|_{i\pm\frac{1}{2},j} = -\frac{x_{i\pm\frac{1}{2},j+\frac{1}{2}} - x_{i\pm\frac{1}{2},j-\frac{1}{2}}}{\delta\eta}, \tag{5.81b}$$

$$m_1|_{i,j\pm\frac{1}{2}} = -\frac{\partial y}{\partial \xi}|_{i,j\pm\frac{1}{2}} = -\frac{y_{i+\frac{1}{2},j\pm\frac{1}{2}} - y_{i-\frac{1}{2},j\pm\frac{1}{2}}}{\delta\xi}, \tag{5.81c}$$

$$m_2|_{i,j\pm\frac{1}{2}} = \frac{\partial x}{\partial \xi}|_{i,j\pm\frac{1}{2}} = \frac{x_{i+\frac{1}{2},j\pm\frac{1}{2}} - x_{i-\frac{1}{2},j\pm\frac{1}{2}}}{\delta\xi}, \tag{5.81d}$$

conditions (5.80) are fulfilled exactly and the freestream will be preserved numerically.

We note that the freestream is never preserved in general if analytical expressions, if available, for the metric terms are used (p. 118, Thompson et al., 1985).

We remark that even though conditions (5.81) look like second-order approximations, they are rather analytical requirements which must be fulfilled by the discretisation of the metric terms in order to preserve the freestream. They are a consequence of the conservative discretisation of the derivatives in equations (3.74). As described in section 3.1, the discrete formulations of the Euler equations as in (3.7) and (3.10) are analytically equivalent to the continuous one in equation (3.3).

For trivial mapping functions such as $M: [-1,1]^2 \to [-1,1]^2$, $M(\xi,\eta) = (\xi,\eta)^{\mathrm{T}}$ with $\frac{\partial y}{\partial \eta} = \frac{\partial x}{\partial \xi} = \text{const}$, $\frac{\partial y}{\partial \xi} = \frac{\partial x}{\partial \eta} = 0$, freestream preservation is of course possible even for the finite difference scheme `WENO-G`.

We conclude that the mapped grid technique should be used only with the WENO finite volume scheme and with the metric terms calculated by (5.80). Non-preservation of the freestream leads to inacceptable errors, as we will demonstrate in the following section. The WENO finite difference scheme cannot preserve the freestream and be conservative at the same time.

Dimensionally Split Time Integration

In Colella and Woodward (1984) and section 3.3, the PPM algorithm to solve the one-dimensional Euler equations, as it is implemented in the finite volume code *Prometheus* (Müller et al., 1991), is described. In theren, the PPM algorithm is used to reconstruct the input for an exact Riemann solver from the given cell averages. The time integration is second order accurate. To extend this method to higher dimensions, Warming and Beam (1976) suggested the second-order split scheme

$$\mathbf{Q}^{(n+2)} = \mathrm{L}_\xi \mathrm{L}_\eta \mathrm{L}_\eta \mathrm{L}_\xi \mathbf{Q}^{(n)}, \tag{5.82}$$

written here for the two-dimensional case. L_ξ means application of the PPM scheme in ξ direction, and L_η application in η direction.

As described in section 3.3, the resulting algorithm is very stable and accurate on Cartesian grids. Unsplit schemes where all information in all directions is updated simultaneously are less stable or more complicated, as the CTU scheme from Colella (1990). In the curvilinear framework, however, the split scheme leads to violation of the the freestream initial condition as defined in Problem 6. To preserve the freestream, exact cancellation of the metric terms as in equation (5.80) is required. This can be achieved by discretising the metric terms by equations (5.81) and simultaneously updating all fluxes as described in the above paragraph.

When the fluxes are updated consecutively, applying L_ξ to $\hat{\mathbf{Q}}_0$ will produce numerical fluxes $\hat{\mathbf{F}}_{i+\frac{1}{2},j}$ of the form

$$\hat{\mathbf{F}}_{i+\frac{1}{2},j} \approx \begin{pmatrix} 0 \\ n_1|_{i+\frac{1}{2},j,k} p_0 \\ n_2|_{i+\frac{1}{2},j,k} p_0 \end{pmatrix}. \tag{5.83}$$

With these fluxes, $\hat{\mathbf{Q}}^\star$ defined by

$$\hat{\mathbf{Q}}^\star_{i,j} = \mathrm{L}_\xi \hat{\mathbf{Q}}_0|_{i,j} = \hat{\mathbf{Q}}_0|_{i,j} - \frac{\delta t}{\Omega_{i,j}} \left(\hat{\mathbf{F}}_{i+\frac{1}{2},j} - \hat{\mathbf{F}}_{i-\frac{1}{2},j} \right) \tag{5.84}$$

where $\Omega_{i,j} = J^{-1}_{i,j}$ is the cell volume, will not be constant unless n_1 and n_2 are constant. When L_η is applied to $\hat{\mathbf{Q}}^\star$, the consistency relation is violated since the pressure is not constant anymore.

We therefore refrain from using the split time integration schemes from Warming and Beam (1976) in the curvilinear framework. Instead, we look for unsplit schemes where freestream preservation can be achieved simply by using equations (5.81) for discretisation of the metric derivatives. Any Runge–Kutta scheme fulfils this property as described in the previous paragraph. For the CTU scheme as outlined in section 3.3, more changes are necessary as we will show in the following.

We check the changes for curvilinear coordinates in the two-dimensional CTU scheme as described in Colella (1990). The CTU algorithm is a predictor–corrector scheme. As in the Cartesian case, the values at the new time step for the curvilinear Euler equations (5.4) in two dimensions are obtained by

$$\mathbf{Q}^{(n+1)}_{i,j} = \mathbf{Q}^{(n)}_{i,j} - \frac{\delta t}{J} \left(\hat{\mathbf{F}}^{(n+\frac{1}{2})}_{i+\frac{1}{2},j} - \hat{\mathbf{F}}^{(n+\frac{1}{2})}_{i-\frac{1}{2},j} \right) - \frac{\delta t}{J} \left(\hat{\mathbf{G}}^{(n+\frac{1}{2})}_{i,j+\frac{1}{2}} - \hat{\mathbf{G}}^{(n+\frac{1}{2})}_{i,j-\frac{1}{2}} \right), \tag{5.85}$$

with the predictions $\mathbf{Q}^{(n+\frac{1}{2})}_{i\pm\frac{1}{2},j}$ and $\mathbf{Q}^{(n+\frac{1}{2})}_{i,j\pm\frac{1}{2}}$ at an intermediate time level. The construction of the predictive variables starts with the extrapolation formula

$$\mathbf{Q}^{(n+\frac{1}{2})}_{i\pm\frac{1}{2},j} = \mathbf{Q}^{(n)}_{i,j} \pm \frac{\delta\xi}{2} \frac{\partial \mathbf{Q}}{\partial \xi} + \frac{\delta t}{2} \frac{\partial \mathbf{Q}}{\partial t}, \tag{5.86}$$

analogously to equation (3.78). From the definition of the curvilinear Euler equations, we get

$$\frac{\partial \mathbf{Q}}{\partial t} = - J^{-1} \left(\frac{\partial \hat{\mathbf{F}}}{\partial \xi} + \frac{\partial \hat{\mathbf{G}}}{\partial \eta} \right), \tag{5.87}$$

$$\frac{\partial \hat{\mathbf{F}}}{\partial \xi} = \frac{\partial \left(n_1 \mathbf{F} + n_2 \mathbf{G} \right)}{\partial \xi} = \frac{\partial n_1}{\partial \xi} \mathbf{F} + n_1 \frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial n_2}{\partial \xi} \mathbf{G} + n_2 \frac{\partial \mathbf{G}}{\partial \xi}. \tag{5.88}$$

With the chain rule,

$$\frac{\partial \mathbf{F}}{\partial \xi} = \frac{\partial \mathbf{F}}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial \xi}, \quad \frac{\partial \mathbf{G}}{\partial \xi} = \frac{\partial \mathbf{G}}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial \xi}. \tag{5.89}$$

Introducing

$$\mathbf{A}_\xi = n_1 \frac{\partial \mathbf{F}}{\partial \mathbf{Q}} + n_2 \frac{\partial \mathbf{G}}{\partial \mathbf{Q}}, \tag{5.90}$$

the equation (5.86) transforms to

$$\mathbf{Q}_{i\pm\frac{1}{2},j}^{(n+\frac{1}{2})} = \mathbf{Q}_{i,j}^{(n)} + \left(\pm\frac{1}{2} - \frac{\delta t}{2J}\mathbf{A}_\xi\right)\frac{\partial \mathbf{Q}}{\partial \xi} - \frac{\delta t}{2J}\left(\frac{\partial n_1}{\partial \xi}\mathbf{F} + \frac{\partial n_2}{\partial \xi}\mathbf{G} + \frac{\partial \hat{\mathbf{G}}}{\partial \eta}\right), \tag{5.91}$$

analogously to equation (3.81). In Colella (1990), all terms on the right-hand side of (5.91) are evaluated at the cell centre. In our implementation, we use the result of the Riemann solver with input $\mathbf{Q}_{i,j}^{(n)}$ for the term $\mathbf{Q}_{i,j}^{(n)} + \left(\pm\frac{1}{2} - \frac{\delta t}{2J}\mathbf{A}_\xi\right)\frac{\partial \mathbf{Q}}{\partial \xi}$. All other terms are evaluated at the cell centre, at (i, j) for the left states and at $(i+1, j)$ for the right states.

The terms $\frac{\partial n_1}{\partial \xi}\mathbf{F}$ and $\frac{\partial n_2}{\partial \xi}\mathbf{G}$ are 0 for Cartesian grids. They are needed to fulfil the consistency requirement (5.80). In the CTU algorithm and its three-dimensional extension from Gardiner and Stone (2008), they must be added as source terms C in step (3.83) to the corrected states in conservation form:

$$\begin{aligned}
c_\xi &= -\frac{\delta t}{2J^{-1}}\left(n_1|_{i+\frac{1}{2},j,k} - n_1|_{i-\frac{1}{2},j,k}\right), \\
c_\eta &= -\frac{\delta t}{2J^{-1}}\left(n_2|_{i+\frac{1}{2},j,k} - n_2|_{i-\frac{1}{2},j,k}\right), \\
c_\zeta &= -\frac{\delta t}{2J^{-1}}\left(n_3|_{i+\frac{1}{2},j,k} - n_3|_{i-\frac{1}{2},j,k}\right),
\end{aligned} \tag{5.92}$$

$$\begin{aligned}
C_{\xi,i,j,k,\rho} &= c_\xi \rho_{i,j,k} u_{i,j,k} \\
&\quad + c_\eta \rho_{i,j,k} v_{i,j,k} + c_\zeta \rho_{i,j,k} w_{i,j,k}, \\
C_{\xi,i,j,k,\rho u} &= c_\xi \rho_{i,j,k} \left(\rho_{i,j,k} u_{i,j,k} u_{i,j,k} + p_{i,j,k}\right) \\
&\quad + c_\eta \rho_{i,j,k} \rho_{i,j,k} u_{i,j,k} v_{i,j,k} + c_\zeta \rho_{i,j,k} \rho_{i,j,k} u_{i,j,k} w_{i,j,k}, \\
C_{\xi,i,j,k,\rho v} &= c_\xi \rho_{i,j,k} \rho_{i,j,k} v_{i,j,k} u_{i,j,k} \\
&\quad + c_\eta \rho_{i,j,k} \left(\rho_{i,j,k} v_{i,j,k} v_{i,j,k} + p_{i,j,k}\right) + c_\zeta \rho_{i,j,k} \rho_{i,j,k} v_{i,j,k} w_{i,j,k}, \\
C_{\xi,i,j,k,\rho w} &= c_\xi \rho_{i,j,k} \rho_{i,j,k} w_{i,j,k} u_{i,j,k} \\
&\quad + c_\eta \rho_{i,j,k} \rho_{i,j,k} w_{i,j,k} v_{i,j,k} + c_\zeta \rho_{i,j,k} \left(\rho_{i,j,k} w_{i,j,k} w_{i,j,k} + p_{i,j,k}\right), \\
C_{\xi,i,j,k,E} &= c_\xi u_{i,j,k} \left(E_{i,j,k} + p_{i,j,k}\right) \\
&\quad + c_\eta v_{i,j,k} \left(E_{i,j,k} + p_{i,j,k}\right) + c_\zeta w_{i,j,k} \left(E_{i,j,k} + p_{i,j,k}\right).
\end{aligned} \tag{5.93}$$

For the right states, the source terms are evaluated at $i + 1$. Analogous expressions hold for the other directions. We note that c_ξ, c_η and c_ζ are 0 for Cartesian coordinates. Apart from that, the CTU algorithm as described in section 3.3 can be applied directly

to the curvilinear equations (5.4) by replacing x and y by ξ and η.

Concerning the implementation, the CTU scheme requires much more memory than the split scheme. In the split scheme, the only multidimensional arrays needed are the ones containing the conserved or primitive variables. For any unsplit scheme, the numerical flux in each direction must be stored in a separate array for the update step as step (3.85) in the CTU scheme. This results in $3 \cdot n_{\text{conserved}}$ additional 3D arrays, where $n_{\text{conserved}}$ is the number of conserved quantities. This can be reduced to $n_{\text{conserved}}$ arrays by adding up the fluxes after each directional sweep.

If the state vectors calculated in (3.82) are saved for the reuse in (3.83), $2 \cdot n_{\text{conserved}} \cdot 3$ additional 3D arrays are needed. Since this is certainly too much, the state vectors $Q^{\star}_{L,x,i+\frac{1}{2},j,k}$ must be recalculated in (3.83) increasing the computational requirements. Then, the flux arrays used to store the fluxes calculated in (3.82) cannot be overwritten since steps (3.83) and (3.84) are executed simultaneously.

At the boundaries, for the leftmost left state and the rightmost right state, the transversal fluxes needed in step (3.83) are not available (see Figure 5.2). For most boundary conditions, the fluxes from the neighbouring cell can be used instead.

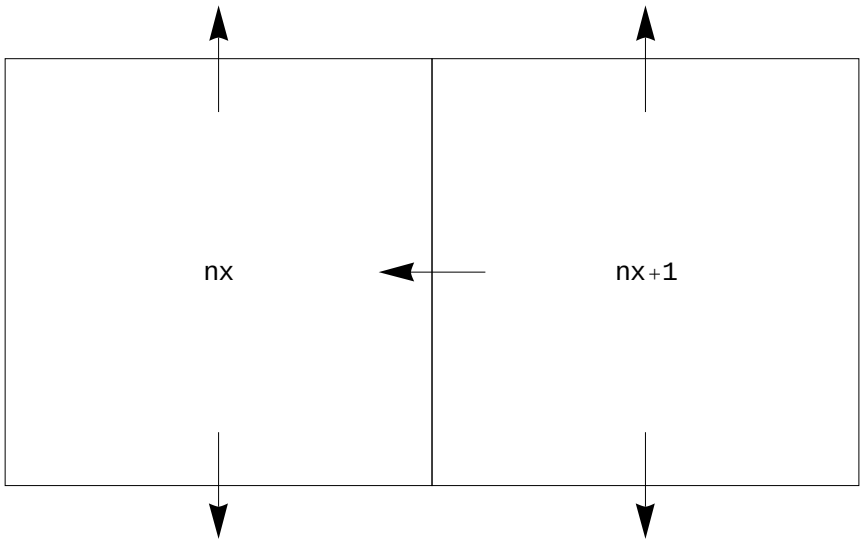

Figure 5.2: Schematic view of the outermost grid cell called `nx`. For the prediction of the states in the CTU scheme, the transversal fluxes at `nx+1` are needed. For all boundary conditions where the transversal velocities in the boundary layers are obtained by constant extrapolation, the transversal corrections for the states at the interface between `nx` and `nx+1` can therefore be obtained by using the values for the outermost grid cell `nx`.

The intermediate update from step (3.83) should be applied to the conservative state variables. In the state vector $\mathbf{S}^{\star}_{L,x,i+\frac{1}{2},j,k}$, only the primitive variables $1/\rho$, u, v, w and

p are stored. They have to be converted to ρ, ρu, ρv, ρw and E at the beginning, and back to the primitive form at the end of the update.

For the Riemann solver, the velocity vector $\mathbf{u}$ given in Cartesian components u, v and w must be transformed to normal and two tangential components u_N, u_T and u_{TT} in the local coordinate system on the cell surface (Colella and Woodward, 1984; Colella, 1990). The local coordinate system in three dimensions is defined by the vector $\mathbf{n} = (n_1, n_2, n_3)^{\mathrm{T}}$ resp. $\mathbf{m} = (m_1, m_2, m_3)^{\mathrm{T}}$ or $\mathbf{o} = (o_1, o_2, o_3)^{\mathrm{T}}$. The precise form of these vectors in three dimensions can be found in section 5.6. In the following, we only consider rotation in the coordinate system defined by $\mathbf{n}$.

According to Wongwathanarat (2013), the rotation is given by composition of two rotations around the η and ζ axis. Their rotation matrices are given by

$$R_\zeta(\phi) = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{5.94a}$$

$$R_\eta\left(-(\frac{\pi}{2} - \theta)\right) = \begin{pmatrix} \sin\theta & 0 & \cos\theta \\ 0 & 1 & 0 \\ -\cos\theta & 0 & \sin\theta \end{pmatrix}. \tag{5.94b}$$

The angles ϕ and θ can be obtained by

$$\sin\theta = \sqrt{n_1^2 + n_2^2},\ \cos\theta = n_3,\ \sin\phi = \frac{n_2}{\sqrt{n_1^2 + n_2^2}},\ \cos\phi = \frac{n_1}{\sqrt{n_1^2 + n_2^2}}. \tag{5.95}$$

Finally,

$$\begin{pmatrix} u_N \\ u_T \\ u_{TT} \end{pmatrix} = \begin{pmatrix} n_1 & n_2 & n_3 \\ -\frac{n_2}{\sqrt{n_1^2+n_2^2}} & \frac{n_1}{\sqrt{n_1^2+n_2^2}} & 0 \\ -\frac{n_1 n_3}{\sqrt{n_1^2+n_2^2}} & -\frac{n_2 n_3}{\sqrt{n_1^2+n_2^2}} & \sqrt{n_1^2 + n_2^2} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}. \tag{5.96}$$

This is a rotation matrix since its determinant is 1 and the inverse is given by the transpose. Note that in two dimensions, $n_3 = 0$ and the transformation is given by

$$\begin{pmatrix} u_N \\ u_T \\ u_{TT} \end{pmatrix} = \begin{pmatrix} n_1 & n_2 & 0 \\ -n_2 & n_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}. \tag{5.97}$$

5.5 Numerical Results

In the following, we illustrate the theoretical results from the preceeding sections by numerical simulations. First, we introduce several mapping functions.

5.5.1 Mapping Functions

There is no strongly differentiable function (a diffeomorphism) from the (unit) sphere to the (unit) square since the square is not a submanifold of $\mathbb{R}^2$. Therefore, if we want to use the mapped grid technique to perform simulations on spherical domains, we have to rely on mapping functions which are only weakly differentiable.

For the mapped grid technique, Calhoun et al. (2008) gave some examples of mapping functions which map a circular domain to $[-1,1]^2$ and vice versa. In this section, we want to investigate how the mapping functions M_1 and M_2 from Calhoun et al. (2008) defined by

$$M_1 : \ [-1,1]^2 \to \Omega, \ x = R \cdot \frac{\max\left(|\xi|, |\eta|\right) \xi}{\sqrt{\xi^2 + \eta^2}}, \tag{5.98a}$$

$$y = R \cdot \frac{\max\left(|\xi|, |\eta|\right) \eta}{\sqrt{\xi^2 + \eta^2}}, \tag{5.98b}$$

$$M_2 : \ [-1,1]^2 \to \Omega, \ w = \max\left(|\xi|, |\eta|\right)^2, \tag{5.98c}$$

$$x = w \cdot x_{M_1} + (1-w) \cdot \frac{R\xi}{\sqrt{2}}, \tag{5.98d}$$

$$y = w \cdot y_{M_1} + (1-w) \cdot \frac{R\eta}{\sqrt{2}}, \tag{5.98e}$$

where x_{M_1} and y_{M_1} are the physical coordinates defined by M_1, and

$$\Omega = \{(x,y) \in \mathbb{R}^2 : \sqrt{x^2 + y^2} \leq R\}, \tag{5.99}$$

perform in numerical simulations when WENO schemes and the PPM scheme with several time integration methods are employed.

