MNRAS **000**, 1–23 (2015) Preprint 19 June 2015 Compiled using MNRAS LATEX style file v3.0

Astrophysical hydrodynamics with a high-order discontinuous Galerkin scheme and adaptive mesh refinement

Kevin Schaal[1,2⋆], Andreas Bauer[1]†, Praveen Chandrashekar[3], Rüdiger Pakmor[1], Christian Klingenberg[4], Volker Springel[1,2]

[1]*Heidelberg Institute for Theoretical Studies, Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg, Germany*
[2]*Zentrum für Astronomie der Universität Heidelberg, Astronomisches Recheninstitut, Mönchhofstr. 12-14, 69120 Heidelberg, Germany*
[3]*TIFR Centre for Applicable Mathematics, Bangalore-560065, India*
[4]*Institut für Mathematik, Universität Würzburg, Emil-Fischer-Str. 30, 97074 Würzburg, Germany*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

Solving the Euler equations of ideal hydrodynamics as accurately and efficiently as possible is a key requirement in many astrophysical simulations. It is therefore important to continuously advance the numerical methods implemented in current astrophysical codes, especially also in light of evolving computer technology, which favours certain computational approaches over others. Here we introduce the new adaptive mesh refinement (AMR) code TENET, which employs a high-order Discontinuous Galerkin (DG) scheme for hydrodynamics. The Euler equations in this method are solved in a weak formulation with a polynomial basis by means of explicit Runge-Kutta time integration and Gauss-Legendre quadrature. This approach offers significant advantages over commonly employed finite volume (FV) solvers. In particular, the higher order capability renders it computationally more efficient, in the sense that the same precision can be obtained at significantly less computational cost. Also, the DG scheme inherently conserves angular momentum in regions where no limiting takes place, and it typically produces much smaller numerical diffusion and advection errors than a FV approach. A further advantage lies in a more natural handling of AMR refinement boundaries, where a fall back to first order can be avoided. Finally, DG requires no deep stencils at high order, and offers an improved compute to memory access ratio compared with FV schemes, which is favorable for current and upcoming highly parallel supercomputers. We describe the formulation and implementation details of our new code, and demonstrate its performance and accuracy with a set of two- and three-dimensional test problems. The results confirm that DG schemes have a high potential for astrophysical applications.

Key words: methods: numerical – hydrodynamics

1 INTRODUCTION

Through the availability of ever more powerful computing resources, computational fluid dynamics has become an important part of astrophysical research. It is of crucial help in shedding light on the physics of the baryonic part of our Universe, and contributes substantially to progress in our theoretical understanding of galaxy formation and evolution, of star and planet formation, and of the dynamics of the intergalactic medium. In addition, it plays an important role in planetary science and in solving engineering problems related to experimental space exploration.

In the astrophysics community, most numerical work thus far on solving the Euler equations of ideal hydrodynamics has been carried out with two basic types of codes. On the one hand, there is the broad class of Eulerian methods which utilise classical hydrodynamic solvers operating on fixed Cartesian grids (e.g. Stone et al. 2008), or on meshes which can adjust their resolution in space with the adaptive mesh refinement (AMR) technique (e.g. Fryxell et al. 2000; Teyssier 2002; Mignone et al. 2007; Bryan et al. 2014). On the other hand, there are pseudo-Lagrangian discretisations in the form of smoothed-particle hydrodynamics (SPH), which are other flexible and popular tools to study many astrophysical problems (e.g. Wadsley et al. 2004; Springel 2005).

Some of the main advantages and drawbacks of these methods become apparent if we recall the fundamental difference in their numerical approach, which is that grid codes discretise space whereas SPH decomposes a fluid in terms of mass elements. The traditional discretisation of space used by Eulerian methods yields good convergence properties, and a high accuracy and efficiency

⋆ e-mail: kevin.schaal@h-its.org
† e-mail: andreas.bauer@h-its.org

for many problems. Furthermore, calculations can be straightforwardly distributed onto parallel computing systems, often allowing a high scalability provided there is only little communication between the cells. The discretisation of mass used by SPH on the other hand results in a natural resolution adjustment in converging flows such that most of the computing time and available resolution is dedicated to dense, astrophysically interesting regions. Moreover, Lagrangian methods can handle gas with high advection velocities without suffering from large errors. However, both methods have also substantial weaknesses, ranging from problems with the Galilei-invariance of solutions in the case of grid codes, to a suppression of fluid instabilities and noise in the case of SPH.

This has motivated attempts to combine the advantages of traditional grid-based schemes and of SPH in new methods, such as moving mesh-codes (Springel 2010; Duffell & MacFadyen 2011) or in mesh-free methods that retain a higher degree of accuracy (Lanson & Vila 2008; Hopkins 2014) than SPH. Both of these new developments conserve angular momentum better than plain Eulerian schemes, while still capturing shocks accurately, a feature that is crucial for many applications in astrophysics. However, these new methods need to give up the simple discretisation of space into regular grids, meaning also that their computational efficiency takes a significant hit, because the regular memory access patterns possible for simple structured discretisations are ideal for leveraging a high fraction of the peak performance of current and future hardware.

In fact, the performance of supercomputers has increased exponentially over the last two decades, roughly following the empirical trend of Moore's law, which states that the transistor count and hence performance of computing chips doubles roughly every two years. Soon, parallel computing systems featuring more than 10 million cores and "exascale" performance in the range of 10^{18} floating point operations per second are expected. Making full use of this enormous compute power for astrophysical research will require novel generations of simulation codes with superior parallel scaling and high resilience when executed on more and more cores. Also, since the raw floating point speed of computer hardware grows much faster than memory access speed, it is imperative to search for new numerical schemes that reduce the average number of memory accesses needed per compute operations.

A very interesting class of numerical methods in this context are so-called discontinuous Galerkin schemes (DG), which can be used for a broad range of partial differential equations. Since the introduction of DG (Reed & Hill 1973) and its generalisation to nonlinear problems (Cockburn & Shu 1991, 1989; Cockburn et al. 1989, 1990; Cockburn & Shu 1998), it has been successfully applied in diverse fields of physics such as aeroacoustics, electromagnetism, fluid dynamics, porous media, etc. (Cockburn et al. 2011; Gallego-Valencia et al. 2014). On the other hand, in the astrophysics community the adoption of modern DG methods has been fairly limited so far. However, two recent works suggest that this is about to change. Mocz et al. (2014) presented a DG method for solving the magnetohydrodynamic (MHD) equations on arbitrary grids as well as on a moving Voronoi mesh, and Zanotti et al. (2015) developed an AMR code for relativistic MHD calculations. The present paper is a further contribution in this direction and aims to introduce a novel DG-based hydrodynamical code as an alternative to commonly employed schemes in the field.

DG is a finite element method which incorporates several aspects from finite volume (FV) methods. The partial differential equation is solved in a weak formulation by means of local basis functions, yielding a global solution that is in general discontinuous across cell interfaces. The approach requires communication only between directly neighbouring cells and allows for the exact conservation of physical quantities including angular momentum. Importantly, the method can be straightforwardly implemented with arbitrary spatial order, since it directly solves also for higher order moments of the solution. Unlike in standard FV schemes, this higher order accuracy is achieved without requiring large spatial stencils, making DG particularly suitable for utilising massive parallel systems with distributed memory because of its favourable compute-to-communicate ratio and enhanced opportunities to hide communication behind local computations.

In order to thoroughly explore the utility of DG in real astrophysical applications, we have developed TENET, an MPI-parallel DG code which solves the Euler equations on an AMR grid to arbitrary spatial order. In our method the solution within every cell is given by a linear combination of Legendre polynomials, and the propagation in time is accomplished with an explicit Runge-Kutta (RK) time integrator. A volume and a surface integral has to be computed numerically for every cell in every timestep. The surface integral involves a numerical flux computation which we carry out with a Riemann solver, similar to how this is done in standard Godunov methods. In order to cope with physical discontinuities and spurious oscillations we use a simple minmod limiting scheme.

The goal of this paper is to introduce the concepts of our DG implementation and compare its performance to a standard FV method based on the Reconstruct-Solve-Average (RSA) approach. In this traditional scheme higher-order information is discarded in the averaging process and recomputed in the reconstruction step, leading to averaging errors and numerical diffusion. DG on the other hand does not only update the cell-averaged solution every cell, but also higher order moments of the solution, such that the reconstruction becomes obsolete. This aspect of DG leads to important advantages over FV methods, in particular an inherent improvement of angular momentum conservation and much reduced advection errors, especially for but not restricted to smooth parts of the solutions. These accuracy gains and the prospect to translate them to a lower computational cost at given computational error form a strong motivation to investigate the use of DG in astrophysical applications.

The paper at hand is structured as follows: In Section 2, we present the general methodology of our DG implementation for Cartesian grids. The techniques we adopt for limiting the numerical solution are described in Section 3, and the generalisation to a mesh with adaptive refinement is outlined in Section 4. These sections give a detailed account of the required equations and discretization formulae for the sake of clarity and definiteness, something that we hope does not discourage interested readers. We then validate our DG implementation and compare it to a standard Godunov FV solver with two- and three-dimensional test problems in Section 5. Finally, Section 6 summarises our findings and gives a concluding discussion.

2 DISCONTINUOUS GALERKIN HYDRODYNAMICS

2.1 Euler equations

The Euler equations are conservation laws for mass, momentum, and total energy of a fluid. They are a system of hyperbolic partial

differential equations and can be written in compact form as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{\alpha=1}^{3} \frac{\partial \boldsymbol{f}_\alpha}{\partial x_\alpha} = 0, \tag{1}$$

with the state vector

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ \rho \boldsymbol{v} \\ \rho e \end{pmatrix} = \begin{pmatrix} \rho \\ \rho \boldsymbol{v} \\ \rho u + \frac{1}{2}\rho \boldsymbol{v}^2 \end{pmatrix}, \tag{2}$$

and the flux vectors

$$\boldsymbol{f}_1 = \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ \rho v_1 v_3 \\ (\rho e + p) v_1 \end{pmatrix} \quad \boldsymbol{f}_2 = \begin{pmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ \rho v_2 v_3 \\ (\rho e + p) v_2 \end{pmatrix} \quad \boldsymbol{f}_3 = \begin{pmatrix} \rho v_3 \\ \rho v_1 v_3 \\ \rho v_2 v_3 \\ \rho v_3^2 + p \\ (\rho e + p) v_3 \end{pmatrix}. \tag{3}$$

The unknown quantities are density ρ, velocity $\boldsymbol{v}$, pressure p, and total energy per unit mass e. The latter can be expressed in terms of the internal energy per unit mass u and the kinetic energy of the fluid, $e = u + \frac{1}{2}\boldsymbol{v}^2$. For an ideal gas, the system is closed with the equation of state

$$p = \rho u(\gamma - 1), \tag{4}$$

where γ denotes the adiabatic index.

2.2 Solution representation

We partition the domain by non-overlapping cubical cells, which may be refined using AMR techniques as explained in later sections. Moreover, we follow the approach of a classical modal DG scheme, where the solution in cell K is given by a linear combination of $N(k)$ orthogonal and normalised basis functions ϕ_l^K:

$$\boldsymbol{u}^K(\boldsymbol{x}, t) = \sum_{l=1}^{N(k)} \boldsymbol{w}_l^K(t) \phi_l^K(\boldsymbol{x}). \tag{5}$$

In this way, the dependence on time and space of the solution is split into time-dependent weights, and basis functions which are constant in time. Consequently, the state of a cell is completely characterised by $N(k)$ weight vectors $\boldsymbol{w}_j^K(t)$.

The above equation can be solved for the weights by multiplying with the corresponding basis function ϕ_j^K and integrating over the cell volume. Using the orthogonality and normalisation of the basis functions yields

$$\boldsymbol{w}_j^K = \frac{1}{|K|} \int_K \boldsymbol{u}^K \phi_j^K \, \mathrm{d}V, \quad j = 1, \ldots, N(k), \tag{6}$$

where $|K|$ is the volume of the cell. The first basis function is chosen to be $\phi_1 = 1$ and hence the weight $\boldsymbol{w}_1^K$ is the cell average of the state vector $\boldsymbol{u}^K$. The higher order moments of the state vector are described by weights $\boldsymbol{w}_j^K$ with $j \geq 2$.

The basis functions can be defined on a cube in terms of scaled variables $\boldsymbol{\xi}$,

$$\phi_l(\boldsymbol{\xi}) : [-1, 1]^3 \to \mathbb{R}. \tag{7}$$

The transformation between coordinates $\boldsymbol{\xi}$ in the cell frame of reference and coordinates $\boldsymbol{x}$ in the laboratory frame of reference is

$$\boldsymbol{\xi} = \frac{2}{\Delta x^K}(\boldsymbol{x} - \boldsymbol{x}^K), \tag{8}$$

where Δx^K and $\boldsymbol{x}^K$ are edge length and cell centre of cell K, respectively. For our DG implementation, we construct a set of three-dimensional polynomial basis functions with a maximum degree of k as products of one-dimensional scaled Legendre polynomials $\tilde{P}$:

$$\{\phi_l(\boldsymbol{\xi})\}_{l=1}^{N(k)} = \left\{ \tilde{P}_u(\xi_1) \tilde{P}_v(\xi_2) \tilde{P}_w(\xi_3) | u, v, w \in \mathbb{N}_0 \wedge u + v + w \leq k \right\}. \tag{9}$$

The first few Legendre polynomials are shown in Appendix A. The number of basis functions for polynomials with a maximum degree of k is

$$N(k) = \sum_{u=0}^{k} \sum_{v=0}^{k-u} \sum_{w=0}^{k-u-v} 1 = \frac{1}{6}(k+1)(k+2)(k+3). \tag{10}$$

Furthermore, when polynomials with a maximum degree of k are used, a scheme with spatial order $p = k+1$ is obtained. For example, linear basis functions lead to a scheme which is of second order in space.

2.3 Initial conditions

Given initial conditions $\boldsymbol{u}(\boldsymbol{x}, t = 0) = \boldsymbol{u}(\boldsymbol{x}, 0)$, we have to provide an initial state for the DG scheme which is consistent with the solution representation. To this end, the initial conditions are expressed by means of the polynomial basis on cell K, which will then be

$$\boldsymbol{u}^K(\boldsymbol{x}, 0) = \sum_{l=1}^{N(k)} \boldsymbol{w}_l^K(0) \phi_l^K(\boldsymbol{x}). \tag{11}$$

If the initial conditions at hand are polynomials with degree $\leq k$, this representation preserves the exact initial conditions, otherwise equation (11) is an approximation to the given initial conditions. The initial weights can be obtained by performing an L^2-projection,

$$\min_{\left\{w_{l,i}^K(0)\right\}_l} \int_K \left(u_i^K(\boldsymbol{x}, 0) - u_i(\boldsymbol{x}, 0) \right)^2 \mathrm{d}V, \quad i = 1, \ldots, 5, \tag{12}$$

where $i = 1, \ldots, 5$ enumerates the conserved variables. The projection above leads to the integral

$$\boldsymbol{w}_j^K(0) = \frac{1}{|K|} \int_K \boldsymbol{u}(\boldsymbol{x}, 0) \phi_j^K(\boldsymbol{x}) \, \mathrm{d}V, \quad j = 1, \ldots, N(k), \tag{13}$$

which can be transformed to the reference frame of the cell, viz.

$$\boldsymbol{w}_j^K(0) = \frac{1}{8} \int_{[-1,1]^3} \boldsymbol{u}(\boldsymbol{\xi}, 0) \phi_j(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi}, \quad j = 1, \ldots, N(k). \tag{14}$$

We solve the integral numerically by means of tensor product Gauss-Legendre quadrature (hereafter called Gaussian quadrature) with $(k+1)^3$ nodes:

$$\boldsymbol{w}_j^K(0) \approx \frac{1}{8} \sum_{q=1}^{(k+1)^3} \boldsymbol{u}(\boldsymbol{\xi}_q^{3\mathrm{D}}, 0) \phi_j(\boldsymbol{\xi}_q^{3\mathrm{D}}) \omega_q^{3\mathrm{D}}, \quad j = 1, \ldots, N(k). \tag{15}$$

Here, $\boldsymbol{\xi}_q^{3\mathrm{D}}$ is the position of the quadrature node q in the cell frame of reference, and $\omega_q^{3\mathrm{D}}$ denotes the corresponding quadrature weight. The technique of Gaussian quadrature is explained in more detail in Appendix B.

2.4 Evolution equation for the weights

In order to derive the DG scheme on a cell K, the Euler equations for a polynomial state vector $\boldsymbol{u}^K$ are multiplied by the basis function

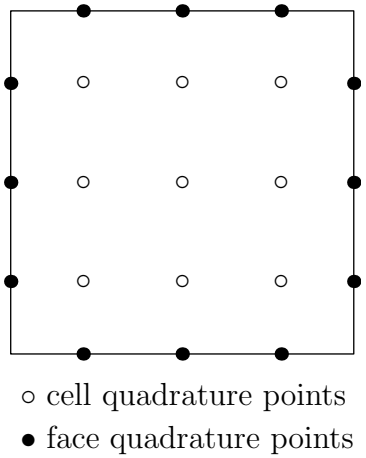

Figure 1. In the DG scheme, a surface and a volume integral has to be computed numerically for every cell, see equation (17). In our approach, we solve these integrals by means of Gauss-Legendre quadrature (Appendix B). This example shows the nodes for our third order DG method (with second order polynomials) when used in a two-dimensional configuration. The black nodes indicate the positions where the surface integral is evaluated, which involves a numerical flux calculation with a Riemann solver. The white nodes are used for numerically estimating the volume integral.

