Go with the Flow

Moving meshes and solution monitoring
for compressible flow simulation

Thesis committee:
Prof. dr. R. Keppens, Katholieke Universiteit Leuven
Prof. dr. ir. B. Koren, Universiteit Leiden
Prof. dr. T. Tang, Hong Kong Baptist University
Prof. dr. ir. J. J. W. van der Vegt, Universiteit Twente
Prof. dr. J. G. Verwer, Universiteit van Amsterdam

Cover: The cover design was inspired by Portishead's magnificent 'comeback' album '*Third*'. One might argue the tracks form a flow as diverse and occasionally abruptly changing as gas flows, but actually it's just plain brilliant. The back cover shows a Schlieren plot of the density in the hydrodynamical 'implosion problem' from Section 4.5.1 at $t = 2$. The front cover also shows an early mesh detail of the jet formation at $t = 0.14$.

This research was financially supported by the Netherlands Organisation for Scientific Research (NWO) project 613.002.055, Adaptive Moving Mesh Methods for Higher-dimensional Nonlinear Hyperbolic Conservation Laws.

Netherlands Organisation for Scientific Research

ISBN 978-90-393-5077-5

Go with the Flow

Moving meshes and solution monitoring for compressible flow simulation

Met alle Winden mee

Bewegende roosters en oplossingsmonitoring voor compressibele stromingssimulatie

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof. dr. J. C. Stoof, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 1 juli 2009 des middags te 2.30 uur

door

Arthur van Dam

geboren op 20 september 1979, te Noordeloos

Promotor: Prof. dr. H. A. van der Vorst
Co-promotor: Dr. P. A. Zegeling

So, so you think you can tell
Heaven from Hell,
blue skies from pain.
Can you tell a green field
from a cold steel rail?
A smile from a veil?
Do you think you can tell?

— Pink Floyd, Wish You Were Here

Preface

This dissertation is the result of 4.5 years of Ph.D. research on the use of moving mesh methods for flow simulations. My first encounter with the subject was in 2001 during a seminar on mesh and graph computation by Rob Bisseling and Paul Zegeling at the Mathematical Institute of Utrecht University. That same year I visited several companies in search for an internship as a graduation project, and ended up at TNO Traffic and Transport in Delft. Highway traffic flow can be modeled similar to gas flow, and moving meshes can be used to obtain better simulation results near the head and tail of traffic jams [vD02]. I still remember how I visited my supervisor Paul Zegeling after I graduated. He showed me the equations of one-dimensional magnetohydrodynamics and without preparation, we got all equations into my developed MATLAB software within ten minutes and the moving mesh algorithm produced promising results right away. The reuse of software for a whole new physical model was very intriguing to me.

Two years later, Paul started a project just on this topic, and I gladly accepted the offered Ph.D. student position. The original project proposal mentions the intention "*to combine analytical and numerical insights in the dynamics generated by the adaptive mesh equations*". Also, "*the main goal and real challenge [..] is to analyze and develop a stable and effective adaptive mesh method [..] generally applicable in many other scientific areas*". Finally, possible new directions for mesh adaptation are suggested: "*geometric conservation laws, harmonic maps and the deformation method*". The second point has gradually taken the main place in this project, together with the 'numerical insights' of the first point. The more example problems I applied my moving mesh solver to, the more feeling I got for what is important and what not in time-dependent mesh adaptation. The result is a solver in Fortran 95 that handles the full class of hyperbolic problems that was originally intended. I have not made the extension to three dimensions. Others did so, but always with adaptation criteria manually tuned for the problems considered. Although this is good for advancing research, I found improving the robustness of the method in two dimensions more important. The adaptation criteria are now automatically regulated, which gives much more often good results immediately in the first run of a new problem. I studied discrete geometric conservation laws in 2007 (see also [vDZ07]), but I preferred keeping the mesh moving and PDE solving parts separated. I studied the above three moving mesh methods amongst many others in Chapter 2. The theoretical foundation of the deformation method is more firm than the direct variable diffusion, which I use (deformation guarantees equidistribution in spaces of any dimension). This is mainly beneficial for unsteady test problems, though; for complicated time-dependent flow problems, the resulting meshes are often too stretched.

I encourage a follow-up on this project that extends the moving mesh

algorithm and monitor function balancing to three dimensions. The risk of mesh collapsing seems imminent there, but again, the smooth balancing of the monitor function may provide good robustness in practice. If not, the deformation method or alternative techniques should be investigated. The monitor function balancing that I developed remains equally useful there, and extends easily to higher dimensions.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my co-promotor Paul Zegeling. Paul, you always gave me great freedom in doing my research. Your knowledge of our field is impressive and your suggestions for further reading proved valuable in many cases. I remember numerous meetings in your office, between ever-changing-never-disappearing piles of paper, where you managed to find any publication or hand-written notes within the minute. Our relation was quite informal, given that we rode Disney's Tower of Terror, made jeep tours through Australia's rainforests ("Always mind the drop bears!") and attended a wonderful Pink Floyd (tribute) performance. I value these experiences.

My promotor Henk van der Vorst must have had more time-consuming projects than mine. Still, Henk, I appreciate your input during the few meetings that we had. In the beginning, you relativized Academia, which made me value my first results more. At the end, you were uprightly interested in the final results. Your comments were to the point and helped me further improve the texts in this dissertation.

Next, I would like to thank the members of the reading committee, prof. Koren, prof. Van der Vegt, prof. Keppens, prof. Tang and prof. Verwer, for the time they invested in reading the manuscript of this dissertation and commenting on it. The five of you represent what I like in this work: cross-field research that unites physics, PDE solvers and numerical mathematics.

Rony, you deserve a special word of thanks. In the first year already, I occasionally visited you in the Rijnhuizen castle, discussing my results on magnetohydrodynamics. After your move to Leuven, you facilitated two one-month research visits to your group for me. These are amongst the best periods in my Ph.D. project. Not only the change of scenery made me extremely productive, but also how you formulated your expectations and set priorities really sped things up.

Over the years, I met numerous people at conferences, or when they were visiting Utrecht. In particular I had helpful discussions with Tao Tang, Mike Baines (who advised the so-successful HLLC solver in Chapter 4 four years ago already), Barry Koren and Jens Lang. Manuel Torrilhon and Huazhong Tang were also helpful when I was working on the article version of Chapter 3.

Next, dear reader, allow me to thank my colleagues and friends in somewhat greater detail. I may have produced the discipline to do all work myself, but the good atmosphere within the group and the pleasure of life outside working hours surely made this easier.

Tammo Jan, despite your complete lack of taste for good rock music, we still managed to get along pretty well for all those years.[1] I liked discussing and criticizing our research, thereby improving readability of our work. And also our unspoken matches in who would be the first to solve a colleague's computer problem involving LaTeX, Mathematica or somewhat lesser fun topics. Best example of your helpfulness is when you went and rescued my laptop when I had 'lost' it somewhere along the way from Amsterdam to Moscow, thanks again.

Buurmannetje, a.k.a. J.R., quiet as you seem, you turned out to be one of the most sincere and enthusiastic colleagues. Your passion for anything English was truly amusing at times, not in the least your imitations of Vinnie Jones and Winston Churchill. Try to cut down a bit on that Prince Philip adoration, though. I greatly enjoyed the many *cuppas* you brought me, the Longjing and Bi Luo Chun in particular. I know you regret being in Montreal at the time of my defense, but we'll make up for that soon.

Job, stubborn at first sight, but you were always interested in the physics part of my research. In that sense, you sometimes even seemed to be my second co-promotor, nagging me for 'nieuwe filmpjes', day after day. Sebastiaan, the same holds for you. Your blunt attitude towards anything established was always amusing, but actually you're really interested in many others' work.

Jan Willem, we share nothing mathematical, except for Riemannian manifolds maybe. But I found a great cycling friend in you. With me leaving the institute, our cycling tours won't stop, if only to prevent you from getting lost... No doubt, we'll complete the five grand tours across the country this summer.

As a big fan of extra-institutional activities, I greatly appreciated our weekly Friday morning Gutenberg coffee meetings. I urge the current 'harde kern' (Charlene, Vincent, Jan Willem, Job, Bart, Janne, Wouter, Jan Jitse, Jaap, Jeroen and Bas) to continue this tradition. Our Rome trip and the gaming nights at Charlene's place were also great fun.

"*Vroeger was alles beter*", as a Dutch saying goes, is not true. Still, I want to mention former members of the coffee club, Taoufik, Arno and Hil, who gave the early mornings a pleasant start (well, not Arno, but still). Also thanks to my former roommates, Erik and Sander.

The numerical group in our department is a small but involved group. Especially the numerical colloquium meetings gave me more and more the feeling of being part of the team. Also, the cooperation with Albert-Jan on the Laboratory Class was pleasant. Make sure the 'numwcol' continues regularly! Gerard, Paul and Rob, I hope someday soon, more funding comes available so that our now bizarrely small group can expand to the size it once had.

Despite my self-reliance in computer facilities, I also needed help desk assistance every now and then. I want to thank André for that, but in particular Frank. In a time where everything is handled centrally, you were incredibly fast in providing me with new hard disks, new compiler versions, port forwards and what not. Thank you very much.

[1] With thanks to Sennheiser electronic GmbH & Co. KG.

I also wish to thank all the institute's secretaries throughout the years. In particular Pet, Els (for the countless room reservations for our colloquium, and your loud singing), and Helga (you always seem to solve problems before they even occur).