It is obvious to see that these functions are only weakly differentiable. Therefore, they should be applied only in the context of the methods developed in section 5.2.

Besides the mapping functions M_1 and M_2, we use the mapping functions

$$M_3 : \ [-1,1]^2 \to \Omega, \quad \begin{aligned} x &= -R + 2R \cdot \left(\xi + 0.1 \sin\left(2\pi\xi\right) \sin\left(2\pi\eta\right)\right), \\ y &= -R + 2R \cdot \left(\eta + 0.1 \sin\left(2\pi\xi\right) \sin\left(2\pi\eta\right)\right), \end{aligned} \tag{5.100}$$

$$M_4 : \ [-1,1]^2 \to \Omega, \quad \begin{aligned} x &= -R + 2R \cdot \xi, \\ y &= -R + 2R \cdot \eta, \end{aligned} \tag{5.101}$$

where $\Omega = [-R,R]^2$. M_3 was used in Colella et al. (2011) to test the order of accuracy of their scheme since it is a smooth function, whereas M_4 gives a Cartesian grid. We will call a grid "smooth" if the mapping function defining this grid is smooth.

Remark 3. *We note that a polar grid is a special case of a mapped grid via the mapping function*

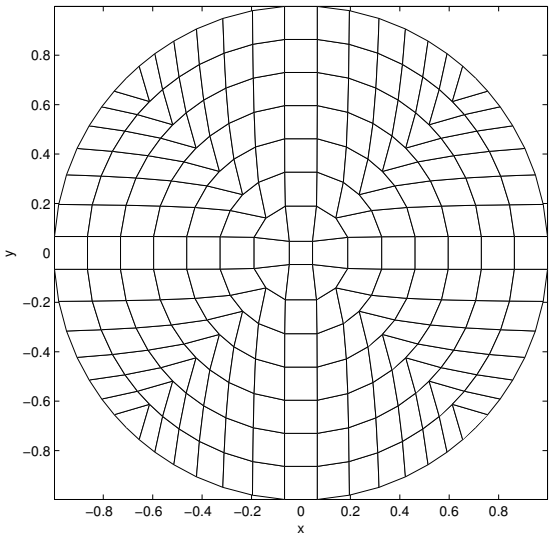

Figure 5.3: Grid defined by function M_1 from Calhoun et al. (2008).

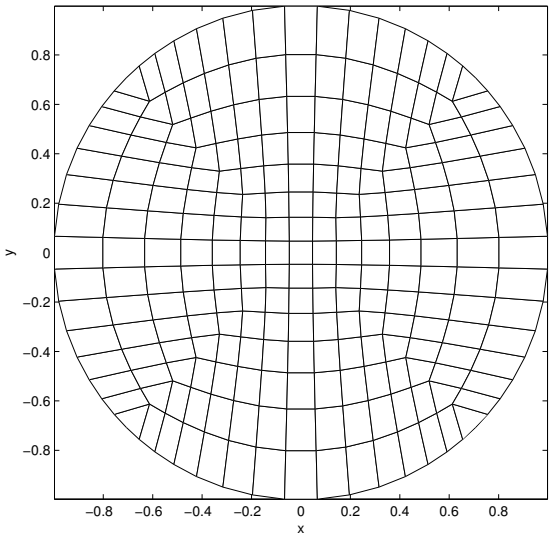

Figure 5.4: Grid defined by function M_2 from Calhoun et al. (2008).

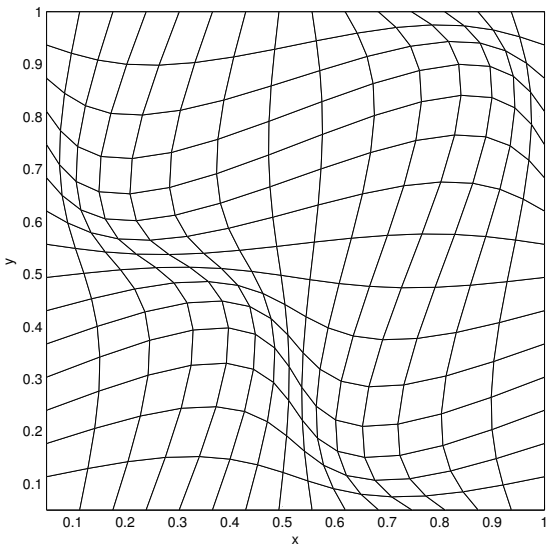

Figure 5.5: Grid defined by function M_3 from Colella et al. (2011).

$$M_{\text{polar}} \ [0,1]^2 \to \Omega, \quad \begin{aligned} x &= R_0 + R\,\xi \cos(2\pi\eta)\,, \\ y &= R_0 + R\,\xi \sin(2\pi\eta)\,, \end{aligned} \tag{5.102}$$

where R_0 is the radius of the innermost grid cell. $R_0 = 0$ leads to "degenerate" cells at the centre, i.e. cells which have one side with length 0. The difference to the classical polar grid is that the velocity vector is not given in the radial and the angular components, but still in the Cartesian ones. This has the advantage that geometrical source terms as in Mundprecht et al. (2013) are avoided. Instead, the transformed equations are in the strong conservation form (5.4).

5.5.2 ANTARES

All simulations in this subsection are performed with the code ANTARES (Muthsam et al., 2010a) using explicit time integration schemes and Marquina flux splitting (Donat and Marquina, 1996). If not stated otherwise, the Runge–Kutta scheme employed in the simulations is SSP RK(3,2), a second-order Runge–Kutta scheme with three stages (Kraaijevanger, 1991; Kupka et al., 2012). In all simulations, the ideal gas equation is used, and the Courant number is fixed to 0.1. The WENO finite difference scheme corresponds to the method `WENO-G` in Nonomura et al. (2010).

Gresho Vortex

The specific setup of the Gresho Vortex used in this paragraph is described in Happenhofer et al. (2013) and Miczek (2013). We repeat the definition here for convenience.

$$\rho = 1, \tag{5.103a}$$

$$p_0 = \frac{\rho}{\gamma \operatorname{Ma}_{\text{ref}}^2}, \tag{5.103b}$$

$$u_\phi = \begin{cases} 5r, & 0 \le r < 0.2, \\ 2 - 5r, & 0.2 \le r < 0.4, \\ 0, & 0.4 \le r, \end{cases} \tag{5.103c}$$

$$p = \begin{cases} p_0 + \frac{25}{2} r^2, & 0 \le r < 0.2, \\ p_0 + \frac{25}{2} r^2 + 4(1 - 5r - \ln 0.2 + \ln r), & 0.2 \le r < 0.4, \\ p_0 - 2 + 4 \ln 2, & 0.4 \le r. \end{cases} \tag{5.103d}$$

$r = \sqrt{x^2 + y^2}$ is the distance from the origin and u_ϕ the angular velocity in terms of the polar angle $\phi = \texttt{atan2}(y, x)$. Note that the difference to the setup as it is described in Liska and Wendroff (2003) is the introduction of a reference Mach number $\operatorname{Ma}_{\text{ref}}$. The pressure p is scaled such that the reference Mach number is the maximum Mach number of the resulting flow.

We performed a simulation with $\operatorname{Ma}_{\text{ref}} = 0.1$ and $\gamma = \frac{5}{3}$ over a time interval of 2 s corresponding to approximately 1.5 rotations of the vortex. The size of the domain is 1 cm in each direction, and we use 100×100 grid points. The analytical solution is pure angular rotation of the vortex. The results on the four grids defined by the mapping functions M_1, M_2, M_3 and M_4 are shown in Figure 5.6 for the finite difference scheme and in Figure 5.7 for the finite volume scheme.

With the non-smooth mapping functions M_1 and M_2, the results with the finite difference scheme are catastrophic, whereas with the finite volume scheme, the difference to the solution on the Cartesian grid is small. It is obvious that the problems come from the grid points where the mapping functions are not smooth. But even with the smooth mapping function M_3, the symmetry of the vortex is destroyed with the finite difference scheme, whereas there is no visible difference to the Cartesian solution with the finite volume scheme.

The Mach number $\operatorname{Ma}_{\text{ref}} = 0.1$ of this test is rather low for an explicit time integration scheme, but Happenhofer et al. (2013) showed that the WENO scheme with Runge–Kutta time integration yields reliable results in this regime. As shown in Figure 5.8, the deformations due to the grid get smaller with higher Mach numbers, but in any case, the results with the finite volume scheme are more accurate. On the other hand, this explains why applying the mapped grid technique to the WENO finite difference scheme in Shu (2003) worked properly: the mapping function were smooth, and the Mach number in the numerical tests was always larger than 1.

We conclude that the WENO finite difference scheme should only be used in combination with the mapped grid technique if the mapping function is smooth and if the Mach number of the simulation is high. The reason for the bad performance lies in the violation of the preservation of the freestream. The finite volume scheme, however,

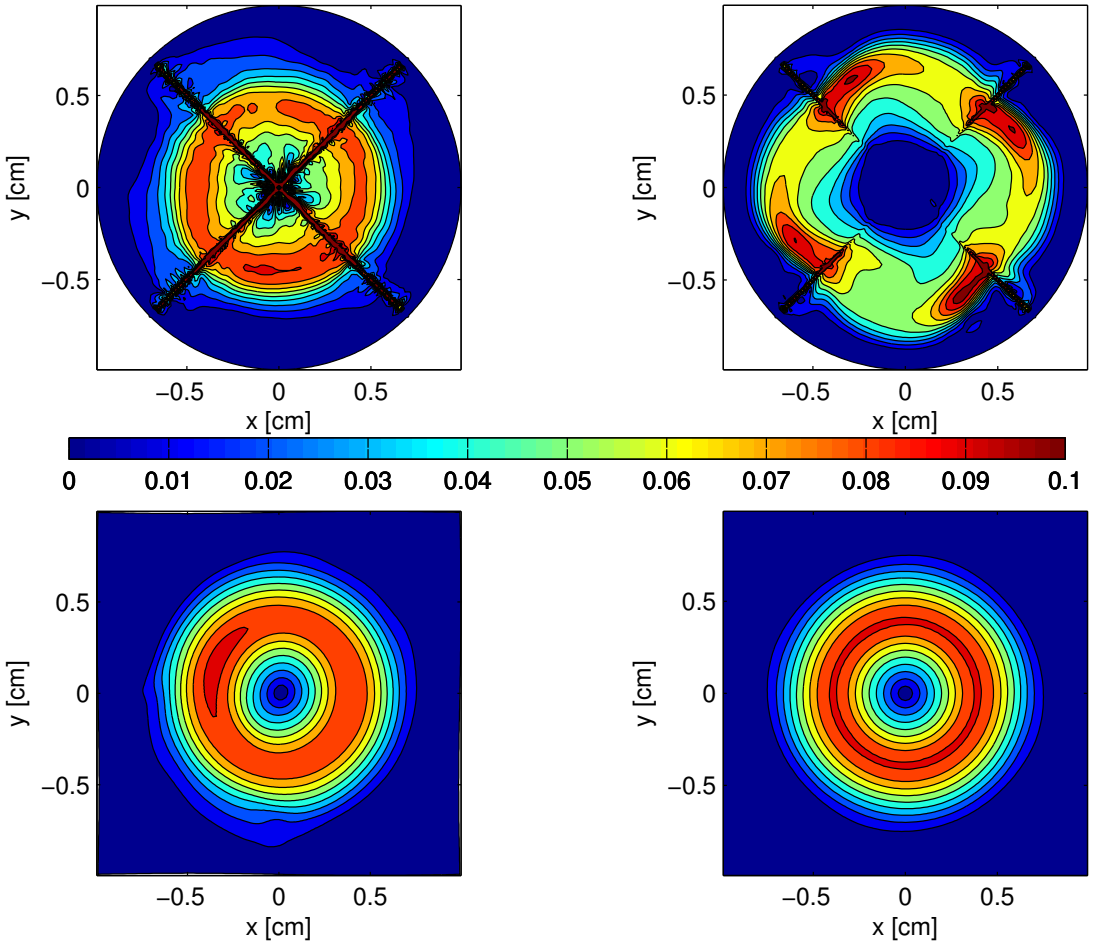

Figure 5.6: Mach number in the Gresho vortex test after 2 s with the WENO finite difference scheme. The initial Mach number was 0.1. In all figures, the results of the simulation with M_1 are shown in the top left panel, with M_2 in the top right, with M_3 in the bottom left and with M_4 in the bottom right panel.

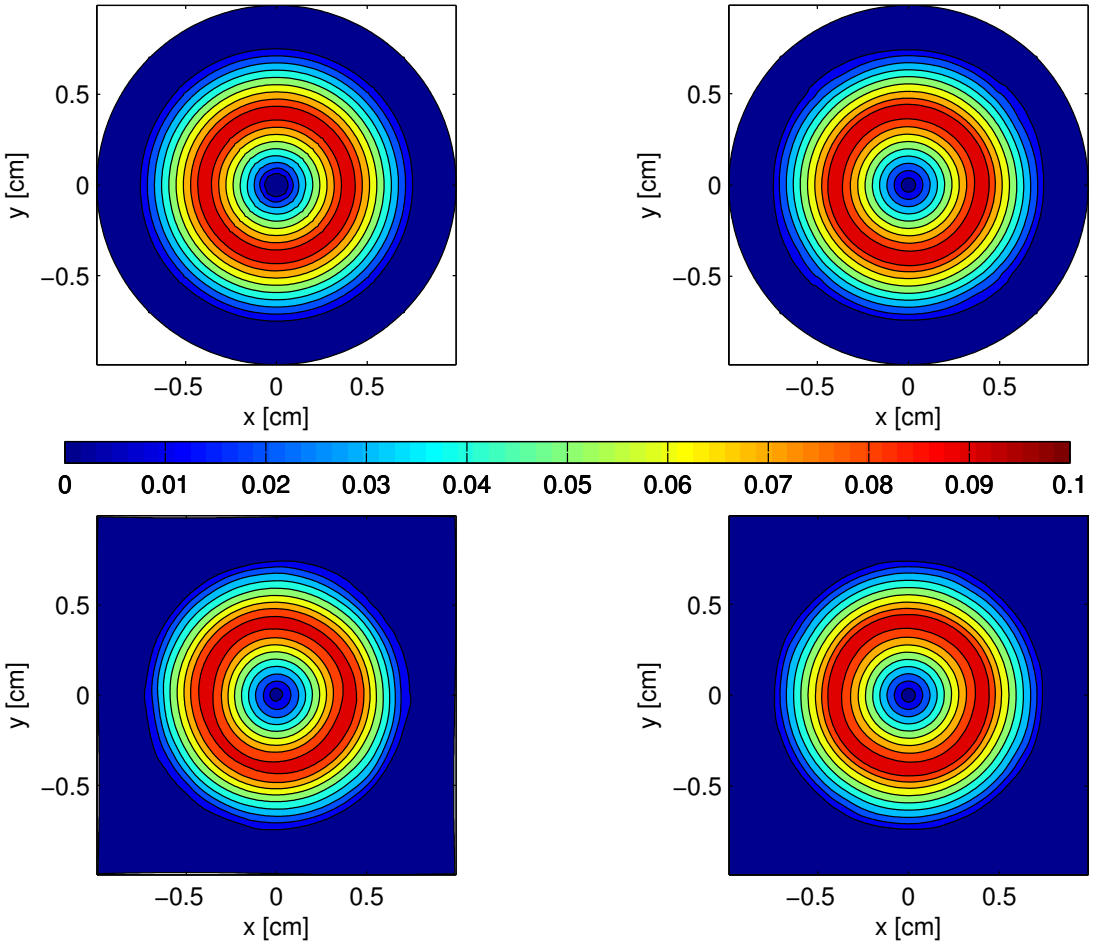

Figure 5.7: Mach number in the Gresho vortex test after 2 s with the WENO finite volume scheme. The initial Mach number was 0.1.

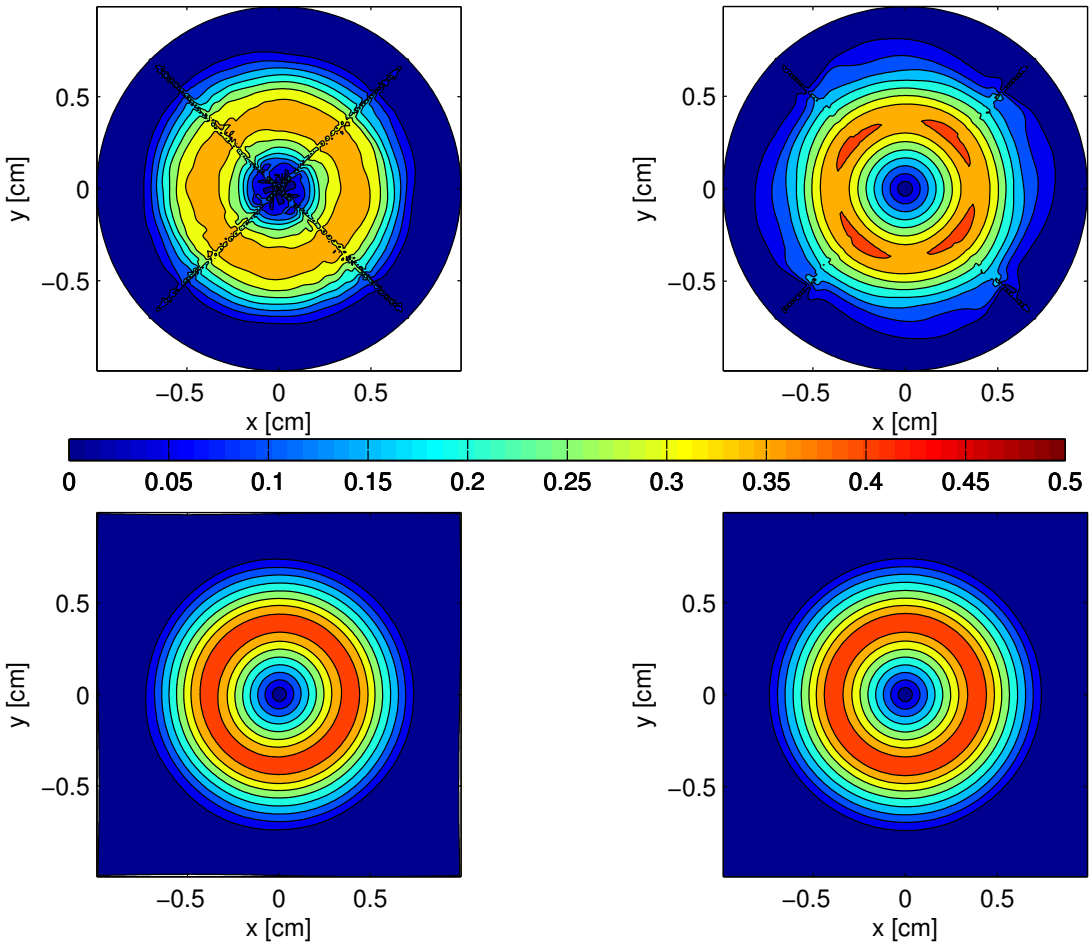

Figure 5.8: Mach number in the Gresho vortex test after 2 s with the WENO finite difference scheme with initial Mach number of 0.5.

yields accurate results even on non-smooth grids and in low Mach number tests.

Sod Shock Tube

With the Sod Shock Tube (Sod, 1978) we test the performance of our schemes in the high Mach number regime. The computational domain is $[-0,5,0.5]^2$ and $\gamma = 1.4$. In each direction, we use 100 grid points. The inital shock position is at 0.2 in x direction.