ϕ_j^K and integrated over the cell volume,

$$\int_K \left[\frac{\partial \boldsymbol{u}^K}{\partial t} + \sum_{\alpha=1}^{3} \frac{\partial \boldsymbol{f}_\alpha}{\partial x_\alpha} \right] \phi_j^K \, \mathrm{d}V = 0. \tag{16}$$

Integration by parts of the flux divergence term and a subsequent application of Gauss' theorem leads to

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_K \boldsymbol{u}^K \phi_j^K \, \mathrm{d}V - \sum_{\alpha=1}^{3} \int_K \boldsymbol{f}_\alpha \frac{\partial \phi_j^K}{\partial x_\alpha} \, \mathrm{d}V + \sum_{\alpha=1}^{3} \int_{\partial K} \boldsymbol{f}_\alpha n_\alpha \phi_j^K \, \mathrm{d}S = 0, \tag{17}$$

where $\hat{n} = (n_1, n_2, n_3)^T$ denotes the outward pointing unit normal vector of the surface ∂K. In the following, we discuss each of the terms separately.

According to equation (6) the first term is simply the time variation of the weights,

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_K \boldsymbol{u}^K \phi_j^K \, \mathrm{d}V = |K| \frac{\mathrm{d}\boldsymbol{w}_j^K}{\mathrm{d}t}. \tag{18}$$

The second and third terms are discretised by transforming the integrals to the cell frame and applying Gaussian quadrature (Fig. 1, Appendix B). With this procedure the second term becomes

$$\begin{aligned} &\sum_{\alpha=1}^{3} \int_K \boldsymbol{f}_\alpha \left(\boldsymbol{u}^K(\boldsymbol{x}, t) \right) \frac{\partial \phi_j^K(\boldsymbol{x})}{\partial x_\alpha} \, \mathrm{d}V \\ &= \frac{(\Delta x^K)^2}{4} \sum_{\alpha=1}^{3} \int_{[-1,1]^3} \boldsymbol{f}_\alpha \left(\boldsymbol{u}^K(\boldsymbol{\xi}, t) \right) \frac{\partial \phi_j(\boldsymbol{\xi})}{\partial \xi_\alpha} \, \mathrm{d}\boldsymbol{\xi} \\ &\approx \frac{(\Delta x^K)^2}{4} \sum_{\alpha=1}^{3} \sum_{q=1}^{(k+1)^3} \boldsymbol{f}_\alpha \left(\boldsymbol{u}^K(\boldsymbol{\xi}_q^{\mathrm{3D}}, t) \right) \left. \frac{\partial \phi_j(\boldsymbol{\xi})}{\partial \xi_\alpha} \right|_{\boldsymbol{\xi}_q^{\mathrm{3D}}} \omega_q^{\mathrm{3D}}. \end{aligned} \tag{19}$$

Note that the transformation of the derivative $\partial / \partial x_\alpha$ gives a factor of 2, see equation (8). The volume integral is computed by Gaussian quadrature with $k+1$ nodes per dimension. These nodes allow the exact integration of polynomials up to degree $2k+1$.

The flux functions $\boldsymbol{f}_\alpha$ in the above expression can be evaluated analytically, this is not the case for the fluxes in the last term of the evolution equation (17). This is because the solution is discontinuous across cell interfaces. We hence have to introduce a numerical flux function $\bar{\boldsymbol{f}} \left(\boldsymbol{u}^{K-}, \boldsymbol{u}^{K+}, \hat{n} \right)$, which in general depends on the states left and right of the interface and on the normal vector. With this numerical flux, the third term in equation (17) takes the form

$$\begin{aligned} &\sum_{\alpha=1}^{3} \int_{\partial K} \boldsymbol{f}_\alpha n_\alpha(\boldsymbol{x}) \phi_j^K(\boldsymbol{x}) \, \mathrm{d}S \\ &= \frac{(\Delta x^K)^2}{4} \int_{\partial [-1,1]^3} \bar{\boldsymbol{f}} \left(\boldsymbol{u}^{K-}(\boldsymbol{\xi}, t), \boldsymbol{u}^{K+}(\boldsymbol{\xi}, t), \hat{n}(\boldsymbol{\xi}) \right) \phi_j(\boldsymbol{\xi}) \, \mathrm{d}S_\xi \\ &\approx \frac{(\Delta x^K)^2}{4} \sum_{A \in \partial [-1,1]^3} \sum_{q=1}^{(k+1)^2} \bar{\boldsymbol{f}} \left(\boldsymbol{u}^{K-}(\boldsymbol{\xi}_{q,A}^{\mathrm{2D}}, t), \boldsymbol{u}^{K+}(\boldsymbol{\xi}_{q,A}^{\mathrm{2D}}, t), \hat{n} \right) \phi_j(\boldsymbol{\xi}_{q,A}^{\mathrm{2D}}) \omega_q^{\mathrm{2D}}. \end{aligned} \tag{20}$$

Here for each interface of the normalised cell a two-dimensional Gaussian quadrature rule with $(k+1)^2$ nodes is applied. The numerical flux across each node can be calculated with a one-dimensional Riemann solver, as in ordinary Godunov schemes. For our DG scheme, we use the positivity preserving HLLC Riemann solver (Toro 2009).

In order to model physics extending beyond ideal hydrodynamics, source terms can be added on the right hand side of equation (16). Most importantly, the treatment of gravity is accomplished by the source term

$$\boldsymbol{s} = \begin{pmatrix} 0 \\ -\rho \boldsymbol{\nabla} \Phi \\ -\rho \boldsymbol{v} \cdot \boldsymbol{\nabla} \Phi \end{pmatrix}, \tag{21}$$

which by projecting onto the basis function ϕ_j inside cell K and discretising becomes

$$\begin{aligned} &\int_K \boldsymbol{s}(\boldsymbol{x}, t) \phi_j^K(\boldsymbol{x}) \, \mathrm{d}V \\ &= \frac{|K|}{8} \int_{[-1,1]^3} \boldsymbol{s}(\boldsymbol{\xi}, t) \phi_j(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi} \\ &\approx \frac{|K|}{8} \sum_{q=1}^{(k+1)^3} \boldsymbol{s}(\boldsymbol{\xi}_q^{\mathrm{3D}}, t) \phi_j(\boldsymbol{\xi}_q^{\mathrm{3D}}) \omega_q^{\mathrm{3D}}. \end{aligned} \tag{22}$$

We have now discussed each term of the basic equation (17) and arrived at a spatial discretisation of the Euler equations of the form

$$\frac{\mathrm{d}\boldsymbol{w}_j^K}{\mathrm{d}t} + \boldsymbol{R}_K = 0, \quad j = 1, \ldots, N(k), \tag{23}$$

which represents a system of coupled ordinary differential equations. For the discretisation in time we apply an explicit strong stability preserving Runge-Kutta (SSP RK) scheme (Gottlieb et al. 2001) of the same order as the spatial discretisation. With a combination of a SSP RK method, a positivity preserving Riemann solver, and a positivity limiter (see Section 3.4), negative pressure and density values in the hydro scheme can be avoided. The Butcher tables of the first to fourth order SSP RK methods we use in our code are listed in Appendix D.

2.5 Timestep calculation

If the Euler equations are solved with an explicit time integrator, the timestep has to fulfil a Courant-Friedrichs-Lewy (CFL) condition for achieving numericaltimestep stability. For the DG scheme with

explicit RK time integration, the timestep depends also on the order $p = k + 1$ of the scheme. We calculate the timestep Δt^K of cell K following Cockburn & Shu (1989) as

$$\Delta t^K = \frac{\mathrm{CFL}}{2k+1}\left(\frac{|v_1^K| + c^K}{\Delta x_1^K} + \frac{|v_2^K| + c^K}{\Delta x_2^K} + \frac{|v_3^K| + c^K}{\Delta x_3^K}\right)^{-1}, \tag{24}$$

timestep where $c = \sqrt{\gamma p/\rho}$ is the sound speed and the denominators in the bracket are the edge lengths of the cell. Since we used only cubic cells, we have $\Delta x_1^K = \Delta x_2^K = \Delta x_3^K = \Delta x^K$. In this work we apply a global timestep given by the minimum of the local timesteps among all cells. The CFL number depends on the choice of flux calculation but in practice should be set to a somewhat smaller value as formally required. This is because especially for a higher order RK time integrator a fundamental problem occurs. For calculating the timestep (24), only the current velocity and sound speed of the gas is known, whereas in principle the minimum speed occurring during all RK sub steps should be used to be on the safe side. Unfortunately, this information is not readily available, and the only straight forward way to cope with this problem is to reduce the CFL number. Consequently, we decided to adopt a conservative choice of CFL = 0.2 for the tests presented in this paper. If the positivity limiter is used, the hydro timestep Δt^K has to be modified and can be slightly more restrictive, as described in Section 3.4.

Source terms $\boldsymbol{s}(\boldsymbol{u}, t)$ on the right hand side of the Euler equations can induce additional timestep criteria. In this case, positivity of the solution can be enforced by determining the timestep such that $\rho(\boldsymbol{u}') > 0$ and $p(\boldsymbol{u}') > 0$, with $\boldsymbol{u}' = \boldsymbol{u} + 2\boldsymbol{s}(\boldsymbol{u}, t)\,\Delta t$ (Zhang 2006). For the gravity source term (21) this leads to the timestep limit

$$\Delta t_{\mathrm{grav}}^K \leq \frac{1}{\sqrt{2\gamma(\gamma-1)}}\frac{c}{|\boldsymbol{\nabla}\Phi|}. \tag{25}$$

The actual timestep has then to be chosen as the minimum of the hydrodynamical and gravity timesteps.

2.6 Angular momentum conservation

A welcome side effect of the computation of higher order moments of the solution in DG schemes is that angular momentum is inherently conserved. Without loss of generality, we show this conservation property explicitly for the two-dimensional case ($z = 0$). In this case, the angular momentum density is defined as

$$L = x\rho v_y - y\rho v_x, \tag{26}$$

and the flux momentum tensor in 2D is given by

$$\begin{pmatrix} f_{1,2} & f_{2,2} \\ f_{1,3} & f_{2,3} \end{pmatrix} = \begin{pmatrix} p + \rho v_1^2 & \rho v_1 v_2 \\ \rho v_1 v_2 & p + \rho v_2^2 \end{pmatrix}. \tag{27}$$

The conservation law for angular momentum can be conveniently derived from the Euler equations (1). Multiplying the x-momentum equation by y and the y-momentum equation by x, applying the product rule and subsequently subtracting the two equations gives

$$\frac{\partial L}{\partial t} + \frac{\partial}{\partial x}(x f_{1,3} - y f_{1,2}) + \frac{\partial}{\partial y}(x f_{2,3} - y f_{2,2}) = 0. \tag{28}$$

In order to obtain the angular momentum conservation law on a cell basis, this can be integrated over element K, resulting in

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K L\,\mathrm{d}V + \int_{\partial K} x(f_{1,3}n_1 + f_{2,3}n_2) - y(f_{1,2}n_1 + f_{2,2}n_2)\,\mathrm{d}S = 0. \tag{29}$$

On the other hand, for a DG scheme with order $p > 1$, we can choose the test function to be $\phi^K = y$ in the x-momentum equation and $\phi^K = x$ in the y-momentum equation in the weak formulation (17) of the Euler equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K \rho v_1 y\,\mathrm{d}V - \int_K f_{2,2}\,\mathrm{d}V + \int_{\partial K} \bar{f}_2 y\,\mathrm{d}S = 0, \tag{30}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K \rho v_2 x\,\mathrm{d}V - \int_K f_{1,3}\,\mathrm{d}V + \int_{\partial K} \bar{f}_3 x\,\mathrm{d}S = 0, \tag{31}$$

where $\bar{f}_2$ and $\bar{f}_3$ are the momentum components of the numerical flux function. By subtracting the above equations and using $f_{1,3} = f_{2,2} = \rho v_1 v_2$ we get the angular momentum equation of DG, viz.

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K Ly\,\mathrm{d}V + \int_{\partial K}(\bar{f}_3 x - \bar{f}_2 y)\,\mathrm{d}S. \tag{32}$$

This equation is consistent with the exact equation (29) and hence DG schemes of at least second order accuracy are angular momentum conserving. However, there is one caveat to this inherent feature of DG. In non-smooth regions of the solution a limiting scheme has to be applied, which can slightly modify the angular momentum within a cell and hence lead to a violation of manifest angular momentum conservation. This is also the case for the simple limiters we shall adopt and describe in the subsequent section.

3 SLOPE LIMITING

The choice of the slope limiting procedure can have a large effect on the quality of a hydro scheme, as we will demonstrate in Section 5.2. Often different limiters and configurations represent a trade-off between dissipation and oscillations, and furthermore, the optimal slope limiter is highly problem dependent. Consequently, the challenge consists of finding a limiting procedure which delivers good results for a vast range of test problems. For the DG scheme this proves to be even more difficult. The higher order terms of the solution should be discarded at shocks and contact discontinuities if needed, while at the same time no clipping of extrema should take place in smooth regions, such that the full benefit of the higher order terms is ensured. In what follows we discuss two different approaches for limiting the linear terms of the solution as well as a positivity limiter which asserts non-negativity of density and pressure.

3.1 Component wise limiter

In order to reduce or completely avoid spurious oscillations, we have to confine possible over- and undershootings of the high order solution of a cell at cell boundaries compared to the cell average of neighbouring cells. For that purpose, the weights $w_{2,i}^K, w_{3,i}^K, w_{4,i}^K$ which are proportional to the slopes in the x-, y-, and z-direction, respectively, are limited by comparing them to the difference of cell average values, viz.

$$\begin{aligned}
\tilde{w}_{2,i}^K &= \frac{1}{\sqrt{3}}\mathrm{minmod}\left(\sqrt{3}w_{2,i}^K, \beta(w_{1,i}^K - w_{1,i}^{W_K}), \beta(w_{1,i}^{E_K} - w_{1,i}^K)\right), \\
\tilde{w}_{3,i}^K &= \frac{1}{\sqrt{3}}\mathrm{minmod}\left(\sqrt{3}w_{3,i}^K, \beta(w_{1,i}^K - w_{1,i}^{S_K}), \beta(w_{1,i}^{N_K} - w_{1,i}^K)\right), \\
\tilde{w}_{4,i}^K &= \frac{1}{\sqrt{3}}\mathrm{minmod}\left(\sqrt{3}w_{4,i}^K, \beta(w_{1,i}^K - w_{1,i}^{B_K}), \beta(w_{1,i}^{T_K} - w_{1,i}^K)\right).
\end{aligned} \tag{33}$$

Here, $\tilde{w}_{2,i}^K, \tilde{w}_{3,i}^K, \tilde{w}_{4,i}^K$ are the new weights, $W_K, E_K, S_K, N_K, B_K, T_K$ denote cell neighbours in the directions west, east, south, north,

bottom, and top, respectively, and the minmod-function is defined as

$$\text{minmod}(a,b,c) = \begin{cases} s\min(|a|,|b|,|c|) & s = \text{sign}(a) = \text{sign}(b) = \text{sign}(c) \\ 0 & \text{otherwise.} \end{cases} \tag{34}$$

Each component of the conserved variables, $i = 1,\dots,5$, is limited separately and hence treated independently. The $\sqrt{3}$-factors in equation (33) account for the scaling of the Legendre polynomial $\tilde{P}_1(\xi)$, and the parameter $\beta \in [0.5, 1]$ controls the amount of limiting. The choice of $\beta = 0.5$ corresponds to a total variation diminishing (TVD) scheme for a scalar problem but introduces more diffusion compared to $\beta = 1$. The latter value is less restrictive and may yield a more accurate solution.

If the limited weights are the same as the old weights, i.e. $\tilde{w}^K_{2,i} = w^K_{2,i}$, $\tilde{w}^K_{3,i} = \tilde{w}^K_{3,i}$, and $w^K_{4,i} = w^K_{4,i}$, we keep the original component of the solution $u^K_i(\boldsymbol{x},t)$ of the cell. Otherwise, the component is set to

$$u^K_i(\boldsymbol{x},t) = w^K_{1,i} + \tilde{w}^K_{2,i}\phi^K_2 + \tilde{w}^K_{3,i}\phi^K_3 + \tilde{w}^K_{4,i}\phi^K_4, \tag{35}$$

where terms with order higher than linear have been discarded. Note that for this limiting procedure the cell averaged values do not change, and thus the conservation of mass, momentum, and energy is unaffected. However, the limiter may modify the angular momentum content of a cell, implying that in our DG scheme it is manifestly conserved only in places where the slope limiter does not trigger. Clearly, it is a desirable goal for future work to design limiting schemes which can preserve angular momentum.

3.2 Characteristic limiter

An improvement over the component wise limiting of the conserved variables can be achieved by limiting the characteristic variables instead. They represent advected quantities, and for the Euler equations we can define them locally by linearising about the cell average value.

