Commun. Comput. Phys.
doi: 10.4208/cicp.2009.09.033

Vol. **x**, No. x, pp. 1-33
xxx 200x

Balanced Monitoring of Flow Phenomena in Moving Mesh Methods

A. van Dam* and P. A. Zegeling

Department of Mathematics, Utrecht University, P.O. Box 80.010, 3508 TA Utrecht, The Netherlands.

Received xxx 200x; Accepted (in revised version) xxx 200x

Available online xxx

Abstract. Adaptive moving mesh research usually focuses either on analytical derivations for prescribed solutions or on pragmatic solvers with challenging physical applications. In the latter case, the monitor functions that steer mesh adaptation are often defined in an ad-hoc way. In this paper we generalize our previously used monitor function to a balanced sum of any number of monitor components. This avoids the trial-and-error parameter fine-tuning that is often used in monitor functions. The key reason for the new balancing method is that the ratio between the maximum and average value of a monitor component should ideally be equal for all components. Vorticity as a monitor component is a good motivating example for this. Entropy also turns out to be a very informative monitor component. We incorporate the monitor function in an adaptive moving mesh higher-order finite volume solver with HLLC fluxes, which is suitable for nonlinear hyperbolic systems of conservation laws. When applied to compressible gas flow it produces very sharp results for shocks and other discontinuities. Moreover, it captures small instabilities (Richtmyer-Meshkov, Kelvin-Helmholtz). Thus showing the rich nature of the example problems and the effectiveness of the new monitor balancing.

AMS subject classifications: 35L65, 65M50, 74S10, 76L05, 76N15

Key words: Moving mesh method, conservative interpolation, balanced monitor function, directional adaptation, hydrodynamics, implosion problem.

1 Introduction

Adaptive mesh methods improve local resolution of numerical solvers and, as a result, improve their performance. Results are significantly sharper than those obtained by using a uniform mesh with more mesh points. True gain in performance is only obtained,

*Corresponding author. *Email addresses:* A.vanDam@uu.nl (A. van Dam), P.A.Zegeling@uu.nl (P. A. Zegeling)

though, when the adaptive methods perform well automatically. This requires a balanced monitoring of flow phenomena, without manual fine-tuning of parameters by trial and error. This paper presents such a balanced monitoring, combined with a powerful finite volume solver, applied to hydrodynamical problems.

Adaptivity Three types of adaptive methods are generally distinguished: *h*-, *r*- and *p*-refinement. The *h*-refinement or *local refinement* splits mesh cells into smaller ones based on some criterion. This can provide great levels of detail and is widely used in CFD-codes. The implementation is nontrivial due to the hierarchical structure of the domain discretisation. The eventual number of mesh cells is sometimes hard to predict, which may lead to unexpectedly long running times. Although the initial structure of the mesh fixes the shape and orientation of the mesh cells, e.g., rectangular, the unlimited amount of possible refinement makes these methods very powerful. In one of our experiments (Section 5) we will make a comparison between our *r*-refinement results and the *h*-refinement results produced by AMRVAC [23,36].

The *r*-refinement or *(adaptive) moving mesh refinement* moves mesh points towards regions that need refinement based on some criterion. The number of points remains constant, which gives fairly predictable running times. Besides, the mesh cells can change shape, position and orientation, so that alignment with, e.g., shocks or vortices is well possible. For specific problems, the fixed number of mesh points may impose a limit on the achievable resolution. This paper deals with *r*-refinement only and shows that it can achieve great levels of detail. Tang [32] provides an extensive historical overview of moving mesh methods and their applications in CFD. Zegeling [43] presents Winslow-type adaptivity applied to a wide range of problems. We also give a detailed and systematic overview [35] on Winslow-type adaptivity, harmonic maps, geometric conservation laws and more related methods.

The combination of the above two methods, called *hr*-refinement combines the advantages of both methods and is occasionally used, e.g., by Lang et al. [20] and Anderson et al. [1].

The *p*- (and *hp*-) refinement is generally applied in different frameworks than what we consider here. It involves the local increase of polynomial order of the basis functions in finite element methods.

Moving mesh research We employ a variational formulation of mesh adaptation, an approach which has become well-known over the past five decades. A short historic overview of moving mesh methods is given in Section 3.3.1. Tang and Tang [30] presented a moving mesh algorithm in a pragmatic combination with a finite volume solver. Over the past five years, this inspired several others. The technique is usually applied to hydrodynamics (HD), e.g., by Tang [31] and Zegeling et al [44], and to magnetohydrodynamics (MHD), e.g., by Han and Tang [13], Tan [27], Van Dam and Zegeling [39] and Zegeling [42]. Moving mesh methods generally have little dependency on the physical PDEs under consideration, as diverse applications show, e.g., the Navier-Stokes equations by Di et al. [11] and the Hamilton-Jacobi equations by Tang et al. [29]. A similar

method, but now using direct minimization of the mesh functional has been used for reactive flows in 2D by Azarenok and Tang [2] and multi-phase fluids in 3D by Di et al. [10].

Monitor functions Based on earlier work by Beckett and Mackenzie [3] and Huang [17], we formulated an adaptive monitor function that makes manual fine-tuning unnecessary [39]. Here we improve this function in two ways: a slightly changed normalization balances all solution components equally, and we propose additional monitor components that detect phenomena that would otherwise be largely overseen.

Huang has done extensive research on analytical properties of monitor functions and the resulting mesh adaptation. Most of that work deals with prescribed solutions, so no physical PDE part is involved in the algorithm. This gives a better opportunity for analytical discussions, which Huang recently summarized in an overview paper [16].

Brackbill has done similar research on the combination of several functionals in the minimization process (see Section 3.3.1). Besides combining mesh quality functionals [6, with Saltzman] he also combined an alignment functional with a solution adaptivity functional in order to obtain directional control [5]. Directional monitor functions have been widely used ever since, for example by Glasser et al. [12] in a more analytical context. Tang [31] applies a directional monitor function to the two-dimensional Euler equations and Tan [27] uses an identical monitor for a two-dimensional resistive MHD model. We will also use such a directional monitor function as it produces much higher quality meshes at negligible costs.

Structure This paper is organized as follows. Section 2 briefly recalls the physics behind compressible gas flow and mentions some relevant flow quantities. Section 3 provides a detailed description of our moving mesh finite volume solver. The first part concerns the finite volume solver with HLLC fluxes, nonuniform solution reconstruction and slope limiting. The second part concerns the mesh movement, its history and our current algorithm. Section 4 presents the main contribution of this paper: a *balanced monitor function* to capture various flow phenomena. Section 5 contains three example problems that were already partly used in the preceding sections and are then further explored. Section 6 summarizes our findings and gives some recommendations for further research.

2 Phenomena in compressible gas flow

The first system of PDEs that comes to mind when testing moving mesh methods on flow problems are the equations of compressible gas dynamics. Forming a nonlinear system of hyperbolic PDEs, they can result in several wave types, possibly interacting, without requiring additional conditions from the numerical solver, such as the divergence-free magnetic field condition in ideal MHD simulations would do.

In the following sections the physical model and several physical features in compressible gas flow are described, where some expressions are already specialized into their two-dimensional form.

2.1 Physical model

The time evolution of a compressible gas is described by the Euler equations:

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ E\end{bmatrix}+\nabla\cdot\begin{bmatrix}\rho\mathbf{v}\\ \rho\mathbf{v}\mathbf{v}+p\mathbf{I}\\ (E+p)\mathbf{v}\end{bmatrix}=\begin{bmatrix}0\\ \mathbf{0}\\ 0\end{bmatrix}, \tag{2.1}$$

where the two-dimensional advection is denoted as $\mathbf{v}:=[u,v]^T$. The divergence of the flux tensor results in, e.g.,

$$\nabla\cdot\rho\mathbf{v}\mathbf{v}\equiv\frac{\partial}{\partial x}\rho u\mathbf{v}+\frac{\partial}{\partial y}\rho v\mathbf{v}.$$

The system (2.1) is closed by the standard equation of state:

$$E=\frac{p}{\gamma-1}+\frac{1}{2}\rho\mathbf{v}\cdot\mathbf{v},$$

where γ is the adiabatic constant, specifying the ratio of specific heats.

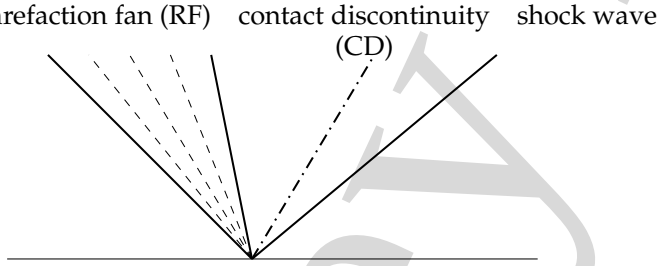

Figure 1: Elementary wave structure of the hyperbolic Euler equations for one-dimensional compressible gas flow. The three wave types form the building blocks of wave interactions in two dimensions.

2.2 Relevant flow features

Analysis of shock waves and other features in compressible gas flow is easiest in a one-dimensional setting. For completeness, Fig. 1 shows the elementary wave structure that results from an initially discontinuous solution. Remember that not all solution quantities change value across all waves. For example, the thermal pressure

$$p=(\gamma-1)(E-\frac{1}{2}\rho\mathbf{v}\cdot\mathbf{v}) \tag{2.2}$$

is constant across the contact discontinuity (CD). Similarly, the entropy, defined as

$$S=\log\left(\frac{p}{\rho^\gamma}\right), \tag{2.3}$$

is constant across the rarefaction fan (RF). It is still an interesting quantity, though, as rapid changes in entropy indicate a high potential for the emergence of new flow features. These observations will be relevant in selecting flow quantities upon which mesh adaptivity is based.

The three elementary wave types above will also appear in problems on two-dimensional domains. They are essentially the same, except that they can appear in any direction. Moreover, they are likely to meet and interact over time. Schulz-Rinne et al. [25] give a classification of fifteen different solutions for two-dimensional Riemann problems, which was later corrected by Lax and Liu [21] to nineteen different solutions. All make the same assumption that each initial discontinuity produces only one type of elementary wave. More freedom in the initial solutions would greatly increase the number of possible outcomes.

Possible new flow features include Mach reflections between shock lines or at rigid walls or CDs bending into spirals. In Section 5 we will use one of these example problems to test our adaptive method on. Amongst others, we will investigate whether entropy gradients or local vorticity $\|\nabla\times\mathbf{v}\|$ are good detectors of more subtle flow features.

3 An adaptive moving mesh solver for conservation laws

The conservation law PDEs are solved by an explicit time integration using finite volumes, combined with time dependent mesh movement to capture evolving flow features. The finite volume solver is discussed in Section 3.2 and the mesh movement in Section 3.3.

3.1 Physical problem description

We use a solver that is suitable for nonlinear systems of hyperbolic PDEs in general:

$$\frac{\partial}{\partial t}\mathbf{q}+\frac{\partial}{\partial x}\mathbf{f}(\mathbf{q})+\frac{\partial}{\partial y}\mathbf{g}(\mathbf{q})=\mathbf{0},\quad \mathbf{q}([x,y],t)\in\mathbb{R}^M. \tag{3.1}$$

The Euler equations for compressible gas dynamics (2.1) are in the above form and will be the leading example in this paper. Basic meteorological models as well as the advection model fit in the same form. We have readily extended the solver to two-dimensional ideal magnetohydrodynamics too.

The domain Ω is defined and discretised as follows:

$$\Omega:=[x_{\min},x_{\max}]\times[y_{\min},y_{\max}]=\bigcup_{j=0,\cdots,N_x-1,\ k=0,\cdots,N_y-1} A_{j+\frac{1}{2},k+\frac{1}{2}}, \tag{3.2}$$

$$A_{j+\frac{1}{2},k+\frac{1}{2}}:=\text{quadrilateral cell with corners } \mathbf{x}_{j+c,k+d},\quad c,d\in\{0,1\}, \tag{3.3}$$

$$\mathbf{x}_{j,k}:=[x_{j,k},y_{j,k}]\quad \text{for}\quad j=-2,\cdots,N_x+2 \text{ and } k=-2,\cdots,N_y+2. \tag{3.4}$$

The mesh points $\mathbf{x}_{j,k}$ are not uniformly distributed: the mesh is *logically* rectangular, but can be solution adaptive in physical space. The domain is now covered by $N_x \times N_y$ convex quadrilaterals $A_{j+\frac{1}{2},k+\frac{1}{2}}$. Besides, there are two rows and columns of ghost cells beyond all four domain boundaries to facilitate the second order stencils of the finite volume solver.

3.2 Second order finite volumes

The finite volume method employed is of second order and uses MUSCL-type solution reconstruction with slope limiting and local Lax-Friedrichs and HLLC numerical fluxes, which we will now discuss in more detail.

Finite volume solvers use average solution values on all mesh cells:

$$\mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} \approx \iint\limits_{A_{j+\frac{1}{2},k+\frac{1}{2}}} \mathbf{q}([x,y],t_n)\,\mathrm{d}x\mathrm{d}y/|A^n_{j+\frac{1}{2},k+\frac{1}{2}}|, \tag{3.5}$$

where $|A^n_{j+\frac{1}{2},k+\frac{1}{2}}|$ is the area of cell $A_{j+\frac{1}{2},k+\frac{1}{2}}$ at time t_n.