Even in this setup where the Mach number is rather high, the WENO finite difference scheme performs badly with the non-smooth mapping functions M_1 and M_2 as we can see in Figure 5.9. With M_1, the simulation crashed after 0.09 s because of negative densities. At the diagonals, numerical artifacts occur even without any flow present at this position as a consequence of the violation of the freestream preservation.

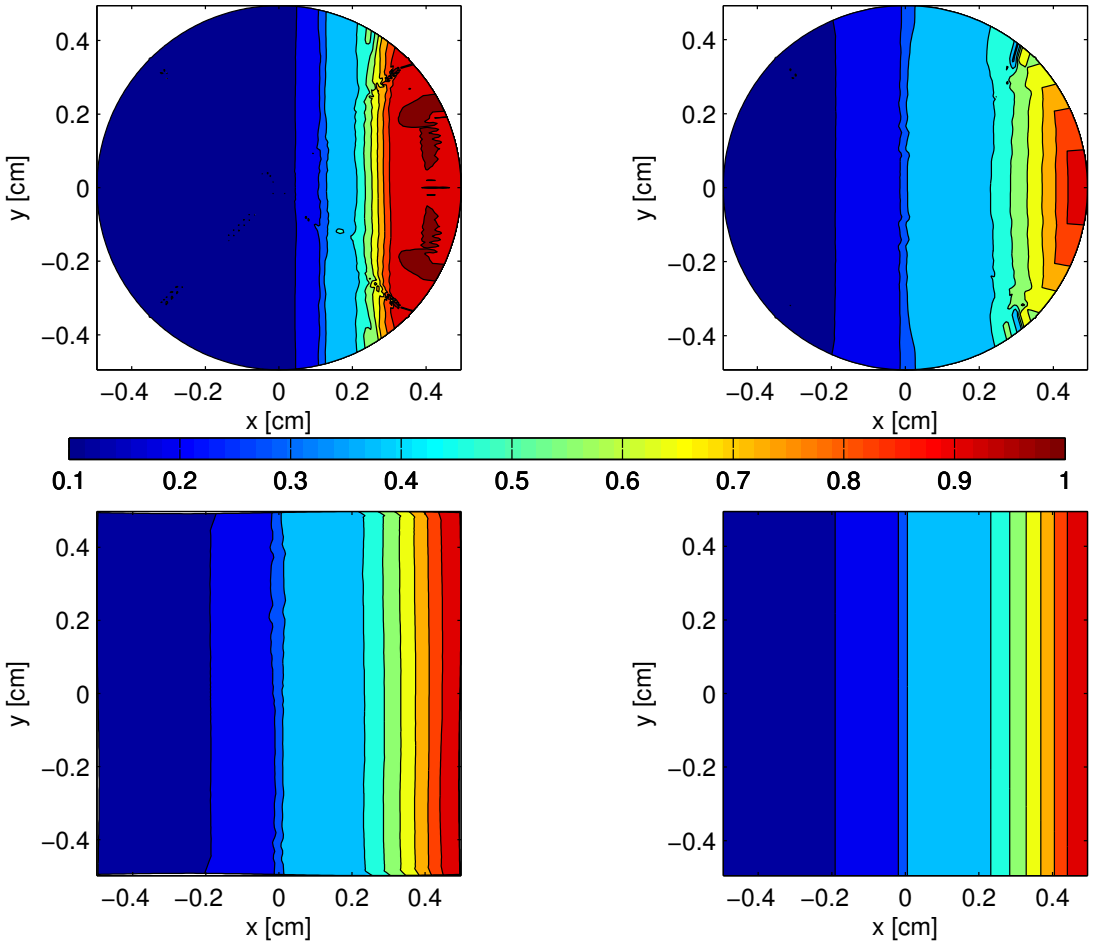

Figure 5.9: Density in the Sod Shocktube Test after 0.22 s with the WENO finite difference scheme. M_1 (top left panel) after 0.09 s. In the outermost three layers, outflow boundary conditions are set.

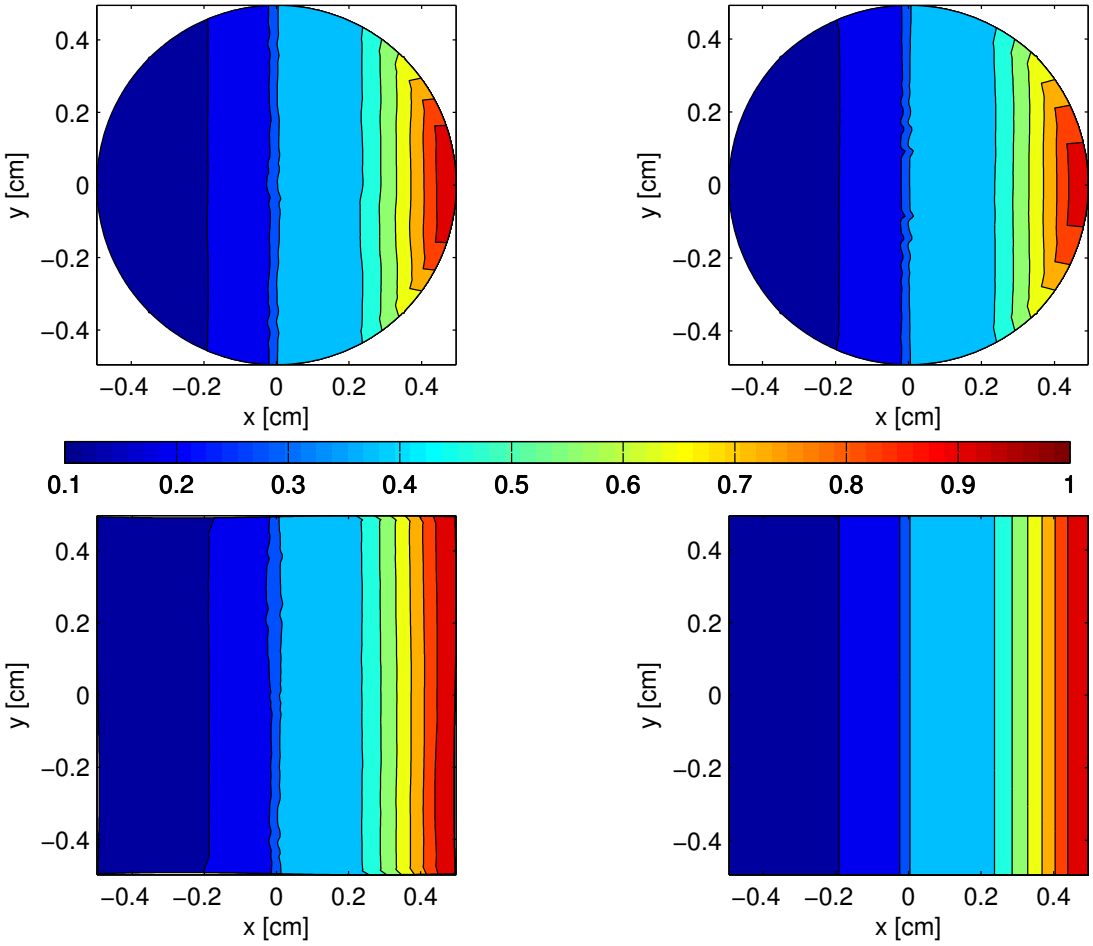

Figure 5.10: Density in the Sod Shocktube Test after 0.22 s with the WENO finite volume scheme. In the outermost three layers, outflow boundary conditions are set.

grid	M_1		M_2		M_3		M_4	
spacing	error	order	error	order	error	order	error	order
6.25e-1	7.74e-3		1.39e-3		2.77e-3		7.74e-4	
3.13e-1	5.89e-3	0.392	1.31e-3	0.088	4.76e-4	2.539	1.23e-4	2.652
1.56e-1	3.83e-3	0.621	9.29e-4	0.492	7.01e-5	2.764	1.05e-5	3.548
7.81e-2	2.95e-3	0.380	8.03e-4	0.209	1.20e-5	2.546	5.46e-7	4.270
3.91e-2	1.98e-3	0.572	5.43e-4	0.566	2.83e-6	2.087	7.21e-8	2.919

Table 5.2: Mean L_1 error sizes and order of accuracy for the finite difference scheme.

From Figure 5.10 we deduce that the simulations with the finite volume scheme do not show any anomalies. On the smooth grids, the results from the two schemes are very similar.

Nonlinear Advection

Zhang et al. (2011) and Kifonidis and Müller (2012) suggested a non-linear flow example to test the order of accuracy of a scheme. They showed that a smooth linear flow is not a suitable test case to test the accuracy of a finite volume method, because for a linear initial condition, the integration in equation (3.14) is exact, and the overall order of the method is not restricted. They suggested to instead use the non-linear initial conditions

$$\rho_{\text{mean}} = 1,\ u_{\text{mean}} = 1,\ v_{\text{mean}} = 1,\ p_{\text{mean}} = 1, \tag{5.104a}$$

with the perturbations of the velocities u and v, the temperature $T \sim p/\rho$, and the entropy $S \sim p/\rho^{1.4}$ defined by

$$(\delta u, \delta v) = \frac{\epsilon}{2\pi} \mathrm{e}^{0.5(1-r^2)}(-\overline{y}, \overline{x}),\ \delta T = -\frac{(\gamma-1)\epsilon^2}{8\gamma\pi^2} \mathrm{e}^{1-r^2},\ \delta S = 0. \tag{5.104b}$$

The computational domain is $[0, 10]^2$, $(\overline{x}, \overline{y}) = (x-5, y-5)$, $r^2 = \overline{x}^2 + \overline{y}^2$, $\gamma = 1.4$ and the vortex strength ϵ is 5. The analytical solution is passive convection of the vortex with the mean flow. For the explicit time integration, we used the third-order TVD3 scheme (Shu and Osher, 1988) in this test.

In Figures 5.11 and 5.12, the error of the WENO finite difference and the finite volume scheme for the nonlinear advection problem are shown. The error is measured by comparing the numerical solution after 0.2 s to the analytical one in the L^1 norm. We conclude that for the Cartesian and the smooth grid defined by the mapping functions M_4 and M_3, both schemes show comparable error sizes. The empirical order of convergence is between two and three for M_3 and higher than three for M_4 in both cases.

For the non-smooth mapping functions M_1 and M_2, the error does not decrease significantly for the finite difference scheme, whereas first- to second-order convergence can be observed with the finite volume scheme.

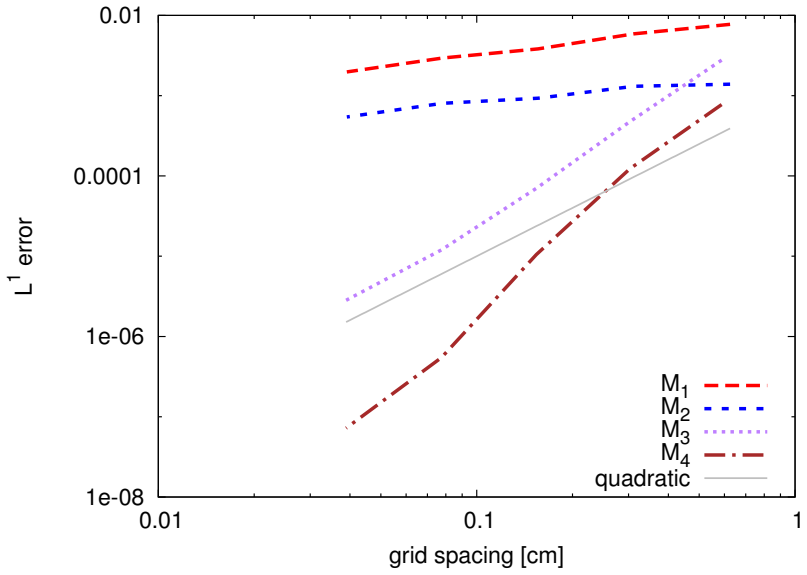

Figure 5.11: Order of accuracy of the WENO finite difference scheme for the nonlinear advection problem measured in the L^1 norm. The grey line indicates second-order convergence.

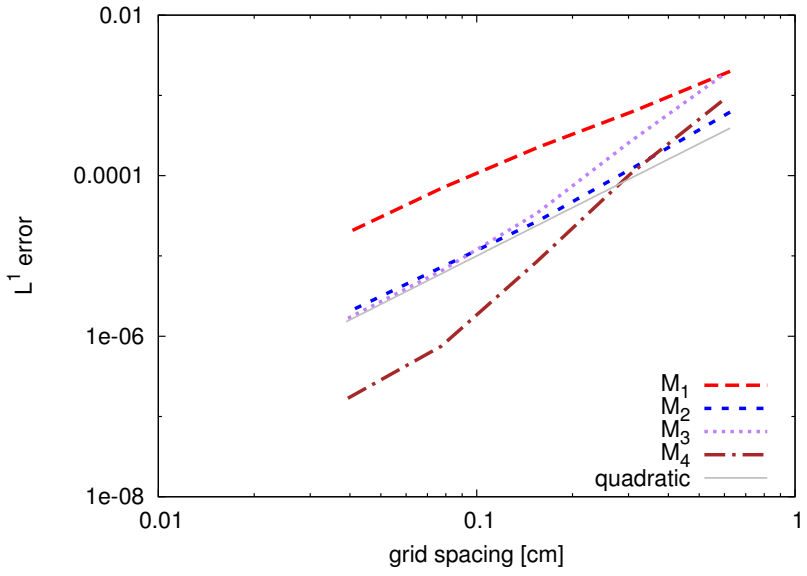

Figure 5.12: Order of accuracy of the WENO finite volume scheme for the nonlinear advection problem measured in the L^1 norm. The grey line indicates second-order convergence.

grid	M_1		M_2		M_3		M_4	
spacing	error	order	error	order	error	order	error	order
6.25e-1	2.00e-3		6.20e-4		1.74e-3		8.46e-4	
3.13e-1	6.40e-4	1.643	1.27e-4	2.283	2.64e-4	2.717	1.04e-4	3.018
1.56e-1	2.26e-4	1.498	2.73e-5	2.222	3.37e-5	2.970	8.40e-6	3.635
7.81e-2	7.06e-5	1.681	7.32e-6	1.900	6.53e-6	2.369	7.53e-7	3.480
3.91e-2	1.93e-5	1.869	1.97e-6	1.894	1.63e-6	1.998	1.64e-7	2.201

Table 5.3: Mean L_1 error sizes and order of accuracy for the finite volume scheme.

In the finite volume case, we expect second-order convergence for all mappings due to the second-order integration formula used to obtain equation (5.14). Consequently, we observe second-order convergence for all grids at least asymptotically for high resolution in Table 5.3. For low resolutions, the numerical solution exhibits third-order convergence indicating that the temporal error dominates in this regime. We note that the Courant number is 0.1 in all of our tests minimising the error due to the time integration scheme. We conclude that the absolute magnitude of the spatial error strongly depends on the mapping function, whereas the convergence order is restricted by the integration rule used to convert equation (5.12) into (5.14). To obtain a higher than second order accurate scheme, high-order integration formulae must be used (Casper and Atkins, 1993; Zhang et al., 2011).

On Cartesian grids, the finite difference algorithm is superior compared to the finite volume algorithm. As expected, the scheme is at least third order accurate for all resolutions whereas the finite volume scheme shows second-order convergence for high resolutions. For the smooth mapping M_3, Visbal and Gaitonde (2002) showed that the violation of the freestream preservation leads to large errors which dominate the overall error and decrease the convergence speed. With the non-smooth functions M_1 and M_2, this error is particularly large at the diagonals as we can observe for the Gresho vortex in Figure 5.6. This error does hardly decrease with grid spacing and leads to the slow convergence found in the data from Table 5.2. We can explain this slow convergence by the fact that in the finite difference approach, the fluxes $\hat{\mathbf{F}}$ and $\hat{\mathbf{G}}$ are reconstructed. But these fluxes are non-smooth themselves since they contain the non-smooth metric terms. Therefore, the reconstruction process is only of low order, too.

This test demonstrates once more that the WENO finite difference scheme should only be combined with the mapped grid technique if the mapping function is smooth. With the finite volume scheme, the simulation converges also on non-smooth grids, but at a much slower rate than on smooth grids. With both algorithms, the "smoother" function M_2 yields more accurate results than M_1. We conclude that also the finite volume scheme performs better the smoother the grid is. Therefore, non-smooth grids should only be used if absolutely necessary.

For coarse resolutions, the results of M_2 with the finite volume scheme are better than the results with M_1, but also with the smooth mapping M_3. They are even comparable

with the Cartesian mapping M_4. In astrophysical applications where the grid resolution usually is very coarse, grids like the one defined by M_2 combined with the WENO finite volume algorithm may well yield sufficiently accurate results. These grids are an acceptable alternative, if a non-smooth grid is needed by the problem geometry.

5.5.3 Prometheus

We compare the accuracy and efficiency of the Prometheus code with the split second-order time integration scheme by Warming and Beam (1976), the unsplit version where all fluxes are updated simultaneously (being equivalent to the first-order Euler forward method), and the second-order accurate unsplit CTU scheme (Colella, 1990). The code version with the CTU time integration uses the changes by Colella and Sekora (2008) whereas the other two code versions use the PPM scheme as it is described in Colella and Woodward (1984), but the differences are rather small in most cases.

As test problems, we select the nonlinear advection test from Zhang et al. (2011), the Taylor–Sedov test as in Wongwathanarat et al. (2010) and the moving Gresho vortex from Liska and Wendroff (2003).

Nonlinear Advection

The setup of the nonlinear advection test from Zhang et al. (2011) is described above. The resulting error decay plots with Prometheus can be seen in Figure 5.13. We observe second-order convergence for the split and the CTU scheme on the Cartesian grid, whereas the unsplit scheme is first order accurate at most. For high resolutions, the results with the unsplit scheme show oscillatory behaviour leading to an increase in mean error. The error size can be decreased by decreasing the Courant number which is 0.9 in these tests. Our tests indicate that the split and the CTU scheme yield stable results of similar accuracy. Both can be run with $\sigma = 0.9$ without any problems. Nevertheless, the computational costs of CTU are about twice as high as the ones of the split scheme, and its memory requirements are higher, too.

We also test the accuracy of the curvilinear extension of the CTU scheme. We use the mapping M_2 and $\sigma = 0.9$ in this paragraph. The empirical error decay as observed from Figure 5.13 is second order, too. We note that the size of the error cannot be compared directly. Even though it is a mean error per grid point, the total size of the domain is different to the simulation on Cartesian grids, but the size of the vortex is the same. This will result in different mean errors even if the accuracy of the scheme were the same. The convergence rate, on the other side, is not affected by this.

Taylor–Sedov Explosion

We test the curvilinear extension of the CTU scheme on the Taylor–Sedov explosion as described in Wongwathanarat et al. (2010). The initial setup is a spherical shock propagating radially. As centre of the explosion, we choose $(7.0 \times 10^{19}\,\mathrm{cm}, 2.5 \times 10^{19}\,\mathrm{cm})$ on a computational domain with side length $3 \times 10^{20}\,\mathrm{cm}$ centred at the origin. The

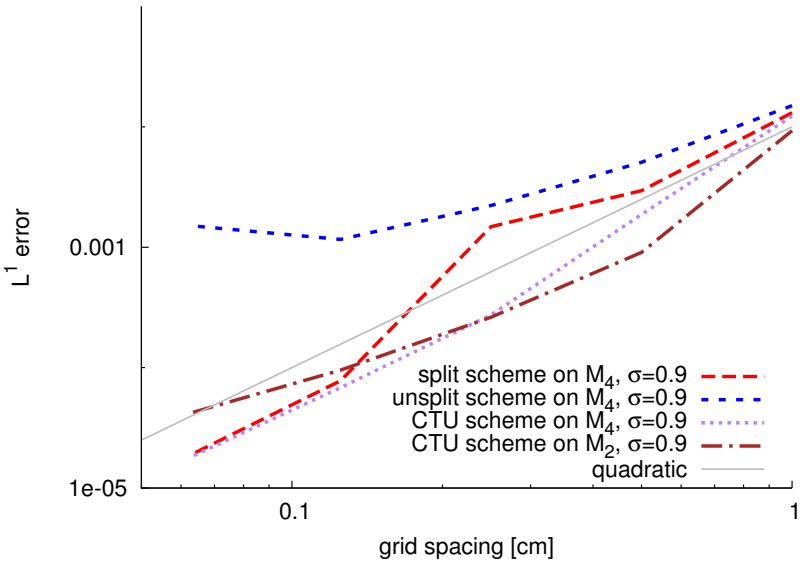

Figure 5.13: Order of accuracy of the PPM scheme with several time integration methods for the nonlinear advection problem measured in the L^1 norm. The grey line indicates second-order convergence.

results of the simulation with M_1, M_2, M_4 and the polar grid defined by M_{polar} after 2.34×10^{11} s are shown in Figure 5.14. The yellow circle indicates the position of the shock front according to the analytical solution assuming cylindrical symmetry.[1] We use 100 grid points in each direction.