The transformation matrices $\mathcal{L}^K_x$, $\mathcal{L}^K_y$, and $\mathcal{L}^K_z$ are formed from the left eigenvectors of the flux Jacobian matrix based on the mean values of the conserved variables of cell K, $\bar{\boldsymbol{u}}^K = \boldsymbol{w}^K_1$. We list all matrices in Appendix E. The slopes of the characteristic variables can then be obtained by the matrix-vector multiplications $\boldsymbol{c}^K_2 = \mathcal{L}^K_x \boldsymbol{w}^K_2$, $\boldsymbol{c}^K_3 = \mathcal{L}^K_y \boldsymbol{w}^K_3$, $\boldsymbol{c}^K_4 = \mathcal{L}^K_z \boldsymbol{w}^K_4$, where $\boldsymbol{w}^K_2$, $\boldsymbol{w}^K_3$ and $\boldsymbol{w}^K_4$ denote the slopes of the conserved variables in the $x-$, $y-$, and $z-$directions, respectively. The transformed slopes are limited as in Section 3.1 with the minmod-limiting procedure, viz.

$$\begin{aligned}
\tilde{\boldsymbol{c}}^K_2 &= \frac{1}{\sqrt{3}}\text{minmod}\Big(\sqrt{3}\boldsymbol{c}^K_2, \beta\mathcal{L}^K_x(\boldsymbol{w}^K_1 - \boldsymbol{w}^{W_K}_1), \beta\mathcal{L}^K_x(\boldsymbol{w}^{E_K}_1 - \boldsymbol{w}^K_1)\Big), \\
\tilde{\boldsymbol{c}}^K_3 &= \frac{1}{\sqrt{3}}\text{minmod}\Big(\sqrt{3}\boldsymbol{c}^K_3, \beta\mathcal{L}^K_y(\boldsymbol{w}^K_1 - \boldsymbol{w}^{S_K}_1), \beta\mathcal{L}^K_y(\boldsymbol{w}^{N_K}_1 - \boldsymbol{w}^K_1)\Big), \\
\tilde{\boldsymbol{c}}^K_4 &= \frac{1}{\sqrt{3}}\text{minmod}\Big(\sqrt{3}\boldsymbol{c}^K_4, \beta\mathcal{L}^K_z(\boldsymbol{w}^K_1 - \boldsymbol{w}^{B_K}_1), \beta\mathcal{L}^K_z(\boldsymbol{w}^{T_K}_1 - \boldsymbol{w}^K_1)\Big).
\end{aligned} \tag{36}$$

If the limited slopes of the characteristic variables are identical to the unlimited ones (i.e. $\tilde{\boldsymbol{c}}^K_2 = \boldsymbol{c}^K_2$, $\tilde{\boldsymbol{c}}^K_3 = \boldsymbol{c}^K_3$, and $\tilde{\boldsymbol{c}}^K_4 = \boldsymbol{c}^K_4$) the original solution is kept. Otherwise, the new slopes of the conserved variables are calculated with the inverse transformation matrices $\mathcal{R}^K_x = (\mathcal{L}^K_x)^{-1}$, $\mathcal{R}^K_y = (\mathcal{L}^K_y)^{-1}$, and $\mathcal{R}^K_z = (\mathcal{L}^K_z)^{-1}$ via $\tilde{\boldsymbol{w}}^K_2 = \mathcal{R}^K_x\tilde{\boldsymbol{c}}^K_2$, $\tilde{\boldsymbol{w}}^K_3 = \mathcal{R}^K_x\tilde{\boldsymbol{c}}^K_3$, and $\tilde{\boldsymbol{w}}^K_4 = \mathcal{R}^K_x\tilde{\boldsymbol{c}}^K_4$. In this case, the higher order terms of the solution are set to zero and the limited solution in the cell becomes

$$\boldsymbol{u}^K(\boldsymbol{x},t) = \boldsymbol{w}^K_1 + \tilde{\boldsymbol{w}}^K_2\phi^K_2 + \tilde{\boldsymbol{w}}^K_3\phi^K_3 + \tilde{\boldsymbol{w}}^K_4\phi^K_4. \tag{37}$$

The difference between the component wise limiting of the conserved variables (Section 3.1) and the limiting of the more natural characteristic variables is demonstrated with a shock tube simulation in Section 5.2.

3.3 Total variation bounded limiting

The limiters discussed so far can effectively reduce overshootings and oscillations, however, they can potentially also trigger at smooth extrema and then lead to a loss of higher order information. Considering the goals of a higher order DG scheme, this is a severe drawback that can negatively influence the convergence rate of our DG code. In order to avoid a clipping of the solution at smooth extrema, the minmod-limiter in Sections 3.1 and 3.2 can be replaced by a bounded version (Cockburn & Shu 1998), viz.

$$\text{minmodB}(a,b,c) = \begin{cases} a & \text{if } |a| \le M(\Delta x^K)^2 \\ \text{minmod}(a,b,c) & \text{otherwise.} \end{cases} \tag{38}$$

Here, M is a free parameter which is related to the second derivative of the solution. The ideal choice for it can vary for different test problems. Furthermore, in the above ansatz the amount of limiting depends on the resolution since $|a| \propto \Delta x^K$, which is not the case for the right hand side of the inequality. For reasons of simplicity and generality we would however like to use fixed limiter parameters without explicit resolution dependence for the tests presented in this paper. Hence we define $\tilde{M} = M\Delta x^K$, and use a constant value for $\tilde{M}$ to control the strength of the bounding applied to the minmod limiter. With the minmod-bounded limiting approach, the high accuracy of the solution in smooth regions is retained while oscillations, especially in post-shock regions, can be eliminated.

3.4 Positivity limiting

When solving the equations of hydrodynamics for extreme flows, care has to be taken in order to avoid negative pressure or density values within the cells and at cell interfaces. A classical example are high Mach number flows, for which in the pre-shock region the total energy is dominated by the kinetic energy. Because the pressure is calculated via the difference of total and kinetic energies, it can easily become negative in numerical treatments without special precautions.

The use of an ordinary slope limiter tends to be only of limited help in this situation, and only delays the blow-up of the solution. In fact, it turns out that even with TVD limiting and arbitrarily small timesteps it is not guaranteed in general that unphysical negative values are avoided. Nevertheless, it is possible to construct positivity preserving finite volume and discontinuous Galerkin schemes, which are accurate at the same time. Remarkably, the latter means that high order accuracy can be retained in smooth solutions and furthermore the total mass, momentum, and energy in each cell is conserved. For our DG code, we adopt the positivity limiting implementation following Zhang & Shu (2010).

For constructing this limiter, a quadrature rule including the boundary points of the integration interval is needed. One possible choice consists of m-point Gauss-Lobatto-Legendre (GLL) quadrature rules (Appendix C), which are exact for polynomials of degree $k \le 2m-3$. Let $\xi^{1D}_1 < \dots < \xi^{1D}_{k+1} \in [-1,1]$ be the one-dimensional Gauss quadrature points and $\hat{\xi}^{1D}_1 < \dots < \hat{\xi}^{1D}_m \in [-1,1]$ the GLL

quadrature points. Quadrature rules for the domain $[-1,-1]^3$ which include the interface Gauss quadrature points can be constructed by tensor products of Gauss and GLL quadrature points:

$$S_x = \{(\hat{\xi}_r^{\rm 1D}, \xi_s^{\rm 1D}, \xi_t^{\rm 1D}) : 1 \le r \le m, 1 \le s \le k+1, 1 \le t \le k+1\},$$
$$S_y = \{(\xi_r^{\rm 1D}, \hat{\xi}_s^{\rm 1D}, \xi_t^{\rm 1D}) : 1 \le r \le k+1, 1 \le s \le m, 1 \le t \le k+1\},$$
$$S_z = \{(\xi_r^{\rm 1D}, \xi_s^{\rm 1D}, \hat{\xi}_t^{\rm 1D}) : 1 \le r \le k+1, 1 \le s \le k+1, 1 \le t \le m\}. \quad (39)$$

The union $S = S_x \cup S_y \cup S_z$ of these sets constitutes quadrature points of a rule which is exact for polynomials of degree k and furthermore contains all the points for which we calculate fluxes in equation (20). It can be shown that if the solution is positive at these union quadrature points, the solution averaged after one explicit Euler step will stay positive if a positivity preserving flux calculation and an adequate timestep is used.

The positivity limiter is operated as follows. For a DG scheme with polynomials of maximum order k, choose the smallest integer m such that $m \ge (k+3)/2$. Carry out the following computations for every cell K. First, determine the minimum density at the union quadrature points,

$$\rho^K_{\rm min} = \min_{\boldsymbol{\xi} \in S} \rho^K(\boldsymbol{\xi}). \quad (40)$$

Use this minimum density for calculating the factor

$$\theta_1^K = \min\left\{ \left| \frac{\bar{\rho}^K - \epsilon}{\bar{\rho}^K - \rho^K_{\rm min}} \right|, 1 \right\}, \quad (41)$$

where $\epsilon \approx 10^{-10}$ is a small number representing the target floor for the positivity limiter. Then, modify the higher order terms of the density by multiplying the corresponding weights with the calculated factor,

$$w^K_{j,1} \leftarrow \theta_1^K w^K_{j,1}, \quad j = 2, \ldots, N(k). \quad (42)$$

At this point, the density at the union quadrature points is positive ($\ge \epsilon$), and as desired, the mean density $\bar{\rho}^K = w^K_{1,1}$ and therefore the total mass has not been changed.

In order to enforce pressure positivity at the union quadrature points $\boldsymbol{\xi} \in S$, we compute the factor

$$\theta_2^K = \min_{\boldsymbol{\xi} \in S} \tau^K(\boldsymbol{\xi}), \quad (43)$$

with

$$\tau^K(\boldsymbol{\xi}) = \begin{cases} 1 & \text{if } p^K(\boldsymbol{\xi}) \ge \epsilon \\ \tau_* & \text{such that } p^K(\boldsymbol{u}^K(\boldsymbol{\xi}) + \tau_*(\boldsymbol{u}^K(\boldsymbol{\xi}) - \bar{\boldsymbol{u}}^K)) = \epsilon, \end{cases} \quad (44)$$

and limit all higher order terms ($j >= 2$) for all components of the state vector:

$$w^K_{j,i} \leftarrow \theta_2^K w^K_{j,i}, \quad j = 2, \ldots, N(k), \quad i = 1, \ldots, 5. \quad (45)$$

The idea of the second case of equation (44) is to calculate a modification factor τ_* for the higher order terms $\boldsymbol{u}^K(\boldsymbol{\xi}) - \bar{\boldsymbol{u}}^K$, such that the new pressure at the quadrature point equals ϵ. Furthermore, this second case represents a quadratic equation in τ_*, which has to be solved carefully by minimising round-off errors. In our implementation, we improve the solution for τ_* iteratively with a small number of Newton-Raphson iterations.

The outlined method ensures positivity of pressure and density of the mean cell state after the subsequent timestep under several conditions: Firstly, a positivity preserving flux calculation has to be used, e.g. the Godunov flux or the Harten-Lax-van Leer flux is suitable. Secondly, when adopting a RK method it should be strong stability preserving (SSP); these methods are convex combinations of explicit Euler methods, and hence a number of relevant properties and proofs valid for Euler's method also hold for these higher order time integrators. Lastly, the local timestep for the DG scheme has to be set to

$$\Delta t^K = \mathrm{CFL} \cdot \min\left\{ \frac{1}{2k+1}, \frac{\hat{\omega}_1^{\rm 1D}}{2} \right\} \left(\frac{|v_1^K| + c^K}{\Delta x_1^K} + \frac{|v_2^K| + c^K}{\Delta x_2^K} + \frac{|v_3^K| + c^K}{\Delta x_3^K} \right)^{-1}, \quad (46)$$

where $\hat{\omega}_1^{\rm 1D}$ is the first GLL weight. Compared to (Zhang & Shu 2010) we obtain an additional factor of $1/2$ since our reference domain is $[-1,1]$ and therefore the sum of the GLL weights is $\sum_{q=1}^m \hat{\omega}_q^{\rm 1D} = 2$. The first GLL weight is $\hat{\omega}_1^{\rm 1D} = 1$ for our second order DG method ($k = 1$) and $\hat{\omega}_1^{\rm 1D} = 1/3$ for our third and fourth order method ($k = 2, 3$). Depending on the order, the timestep with positivity limiting can be slightly more restrictive. We conclude this section by remarking that the positivity limiting procedure is a local operation and does not introduce additional inter-cell communication in the DG scheme.

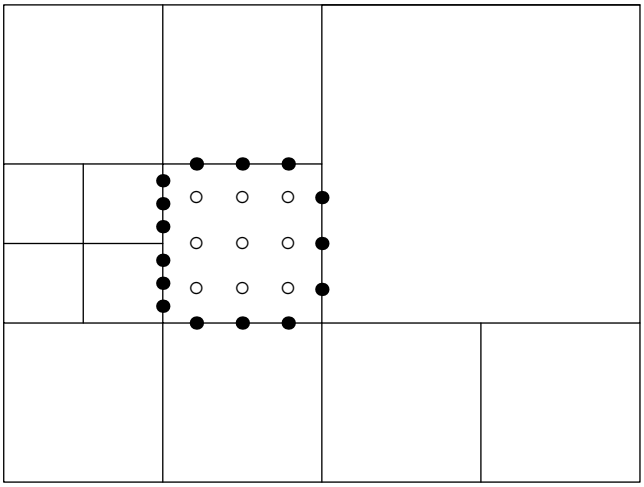

Figure 2. Quadrature points of an AMR boundary cell in our 2D, third order ($k = 2$) version of the code. Interfaces with neighbouring cells on a finer AMR level are integrated with the quadrature points of the smaller cells. In this way no accuracy is lost at AMR boundaries and the order of our DG code is unaffected.

4 DG WITH ADAPTIVE MESH REFINEMENT

Many astrophysical applications involve a high dynamic range in spatial scales. In grid based codes, this multi-scale challenge can be tackled with the adaptive mesh refinement technique (AMR). In this approach, individual cells are split into a set of smaller subcells if appropriate (see Fig. 3 for a sketch of the two-dimensional case), thereby increasing the resolution locally. We adopt a tree-based AMR implementation where cubical cells are split into 8 subcells when a refinement takes place. This allows a particularly flexible adaption to the required spatial resolution.

Our implementation largely follow that in the RAMSES code (Teyssier 2002). The tree is organized into cells and nodes. The root node encompasses the whole simulation domain and is designated as level $l = 0$. A node at level l always contains 8 children, which can be either another node on a smaller level $l' = l + 1$, or a leave

cell. In this way, the mesh of leave cells is guaranteed to be volume filling. An example of such an AMR mesh in 2D is shown in Fig. 2.

To distribute the work load onto many MPI processes, the tree is split into an upper part, referred to as 'top tree' in the following, and branches that hang below individual top tree nodes. Every MPI process stores a full copy of the top tree, whereas each of the lower branches are in principle stored only on one MPI rank. However, some of the branch data is replicated in the form of a ghost layer around each local tree to facilitate the exchange of boundary information. The mesh can be refined and structured arbitrarily, with only one restriction. The level jump between a cell and its 26 neighbours has to be at most ±1. This implies that a cell can either have 1 or 4 split cells as direct neighbours in a given direction. In order to fulfil this level jump condition additional cells may have to be refined beyond those where the physical criterion demands a refinement.

To simplify bookkeeping, we store for each cell and node the indices of the father node and the 6 neighbouring cells or nodes. In case of a split neighbour, the index of the node at the same level is stored instead. To make the mesh smoother and guarantee a buffer region around cells which get refined due to the physical refinement criterion, additional refinement layers are added as needed. In the AMR simulations presented in this work, we use one extra layer of refined cells around each cell flagged by the physical criterion for refinement.

4.1 Refinement criterion

Many different refinement strategies can be applied in an AMR code, for example, the refinement and derefinement criterion may aim to keep the mass per cell approximately constant. Another common strategy for refinement is to focus on interesting regions such as shocks, contact discontinuities, or fluid instabilities. In this work, we apply the mesh refinement simply at locations where the density gradient is steep. To be precise, cell K is refined if the following criterion is met:

$$\max(w_{2,0}^K, w_{3,0}^K, w_{4,0}^K) > \alpha \cdot w_\mathrm{t}. \tag{47}$$

Here, $w_{2,0}^K$, $w_{3,0}^K$, and $w_{4,0}^K$ are the density changes divided by $\sqrt{3}$ along the $x-$, $y-$, and $z-$ directions, respectively. The refinement is controlled by the target slope parameter w_t and a range factor α. We have introduced the latter with the purpose of avoiding an oscillating refinement-derefinement behaviour. In this work, we adopt the values $w_\mathrm{t} = 0.01$ and $\alpha = 1.1$. The reason for using the density changes instead of the physical slopes is that in this way a runaway refinement can be avoided.

Leave nodes of the AMR tree are kept refined if the following equation is true:

$$\max(w_{2,0}^L, w_{3,0}^L, w_{4,0}^L) > \frac{1}{\alpha} \cdot w_\mathrm{t}. \tag{48}$$

The weights on the left hand side are the leave node weights calculated with a projection of the 8 subcell solutions. If inequality (48) does not hold, the node gets derefined and the 8 subcells are merged into one.

4.2 Mesh refinement

Let $K = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [-1,1]^3\}$ be the cell which we want to refine into 8 subcells, and A, B, C, ... , H denote the daughter cells, as illustrated in Fig. 3 for the two-dimensional case. The refinement can be

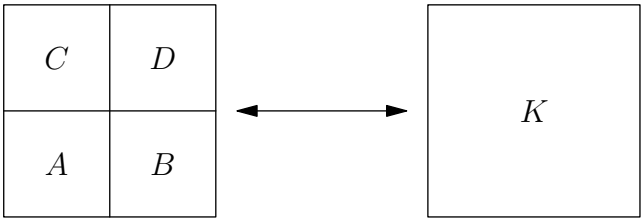

Figure 3. Refinement (arrow to the left) and coarsening (arrow to the right) of a cell in the 2D version of TENET. In both operations the solution on the new cell structure is inferred by an L^2-projection of the current polynomial solution. In doing so, no information is lost in a refinement, and as much information as possible is retained in a derefinement.

achieved without higher-order information loss if the solution polynomial of the coarse cell is correctly projected onto the subcells. In the following, we outline the procedure for obtaining the solution on subcell $A = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [-1,0] \times [-1,0] \times [-1,0]\}$, the calculations for the other subcells are done in an analogous way.