The integral form of the PDEs (3.1) leads to the well-known finite volume discretisation:

$$\begin{aligned}\mathbf{Q}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} &= \mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} - \frac{\Delta t_n}{|A^n_{j+\frac{1}{2},k+\frac{1}{2}}|}\Big(|h^n_{j+1,k+\frac{1}{2}}|\check{\mathbf{F}}_{j+1,k+\frac{1}{2}} - |h^n_{j,k+\frac{1}{2}}|\check{\mathbf{F}}_{j,k+\frac{1}{2}} \\ &\quad + |h^n_{j+\frac{1}{2},k+1}|\check{\mathbf{G}}_{j+\frac{1}{2},k+1} - |h^n_{j+\frac{1}{2},k}|\check{\mathbf{G}}_{j+\frac{1}{2},k}\Big) \\ &=: \mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t_n L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^n),\end{aligned} \tag{3.6}$$

where $\check{\mathbf{F}}$ and $\check{\mathbf{G}}$ approximate the normal fluxes across the logically vertical and horizontal edges, respectively, averaged over the time range $[t_n,t_{n+1}]$. The length of the left edge of a cell $A_{j+\frac{1}{2},k+\frac{1}{2}}$ is denoted by $|h_{j,k+\frac{1}{2}}|$.

Fig. 2 depicts the construction of the net, i.e., normal flux across a logically *vertical* edge. In general we can define the flux normal across an edge by taking the inner product of the flux tensor $[\mathbf{F},\mathbf{G}]^T$ with the edge's normal, but here we can instead exploit the rotational invariance of the Euler equations:

$$\check{\mathbf{F}}(\mathbf{Q}) = \cos(\theta)\mathbf{F}(\mathbf{Q}) + \sin(\theta)\mathbf{G}(\mathbf{Q}) = \mathbf{T}^{-1}(\mathbf{F}(\mathbf{T}(\mathbf{Q}))), \tag{3.7}$$

where $\mathbf{T} := \mathbf{T}(\theta)$ is the rotation matrix and $\mathbf{T}^{-1} := \mathbf{T}^{-1}(\theta)$ its inverse for rotating back:

$$\mathbf{T}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}^{-1}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.8}$$

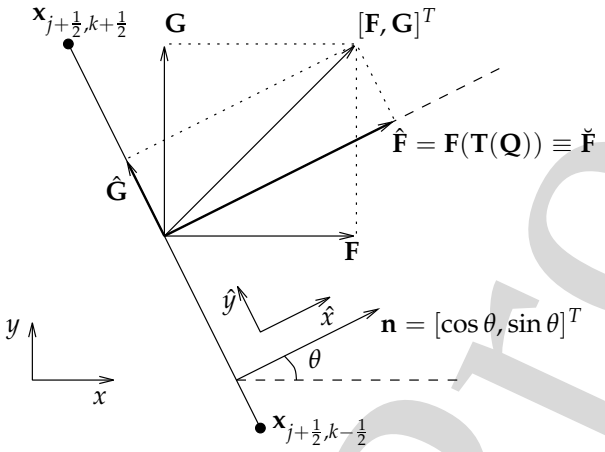

Figure 2: Rotation of the flux tensor into the (logically vertical) edge's normal reference frame. For ease of notation, $\mathbf{F}$ denotes the tensor $[\mathbf{F},\mathbf{0}]^T$, and similar for $\mathbf{G}$, $\hat{\mathbf{F}}$ and $\hat{\mathbf{G}}$.

Eq. (3.7) describes how $\mathbf{Q}$ is first rotated over an angle θ into a new reference frame as $\hat{\mathbf{Q}} = \mathbf{T}(\mathbf{Q})$ where the new $\hat{x}$-direction aligns with the edge's normal. Now, only $\mathbf{F}(\mathbf{T}(\mathbf{Q}))$ needs to be evaluated, since flux $\mathbf{G}(\mathbf{T}(\mathbf{Q}))$ aligns exactly with the edge and thus has zero contribution to the net flux through the edge. Finally, the flux is rotated back into the physical reference frame. This saves us one flux evaluation at each edge and more importantly, the numerical flux evaluation on the edge is now essentially reduced to a one-dimensional problem. This greatly simplifies the use of more advanced approximate Riemann solvers, such as the HLLC solver described in Section 3.2.2. Also note that for $\check{\mathbf{G}}$ on logically *horizontal* edges, the exact same procedure can be used and *again* only flux $\mathbf{F}(\hat{\mathbf{Q}})$ needs to be evaluated, i.e., $\mathbf{G}$ can be discarded completely.

3.2.1 Solution reconstruction and slope limiting on nonuniform meshes

The fluxes are functions of the solution $\mathbf{q}$, so for evaluating the fluxes at the cell edges, solution values first need to be reconstructed from the cell centered values $\mathbf{Q}_{j+\frac{1}{2},k+\frac{1}{2}}$. We use piecewise linear MUSCL reconstruction as proposed by Van Leer [37,38], combined with the van Leer slope limiter. We will now describe the logically horizontal reconstruction of $\mathbf{Q}^n_{j,k+\frac{1}{2}}$ at a vertical edge, the procedure for the other direction is of course similar.

The solution reconstruction is depicted in Fig. 3. It is done over the line segment $l^n_{j,k+\frac{1}{2}}$ between the cell centers $\mathbf{x}^n_{j-\frac{1}{2},k+\frac{1}{2}}$ and $\mathbf{x}^n_{j+\frac{1}{2},k+\frac{1}{2}}$, which intersects with the edge $h^n_{j,k+\frac{1}{2}}$. The adaptive meshes that occur in our experiments have a smooth enough 'curvature' to assume that the intersection of this line and the edge lies approximately at the center of the edge:

$$\mathbf{a}^n_{j,k+\frac{1}{2}} := l^n_{j,k+\frac{1}{2}} \cap h^n_{j,k+\frac{1}{2}} \approx (\mathbf{x}^n_{j,k} + \mathbf{x}^n_{j,k+1})/2 =: \mathbf{x}^n_{j,k+\frac{1}{2}}. \tag{3.9}$$

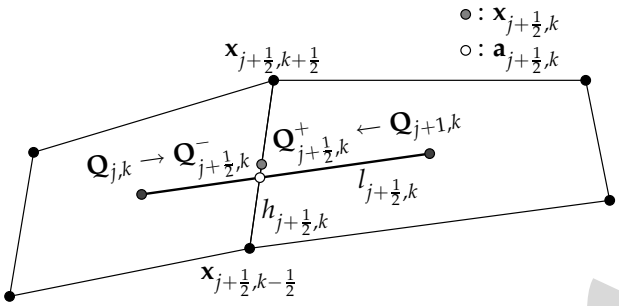

Figure 3: Solution reconstruction on a nonuniform mesh.

The linear reconstructions on a nonuniform mesh are then given by:

$$\mathbf{Q}^{n,\pm}_{j,k+\frac{1}{2}} = \mathbf{Q}^{n}_{j\pm\frac{1}{2},k+\frac{1}{2}} \mp \|\mathbf{x}^{n}_{j,k+\frac{1}{2}} - \mathbf{x}^{n}_{j\pm\frac{1}{2},k+\frac{1}{2}}\|_2 \bar{\mathbf{S}}^{n}_{j\pm\frac{1}{2},k+\frac{1}{2}}, \tag{3.10}$$

where $\bar{\mathbf{S}}^{n}_{j+\frac{1}{2},k+\frac{1}{2}}$ is the limited slope approximation on the cell as defined below,

$$\bar{\mathbf{S}}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} = \phi(\mathbf{S}^{n}_{j+1,k+\frac{1}{2}} / \mathbf{S}^{n}_{j,k+\frac{1}{2}}) \mathbf{S}^{n}_{j,k+\frac{1}{2}}, \tag{3.11}$$

$$\mathbf{S}^{n}_{j,k+\frac{1}{2}} = \frac{\mathbf{Q}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} - \mathbf{Q}^{n}_{j-\frac{1}{2},k+\frac{1}{2}}}{\|\mathbf{x}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} - \mathbf{x}^{n}_{j-\frac{1}{2},k+\frac{1}{2}}\|}, \tag{3.12}$$

where ϕ is the van Leer limiter:

$$\phi(r) = \frac{r+|r|}{1+r}. \tag{3.13}$$

Other well-known slope limiters, such as the Woodward or Koren limiters may be used instead. Note that the reconstruction (3.10) incorporates the mesh nonuniformity and thus is more accurate than when reconstruction is done entirely in the logical domain.

Genuinely multidimensional reconstruction and slope limiting was studied by, e.g., Hubbard [18] and Berger et al. [4]. In the latter, this even involves solving linear programming problems at each cell interface. Our experience is that our quasi-one-dimensional approach (3.10) proves robust for a wide range of problems.

3.2.2 Approximate Riemann solvers

The numerical fluxes in (3.6) are supposed to be averaged at the edge over the time interval $[t_n, t_{n+1}]$. The reconstructed solution values

$$\mathbf{Q}_L := \mathbf{Q}^{n,-}_{j,k+\frac{1}{2}} \text{ and } \mathbf{Q}_R := \mathbf{Q}^{n,+}_{j,k+\frac{1}{2}} \tag{3.14}$$

form a local Riemann problem at the edge. We consider an exact Riemann solver too expensive here, so we use the local Lax-Friedrichs (LLF) flux and the HLLC flux approximations.

Local Lax Friedrichs

The LLF—or Rusanov [24]—flux averages the left and right fluxes and adds a stabilizing numerical diffusion locally:

$$\mathbf{F}^{\text{llf}}(\mathbf{Q}_L,\mathbf{Q}_R)=\frac{1}{2}(\mathbf{f}(\mathbf{Q}_L)+\mathbf{f}(\mathbf{Q}_R))-\frac{1}{2}\max_{K\in\{L,R\}}|\lambda_{\max}(\mathbf{Q}_K)|(\mathbf{Q}_R-\mathbf{Q}_L), \tag{3.15}$$

where $|\lambda_{\max}|$ is the largest absolute eigenvalue of the flux Jacobian $\partial\mathbf{f}/\partial\mathbf{q}$ and $\mathbf{Q}_K$ represents either the left or right solution state. The Lax-Friedrichs flux thanks its robustness to its diffusive nature and the *local* diffusion constant of LLF limits this enough to still obtain accurate results. Especially in combination with adaptive meshes we obtained good results for one-dimensional magnetohydrodynamics [39]. In two dimensions shock waves are also captured very well, but more delicate flow features are not. The moving mesh algorithm can not properly detect these delicate features, because LLF has diffused the solution beforehand. This is the reason why we will use HLLC fluxes instead. Section 5.1.1 compares results obtained with LLF and HLLC fluxes.

HLLC

The MUSCL-LLF combination may be of second order accuracy, it still uses a fairly crude approximation of the actual fluxes across cell edges. This is because it always uses the fastest wave speed to add some local numerical viscosity to the numerical flux function (3.15). It is especially the middle wave, the contact discontinuity (CD) that is harmed by this approximation. Much more viscosity than necessary is added and the sharpness of the CD is generally worse than that of the faster left and right waves.

Harten et al. [14] proposed a new approximate Riemann solver to obtain Godunov-type fluxes, which is now widely known as the HLL Riemann solver. It distinguishes between the leftmost and rightmost waves in a local Riemann problem and approximates the intermediate state by averaging. It still overlooks the CD though. We will now elaborate on this approach in the more general setting of the HLLC solver. The definitions below are complete, but a more in-depth discussion is given by Toro [34].

Toro et al. [33] proposed an improved version of the HLL solver that accurately captures the middle CD wave from the Euler equations. Hence the name HLLC solver. The underlying idea is to distinguish three instead of two waves that emanate from the local Riemann problem between two solution states on the neighboring cells, see Fig. 4.

The local Riemann problem $(\mathbf{Q}_L,\mathbf{Q}_R)$ is one-dimensional due to the rotated reference frame (see Section 3.2). Along each path $\tilde{x}/t$ the solution is constant (where $\tilde{x}:=\hat{x}-\hat{x}_j$ denotes the local coordinate) and the position of this path relative to the three characteristic waves determines in which regime the solution falls:

$$\mathbf{Q}_j^{\text{hllc}}=\begin{cases}\mathbf{Q}_L & \text{if } \tilde{x}/t\le S_L,\\ \mathbf{Q}_{*L} & \text{if } S_L\le\tilde{x}/t\le S_*,\\ \mathbf{Q}_{*R} & \text{if } S_*\le\tilde{x}/t\le S_R,\\ \mathbf{Q}_R & \text{if } S_R\le\tilde{x}/t.\end{cases} \tag{3.16}$$

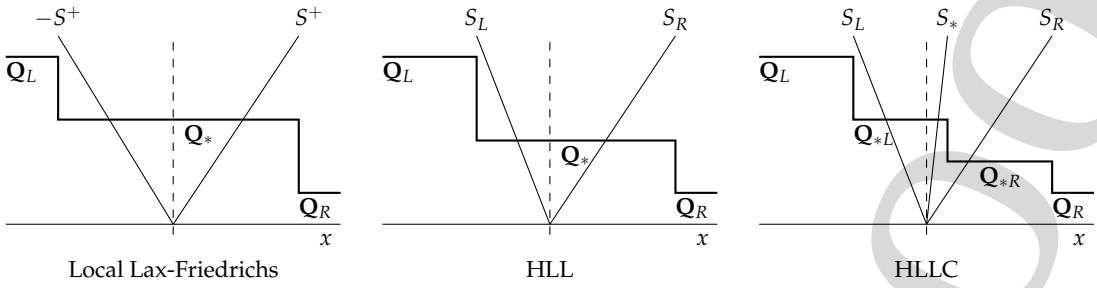

Figure 4: Approximate Riemann solvers: Local Lax Friedrichs, HLL and HLLC. The difference lies in the number of different wave speeds that are used. In between waves, the solution is approximated by a constant state.