We observe that the position of the shock is correct for each grid, but there are asymmetries introduced by the grid structure and insufficient resolution. Since the explosion centre is not in the centre of the grid, a huge amount of grid cells would be needed to keep the spherical shape of the initial explosion. From Table 5.4 we deduce that the cell volume ratio r_V defined as the quotient of the smallest and the largest cell volume of grids M_1 and M_2 is much larger than for M_{polar}, allowing much larger time steps. Nevertheless, the accuracy of the numerical solution is the same on the three grids. The mean L^1 error is smallest on the Cartesian grid partly due to the fact that the domain size is larger by nearly 25 %. The additional parts of the domain do not contribute any relevant information, in particular no error, and therefore decrease the mean error.

In conclusion, the curvilinear extension of the CTU scheme yields stable and reasonably accurate results for this test case. The advantage of the curvilinear approach with the mapping function M_{polar} compared to classical polar and spherical coordinate systems is that no geometrical source terms are introduced, and linear momentum is

[1] We thank Annop Wongwathanarat for supplying the IDL script calculating the analytical solution of the Taylor–Sedov problem

grid	r_V	domain size [cm]	mean L^1 error over time [s] 3.4e10	8.4e10	1.34e11	1.84e11	2.34e11
M_1	5.05e-1	7.07e40	7.20e6	1.08e7	1.29e7	1.47e7	1.61e7
M_2	3.21e-1	7.07e40	7.18e6	1.07e7	1.28e7	1.46e7	1.61e7
M_4	1.0	9.00e40	5.56e6	8.35e6	9.91e6	1.13e7	1.24e7
M_{polar}	3.84e-2	7.13e40	7.12e6	1.07e7	1.27e7	1.43e7	1.58e7

Table 5.4: Maximum cell volume ratio, domain size, and mean L^1 error over time for each of the grids considered in the Taylor–Sedov example. The cell aspect ratio r_V is defined as the ratio of the smallest to the largest cell volume. The mean L^1 error was obtained by calculating the differences in radial velocity $u_r = \sqrt{u^2 + v^2}$ of the numerical solution compared to the analytical one in the L^1 norm, and then dividing by the domain size.

conserved to machine precision. With the grids M_1 and M_2, we are able to circumvent the grid singularity at $r = 0$.

Moving Gresho

The moving Gresho vortex results from adding a mean flow in x direction to the stationary vortex as defined above. The mean flow is $(u, v) = (1, 0)$ and the computational domain $[0, 4] \times [0, 1]$.

We run the simulation on 160×40 grid points for 3 s with a Courant factor $\sigma = 0.9$ on a Cartesian grid. The analytical solution is advection of the rotating vortex with the mean flow. In Figure 5.15, we show the vorticity $\omega = \nabla \cdot \mathbf{u}$ at the end of the simulation. We see that the symmetry of the vortex is lost in all cases. Nevertheless, the result with the CTU scheme is closest to the analytical solution. The result with the unsplit scheme is worst, whereas with the split integration, the magnitude of the vorticity is correct, but the vortex is deformed considerably. We note that the results for PPM in Liska and Wendroff (2003) look similar. There, they used the split time integration scheme.

On the top of each panel in Figure 5.15, we show the end time of the simulation in seconds, the kinetic energy scaled by the initial kinetic energy, and the maximum Mach number of the flow at the end of the simulation. The initial Mach number is about 0.65. We expect Mach number and total kinetic energy of the simulation to stay unchanged since the problem consists only of advective motions and rotation. Also in this respect, the CTU scheme gives the results closest to the analytical solution, whereas the unsplit scheme is worst.

Similar conclusions can be drawn when looking at the angular velocity in Figure 5.16. The results with the CTU scheme are superior to the ones with the split and even more with the unsplit time integration. In all cases, the symmetry of the vortex gets lost.

Next, we investigate the Mach number dependence of the results. In Figure 5.17, the angular velocity of three simulations after a simulation time of 2.5 s is shown where we modified the reference Mach number Ma_{ref} of the rotation of the vortex. The mean flow

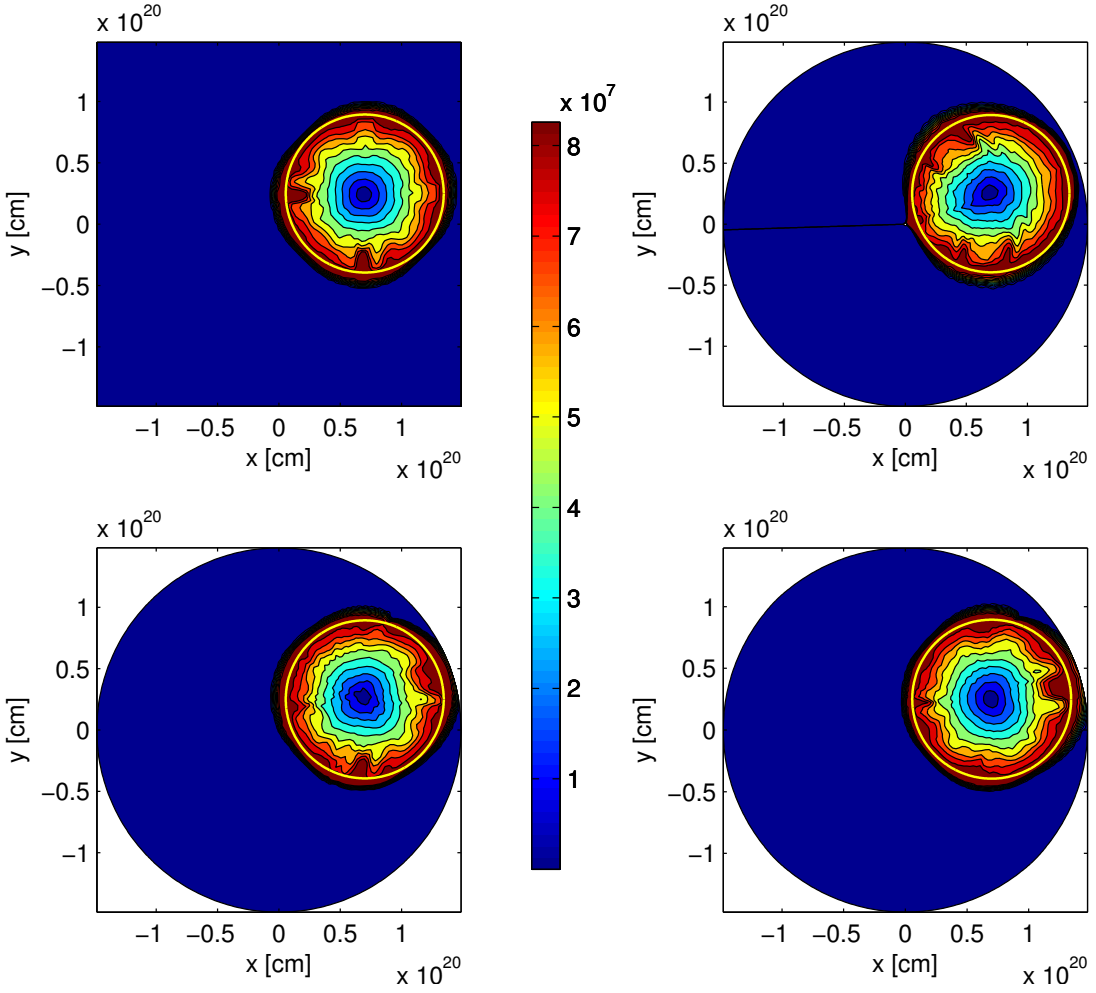

Figure 5.14: Numerical solution of the Taylor–Sedov explosion 2.34×10^{11} s after the explosion with the Prometheus code and CTU time integration. The Courant number is 0.9. Color scale according to radial velocity $u_r = \sqrt{u^2 + v^2}$. The yellow circle indicates the position of the exact solution (assuming cylindrical symmetry). The initial setup is as in Wongwathanarat et al. (2010). From top left to bottom right: grid M_4, polar grid as defined by M_{polar}, grid M_1 and grid M_2. The inner circle of the polar grid is at 5.0×10^{18} cm. The horizontal line is due to visualisation.

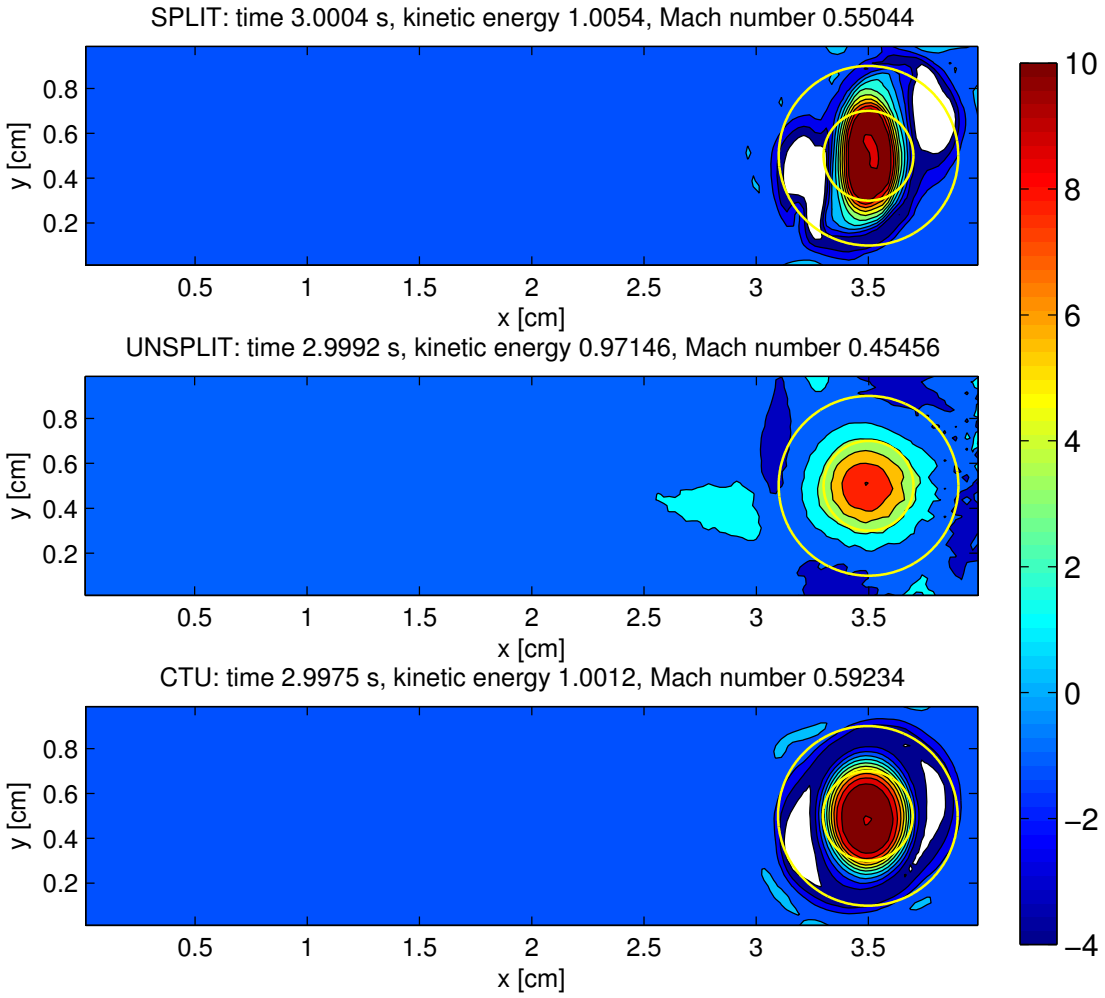

Figure 5.15: Magnitude of the vorticity of the moving Gresho vortex. The yellow circles indicate the exact position of the vortex as it is advected with the mean flow. From top to bottom: results with split, unsplit and CTU scheme. On the top of each panel, the relative kinetic energy scaled by the initial kinetic energy, and the maximum Mach number of the flow at the end of the simulation are given.

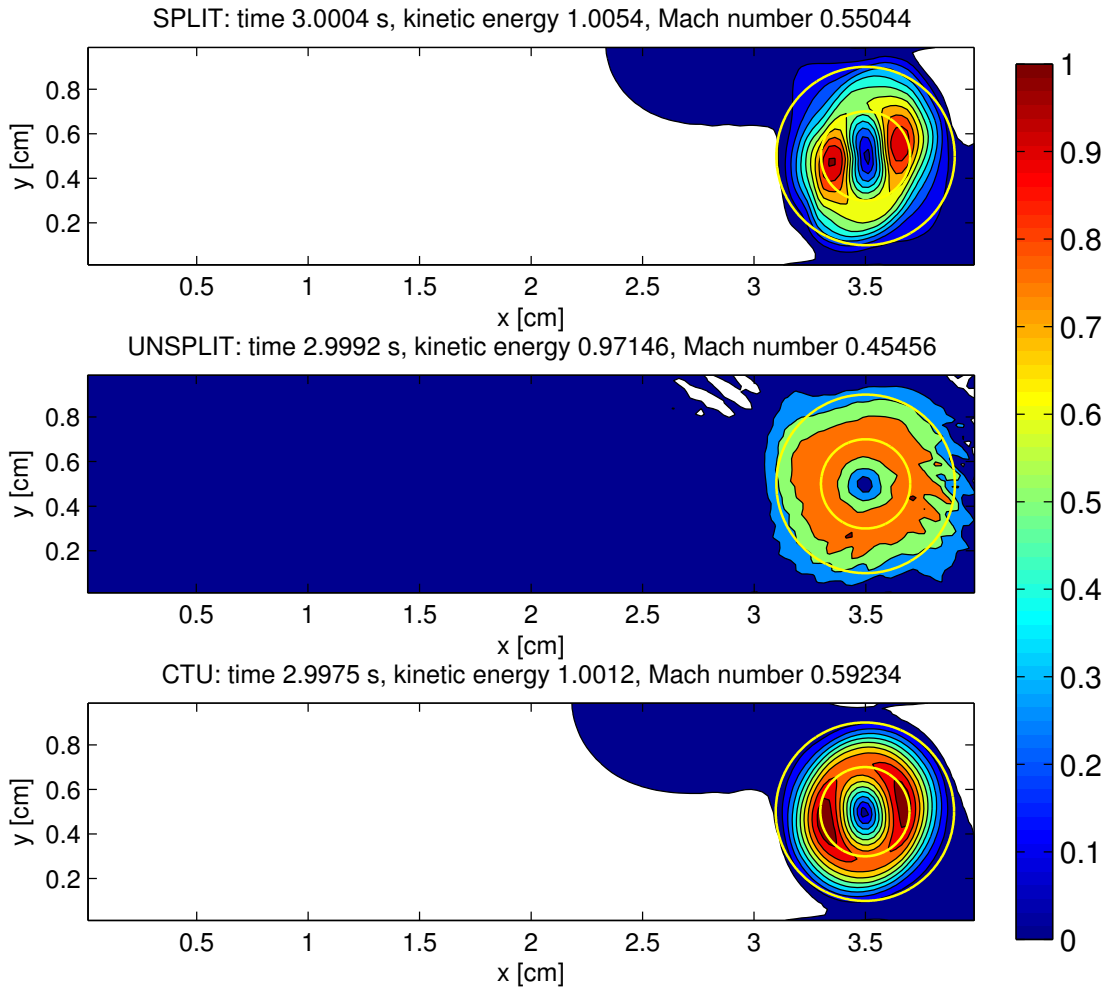

Figure 5.16: Angular velocity of the moving Gresho vortex on the Cartesian grid defined by M_4. The yellow circles indicate the exact position of the vortex. From top to bottom: results with split, unsplit and CTU scheme. On the top of each panel, the relative kinetic energy scaled by the initial kinetic energy, and the maximum Mach number of the flow at the end of the simulation are given.

is the same in all cases. All simulations are done with the CTU scheme and $\sigma = 0.9$. We observe that the strength of the deformation increases when the Mach number gets lower. Also the damping in terms of kinetic energy and magnitude of the angular velocity increases. This is similar to what was observed in Happenhofer et al. (2013) for the WENO5 finite difference scheme, even though the results with PPM look slightly better.

Next, we calculate the moving Gresho test on the curvilinear grids defined by the mapping functions M_1, M_2 and M_3. Therefore, we have to change the computational domain to the circle $K = \{(x, y) \in \mathbb{R}^2 : \sqrt{(x-2)^2 + (y-2)^2} \leq 2\}$ resp. the square $[0, 4]^2$ in the case of M_3. The number of grid points is reset to 160 in both directions. In the Figures 5.18, 5.19 and 5.20, the angular velocity in the region $1.5 \leq y \leq 2.5$ is shown for the grids M_1, M_2 and M_3, respectively.

We observe that the result with the split scheme is catastrophic on all grids. The vortex is destroyed completely and instead, high artificial velocities at the diagonals occur where the mapping function is not differentiable. This is a result of the violation of the freestream preservation property due to the split time integration. In contrast, with the unsplit and the CTU scheme, the vortex is preserved. Clearly, the result with the CTU scheme is the most accurate one, also in terms of damping of kinetic energy.

The results for the mapping M_3 are similar, but the accuracy with all schemes is worse. Also with the unsplit and the CTU schemes, the vortex is deformed considerably. By lowering the Courant number, we obtain much better results, as shown in Figure 5.21 for the CTU scheme. With $\sigma = 0.9$, the symmetry of the vortex is destroyed completely, whereas with $\sigma = 0.1$ the solution looks quite accurate. This proves that the accuracy and the stability of the curvilinear CTU scheme is worse than its Cartesian version. The behaviour is strongly influenced by the mapping function used. Even though M_3 is a smooth grid in the sense that the mapping function is strongly differentiable, it varies quite rapidly from cell to cell. We suspect that this is the reason for the instability and bad accuracy of the method in this case.

Nevertheless, for reasonably low Mach numbers, the curvilinear CTU scheme works well for $\sigma = 0.9$ and the mapping functions M_1 and M_2 for the moving Gresho vortex test. For M_3, the results are much worse due to the fast cell–to–cell variation of the grid.

5.5.4 A Direct Comparison

We compare the accuracy and efficiency of the finite volume variant of ANTARES and Prometheus with the CTU scheme for the Gaussian initial condition

$$r = \sqrt{(x-0.5)^2 + (y-0.25)^2}, \ \rho(x, y) = \begin{cases} 1 + \exp(-16r), & r < 0.5, \\ 1, & \text{else}, \end{cases} \tag{5.105}$$

advected with the mean flow $u = 1$ and $v = 0.5$. Furthermore, $p = 1$ and $\gamma = \frac{5}{3}$. We solve the Euler equations on $[-1, 1]$ with periodic boundary conditions using the mappings M_3 and M_4 on five grids with 10, 20, 40, 80 and 160 grid points in each direction.