The weights of the solution on subcell A are obtained by solving the minimisation problem

$$\min_{\left\{w_{l,i}^A\right\}_l} \int_A \left(u_i^K - u_i^A\right)^2 \mathrm{d}V, \quad i = 1, \ldots, 5, \tag{49}$$

where $\boldsymbol{u}^K = \sum_{l=1}^{N(k)} \boldsymbol{w}_l^K \phi_l^K$ is the given solution on the coarse cell and $\boldsymbol{u}^A = \sum_{l=1}^{N(k)} \boldsymbol{w}_l^A \phi_l^A$ are the polynomials of the conserved variables on cell A we are looking for. The minimisation ansatz (49) is solved by projecting the solution of the coarse cell onto the basis functions of subcell A,

$$\boldsymbol{w}_j^A = \frac{1}{|A|} \int_A \boldsymbol{u}^K \phi_j^A \, \mathrm{d}V, \quad j = 1, \ldots, N(k). \tag{50}$$

We can plug in the solution $\boldsymbol{u}^K$ and move the time- but not space-dependent weights in front of the integral,

$$w_{j,i}^A = \sum_{l=1}^{N(k)} w_{l,i}^K \frac{1}{|A|} \int_A \phi_l^K \phi_j^A \, \mathrm{d}V, \quad i = 1, \ldots, 5, j = 1, \ldots, N(k). \tag{51}$$

If we define the projection matrix

$$(\mathsf{P}_A)_{l,j} = \frac{1}{|A|} \int_A \phi_l^A \phi_j^K \, \mathrm{d}V, \tag{52}$$

and introduce for each conserved variable $i = 1, \ldots, 5$ the weight vector $\hat{\boldsymbol{w}}_i = (w_{1,i}, w_{2,i}, \ldots, w_{N,i})^\top$, the weights of the sub-cell solution $\boldsymbol{u}^A$ are simply given by the matrix-vector multiplications

$$\hat{\boldsymbol{w}}_i^A = \mathsf{P}_A \hat{\boldsymbol{w}}_i^K \quad i = 1, \ldots, 5. \tag{53}$$

The matrix (52) can be computed exactly by transforming the integral to the reference domain of cell A,

$$(\mathsf{P}_A)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 - 1}{2}, \frac{\xi_2 - 1}{2}, \frac{\xi_3 - 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}, \tag{54}$$

and applying a Gaussian quadrature rule with $(k+1)^3$ points. The projection integrals for the other subcells (B, C, ..., H) are given in Appendix F. The refinement matrices are the same for all the cells, we calculate them once in the initialisation of our DG code.

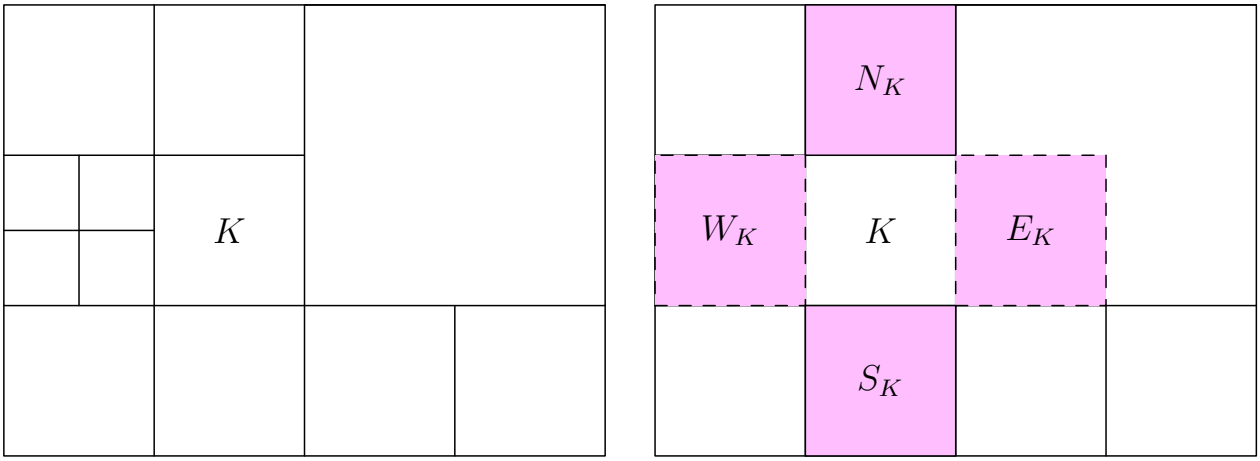

Figure 4. Definition of neighbours in the slope limiting procedure at AMR boundaries, shown for the 2D version of TENET. The neighbours of cell K on the left are on a finer level. For the slope limiting of cell K, the node weights are used, which are calculated by projecting the solutions of the subcells onto the encompassing node volume W_K. The right neighbour of cell K on the other hand is here a coarser cell, in this case the solution of this neighbour has to be projected onto the smaller volume E_K. In the slope limiting routine of TENET, this is only done in a temporary fashion without actually modifying the mesh.

4.3 Mesh derefinement

When the 8 cells A, B, ..., H are merged into one coarser cell K, some information is unavoidably lost. Nevertheless, in order to retain as much accuracy as possible, the derefinement should again be carried out by means of an L^2-projection,

$$\min_{\left\{w_{l,i}^K\right\}_l} \int_A \left(u_i^K - u_i^A\right)^2 \,\mathrm{d}V + \int_B \left(u_i^K - u_i^B\right)^2 \,\mathrm{d}V + \ldots + \int_H \left(u_i^K - u_i^H\right)^2 \,\mathrm{d}V, \quad i = 1, \ldots, 5. \tag{55}$$

Here, $\boldsymbol{u}^A$, $\boldsymbol{u}^B$, ..., $\boldsymbol{u}^H$ denote the given solutions on the cells to be derefined, and $\boldsymbol{u}^K$ is the solution on the coarser cell we are looking for. The minimisation problem is solved by the weights

$$\boldsymbol{w}_j^K = \frac{1}{|K|}\left(\int_A \boldsymbol{u}^A \phi_j^K \,\mathrm{d}V + \int_B \boldsymbol{u}^B \phi_j^K \,\mathrm{d}V + \ldots + \int_H \boldsymbol{u}^H \phi_j^K \,\mathrm{d}V\right), \quad j = 1, \ldots, N(k). \tag{56}$$

We insert the solutions on the cells A, B, ..., H and use the definition of the refinement matrices,

$$\begin{aligned} w_{j,i}^K &= \frac{1}{|K|}\left(\int_A \sum_{l=1}^{N(k)} w_{l,i}^A \phi_l^A \phi_j^K \,\mathrm{d}V + \ldots + \int_H \sum_{l=1}^{N(k)} w_{l,i}^H \phi_l^H \phi_j^K \,\mathrm{d}V\right) \\ &= \frac{1}{|K|}\left(\sum_{l=1}^{N(k)} w_{l,i}^A |A| (\mathsf{P}_A)_{j,l} + \ldots + \sum_{l=1}^{N(k)} w_{l,i}^H |H| (\mathsf{P}_H)_{j,l}\right), \\ & i = 1, \ldots, 5 \quad , \quad j = 1, \ldots, N(k). \end{aligned} \tag{57}$$

If we again use the vector $\hat{\boldsymbol{w}}_i^K$ consisting of the weights of the solution for the conserved variable i, the weights of the solution on cell K can be computed with the transposed refinement matrices,

$$\hat{\boldsymbol{w}}_i^K = \frac{1}{8}\left(\mathsf{P}_A^\top \hat{\boldsymbol{w}}_i^A + \mathsf{P}_B^\top \hat{\boldsymbol{w}}_i^B + \ldots + \mathsf{P}_H^\top \hat{\boldsymbol{w}}_i^H\right), \quad i = 1, \ldots, 5. \tag{58}$$

Here we have used the fact that the cells to be merged have the same volume, $|A| = |B| = \ldots = |H| = \frac{1}{8}|K|$. While the refinement of 8 cells into a subcell preserves the exact shape of the solution, this is not the case for the derefinement. After derefining we limit the solution before further calculations are carried out, especially for asserting positivity.

4.4 Limiting with AMR

In the case of AMR boundary cells, the limiting procedure has to be well defined. For the limiting of cell K, the slope limiters described in Sections 3.1 and 3.2 need the average values of the neighbouring cells in each direction in order to adjust the slope of the conserved variables. However, if the cell neighbours are on a different AMR level, they are smaller or larger compared to the cell to limit, and a single neighbouring cell in a specific direction is not well defined.

Fortunately, there is a straight forward way in DG to cope with this problem. If a neighbouring cell is on a different level, the polynomials of the neighbour are projected onto a volume which is equal to the volume of the cell to refine, see Fig. 4. This can be done with the usual refinement and derefinement operations as described in Sections 4.2 and 4.3. By doing so, the limiting at AMR boundaries can be reduced to the limiting procedure on a regular grid. In AMR runs, we also slightly adjust the positivity limiting scheme. For cells which have neighbours on a higher level, the positivity limiter is not only applied at the locations of the union quadrature points (39), but additionally at the quadrature points of the cell interfaces to smaller neighbours. By doing so, negative values in the initial conditions of the Riemann problems are avoided, which have to be solved in the integration of these interfaces.

4.5 Main simulation loop

Our new TENET code has been developed as an extension of the AREPO code (Springel 2010), allowing us to reuse AREPO's input-output routines, domain decomposition, basic tree infrastructure, neighbour search routines, and gravity solver. This also helps to make our DG scheme quickly applicable in many science applications. We briefly discuss the high-level organisation of our code and the structure of the main loop in a schematic way, focussing on the DG part:

(i) Compute and store quadrature points and weights, basis function values, and projection matrices.

(ii) Set the initial conditions by means of an L^2-projection.

(iii) Apply slope limiter.

(iv) Apply positivity limiter.

(v) While $t < t_{\rm max}$:

(a) Compute timestep.

(b) For every Runge-Kutta stage:

(1) Calculate $\boldsymbol{R}_K$ of the differential equation (23) (inner and outer integral).

(2) Update solution to next RK stage.

(3) Update node data.

(4) Apply slope limiter.

(5) Apply positivity limiter.

(c) Do refinement and derefinement.

(d) $t = t + \Delta t$.

The basis function values and the quadrature data are computed in a general way for arbitrary spatial order as outlined in Appendices A, B, and C. For the time integration, our code can be provided with a general Butcher tableau (Table D1), specifying the desired RK method. By keeping these implementations general, the spatial order and time integrator can be changed conveniently.

The first step of a RK stage consists of calculating $\boldsymbol{R}_K$ of the differential equation (23), including possible source terms. This involves computing the inner integral by looping over the cells and the outer integral by looping over the cell interfaces. After updating the solution weights for every cell, the hydrodynamical quantities on the AMR nodes have to be updated, such that they can be subsequently accessed during the slope-limiting procedure (Section 4.4). After the RK step, all cells are checked for refinement and derefinement and the mesh is adjusted accordingly.

5 VALIDATION

In this section we discuss various test problems which are either standard tests or chosen for highlighting a specific feature of the DG method. For most of the test simulations we compare the results to a traditional second-order FV method. For definiteness, we use the AREPO code to this end, with a fixed Cartesian grid and its standard solver as described in Springel (2010). The latter consists of a second-order unsplit Godunov scheme with an exact Riemann solver and a non-TVD slope limiter. Recently, some modifications to the FV solver of AREPO have been introduced for improving its convergence properties (Pakmor et al. 2015) when the mesh is dynamic. However, for a fixed Cartesian grid this does not make a difference and the old solver used here performs equally well.

There are several important differences between FV and DG methods. In a FV scheme, the solution is represented by piecewise constant states, whereas in DG the solution within every cell is a polynomial approximation. Moreover, in FV a reconstruction step has to be carried out in order to recreate higher order information. Once the states at the interfaces are calculated, numerical fluxes are computed and the mean cell values updated. In the DG method, no higher order information is discarded after completion of a step and therefore no subsequent reconstruction is needed. DG directly solves also for the higher order moments of the solution and updates the weights of the basis functions in every cell accordingly.

For all the DG tests presented in this section we use a SSP RK time integrator of an order consistent with the spatial discretisation, and the fluxes are calculated with the HLLC approximate Riemann solver. Furthermore, if not specified otherwise, we use for all tests the positivity limiter in combination with the characteristic limiter in the bounded version with the parameters $\beta = 1$ and $\tilde{M} = 0.5$. Specifically, we only deviate from this configuration when we compare the limiting of the characteristic variables to the limiting of the conserved variables in the shock tube test in Section 5.2, and when the angular momentum conservation of our code is demonstrated in Section 5.5 with the cold Keplerian disc problem. The higher order efficiency of our DG code is quantified in the test problem of Section 5.1, and the 3D version of our code is tested in a Sedov-Taylor blast wave simulation in Section 5.3. In Section 5.4 we show that advection errors are much smaller for DG compared to FV. Finally, the AMR capabilities of TENET are illustrated with a high-resolution Kelvin-Helmholtz instability test in Section 5.6.

5.1 Isentropic vortex

In this first hydrodynamical test problem, we verify the correctness of our DG implementation by measuring the convergence rate towards the analytic solution for different orders of the scheme. Additionally, we investigate the precision of the FV and DG schemes as a function of computational cost.

An elementary test for measuring the convergence of a hydrodynamical scheme is the simulation of one-dimensional traveling waves at different resolutions (e.g. Stone et al. 2008). However, the DG scheme performs so well in this test, especially at higher order, that the accuracy is very quickly limited by machine precision, making the traveling sound wave test impractical for convergence studies of our DG implementation. We hence use a more demanding setup, the stationary and isentropic vortex in two dimensions (Yee et al. 1999). The primitive variables density, pressure, and velocities in the initial conditions are

$$p(r) = \rho^\gamma,$$

$$\rho(r) = \left[1 - \frac{(\gamma-1)\beta^2}{8\gamma\pi^2}\exp(1-r^2)\right]^{\frac{1}{\gamma-1}},$$

$$v_x(r) = -(y-y_0)\frac{\beta}{2\pi}\exp\left(\frac{1-r^2}{2}\right),$$

$$v_y(r) = (x-x_0)\frac{\beta}{2\pi}\exp\left(\frac{1-r^2}{2}\right),$$

with $\beta = 5$, and the adiabatic index $\gamma = 7/5$. With this choice for the primitive variables, the centrifugal force at each point is exactly balanced by the pressure gradient pointing towards the centre of the vortex, yielding a time invariant situation.

The vortex is smooth and stationary, and every change during the time integration with our numerical schemes can be attributed mainly on numerical truncation errors. In order to break the spherical symmetry of the initial conditions, we additionally add a mild velocity boost of $v_{\rm b} = (1, 1)$ everywhere. The simulation is carried out in the periodic domain $(x, y) \in [0, 10]^2$ with the centre of the vortex at $(x_0, y_0) = (5, 5)$ and run until $t = 10$, corresponding to one box crossing of the vortex. We compare the obtained numerical solution with the analytic solution, which is given by the initial conditions. The error is measured by means of the L1 density norm, which we define for the FV method as

$$\mathrm{L1} = \frac{1}{N}\sum_i |\rho_i' - \rho_i^0|, \tag{59}$$

where N is the total number of cells, and ρ_i' and ρ_i^0 denote the den-

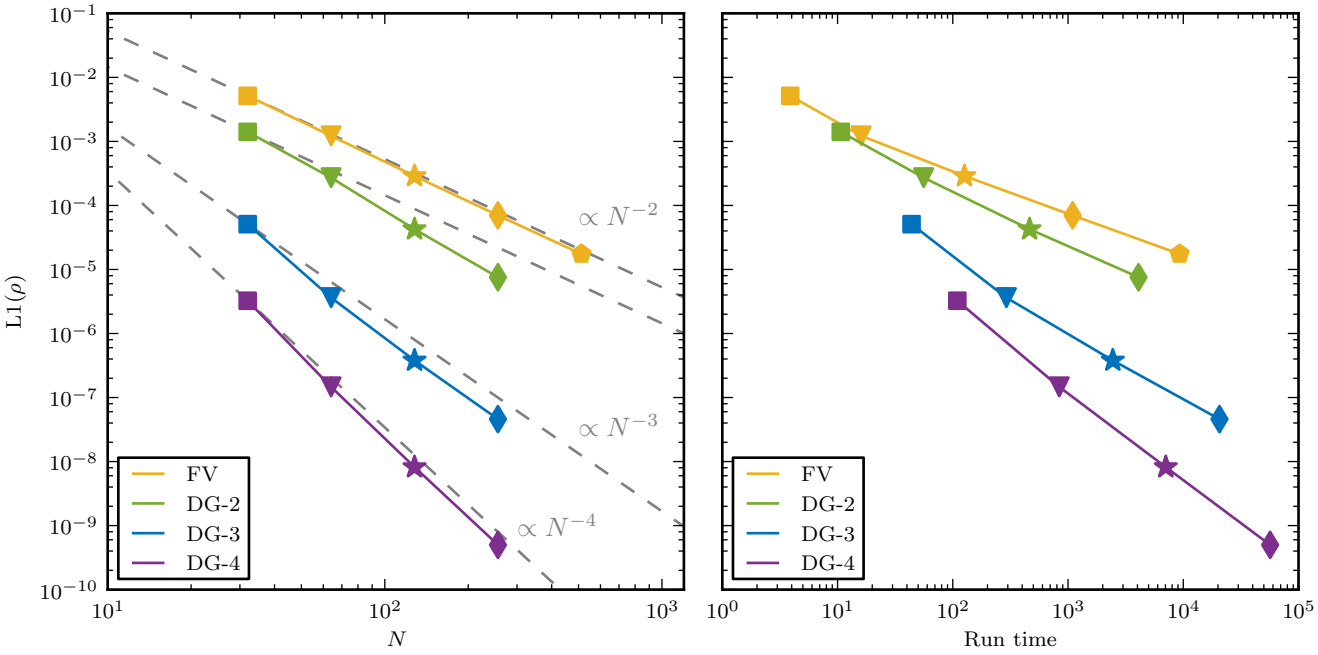

Figure 5. *Left panel:* L1 error norm as a function of linear resolution for the two-dimensional isentropic vortex test. Each data point corresponds to a simulation, different colours indicate the different methods. We find a convergence rate as expected (dashed line) or slightly better for all schemes, indicating the correctness of our DG implementation. *Right panel:* L1 error norm versus the measured run time of the simulations. The second order FV implementation and the second order DG (DG-2) realisation are approximately equally efficient in this test, i.e. a given precision can be obtained with a similar computational cost. In comparison, the higher order methods can easily be faster by more than an order of magnitude for this smooth problem. This illustrates the fact that an increase of order (p-refinement) of the numerical scheme can be remarkably more efficient than a simple increase of grid resolution (h-refinement).

sity in cell i in the final state and in the initial conditions, respectively. This norm should be preferred over the L2 norm, since it is more restrictive.