By applying Rankine-Hugoniot conditions to the jumps across each of the waves S_L, S_* and S_R, and using additional knowledge about the exact solution jumps across these waves, we obtain the solution vectors in the two intermediate states:

$$\mathbf{Q}_{*K} = \rho_K \left(\frac{S_K - u_K}{S_K - S_*} \right) \begin{bmatrix} 1 \\ S_* \\ v_K \\ \frac{E_K}{\rho_K} + (S* - u_K)\left(S_* + \frac{p_K}{\rho_K (S_K - u_K)} \right) \end{bmatrix} \text{ for } K = L \text{ and } K = R. \tag{3.17}$$

The numerical flux is evaluated at the edge, i.e., $\tilde{x}/t = 0$:

$$\mathbf{F}^{\text{hllc}}_{j+\frac{1}{2}} = \begin{cases} \mathbf{F}_L & \text{if } 0 \le S_L, \\ \mathbf{F}_{*L} = \mathbf{F}_L + S_L(\mathbf{Q}_{*L} - \mathbf{Q}_L) & \text{if } S_L \le 0 \le S_*, \\ \mathbf{F}_{*R} = \mathbf{F}_R + S_R(\mathbf{Q}_{*R} - \mathbf{Q}_R) & \text{if } S_* \le 0 \le S_R, \\ \mathbf{F}_R & \text{if } S_R \le 0. \end{cases} \tag{3.18}$$

The left- and right-most wave speeds are chosen as follows:

$$S_L = \min\{(u-c)_L, (u-c)_R\} \text{ and } S_R = \max\{(u+c)_L, (u+c)_R\}, \tag{3.19}$$

where c is the sound speed. The speed S_* of the intermediate wave can be obtained by realizing that the pressure is constant across a CD: $p_{*L} = p_{*R}$, which gives:

$$S_* = \frac{p_R - p_L + \rho_L u_L (S_L - u_L) - \rho_R u_R (S_R - u_R)}{\rho_L (S_L - u_L) - \rho_R (S_R - u_R)}. \tag{3.20}$$

When S_R (or S_L) and S_* coincide, HLLC reduces to HLL. Besides, when we set $S_L = -S_R = -\max_{K \in \{L,R\}} |\lambda_{\max}|_K$ in HLL, the method reduces to LLF. Experiments confirm this up to machine precision.

3.2.3 Two-step explicit time integration

The PDEs (3.1) are integrated in time by Heun's predictor-corrector method, which has second-order accuracy. Using the notation from (3.6), we now use two FV-steps:

$$\begin{aligned}\mathbf{Q}^{*}_{j+\frac{1}{2},k+\frac{1}{2}} &= \mathbf{Q}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t_n L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^{n}),\\ \mathbf{Q}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} &= \mathbf{Q}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t_n \frac{L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^{n}) + L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^{*})}{2},\end{aligned} \tag{3.21}$$

where the $L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q})$ operator computes the discretised flux gradients by the MUSCL-HLLC combination described before.

Since the underlying mesh is nonuniform, the CFL stability condition is enforced on each mesh cell locally. We define the quasi-one-dimensional mesh cell sizes in the rotated frame as:

$$\Delta\hat{x}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} := \frac{|A^{n}_{j+\frac{1}{2},k+\frac{1}{2}}|}{\max\{|h^{n}_{j,k+\frac{1}{2}}|, |h^{n}_{j+1,k+\frac{1}{2}}|\}}, \quad \Delta\hat{y}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} := \frac{|A^{n}_{j+\frac{1}{2},k+\frac{1}{2}}|}{\max\{|h^{n}_{j+\frac{1}{2},k}|, |h^{n}_{j+\frac{1}{2},k+1}|\}}. \tag{3.22}$$

Next, we apply the CFL-stability criterion in the following way:

$$\Delta t^{n} \leq \mathcal{C} \min_{j,k} \frac{\min\{\Delta\hat{x}^{n}_{j+\frac{1}{2},k+\frac{1}{2}}, \Delta\hat{y}^{n}_{j+\frac{1}{2},k+\frac{1}{2}}\}}{\max\{|\lambda_{1,\max}(\mathbf{Q}^{n}_{j+\frac{1}{2},k+\frac{1}{2}})|, |\lambda_{1,\max}(\mathbf{Q}^{n}_{j+\frac{1}{2},k+\frac{1}{2}})|\}}, \tag{3.23}$$

where $\lambda_{1,\max}$ and $\lambda_{2,\max}$ are the largest eigenvalues in the x- and y-direction, respectively.

A looser CFL-criterion will not improve performance very much. More severe is the fact that the smallest cell sizes that will occur during mesh adaptation—typically 5 to 50 times smaller than the original uniform mesh—limit the overall time step. This is a general problem of adaptive mesh methods and could be solved by local time stepping. For h-refinement methods this is widely used, and for moving mesh methods the procedure is essentially the same and fairly straightforward. Tan et al. [28] have done so and report performance improvements by a factor 2 to 3.

3.3 Adaptive moving mesh method

The problem domain is discretised as a structured mesh with a fixed number of mesh points. The moving mesh algorithm moves the mesh points towards interesting flow phenomena, which are time-dependent.

3.3.1 Background

The adaptation is represented by a mesh map $\mathbf{x}(\boldsymbol{\xi})$, where $\boldsymbol{\xi} := [\xi, \eta] \in \Omega_c$ are the reference coordinates in the computational domain $\Omega_c := [0,1] \times [0,1]$.

One of the earliest works here is by Winslow [40] in which he generalizes the elliptic mesh generator to a solution-adaptive form:

$$\nabla\cdot(D\nabla\xi)=0,\quad \nabla\cdot(D\nabla\eta)=0. \tag{3.24}$$

Here the 'diffusion' coefficient $D>0$ could depend on the solution gradient. Instead of inverting (3.24) to obtain $\mathbf{x}(\boldsymbol{\xi})$, Ceniceros and Hou [8] formulate the elliptic generator directly in the covariant form:

$$\nabla_\xi\cdot(\mathbf{G}\nabla_\xi x)=0,\quad \nabla_\xi\cdot(\mathbf{G}\nabla_\xi y)=0, \tag{3.25}$$

where $\nabla_\xi:=[\partial/\partial\xi,\partial/\partial\eta]^T$ is the computational gradient. Solving these equations directly yields the adaptive mesh $[x(\xi,\eta),y(\xi,\eta)]$.

In the above, D and $\mathbf{G}$ are still scalar functions (take $\mathbf{G}=g\mathbf{I}$), but later work by, e.g., Cao et al. [7] and Tang [31] proposes to make $\mathbf{G}$ a symmetric positive definite matrix with elements $g_{1,1}\neq g_{2,2}$[†] such that the solution adaptivity becomes directional (comparable with anisotropic local mesh refinement). This is also what we do.

All of the above work except Winslow's uses an elliptic PDE system that originates from a variational formulation of a minimization problem. For example the solution to PDE (3.25) is the minimizer of the 'energy' functional

$$E(\mathbf{x}(\boldsymbol{\xi}))=\frac{1}{2}\iint_{\Omega_c}\left[\nabla_\xi^T\mathbf{G}\nabla_\xi x+\nabla_\xi^T\mathbf{G}\nabla_\xi y\right]\mathrm{d}\xi\mathrm{d}\eta. \tag{3.26}$$

The moving mesh algorithm aims to find a mesh map with a low energy value. The lower the energy, the more appropriate is the mesh map $\mathbf{x}(\boldsymbol{\xi})$ according to the monitor function $\mathbf{G}$.

Brackbill and Saltzman [6] were amongst the first to start from a variational formulation and they combined three functionals to control both mesh smoothness, orthogonality and adaptivity. We will only study adaptivity here, since the balanced monitor function (4.9) in Section 4 helps keeping the mesh smooth. Still, we do see advantages in orthogonality monitors in future work.

3.3.2 Algorithm

The mesh movement algorithm is similar to the one set out by Tang and Tang [30]. We propose a much more versatile and robust monitor function, though, which will be described in Section 4.

The mesh movement equations (3.25) are solved separately from the physical PDEs (3.1). In each iteration step, first the mesh is moved to adapt to the latest solution features. Next, the nonuniform mesh is kept fixed during one forward time integration step. We omit the time index n in the coordinates and solution values, since it does not change during the mesh adaptation step. Algorithm 3.1 summarises it all.

[†]Cao et al. [7] even proposes nonzero elements off the diagonal, but we do not consider that here.

Algorithm 3.1: MMFVSOLVE – 2D adaptive moving mesh finite volume PDE solver.

1: $n \leftarrow 0$; $t_0 \leftarrow 0$
2: Generate an initial uniform mesh $\mathbf{x}^0_{j,k}$.
3: Compute initial values $\mathbf{Q}^0_{j+1/2,k+1/2}$.
4: **while** $t_n < T$ **do**
5: $\quad \nu \leftarrow 0$; $\mathbf{x}^{[0]}_{j,k} \leftarrow \mathbf{x}^n_{j,k}$; $\mathbf{Q}^{[0]}_{j+1/2,k+1/2} \leftarrow \mathbf{Q}^n_{j+1/2,k+1/2}$.
6: $\quad$ **repeat**
7: $\quad\quad$ Evaluate monitor function (4.9) and filter it (Section 4.5).
8: $\quad\quad$ Move mesh $\mathbf{x}^{[\nu]}_{j,k}$ to $\mathbf{x}^{[\nu+1]}_{j,k}$, by Gauss-Seidel of (3.27).
9: $\quad\quad$ Conservative interpolation of $\mathbf{Q}^{[\nu+1]}_{j+1/2,k+1/2}$ by (3.29) (or re-initialize at $t=t_0$).
10: $\quad\quad$ $\nu \leftarrow \nu+1$
11: $\quad$ **until** $\nu \geq \nu_{\max}$ or $\left\| \mathbf{x}^{[\nu]} - \mathbf{x}^{[\nu-1]} \right\|_{\text{rel}} \leq \epsilon$
12: $\quad$ Fix new mesh $\mathbf{x}^n \leftarrow \mathbf{x}^{[\nu]}$ and solution $\mathbf{Q}^n \leftarrow \mathbf{Q}^{[\nu]}$.
13: $\quad$ $t_{n+1} \leftarrow t_n + \Delta t_n$ by CFL criterion (3.23).
14: $\quad$ Compute $\mathbf{Q}^{n+1}$ using finite volumes (Section 3.2).
15: $\quad$ $\mathbf{x}^{n+1} \leftarrow \mathbf{x}^n$.
16: $\quad$ $n \leftarrow n+1$.
17: **end while**

Mesh adaptation We combine the moving mesh PDEs (3.25) with a directional‡ monitor function $\mathbf{G} = \text{diag}(\omega^{(1)}, \omega^{(2)})$. All gradients are discretised by central differences and the monitor values on the middle of each edge are averaged between two cell centers. Next, a Gauss-Seidel step is used to compute the new mesh points:

$$\mathbf{x}^{[\nu+1]}_{j,k} = \left[\frac{\omega^{(1)}_{j-\frac{1}{2},k}\mathbf{x}^{[\nu+1]}_{j-1,k} + \omega^{(1)}_{j+\frac{1}{2},k}\mathbf{x}^{[\nu]}_{j+1,k}}{(\Delta\xi)^2} + \frac{\omega^{(2)}_{j,k-\frac{1}{2}}\mathbf{x}^{[\nu+1]}_{j,k-1} + \omega^{(2)}_{j,k+\frac{1}{2}}\mathbf{x}^{[\nu]}_{j,k+1}}{(\Delta\eta)^2} \right] \Big/ \left(\omega^{(1)}_{j-\frac{1}{2},k} + \omega^{(1)}_{j+\frac{1}{2},k} + \omega^{(2)}_{j,k-\frac{1}{2}} + \omega^{(2)}_{j,k+\frac{1}{2}} \right). \tag{3.27}$$

Notice how these are in fact two equations: one for x and one for y. The coefficients for the two are identical, yet the equations for x and y are independent, i.e., they do not affect each other directly. The boundary points can move *along* the boundary. We do this by setting $x_{0,k} = x_{\min}$, $y_{0,k} = y_{1,k}$ in (3.27) for the left boundary and similar for the other three.

Conservative solution interpolation The discrete solution values have to be updated after the mesh cells have been changed. Conservation of the solution variables (mass, energy, etc.) is an important requirement for an accurate compressible flow solver. Well-known is the approach by Tang and Tang [30], which considers the velocity of the mesh

‡See Section 4.4. The direction indices (1), (2) are in superscript here, to avoid confusion with point indices j,k.