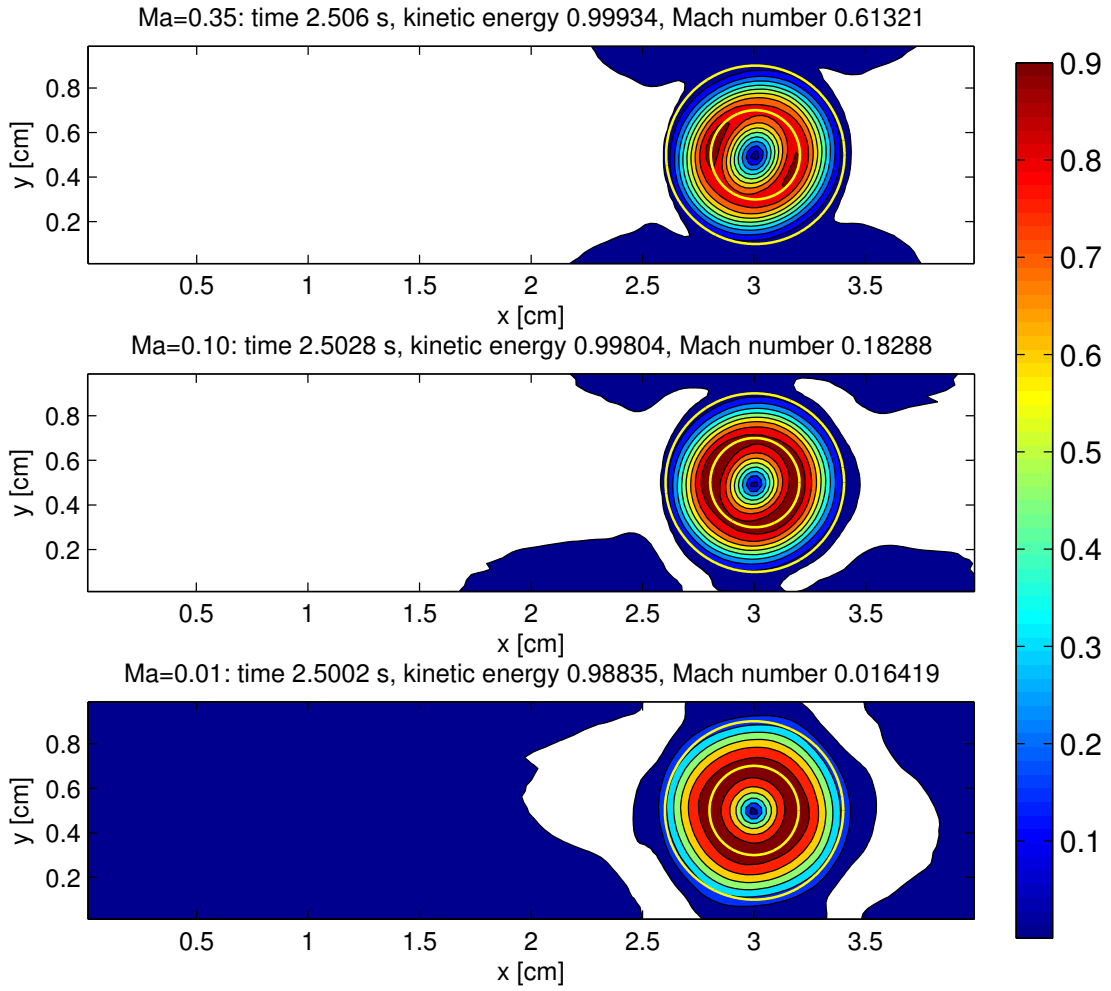

Figure 5.17: Angular velocity of the moving Gresho vortex in dependence of the Mach number. The yellow circles indicate the exact position of the vortex. In each panel, a different value for the Mach number of the rotation was chosen, whereas the mean flow was always set to $(u, v) = (1, 0)$. From top to bottom: $\mathrm{Ma}_{\mathrm{ref}} = 0.35$, $\mathrm{Ma}_{\mathrm{ref}} = 0.1$ and $\mathrm{Ma}_{\mathrm{ref}} = 0.01$. All simulations are done with the CTU scheme. On the top of each panel, the relative kinetic energy scaled by the initial kinetic energy, and the maximum Mach number of the flow (taking both rotation and mean flow into account) at the end of the simulation are given.

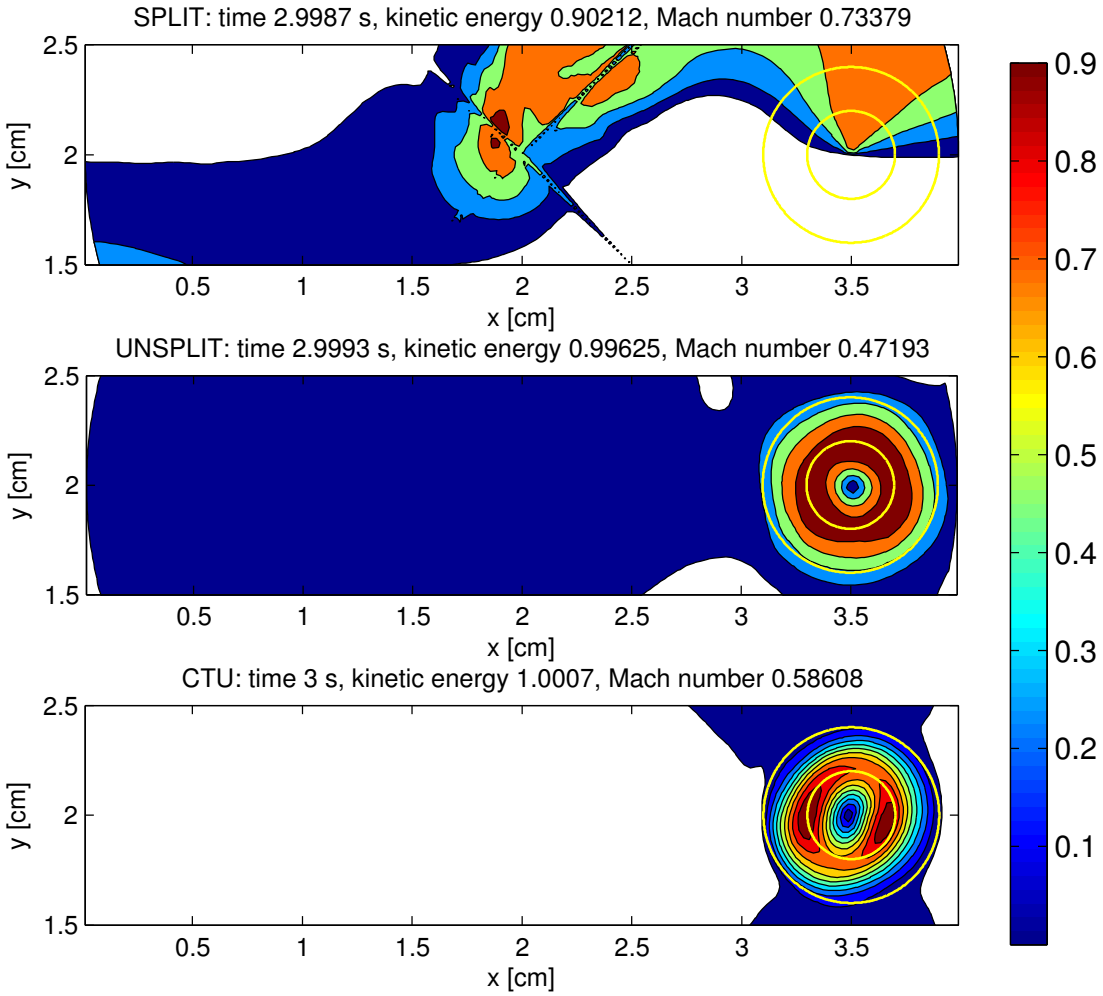

Figure 5.18: Angular velocity of the moving Gresho vortex on the curvilinear grid defined by M_1. Only the region $1.5 \leq y \leq 2.5$ is shown. The yellow circles indicate the exact position of the vortex. From top to bottom: results with split, unsplit and CTU scheme. On the top of each panel, the relative kinetic energy scaled by the initial kinetic energy, and the maximum Mach number of the flow at the end of the simulation are given.

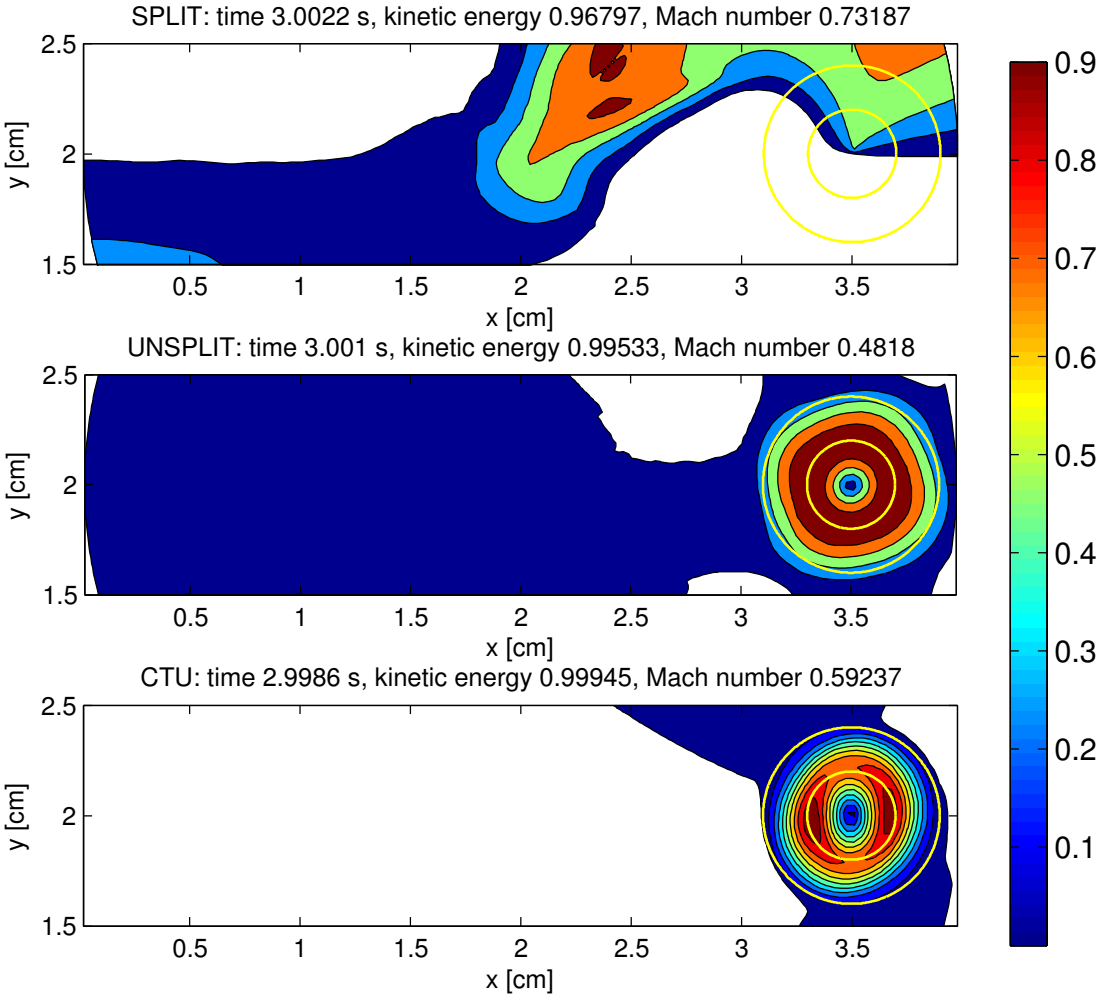

Figure 5.19: Angular velocity of the moving Gresho vortex on the curvilinear grid defined by M_2. Only the region $1.5 \leq y \leq 2.5$ is shown. The yellow circles indicate the exact position of the vortex. From top to bottom: results with split, unsplit and CTU scheme. On the top of each panel, the relative kinetic energy scaled by the initial kinetic energy, and the maximum Mach number of the flow at the end of the simulation are given.

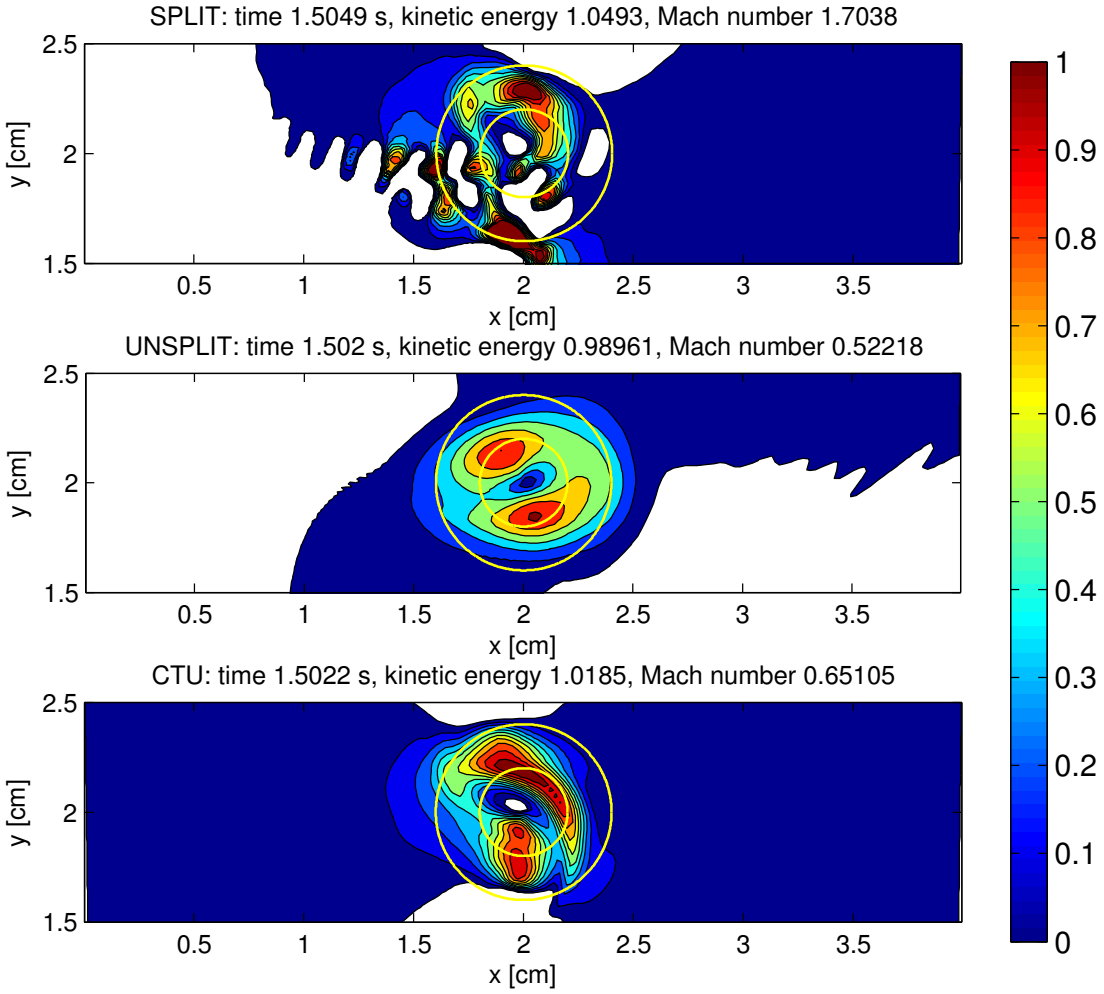

Figure 5.20: Angular velocity of the moving Gresho vortex on the curvilinear grid defined by M_3. Only the region $1.5 \leq y \leq 2.5$ is shown. The yellow circles indicate the exact position of the vortex. From top to bottom: results with split, unsplit and CTU scheme. On the top of each panel, the relative kinetic energy scaled by the initial kinetic energy, and the maximum Mach number of the flow at the end of the simulation are given.

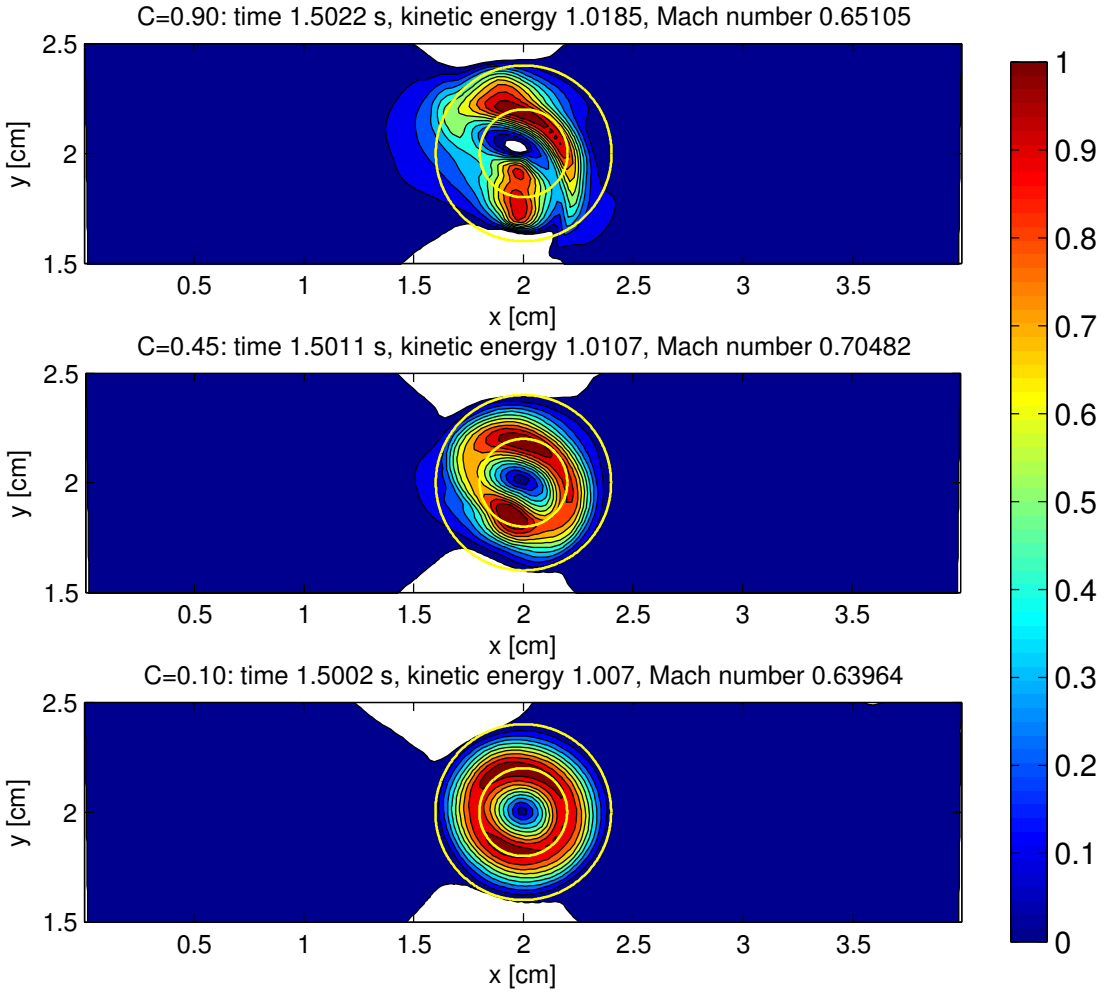

Figure 5.21: Angular velocity of the moving Gresho vortex on the curvilinear grid defined by M_3. Only the region $1.5 \leq y \leq 2.5$ is shown. The yellow circles indicate the exact position of the vortex. From top to bottom: results with $\sigma = 0.9$, 0.45 and 0.1. On the top of each panel, the relative kinetic energy scaled by the initial kinetic energy, and the maximum Mach number of the flow at the end of the simulation are given.

We compare the error size of the L^1 norm and the maximum value of density after 2 s for both codes in Figure 5.22. For ANTARES, we use the SSP RK(3,2) Runge–Kutta scheme with $\sigma = 0.5$, whereas we set $\sigma = 0.8$ for the CTU scheme used in Prometheus. We observe that ANTARES is more accurate on both grids, especially for high resolutions.