For DG the error is inferred by calculating the integral

$$\mathrm{L1} = \frac{1}{V}\int_V |\rho'(x,y) - \rho^0(x,y)|\,\mathrm{d}V, \tag{60}$$

with the density solution polynomial $\rho'(x,y)$ and the analytic expression $\rho^0(x,y)$ of the initial conditions. We compute the integral numerically with accuracy of order $p+2$, where p is the order of the DG scheme. In this way, we assert that the error measurement can not be dominated by errors in the numerical integration of the error norm.

The result of our convergence study is shown in the left panel of Figure 5. For every method we run the simulation with several resolutions, indicated by different symbols. The L1 errors decrease with resolution and meet the expected convergence rates given by the dashed lines, pointing towards the correct implementation of our DG scheme. At a given resolution, the second order DG code achieves a higher precision compared to the second order FV code, reflecting the larger number of flux calculations involved in DG. The convergence rates are equal, however, as expected. The higher order DG formulations show much smaller errors, which also drop more rapidly with increasing resolution.

In the right panel of Fig. 5, we show the same plot but substitute the linear resolution on the horizontal axis with the measured run time, which directly indicates the computational cost.[1] By doing so we try to shed light on the question of the relative speed of the methods (or computational cost required) for a given precision. Or alternatively, this also shows the precision attained as a function of the computational cost.

For the following discussion, we want to remark that the isentropic vortex is a smooth 2D test problem; slightly different conclusions may well hold for problems involving shocks or contact discontinuities, as well as for 3D tests. The finite volume method consumes less time than the DG-2 method when runs with equal resolution are compared. On the other hand, the error of the DG-2 scheme is smaller. As Fig. 5 shows, when both methods are quantitatively compared at equal precision, the computational cost is essentially the same, hence the overall efficiency of the two second-order schemes is very similar for this test problem. Interestingly, the 64^2 cells FV run (yellow triangle) and the 32^2 cells DG-2 simulation (green square) give an equally good result at almost identical runtime.

However, the higher order schemes (DG-3, DG-4) show a significant improvement over the second order methods in terms of efficiency, i.e. a prescribed target accuracy can be reached much faster by them. In particular, the third order DG scheme performs clearly favourably over the second order scheme. The fourth order DG method uses the SSP RK-4 time integrator, which has already 5 stages, in contrast to the time integrators of the lower order methods, which have the optimal number of p stages for order p. Moreover, the timestep for the higher order methods is smaller, accord-

[1] The simulations were performed with four MPI-tasks on a conventional desktop machine, shown is the wall-clock time in seconds. A similar trend could also be observed for larger parallel environments; while DG is a higher order scheme, due to its discontinuous nature the amount of required communication is comparable to that of a traditional second-order FV method.

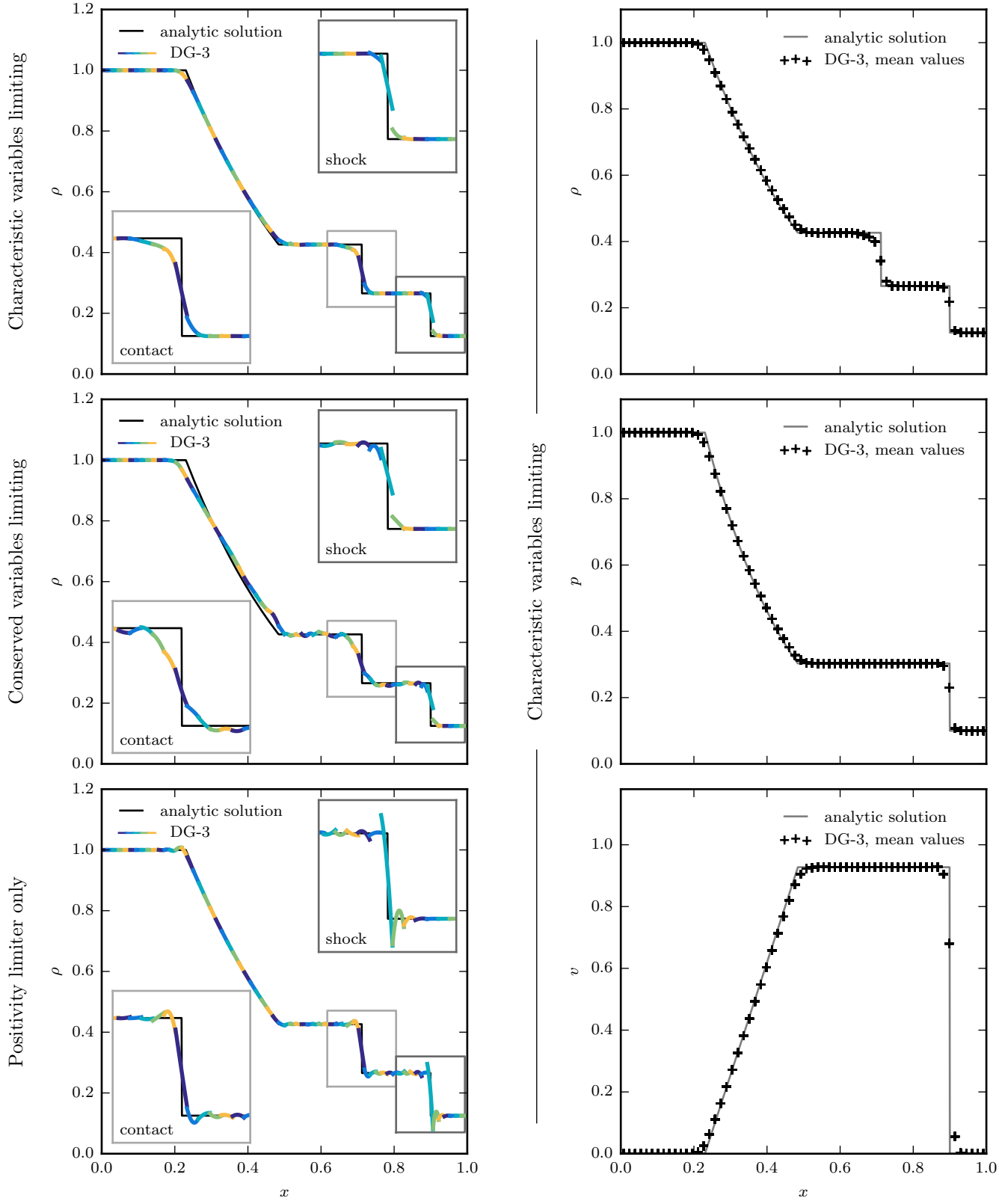

Figure 6. *Left panels:* Sod shock tube problem calculated with third order DG and different limiting approaches. We show the full polynomial DG solutions of the density, where the different colours correspond to different cells. The best result is achieved when the limiting is carried out based on the characteristic variables. As desired, the numerical solution is discontinuous and of low order at the shock. If the conserved variables are limited instead, the obtained solution is much less accurate, including numerical oscillations. The bottom left panel shows the result without a slope limiter. The solution is of third order in every cell (parabolas) leading to an under- and overshooting at the shock as well as spurious oscillations. *Right panels:* Comparison of the mean cell values with the analytic solution for the characteristic variables limiter. Advection errors wash out the solution at the contact rapidly, until it can be represented smoothly by polynomials. Overall, we find a very good general agreement with the analytic solution, especially at the position of the shock.

ing to equation (24) it is proportional to $1/(2p-1)$. Nevertheless, DG-4 is the most efficient method for this test. Consequently, for improving the calculation efficiency in smooth regions, the increase of the order of the scheme (p-refinement) should be preferred over the refinement of the underlying grid (h-refinement).

For FV methods this principle is cumbersome to achieve from a programming point of view, because for every order a different reconstruction scheme has to be implemented, and moreover, there are no well established standard approaches for implementing arbitrarily higher order FV methods. Higher order DG methods on the other hand can be implemented straightforwardly and in a unified way. If the implementation is kept general, changing the order of the scheme merely consists of changing a simple parameter. In principle, it is also possible to do the p-refinement on-the-fly. In this way, identified smooth regions can be integrated with large cells and high order, whereas regions close to shocks and other discontinuities can be resolved with many cells and a lower order integration scheme.

5.2 Shock tube

Due to the non-linearity of the Euler equations, the characteristic wave speeds of the solution depend on the solution itself. This dependency and the fact that the Euler equations do not diffuse momentum can lead to an inevitable wave steepening, ultimately producing wave breaking and the formation of mathematical discontinuities from initially smooth states (Toro 2009, and references herein). Such hydrodynamic shocks are omnipresent in many hydrodynamical simulations, particularly in astrophysics. The proper treatment of these shocks is hence a crucial component of any numerical scheme for obtaining accurate hydrodynamical solutions.

In a standard finite volume method based on the RSA approach, the solution is averaged within every cell at the end of each timestep. The solution is then represented through piece-wise constant states that can have arbitrarily large jumps at interfaces, consequently allowing shocks to be captured by construction. Similarly, the numerical solution in a DG scheme is discontinuous across cell interfaces, allowing for an accurate representation of hydrodynamic shocks. We demonstrate the shock-capturing abilities of our code with a classical Sod shock tube problem (Sod 1978) in the two-dimensional domain $(x, y) \in [0, 1]^2$ with 64^2 cells. The initial conditions consist of a constant state on the left, $\rho_\mathrm{l} = 1$, $p_\mathrm{l} = 1$, and a constant state on the right, $\rho_\mathrm{r} = 0.125$, $p_\mathrm{r} = 0.1$, separated by a discontinuity at $x = 0.5$. The velocity is initially zero everywhere, and the adiabatic index is chosen to be $\gamma = 7/5$.

In Figure 6 we show the numerical and analytic solution at $t_\mathrm{end} = 0.228$ for third order DG (DG-3), and for different limiting strategies. Several problems are apparent when a true discontinuity is approximated with higher order functions, i.e. the rapid convergence of the approximation at the jump is lost, the accuracy around the discontinuity is reduced, and spurious oscillations are introduced (Gibbs phenomenon, see for example Arfken & Weber 2013). In the bottom left panel of Fig. 6, we can clearly observe oscillations at both discontinuities, the contact and the shock. Here the result is obtained without any slope limiter, merely the positivity limiter slightly adjusts higher order terms such that negative density and pressure values in the calculation are avoided. Nevertheless, the obtained solution is accurate at large and has even the smallest L1 error of the three approaches tested.

The oscillations can however be reduced by limiting the second order terms of the solution, either expressed in the characteristic variables (upper left panel), or in the conserved variables (middle left panel). Strikingly, the numerical solution has a considerably higher quality when the characteristic variables are limited instead of the conserved ones, even though in both tests the same slope limiting parameters are used ($\beta = 1$, $\tilde{M} = 0.5$). The former gives an overall satisfying result with a discontinuous solution across the shock, as desired. The contact discontinuity is less sharp and smeared out over ~5 cells. This effect arises from advection errors inherent to grid codes and can hardly be avoided. Nevertheless, the DG scheme produces less smearing compared to a finite volume method once the solution is smooth, see also Section 5.4. Hence the higher order polynomials improve the sharpness of the contact. In the right panels of Fig. 6, we compare the mean density, pressure, and velocity values of the numerical solution calculated with the characteristic limiter with the analytic solutions, and find good agreement. Especially the shock is fitted very well by the mean values of the computed polynomials. To summarise, limiting of the characteristic variables is favourable over the limiting of the conserved variables. With the former our DG code produces good results in the shock tube test thanks to its higher order nature, which clearly is an advantage also in this discontinuous problem.

5.3 Sedov-Taylor blast wave

The previous test involved a relatively weak shock. We now confront our DG implementation with a strong spherical blast wave by injecting a large amount of thermal energy E into a point-like region of uniform and ultra cold gas of density ρ. The well-known analytic solution of this problem is self-similar and the radial position of the shock front as a function of time is given by $R(t) = R_0(Et^2/\rho)^{1/5}$ (see for example Padmanabhan 2000). The coefficient R_0 depends on the geometry (1D, 2D, or 3D) and the adiabatic index γ, it can be obtained by numerically integrating the total energy inside the shock sphere. Under the assumption of a negligible background pressure the shock has a formally infinite Mach number with the maximum density compression $\rho_\mathrm{max}/\rho = (\gamma+1)/(\gamma-1)$.

Numerically, we set up this test with 64^3 cells in a three-dimensional box $(x, y, z) \in [0, 1]^3$ containing uniform gas with $\rho = 1$, $p = 10^{-6}$, and $\gamma = 5/3$. The gas is initially at rest and the thermal Energy $E = 1$ is injected into the 8 central cells. Figure 7 shows numerical solutions obtained with FV as well as with second and third order DG at $t = 0.05$. The density slices in the top panels indicate very similar results, only the logarithmic colour-coding reveals small differences between the methods in this test. Unlike Lagrangian particle codes, Cartesian grid codes have preferred coordinate directions leading to asymmetries in the Sedov-Taylor blast wave problem, especially in the inner low density region. FV produces a characteristic cross in the center. In DG the asymmetries are a bit weaker, but traces of the initial geometry of energy injection are clearly visible as well. These effects can be minimised by distributing the injected energy across more cells, but they cannot be avoided in our grid codes for a point-like injection.

In the bottom panels of Fig. 7 we compare the numerical results to the analytic solution obtained with a code provided by Kamm & Timmes (2007), in particular we have $R_0 \approx 1.152$ for $\gamma = 5/3$ in 3D. Due to the finite and fixed resolution of our grid codes, the numerical solutions do not fully reach the analytic peak compression of $\rho_\mathrm{max} = 4$. More importantly, the DG method does not provide a visible improvement in this particular test, and furthermore, the results obtained with second order DG and third order DG are very similar. The reason behind these observations is the aggressive slope limiting due to the strong shock, suppressing the higher order polynomials in favour of avoiding over- and un-

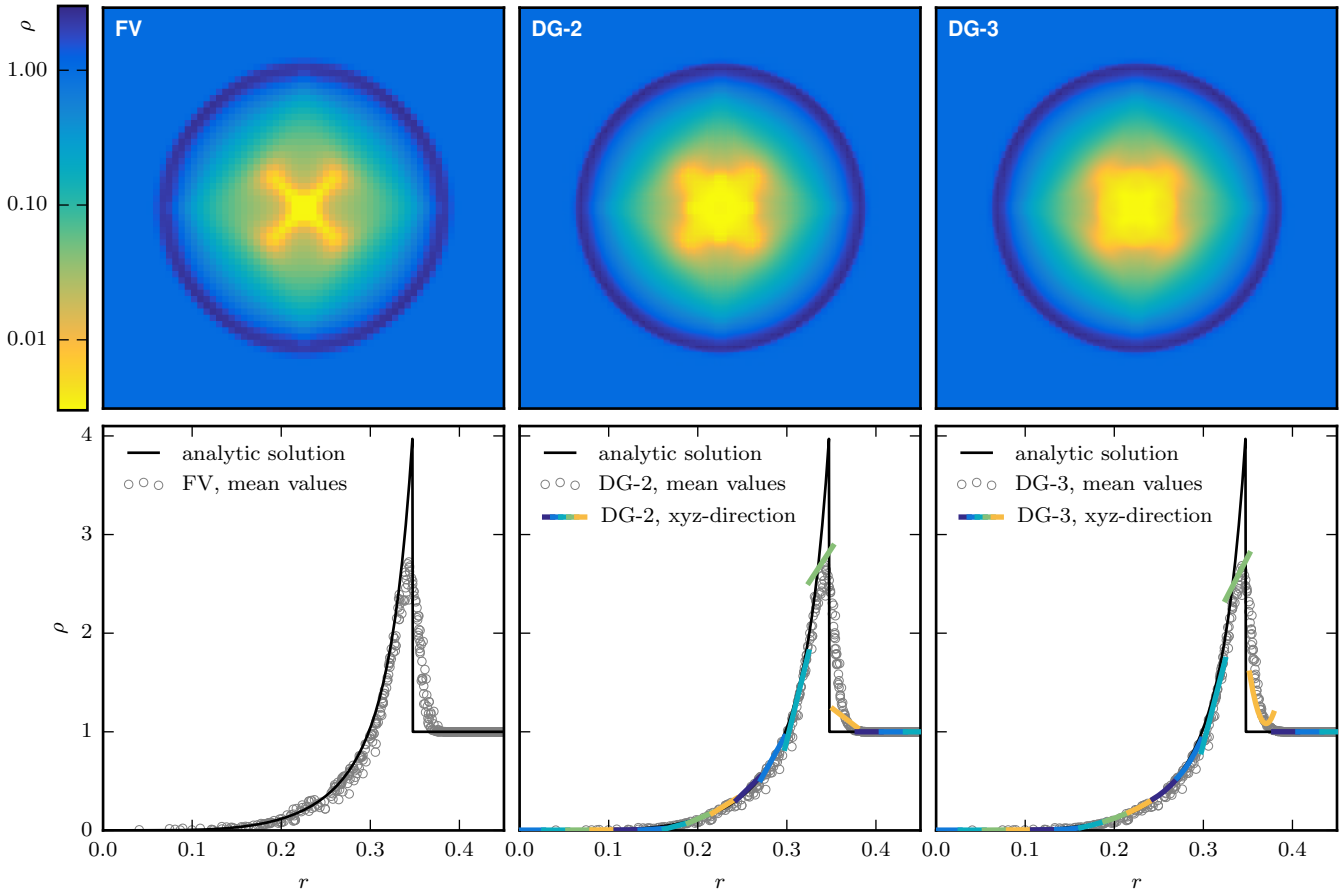

Figure 7. Three-dimensional Sedov-Taylor shock wave simulations at $t = 0.05$ calculated on a 64^3 cells grid with FV as well as with second and third order DG. The panels on top display central density slices ($z = 0$) on a logarithmic scale. In this test, Cartesian grid codes deviate noticeably from spherical symmetry in the central low density region, and the shape of this asymmetry depends on details of the different methods. In the bottom panels, we compare the numerical results with the analytic solution. For each method we plot the mean density of about every 200-th cell, and for DG also the solution polynomials in the diagonal direction along the coordinates from $(0.5, 0.5, 0.5)$ to $(1, 1, 1)$. The obtained results are very similar in all three methods considered here, and DG does not give a significant improvement over FV for this test.

dershootings of the solution. We note that a better result could of course be achieved by refining the grid at the density jump and thereby increasing the effective resolution. However, arguably the most important outcome of this test is that the DG scheme copes with an arbitrarily strong shock at least as well as a standard FV scheme, which is reassuring given that DG's primary strength lies in the representation of smooth parts of the solution.