298 points as an artificial flux. Han and Tang [13] formulate an alternative geometrical ap-
299 proach, which may be slightly more accurate. Both maintain global conservation in the
300 following way:

$$\sum_{j,k}\left|A^{[\nu+1]}_{j+\frac{1}{2},k+\frac{1}{2}}\right|\mathbf{Q}^{[\nu+1]}_{j+\frac{1}{2},k+\frac{1}{2}}=\sum_{j,k}\left|A^{[\nu]}_{j+\frac{1}{2},k+\frac{1}{2}}\right|\mathbf{Q}^{[\nu]}_{j+\frac{1}{2},k+\frac{1}{2}}.$$

301 Zhang [45] devises a new approach based on L^2-projection, where solution conservation
302 is even preserved in each cell. We employ the first method, though, because of our good
303 experiences with it in the past.

304 The movement of a cell's edges causes an artificial flux across them. The difference
305 between the old and new mesh points is defined as

$$\mathbf{c}_{j,k}:=\mathbf{x}^{[\nu]}_{j,k}-\mathbf{x}^{[\nu+1]}_{j,k}. \tag{3.28}$$

Assuming that this difference is small, the following approximation for the new solution can be derived:

$$\begin{aligned}&\left|A^{[\nu+1]}_{j+\frac{1}{2},k+\frac{1}{2}}\right|\mathbf{Q}^{[\nu+1]}_{j+\frac{1}{2},k+\frac{1}{2}}\\ =&\left|A^{[\nu]}_{j+\frac{1}{2},k+\frac{1}{2}}\right|\mathbf{Q}^{[\nu]}_{j+\frac{1}{2},k+\frac{1}{2}}-\left((\widehat{c_n\mathbf{Q}})_{j+1,k+\frac{1}{2}}|h_{j+1,k+\frac{1}{2}}|-(\widehat{c_n\mathbf{Q}})_{j,k+\frac{1}{2}}|h_{j,k+\frac{1}{2}}|\right)\\ &-\left((\widehat{c_n\mathbf{Q}})_{j+\frac{1}{2},k+1}|h_{j+\frac{1}{2},k+1}|-(\widehat{c_n\mathbf{Q}})_{j+\frac{1}{2},k}|h_{j+\frac{1}{2},k}|\right).\end{aligned} \tag{3.29}$$

306 The numerical fluxes $\widehat{c_n\mathbf{Q}}$ are defined by upwind fluxes

$$\widehat{c_n\mathbf{Q}}_{j,k+\frac{1}{2}}=(c_n^+\mathbf{Q})^-_{j,k+\frac{1}{2}}+(c_n^-\mathbf{Q})^+_{j,k+\frac{1}{2}}, \tag{3.30}$$

307 where the two MUSCL-type solution reconstructions $\mathbf{Q}^-$ and $\mathbf{Q}^+$ are again defined by
308 (3.10). The upwind choice is defined by:

$$c_n^{\pm}=\frac{c_n\pm|c_n|}{2} \tag{3.31}$$

and the artificial advection c_n through an edge is simply the inner product of the mid-point movement with the edge's right- or upward normal, e.g.,

$$(c_n)_{j,k+\frac{1}{2}}=\frac{\mathbf{c}_{j,k}+\mathbf{c}_{j,k+1}}{2}\cdot\mathbf{n}_{j,k+\frac{1}{2}}, \tag{3.32a}$$

$$\mathbf{n}_{j,k+\frac{1}{2}}=[y_{j,k+1}-y_{j,k},-(x_{j,k+1}-x_{j,k})]/\|\mathbf{x}_{j,k+1}-\mathbf{x}_{j,k}\|. \tag{3.32b}$$

Degeneracy of the mesh The solution of the mesh PDEs (3.25) is a mesh map $\mathbf{x}(\boldsymbol{\xi})$, which is *unique* and *regular* as long as the monitor matrix $\mathbf{G}$ is diagonal and strictly positive. This result is a special case of the proof by Clément et al. [9]. In other words: the mesh map has a strictly positive Jacobian

$$J:=\det((\nabla_{\xi}\mathbf{x})^T),$$

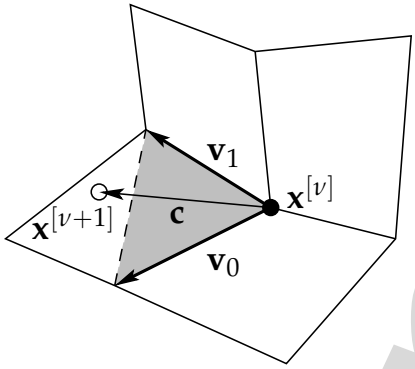

Figure 5: Protection against mesh collapse by limiting the mesh velocity vector. The new mesh point (white circle) should lie in the gray region.

so mesh cells can not collapse in the exact solution.

The nonlinear mesh PDEs (3.25) are linearized and then solved, though, so nondegeneracy can not be guaranteed anymore. Our experience is that with our balanced monitor function and monitor filtering, this hardly ever occurs. To strictly ensure nondegeneracy and—even stronger—convexity of cells mesh, the following check can be used while moving the mesh points. The displacement of a mesh point as defined in (3.28) must not cross the 'antidiagonal' of the one (out of four) cells it moves into. Fig. 5 depicts this requirement. Expanding the displacement vector $\mathbf{c}$ in the two edge vectors:

$$\mathbf{c} = a_0\mathbf{v}_0 + a_1\mathbf{v}_1,$$

convexity is maintained as long as $a_0 + a_1 \leq 1$ holds. To prevent violation of this requirement, the new point location is limited as follows:

$$\mathbf{x}_{j,k}^{[\nu+1]*} := \mathbf{x}_{j,k}^{[\nu]} + \min\left\{\frac{\mu}{a_0+a_1}, 1\right\}\mathbf{c}_{j,k}, \quad \text{with } 0 < \mu \leq 1, \tag{3.33}$$

where the parameter μ controls how strict the convexity condition is ensured.

The mesh adaptation algorithm is now complete. The next section will introduce a new monitor function that makes the algorithm very robust.

4 Monitor functions

The key to a successful moving mesh method is a proper monitor function. Firstly, it should detect the relevant flow features, thereby reducing errors caused by the physical flow solver. Secondly, it should be widely applicable, that is, require no or little manual fine-tuning for each new problem at hand. Finally, it should be relatively smooth in space and time, thereby reducing errors caused by mesh movement.

4.1 What makes a good monitor?

A basic monitor function is quickly defined, but true gain in more complex flow simulations can only be obtained with a more sophisticated monitor. The arc length based (AL) monitor may be intuitive, since it refines the mesh where the solution is steep:

$$\omega = \sqrt{1+\alpha\|\nabla q\|^2}. \tag{4.1}$$

The assumption of a scalar solution q is for simplicity here and will be extended to vector-valued solutions $\mathbf{q}$ in Section 4.3. The AL monitor is not balanced, though, more specifically it has three main disadvantages:

- The AL monitor requires the user to properly choose adaptation parameter α, which is problem dependent and not dimensionless.
- The AL monitor has no time-dependent adaptivity, since α is fixed during a run. Whenever the solution gradients change significantly over time, the chosen value for α becomes unsuitable.
- The AL monitor often lacks smoothness, resulting in too rapidly varying cell sizes.

We solve the above problems by several improvements, which will be discussed in this and the following sections.

4.2 An adaptive monitor function

The disadvantages of the adaptivity parameter α in the AL monitor (4.1) mentioned in the previous section, can be summarized as follows: α is not dimensionless, nor scaling invariant, nor time-dependent. Beckett and Mackenzie (BM) [3] have proposed an alternative for the AL-monitor: $\alpha(q)+\|\nabla q\|^{1/m}$, where the new floor value $\alpha(q)$ is the average value of the gradient on the entire domain. The smoothness parameter m replaces the square root in (4.1) and controls the importance given to gradient differences (the limit $m\to\infty$ produces a uniform mesh). We fix it at $m=1$ unless specified otherwise. We start with an 'average normalization' of the solution gradient, which is equivalent to the BM-monitor:

$$\omega(q) = 1+\frac{\|\nabla q\|^{\frac{1}{m}}}{\alpha(q)}, \quad \text{where } \alpha(q) = \iint_\Omega \|\nabla q\|^{\frac{1}{m}}\,\mathrm{d}\mathbf{x}, \quad m>0. \tag{4.2}$$

The next section will generalize this to the case where the solution is a vector $\mathbf{q}$, i.e., as in (3.1). We call this an *adaptive monitor function* for the following reasons. In the above monitor the solution gradients have now become dimensionless. There is also a much better balance between large and small gradients ($\omega(q)\in[1,2]$), so the mesh points are

spread in a more balanced way. Moreover, the normalization by $\alpha(q)$ is time-dependent, since solution $q(\mathbf{x},t)$ itself is time-dependent. Further smoothness is obtained by using computational gradients ∇_ξ instead of physical gradients ∇. This is also motivated by the following [8]. On the reference domain, the solution $\tilde{q}(\boldsymbol{\xi}) := q(\mathbf{x}(\boldsymbol{\xi}))$ should be more regular. Hence, a good mesh map should minimize the computational gradient $\nabla_\xi \tilde{q}(\boldsymbol{\xi})$ in the functional formulation (3.26).

The above monitor assigns approximately half of the mesh points to 'important' areas (see for example [17]). If a solution needs refinement in only a small part of the domain's total area, this may be too much. We include a dimensionless and solution-independent parameter β (see (4.3)) that gives the user generic refinement control. All experiments in Section 5 use $\beta = 0.3$, which means approximately 30% of the mesh points in important areas. It is the only essential parameter that the user may choose for a particular problem.

4.3 Balancing monitor components

We now generalize the adaptive monitor function to systems of PDEs, i.e., solution *vectors* $\mathbf{q}(\mathbf{x},t) \in \mathbb{R}^M$, where M is the number of solution components. Besides, we allow for monitor components other than solution gradients. For now, the generalized monitor function is defined as a weighted sum of P nonnegative monitor components $\phi_{i,p}(\mathbf{q})$:

$$\omega_i(\mathbf{q}) = \sum_{p=1}^{P} \omega_{i,p}(\mathbf{q}) = \sum_{p=1}^{P} \left[(1-\beta) + \frac{\beta}{\alpha_{i,p}(\mathbf{q})} \phi_{i,p}(\mathbf{q}) \right], \quad i \in \{1,2\}, \tag{4.3}$$

with a separate normalization for each monitor component and spatial direction:

$$\alpha_{i,p}(\mathbf{q}) = \max \left[\iint_{\Omega_c} \phi_{i,p}(\mathbf{q}) \, \mathrm{d}\boldsymbol{\xi}, \, \epsilon \right]. \tag{4.4}$$

Here, $0 < \epsilon \ll 1$ prevents division by zero if the component $\phi_{i,p}$ is zero everywhere on the domain. The normalization by $\alpha_{i,p}(\mathbf{q})$ will be reconsidered in Section 4.3.1.

The default choice for the monitor components is to use all M solution gradients:

$$\phi_{i,p}(\mathbf{q}) = \left| \frac{\partial q_p}{\partial \xi_i} \right|^{1/m}, \quad p = 1, \cdots, M, \text{ i.e., } P = M. \tag{4.5}$$

Other monitor components will be used in Sections 4.3.2 and 5.

Notice how the monitor values and gradients are subscripted by i. This defines a *directional* monitor function, which will be discussed in Section 4.4.

4.3.1 Component imbalance

The adaptive monitor function (4.3) automatically gives proper weight to both steep and smooth solution parts. This is per component, though. The function within the summation may have very different ranges for the various $\omega_{i,p}$. This is because no standard

normalization of components $\phi_{i,p}$ was done. Instead of divided by the *maximum* as in (4.6) below, the components were divided by the *average* as in (4.7). We will now elaborate on the above.