The PhDays I attended all years and co-organized in 2006 and 2007 were not only fun, but actually proved valuable during my later visits to Leuven. Liesbeth, Liesbet and Yves, thank you for your hospitality! Thanks also to the other organizers, Tammo Jan, Jok, Sander, Hendrik and Ward.

Outside of the institute, outside of mathematics at all, I was pleasantly distracted on a daily basis by the people on the IRC channel whose name must not be mentioned. Martin, of course I despise your flight to the States, thereby depriving me of any opportunity to beat you on the Amerongse Berg. Still, I remember our 'monstertocht' cycling tours from Leeuwarden, Maastricht and Zeeland as great adventures. Next, there's the storytellers. Dick, the master himself, try not to work so hard, we miss your stories! Arie, your unworldly stories are sometimes just plain funny; keep 'em coming. Eelco, the erudite type, I'd rather bike with you than try to win a debate; when shall we hit some dunes? Armijn, I'll just text you: "SCHDDH! BLLH!".

Rob and Lizi, last year's West Coast trip with you was one of my best holidays ever. It was the perfect rest before the final half year of thesis writing. Lizi, your ambition, and Rob, your relativism make you a nice couple to hang out with. Let's continue our weekly Saturday morning Cortados and occasional concerts for a long time still.

Arjan, paranimf together with Rob, we go back a long time. We started out as teenagers driving fork-lift trucks, drank beers after nights of Visual Basic hacking, and started making careers. All this time, you were honestly interested in my daily pursuits, actually remembering the next time what was keeping me busy before (many people don't). I valued our numerous 'Monday mails' and also our discussions, which helped me explain my research to non-mathematicians even better.

To the friends that I did not mention here, I apologize! My four pages are full. I'll cook you a nice dinner to make up for it.

Finally, tot slot, mijn familie. Alle Mollema's en Van Dammen, dank voor de interesse en alle leuke activiteiten door de jaren heen. Barbera en Timo, en de lieve, kleine Lotte; was het al gezellig om op visite te komen bij jullie, dat is dit jaar alleen maar beter geworden! Ik heb diep respect voor jullie, en wens jullie een geweldige tijd toe.

En mijn ouders, pap en mam. Jullie hebben me altijd in staat gesteld om me te ontwikkelen, op school, in werk en daarbuiten. Zonder iets op te dringen, want ik mocht altijd op mijn eigen keuze vertrouwen. En ook al was het vaak abracadabra wat ik in Utrecht uitspookte, jullie bleven op de hoogte van mijn bezigheden, tot in vaktermen zelfs. Deze steun en interesse is wellicht normaal voor alle ouders, maar dat maakt het niet minder plezierig. Dank jullie wel!

Arthur van Dam
Utrecht, June 2009

Contents

Introduction

1

THESIS

The true power of a solver for compressible flow problems that employs adaptive moving meshes lies not only in the use of higher-order numerical techniques, but also in the elimination of redundant user-defined and problem-dependent parameters.

1.1 MOVING MESH RESEARCH

Adaptive moving mesh research emerged from the field of mesh[1] generation in the 1970s and has evolved into a research area of its own. *Mesh generation* is about generation of domain discretizations on complicated—e.g., nonconvex—domains and in or around complicated geometries in computer aided design (CAD), such as airfoils and propellers. *Adaptive mesh movement* adapts a given mesh to features of solutions that live on the domain. It starts with a reference mesh and moves the mesh nodes towards regions where a finer discretization is required for increased accuracy. Mesh movement is also used for free surface flow problems and moving boundary problems. The mesh adaptation can be time-dependent to continuously adapt, e.g., to unsteady flow features.

Origins

Earlier works on analysis of partial differential equations (PDEs) can be seen as the forerunners of adaptive meshes. Karl F. Sundman, for example, introduces a new scale-invariant time variable in his '*Mémoire sur le problème des trois corps*' in 1912. N.A. Philips proposes a meteorological model with vertical coordinate scaling with reference to atmospheric pressure at ground level such that both sea level and mountains all lie at the same coordinate isosurface, in '*A coordinate system having some special advantages for numerical forecasting*' in 1956.

The earliest work on adaptive meshes involved an explicit coordinate transformation, for example radially expanding mesh cell sizes in hurricane simulations [Ant70]. Gradually, this shifted to implicit specification of the nonuniform coordinates. This specification is generally done through strictly positive weights to attract mesh points and penalties to guard mesh qualities, such as smoothness and orthogonality. Usually, an elliptic generator PDE then prescribes the actual mesh.

[1] Throughout this dissertation, the word 'mesh' can always be read as 'grid'.

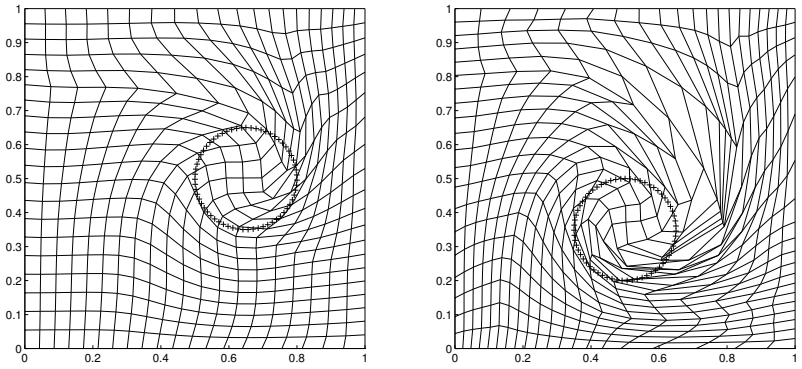

Figure 1.1 Velocity-based mesh movement bears a high risk of mesh skewness and degeneracy. Moving finite element results for a 2D circular advection problem at $t = 0.5$ and $t = 1$; the mesh follows a pulse (marked by +'s).

Theoretical foundations

In the 1980s and early '90s, typical research was fundamental. Due to limited computing power, the adaptivity was often prescribed as a function of the location, i.e., no solution adaptivity, nor a physical PDE that needed to be solved on the resulting mesh. The various forms of mesh generator equations are still in use, though. Winslow's variable diffusion method (Section 2.3.2.3) and the general variational approach (Section 2.3.2.4) form the basis of many of today's methods.

Winslow's variable diffusion method [Win81] worked well, but needed theoretical foundations too. This mainly concerns the *existence* of a mesh map and its *invertibility*. The latter prevents mesh points to collapse or mesh cells to 'fold' and become degenerate. This is equivalent to the positivity of the map's Jacobian determinant.

The theory of harmonic mappings has been well-developed since the 1950s and guarantees existence and invertibility of the mesh map, given a convex target domain. Dvinsky [Dvi91] was the first to use harmonic mappings in mesh adaptation. Although well-defined, the resulting meshes can have extremely large mesh cells in areas of little adaptation, close to very small mesh cells. This makes harmonic mappings mainly suitable for stationary or prescribed solutions. In unsteady problems, the mesh cells keep changing and such large stretching factors between neighboring cells produce large errors in solution interpolation and flux evaluation. Other approaches are better suited for such time-dependent problems.

Another important aspect in mesh adaptation is the widely appreciated equidistribution principle in one dimension due to De Boor [dB74]. The distribution of local error measures equally over the domain by modifying cell sizes is evidently a good technique for increasing overall accuracy. This principle does not extend to multiple dimensions for several methods, though, so researchers have sought for new methods that do ensure equidistribu-

tion in higher dimensions. Liao's deformation method [LLdlP02] maintains equidistribution by definition, as well as approaches based on the geometric conservation law, e.g, by Cao et al. [CHR02] and Baines et al. [BHJM09]. Recently, Monge–Kantorovich optimization was employed for mesh adaptation with guaranteed equidistribution, by Budd et al. [BW06] and Delzanno et al. [DCF+08].

Adaptive meshes may still be far from optimal, despite their theoretical foundations. Firstly, the invertibility is guaranteed for the *analytic* mesh map, whereas the actual mesh is a *discrete representation* of the map, and mesh nodes might collapse anyway. Robust solvers for the nonlinear mesh equations are necessary or the mesh adaptation should be smooth enough to prevent degeneracy of the mesh. Secondly, an invertible mesh map may still yield mediocre cells, for example heavily rotated or very skew cells. As an illustration, consider the diagrams in Figure 1.1. They show an adaptive mesh obtained by moving finite elements (Section 2.3.3.2) for a pulse wave that is advected over a circular path. The mesh points constantly follow this pulse and as a result the mesh cells are pulled into increasingly bad shape. This is a common risk of velocity-based approaches (Section 2.3.3). Location-based approaches (Section 2.3.2) allow mesh cells to 'let go' of passing flow phenomena, after which neighboring cells take over the refinement.

Practical applications and robustness

Over the past two decades, the vastly increased computing power facilitated more complicated applications, such as unsteady flow. As the flow features can change substantially over time, proper solution adaptivity becomes increasingly difficult. Researchers have attempted to prevent overly refined or underrefined meshes by carefully introducing and tuning parameters that affect the mesh adaptation.