5.4 Square advection

SPH, moving mesh and mesh-free approaches can be implemented such that the resulting numerical scheme is manifestly Galilean invariant, implying that the accuracy of the numerical solution does not degrade if a boost velocity is added to the initial conditions. On the other hand, numerical methods on stationary grids are in general not manifestly Galilean-invariant. Instead, they produce additional advection errors when a bulk velocity is added, which have the potential to significantly alter, e.g., the development of fluid instabilities (Springel 2010). While the numerical solution is still expected to converge towards the reference solution with increasing resolution in this case (Robertson et al. 2010), this comes at the price of a higher computational cost in the form of a (substantial) increase of the number of cells and an accompanying reduction of the timestep size.

We test the behaviour of the DG method when confronted with high advection velocities by simulating the supersonic motion of a square-shaped overdensity in hydrostatic equilibrium (following Hopkins 2014). For the initial conditions we choose a $\gamma = 7/5$ fluid with $\rho = 1$, $p = 2.5$, $v_x = 100$, and $v_y = 50$ everywhere, except for a squared region in the centre of the two-dimensional periodic box, $(x, y) \in [0, 1]^2$ with side lengths of 0.5, where the density is $\rho_s = 4$. The test is run with a resolution of 64^2 cells until $t = 10$, corresponding to 1000 transitions of the square in x-direction and 500 in the y-direction.

In Figure 8 we visually compare the results obtained with second order DG, third order DG, and the finite volume scheme. Already at $t = 1$, the finite volume method has distorted the square to a round and asymmetric shape, and at $t = 10$, the numerical solution is completely smeared out[2]. In comparison, DG shows fewer

[2] In our finite volume method the overdensity moves slightly faster in the direction of advection and is clearly not centered any more at $t = 10$. We have investigated this additional error and find that its occurrence depends on the choice of the slope limiter. Either way, the solution is completely

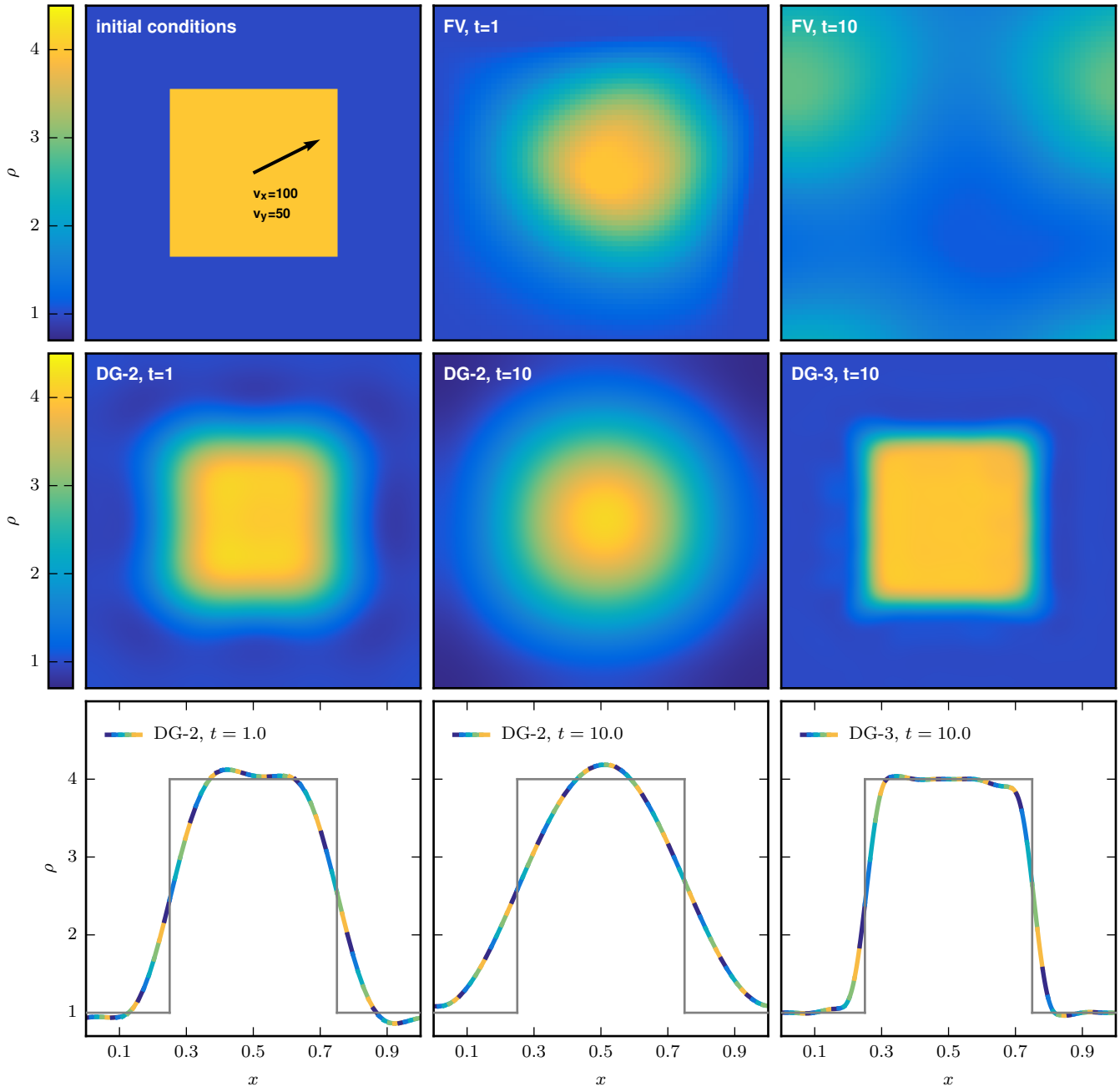

Figure 8. Density maps and centred slices for the advection test with 64^2 cells: a fluid with a square-shaped overdensity in hydrostatic equilibrium is advected supersonically, crossing the periodic box several hundred times. The second order finite volume method (FV) shows large advection errors in this test and the square is smeared out completely by $t = 10$. On the other hand, the advection errors in the DG method become small once the solution is smooth. Third order DG (DG-3) shows less diffusion than second order (DG-2) due to the higher order representation of the advected shape. The time evolution of the density errors for the three simulations is shown in Fig. 9.

advection errors and a better approximation of the initial shape can be sustained for a longer time. Especially the run with third order accuracy produces a satisfying result. Note that due to the high advection speed the CFL timestep is very small, leading to around 1.7 and 3.3 million timesteps at $t = 10$ with DG-2 and DG-3, respectively.

washed out and the finite volume method does not provide a satisfying result in this test.

In the bottom panels of Fig. 8 we show one-dimensional slices of the density solution polynomials from $(x, y) = (0, 0.5)$ to $(x, y) = (1, 0.5)$, and in grey the initial conditions as reference. With second order DG, an immediate over- and undershooting at the discontinuity can be observed owing to the adopted non-TVD slope limiter ($\beta = 1$, $\tilde{M} = 0.5$). More interestingly, while in the finite volume method the solution is only washed out by diffusion, for second order DG it is also steepened, which leads to a higher maximum density at $t = 10$ than present in the initial conditions. This

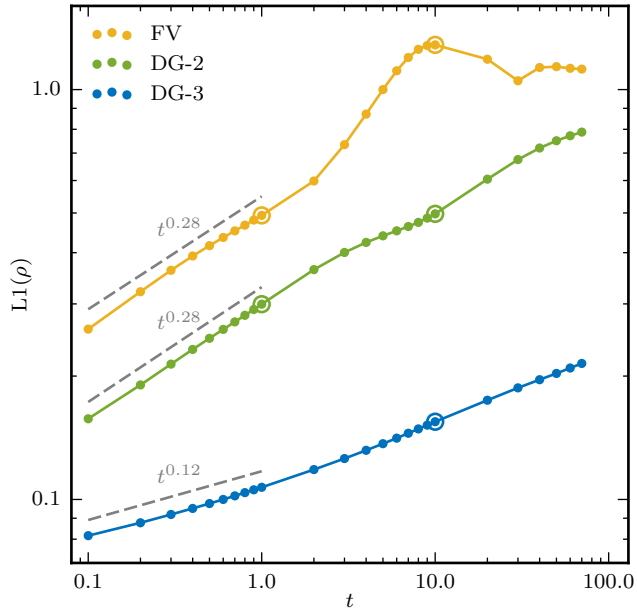

Figure 9. Time evolution of the L1 density error norm for the square advection test. The gray lines indicate the initial slopes of error growth, and the encircled data points correspond to the plots shown in Fig. 8. Interestingly, second order finite volume (FV) and second order DG (DG-2) have the same polynomial growth, though the absolute error is smaller by a factor of ≈ 1.5 for DG. On the other hand, the third order DG method (DG-3) does not only decrease the absolute error in this test, but the error growth rate is also significantly smaller.

difference arises from the updating of the slopes in DG instead of reconstructing them, and moreover, it vanishes if we use TVD limiter parameters. With the latter the slopes are reduced aggressively, resulting in pure diffusion with a result similar to the finite volume method.

We quantify the quality of each scheme in this test by measuring the time evolution of the L1 density error norm, the result of which is presented in Figure 9. The second order finite volume and second order DG codes show the same initial polynomial error growth, $\propto t^{0.28}$, however, the absolute error is much smaller for DG, i.e. the finite volume error at $t = 1$ is reached with DG-2 only at a much later time, at $t = 10$. With the third order DG code (quadratic basis functions), the error grows only as $\propto t^{0.12}$, leading to an acceptably small error even until $t = 70$, which corresponds to 23 million timesteps. Also with this higher order approximation the solution is transformed to a smooth numerical solution, but the mean cell values are less modified compared to the second order code with linear basis functions.

As demonstrated above, DG produces far smaller advection errors compared to a finite volume method. But how can this be understood, especially since the DG scheme is also not Galilean invariant? A powerful tool for studying the behaviour of discretisations in computational fluid dynamics is the modified equation analysis. Here, the discrete equations are expanded by means of a Taylor series, leading to a so-called modified equation which includes the target equation to solve and additional terms. For example, the modified equation of the first order upwind method for the scalar advection equation is an advection-diffusion equation (LeVeque 2002). The scheme solves the advection equation to first order by construction, but at the same time it effectively solves this advection-diffusion equation to second order accuracy, explaining the large diffusion errors when adopting this simple scheme, and pointing towards an explanation for the observed diffusion in our second order finite volume method. Such an analysis proves to be more difficult for DG, nevertheless Moura et al. (2014) recently accomplished a modified equation analysis for the second order DG scheme applied to the linear advection equation. In this case, the modified equation consists of a physical mode and an unphysical mode moving at the wrong speed, which is however damped very quickly. For upwinded fluxes the leading error term of the physical mode is diffusive and of third order, which is better than naively expected for a second order method, and this is likely one of the keys for the improved behavior of DG we find. A heuristic argument for the superiority of DG in this test is that after an initial smoothing of the global numerical solution, it is not only continuous inside every cell but also at the cell interfaces. If the solution is perfectly continuous across cell interfaces, the left and right state entering the Riemann solver are identical, and the calculated flux is always related by a simple Galilei boost to the flux calculated in any other frame of reference. In this case no manifest differences in the conserved quantities in a cell due to the flux calculation of the surface integral (20) can arise under a Galilei transformation, implying that advection errors must be minimal. Nevertheless, some small averaging errors will arise in practice if the current profile can not be represented exactly at an arbitrary grid position with the given set of cell basis functions.

5.5 Keplerian Disc

Rotating gas discs are omnipresent in our Universe, for example in galaxies, accretion discs around black holes, or protostellar and protoplanetary discs. Numerically, such objects are ideally modelled either with a Lagrangian method, or with a grid code which operates with a suitably tailored mesh geometry and furthermore accounts for part of the angular rotation in the solver (Masset 2000). If a simulation contains however several rotating objects at different locations and with different orientations, as for example is the case in cosmological galaxy formation simulations, no preferable grid geometry exists. In this case, Cartesian grids are often adopted, optionally with adaptive mesh refinement, making it much more challenging to avoid spurious errors in differentially rotating gas flows.

These problems can be exposed by the demanding problem of a cold Keplerian disc, where real pressure forces are negligibly small and any spurious hydrodynamic forces from numerical errors result in readily visible perturbations of the system. We subject our DG code to this test following a similar setup as recently used in other works (Hopkins 2014; Pakmor et al. 2015). The ambient gas of the disc has negligible density and pressure, and is initially confined to lie between two radii. Every fluid element of the disc is rotating on a Keplerian orbit where the centrifugal forces are balanced by an external and static central gravitational potential. Gas self-gravity is not included. Analytically, the system is in perfect equilibrium and the initial state should not change in time. The difficulty for hydro codes applied to this test lies in the lack of pressure support, as well as the differential rotation which leads to shearing flows. Both can trigger numerical instabilities and the eventual disruption of the disk. In particular, it is clear that for codes which do not conserve angular momentum exactly, this point will inevitably be reached eventually. In SPH codes, angular momentum is conserved but angular momentum transport is caused by the use of artificial viscosity, which typically is active at a small level in strong shear flows. Using an improved switch for the viscosity, however,

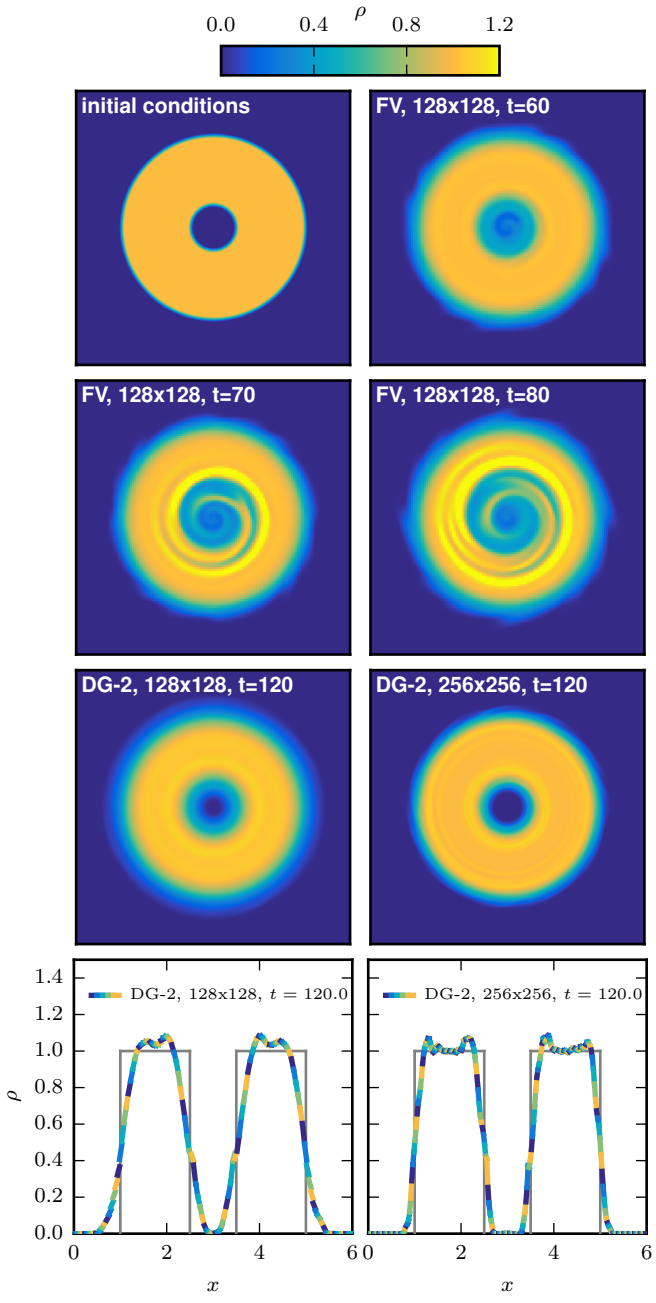

Figure 10. Density evolution in the cold Keplerian disc problem. The centrifugal force acting on the rotating disc is balanced by an external static central potential such that every fluid element is on a Keplerian orbit. This test is very challenging for Cartesian grid codes, as shear flows without pressure support are prone to numerical instabilities. Our finite volume method is stable for around 10 orbits, at which point the disc gets disrupted eventually. In contrast, without a slope limiter DG conserves angular momentum accurately and can handle this problem very well.

the Keplerian disc problem can be integrated accurately (Cullen & Dehnen 2010).