Without loss of generality, in the following we set $\beta = 0.5$ and consider the monitor components in one direction (ignore i in (4.3), (4.4), (4.5)). Starting from the AL monitor (4.1) a possible way of balancing monitor components would be standard normalization by dividing by the maximum component value:

$$\omega_p(\mathbf{q}) = 1 + \frac{\phi_p(\mathbf{q})}{\max_\Omega \phi_p(\mathbf{q})} \equiv 1 + \frac{\phi_p(\mathbf{q})}{\mathcal{M}_p(\mathbf{q})} \quad \in [1,2]. \tag{4.6}$$

The disadvantage is that a single very large maximum value $\mathcal{M}_p(\mathbf{q})$ will dominate all other monitor values on the rest of the domain. Instead, the adaptive monitor (4.3) uses the average value $\alpha_p(\mathbf{q})$ for normalization:

$$\omega_p(\mathbf{q}) = 1 + \frac{\phi_p(\mathbf{q})}{\iint_{\Omega_c} \phi_p(\mathbf{q}) \mathrm{d}\boldsymbol{\xi}} \equiv \frac{\alpha_p(\mathbf{q}) + \phi_p(\mathbf{q})}{\alpha_p(\mathbf{q})} \quad \in \left[1, 1 + \frac{\mathcal{M}_p(\mathbf{q})}{\alpha_p(\mathbf{q})}\right]. \tag{4.7}$$

If one component $\phi_p(\mathbf{q})$ of the P monitor components has a large maximum and relatively small average it will dominate the other components, because of the upper limit of its range: $\mathcal{M}_p(\mathbf{q})/\alpha_p(\mathbf{q}) \gg 1$. This is solved by a second normalization of (4.7):

$$\begin{aligned} \omega_p(\mathbf{q}) &= \frac{(\alpha_p(\mathbf{q}) + \phi_p(\mathbf{q}))/\alpha_p(\mathbf{q})}{\max\limits_\Omega [(\alpha_p(\mathbf{q}) + \phi_p(\mathbf{q}))/\alpha_p(\mathbf{q})]} \equiv \frac{\alpha_p(\mathbf{q}) + \phi_p(\mathbf{q})}{\alpha_p(\mathbf{q}) + \mathcal{M}_p(\mathbf{q})} \\ &\in \left[\frac{\alpha_p(\mathbf{q})}{\alpha_p(\mathbf{q}) + \mathcal{M}_p(\mathbf{q})}, 1\right] \subseteq \langle 0,1]. \end{aligned} \tag{4.8}$$

This 'average-max' normalization defines our final form, hereafter called the *balanced monitor function*:

$$\omega_i(\mathbf{q}) = \sum_{p=1}^{P} \omega_{i,p}(\mathbf{q}) = \sum_{p=1}^{P} \left[\frac{(1-\beta)\alpha_{i,p} + \beta\phi_{i,p}(\mathbf{q})}{(1-\beta)\alpha_{i,p} + \beta\mathcal{M}_{i,p}(\mathbf{q})}\right], \quad i \in \{1,2\}. \tag{4.9}$$

There are four important points to note on the above. Firstly, the reason for using normalizations at all is that all components $\phi_p(\mathbf{q})$ are summed. If one component has very large values, without normalization the total monitor value $\omega = \sum_{p=1}^{P} \omega_p$ would be large there as well, and all other monitor values on the rest of the domain would have a negligible effect on the mesh movement. Secondly, a standard normalization as in (4.6) is balanced across components, as the range is always equal to $[1,2]$. *Within* one component, though, all subtle variations may be diminished by a very large maximum value. We have shown the disadvantages of (4.6) in a 1D MHD setting [39] and therefore discard it here. Thirdly, for a single component $\phi_p(\mathbf{q})$, variants (4.7) and (4.8) are completely equivalent. The latter is obtained by dividing the former by its maximum value and scalar multiplication

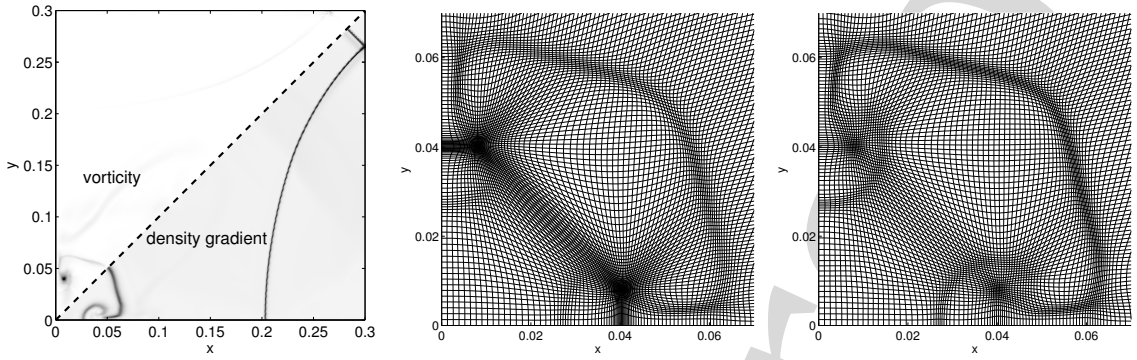

Figure 6: Solving the imbalance between monitor components in the HD22IMPDIAG example. Left: solution components at $t=0.4$, upper triangular half shows the vorticity, lower half shows the norm of the density gradient. Middle: mesh detail using the unbalanced monitor function (4.3). Right: mesh detail using the balanced monitor function (4.9).

of a monitor function has no effect on the mesh refinement. Hence, the new form (4.8) is consistent with (4.7) for the case $P=1$. Also note that the evaluation of the new form is hardly anything more expensive than the old form: only the maximum of all (readily known) component values ϕ_p needs to be determined and added to the already known average. Fourthly, all components ω_p are now in better balance with each other, since they share the same upper limit of 1. There is no risk of dividing by zero, since this minimal value in $\langle 0,1]$ is never reached.

4.3.2 Proof of concept

The necessity for replacing variant (4.7) is illustrated by the HD22IMPDIAG example problem (full specification in Section 5.1). We only include the density gradient and vorticity in the monitor summation:

$$\phi_{i,1}(\mathbf{q})=\left|\frac{\partial\rho}{\partial\xi_i}\right|,\quad \phi_{i,2}(\mathbf{q})=\|\nabla\times\mathbf{v}\|,\quad i\in\{1,2\}. \tag{4.10}$$

The left diagram in Fig. 6 shows the two monitor components at $t=0.4$ together, since the solution is symmetric in the diagonal $x=y$. The density gradient shows several flow features, spread throughout the domain. In contrast, the vorticity reveals only one local feature in the jet's head near $(0.008,0.04)$. The ratio $\mathcal{M}_2/\alpha_2$ for the vorticity will therefore be much larger than the ratio $\mathcal{M}_1/\alpha_1$ for the density gradients.

The middle diagram in Fig. 6 shows the bottom left part of the domain. The result is a bad mesh for the unbalanced monitor variant (4.3): the vorticity attracts the mesh too much towards the two rotational points, and the other features receive less attention. Also between the two rotational points, unnecessary mesh skewness occurs. The third diagram shows how the new monitor variant (4.9) properly balances the two monitor components, resulting in a high quality adaptive mesh.

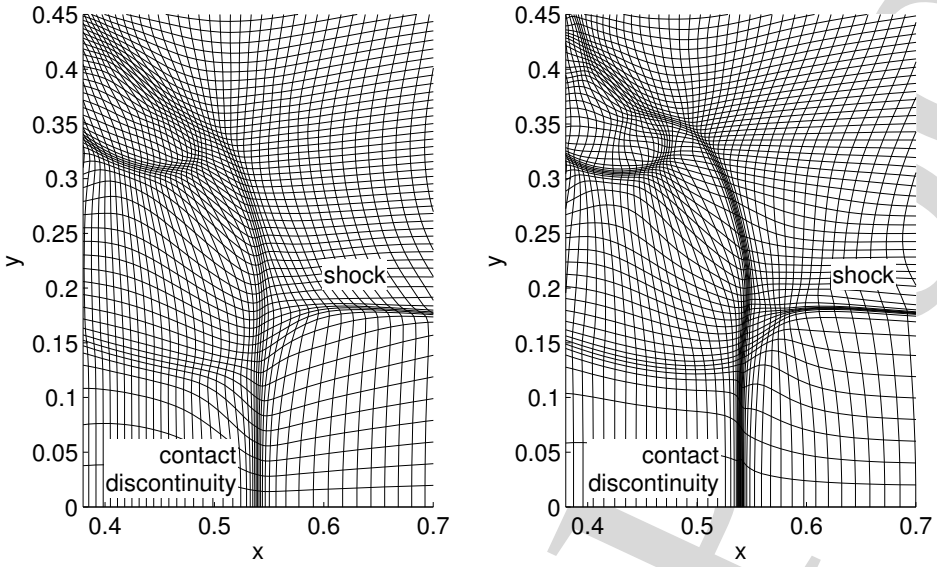

Figure 7: Nondirectional (left) and directional (right) mesh adaptation for the HD22CONF11 example problem. Close-up of mesh near $(0.53,0.2)$.

4.4 The importance of directionality

In two- or higher-dimensional mesh adaptivity a directional—or equivalently: anisotropic—monitor function is essential. It attracts mesh points from the direction in which a solution feature is observed; points from other directions are more or less unaffected. This leads to sharper refinement at points where multiple solution features from different directions meet, since there is less competition in attracting points.

In the mesh PDEs (3.25) the monitor matrix $\mathbf{G}$ prescribes the monitor values for all directions. Usually a diagonal matrix is used; if the diagonal elements are identical, the mesh adaptation is nondirectional (isotropic). We use directional monitor values as in (4.9), i.e., the second form below:

$$\mathbf{G}_{\text{nondir}} = \begin{bmatrix} \omega & 0 \\ 0 & \omega \end{bmatrix}, \quad \mathbf{G}_{\text{dir}} = \begin{bmatrix} \omega_1 & 0 \\ 0 & \omega_2 \end{bmatrix}. \tag{4.11}$$

The HD22CONF11 example problem (full specification in Section 5.2) illustrates the improved mesh for a directional monitor. Fig. 7 shows the adapted mesh for a nondirectional monitor (left diagram) and a directional monitor (right diagram). The mesh adaptation across the vertical contact discontinuity (CD) is good in both cases. In the nondirectional case, however, mesh points have been attracted tangential to the CD as well. This is not only unnecessary, it also pulls mesh points away from the area around $(0.53,0.23)$ where the CD and the horizontal shock meet. The directional case shows a higher quality mesh near all of these features. Also the spirals, e.g., near $(0.46,0.35)$, are better captured in this case.

4.5 Monitor filtering

Flow phenomena are now properly captured, but since they generally move, it is sensible to also refine the mesh in a small region around them. This is done by *filtering* of the discrete monitor values. We apply the following widely-used Gaussian filter, typically 2 to 5 times:

$$\omega_{j,k}^{(i),\text{filter}} := \frac{1}{16}\Big(4\omega_{j,k}^{(i)} + 2\cdot(\omega_{j-1,k}^{(i)} + \omega_{j+1,k}^{(i)} + \omega_{j,k-1}^{(i)} + \omega_{j,k+1}^{(i)}) + \omega_{j-1,k-1}^{(i)} + \omega_{j+1,k-1}^{(i)} + \omega_{j-1,k+1}^{(i)} + \omega_{j+1,k+1}^{(i)}\Big) \tag{4.12}$$

for $i=1$ and 2 independently.

5 Experiments

5.1 HD22IMPDIAG: a symmetric implosion with jets

The implosion problem (HD22IMPDIAG) by Hui et al. [19]—also extensively studied by Liska and Wendroff [22]—describes an initial discontinuity across the line $x+y=0.15$ on a domain $[0,0.3]\times[0,0.3]$:

$$\begin{aligned}[\rho,u,v,p] &= [0.125,\ 0,\ 0,\ 0.14], \quad \text{if } x+y\le 0.15,\\ [\rho,u,v,p] &= [\quad 1,\ 0,\ 0,\quad 1], \quad \text{otherwise.}\end{aligned}$$

Actually, this forms only the first quadrant of the full configuration, but the full solution can be obtained by symmetry in both coordinate axes. All boundaries are reflective, i.e., homogeneous Neumann conditions except for the antisymmetric normal velocity component.

Fig. 8 shows the density evolution over time. Along the diagonal, the initial discontinuity breaks up into a shock, a contact discontinuity (CD) and a rarefaction fan. The shock causes a Mach reflection, and the reflected wave causes a second Mach reflection where it meets the CD (see first diagram). Along the axes $x=0$ and $y=0$ two jets emanate from this CD (see second diagram). This wall-jetting effect has been studied extensively, e.g., by Henderson et al. [15]. However, quantitative analysis is very complicated and rarely seen. We will study the sharpness of the jet front and its velocity in Section 5.1.3.

After some time, the two jets meet at the origin and merge into one jet that continues to move up the diagonal (see fourth diagram). The evolution of this and following jets are often watched to test numerical methods on symmetry preservation. In the meantime the shock and its reflections repeatedly interact with these jets and the original CD, causing Richtmyer-Meshkov instabilities along the latter. Also, along the interface between forward moving jets and the outward expanding surroundings, Kelvin-Helmholtz instabilities are formed (see third diagram).

We will first make a comparison between the LLF and HLLC fluxes from Section 3.2.2. Next, we will illustrate the monitor component balancing from Section 4.3. Finally, we will focus on details of the jets and the formation of instabilities.