The strive for a fully automatic—i.e., parameter-free—and all-knowing mesh adaptation process is immoderate. Mesh adaptation is a supportive technique for solving the actual physical problems, where the scientist is still the expert. Sometimes, an engineer is not so much interested in extreme sharpness of shocks, but rather in the emergence of unstable vortices in seemingly quiet areas. Or a physicist who knows that the curl of the magnetic field will be a good indicator to detect current sheets. However, the automation of mesh adaptation *does* have an important goal in the elimination of parameters that are too ad hoc to be clear to inexperienced users of mesh adaptation. Certain parameters *do* make sense and should not be excluded, for example the balance between adaptation to strong and subtle phenomena, and the relative amount of adaptation at all. It is crucial to make these parameters intuitive and problem-independent by an appropriate and automatic scaling. Without such problem-independent parameters, the amount of time spent on reruns and manual parameter fine-tuning could just as well have been spent on nonadaptive simulations.

We consider the colliding blast wave problem by Woodward and Colella [WC84] in one-dimensional gas dynamics as an example. Two different shock

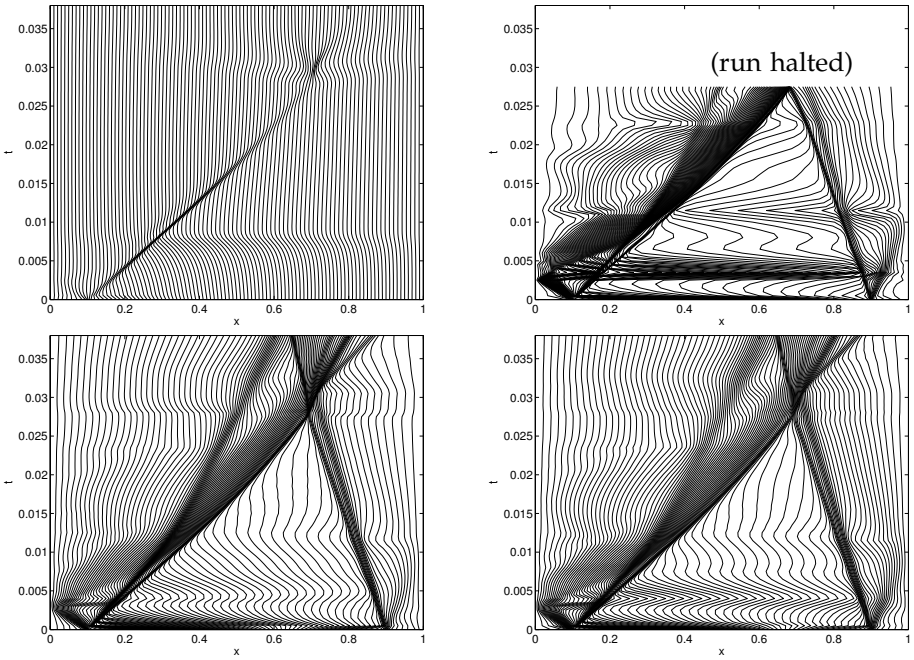

Figure 1.2 Bad choices for adaptation parameters waste time. Mesh point trajectories for the one-dimensional colliding blast wave problem. Top row: arc length monitor. Left diagram: q_x gradient, $\alpha = 8 \cdot 10^{-10}$: the left shock dominates all mesh refinement. Right diagram: q_ξ gradient, $\alpha = 1.5 \cdot 10^{-4}$: upon impact of the two shocks the run halts. Bottom row: the time-dependent balanced monitor. Left diagram: q_x gradient, $\beta = 80\%$. Right diagram: q_ξ gradient, $\beta = 80\%$. The adaptation is automatically balanced.

waves at $x = 0.1$ and $x = 0.9$ at the initial time move towards each other between reflective walls. Figure 1.2 shows the mesh point trajectories of four runs with different adaptivity settings. The top row uses a standard arc length monitor which scales solution gradients with a manually set constant parameter α:

$$\omega(q) = \sqrt{1 + \alpha |\mathrm{d}q/\mathrm{d}x|^2}.$$

Too low values give too little refinement, whereas too large values cause a crash at $t \approx 0.027$. The crash is due to a large stretching ratio for certain neighboring cells, after which the solution reconstruction yields a negative gas pressure. All computing time spent thus far is wasted. The left and right diagrams use a different type of solution gradient, which requires a totally different value for α ($O(10^{-10})$ instead of the former $O(10^{-4})$), which is found by trial and error. Other values than these would cause even earlier crashes or underresolution. The bottom row uses an automatic and time-dependent balancing of the monitor function (see Section 3.3.3). The intuitive parameter β represents the relative amount of refinement. It produces similarly smooth

results for both runs without changing its value, whereas the two arc length results were fundamentally different. Chapter 3 elaborates on the monitor balancing in one dimension, which is the crucial point in this example.

A crucial, but easily satisfied requirement is that the overall computing time for an adaptive method should be less than for the same method on a uniform mesh. In other words: the additional costs for the mesh adaptation should be (significantly) less than the saved computing time and gained accuracy. Typically, our mesh adaptation takes 25% of the total CPU time spent, whereas the effective local resolution can increase by a factor of several tens.

1.2 THE STATE OF THE ART

The research on adaptive moving mesh methods in the past decade shows a division in two directions. One is the fundamental analysis of the optimality of adaptive meshes. This involves analysis of the local errors of the physical PDE solver at hand and subsequent inclusion of these (a priori or a posteriori) error measures into the mesh adaptation criteria. The combination with finite element solvers is evident; in finite volume solvers, error analysis for nonlinear PDEs is notoriously difficult. Rigorous analysis of mesh quality is therefore often done for prescribed example solutions, without an underlying physical PDE problem.

The other direction is aimed at increasingly challenging physical applications. Mesh refinement at a shock wave poses no problem even with today's more basic methods. Doing so over long periods of time, however, with the continuous emergence of new flow features such as shocks, local physical instabilities and vortices is significantly more difficult. Due to the increased computing power, we can now easily reach spatial scales at which the latter local phenomena become visible. Consequently, the adaptation criteria need to focus on a range of solution features.

The work in this dissertation is mainly in the second direction, but also leans towards the theoretical by guaranteeing a balance in the amount of adaptation for all times. We present a new method that automatically prevents imbalance between multiple monitor components (adaptation criteria). This results in a robustness that allows for very long time integrations in which the mesh continuous to adapt to evolving flow features. We demonstrate this on a range of challenging examples from hydrodynamics and magnetohydrodynamics in one and two dimensions. The detection of small-scale flow features proves the quality of the adaptive method. We will now present some of the most relevant work done by others in both directions.

1.2.1 *Innovations in analysis*

The lion's share of recent theoretical analysis of the monitor matrices that define the criteria for mesh adaptation is due to Weizhang Huang. He begins the analysis of functionals for variational mesh adaptation in a two-part publication [Hua01b, HS03]. The first part concerns the combination of two functionals for equidistribution and isotropy, where isotropy aims to keep

mesh cells close to their original equilateral shape. The second part uses analysis of the linear interpolation errors to define the adaptation criteria. Subsequent work on the mathematical principles behind anisotropic mesh adaptation is summarized in [Hua06].

A new development is the use of Monge–Kantorovich optimization for mesh adaptation by Budd and Williams [BW06] and Delzanno et al. [DCF+08]. This approach takes exact equidistribution as a constraint, and then minimizes a measure of mesh distortion. Such strict equidistribution is mainly useful for blow-up solutions, but the accompanying robust Newton–Krylov solver may well prove valuable for different applications as well. It achieves fast convergence towards the solution to the nonlinear mesh equations by preconditioning and is even more efficient by using a multigrid preconditioner and avoiding Jacobian evaluations.

1.2.2 *Expanding physical applications*

Tang and Tang [TT03] were the first to really put the direct method proposed by Ceniceros and Hou [CH01] to work. They devised a conservative solution interpolation, which is crucial for the solution of hyperbolic PDEs from conservation laws. Tang and coworkers employ variants of the standard arc length monitor with manually chosen parameters for diverse problems: Euler gas flow [Tan06], Hamilton–Jacobi equations [TTZ03] and ideal magnetohydrodynamics [HT07]. The preceding work involves finite volume solvers, but the mesh adaptation can also be combined with a finite element solver, e.g., for the incompressible Navier–Stokes equations [DLTZ05], which is also extended to two-phase flows [DLTZ07] and multi-phase flows in three dimensions [DLT08]. In the second application the curvature of a level set function is added to the monitor function to refine at the interface.

Mackenzie and coworkers use the MMPDE approach (Section 2.3.2.7) for mesh adaptation. Their results for the Hamilton–Jacobi equations [MN07] are comparable to Tang's aforementioned results. Beckett and Mackenzie have also proposed a time-dependent monitor parameter [BM00]. They apply this to the same phase-field problems [BMR06] as Tan et al. [TTZ06] did. They obtain similar results, but without the need for choosing appropriate monitor parameters. Recently, Zegeling et al. [ZLI] used a similar monitor to detect Liesegang patterns in a reaction-diffusion system coupled with a fourth-order PDE.

The methods presented in this dissertation were partly inspired by the aforementioned monitor parameter. We use it on gas dynamics and ideal magnetohydrodynamics, both in one (Chapter 3) and two dimensions (Chapters 4 and 5). We improve the monitor balancing such that it properly combines multiple components with different singularity characteristics into the monitor function. This has led to impressive mesh adaptation, which occasionally captures unanticipated physical phenomena.