For definiteness, the initial conditions we use for our DG test are as follows. We use the computational domain $(x, y) \in [0, 6]^2$ with periodic boundaries, and gas with an adiabatic index of $\gamma = 5/3$. The primitive variables are initially set to

$$p = p_0,$$

$$\rho(r') = \begin{cases} \rho_0 & \text{if } r' < 0.5 - \frac{\Delta r}{2} \\ \frac{\rho_D - \rho_0}{\Delta r}(r' - (0.5 - \frac{\Delta r}{2})) + \rho_0 & \text{if } 0.5 - \frac{\Delta r}{2} \le r' < 0.5 + \frac{\Delta r}{2} \\ \rho_D & \text{if } 0.5 + \frac{\Delta r}{2} \le r' < 2 - \frac{\Delta r}{2} \\ \frac{\rho_0 - \rho_D}{\Delta r}(r' - (2 - \frac{\Delta r}{2})) + \rho_D & \text{if } 2 - \frac{\Delta r}{2} \le r' < 2 + \frac{\Delta r}{2} \\ \rho_0 & \text{if } r' \ge 2 + \frac{\Delta r}{2}, \end{cases}$$

$$v_x(x', y') = \begin{cases} -y'/r'^{3/2} & \text{if } 0.5 - 2\Delta r < r' < 2 + 2\Delta r \\ 0 & \text{else,} \end{cases}$$

$$v_y(x', y') = \begin{cases} x'/r'^{3/2} & \text{if } 0.5 - 2\Delta r < r' < 2 + 2\Delta r \\ 0 & \text{else,} \end{cases}$$

where the coordinates x', y', and r' are measured in a coordinate system with the origin at the centre $(x_0, y_0) = (3, 3)$ of the box. The values used for the background gas are $p_0 = 10^{-5}$ and $\rho_0 = 10^{-5}$. The disc has a density of $\rho_D = 1$ and a radial extent of $[r_{\rm min} = 0.5, r_{\rm max} = 2]$, with a small transition region of width $\Delta r = 0.1$. We adopt a time independent external acceleration $\boldsymbol{a} = -\boldsymbol{\nabla}\Phi$ with the components

$$a_x(x', y') = \begin{cases} -x'/r'^3 & \text{if } 0.5 - 0.5\Delta r < r' \\ -x'/[r'(r'^2 + \epsilon^2)] & \text{else,} \end{cases}$$

$$a_y(x', y') = \begin{cases} -y'/r'^3 & \text{if } 0.5 - 0.5\Delta r < r' \\ -y'/[r'(r'^2 + \epsilon^2)] & \text{else,} \end{cases}$$

where $\epsilon = 0.25$ smoothes the potential in the very inner regions in order to avoid a singularity there. The orbital period of the Keplerian disk depends on the radius and is given by $T = 2\pi r^{3/2}$.

We evolve the system until $t = 120$, corresponding to around 20 orbits at $r = 1$, and present the results in Figure 10. When the FV method is used, the edges of the disc are washed out immediately but the disc is stable for around 10 orbits. Like the majority of FV methods in use, our scheme does not manifestly preserve angular momentum, resulting in secular integration errors and an eventual breakdown of the quasi-static solution. The number of orbits the disc survives depends mainly on how carefully the problem has been set up, as well as on the choice of slope limiter and resolution used. However, the disruption of the disc is inevitable and can only be delayed with adjustments of these parameters.

On the other hand, DG schemes of second order and higher are inherently angular momentum conserving and can hence potentially handle this test problem much more accurately. To test for this, we have disabled the slope limiter of the DG scheme and use only the positivity limiter. This is because with our simple minmod limiter, angular momentum conservation is also violated and would result in a similar disruption as observed with FV. The construction of an improved angular momentum preserving limiting scheme is hence desirable and worthwhile subject for future work. With the positivity limiter alone, the second order DG scheme can integrate the disc stably and gives good results at $t = 120$, corresponding to about 20 orbits at $r = 1$. Merely two fine rings with a slightly higher density can be observed in the inner and outer regions of the disc, which can be attributed to the gentle overshooting of the solution at the rims of the disc. We have also carried out this simulation with the third order DG code. In this case, the disc also does not break down, but the solution shows some mild oscillations with amplitude up to 20 percent of the initial density; without applying a limiter these cannot be suppressed.

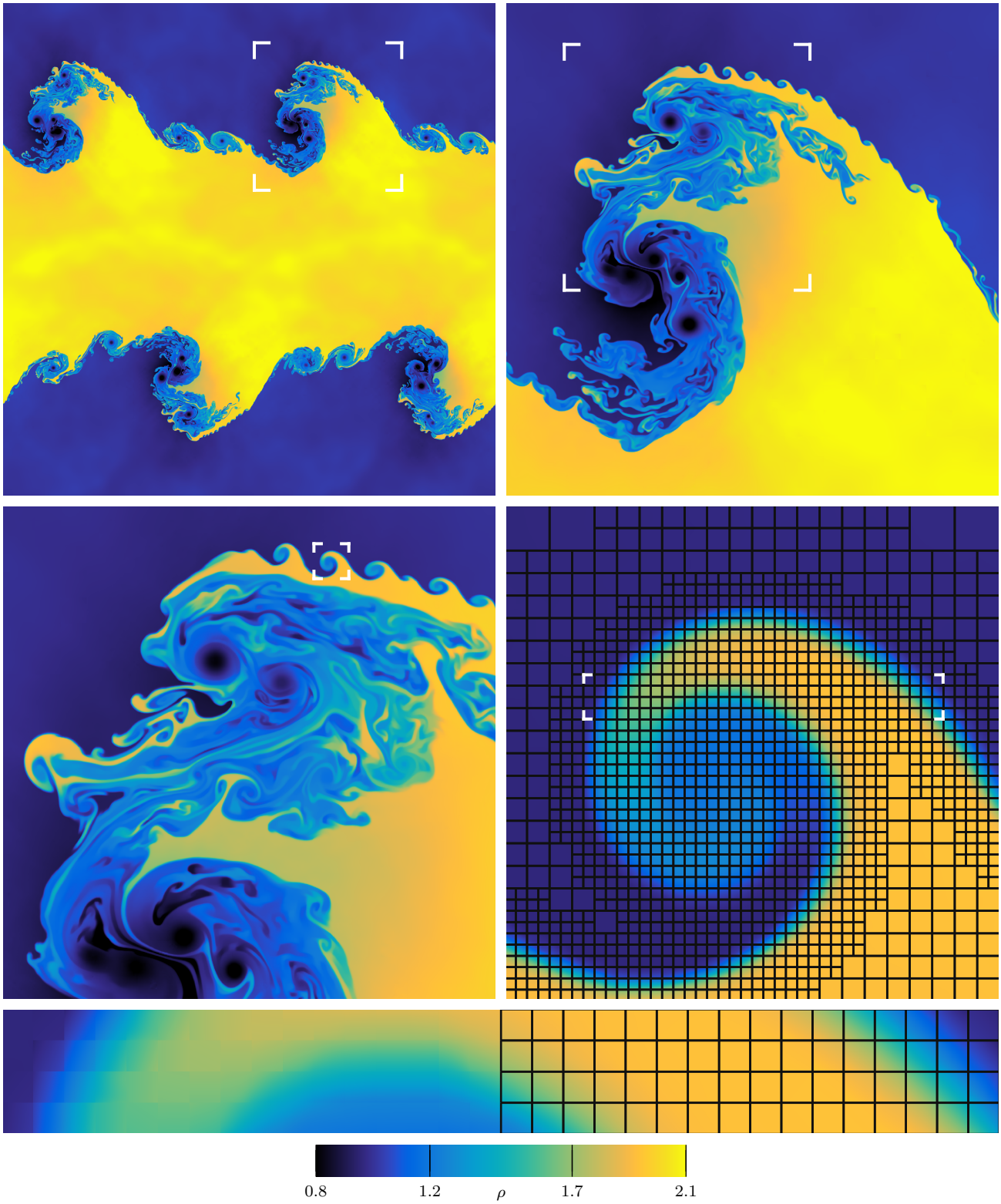

Figure 11. High-resolution Kelvin-Helmholtz simulation with fourth order DG and adaptive mesh refinement at time $t = 0.8$. The simulation starts with 64^2 cells (level 6) and refines down to level 12, corresponding to an effective resolution of 4096^2. We illustrate the AMR levels in Fig. 12. The mesh refinement approach renders it possible to resolve fractal structures created by secondary billows on top of the large-scale waves. Furthermore, as can be seen in the bottom panel, the solution within every cell contains rich information, consisting of a third order polynomial. A movie of the simulation until $t = 2$ may be accessed online: http://youtu.be/cTRQP6DSaqA

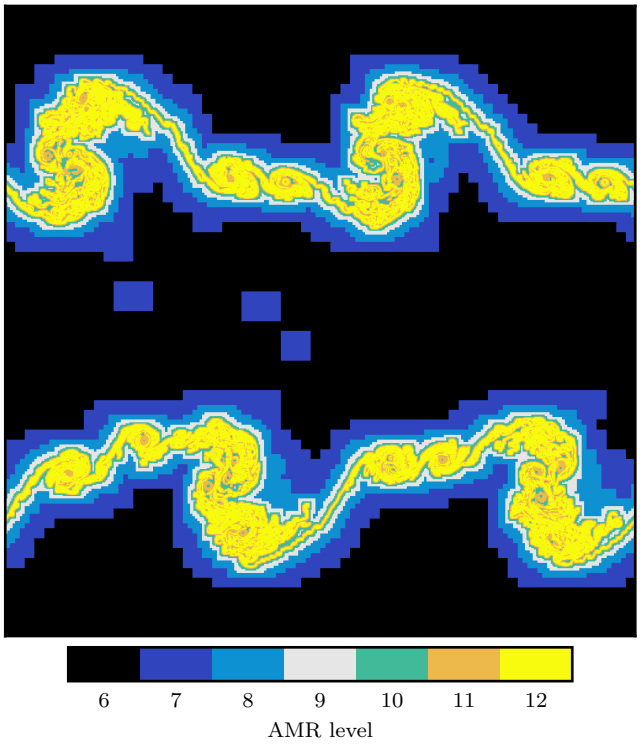

Figure 12. Map of the AMR levels of the Kelvin-Helmholtz simulation at $t = 0.8$. We here refine in space where the density gradient is steep, allowing us to capture interesting regions with high resolution and save computational time in places where the solution is smooth.

5.6 Kelvin-Helmholtz instability

The Kelvin-Helmholtz (KH) instability is one of the most important fluid instabilities in astrophysics, for example, it plays an important role in the stripping of gas from galaxies falling into a galaxy cluster. The instability is triggered by shear flows, often also involving fluids with different densities, and grows exponentially until the primary billows break, subsequently leading to a turbulent mixing of the two phases. The KH instability can be investigated with initial conditions consisting either of a sharp surface between two phases, or with a transition region separating the layers smoothly. Analytic growth rates for the linear regime can be derived for both cases (Chandrasekhar 1961; Hendrix & Keppens 2014), however the thickness of a smooth transition layer sets a limit on the minimum wave length which can become unstable in the linear phase. This suppression is desired if one wants to compare growth rates inferred from simulations with the analytic growth rate (McNally et al. 2012), since the underlying mesh can trigger the instability of waves at the resolution scale due to noise, a numerical effect which does not vanish with increasing resolution. Nevertheless, we set up a sharp discontinuity and use the Kelvin-Helmholtz instability as a high-resolution test for the robustness of our AMR implementation and DG's capabilities of capturing small-scale turbulent structures.

The initial conditions are chosen as in Springel (2010), in the periodic box $(x, y) \in [0, 1]^2$ the primitive variables at $t = 0$ are set to

$$p = 2.5,$$

$$\rho(x, y) = \begin{cases} 2 & \text{if } 0.25 < y < 0.75 \\ 1 & \text{else,} \end{cases}$$

$$v_x(x, y) = \begin{cases} 0.5 & \text{if } 0.25 < y < 0.75 \\ -0.5 & \text{else,} \end{cases}$$

$$v_y(x, y) = w_0 \sin(4\pi x) \left\{ \exp\left[-\frac{(y - 0.25)^2}{2\sigma^2} \right] + \exp\left[-\frac{(y - 0.75)^2}{2\sigma^2} \right] \right\},$$

with $w_0 = 0.1$, $\sigma = 0.05/\sqrt{2}$, and the adiabatic index $\gamma = 7/5$. The velocity perturbation in the $y-$direction supports the development of a specific single mode on large-scales. We start with an initial resolution of 64^2 cells (level 6) and refine where the density slope is steep, as described in 4.1. The maximum refinement level is set to 12, corresponding to an effective resolution of 4096^2 cells. A sharp discontinuity between the two layers in combination with AMR leads to a fast transition into the non-linear regime and generates secondary billows early on.

We illustrate the state of the simulation at $t = 0.8$ in Figure 11. The panel on the top left shows the density for the whole two-dimensional simulation box, in the following panels we zoom in repeatedly. The DG scheme shows only little diffusion and mixing thanks to the higher order and the avoidance of reconstruction steps, allowing the formation of very fine structures. Smaller Kelvin-Helmholtz instabilities arise on top of the large-scale waves demonstrating the fractal nature of this test problem. Self-similar instabilities are present across different scales, and an ending of this pattern is only set by the limited resolution.

The adaptive mesh used by the calculation is overlaid in the bottom right panel, revealing 3 different AMR levels in this subbox. The density gradients are well resolved with the finest AMR level (level 12), whereas smooth regions are on smaller levels. Furthermore, technical features of the AMR mesh structure can be seen in the plot: the level difference between neighbouring cells is at most one, and the mesh smoothing algorithm introduces an additional cell layer around physically refined cells. Figure 12 shows a map of the AMR levels of the whole simulation box at $t = 0.8$, which can be directly compared to the top left panel of Fig. 11. The number of cells at the displayed instance is around 1.8 million cells, which corresponds to about 10 percent of the effective level 12 resolution 4096^2, highlighting the efficiency gain possible with the AMR approach. Ideally, we would also like to use a corresponding adaptiveness in time by utilizing local timesteps, something that is planned as a code extension in future work.

6 SUMMARY

In this work, we have developed a 3D MPI-parallel higher order DG code with AMR for solving the Euler equations, called TENET, and investigated its performance in several astrophysically relevant hydrodynamic test problems. DG methods are comparatively new in astrophysics, despite a vast body of literature and long history of these approaches in the applied mathematics community. A number of highly attractive features of DG however suggest that it is timely to introduce these methods as standard tools in computational astrophysics. In particular, DG allows higher order to be reached without introducing communication beyond the immediate neighboring cells, making it particularly well suited for parallel computing. Also, the method is characterized by a higher compute to memory access ratio, making it a better match for modern computer architectures, where floating point operations are "almost free" in comparison to slow and costly memory access. In addition, DG allows an easy and flexible way to reach arbitrarily higher order, quite unlike FV schemes. This makes it possible to also vary the order of

a scheme in space, providing for additional flexibility in AMR approaches.

Our tests furthermore clearly show that it is computationally worthwhile to employ higher-order methods. While in general it depends on the problem which order is optimal, we have found in our tests that third and fourth order DG implementations are typically computationally much more efficient compared with a second order code, at least in regions where the solution is reasonably smooth. Moreover, the numerical result is of higher quality also in non-smooth regions, especially shocks are represented very well, thanks to the discontinuous nature of the DG representation. These discontinuities are captured within at most 2-3 cells, and spurious oscillations in the pre- and postshock regions can be prevented effectively by limiting the slopes and discarding higher order terms of the solution when appropriate.

The fundamental difference between DG and FV is that DG solves directly also for higher order moments of the solution, whereas in FV all higher order information is discarded in the implicit averaging at the end of each timestep, necessitating a subsequent reconstruction. This aspect of DG leads directly to two major advantages over traditional FV methods. Firstly, DG produces significantly less advection errors, and furthermore, if the solution is smooth across cell interfaces the numerical solution does not depend on the chosen Riemann solver. Secondly, DG inherently conserves not only mass, momentum and energy, but also angular momentum. The conservation of angular momentum is very desirable for many astrophysical applications, e.g. simulations involving discs, or objects like stars or molecular clouds in rotation. There is however one caveat which has to be kept in mind. The conservation can be spoiled by the limiting procedure, which is reminiscent of the situation in SPH, where angular momentum is spuriously transported by artificial viscosity. Improving the limiter with the goal of global angular momentum conservation is hence desirable and a promising direction for future improvements of the DG implementation. Finally, DG can also be comparatively easily generalised to the AMR technique, and importantly, this can be done without loss of accuracy, unlike in standard FV approaches. The higher order is formally preserved due to the usage of 'hanging nodes', which are the quadrature points of interfaces between cells of different sizes.

The present work has focussed on introducing our new code and highlighting its differences relative to FV schemes. Future developments will focus on developing more sophisticated limiting schemes (e.g., Sonntag & Munz 2014), scaling improvements through Open-MP hybrid parallelisation, as well as on incorporating magnetic fields and astrophysical source terms relevant in galaxy formation. The latter is greatly facilitated by the modular structure of TENET and its parent code AREPO. In a first science application of our DG code, we quantitatively analyse its capabilities of capturing driven turbulence structures (Bauer et al. 2015, in preparation).

ACKNOWLEDGEMENTS

We thank Gero Schnücke, Juan-Pablo Gallego, Johannes Löbbert, Federico Marinacci, Christoph Pfrommer, and Christian Arnold for very helpful discussions. The authors acknowledge financial support through subproject EXAMAG of the Priority Programme 1648 'SPPEXA' of the German Science Foundation, and through the European Research Council through ERC-StG grant EXAGAL-308037. KS and AB acknowledge support by the IMPRS for Astronomy and Cosmic Physics at the University of Heidelberg. PC was supported by the AIRBUS Group Corporate Foundation Chair in Mathematics of Complex Systems established in TIFR/ICTS, Bangalore.