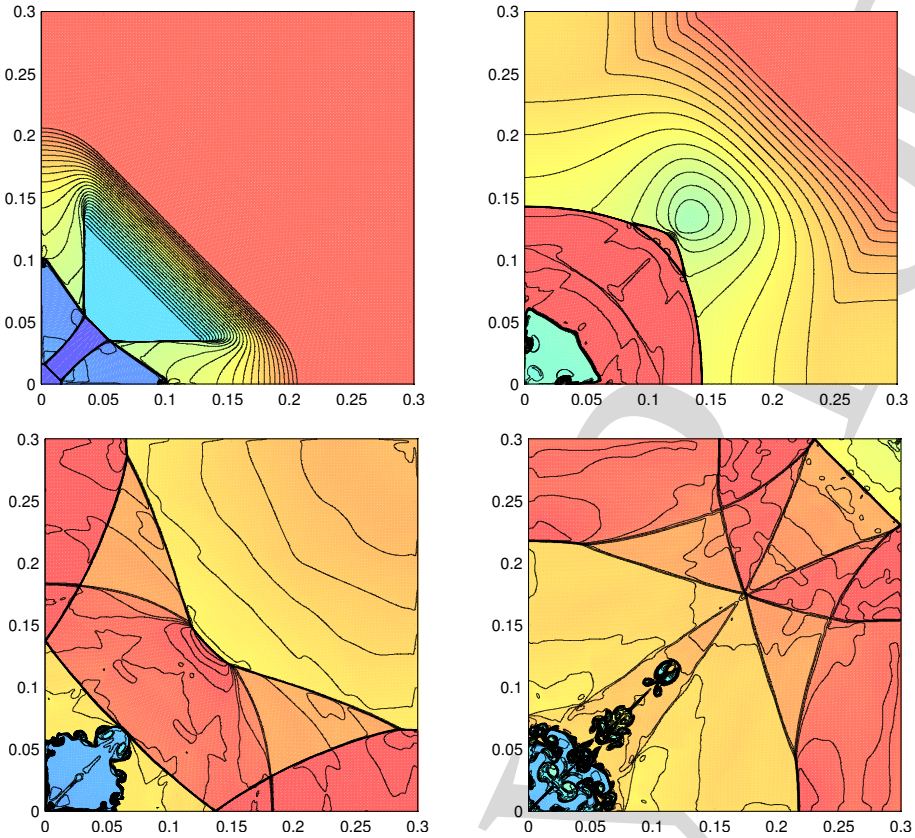

Figure 8: Time evolution of the HD22IMPDIAG problem. The colors and contours show the density. From top left, top right, bottom left and bottom right are the snapshots at $t=0.05$, 0.175, 0.7 and 1.5.

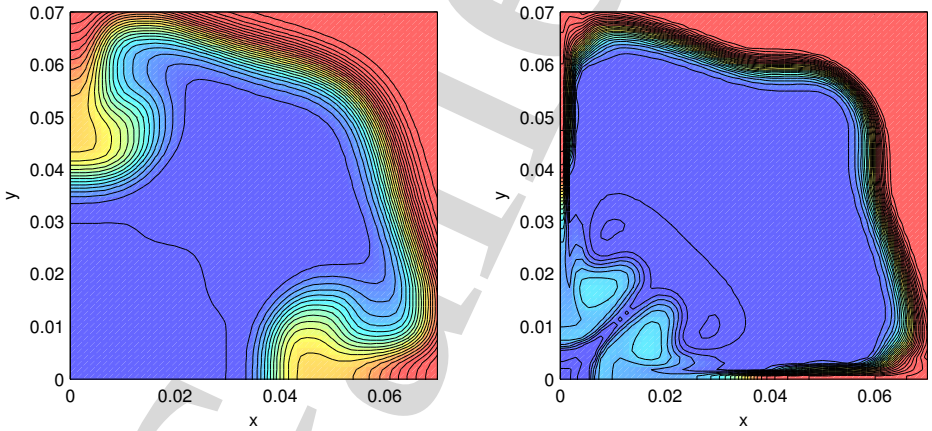

Figure 9: Non-adaptive local Lax-Friedrichs versus HLLC flux. Close-up of the HD22IMPDIAG problem on a uniform 250×250 mesh at $t=0.45$. The colors and contours show the density. HLLC is much less diffusive.

5.1.1 Local Lax-Friedrichs and HLLC

The LLF and HLLC fluxes from Section 3.2.2 are now compared on a uniform (250×250) mesh. This will serve as the motivation for the further use of HLLC. Two simulations are set up identically except for the numerical flux function: both use van Leer limiting of the primitive variables and use a CFL limit of 0.5.

Fig. 9 shows a close-up of the solution near the origin at $t=0.45$. In the exact solution, the two jets along the axes reach the origin just before $t=0.2$ already, but the more numerical viscosity there is, the slower the jets move (See Section 5.1.3). The left diagram shows the result for LLF. The jets have formed, but are not sharp at all, just like the CD itself. The difference with the right diagram for HLLC is striking. The jets have formed and already reached the origin and are now starting to merge onto the diagonal. Not only the speed, but also the sharpness of the jet head—and the CD itself—is much sharper.

The HLLC results are significantly better. The amount of discrete time steps is almost identical, and the HLLC flux evaluations increase the total CPU time by a mere 10%. Clearly, this is well worth it. Therefore in all following experiments we will use HLLC fluxes.

5.1.2 Balancing monitor components

The main purpose of this paper is the improved monitoring of flow features. We will now consider the ability of three different monitor functions to capture the jets and various instabilities. Moreover, we will compare the *unbalanced* adaptive monitor variant (4.3) with the balanced variant (4.9) for each of these three functions.

We take a look at the bottom jet some time after the shock has hit it for the first time. A small trail of the jet was hit rightward but has now curled back up into the jet head again, see the top left diagram in Fig. 10. The other five diagrams show the results of

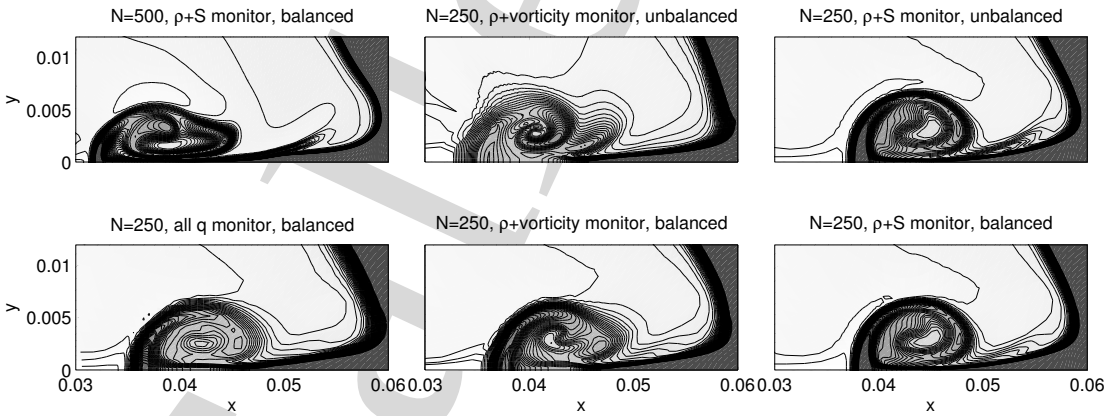

Figure 10: Detail of one of the jets in the HD22IMPDIAG problem. The top left diagram shows a high-resolution version ($N=500$, adaptive). The other five show the results for adaptive meshes ($N=250$) with several choices of monitor components, unbalanced versus unbalanced, i.e., Eq. 4.7 versus (4.8). The colors and contours depict density.

simulations on a 250×250 mesh.

The first simulation, bottom left diagram, was obtained by simply including all solution components in the balanced monitor, i.e., (4.5) and (4.9). The high-density sheet at the front of the head is properly captured, but the inner of the head is somewhat diffused. The unbalanced variant produced almost identical results, because none of the solution components severely dominates the others.

We try to improve the inner of the jet head by including the vorticity in the monitor, combined with density gradients. See also Eq. (4.10) in Section 4.3.2. However, there is only large vorticity within the back of the jet head. The mesh in the first diagram in Fig. 11 shows the strongly localized refinement in the two points with high vorticity. The top middle diagram in Fig. 10 shows how this harms the solution: a strong spiral is formed, but the outer of the jet head is not sharp at all. The balanced version in the bottom middle diagram performs a lot better: the solution features closely resemble those in the top left diagram.

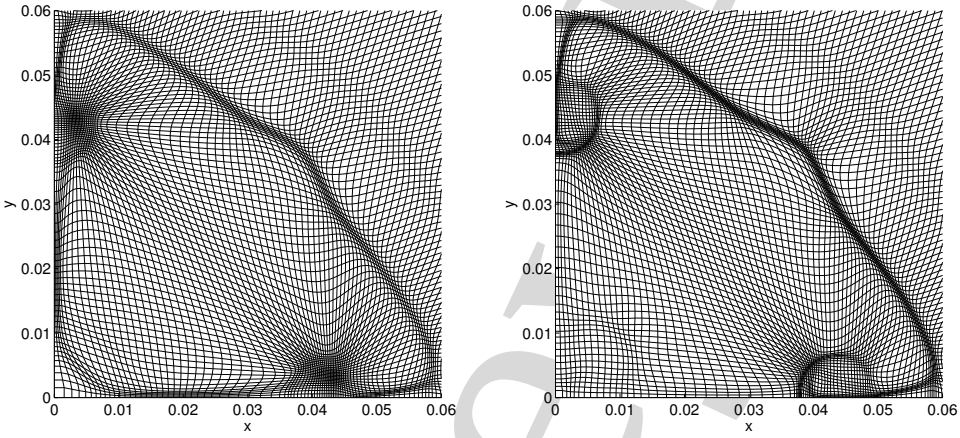

Figure 11: Monitor components: vorticity versus entropy gradients. Adaptive mesh details for the HD22IMPDIAG problem at $t=0.15$. Left: 250×250, ρ+vorticity monitor, right: 250×250, $\rho+S$ monitor.

The vorticity is now replaced by entropy gradients in the monitor, i.e.,

$$\phi_{i,1}(\mathbf{q})=\left|\frac{\partial\rho}{\partial\xi_i}\right|,\quad \phi_{i,2}(\mathbf{q})=\left|\frac{\partial S}{\partial\xi_i}\right|,\quad i\in\{1,2\}. \tag{5.1}$$

The two rightmost diagrams show the unbalanced and balanced variant, notice how they are hardly any different. Compared to the balanced vorticity result, the jet head is rounder and still very sharp. Also, the roll-up in its inner and the split-off trail at its tail is properly captured. Entropy gradients turn out to be a very good solution monitor. This is because it captures both density and pressure fluctuations.

Fig. 11 illustrates the above observations. The density-vorticity monitor refines mainly at the back of the jet head. The mesh adaptation is much better for the density-entropy-monitor, which captures all shocks, reflections and rotations. For the unbalanced monitor,

the large vorticity dominates the density gradients so much that hardly any adaptation occurs outside of the vortices, as was previously shown in Fig. 6.

In conclusion, firstly the component balancing proved effective for the vorticity monitor. Even though this was not necessary for the entropy monitor, component balancing is always our preferred method, since one can not know in advance whether it is necessary or not. Besides, as we mentioned before it is hardly anything more expensive. Secondly, the vorticity does improve results, although it is still quite localized, the combination with density is crucial. The entropy gradients proved very effective and will be used henceforth.

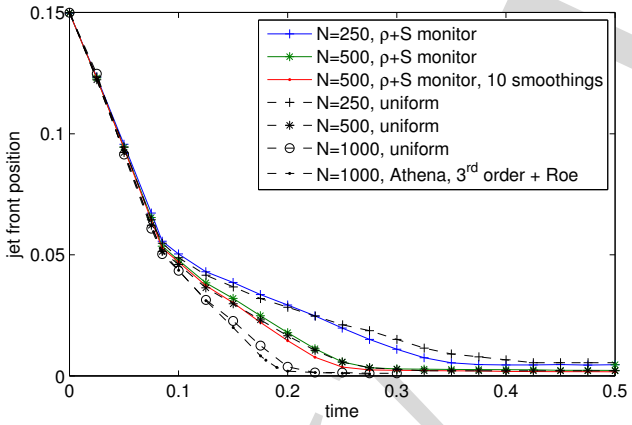

Figure 12: Position of the jet front in the HD22IMPDIAG problem for several numerical methods.

5.1.3 Details of the jet formation

We now turn to a more quantitative look at the jets. All of our simulations use van Leer limiting of the primitive solution variables and HLLC fluxes. Fig. 12 shows the position of the jet front over time for several simulations. The bottommost line is a reference solution by Athena, an astrophysical gas dynamics code [26], with third-order spatial reconstruction and Roe's Riemann solver on a uniform 1000×1000 mesh. All other lines are by our own software. The line just above the Athena line is a uniform 1000×1000 run. These runs show that the movement of the jet occurs in two stages: first it forms at the CD and starts moving with constant speed -1.17. All simulations agree with this. Then at $t\approx0.85$ the shock wave that was reflected at the origin has returned and collides with the jets. The jets lose some speed, but still continue towards the origin. The interior of the jet heads was already rotating, but after the impact the vorticity values have doubled. As a result, the head widens a little because of material that was hit backwards but now curls back up again into it.

The figure shows two more 'groups' of lines: the first group above the reference lines is around the uniform 500×500 run. The second is around the uniform 250×250 run (both have dashed lines). Clearly, an increase in numerical viscosity leads to slower jets

after the impact. Interestingly, all other waves in the solution maintain correct speed. Simulations at various mesh sizes and various amounts of adaptivity have shown this up to late in the simulation, e.g., $t=2.5$.

The solid lines represent three adaptive runs with density and entropy gradients in the monitor. What strikes is that the segments after the impact are indeed somewhat steeper than for the uniform runs, but the increase in accuracy is generally less than 10%. Possibly at the moment of impact a bigger region needs high resolution, now it is mainly the shock line and the jet contour that are refined. The $N=500$ adaptive run with increased monitor filtering (10 times) achieves this, but still is only 5% better.