1.2.3 *Alternative techniques*

Adaptive *local* mesh refinement—also known as h-refinement or AMR—has gotten more attention than moving mesh refinement over the past decades. This technique recursively splits mesh cells into finer cells or coarsens them where necessary. The implementation is more complicated, as the nested levels of refined mesh cells require proper bookkeeping, especially in the evaluation of inter-cell fluxes. Mathematically, though, this technique is more straightforward. Given an initial, level-0 mesh, all refined mesh cells have the same shape as their parents. Guarding mesh quality, e.g., smoothness or skewness is not an issue, but this lack of freedom also means that alignment with flow structures is impossible. AMR is an extremely powerful method nonetheless, as tremendous effective resolutions can be obtained very locally.

AMRVAC [vdHK07, vdHKM08] is a parallel block-AMR finite volume solver for systems of conservation laws. It supports curvilinear coordinates that are fixed in time. Aimed at astrophysical models, equipped with high-order approximate Riemann solvers and running on clusters, it delivers unprecedented detail in astrophysical simulations, see, e.g., Keppens et al. [KMvdHC08].

A recent variant of plain h-refinement is the multi-mesh approach to h-refinement, proposed by Li [Li05]. The idea is to perform the local refinement and coarsening steps for each component separately when they have significantly different singularity behavior. This does not increase the overall effective resolution, but saves some computing time.

Van der Vegt and coworkers have developed a space–time discontinuous Galerkin solver, which considers the space–time continuum as the 4-dimensional domain. They apply this to domains with moving boundaries, e.g., an oscillating airfoil [vdVvdV02]. The moving mesh does not adapt to solution features; this is done by local h-refinement.

Morrell et al. [MSB07] include slightly more mesh movement into their anisotropic h-adaptive arbitrary Lagrangian Eulerian scheme. The mesh movement is governed by the flow velocity, but it is relaxed to such an extent, that most adaptivity still comes from the the local h-refinement.

The truly hybrid combination of h- and r-refinement has been studied in only a few publications. Lang et al. [LCHR03] present a finite element solver that uses mesh movement by MMPDEs (Section 2.3.2.7) and local refinement based on a posteriori error measures. We feel that hr-refinement can truly deliver increased accuracy for problems with curved flow features, e.g., circular fronts, that extend from one side of the domain to the opposite side. h-Refinement alone will not be able to align with the curved front, and r-refinement will properly refine *across* the shock front, but not *along* it, due to the lack of points in the tangential direction.

1.3 COMPRESSIBLE FLOW APPLICATIONS

In this dissertation, the moving mesh process is essentially *decoupled* from the physical PDE solver. Hence, the mesh adaptation requires no assumptions about the actual physical PDEs that are to be solved.

Various applications have been solved by others using adaptive methods, such as reaction-diffusion problems [Lan98], phase-change problems [BMR06], and incompressible (Navier–Stokes) flow [DLTZ05].

We employ our mesh adaptation algorithm to solve compressible flow problems, or nonlinear hyperbolic conservation laws in general:

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathcal{F}(\mathbf{q}) = \mathbf{0}, \tag{1.1}$$

where the $p \times n$ flux tensor $\mathcal{F}$ defines the fluxes for the p solution components in n dimensions. We consider one- and two-dimensional domains.

One of the earliest test problems in the research on one-dimensional mesh adaptation is the inviscid Burgers' equation:

$$\frac{\partial q}{\partial t} + q q_x = 0.$$

This scalar equation fits in the hyperbolic conservative form by defining:

$$\mathbf{q} = q, \quad \mathcal{F}(q) = \frac{1}{2} q^2.$$

Starting with an initial sine wave, e.g., $q(x, 0) = 0.5 + \sin(x)$, the solution develops a shock after some time. This problem is hardly a challenge for a moving mesh method, since only one typical shock wave is formed, within a short time, after which it moves along with constant speed. More challenging applications contain multiple phenomena that change substantially over time.

The multidimensional ideal magnetohydrodynamics (MHD) equations fit in the same nonlinear hyperbolic form:

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho \mathbf{v} \\ \mathbf{B} \\ E \end{bmatrix}, \quad \mathcal{F}(\mathbf{q}) = \begin{bmatrix} \rho u & \rho v & \dots \\ \rho u^2 + p_{\text{tot}} - B_1^2 & \rho v u - B_2 B_1 & \dots \\ \rho u v - B_1 B_2 & \rho v^2 + p_{\text{tot}} - B_2^2 & \dots \\ \vdots & \vdots & \vdots \\ 0 & v B_1 - B_2 u & \dots \\ u B_2 - B_1 v & 0 & \dots \\ \vdots & \vdots & \vdots \\ (E + p_{\text{tot}}) u - B_1 \mathbf{B} \cdot \mathbf{v} & (E + p_{\text{tot}}) v - B_2 \mathbf{B} \cdot \mathbf{v} & \dots \end{bmatrix},$$

where the dots hide all third-dimension terms.

Both our algorithms and the software implementations of them are generic for systems of conservation laws (1.1) in two dimensions. There are several concise program modules that define the physics, e.g., for scalar advection, gas dynamics (Euler) and ideal MHD. The physical quantities are conserved up to machine precision.

Sometimes, physical models lead to additional requirements. The Navier–Stokes model for incompressible flow assumes a divergence-free velocity field ($\nabla \cdot \mathbf{v} = 0$). This requires additional measures in the algorithm. Also, for multidimensional MHD, the magnetic field should be divergence-free. We show how to ensure this in Chapter 5.

1.4 OUTLINE

In Chapter 2 we give an extensive overview of the field of mesh refinement, moving mesh adaptation in particular. This involves the history and evolution of the methods and—more importantly—the relations between them. Also, we introduce the underlying mathematical concepts and the notation, which will be used throughout this dissertation. It is not only a necessary introduction to the next chapters, but also a complete and up-to-date review of moving mesh methods, worth reading on its own. The succeeding chapters contain our own work and may have some overlap with this chapter in their theoretical parts, but we expect that this improves readability.

In Chapter 3 we apply one-dimensional moving mesh adaptation to ideal magnetohydrodynamics. We present an algorithm that combines mesh movement and a higher-order finite volume solver. Main features of this are the self-regulating monitor function for controlling mesh movement and a detailed study of interesting physical phenomena, such as regular versus critical solutions, and physical staircasing. These would have been very hard to obtain on uniform meshes. Finally, we present some benchmark results, also in comparison with local h-refinement. This chapter has been published as [vDZ06]:

> 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.

In Chapter 4 we present the full two-dimensional solver. The main contribution of this chapter is the study of an even better balancing of monitor components to capture various flow phenomena. This is motivated by the fact that for *one* given monitor component, the former monitor function is automatically balanced, but the difference in locality of *multiple* components might cause one component to dominate all the others. This can be elegantly remedied by a rethought normalization. This type of monitor balancing is new and greatly improves adaptivity at only minimal extra costs. Besides, the solver is enhanced with an HLLC approximate Riemann solver and we show the details of slope limiting and upwinding on nonuniform meshes. The solver is tested on problems from hydrodynamics. We show a study of jet formation and emergence of instabilities in an implosion problem with unprecedented detail. This chapter will appear as [vDZ09]:

> A. van Dam and P.A. Zegeling. *Balanced monitoring of flow phenomena in moving mesh methods.* To appear in Commun. Comput. Phys. DOI:10.4208/cicp.2009.09.033, 2009.

In Chapter 5 we extend the two-dimensional solver to ideal magnetohydrodynamics in two dimensions. A crucial aspect is the preservation of the divergence-free magnetic field through a vector potential approach. An additional difficulty is posed by the interpolation and numerical integration of this vector potential on the adaptive mesh. The solver is tested on several

MHD problems, including the formation of current sheets in the well-known Orzsag–Tang test problem.

In the appendices we discuss some of the enhancements that we appended to our solver. They are of a pragmatic nature, but nevertheless nontrivial.

An Overview of Mesh Adaptivity

2

This chapter contains an extensive overview of approaches for mesh adaptation. We focus on moving mesh adaptivity, but also discuss alternative approaches. The overview is not only a good introduction to the methods used in the following chapters, but—more importantly—puts all approaches in perspective, describing their equivalences and differences. This includes mathematical foundations for the methods. It also adds to some older reviews with the field's publications over the last years.

2.1 INTRODUCTION

Numerical simulations should result in highly accurate solutions to the problems they are to solve. If the problem involves solving partial differential equations (PDEs), the accuracy generally improves by increasing the degrees of freedom. This describes the number of unknowns and is closely related to the amount of mesh points and—for finite element methods—the order of the basis functions. A finer discretisation and basis functions of higher order will therefore yield better results.

Numerical simulations have to be efficient too, though. Doubling the resolution becomes increasingly expensive on multidimensional domains. Besides, large areas may not need all that costly resolution. The key point of adaptive methods is to only *locally* increase the resolution of the method.

2.1.1 *Three classes of adaptivity*

The accuracy of PDE solvers can be improved by dividing certain mesh cells into multiple smaller cells (h-refinement), moving mesh nodes freely (r-refinement), or locally increasing the order of basis functions (p-refinement).