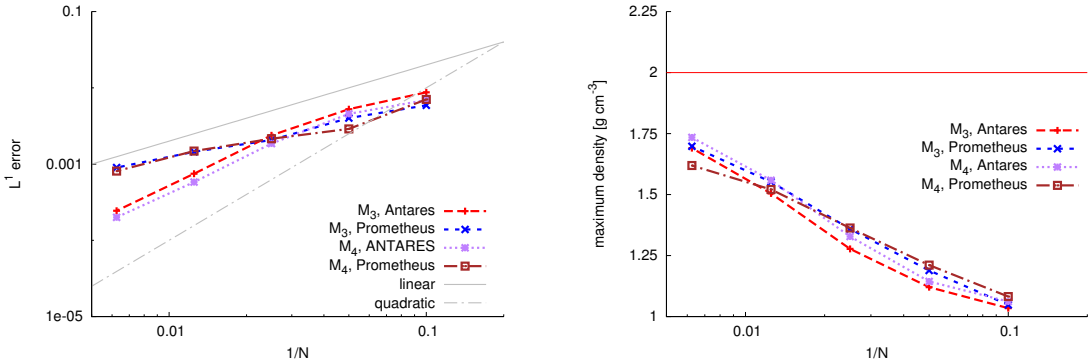

Figure 5.22: Error decay (left) and maximum density after 2 s (right) for the Gaussian test (5.105) with the ANTARES and the Prometheus code on the grids defined by M_3 and M_4. N is the number of grid points in each direction. On the left, the grey lines indicate first- and second-order convergence. On the right, the grey line indicates the maximum density of the analytical solution.

In Table 5.5, we compare the computational costs. The wall-clock times of ANTARES are much higher than for Prometheus even though the number of time steps is quite similar. We conclude that each integration step with ANTARES takes much longer than with Prometheus, but also yields more accurate results. However, we refrain from taking the precise numbers too seriously, since they depend on the compiler, the processor and many other factors as well.

	ANTARES				Prometheus			
	M_3		M_4		M_3		M_4	
N	steps	time	steps	time	steps	time	steps	time
40	690	0:10	227	00:03	544	00:07	236	00:03
80	1440	1:20	464	00:27	1067	00:55	437	00:22
160	2902	11:01	940	03:48	2111	07:14	841	02:44

Table 5.5: Number of time integration steps and wall-clock time of the Gaussian test (5.105) with the ANTARES and the Prometheus code on the grids defined by M_3 and M_4. Wall-clock time is in the format mm:ss and was measured on an Intel Xeon CPU E5462 at 2.80 GHz. ANTARES was compiled with the *ifort* and Prometheus with the *gfortran* compiler.

From this test we deduce that ANTARES is slightly more accurate, but also slower than Prometheus. For both codes, the use of M_3 does not significantly reduce the accuracy. The efficiency is much worse with M_3, but we can explain this with the non-uniform cell volume leading to stronger time step restrictions. In general, curvilinear grids can even lead to larger time steps, e.g. compared to a spherical grid.

5.5.5 Conclusions

Curvilinear coordinates are an easy and efficient way to extend existing codes written for Cartesian grids to more complicated geometries. The grid can be generated either by a (analytical or numerical) mapping function, or by an external grid generation program. It must always be structured.

For the case of spherical domains, Calhoun et al. (2008) provided mapping functions which are not strongly differentiable. We showed that the analytical transformation also works in a weak sense without assuming strong differentiability of the mapping function.

WENO Schemes

In Shu (2003), the application of the WENO finite difference scheme to smooth curvilinear grids is shown. The results are of high accuracy, since the mapping functions are smooth and the Mach number of the numerical tests are high.

In this chapter, the WENO finite difference and finite volume scheme were applied with non-smooth mapping functions as those defined by the mapping functions from Calhoun et al. (2008) for the first time. Particular attention is paid to problems arising when the Mach number of the flow is rather low. Whereas the WENO finite volume scheme performs well in these cases, the WENO finite difference scheme does not give a convergent solution. It only works if the mapping function is smooth, but even in this case the finite volume scheme yields better results for the low Mach number tests presented in this paper.

Since it is only a minor algorithmic change from the WENO finite difference to the WENO finite volume scheme, we recommend to switch to the WENO finite volume scheme when the mapped grid technique is used for non-smooth mapping functions as well as for low Mach numbers. The smoother the mapping function is, the more accurate are the results. Therefore, one should always use the smoothest mapping function allowed by the simulation setup.

In theory, the finite volume scheme is only second order accurate. In our calculations, however, the empirical order of accuracy of the finite volume scheme was higher. To increase the theoretical order of accuracy the computational requirements and the complexity of the code has to be increased considerably (Casper and Atkins, 1993; Zhang et al., 2011; Colella et al., 2011). In practice, WENO schemes usually are combined with lower order Runge–Kutta methods for time integration with highest-possible Courant numbers. The overall error of the scheme will be dominated by the temporal error, and the overall order of the method will be limited to two or three, anyway.

In astrophysical simulations, the typical resolution is rather coarse. In this regime, non-smooth mapping functions perform nearly as well as smooth ones, when combined with the WENO finite volume scheme. We conclude that they are an acceptable alternative, if the problem geometry requires the use of non-smooth mapping functions. We conclude that curvilinear coordinates provide enough flexibility for most problems in computational astrophysics whereas keeping the high efficiency and accuracy of the Cartesian schemes they are based on.

PPM Scheme

The applicability and efficiency of curvilinear coordinates for the PPM scheme depends on the time integration method it is combined with. With the dimensionally split time integration from Warming and Beam (1976), the violation of the freestream preservation leads to unacceptable results, as in the case of the finite difference WENO scheme.

Updating all fluxes simultaneously — in an "unsplit" fashion —, leads to preservation of the freestream, but the order of the scheme is degraded to first order, and the scheme is not stable any more with the high Courant numbers of the split version. This approach corresponds to using the Euler forward scheme for time integration.

For the Cartesian case, the CTU scheme from Colella (1990) is superior compared to both the split and the unsplit scheme. Since it is an unsplit scheme itself, the freestream is preserved, and curvilinear coordinates can be implemented in the way we outlined before. Nevertheless, in certain tests and with some mapping functions, the accuracy and stability of the curvilinear version is much worse than the Cartesian version.

Therefore, we do not recommend to use the curvilinear version of the CTU scheme. It is hard to predict in which situations it is stable and in which it is not, making the use of very low Courant numbers necessary. Alternatively, a high-order Runge–Kutta scheme combined with the PPM scheme as described in Colella et al. (2011) may lead to stable and high-order accurate results.

5.6 Metric Terms in Three Dimensions

A (system of) conservation law(s) in Cartesian space in three dimensions is given by

$$\frac{\partial}{\partial t}\mathbf{Q} = \frac{\partial}{\partial x}\mathbf{F} + \frac{\partial}{\partial y}\mathbf{G} + \frac{\partial}{\partial z}\mathbf{H}, \tag{5.106}$$

where $\mathbf{Q}$ is the vector of the conserved quantities, e.g. for the Euler equations

$$\mathbf{Q} = (\rho, \rho u, \rho v, \rho w, E)^{\mathrm{T}}, \tag{5.107}$$

and $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ are the (analytical) flux functions. For the Euler equations, they are given by

$$\mathbf{F} = \begin{pmatrix} \rho u \\ \rho uu + p \\ \rho uv \\ \rho uw \\ (p+E)u \end{pmatrix}, \mathbf{G} = \begin{pmatrix} \rho v \\ \rho vu \\ \rho vv + p \\ \rho vw \\ (p+E)v \end{pmatrix}, \mathbf{H} = \begin{pmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho ww + p \\ (p+E)w \end{pmatrix}. \tag{5.108}$$

Let a (weakly differentiable) transformation M be given such that

$$M: \ [-1,1]^3 \to \Omega, \ M(\xi,\eta,\zeta) = (x,y,z). \tag{5.109}$$

We assume that the mapping function does not change in time. The inverse Jacobian J^{-1} of the transformation M is given by

$$\begin{aligned} J^{-1} = \left|\frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)}\right| = &\left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} + \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \xi} + \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta}\right) \\ &- \left(\frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta} + \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \zeta}\right). \end{aligned} \tag{5.110}$$

Since $\frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)} = \left(\frac{\partial(\xi,\eta,\zeta)}{\partial(x,y,z)}\right)^{-1}$, we can derive

$$\frac{\partial \xi}{\partial x}J^{-1} = y_\eta z_\zeta - y_\zeta z_\eta, \frac{\partial \xi}{\partial y}J^{-1} = x_\zeta z_\eta - x_\eta z_\zeta, \frac{\partial \xi}{\partial z}J^{-1} = x_\eta y_\zeta - x_\zeta y_\eta, \tag{5.111a}$$

$$\frac{\partial \eta}{\partial x}J^{-1} = y_\zeta z_\xi - y_\xi z_\zeta, \frac{\partial \eta}{\partial y}J^{-1} = x_\xi z_\zeta - x_\zeta z_\xi, \frac{\partial \eta}{\partial z}J^{-1} = x_\zeta y_\xi - x_\xi y_\zeta, \tag{5.111b}$$

$$\frac{\partial \zeta}{\partial x}J^{-1} = y_\xi z_\eta - y_\eta z_\xi, \frac{\partial \zeta}{\partial y}J^{-1} = x_\eta z_\xi - x_\xi z_\eta, \frac{\partial \zeta}{\partial z}J^{-1} = x_\xi y_\eta - x_\eta y_\xi. \tag{5.111c}$$

With the chain rule of differentiation,

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial x}\frac{\partial}{\partial \zeta}, \tag{5.112a}$$

$$\frac{\partial}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial y}\frac{\partial}{\partial \zeta}, \tag{5.112b}$$

$$\frac{\partial}{\partial z} = \frac{\partial \xi}{\partial z}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial z}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial z}\frac{\partial}{\partial \zeta}. \tag{5.112c}$$

the system is transformed to a conservative system in computational space in the form

$$\frac{\partial}{\partial t}J^{-1}Q = \frac{\partial}{\partial \xi}\hat{\mathbf{F}} + \frac{\partial}{\partial \eta}\hat{\mathbf{G}} + \frac{\partial}{\partial \zeta}\hat{\mathbf{H}}. \tag{5.113}$$

similar to what was done in sections 5.1 and 5.2. The transformed fluxes are defined by

$$\hat{\mathbf{F}} := n_1\mathbf{F} + n_2\mathbf{G} + n_3\mathbf{H}, \tag{5.114a}$$
$$\hat{\mathbf{G}} := m_1\mathbf{F} + m_2\mathbf{G} + m_3\mathbf{H}, \tag{5.114b}$$
$$\hat{\mathbf{H}} := o_1\mathbf{F} + o_2\mathbf{G} + o_3\mathbf{H}, \tag{5.114c}$$

where we have written

$$n_1 := y_\eta z_\zeta - z_\eta y_\zeta, m_1 := y_\zeta z_\xi - z_\zeta y_\xi, o_1 := y_\xi z_\eta - z_\xi y_\eta, \tag{5.115a}$$
$$n_2 := z_\eta x_\zeta - x_\eta z_\zeta, m_2 := z_\zeta x_\xi - x_\zeta z_\xi, o_2 := z_\xi x_\eta - x_\xi z_\eta, \tag{5.115b}$$
$$n_3 := x_\eta y_\zeta - y_\eta x_\zeta, m_3 := x_\zeta y_\xi - y_\zeta x_\xi, o_3 := x_\xi y_\eta - y_\xi x_\eta. \tag{5.115c}$$

As before, we have to find a discretisation of these terms such that no spurious source terms arise. Therefore, we check if the analytical solution to the freestream problem 6 is preserved by the numerical discretisation.

5.6.1 Discretisation

We discretise the equations in space according to the finite volume approach. Then, the update is defined by

$$\begin{aligned}\frac{\partial}{\partial t} J^{-1}\mathbf{Q} = &\left(\hat{\mathbf{F}}_{i+\frac{1}{2},j,k} - \hat{\mathbf{F}}_{i-\frac{1}{2},j,k}\right) \\ &+ \left(\hat{\mathbf{G}}_{i,j+\frac{1}{2},k} - \hat{\mathbf{G}}_{i,j-\frac{1}{2},k}\right) \\ &+ \left(\hat{\mathbf{H}}_{i,j,k+\frac{1}{2}} - \hat{\mathbf{H}}_{i,j,k-\frac{1}{2}}\right).\end{aligned} \tag{5.116}$$

For the Euler equations, the initial conditions $u = v = w = 0$, $\rho = \rho_0$, $E = e_0$ with some constants ρ_0 and e_0 lead to

$$\begin{aligned}0 = &\left(n_1|_{i+\frac{1}{2},j,k} - n_1|_{i-\frac{1}{2},j,k}\right) + \left(m_1|_{i,j+\frac{1}{2},k} - m_1|_{i,j-\frac{1}{2},k}\right) \\ &+ \left(o_1|_{i,j,k+\frac{1}{2}} - o_1|_{i,j,k-\frac{1}{2}}\right),\end{aligned} \tag{5.117a}$$
$$\begin{aligned}0 = &\left(n_2|_{i+\frac{1}{2},j,k} - n_2|_{i-\frac{1}{2},j,k}\right) + \left(m_2|_{i,j+\frac{1}{2},k} - m_2|_{i,j-\frac{1}{2},k}\right) \\ &+ \left(o_2|_{i,j,k+\frac{1}{2}} - o_2|_{i,j,k-\frac{1}{2}}\right),\end{aligned} \tag{5.117b}$$
$$\begin{aligned}0 = &\left(n_3|_{i+\frac{1}{2},j,k} - n_3|_{i-\frac{1}{2},j,k}\right) + \left(m_3|_{i,j+\frac{1}{2},k} - m_3|_{i,j-\frac{1}{2},k}\right) \\ &+ \left(o_3|_{i,j,k+\frac{1}{2}} - o_3|_{i,j,k-\frac{1}{2}}\right),\end{aligned} \tag{5.117c}$$

since $\frac{\partial}{\partial t}(\rho u) = \frac{\partial}{\partial t}(\rho v) = \frac{\partial}{\partial t}(\rho w)$ and $p = p(\rho_0, e_0)$ is constant, too. These analytical conditions must be fulfilled numerically to solve the freestream problem. This can be accomplished by choosing a suitable discretisation for the metric terms. Even if analytical expressions are known, they should not be used because then the above conditions are not fulfilled.

Direct discretisation of equations (5.115) with second order accurate finite differences does lead to a violation of the freestream, which can be checked by a straightforward calculation.

In precise terms, replacing $n_1|_{i+\frac{1}{2},j,k}$, $m_1|_{i,j\pm\frac{1}{2},k}$ and $o_1|_{i,j,k\pm\frac{1}{2}}$ by

$$\begin{aligned}
n_1|_{i\pm\frac{1}{2},j,k} &= (y_\eta z_\zeta - z_\eta y_\zeta)\,|_{i\pm\frac{1}{2},j,k} \\
&\approx \left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k}\right)\left(z_{i\pm\frac{1}{2},j,k+\frac{1}{2}} - z_{i\pm\frac{1}{2},j,k-\frac{1}{2}}\right), \qquad (5.118a)\\
&- \left(z_{i\pm\frac{1}{2},j+\frac{1}{2},k} - z_{i\pm\frac{1}{2},j-\frac{1}{2},k}\right)\left(y_{i\pm\frac{1}{2},j,k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j,k-\frac{1}{2}}\right), \\
m_1|_{i,j\pm\frac{1}{2},k} &= (y_\zeta z_\xi - z_\zeta y_\xi)\,|_{i,j\pm\frac{1}{2},k} \\
&\approx \left(y_{i,j\pm\frac{1}{2},k+\frac{1}{2}} - y_{i,j\pm\frac{1}{2},k-\frac{1}{2}}\right)\left(z_{i+\frac{1}{2},j\pm\frac{1}{2},k} - z_{i-\frac{1}{2},j\pm\frac{1}{2},k}\right), \qquad (5.118b)\\
&- \left(z_{i,j\pm\frac{1}{2},k+\frac{1}{2}} - z_{i,j\pm\frac{1}{2},k-\frac{1}{2}}\right)\left(y_{i+\frac{1}{2},j\pm\frac{1}{2},k} - y_{i-\frac{1}{2},j\pm\frac{1}{2},k}\right), \\
o_1|_{i,j,k\pm\frac{1}{2}} &= (y_\xi z_\eta - z_\xi y_\eta)\,|_{i,j,k\pm\frac{1}{2}} \\
&\approx \left(y_{i+\frac{1}{2},j,k\pm\frac{1}{2}} - y_{i-\frac{1}{2},j,k\pm\frac{1}{2}}\right)\left(z_{i,j+\frac{1}{2},k\pm\frac{1}{2}} - z_{i,j-\frac{1}{2},k\pm\frac{1}{2}}\right) \qquad (5.118c)\\
&- \left(z_{i+\frac{1}{2},j,k\pm\frac{1}{2}} - z_{i-\frac{1}{2},j,k\pm\frac{1}{2}}\right)\left(y_{i,j+\frac{1}{2},k\pm\frac{1}{2}} - y_{i,j-\frac{1}{2},k\pm\frac{1}{2}}\right),
\end{aligned}$$

we see by comparing coefficients for, e.g., $y_{i+\frac{1}{2},j+\frac{1}{2},k}$,

$$\left(z_{i+\frac{1}{2},j,k+\frac{1}{2}} - z_{i+\frac{1}{2},j,k-\frac{1}{2}}\right) + \left(z_{i,j+\frac{1}{2},k+\frac{1}{2}} - z_{i,j+\frac{1}{2},k-\frac{1}{2}}\right) \neq 0, \qquad (5.119)$$

and the terms do not cancel out in general.

Instead, the metric derivatives can be rewritten as (Visbal and Gaitonde, 2002)

$$\begin{aligned}
n_1 &:= (y_\eta z)_\zeta - (z y_\zeta)_\eta, m_1 := (y_\zeta z)_\xi - (z y_\xi)_\zeta, o_1 := (y_\xi z)_\eta - (z y_\eta)_\xi, && (5.120a)\\
n_2 &:= (z_\eta x)_\zeta - (x z_\zeta)_\eta, m_2 := (z_\zeta x)_\xi - (x z_\xi)_\zeta, o_2 := (z_\xi x)_\eta - (x z_\eta)_\xi, && (5.120b)\\
n_3 &:= (x_\eta y)_\zeta - (y x_\zeta)_\eta, m_3 := (x_\zeta y)_\xi - (y x_\xi)_\zeta, o_3 := (x_\xi y)_\eta - (y x_\eta)_\xi, && (5.120c)
\end{aligned}$$

assuming that the second derivatives commute. Discretising all metric terms in the same manner by second order central differences,

$$
\begin{aligned}
n_1|_{i\pm\frac{1}{2},j,k} &= (y_\eta z)_\zeta\,|_{i\pm\frac{1}{2},j,k} - (zy_\zeta)_\eta\,|_{i\pm\frac{1}{2},j,k} \\
&\approx \left((y_\eta z)\,|_{i\pm\frac{1}{2},j,k+\frac{1}{2}} - (y_\eta z)\,|_{i\pm\frac{1}{2},j,k-\frac{1}{2}}\right) \\
&- \left((zy_\zeta)\,|_{i\pm\frac{1}{2},j+\frac{1}{2},k} - (zy_\zeta)\,|_{i\pm\frac{1}{2},j-\frac{1}{2},k}\right) \\
&\approx \left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}}\right) z_{i\pm\frac{1}{2},j,k+\frac{1}{2}} \\
&- \left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right) z_{i\pm\frac{1}{2},j,k-\frac{1}{2}} \\
&- z_{i\pm\frac{1}{2},j+\frac{1}{2},k}\left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}\right) \\
&+ z_{i\pm\frac{1}{2},j-\frac{1}{2},k}\left(y_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right) \\
&= \left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}}\right) z_{i\pm\frac{1}{2},j,k+\frac{1}{2}} \\
&- \left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right) z_{i\pm\frac{1}{2},j,k-\frac{1}{2}} \\
&- z_{i\pm\frac{1}{2},j+\frac{1}{2},k}\left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}\right) \\
&+ z_{i\pm\frac{1}{2},j-\frac{1}{2},k}\left(y_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right).
\end{aligned}
\tag{5.121}
$$

Similarly,

$$
\begin{aligned}
m_1|_{i,j\pm\frac{1}{2},k} &= (y_\zeta z)_\xi\,|_{i,j\pm\frac{1}{2},k} - (zy_\xi)_\zeta\,|_{i,j\pm\frac{1}{2},k} \\
&\approx \left(y_{i+\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - y_{i+\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right) z_{i+\frac{1}{2},j\pm\frac{1}{2},k} \\
&- \left(y_{i-\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - y_{i-\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right) z_{i-\frac{1}{2},j\pm\frac{1}{2},k} \\
&- z_{i,j\pm\frac{1}{2},k+\frac{1}{2}}\left(y_{i+\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - y_{i-\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}}\right) \\
&+ z_{i,j\pm\frac{1}{2},k-\frac{1}{2}}\left(y_{i+\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}} - y_{i-\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right),
\end{aligned}
\tag{5.122}
$$

$$
\begin{aligned}
o_1|_{i,j,k\pm\frac{1}{2}} &= (y_\xi z)_\eta\,|_{i,j,k\pm\frac{1}{2}} - (zy_\eta)_\xi\,|_{i,j,k\pm\frac{1}{2}} \\
&\approx \left(y_{i+\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - y_{i-\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}}\right) z_{i,j+\frac{1}{2},k\pm\frac{1}{2}} \\
&- \left(y_{i+\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}} - y_{i-\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\right) z_{i,j-\frac{1}{2},k\pm\frac{1}{2}} \\
&- z_{i+\frac{1}{2},j,k\pm\frac{1}{2}}\left(y_{i+\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - y_{i+\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\right) \\
&+ z_{i-\frac{1}{2},j,k\pm\frac{1}{2}}\left(y_{i-\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - y_{i-\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\right).
\end{aligned}
\tag{5.123}
$$

Comparing coefficients for, e.g., $y_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}$ in (5.117), we see that

$$\begin{aligned} z_{i+\frac{1}{2},j,k+\frac{1}{2}} - z_{i+\frac{1}{2},j+\frac{1}{2},k} + z_{i+\frac{1}{2},j+\frac{1}{2},k} - z_{i,j+\frac{1}{2},k+\frac{1}{2}} \\ + z_{i,j+\frac{1}{2},k+\frac{1}{2}} - z_{i+\frac{1}{2},j,k+\frac{1}{2}} = 0. \end{aligned} \tag{5.124}$$

Analogously, all metric terms cancel out now, and the freestream is preserved also by the numerical discretisation.