The adaptive runs do achieve much better sharpness of shocks, CDs, jet heads and instabilities, though. Fig. 13 shows the $N=500$ and $N=1000$ uniform and adaptive runs at $t=0.125$. The contour lines indicate the sharper CD and jet in the adaptive runs. The jet's front and the rotation in its head in the $N=500$ adaptive run are even significantly sharper than in the $N=1000$ uniform run. The adaptive $N=500$ run took 57% more CPU time than the $N=1000$ uniform run. The mesh movement costs in terms of CPU are approximately 25%. The significantly smaller time step (due to the small mesh cell sizes in (3.23) is the major cause of the increased running time. To get equally accurate results with a uniform mesh is not feasible: it would increase the running time by several factors. For example, doubling the mesh points in both directions would give an approximately eightfold larger CPU time.

In conclusion, adaptive methods produce sharp results, both for shocks and smaller rotations. Concerning jet movement, it turns out that the wall-jetting effect is very sensitive to numerical errors, as opposed to the shocks, which have an accurate speed. Quantitative analysis is difficult and rarely seen in other research. Simulations with higher-order solvers will have to reveal a truly accurate solution.

5.1.4 Formation of instabilities

We conclude with the formation of physical instabilities in this implosion problem. The two jets have a 'negative' velocity directed towards the origin. In contrast, their surrounding area is expanding due to the reflected shock. The resulting slip lines (CDs) along the two coordinate axes, which form the tails of the jets can develop Kelvin-Helmholtz (KH) instabilities over time. Fig. 14 shows these in the small bottom left part of the domain at $t=0.5$ for simulations by three different packages. We use the Athena run on a 1000×1000 mesh with Roe solver and third-order reconstruction as a reference solution. Next, we compare our *r*-refinement with *h*-refinement in AMRVAC [23,36] for an approximately equal amount of mesh cells and identical finite volume solver (HLLC+van Leer). The top right diagram shows our moving mesh solver on a 500×500 mesh. The smallest mesh cell widths give an effective resolution of approximately 3000×3000. The bottom left diagram shows an AMRVAC run with a 100×100 mesh with 4 levels of refinement, giving an effective resolution of 800×800 and resulting in approximately 300,000 mesh cells. It is hard to speak of exact solutions in this sensitive case of instability growth, but evidently, the higher-order Athena run is considered the most accurate. Notice that

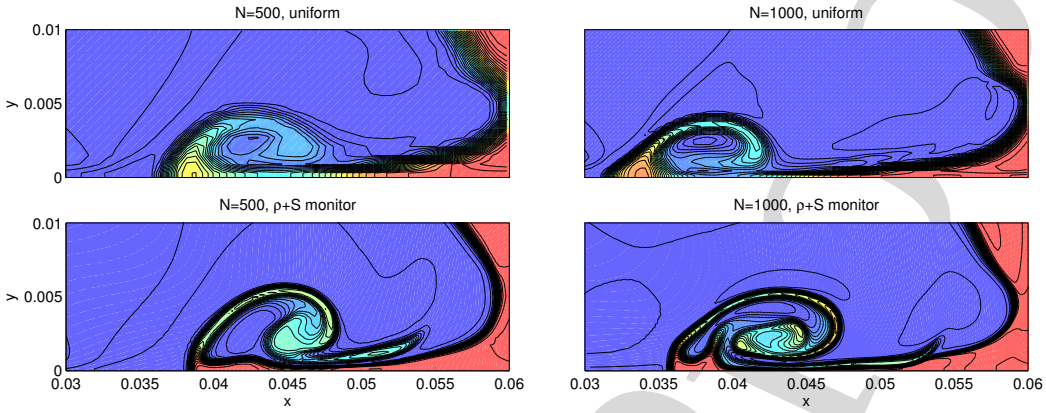

Figure 13: Detail of the horizontal jet in the HD22IMPDIAG problem at $t=0.125$. The two uniform runs ($N=500$ and $N=1000$) are both expectedly more diffusive than the $N=500$ adaptive run.

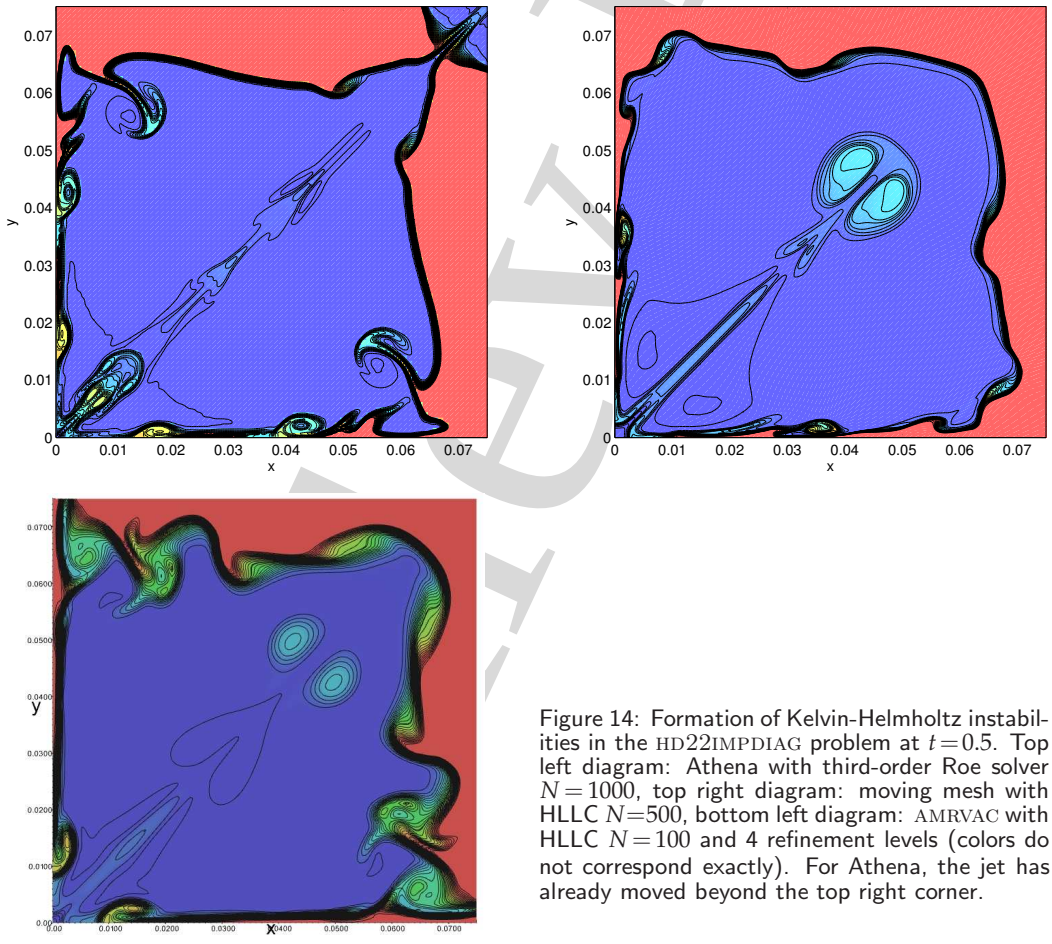

Figure 14: Formation of Kelvin-Helmholtz instabilities in the HD22IMPDIAG problem at $t=0.5$. Top left diagram: Athena with third-order Roe solver $N=1000$, top right diagram: moving mesh with HLLC $N=500$, bottom left diagram: AMRVAC with HLLC $N=100$ and 4 refinement levels (colors do not correspond exactly). For Athena, the jet has already moved beyond the top right corner.

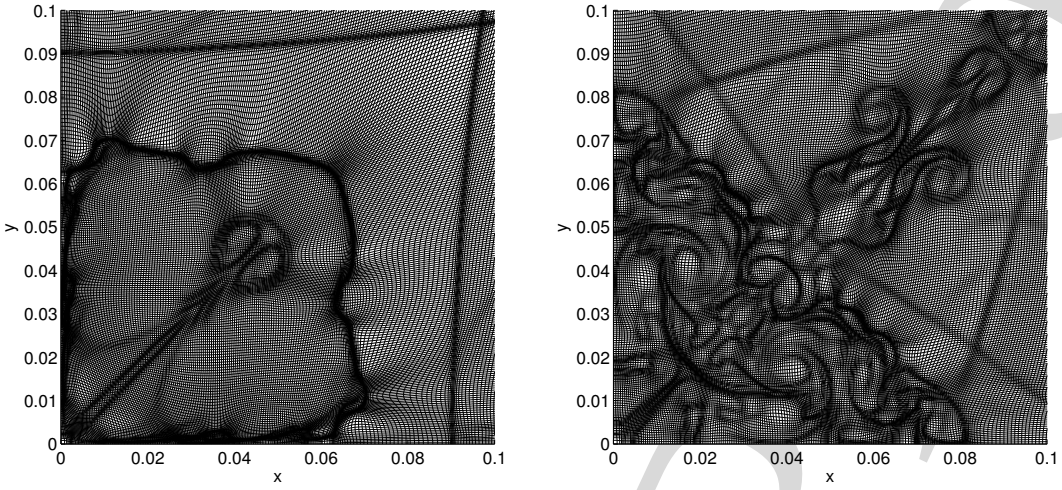

Figure 15: Moving mesh details ($N=500$) showing Richtmyer-Meshkov instabilities in the HD22IMPDIAG problem. Left diagram: $t=0.5$, right diagram: $t=2.25$.

the merged jet along the diagonal has already moved beyond the top right corner in the Athena run, but this was already discussed in Section 5.1.3. Our adaptive result in the middle diagram *does* show several KH instabilities along both slip lines very similar to the Athena solution. Since the jet has not jet crossed the CD, the CD looks different in our case. Still, the emergence of four instabilities on the CD is already visible.

The AMRVAC run has refined approximately 75% of its domain up to the finest level, giving maximal detail there. The slip lines show no KH instabilities, apparently the resolution is still too low there. On the CD, though, instabilities have formed since the beginning—even earlier than in the Athena run—and have formed big structures. We see that qualitatively the same physical phenomena show up in the three simulations, but that there is still a big quantitative difference as to *how strong* and *when* instabilities develop. This is inherent to simulation of instabilities, though. Also, honest comparisons between different packages is difficult. Our solver achieves very good refinement (up to 3000×3000 effective resolution in this example), but lacks local time stepping. AMRVAC on the other hand, has efficient time stepping, but can suffer from a very large number of mesh points when more refinement levels are allowed.

When we continue the simulation for an even longer time, the shock and its reflections will return from the top right direction and hit the CD repeatedly. This increases the vorticity on the CD. At some point, Richtmyer-Meshkov instabilities emerge from this. Note that this is a longer term process than the formation of the initial jets. Fig. 15 shows two detailed views of the adaptive mesh at an early time $t=0.5$, i.e., as in Fig. 14, and a much later time $t=2.25$. The mesh details show that both strong, isolated structures and widespread, subtle structures are captured by the mesh adaptation, without any change of parameters; both diagrams were taken from the same run.

5.2 HD22CONF11: A Riemann problem with spirals

The HD22CONF11 problem is one of the nineteen different Riemann problems classified by Lax and Liu [21]. It is set on the unit square, and in each quadrant the solution state is constant:

$$[\rho,\mathbf{v},p]=\begin{cases}[\quad 1, \quad 0.1, \quad 0, \quad 1] & \text{in the first quadrant,}\\ [0.5313, 0.8276, \quad 0, 0.4] & \text{in the second quadrant,}\\ [\quad 0.8, \quad 0.1, \quad 0, 0.4] & \text{in the third quadrant,}\\ [0.5313, \quad 0.1, 0.7276, 0.4] & \text{in the fourth quadrant.}\end{cases}$$

A backward shock will form between quadrant 2 and 1 and between 4 and 1. Between 2 and 3, and between 3 and 4 a CD will form.

In Fig. 7 this problem was used to show the effect of a directional monitor function. The directionality is crucial to properly represent the two main challenging parts to this problem. Firstly, the proper representation of the Mach stem between the CD and the shock. Secondly, the proper representation of the spiral at the end of each CD. We refer to Section 4.4 for further details.

5.3 HD22DMR: Double Mach reflection

The double Mach reflection problem (DMR) by Woodward and Colella [41] is a standard test problem that consists of a rightward moving Mach 10 shock hitting an inclined floor. We keep the domain $[0,4]\times[0,1]$ horizontal and incline the shock at an angle $\pi/3$ with the reflective bottom wall at $x=1/6$, which gives the following initial conditions:

$$\begin{aligned}[\rho,\mathbf{v},p]_{\text{post}} &= [\quad 8, 8.25\cos(-\tfrac{\pi}{6}), 8.25\sin(-\tfrac{\pi}{6}), 116.5],\\ [\rho,\mathbf{v},p]_{\text{pre}} &= [1.4, \quad 0, \quad 0, \quad 1].\end{aligned}$$

The initial post-shock, i.e., left state is prescribed by $y>\tan(\pi/3)(x-1/6)$. The left, top and bottom boundary for $x\leq 1/6$ have Dirichlet conditions with the exact shock solution. The bottom boundary for $x>1/6$ is reflective, and the right boundary has a homogeneous Neumann outflow condition.