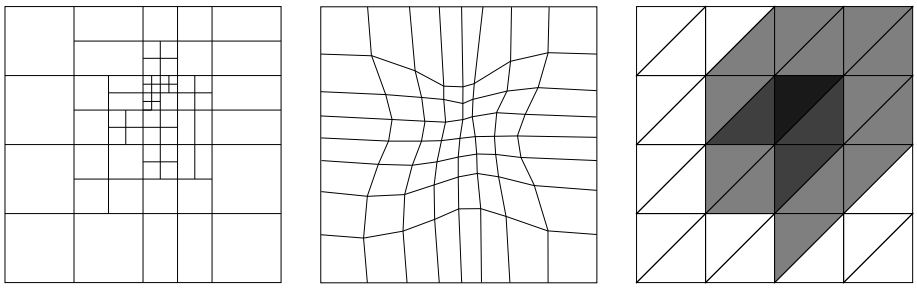

Figure 2.1 Three classes of adaptivity: h-refinement, r-refinement and p-refinement.

Figure 2.1 depicts the three types of adaptivity schematically. h-Refinement starts with a coarse mesh and recursively splits individual cells until certain criteria are met. A relevant concept for h-refinement is *proper nesting*, meaning that the difference between refinement level of two neighboring cells is at most one, to prevent large differences in cell size. The first diagram also shows how refinement can be *anisotropic*, which means possibly different in each direction. For example, some cells are only split vertically. Cells may also be merged, or *coarsened* again, when the local solution behavior has become less demanding.

r-Refinement, depicted in the second diagram, starts with a reasonably dense mesh and keeps the number of mesh cells constant all the time. It moves the mesh points towards regions of interest, such that the mesh cells there become smaller, thus increasing local resolution. Here also, anisotropic adaptivity is often used, but is is generally called *directional* adaptivity.

p-Refinement, finally, is mainly used in finite element solvers. It locally increases the order of the basis functions. As these solvers are less related to the solvers that we use, this type of refinement is not further discussed.

The above three methods can also be combined. hp-Type adaptivity is widely used. hr-Type adaptivity is also investigated, albeit to a lesser extent.

Each type of adaptivity needs good criteria for refinement, as it is an automatic process, incorporated into the solver, often for time-dependent problems. Evidently, local error measures, either a priori or a posteriori are important for *volume control*: small mesh cells for increased accuracy, and larger mesh cells where the solution is smooth. Besides, additional measures may guard the quality of the mesh, e.g., in terms of orthogonality or skewness.

2.1.2 *Related publications*

Each overview has to limit its scope, therefore we suggest additional books and articles here that are truly worth reading.

A broad discussion of computational grids is given by Carey [Car97]. Topics include triangulation, local (h-)refinement and its data structures, p-refinement and combination with finite element methods, and—to a lesser extent—adaptive moving r-refinement. In Section 2.6 we only give a brief discussion of h-refinement.

Castillo [Cas91] collected chapters on grid generation that cover elliptical generators, variational formulations and harmonic maps, which we discuss in Section 2.3.2. It is a good introduction to the various approaches with references to more in-depth coverage of those.

Knupp and Steinberg [KS93] take a more geometrical viewpoint. They start with basic (structured) grid generation techniques and then focus on the variational approach on lines, planes and manifolds, all fairly in-depth. This also involves functionals for volume, alignment and orthogonality control, which we discuss in Sections 2.3.2.4 and following and also in Section 2.5.4.

The classic book by Thompson, Warsi and Mastin from 1985 is still available online [TWM85]. It offers the same in-depth treatment of all geometrical concepts of mesh generation and adaptation. It lacks the developments from

the early 1990s on harmonic mappings and functionals in the variational approach, which makes Knupp's above book more suitable.

Azarenok has collected chapters on grid generation and applications by Ivanenko [Iva04]. It treats similar methods as the previous two books, but has an excellent historical introduction with a wealth of references.

Hansen, Douglass and Zardecki [HDZ05] cover the same elliptical mesh enhancement methods, but in a more pragmatic way, illustrated with examples and extended to unstructured meshes as well.

The well-known handbook by Thompson, Soni and Weatherill [TSW98] is a large collection of chapters on diverse topics, from the elliptic generators in Section 2.3.2 to unstructured meshes, from complex geometries and CAD design to mesh qualities. It lacks the developments of the past decade in solution monitoring as we present in Section 2.5.

There also exist some good overview papers on r-refinement, which differ in scope and up-to-dateness. Liseikin [Lis96] treats the classical location-based adaptive methods, e.g., equidistribution, variational formulation and harmonic mapping. It complements other review papers with many references to publications on these methods by Russian researchers.

Cao et al. [CHR03] have given a well-structured overview of approaches for the generation of adaptive meshes, which forms the basis of our discussions in Section 2.3. They compare several methods for prescribed monitor functions, including their deviation from equidistribution. A very recent review by Budd, Huang and Russel [BHR09] covers the same approaches as we do, with focus on the authors' work (i.e., mesh qualities, MMPDEs, blow-up problems and Monge–Kantorovich optimization). It includes illustrative examples.

Huang, coauthor of the previous overviews has also written a workshop chapter on the same methods [Hua07]. Its main contribution is the discussion of (an)isotropic mesh adaptation through monitor functions and mesh quality measures. We cover this in Section 2.5.

Hægland and Skaflestad [HS02] have given a survey that has similar coverage as the previous overview, with more details on moving finite elements and the geometric conservation law. They also include some h-adaptivity.

Tang [Tan05] has mainly focused on the coupling between the mesh adaptation and the physical PDEs, e.g., the various approaches for solution interpolation, which we discuss in Section 2.4.2. The physical PDEs are solved by both finite element and finite volume approaches. The methods are illustrated with challenging examples of incompressible flow and reactive flow.

Zegeling [Zeg07] considers equidistribution and the underlying concepts, e.g., truncation errors and mesh smoothness. He has collected a wide range of applications of moving mesh methods, e.g., singular blow-up problems, transport in porous media and resistive magnetohydrodynamics.

The current chapter is not a book on its own and therefore not as in-depth as the books mentioned. It *does* emphasize the equivalences and differences between the many approaches, though. Besides it has a wider scope than the individual review papers mentioned, including new relevant publications over the past five years.

2.1.3 *Outline*

This overview mainly concerns adaptive moving mesh refinement. We clarify the discussion by structuring it according to three aspects. Section 2.2 introduces these aspects involved in r-refinement methods. The subsequent sections 2.3, 2.4 and 2.5 are dedicated to each respective aspect. We briefly discuss local h- and hr-refinement in Section 2.6, which is not as complete as its predecessors, but is interesting nonetheless as these two types of refinement are often compared. The chapter ends with a short overview of the techniques that together form our solvers in the following chapters.

2.2 ADAPTIVE MOVING MESH OR R-REFINEMENT

r-Refinement is the main topic of this dissertation, therefore an extensive background of the surrounding research field is given. This section gives an accessible overview of the various aspects involved in r-refinement research. The sections thereafter classify a large body of literature according to these aspects. Figure 2.2 depicts the schematic structure of all approaches that we discuss.

Approaches for mesh adaptation

Mesh adaptation serves intuitive goals, but how can it be prescribed mathematically? The specification of the adaptation goals is left to Section 2.5. First, we distinguish two approaches of prescribing adaptive meshes, inspired by Cao et al. [CHR03]. The first is *location based adaptation*, which—at each time—defines mesh points by a mesh map with respect to some uniform reference mesh. The second is *velocity based adaptation*, which defines mesh point velocities, resulting in mesh points that move towards their desired position. These two approaches form the first defining aspect of r-refinement techniques and are discussed in Section 2.3.

Coupling with physical PDEs

Mesh *generation* is generally considered a research area of its own, but mesh *adaptation* is usually incorporated into a numerical (PDE-)solver, aimed at making the mesh more suitable for representing the numerical solution. The physical PDEs now have to be solved on a time-dependent non-uniform mesh. The second defining aspect of these adaptive solvers is how either these PDEs are transformed or the actual solver is enhanced for this, e.g., by solution interpolation. It is discussed in Section 2.4.

Control by monitor functions

The goals that drive mesh adaptation can be diverse. Evidently, solution-based adaptation is important for controlling mesh cell volumes. Preserving good mesh qualities is another goal. Direction-dependent adaptation or even alignment with vector fields form possible enhancements. The choices made in these form the third and final aspect of r-refinement. They are discussed in Section 2.5.