By writing $z_{i\pm\frac{1}{2},j,k+\frac{1}{2}} = \frac{1}{2}\left(z_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} + z_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}}\right)$, the expressions for the normals can be simplified to

$$\begin{aligned} n_1|_{i\pm\frac{1}{2},j,k} = \frac{1}{2}\Big(&\left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right)\left(z_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - z_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}\right) \\ &- \left(y_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}\right)\left(z_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - z_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right)\Big), \end{aligned} \tag{5.125a}$$

$$\begin{aligned} n_2|_{i\pm\frac{1}{2},j,k} = \frac{1}{2}\Big(&\left(z_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - z_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right)\left(x_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - x_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}\right) \\ &- \left(z_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - z_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}\right)\left(x_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - x_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right)\Big), \end{aligned} \tag{5.125b}$$

$$\begin{aligned} n_3|_{i\pm\frac{1}{2},j,k} = \frac{1}{2}\Big(&\left(x_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - x_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right)\left(y_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}\right) \\ &- \left(x_{i\pm\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - x_{i\pm\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}\right)\left(y_{i\pm\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - y_{i\pm\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}\right)\Big), \end{aligned} \tag{5.125c}$$

$$\begin{aligned} m_1|_{i,j\pm\frac{1}{2},k} = \frac{1}{2}\Big(&\left(y_{i+\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - y_{i-\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\left(z_{i-\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - z_{i+\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right) \\ &- \left(y_{i-\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - y_{i+\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\left(z_{i+\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - z_{i-\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\Big), \end{aligned} \tag{5.126a}$$

$$\begin{aligned} m_2|_{i,j\pm\frac{1}{2},k} = \frac{1}{2}\Big(&\left(z_{i+\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - z_{i-\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\left(x_{i-\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - x_{i+\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right) \\ &- \left(z_{i-\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - z_{i+\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\left(x_{i+\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - x_{i-\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\Big), \end{aligned} \tag{5.126b}$$

$$\begin{aligned} m_3|_{i,j\pm\frac{1}{2},k} = \frac{1}{2}\Big(&\left(x_{i+\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - x_{i-\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\left(y_{i-\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - y_{i+\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right) \\ &- \left(x_{i-\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - x_{i+\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\left(y_{i+\frac{1}{2},j\pm\frac{1}{2},k+\frac{1}{2}} - y_{i-\frac{1}{2},j\pm\frac{1}{2},k-\frac{1}{2}}\right)\Big), \end{aligned} \tag{5.126c}$$

$$o_1|_{i,j,k\pm\frac{1}{2}} = \frac{1}{2}\Big(\Big(y_{i+\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - y_{i-\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big(z_{i-\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - z_{i+\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big) - \Big(y_{i-\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - y_{i+\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big(z_{i+\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - z_{i-\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big), \tag{5.127a}$$

$$o_2|_{i,j,k\pm\frac{1}{2}} = \frac{1}{2}\Big(\Big(z_{i+\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - z_{i-\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big(x_{i-\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - x_{i+\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big) - \Big(z_{i-\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - z_{i+\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big(x_{i+\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - x_{i-\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big), \tag{5.127b}$$

$$o_3|_{i,j,k\pm\frac{1}{2}} = \frac{1}{2}\Big(\Big(x_{i+\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - x_{i-\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big(y_{i-\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - y_{i+\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big) - \Big(x_{i-\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - x_{i+\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big(y_{i+\frac{1}{2},j+\frac{1}{2},k\pm\frac{1}{2}} - y_{i-\frac{1}{2},j-\frac{1}{2},k\pm\frac{1}{2}}\Big)\Big). \tag{5.127c}$$

The surface area of the cell sides is given by

$$S_{i\pm\frac{1}{2},j,k} = \sqrt{n_1|^2_{i\pm\frac{1}{2},j,k} + n_2|^2_{i\pm\frac{1}{2},j,k} + n_3|^2_{i\pm\frac{1}{2},j,k}}, \tag{5.128a}$$

$$S_{i,j\pm\frac{1}{2},k} = \sqrt{m_1|^2_{i,j\pm\frac{1}{2},k} + m_2|^2_{i,j\pm\frac{1}{2},k} + m_3|^2_{i,j\pm\frac{1}{2},k}}, \tag{5.128b}$$

$$S_{i,j,k\pm\frac{1}{2}} = \sqrt{o_1|^2_{i,j,k\pm\frac{1}{2}} + o_2|^2_{i,j,k\pm\frac{1}{2}} + o_3|^2_{i,j,k\pm\frac{1}{2}}}. \tag{5.128c}$$

6 Conclusions and Future Work

The physical realism of astrophysical hydrodynamical simulations and their feasibility in terms of computational requirements directly depend on the design and the numerical methods of the simulation code, including the choice of the numerical grid, the time integration method and the boundary conditions. In this thesis, we investigate the influence of the numerical grid. Thereby, we focus on methods which are computationally efficient and can easily be implemented in existing codes written for Cartesian and spherical coordinates. These classical coordinate systems do not provide enough flexibility for a wide range of astrophysical applications, including core convection or convection in spherical shells in three dimensions, or their use is not efficient due to grid cells lying outside of the domain of interest or converging grid lines leading to small time steps.

Unstructured grid methods would require to rewrite codes from the scratch, rendering years of code development and testing useless. The computational costs, already exorbitant on the simple and efficient standard grids, would be further increased. Similarly, discontinuous Galerkin methods (e.g., Hesthaven and Warburton, 2008; Mocz et al., 2013) provide in theory high accuracy and flexibility, but at high computational costs. To make discontinuous Galerkin methods efficient enough for any astrophysical application, a lot of methodological improvements are needed.

Composite grids allow, on the one hand, the use of existing codes developed for Cartesian and spherical grid geometries, while on the other hand offering great flexibility in covering complicated computational domains. One very successful example is the combination of two spherical grids to cover spherical shells in three dimensions, the *Yin–Yang* grid (Kageyama and Sato, 2004; Wongwathanarat et al., 2010). But the big disadvantage of this approach is that the necessary interpolation at the grid interface cannot, as we describe in Chapter 4, be done with high order and, at the same time, in a conservative fashion. Noise generated at the grid interfaces lowers the stability of the overall method. We therefore refrain from using composite grids.

Instead, we consider the technique of curvilinear coordinates to be the method of choice. Using a mapping procedure, the problem given in a complicated physical space is transformed in a Cartesian and equidistant computational space, where all standard methods can be directly applied to solve the transformed problem.

Of course, numerical difficulties arise when this technique is implemented. We found the freestream problem to be a very helpful test, despite its evident simplicity. Methods leading to a violation of the freestream preservation as, e.g., WENO finite difference methods and split time integration schemes, will in most cases not yield stable and accurate results. A detailed discussion can be found in Chapter 5.

With the WENO finite volume scheme, however, we are able to solve several challenging test problems both on smooth and on non-smooth grids as the ones for spherical

domains from Calhoun et al. (2008). There is a close correlation between the analytical fact that these mapping functions are not strongly differentiable and the failure of finite difference methods in numerical simulations on these grids. Only when the analytically motivated conditions (5.76) are exactly fulfilled in the numerical implementation, the method will deliver accurate and stable results.

For the PPM scheme, we implemented the curvilinear version of the CTU scheme from Colella (1990). Even though this method works nicely on Cartesian grids and is even superior compared to the split scheme in terms of accuracy (but not in terms of computational efficiency), its stability on both smooth and non-smooth curvilinear grids is much worse. To achieve reliable results, low Courant numbers must be chosen such that the computational costs increase considerably.

Therefore, we plan to implement the method of Colella et al. (2011) instead. They extended their scheme to fourth order by using the *method of lines* approach with Runge–Kutta schemes and suitable methods of calculating the metric derivatives. In this way, the modified PPM scheme can be extended to curvilinear coordinates while keeping its high stability and accuracy. It is even possible to extend this approach to higher orders of accuracy.

For the WENO finite volume scheme, we plan to extend our work to the Navier–Stokes equations with gravity and diffusion in three spatial dimensions. It would be interesting to apply low Mach number methods as the one presented in Kwatra et al. (2009) and Happenhofer et al. (2013) to curvilinear coordinates and further improve their performance for low Mach numbers.

For this purpose, elliptic and parabolic equations must be solved on curvilinear grids. Finite element solvers as the ones developed in Grimm-Strele (2010) and Happenhofer (2013) are easily extendible to curvilinear coordinates. Either the equations must be transformed to the computational space and then solved there with standard methods as described in section 5.3, or they are solved directly in the physical space. Since the grid is structured both in the computational as well as the physical space, the structure of the stiffness matrix does not change. All integration formulae used for calculating the stiffness matrix in the finite element approach can be reused for curvilinear grids with only little or even no changes. It must be verified by numerical experiments which way is the more efficient and accurate one.

Furthermore, the influence of the Mach number on the results needs additional investigations. We assume that the distortions due to the freestream preservation violation for finite difference schemes are hidden when there are fast motions of the fluid. Only with low Mach numbers, the numerical errors get large enough to disturb the numerical solution considerably.

Theoretically, the mapping functions used to create the curvilinear grid can have discontinuities when used in a finite volume formulation. But from numerical experiments we observe that the accuracy of the numerical solution degrades the less smooth the mapping function is. As long as it is possible to use smooth grids, it is advisible to do so.

In the current implementation, the WENO method on curvilinear grids is only second

order accurate since we used the midpoint rule to obtain equation (5.14). Increasing the order is theoretically possible by using higher order quadrature rules (Casper and Atkins, 1993; Zhang et al., 2011), but this goes along with a considerable increase in computational costs. In section 3.2.1, we showed that due to small error constants, second order time integration methods can be more efficient than higher order ones when coupled with the fifth order accurate WENO scheme. Similarly, even if using second order approximations theoretically restricts the overall order of the method, the overall method can still be more cost efficient than higher order ones in practice. Considering the fact that the ansatz spaces in the finite element solvers from Grimm-Strele (2010) and Happenhofer (2013) as well as time-dependent boundary conditions (Carpenter et al., 1995; Fornberg, 1998) limit the overall order to two anyway, we consider the additional programming and computational work required to make the method higher order accurate not worth the effort within the ANTARES framework. But in contrast to the composite grid approach where the order cannot be increased above two without destroying conservation properties, higher orders of accuracy are possible in principle.

From the analytical and numerical results obtained in this thesis we conclude that curvilinear coordinates are an efficient and flexible choice with which a wide range of astrophysical problems can be investigated. They combine the simplicity of standard coordinate systems with enough grid adaptability as it is needed in many astrophysical problems.

Danksagung

Diese Dissertation hätte ohne die Mithilfe vieler nicht geschrieben werden können. Meinem Doktorvater Herbert Muthsam danke ich für die interessante Themenstellung, die Aufnahme in seine Forschungsgruppe und die kenntnisreiche Hilfestellung während des Schreibens. Friedrich Kupka hat mir in vielen Gesprächen die Astrophysik und die Turbulenztheorie näher gebracht und stand jederzeit zu ausgiebigen fachlichen Diskussionen zur Verfügung. Natalie Happenhofer, Patrick Blies, Eva Mundprecht und Bernhard Löw-Baselli haben eine sehr angenehme und fruchtbare Atmosphäre innerhalb der Forschungsgruppe geschaffen. Darüber hinaus danke ich Marcus Page und Steffen Kionke für viele interessante Diskussionen und Anregungen.

Weite Teile meiner Dissertation sind am Max–Planck–Institut für Astrophysik in Garching bei München entstanden. Für die Gastfreundschaft bedanke ich mich sehr, insbesondere bei Ewald Müller, der auch fachlich stets ein kompetenter Ansprechpartner war. Viele wertvolle Anregungen erhielt ich aus Diskussionen mit Annop Wongwathanarat, Philipp Edelmann und Fabian Miczek. Meine Bürogenossen Robert Andrassy, Alexander Kolodzig und Monika Soraisam sorgten für spannenden fachlichen Austausch und auch Ablenkung an so manchem langen Nachmittag.

Ganz besonderer Dank gilt meinen Eltern, die mich in jedem Stadium meiner Dissertation inhaltlich, finanziell und vor allen Dingen psychisch unterstützt haben.

Die Anwendungsbereiche dieser Arbeit sind vielfältig. Daher wurde sie auch von mehreren Seiten finanziell gefördert: durch die Projekte P21742, P25229, P20762, P20973 und P18224 des Fonds zur Förderung der wissenschaftlichen Forschung (FWF) sowie vom Max–Planck–Institut für Astrophysik in Garching. Die meisten Berechnungen wurden auf lokal zur Verfügung stehenden Rechnern durchgeführt. Umfangreichere Berechnungen wurden auf den VSC–Clustern der Wiener Universitäten sowie auf dem Hydra–Cluster des RZG Garching durchgeführt.

Curriculum Vitae

Hannes Grimm-Strele

eMail address `hannes.grimm-strele@gmx.net`

Education

5/2010 – ongoing	**University of Vienna** PhD studies in mathematics Thesis: *Numerical Grids for Spherical Shells and Other Complex Domains* Supervisor: H. J. Muthsam
7/2013 – 8/2013	**Technical University of Istanbul** Research stay as part of the *Summer of HPC* programme
8/2009, 9/2011 – 3/2012, 9/2012 – 10/2012, 3/2013 – 4/2013	**Max–Planck–Institute for Astrophysics, Garching** Extended research stays, supervisor: E. Müller
10/2004 – 4/2010	**University of Vienna** Diploma studies in mathematics Thesis: *Numerical solution of the generalised Poisson equation on parallel computers* Supervisor: H. J. Muthsam
June 24, 2004	**Bismarck-Gymnasium Karlsruhe** Abitur

Conferences and Talks

April 26, 2013	Talk at the University of Würzburg
March 18, 2013	MPA Institute Seminar, Garching
November 27, 2012	Vienna Lunch Talk, Vienna TU
June 24–28, 2012	EU–US HPC Summer School, Dublin
February 27–28, 2012	VSC User Meeting, Neusiedl am See
April, 2011 & April, 2012	Workshops on small scale magnetic fields, Bairisch-Kölldorf
June 14–17, 2011	SimTech 2011, Stuttgart

List of my Publications

Grimm-Strele, H., Kupka, F., Löw-Baselli, B., Mundprecht, E., Zaussinger, F., and Schiansky, P. (2013a). Realistic Simulations of Stellar Surface Convection with ANTARES: I. Boundary Conditions and Model Relaxation. *NewA*. Available at `http://arxiv.org/abs/1305.0743`.

Grimm-Strele, H., Kupka, F., and Muthsam, H. J. (2013b). Curvilinear Grids for WENO Methods in Astrophysical Simulations. *Computer Physics Communications*. Available at `http://arxiv.org/abs/1308.3066`.

Happenhofer, N., Grimm-Strele, H., Kupka, F., Löw-Baselli, B., and Muthsam, H. J. (2013). A low Mach number solver: Enhancing applicability. *JCP*, 236:96 – 118.

Kupka, F., Muthsam, H. J., Zaussinger, F., Grimm-Strele, H., Happenhofer, N., Löw-Baselli, B., Mundprecht, E., and Obertscheider, C. (2010). *Solar Surface Flow Simulations at Ultra-High Resolution*, chapter 4, pages 415 – 425. High Performance Computing in Science and Engineering. Springer Berlin New–York Heidelberg.

Lemmerer, B., Utz, D., Hanslmeier, A., Veronig, A., Thonhofer, S., Grimm-Strele, H., and Karyappa, R. (2013). 2D Segmentation of small convective patterns in RHD simulations. *A&A*. Submitted to A&A.

Muthsam, H. J., Kupka, F., Mundprecht, E., Zaussinger, F., Grimm-Strele, H., and Happenhofer, N. (2010). Simulations of stellar convection, pulsation and semiconvection. In Brummell, N., Brun, A., Miesch, M., and Ponty, Y., editors, *Astrophysical Dynamics: From Stars to Galaxies*, number 271 in Proceedings IAU Symposium, pages 179 – 186.

Bibliography

Asplund, M., Nordlund, A., Trampedach, R., Allende Prieto, C., and Stein, R. F. (2000a). Line formation in solar granulation. I. Fe line shapes, shifts and asymmetries. *A&A*, 359:729 – 742.

Asplund, M., Nordlund, A., Trampedach, R., and Stein, R. F. (2000b). Line formation in solar granulation. II. The photospheric Fe abundance. *A&A*, 359:743 – 754.

Beeck, B., Collet, R., Steffen, M., Asplund, M., Cameron, R. H., Freytag, B., Hayek, W., Ludwig, H.-G., and Schüssler, M. (2012). Simulations of the solar near-surface layers with the CO5BOLD, MURaM, and Stagger codes. *A&A*, 539:A121.

Berger, M. J. (1987). On Conservation at Grid Interfaces. *SIAM Journal on Numerical Analysis*, 24:967 – 984.

Brown, B. P., Vasil, G. M., and Zweibel, E. G. (2012). Energy Conservation and Gravity Waves in Sound-Proof Treatments of Stellar Interiors. Part I. Anelastic Approximations. *ApJ*, 756:109 – 128.

Browning, M. K., Brun, A. S., Miesch, M. S., and Toomre, J. (2007). Dynamo action in simulations of penetrative solar convection with an imposed tachocline. *Astron. Nachr.*, 328:1100 – 1103.

Browning, M. K., Brun, A. S., and Toomre, J. (2004). Simulations Of Core Convection In Rotating A-type Stars: Differential Rotation And Overshooting. *ApJ*, 601:512 – 529.

Cai, T., Chan, K. L., and Deng, L. (2011). Numerical simulation of core convection by a multi-layer semi-implicit spherical spectral method. *JCP*, 230:8698 – 8712.