Henderson et al. [15] have analyzed the wall-jetting effect that occurs here. The left diagram in Fig. 16 shows an interesting part of an adaptive result at $t=0.25$. The color represents pressure, and the black lines are streamlines of the self-similar flow field. A part of the flow has gone through both the initial shock and the reflected shock (i.e., the triangular part that contains, e.g., $(3,0.4)$). The other part has only gone through the Mach stem near $x\approx 3.4$. The former part has a higher kinetic energy, resulting in increased pressure in the left part of the subdomain shown. This high pressure drives the formation of a jet that connects to the slip line (CD), which is the line through $(3,0.25)$ and the triple point near $(3.4,0.55)$.

We performed the adaptive simulation on a 160×80 mesh, again using the balanced monitor function with density and entropy components. The right diagram in Fig. 16

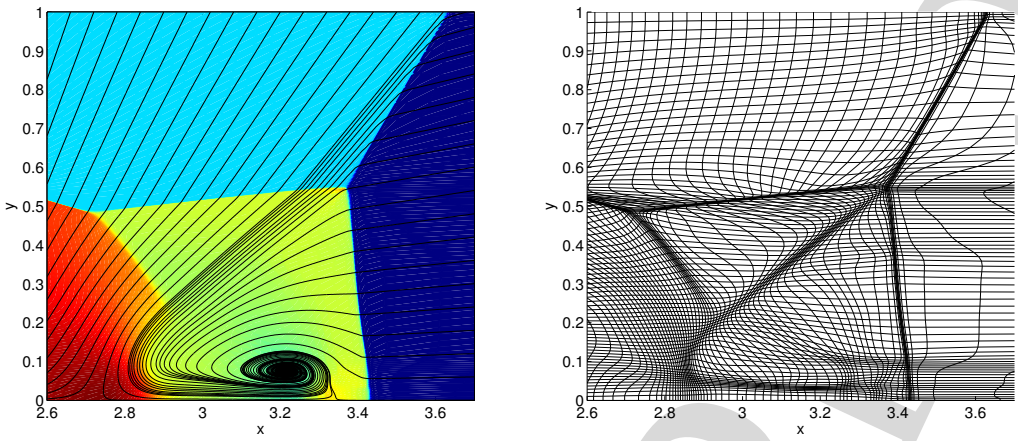

Figure 16: Left: Pressure detail for the double Mach reflection problem at t=0.25. The black lines are streamlines of the self-similar velocity field. Right: Adaptive mesh detail for the same problem. Total mesh size is 160×80 and the density-entropy monitor was used.

shows the adapted mesh in the subdomain where the double Mach reflection is present at t=0.25. The initial and reflected shocks as well as the CDs and Mach stems are properly captured. Moreover, along the jet stream and its head the mesh has also been adapted.

The vorticity is again not so useful here if we want to attract mesh points to the slip line and the jet. This is due to the triple point near $(3.4,0.55)$. The vorticity there is almost ten times larger than at the slip line: even with the balanced monitoring this will attract little points.

6 Conclusion

The main contribution of this work is the introduction of a new *balanced* monitor function. To prevent spurious solution features, the mesh should not be overly distorted, e.g., as the unbalanced vorticity monitor in Section 4.3.2 did. The new monitor balancing has negligible extra costs, results in robust mesh adaptation, and leaves the user with only one intuitive parameter: the relative percentage β of refinement.

The motive for the new balancing was the inclusion of additional physical quantities in the monitor function. Solution variables such as density have proved themselves capable of capturing the important discontinuities in many problems. However, the more subtle features such as jets and instabilities along contact discontinuities have smaller monitor values and hence receive less mesh refinement. The vorticity in flow problems adds detail to jets, for example, but still has the problem that it is extremely localized. The triple point in the double Mach reflection (DMR) problem (Section 5.3) is a good example of this. Entropy is a much more meaningful quantity. In both the DMR and the implosion problem (Section 5.1) it adds more resolution to the jets.

In an earlier state of this work we still used the local Lax-Friedrichs numerical flux,

which we now replaced by the HLLC approximate Riemann solver. The tremendous improvement in the implosion problem showed that the strength of a solver lies not only in adaptivity, but also significantly in the numerical method used.

Future work could involve the combination of our *r*-refinement with *h*-refinement. This will improve the resolution of instabilities along contact discontinuities or other interfaces. This will require major research effort, though. More feasible improvements could be local time stepping, or flux limiters such as the Woodward or Koren limiter.

Acknowledgments

The first author performs his research in the project 'Adaptive moving mesh methods for higher-dimensional nonlinear hyperbolic conservation laws', funded by the Netherlands Organisation for Scientific Research (NWO) under project number 613.002.055.

The authors wish to thank Rony Keppens from K.U. Leuven for pointing out the implosion problem (Section 5.1) and for several useful discussions on flow properties. Also, the results obtained during the first author's visits to K.U. Leuven in 2007 and 2008 formed the main cause to most improvements discussed in this paper. We also wish to thank Barry Koren from CWI for the useful discussions on approximate Riemann solvers and solution reconstruction. Finally, we thank the anonymous referees for closely reading this paper and giving several useful suggestions.

References

[1] R. W. Anderson, N. S. Elliott, and R. B. Pember. An arbitrary Lagrangian-Eulerian method with adaptive mesh refinement for the solution of the Euler equations. J. Comput. Phys., 199:598–617, Sep 2004.

[2] B. N. Azarenok and T. Tang. Second-order Godunov-type scheme for reactive flow calculations on moving meshes. J. Comput. Phys., 206(1):48–80, 2005.

[3] G. Beckett and J. A. Mackenzie. Convergence analysis of finite difference approximations on equidistributed grids to a singularly perturbed boundary value problem. Appl. Numer. Math., 35:87–109, October 2000.

[4] M. J. Berger, M. J. Aftosmis, and S. M. Murman. Analysis of slope limiters on irregular grids. Technical Report 2005-0490, AIAA Paper, May 2005. Reno, NV, Jan. 2005.

[5] J. U. Brackbill. An adaptive grid with directional control. J. Comput. Phys., 108(1):38–50, 1993.

[6] J. U. Brackbill and J. S. Saltzman. Adaptive zoning for singular problems in two dimensions. J. Comput. Phys., 46:342–368, 1982.

[7] W.-M. Cao, W.-Z. Huang, and R. D. Russell. A study of monitor functions for two-dimensional adaptive mesh generation. SIAM J. Sci. Comput., 20(6):1978–1994, 1999.

[8] H. D. Ceniceros and T. Y. Hou. An efficient dynamically adaptive mesh for potentially singular solutions. J. Comput. Phys., 172(2):609–639, 2001.

[9] P. Clément, R. Hagmeijer, and G. Sweers. On the invertibility of mappings arising in 2D grid generation problems. Numerische Mathematik, 73(1):37–51, 1996.

[10] Y.-N. Di, R. Li, and T. Tang. A general moving mesh framework in 3D and its application for simulating the mixture of multi-phase flows. Commun. Comput. Phys., 3(3):582–602, 2008.
[11] Y.-N. Di, R. Li, T. Tang, and P.-W. Zhang. Moving mesh finite element methods for the incompressible Navier-Stokes equations. SIAM J. Sci. Comput., 26(3):1036–1056, 2005.
[12] A. H. Glasser, V. D. Liseikin, and I. A. Kitaeva. Specification of monitor metrics for generating vector field-aligned numerical grids. Russian Journal of Numerical Analysis and Mathematical Modelling, 20(5):439–461, 2005.
[13] J.-Q. Han and H. Z. Tang. An adaptive moving mesh method for two-dimensional ideal magnetohydrodynamics. J. Comput. Phys., 220:791–812, Jan 2007.
[14] A. Harten, P. D. Lax, and B. Van Leer. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. SIAM Review, 25(1):35–61, 1983.
[15] L. F. Henderson, E. I. Vasilev, G. Ben-Dor, and T. Elperin. The wall-jetting effect in Mach reflection: theoretical consideration and numerical investigation. J. Fluid Mech., 479:259–286, 2003.
[16] W.-Z. Huang. Mathematical principles of anisotropic mesh adaptation. Commun. Comput. Phys., 1:276–310, 2006.
[17] W.-Z. Huang. Practical aspects of formulation and solution of moving mesh partial differential equations. J. Comput. Phys., 171(2):753–775, 2001.
[18] M. E. Hubbard. Multidimensional slope limiters for MUSCL-type finite volume schemes on unstructured grids. J. Comput. Phys., 155(1):54–74, October 1999.
[19] W. H. Hui, P. Y. Li, and Z. W. Li. A unified coordinate system for solving the two-dimensional Euler equations. J. Comput. Phys., 153:596–637, Aug 1999.
[20] J. Lang, W.-M. Cao, W.-Z. Huang, and R. D. Russell. A two-dimensional moving finite element method with local refinement based on a posteriori error estimates. Appl. Numer. Math., 46(1):75–94, 2003.
[21] P. D. Lax and X.-D. Liu. Solution of two-dimensional Riemann problems of gas dynamics by positive schemes. SIAM J. Sci. Comput., 19(2):319–340, 1998.
[22] R. Liska and B. Wendroff. Comparison of several difference schemes on 1D and 2D test problems for the Euler equations. SIAM J. Sci. Comput., 25(3):995–1017, 2003.
[23] M. Nool and R. Keppens. AMRVAC: a multidimensional grid-adaptive magnetofluid dynamics code. Comp. Meth. Appl. Math., 2:92–109, 2002.
[24] V. V. Rusanov. Calculation of intersection of non-steady shock waves with obstacles. J. Comput. Math. Phys. USSR, 1:267–279, 1961.
[25] C. W. Schulz-Rinne, J. P. Collins, and H. M. Glaz. Numerical solution of the Riemann problem for two-dimensional gas dynamics. SIAM J. Sci. Comput., 14(6):1394–1414, 1993.
[26] J. M. Stone, T. A. Gardiner, P. Teuben, J. F. Hawley, and J. B. Simon. Athena: a new code for astrophysical MHD. The Astrophysical Journal Supplement Series, 178(1):137–177, 2008.
[27] Z.-J. Tan. Adaptive moving mesh methods for two-dimensional resistive magnetohydrodynamic PDE models. Comput. Fluids, 36:758–771, May 2007.
[28] Z.-J. Tan, Z.-R. Zhang, Y.-Q. Huang, and T. Tang. Moving mesh methods with locally varying time steps. J. Comput. Phys., 200:347–367, October 2004.
[29] H. Z. Tang, T. Tang, and P.-W. Zhang. An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions. J. Comput. Phys., 188(2):543–572, 2003.
[30] H. Z. Tang and T. Tang. Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. SIAM J. Numer. Anal., 41(2):487–515, 2003.
[31] H. Z. Tang. A moving mesh method for the Euler flow calculations using a directional

monitor function. Commun. Comput. Phys., 1:656–676, 2006.
[32] T. Tang. Moving mesh methods for computational fluid dynamics. In Z.-C. Shi, Z. Chen, T. Tang, and D. Yu, editors, Recent Advances in Adaptive Computation, volume 383 of Contemporary Mathematics, pp. 185–218. American Mathematical Society, 2005.
[33] E. F. Toro, M. Spruce, and W. Speares. Restoration of the contact surface in the HLL-Riemann solver. Shock Waves, 4(1):25–34, 1994.
[34] E. F. Toro. Riemann solvers and numerical methods for fluid dynamics — A practical introduction. Springer-Verlag, 1999.
[35] A. van Dam. Go with the Flow. Moving meshes and solution monitoring for compressible flow simulation. PhD thesis, Utrecht University, Dept. of Mathematics, July 2009.
[36] B. van der Holst and R. Keppens. Hybrid block-AMR in cartesian and curvilinear coordinates: MHD applications. J. Comput. Phys., 226(1):925–946, September 2007.
[37] B. van Leer. Towards the ultimate conservative difference scheme III. Upstream-centered finite-difference schemes for ideal compressible flow. J. Comput. Phys., 23:263–275, March 1977.
[38] B. van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. J. Comput. Phys., 23:276–299, March 1977.
[39] A. van Dam and P. A. Zegeling. A robust moving mesh finite volume method applied to 1D hyperbolic conservation laws from magnetohydrodynamics. J. Comput. Phys., 216:526–546, 2006.
[40] A. M. Winslow. Adaptive-mesh zoning by the equipotential method. Technical Report UCID-19062, Lawrence Livermore Laboratory, 1981.
[41] P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. J. Comput. Phys., 54:115–173, April 1984.
[42] P. A. Zegeling. On resistive MHD models with adaptive moving meshes. J. Sci. Comput., 24(2):263–284, 2005.
[43] P. A. Zegeling. Theory and application of adaptive moving grid methods. In T. Tang and J. Xu, editors, Adaptive Computations: Theory and Algorithms, chapter 7, pp. 251–296. Science Press, Beijing, 2007.
[44] P. A. Zegeling, W. D. de Boer, and H. Z. Tang. Robust and efficient adaptive moving mesh solution of the 2-D Euler equations. In Z.-C. Shi, Z. Chen, T. Tang, and D. Yu, editors, Recent Advances in Adaptive Computation, volume 383 of Contemporary Mathematics, pp. 419–430. American Mathematical Society, 2005.
[45] Z. Zhang. Moving mesh method with conservative interpolation based on L_2-projection. Commun. Comput. Phys., 1:930–944, 2006.