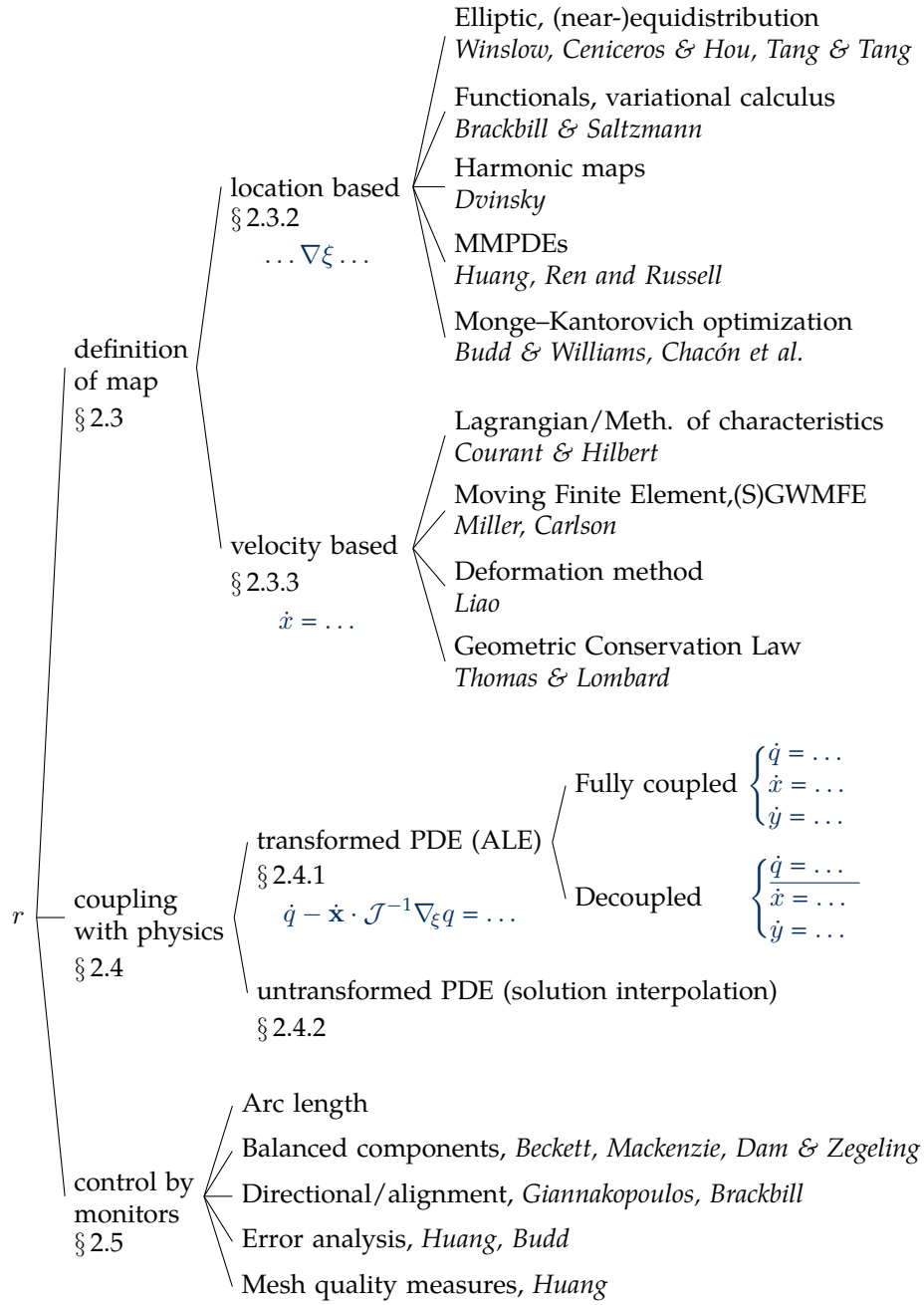

Figure 2.2 Categorization of r-refinement methods. The *author names* are only some representatives, more references are given in the associated sections.

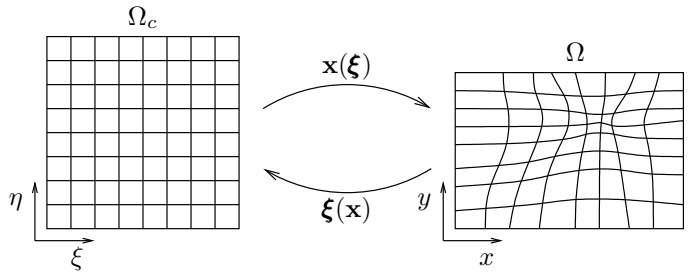

Figure 2.3 Mesh map in two dimensions. The computational domain Ω_c gets mapped onto the physical domain Ω by $\mathbf{x}(\boldsymbol{\xi})$. The inverse map $\boldsymbol{\xi}(\mathbf{x})$ should also exist.

2.3 R-REFINEMENT: DEFINITION OF MESH ADAPTATION

Nonuniform meshes can be described by a mesh map from a *reference* or *computational domain* to the *physical domain*. *How* this mesh map is prescribed or obtained will be discussed in Sections 2.3.2 and 2.3.3. They make the general distinction between location based and velocity based approaches, respectively. First, we will introduce some general terminology and properties of mesh maps.

2.3.1 *Concepts of mesh maps*

The definition and some properties of mesh maps are now introduced. We will specialize most of the notation to two dimensions. Simplification to one-dimensional spaces—as used in Chapter 3—is trivial. Extension to three dimensions is sometimes straightforward, but potential difficulties will be indicated in later sections.

2.3.1.1 *Spaces and mesh maps*

The computational domain Ω_c is the unit n-cube—here only the interval and square—that resides in the logical space. It gets mapped onto the physical domain Ω in physical space:

$$\begin{aligned} \boldsymbol{\xi} &:= [\xi_1, \ldots, \xi_n]^T = [\xi, \eta]^T \in \Omega_c := [0,1]^n, \\ \mathbf{x} = \mathbf{x}(\boldsymbol{\xi}) &:= [x(\xi,\eta), y(\xi,\eta)]^T : \Omega_c \to \Omega. \end{aligned} \tag{2.1}$$

The domain has boundary $\partial\Omega$ with outward normal $\mathbf{n}$. The *normal derivative* represents the gradient in this direction: $\partial/\partial n := \mathbf{n} \cdot \nabla$. Figure 2.3 depicts this map. Mesh mapping can serve two purposes. One is the generation of a nonuniform domain discretization, which is our only purpose in this work. Second is the generation of boundary-conforming discretizations on complex geometries. In the latter case, one often includes an intermediate *parametric domain*. This results from a nonuniform, e.g., solution adaptive map from the reference domain and in its turn gets mapped onto the physical domain. We only employ a direct map from reference to physical domain, since in our

case the physical domain is convex, more specifically: a straight line in 1D and a rectangle in 2D.

2.3.1.2 *Jacobians and invertibility*

We require the mesh map to be bijective, which is satisfied if the map is continuously differentiable and its Jacobian determinant is nonzero. More specifically, we require the mapping to preserve the orientation of the boundary, so that the Jacobian determinant should be strictly positive. It differs per method how the positive Jacobian criterion is satisfied. Later sections will present proofs for it. The mapping's Jacobian matrix and its determinant are denoted as follows:

$$\mathcal{J} := \nabla_\xi \mathbf{x} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}, \tag{2.2}$$

$$J := \det(\mathcal{J}), \tag{2.3}$$

where the subscripts denote partial derivatives and the *computational* gradient operator is defined as follows[1]:

$$\nabla_\xi := \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix}. \tag{2.4}$$

In the following, the term 'Jacobian' may refer to both the matrix and the determinant, depending on the context.

The inverse map $\boldsymbol{\xi}(\mathbf{x})$ has Jacobian matrix $\mathcal{J}^{-1}$ and determinant $1/J$ and its partial derivatives may be rewritten into computational derivatives as follows:

$$\begin{aligned} \xi_x &= \frac{1}{J} y_\eta \qquad \eta_x = \frac{-1}{J} y_\xi \\ \xi_y &= \frac{-1}{J} x_\eta \qquad \eta_y = \frac{1}{J} x_\xi. \end{aligned} \tag{2.5}$$

This follows from the chain rule $\mathcal{J}\mathcal{J}^{-1} = I$, where I is the identity matrix.

Many location-based methods (Section 2.3.2), e.g., Winslow and harmonic mapping use the inverse map, since its existence and one-to-one property is more easily satisfied. This is due to the requirement that the boundary of the codomain is convex, which is always the case for the computational domain Ω_c, but may not always be the case for physical domain Ω. The sections on the aforementioned methods will elaborate on this.

The Jacobian is a local measure of area in the physical space, which is easily seen by a change of variables to the fixed—i.e., area 1—computational domain:

$$|\Omega| := \int_\Omega 1 \, \mathrm{d}\mathbf{x} = \int_{\Omega_c} 1 \cdot J \, \mathrm{d}\boldsymbol{\xi}. \tag{2.6}$$

[1] Any gradient operator ∇ is defined in the usual sense: when applied to a scalar function it yields a column vector. When applied to a column vector-valued function it yields the Jacobian matrix: $\nabla \mathbf{v} := (\nabla \mathbf{v}^T)^T$. Without subscript, physical coordinates are assumed: $\nabla := \nabla_{\mathbf{x}}$.

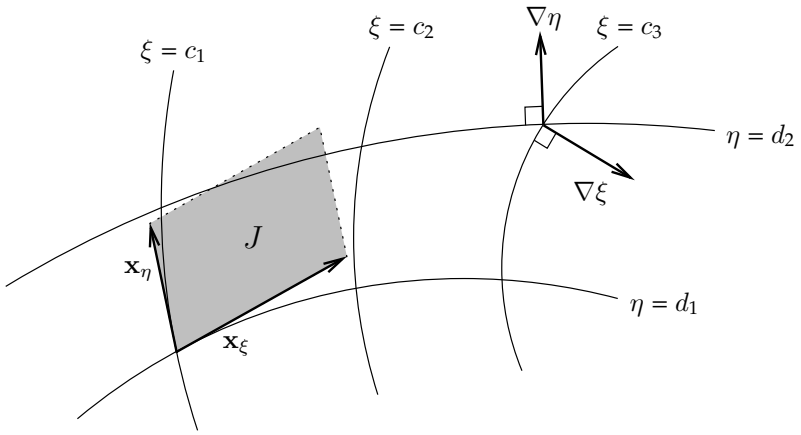

Figure 2.4 Curvilinear coordinate mapping on a planar domain. The tangent and normal vectors are not necessarily up to scale.

In a discretized domain, the Jacobian thus gives an approximation of local cell areas. Positivity of the Jacobian is equivalent to non-collapsing of mesh cells.