Calder, A. C., Fryxell, B., Plewa, T., Rosner, R., Dursi, L. J., Weirs, V. G., Dupont, T., Robey, H. F., Kane, J. O., Remington, A. B., Drake, R. P., Dimonte, G., Zingale, M., Timmes, F. X., Olson, K., Ricker, P., MaxNeice, P., and Tufo, H. M. (2002). On Validating an Astrophysical Simulation Code. *The Astrophysical Journal Supplement Series*, 143:201 – 229.

Calhoun, D. A., Helzel, C., and LeVeque, R. J. (2008). Logically Rectangular Grids and Finite Volume Methods for PDEs in Circular and Spherical Domains. *SIAM Review*, 50:723 – 752.

Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. (1988). *Spectral Methods in Fluid Dynamics*. Springer Series in Computational Physics. Springer, Berlin Heidelberg New–York.

Carlson, B. G. (1963). *The numerical theory of neutron transport*, volume 1 (Statistical Physics), chapter 1, pages 1 – 42. Methods in Computational Physics. Advances in Research and Applications.

Carpenter, M. H., Gottlieb, D., Abarbanel, S., and Don, W.-S. (1995). The Theoretical Accuracy of Runge–Kutta Time Discretizations for the Initial Boundary Value Problem: A Study of the Boundary Error. *SIAM J. Sci. Comput.*, 16:1241 – 1252.

Casper, J. and Atkins, H. L. (1993). A Finite-Volume High-Order ENO Scheme for Two-Dimensional Hyperbolic Systems. *JCP*, 106:62 – 76.

Chesshire, G. and Henshaw, W. D. (1990). Composite overlapping meshes for the solution of partial differential equations. *JCP*, 90:1 – 64.

Chiavassa, A., Plez, B., Josselin, E., and Freytag, B. (2009). Radiative hydrodynamics simulations of red supergiant stars: I. Interpretation of interferometric observations. *A&A*, 506:1351 – 1365.

Chorin, A. J. and Marsden, J. E. (1993). *A Mathematical Introduction to Fluid Mechanics*, volume 4 of *Texts in Applied Mathematics*. Springer, New–York Berlin Heidelberg, 3rd edition.

Clune, T. L., Elliott, J. R., Miesch, M. S., Toomre, J., and Glatzmaier, G. A. (1999). Computational aspects of a code to study rotating turbulent convection in spherical shells. *Parallel Computing*, 25:361 – 380.

Colella, P. (1990). Multidimensional upwind methods for hyperbolic conservation laws. *JCP*, 87(1):171 – 200.

Colella, P., Dorr, M. R., Hittinger, J. A. F., and Martin, D. F. (2011). High-order, finite-volume methods in mapped coordinates. *JCP*, 230:2952 – 2976.

Colella, P. and Glaz, H. M. (1985). Efficient solution algorithms for the Riemann problem for real gases. *JCP*, 59:264 – 289.

Colella, P. and Sekora, M. D. (2008). A limiter for PPM that preserves accuracy at smooth extrema. *JCP*, 227:7069 – 7076.

Colella, P. and Woodward, P. R. (1984). The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations. *JCP*, 54:174 – 201.

DeRosa, M. L., Gilman, P. A., and Toomre, J. (2002). Solar Multiscale Convection and Rotation Gradients Studied in Shallow Spherical Shells. *ApJ*, 581:1356.

Donat, R. and Marquina, A. (1996). Capturing Shock Reflections: An Improved Flux Formula. *JCP*, 125:42 – 58.

Evans, L. C. (2002). *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, 2nd edition.

Evonuk, M. and Glatzmaier, G. A. (2006). 2D study of the effects of the size of a solid core on the equatorial flow in giant planets. *Icarus*, 181:458 – 464.

Evonuk, M. and Glatzmaier, G. A. (2007). The effects of rotation rate on deep convection in giant planets with small solid cores. *Planetary and Space Science*, 55:407 – 412.

Fehlberg, E. (1970). Klassische Runge–Kutta–Formeln vierter und niedrigerer Ordnung mit Schrittweiten–Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme. *Computing*, 6(1):61 – 71.

Ferziger, J. H. and Perić, M. (2002). *Computational Methods for Fluid Dynamics*. Springer, Berlin, 3rd edition.

Fornberg, B. (1998). *A Practical Guide to Pseudospectral Methods*, volume 1 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press.

Freytag, B., Steffen, M., and Dorch, B. (2002). Spots on the surface of Betelgeuse — Results from new 3D stellar convection models. *Astron. Nachr.*, 323:213 – 219.

Freytag, B., Steffen, M., Ludwig, H.-G., Wedemeyer-Böhm, S., Schaffenberger, W., and Steiner, O. (2012). Simulations of stellar convection with CO5BOLD. *JCP*, 231:919 – 959.

Gander, M. J. (2005). Optimized Schwarz Methods. *SIAM J. Numer. Anal.*, 44:699 – 731.

Gardiner, T. A. and Stone, J. M. (2008). An unsplit Godunov method for ideal MHD via constrained transport in three dimensions. *JCP*, 227(8):4123 – 4141.

Georgobiani, D., Kosovichev, A., Nigam, R., Nordlund, A., and Stein, R. F. (2000). Numerical Simulations of Oscillation Modes of the Solar Convection Zone. *ApJL*, 530(2):L139.

Glatzmaier, G. A. (1984). Simulations of Stellar Convective Dynamos. I. The Model and Method. *JCP*, 55:461 – 484.

Gottlieb, S., Shu, C.-W., and Tadmor, E. (2001). Strong Stability-Preserving High-Order Time Discretization Methods. *SIAM Review*, 43(1):89 – 112.

Grimm-Strele, H. (2010). Numerical solution of the generalised Poisson equation on parallel computers. Master's thesis, Universität Wien.

Grimm-Strele, H., Kupka, F., Löw-Baselli, B., Mundprecht, E., Zaussinger, F., and Schiansky, P. (2013a). Realistic Simulations of Stellar Surface Convection with ANTARES: I. Boundary Conditions and Model Relaxation. *NewA*. Available at `http://arxiv.org/abs/1305.0743`.

Grimm-Strele, H., Kupka, F., and Muthsam, H. J. (2013b). Curvilinear Grids for WENO Methods in Astrophysical Simulations. *Computer Physics Communications*. Available at `http://arxiv.org/abs/1308.3066`.

Grinstein, F. F., Margolin, L. O., and Rider, W. J. (2007). *Implicit large eddy simulation: computing turbulent fluid dynamics*. Cambridge University Press.

Guillard, H. and Murrone, A. (2004). On the behavior of upwind schemes in the low Mach number limit: II. Godunov type schemes. *Computers & Fluids*, 33:655 – 675.

Guillard, H. and Viozat, C. (1999). On the behaviour of upwind schemes in the low Mach number limit. *Computers & Fluids*, 28:63 – 86.

Happenhofer, N. (2013). *Efficient Time Integration of the Governing Equations in Astrophysical Hydrodynamics*. PhD thesis, Universität Wien.

Happenhofer, N., Grimm-Strele, H., Kupka, F., Löw-Baselli, B., and Muthsam, H. J. (2013). A low Mach number solver: Enhancing applicability. *JCP*, 236:96 – 118.

Hesthaven, J. S. and Warburton, T. (2008). *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, volume 54 of *Texts in Applied Mathematics*. Springer Berlin New–York Heidelberg.

Heun, K. (1900). Neue Methoden zur approximativen Integration der Differentialgleichungen einer unabhängigen Veränderlichen. *Z. Math. Phys*, 45:23 – 38.

Hotta, H., Rempel, M., Yokoyama, T., Iida, Y., and Fan, Y. (2012). Numerical calculation of convection with reduced speed of sound technique. *A&A*, 539:A30.

Jiang, G.-S. and Shu, C.-W. (1996). Efficient Implementation of Weighted ENO Schemes. *JCP*, 126:202 – 228.

Kageyama, A. and Sato, T. (2004). The "Yin-Yang Grid": An Overset Grid in Spherical Geometry. *Geochemistry Geophysics Geosystems*, 5:1 – 15.

Ketcheson, D. I., Macdonald, C. B., and Gottlieb, S. (2009). Optimal implicit strong stability preserving Runge–Kutta methods. *Applied Numerical Mathematics*, 59:373 – 392.

Kifonidis, K. and Müller, E. (2012). On multigrid solution of the implicit equations of hydrodynamics. Experiments for the compressible Euler equations in general coordinates. *A&A*, 544:A47.

Kraaijevanger, J. F. B. M. (1991). Contractivity of Runge-Kutta methods. *BIT*, 31(3):482 – 528.

Kupka, F. (2009a). 3D stellar atmospheres for stellar structure models and asteroseismology. *MmSAI*, 80:701 – 710.

Kupka, F. (2009b). *Turbulent Convection and Simulations in Astrophysics*, chapter 3, pages 49–105. Springer Lecture Notes in Physics 756.

Kupka, F., Happenhofer, N., Higueras, I., and Koch, O. (2012). Total-variation-diminishing implicit–explicit Runge–Kutta methods for the simulation of double-diffusive convection in astrophysics. *JCP*, 231:3561 – 3586.

Kwatra, N., Su, J., Gretarsson, J. T., and Fedkiw, R. (2009). A method for avoiding the acoustic time step restriction in compressible flow. *JCP*, 228:4146 – 4161.

Lemmerer, B., Utz, D., Hanslmeier, A., Veronig, A., Thonhofer, S., Grimm-Strele, H., and Karyappa, R. (2013). 2D Segmentation of small convective patterns in RHD simulations. *A&A*. Submitted to A&A.

LeVeque, R. J. (2004). *Finite–Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press.

LeVeque, R. J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations. Steady State and Time Dependent Problems*. Society for Industrial and Applied Mathematics (SIAM).

Lilly, D. K. (1996). A comparison of incompressible, anelastic and Boussinesq dynamics. *Atmospheric Research*, 40:143 – 151.

Liska, R. and Wendroff, B. (2003). Comparions of Several Difference Schemes on 1D and 2D Test Problems for the Euler Equations. *SIAM Journal on Scientific Computing*, 25:1 – 30. Available at `http://www-troja.fjfi.cvut.cz/~liska/CompareEuler/compare8/`.

Macdonald, C. B. and Ruuth, S. J. (2008). Level Set Equations on Surfaces via the Closest Point Method. *Journal Of Scientific Computing*, 35:219 – 240.

Magic, Z., Collet, R., Asplund, M., Trampedach, R., Hayek, W., Chiavassa, A., Stein, R. F., and Nordlund, A. (2013). The Stagger-grid: A Grid of 3D Stellar Atmosphere Models. I. Methods and General Properties. *A&A*, 557:A26.

Merriman, B. (2003). Understanding the Shu–Osher Conservative Finite Difference Form. *Journal Of Scientific Computing*, 19:309 – 322.

Miczek, F. (2013). *Simulation of low Mach number astrophysical flows*. PhD thesis, TU München.

Mihalas, D. (1978). *Stellar Atmospheres*. W. H. Freeman and Co.

Mocz, P., Vogelsberger, M., Sijacki, D., Pakmor, R., and Hernquist, L. (2013). A discontinuous Galerkin method for solving the fluid and MHD equations in astrophysical simulations. *MNRAS*. Available at `http://arxiv.org/abs/1305.5536`.

Motamed, M., Macdonald, C. B., and Ruuth, S. J. (2011). On the Linear Stability of the Fifth-Order WENO Discretization. *Journal Of Scientific Computing*, 47(2):127 – 149.

Müller, E., Fryxell, B., and Arnett, D. (1991). Instability and clumping in SN 1987A. *A&A*, 251:505 – 514.

Mundprecht, E., Muthsam, H. J., and Kupka, F. (2013). Multidimensional realistic modelling of Cepheid-like variables. I. Extensions of the ANTARES code. *MNRAS*, 435:3191 – 3205.

Muthsam, H. J., Kupka, F., Löw-Baselli, B., Obertscheider, C., Langer, M., and Lenz, P. (2010a). ANTARES – A Numerical Tool for Astrophysical RESearch with applications to solar granulation. *NewA*, 15:460 – 475.

Muthsam, H. J., Kupka, F., Mundprecht, E., Zaussinger, F., Grimm-Strele, H., and Happenhofer, N. (2010b). Simulations of stellar convection, pulsation and semiconvection. In Brummell, N., Brun, A., Miesch, M., and Ponty, Y., editors, *Astrophysical Dynamics: From Stars to Galaxies*, number 271 in Proceedings IAU Symposium, pages 179 – 186.

Muthsam, H. J., Löw-Baselli, B., Obertscheider, C., Langer, M., Lenz, P., and Kupka, F. (2007). High–resolution models of solar granulation: the two–dimensional case. *MNRAS*, 380:1335 – 1340.

Nonomura, T., Iizuka, N., and Fujii, K. (2010). Freestream and vortex preservation properties of high-order WENO and WCNS on curvilinear grids. *Computers & Fluids*, 39(2):197 – 214.

Nordlund, A. (1982). Numerical simulations of the solar granulation. I. Basic equations and methods. *A&A*, 107:1 – 10.

Nordlund, A., Stein, R. F., and Asplund, M. (2009). Solar Surface Convection. *Living Rev. Solar Phys.*, 6:1 – 117. Cited on January 16, 2013.

Obertscheider, C. (2007). *Modelling of solar granulation — Implementation and comparison of numerical schemes*. PhD thesis, Universität Wien.

Pärt-Enander, E. and Sjögreen, B. (1994). Conservative and non-conservative interpolation between overlapping grids for finite volume solutions of hyperbolic problems. *Computers & Fluids*, 23:551 – 574.

Peng, X., Xiao, F., and Takahashi, K. (2006). Conservative constraint for a quasi-uniform overset grid on the sphere. *Q. J. R. Meteorol. Soc.*, 132:979 – 996.

Pereira, T. M. D., Asplund, M., Collet, R., Thaler, I., Trampedach, R., and Leenaarts, J. (2013). How realistic are solar model atmospheres? *A&A*, 554:A118.

Pope, S. B. (2000). *Turbulent Flows*. Cambridge University Press.

Rai, M. M. (1986). A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations. *JCP*, 62:472 – 503.

Rempel, M., Schüssler, M., and Knölker, M. (2009). Radiative Magnetohydrodynamic Simulation of Sunspot Structure. *ApJ*, 691:640.

Robinson, F. J., Demarque, P., Li, L. H., Sofia, S., Kim, Y.-C., Chan, K. L., and Guenther, D. B. (2004). Three-dimensional simulations of the upper radiation–convection transition layer in subgiant stars. *MNRAS*, 347(4):1208 – 1216.

Ronchi, C., Iacono, R., and Paolucci, P. S. (1996). The "Cubed Sphere": A New Method for the Solution of Partial Differential Equations in Spherical Geometry. *JCP*, 124:91 – 114.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics.

Sadourny, R. (1972). Conservative Finite-Difference Approximations of the Primitive Equations on Quasi-Uniform Spherical Grids. *Monthly Weather Review*, 100:136 – 144.

Saltzman, J. (1994). An Unsplit 3D Upwind Method for Hyperbolic Conservation Laws. *JCP*, 115(1):153 – 168.

Sebastian, K. and Shu, C.-W. (2003). Multidomain WENO Finite Difference Method with Interpolation at Subdomain Interfaces. *Journal Of Scientific Computing*, 19:405 – 438.

Shu, C.-W. (2001). High Order Finite Difference and Finite Volume WENO Schemes and Discontinuous Galerkin Methods for CFD. Technical Report 2001-11, ICASE, NASA Langley Research Center.

Shu, C.-W. (2003). High-order Finite Difference and Finite Volume WENO Schemes and Discontinuous Galerkin Methods for CFD. *International Journal of Computational Fluid Dynamics*, 17(2):107 – 118.

Shu, C.-W. and Osher, S. (1988). Efficient implementation of essentially non-oscillatory shock-capturing schemes. *JCP*, 77(2):439 – 471.

Smagorinsky, J. (1963). General Circulation Experiments with the Primitive Equations. I. The Basic Experiment. *MWR*, 91:99 – 164.

Sod, G. A. (1978). A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *JCP*, 27(1):1 – 31.

Stein, R. F. and Nordlund, A. (1998). Simulations of Solar Granulation. I. General Properties. *ApJ*, 499:914 – 933.

Strikwerda, J. C. (1989). *Finite Difference Schemes and Partial Differential Equations*. Wadsworth & Brooks/Cole.

Tannehill, J. C., Anderson, D. A., and Pletcher, R. H. (1997). *Computational Fluid Mechanics and Heat Transfer*. Taylor & Francis.

Taylor, E. M., Wu, M., and Martin, M. P. (2007). Optimization of nonlinear error for weighted essentially non-oscillatory methods in direct numerical simulations of compressible turbulence. *JCP*, 223:384 – 397.

Thompson, J. F., Warsi, Z. U. A., and Wayne Mastin, C. (1985). *Numerical Grid Generation. Foundations and Applications*. North–Holland.

Toomre, J., Augustson, K., Brown, B. P., Browning, M. K., Brun, A. S., Featherstone, N. A., and Miesch, M. S. (2012). New Era in 3-D Modeling of Convection and Magnetic Dynamos in Stellar Envelopes and Cores. In Shibahashi, H., Takata, M., and Lynas-Gray, A. E., editors, *Progress in Solar/Stellar Physics with Helio- and Asteroseismology*, volume 462, page 331.

Toro, E. F. (2009). *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Berlin New–York Heidelberg.

Trampedach, R., Asplund, M., Collet, R., Nordlund, A., and Stein, R. F. (2013). A Grid of 3D Stellar Atmosphere Models of Solar Metallicity: I. General Properties, Granulation and Atmospheric Expansion. *ApJ*, 769:18.

Viallet, M., Baraffe, I., and Walder, R. (2011). Towards a new generation of multi-dimensional stellar evolution models: development of an implicit hydrodynamic code. *A&A*, 531:A86.

Vinokur, M. (1974). Conservation Equations of Gasdynamics in Curvilinear Coordinate Systems. *JCP*, 14:105 – 125.

Visbal, M. R. and Gaitonde, D. V. (2002). On the Use of Higher-Order Finite-Difference Schemes on Curvilinear and Deforming Meshes. *JCP*, 181:155 – 185.

Vögler, A., Shelyag, S., Schüssler, M., Cattaneo, F., Emonet, T., and Linde, T. (2005). Simulations of magneto-convection in the solar photosphere. *A&A*, 429:335 – 351.

Wang, R. and Spiteri, R. J. (2007). Linear Instability of the Fifth-Order WENO Method. *SIAM J. Numer. Anal.*, 45(5):1871 – 1901.

Warming, R. F. and Beam, R. M. (1976). Upwind Second-Order Difference Schemes and Applications in Aerodynamic Flows. *AIAA Journal*, 14(9):1241 – 1249.

Washington, W. M., Buja, L., and Craig, A. (2009). The computational future for climate and Earth system models: on the path to petaflop and beyond. *Phil. Trans. R. Soc. A*, 367(1890):833 – 846.

Wesseling, P. (2001). *Principles of Computational Fluid Dynamics*, volume 29 of *Springer Series in Computational Mathematics*. Springer, Berlin Heidelberg New–York.

Wongwathanarat, A. (2013). Rotation of vectors. Private communication.

Wongwathanarat, A., Hammer, N. J., and Müller, E. (2010). An axis-free overset grid in spherical polar coordinates for simulating 3D self-gravitating flows. *A&A*, 514:1 – 14.

Zaussinger, F. and Spruit, H. (2013). Semiconvection: numerical simulations. *A&A*, 554:A119.

Zhang, R., Zhang, M., and Shu, C.-W. (2011). On the Order of Accuracy and Numerical Performance of Two Classes of Finite Volume WENO Schemes. *Commun. Comput. Phys.*, 9(3):807 – 827.

Zingale, M., Nonaka, A., Almgren, A. S., Bell, J. B., Malone, C. M., and Orvedahl, R. (2013). Low Mach Number Modeling of Convection in Helium Shells on Sub-Chandrasekhar White Dwarfs. I. Methodology. *ApJ*, 764:97 – 110.