REFERENCES

Abramowitz M., Stegun I., 2012, Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables. Dover Books on Mathematics, Dover Publications
Arfken G., Weber H., 2013, Mathematical Methods for Physicists. Elsevier Science
Bryan G. L., Norman M. L., O'Shea B. W., Abel T., Wise J. H., Turk M. J., Reynolds D. R., Collins D. C., 2014, ApJS, 211, 19
Chandrasekhar S., 1961, Hydrodynamic and Hydromagnetic Stability. Dover Books on Physics Series, Dover Publications
Cockburn B., Shu C.-W., 1989, Mathematics of Computation, 52, 411
Cockburn B., Shu C.-W., 1991, RAIRO-Modélisation mathématique et analyse numérique, 25, 337
Cockburn B., Shu C.-W., 1998, Journal of Computational Physics, 141, 199
Cockburn B., Lin S.-Y., Shu C.-W., 1989, Journal of Computational Physics, 84, 90
Cockburn B., Hou S., Shu C.-W., 1990, Mathematics of Computation, 54, 545
Cockburn B., Karniadakis G., Shu C., 2011, Discontinuous Galerkin Methods: Theory, Computation and Applications. Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg
Cullen L., Dehnen W., 2010, MNRAS, 408, 669
Duffell P. C., MacFadyen A. I., 2011, ApJS, 197, 15
Fryxell B., et al., 2000, ApJS, 131, 273
Gallego-Valencia J. P., Löbbert J., Müthing S., Bastian P., Klingenberg C., Xia Y., 2014, PAMM, 14, 953
Gottlieb S., Shu C.-W., 1998, Mathematics of computation of the American Mathematical Society, 67, 73
Gottlieb S., Shu C.-W., Tadmor E., 2001, SIAM review, 43, 89
Hendrix T., Keppens R., 2014, A&A, 562, A114
Hopkins P. F., 2014, ArXiv e-prints, 1409.7395,
Kamm J. R., Timmes F., 2007, Technical report, On efficient generation of numerically robust Sedov solutions. Technical Report LA-UR-07-2849, Los Alamos National Laboratory
Lanson N., Vila J.-P., 2008, SIAM Journal on Numerical Analysis, 46, 1912
LeVeque R., 2002, Finite Volume Methods for Hyperbolic Problems. Cambridge Texts in Applied Mathematics, Cambridge University Press
Lether F., Wenston P., 1995, Journal of Computational and Applied Mathematics, 59, 245
Masset F., 2000, A&AS, 141, 165
McNally C. P., Lyra W., Passy J.-C., 2012, ApJS, 201, 18
Mignone A., Bodo G., Massaglia S., Matsakos T., Tesileanu O., Zanni C., Ferrari A., 2007, ApJS, 170, 228
Mocz P., Vogelsberger M., Sijacki D., Pakmor R., Hernquist L., 2014, MNRAS, 437, 397
Moura R., Sherwin S., Peiró J., 2014, Technical report, Modified equation analysis for linear advection with DG. Research report, Imperial College London
Padmanabhan T., 2000, Theoretical Astrophysics: Volume 1, Astrophysical Processes. Theoretical Astrophysics, Cambridge University Press
Pakmor R., Springel V., Bauer A., Mocz P., Munoz D. J., Ohlmann S. T., Schaal K., Zhu C., 2015, preprint, (arXiv:1503.00562)
Reed W. H., Hill T., 1973, Los Alamos Report LA-UR-73-479
Robertson B. E., Kravtsov A. V., Gnedin N. Y., Abel T., Rudd D. H., 2010, MNRAS, 401, 2463
Sod G. A., 1978, Journal of Computational Physics, 27, 1
Sonntag M., Munz C.-D., 2014, in Fuhrmann J., Ohlberger M., Rohde C., eds, Springer Proceedings in Mathematics & Statistics, Vol. 78, Finite Volumes for Complex Applications VII-Elliptic, Parabolic and Hyperbolic Problems. Springer International Publishing, pp 945–953
Spiteri R. J., Ruuth S. J., 2002, SIAM J. Numer. Anal., 40, 469
Springel V., 2005, MNRAS, 364, 1105
Springel V., 2010, MNRAS, 401, 791

Stone J. M., Gardiner T. A., Teuben P., Hawley J. F., Simon J. B., 2008, ApJS, 178, 137
Teyssier R., 2002, A&A, 385, 337
Toro E., 2009, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction. Springer
Wadsley J. W., Stadel J., Quinn T., 2004, New Astron., 9, 137
Yee H. C., Sandham N. D., Djomehri M. J., 1999, Journal of Computational Physics, 150, 199
Zanotti O., Fambri F., Dumbser M., 2015, preprint, (arXiv:1504.07458)
Zhang X., 2006, Dissertation, U. of Science and Technology of China
Zhang X., Shu C.-W., 2010, Journal of Computational Physics, 229, 8918

APPENDIX A: LEGENDRE POLYNOMIALS

We use Legendre Polynomials as basis functions for our discontinuous Galerkin scheme. They can be calculated by solving Legendre's differential equation:

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\left[(1-\xi^2)\frac{\mathrm{d}}{\mathrm{d}\xi}P_n(\xi)\right] + n(n+1)P_n(\xi) = 0, \quad n \in \mathbb{N}_0. \tag{A1}$$

The first Legendre Polynomials are

$$\begin{aligned} P_0(\xi) &= 1 & P_1(\xi) &= \xi \\ P_2(\xi) &= \frac{1}{2}(3\xi^2-1) & P_3(\xi) &= \frac{1}{2}(5\xi^3-3\xi) \\ P_4(\xi) &= \frac{1}{8}(35\xi^4-30\xi^2+3) & P_5(\xi) &= \frac{1}{8}(63\xi^5-70\xi^3+15\xi). \end{aligned} \tag{A2}$$

They are pairwise orthogonal to each other, and moreover, we define the scaled polynomials

$$\tilde{P}(\xi)_n = \sqrt{2n+1}P(\xi)_n \tag{A3}$$

such that

$$\int_{-1}^{1}\tilde{P}_i(\xi)\tilde{P}_j(\xi)\,\mathrm{d}\xi = \begin{cases} 0 & \text{if } i \neq j \\ 2 & \text{if } i = j. \end{cases} \tag{A4}$$

APPENDIX B: GAUSS-LEGENDRE QUADRATURE

The numerical integration of a function $f : [-1,+1] \to \mathbb{R}$ with the Gaussian quadrature rule of n points is given by a weighted sum,

$$\int_{-1}^{+1} f(\xi)\,\mathrm{d}\xi \approx \sum_{q=1}^{n} f(\xi_q^{\mathrm{1D}})\omega_q^{\mathrm{1D}}. \tag{B1}$$

Here, $\xi_q^{\mathrm{1D}} \in (-1,+1)$ are the Gaussian quadrature nodes and ω_q^{1D} are the corresponding weights. To integrate a 2D function $f : [-1,+1]^2 \to \mathbb{R}$ the tensor product of the n Gauss points can be used, viz.

$$\begin{aligned} &\int_{-1}^{+1}\int_{-1}^{+1} f(\xi_1,\xi_2)\,\mathrm{d}\xi_1\mathrm{d}\xi_2 \\ &\approx \sum_{q=1}^{n}\sum_{r=1}^{n} f(\xi_{1,q}^{\mathrm{1D}},\xi_{2,r}^{\mathrm{1D}})\omega_q^{\mathrm{1D}}\omega_r^{\mathrm{1D}} = \sum_{q=1}^{n^2} f(\boldsymbol{\xi}_q^{\mathrm{2D}})\omega_q^{\mathrm{2D}}. \end{aligned} \tag{B2}$$

The n-point Gaussian quadrature rule is exact for polynomials of degree up to $2n-1$, and the one-dimensional nodes are given as the roots of the Legendre polynomial $P_n(\xi)$. We calculate them numerically by means of the Newton-Raphson method. As starting values of the iterative root finding approximate expressions for the roots can be used (see for example Lether & Wenston 1995),

$$\xi_q \approx \left(1 - \frac{1}{8n^2} + \frac{1}{8n^3}\right)\cos\left(\pi\frac{4q-1}{4n+2}\right), \quad q = 1,\ldots,n. \tag{B3}$$

$$\begin{array}{llllll} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

Table D1. Runge-Kutta butcher tableau.

$$\begin{array}{lll} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Table D2. SSP-RK2.

$$\begin{array}{llll} 0 & & & \\ 1 & 1 & & \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array}$$

Table D3. SSP-RK3.

Furthermore, the corresponding weights can be calculated as (Abramowitz & Stegun 2012)

$$\omega_q = \frac{2}{(1-\xi_q^2)P_n'(\xi_q)^2}, \quad q = 1,\ldots,n. \tag{B4}$$

With this approach, we can compute and store the necessary quadrature data in the initialisation routine of our DG code for arbitrary spatial order.

APPENDIX C: GAUSS-LEGENDRE-LOBATTO QUADRATURE

Compared with Gaussian quadrature, the GLL quadrature rule is very similar but includes also the endpoints of the integration interval. Therefore, the n-point GLL rule is exact for polynomials of degree $2n-3$. The nodes are the roots $\hat{\xi}_q$ of the function $(1-\xi^2)P_{n-1}'(\xi)$, and the corresponding weights are given by (Abramowitz & Stegun 2012)

$$\hat{\omega}_q = \frac{2}{n(n-1)P_{n-1}(\hat{\xi}_q)^2}, \quad q = 2,\ldots,n-1. \tag{C1}$$

The weights of the endpoints are equal, $\hat{\omega}_1 = \hat{\omega}_n$, and the sum of the weights is $\sum_{q=1}^{n}\hat{\omega}_q = 2$.

APPENDIX D: STRONG STABILITY PRESERVING RUNGE-KUTTA METHODS

Let $w(t)$ be the unknown scalar solution of the ordinary differential equation

$$\frac{\mathrm{d}w}{\mathrm{d}t} + R(t,w) = 0. \tag{D1}$$

The propagation of the numerical solution from timestep n to $n+1$ with an $s-$stage explicit RK method can be written as

$$w^{n+1} = w^n + \Delta t^n \sum_{i=1}^{s} b_i k_i. \tag{D2}$$

The factors b_i weight the sum over the solution derivatives k_i, which are evaluations of $R(t,w)$, viz.

$$k_i = -R(t^n + c_i\Delta t^n, w^n + \Delta t^n(a_{i1}k_1 + a_{i2}k_2 + \ldots + a_{i,i-1}k_{i-1})), \tag{D3}$$

where the c_i are nodes of the timestep interval. A RK scheme is fully specified by the weights b_i, the nodes c_i, and the RK matrix a_{ij}, it can be represented in compact form by means of a Butcher

0					
0.39175222700392	0.39175222700392				
0.58607968896779	0.21766909633821	0.36841059262959			
0.47454236302687	0.08269208670950	0.13995850206999	0.25189177424738		
0.93501063100924	0.06796628370320	0.11503469844438	0.20703489864929	0.54497475021237	
	0.14681187618661	0.24848290924556	0.10425883036650	0.27443890091960	0.22600748319395

Table D4. SSP-RK4.

tableau (Table D1). For our DG code we use strong stability preserving RK methods, which are convex combinations of forward Euler steps. In combination with a positivity preserving Riemann solver and the positivity limiter outlined in Section 3.4 negative pressure and density values can effectively be avoided in the hydro scheme. For the second order DG code we use SSP RK2 Heun's method (D2), for our third order code the SSP RK3 from Table D3. Theses methods are optimal in the sense that the number of stages is equal to the order of the scheme. It can be proven that a fourth order method with this feature does not exist (Gottlieb & Shu 1998), we therefore adopt the 5-stage SSP RK 4 method tabulated in Table D4 (Spiteri & Ruuth 2002).

APPENDIX E: EIGENVECTORS OF THE EULER EQUATIONS

The Eigenvalues of the flux Jacobian Matrix $\mathbf{A}_1 = \frac{\partial f_1(\boldsymbol{u})}{\partial \boldsymbol{u}}$ are $\lambda_i = \{v_1 - c, v_1, v_1 + c, v_1, v_1\}$. The corresponding Eigenvectors are the columns of the matrix

$$\mathcal{R}_x = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ v_1 - c & v_1 & v_1 + c & 0 & 0 \\ v_2 & v_2 & v_2 & -1 & 0 \\ v_3 & v_3 & v_3 & 0 & 1 \\ h - cv_1 & k & h + cv_1 & -v_2 & v_3 \end{pmatrix}, \tag{E1}$$

with the specific kinetic energy $k = \frac{1}{2}(v_1^2 + v_2^2 + v_3^3)$ and the specific stagnation enthalpy $h = c^2/(\gamma - 1) + k$. The left eigenvectors of $\mathbf{A}_1$ are the rows of the Matrix $\mathcal{L}_x = \mathcal{R}_x^{-1}$. $\mathcal{L}_x$ is the linear transformation operator from the conserved to the characteristic variables, $\boldsymbol{c} = \mathcal{L}_x \boldsymbol{u}$, and can be calculated as

$$\mathcal{L}_x = \begin{pmatrix} \beta(\phi + cv_1) & -\beta(\gamma_1 v_1 + c) & -\beta\gamma_1 v_2 & -\beta\gamma_1 v_3 & \beta\gamma_1 \\ 1 - 2\beta\phi & 2\beta\gamma_1 v_1 & 2\beta\gamma_1 v_2 & 2\beta\gamma_1 v_3 & -2\beta\gamma_1 \\ \beta(\phi - cv_1) & -\beta(\gamma_1 v_1 - c) & -\beta\gamma_1 v_2 & -\beta\gamma_1 v_3 & \beta\gamma_1 \\ v_2 & 0 & -1 & 0 & 0 \\ -v_3 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{E2}$$

where the definitions $\gamma_1 = \gamma - 1$, $\phi = \gamma_1 k$, and $\beta = 1/(2c^2)$ have been used. The eigenvector matrices for the flux Jacobians $\frac{\partial f_2(\boldsymbol{u})}{\partial \boldsymbol{u}}$ and $\frac{\partial f_3(\boldsymbol{u})}{\partial \boldsymbol{u}}$ are

$$\mathcal{R}_y = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ v_1 & v_1 & v_1 & 1 & 0 \\ v_2 - c & v_2 & v_2 + c & 0 & 0 \\ v_3 & v_3 & v_3 & 0 & -1 \\ h - cv_2 & k & h + cv_2 & v_1 & -v_3 \end{pmatrix}, \tag{E3}$$

$$\mathcal{L}_y = \begin{pmatrix} \beta(\phi + cv_2) & -\beta\gamma_1 v_1 & -\beta(\gamma_1 v_2 + c) & -\beta\gamma_1 v_3 & \beta\gamma_1 \\ 1 - 2\beta\phi & 2\beta\gamma_1 v_1 & 2\beta\gamma_1 v_2 & 2\beta\gamma_1 v_3 & -2\beta\gamma_1 \\ \beta(\phi - cv_2) & -\beta\gamma_1 v_1 & -\beta(\gamma_1 v_2 - c) & -\beta\gamma_1 v_3 & \beta\gamma_1 \\ -v_1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & 0 & -1 & 0 \end{pmatrix}, \tag{E4}$$

and

$$\mathcal{R}_z = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ v_1 & v_1 & v_1 & -1 & 0 \\ v_2 & v_2 & v_2 & 0 & 1 \\ v_3 - c & v_3 & v_3 + c & 0 & 0 \\ h - cv_3 & k & h + cv_3 & -v_1 & v_2 \end{pmatrix}, \tag{E5}$$

$$\mathcal{L}_z = \begin{pmatrix} \beta(\phi + cv_3) & -\beta\gamma_1 v_1 & -\beta\gamma_1 v_2 & -\beta(\gamma_1 v_3 + c) & \beta\gamma_1 \\ 1 - 2\beta\phi & 2\beta\gamma_1 v_1 & 2\beta\gamma_1 v_2 & 2\beta\gamma_1 v_3 & -2\beta\gamma_1 \\ \beta(\phi - cv_3) & -\beta\gamma_1 v_1 & -\beta\gamma_1 v_2 & -\beta(\gamma_1 v_3 - c) & \beta\gamma_1 \\ v_1 & -1 & 0 & 0 & 0 \\ -v_2 & 0 & 1 & 0 & 0 \end{pmatrix}, \tag{E6}$$

respectively.

APPENDIX F: REFINEMENT MATRICES

Below, we list the refinement matrices for merging the cells $A, B, \ldots, H$ into a coarser cell $K = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [-1, 1]^3\}$. We calculate the integrals by means of an exact numerical integration with $(k + 1)^3$ quadrature points before the main loop of our code.

Subcell $A = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [-1, 0] \times [-1, 0] \times [-1, 0]\}$:

$$(\mathsf{P}_A)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 - 1}{2}, \frac{\xi_2 - 1}{2}, \frac{\xi_3 - 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}. \tag{F1}$$

Subcell $B = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [0, 1] \times [-1, 0] \times [-1, 0]\}$:

$$(\mathsf{P}_B)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 + 1}{2}, \frac{\xi_2 - 1}{2}, \frac{\xi_3 - 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}. \tag{F2}$$

Subcell $C = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [-1, 0] \times [0, 1] \times [-1, 0]\}$:

$$(\mathsf{P}_C)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 - 1}{2}, \frac{\xi_2 + 1}{2}, \frac{\xi_3 - 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}. \tag{F3}$$

Subcell $D = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [0, 1] \times [0, 1] \times [-1, 0]\}$:

$$(\mathsf{P}_D)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 + 1}{2}, \frac{\xi_2 + 1}{2}, \frac{\xi_3 - 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}. \tag{F4}$$

Subcell $E = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [-1, 0] \times [-1, 0] \times [0, 1]\}$:

$$(\mathsf{P}_E)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 - 1}{2}, \frac{\xi_2 - 1}{2}, \frac{\xi_3 + 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}. \tag{F5}$$

Subcell $F = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [0, 1] \times [-1, 0] \times [0, 1]\}$:

$$(\mathsf{P}_F)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 + 1}{2}, \frac{\xi_2 - 1}{2}, \frac{\xi_3 + 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}. \tag{F6}$$

Subcell $G = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [-1, 0] \times [0, 1] \times [0, 1]\}$:

$$(\mathsf{P}_G)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 - 1}{2}, \frac{\xi_2 + 1}{2}, \frac{\xi_3 + 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}. \quad \text{(F7)}$$

Subcell $H = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in [0, 1] \times [0, 1] \times [0, 1]\}$:

$$(\mathsf{P}_H)_{l,j} = \frac{1}{8} \iiint_{[-1,1]^3} \phi_l \left(\frac{\xi_1 + 1}{2}, \frac{\xi_2 + 1}{2}, \frac{\xi_3 + 1}{2} \right) \phi_j(\xi_1, \xi_2, \xi_3) \, \mathrm{d}\boldsymbol{\xi}. \quad \text{(F8)}$$

This paper has been typeset from a TEX/LATEX file prepared by the author.