2.3.1.3 *Differential geometry*

The theory of mesh maps is generally formulated on manifolds; lines and planes are special cases of these. To support the description of some of the variational methods in Section 2.3.2, we will introduce some elementary differential geometry. In particular, the energy (2.38) associated with a map between two manifolds will be expressed using the theory below, in Section 2.3.2.6 on harmonic maps. The aforementioned handbooks in Section 2.1.2 provide an in-depth coverage of this field. Figure 2.4 depicts the planar domain with curvilinear coordinates to illustrate the normal and tangent vectors used below.

The *metric tensor* or *matrix* G associated with a manifold contains important geometrical interpretations of the map that defines the manifold and is defined by:

$$G = [g_{ij}] := \mathcal{J}^T \mathcal{J} = \begin{bmatrix} x_\xi^2 + y_\xi^2 & x_\xi x_\eta + y_\xi y_\eta \\ x_\xi x_\eta + y_\xi y_\eta & x_\eta^2 + y_\eta^2 \end{bmatrix}, \tag{2.7}$$

where the latter term is specialized for two dimensional surfaces. The matrix elements g_{ij} are in fact dot products of the tangent vectors to two coordinate curves:

$$g_{ij} := \mathbf{x}_{\xi_i} \cdot \mathbf{x}_{\xi_j} = \sum_{l=1}^{n} \frac{\partial x_l}{\partial \xi_i} \frac{\partial x_l}{\partial \xi_j} \quad \text{for} \quad i, j = 1, \ldots, n. \tag{2.8}$$

Note that the metric tensor G is square symmetric with its dimension n equal to the dimension of the computational space. For planar mesh generation, the

physical space has the same dimension. A coordinate system is orthogonal if and only if the metric matrix is diagonal, that is, $g_{ij} = 0$ for $i \neq j$. More specifically, the Euclidean metric is $g_{ij} = \delta_{ij}$, which equals 1 if $i = j$ and 0 otherwise.

The *metric* g is defined as the matrix determinant:

$$g := \det(G) = 1/\det(G^{-1}). \tag{2.9}$$

A change of variables in integration can be expressed using the above, since by definition (2.7) the Jacobian determinant (2.3) can be expressed as:

$$J = \sqrt{g}. \tag{2.10}$$

This makes it also easy to see that a map from logical to physical space is nonsingular in the point $\boldsymbol{\xi}$ if and only if the metric $g \neq 0$ at $\boldsymbol{\xi}$.

The above concepts are generally referred to as *covariant*, dealing with tangent vectors $\mathbf{x}_{\xi_j}$. For the inverse map $\boldsymbol{\xi}(\mathbf{x})$ similar concepts exist, now tagged as *contravariant*, dealing with normal vectors $\nabla\xi_j$.

The contravariant metric tensor G^{-1} has components g^{ij}, defined by:

$$g^{ij} := \sum_{l=1}^{n} \frac{\partial \xi_i}{\partial x_l} \frac{\partial \xi_j}{\partial x_l}, \tag{2.11}$$

which specializes for two-dimensional planes to:

$$G^{-1} = \begin{bmatrix} \xi_x^2 + \xi_y^2 & \xi_x\eta_x + \xi_y\eta_y \\ \xi_x\eta_x + \xi_y\eta_y & \eta_x^2 + \eta_y^2 \end{bmatrix}. \tag{2.12}$$

The elements g^{ij} are in fact dot products of the *normals* to two coordinate surfaces.

2.3.1.4 *Discretization*

For later reference we now introduce the notation for discretized coordinates. The mesh adaptation approaches that we will discuss apply to arbitrary meshes, both structured and unstructured. In our own work we limit ourselves to structured meshes. The physical domain is rectangular and discretized into $N_x \times N_y$ quadrangles:

$$\Omega := [a_1, b_1] \times [a_2, b_2], \tag{2.13}$$

$$\boldsymbol{\xi}_{j,k} := [\xi_j, \eta_k]^T = \left[\frac{j}{N_x}, \frac{k}{N_y}\right]^T \quad \text{for} \quad j = 0, \ldots, N_x \text{ and } k = 0, \ldots, N_y, \tag{2.14}$$

$$\mathbf{x}_{j,k} := [x_{j,k}, y_{j,k}]^T \quad \text{for} \quad j = 0, \ldots, N_x \text{ and } k = 0, \ldots, N_y. \tag{2.15}$$

Notice that the discretization of the computational domain is Cartesian and uniform in each dimension. The physical domain discretization is nonuniform, but still *logically rectangular*: each mesh point corresponds to a reference point on the Cartesian computational mesh.

In one dimension, the index k drops out, and the cell size is defined as:

$$\Delta x_j := x_{j+1} - x_j \quad \text{for} \quad j = 0, \ldots, N_x - 1. \tag{2.16}$$

The discretization indices j, k are not to be confused with the coordinate indices ij from the previous subsection.

2.3.2 *Location-based mesh adaptation*

Location-based methods obtain a mapping $\mathbf{x}(\boldsymbol{\xi})$ (or its inverse), which is repeatedly computed while solving the physical PDEs, thus making it time-dependent. There is no explicit dependence on time t, but still the (discrete) mesh velocities play a role in the coupling with the physical PDEs (Section 2.4). We now give a classification of the best-known location-based approaches.

2.3.2.1 *Equidistribution in one dimension*

Equidistribution is the concept of equally distributing weights—which possibly measure some local error—over the domain. In one dimension this reads:

$$\int_{x_j}^{x_{j+1}} \omega \, \mathrm{d}x = \frac{\int_\Omega \omega(x) \, \mathrm{d}x}{N_x} \quad \text{for} \quad j = 0, \ldots, N_x - 1. \tag{2.17}$$

This form is generally attributed to De Boor [dB74]. When the weight function ω is related to the local discretization error, such a mesh map gives the minimal overall error. The above form defines equidistribution on a 'per-cell' basis; we can generalize this to any interval in terms of the continuous inverse map (remember that $\xi(x) \in [0, 1]$):

$$\xi(x) = \frac{\int_{a_1}^{x} \omega(s) \, \mathrm{d}s}{\int_{a_1}^{b_1} \omega(s) \, \mathrm{d}s}. \tag{2.18}$$

Substituting x_{j+1} and x_j for x in (2.18) and subtracting yields De Boor's form (2.17). The x-derivative of (2.18) yields an equivalent, but simpler form:

$$\xi_x = \frac{\omega}{c} \quad \Longrightarrow \quad \omega x_\xi = c, \quad \text{where} \quad c = \int_{a_1}^{b_1} \omega(s) \, \mathrm{d}s. \tag{2.19}$$

The discrete variant of this form is probably the most intuitive formulation of mesh adaptation:

$$\omega_j \Delta x_j = c \quad \text{for} \quad j = 0, \ldots, N_x - 1, \tag{2.20}$$

where the weight value ω_j is an approximation of the cell average of the weight function $\omega(x(\xi))$, e.g., by midpoint averaging. Clearly, the larger the weight is, the smaller the mesh cell becomes and vice versa.

In practice, the location-based approaches in this section attempt to satisfy form (2.19) differentiated once more:

$$(\omega x_\xi)_\xi = 0 \quad \text{or equivalently:} \quad \left(\frac{1}{\omega} \xi_x\right)_x = 0. \tag{2.21}$$

This shows that equidistribution is equivalent to Winslow's variable diffusion method (2.27) in one dimension.
Some of the velocity-based approaches in Section 2.3.3 prescribe zero change in time:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\omega x_\xi) = 0. \tag{2.22}$$

The principle of equidistribution, e.g., (2.19) generalizes to multiple dimensions through the Jacobian, which measures the local area:

$$\omega J = \text{constant}. \tag{2.23}$$

Most multidimensional extensions of one-dimensional methods do not trivially satisfy this form, as we will show for the direct variant of Winslow's method in Section 2.3.2.9. The deformation method (Section 2.3.3.3) and GCL-based approaches (Section 2.3.3.4) *do* satisfy (2.23) for any number of dimensions.

2.3.2.2 *Direct or inverse maps*

In 1966 Winslow used Laplace's equation for computational coordinates to generate 'equipotential' meshes [Win66]:

$$\nabla^2 \xi = 0, \quad \nabla^2 \eta = 0, \tag{2.24}$$

an approach for which he gave two arguments. Firstly, the averaging property of solutions to the Laplace equations will lead to meshes that are, in some sense, smooth. Secondly, the use of physical coordinates x and y in (2.24) instead would easily lead to collapse of mesh cells on nonconvex domains. Note that the above and following generators are *not* solution-adaptive. The only goal is to generate a nonuniform mesh that fits the irregular domain. The next sections will introduce solution adaptivity.

Laplace's equation for *physical* coordinates reads:

$$\nabla_\xi^2 x = 0, \quad \nabla_\xi^2 y = 0, \tag{2.25}$$

which has a unique solution that is determined by the boundary mapping. The solution is directly in terms of the desired physical coordinates, so we call this a *direct map*. The system is easy to solve thanks to the uniform logical coordinates and the convex domain. As said, though, there is a big risk of collapsing mesh cells, so we will now describe the use of the computational form (2.24). It is difficult to solve this system on an arbitrary domain with nonuniform physical coordinates, so it is generally translated into computational coordinates. This yields the inverse formulation of (2.24):

$$\begin{aligned} g_{22}\, x_{\xi\xi} - 2g_{12}\, x_{\xi\eta} + g_{11}\, x_{\eta\eta} &= 0, \\ g_{22}\, y_{\xi\xi} - 2g_{12}\, y_{\xi\eta} + g_{11}\, y_{\eta\eta} &= 0, \end{aligned} \tag{2.26}$$

where the covariant metric coefficients g_{ij} were defined in (2.8). The equations are now nonlinear due to the coefficients that contain first-order derivatives

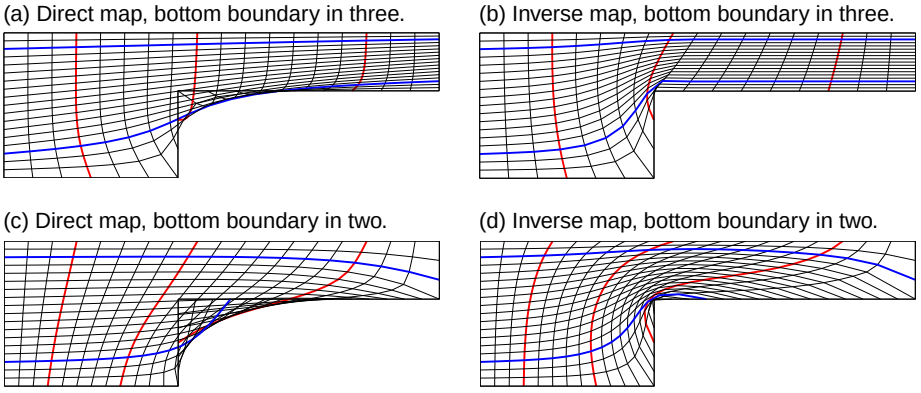

Figure 2.5 Elliptic mesh generation for a nonconvex domain. The direct map is obtained from (2.25), the inverse map from (2.26).

of x and y. Besides, the equations for x and y are now coupled, again due to the coefficients. The inverse formulation does have its price, as we now see. Still, we can linearize the inverse system (2.26). Dvinsky [Dvi91] provides some practical hints for the numerical solution of it.

As an example we consider a nonconvex domain that is similar to a forward facing step, well-known from hydrodynamical problems. Figure 2.5 shows four different discretizations. The left diagrams show the mesh maps obtained from the direct formulation (2.25). The right diagrams show the inverse formulation (2.26). The collapse—or folding—of mesh cells for the direct map is clear, whereas the inverse formulation produces valid meshes using the same boundary conditions.

The difference between the top and bottom diagrams in Figure 2.5 is the choice of boundary mappings. Evidently, the left and top boundary are mapped to the left and top side of the unit square. The four remaining segments provide some freedom. We can map the rightmost boundary to the right side and map the remaining three segments all to the bottom side of the unit square. This is shown in the top diagrams. Notice how three red, logically vertical, sample mesh lines touch the three segments. Alternatively we can map the first two segments to the bottom side and the remaining two segments to the right side. The bottom diagrams show the result. The horizontal segment of the 'step' now is touched by a blue, logically horizontal line. The choice of boundary mapping does change the mesh characteristics, but it has *no* guaranteed effect on the non-collapsing of the mesh. This is only guaranteed by the convexity of the codomain.

The inverse formulation as used by Winslow and many others can handle nonconvex domains without a problem. However, for convex domains the direct approach has none of the above problems and is more straightforward to use. We discuss its properties in Section 2.3.2.8. It has been used success-

fully by, e.g., Ceniceros and Hou [CH01], Tang et al. [TT03] and Zegeling and coworkers [vDZ09, ZdBT05, Zeg05].

2.3.2.3 *Winslow's variable diffusion method*

In 1981 Winslow investigates an adaptive form of the above generator [Win81]:

$$\nabla \cdot (D\nabla\xi) = 0, \quad \nabla \cdot (D\nabla\eta) = 0, \tag{2.27}$$

where the 'diffusion' coefficient $D > 0$ can be some function of the solution to the physical PDEs. In one dimension this is the same as equidistribution (2.21) when $D = 1/\omega$. Expanding the divergence operators yields:

$$\begin{aligned} \nabla^2\xi &= -\nabla\xi \cdot \frac{\nabla D}{D}, \\ \nabla^2\eta &= -\nabla\eta \cdot \frac{\nabla D}{D}. \end{aligned} \tag{2.28}$$

This shows that Winslow's method is a special case of the inhomogeneous TTM generator by Thompson, Thames and Mastin [TTM74], where the right hand sides are the control functions P and Q. In fact, the equipotential generator (2.24) is often called the homogeneous TTM generator. In general, choosing the source terms P and Q requires experience and skill, so Winslow's new formulation was a major improvement. Several handbooks on grid generation give an overview of possible choices, e.g., Knupp and Steinberg [KS93].

Again, the inverse map in (2.28) is found by first translating the generator equations to computational coordinates. Using the inverted partial derivatives from (2.5) yields the inverse formulation of Winslow's generator:

$$\begin{aligned} g_{22}\, x_{\xi\xi} - 2g_{12}\, x_{\xi\eta} + g_{11}\, x_{\eta\eta} &= (D_\xi y_\eta - D_\eta y_\xi)\frac{J}{D}, \\ g_{22}\, y_{\xi\xi} - 2g_{12}\, y_{\xi\eta} + g_{11}\, y_{\eta\eta} &= (D_\xi x_\eta - D_\eta x_\xi)\frac{J}{D}, \end{aligned} \tag{2.29}$$

where the covariant metric coefficients are the same as for the Laplace generator, i.e., as in (2.8). Again, the need to invert the generator equations makes this method more complicated. In three dimensions this inversion needs to be redone. Section 2.3.2.8 discusses a direct variant of Winslow's variable diffusion method.

Winslow's approach inspired many others, mainly due to the intuitive specification of solution adaptivity through D and the resulting smooth meshes. The approaches discussed in the next sections will often have some equivalence relation with it, as we will show.

2.3.2.4 *Variational formulations*

Winslow already described how his variable diffusion method can also be derived from a variational formulation. The calculus of variations deals with functionals—often as integrals of unknown functions—and their minimization. In this context, we can define a functional, sometimes called 'mesh

energy', that measures the appropriateness of a mesh map according to certain adaptation criteria. For example, Winslow's variable diffusion generator (2.27) can also be obtained from the minimization of a weighted-smoothness functional:

$$I_s := \int_\Omega \frac{1}{\omega}((\nabla\xi)^2 + (\nabla\eta)^2)\,\mathrm{d}\mathbf{x}, \tag{2.30}$$

where we use $D = 1/\omega$.

Finding the mesh map that minimizes such a mesh energy functional is equivalent to solving the associated Euler–Lagrange equations. The interesting aspect of this approach is that the preceding generators can also be formulated in a variational way. Moreover, it is easier to define additional adaptation criteria, such as mesh alignment or orthogonality and other mesh quality measures. A thorough investigation of variational calculus and the derivation of Euler–Lagrange equations is beyond the scope of this work, but many good references exist, such as the book by Arthurs [Art75], or—in the context of mesh generation—Knupp and Steinberg [KS93]. In the following subsections we consider several approaches that start from a variational formulation.

2.3.2.5 *Brackbill and Saltzmann's combined functional*

Brackbill and Saltzmann [BS82] devised their variational generator around the same time as Winslow published his adaptive generator. They use a combination of three functionals. The first measures *smoothness* and is equal to (2.30) with $D = 1$, which is equivalent to Winslow's equipotential zoning (2.24). The second measures orthogonality:

$$I'_o := \int_\Omega (\nabla\xi \cdot \nabla\eta)^2 J^3\,\mathrm{d}\mathbf{x}, \tag{2.31}$$

which is volume-weighted. Note that this functional is equal to $\int_{\Omega_c}(\mathbf{x}_\xi \cdot \mathbf{x}_\eta)\,\mathrm{d}\boldsymbol{\xi}$, which may be more intuitive as it uses the dot product of the two tangent vectors to a mesh cell (see Figure 2.4). The third is for volume control:

$$I_v := \int_\Omega wJ\,\mathrm{d}\mathbf{x}, \tag{2.32}$$

where $w := w(\mathbf{x})$ is a given function. The combined functional is then:

$$I_{BS} := I_s + \lambda_v I_v + \lambda'_o I'_o. \tag{2.33}$$

The smoothness measure is always included, such that existence of solutions is guaranteed. Next, parameters $\lambda_v \geq 0$ and $\lambda'_o \geq 0$ are user-defined. Castillo et al. [CSR88] state that volume control is most important, followed by smoothness, and only some orthogonality control is beneficial.

The Euler–Lagrange equation for the minimization of the combined functional (2.33) are solved by a Jacobi iteration. Very large values for λ_v and λ'_o harm the ellipticity, in which case a solution may not always be found,

cf. [Lia92]. Brackbill [Bra93] remedies this by a normalization and modified functionals. We will further discuss these choices in Section 2.5.2.

Some years earlier, Yanenko et al. [YLK77, YKL+79] used a similar approach with the following combined functional:

$$I_{\mathrm{Y}} := \int_{\Omega} (\epsilon_1 h_1 + \epsilon_2 h_2 + \epsilon_3 h_3)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t. \tag{2.34}$$

The first term measures the mesh distortion through deviation from a conformal map:

$$h_1 := (\xi_x - \eta_y)^2 + (\xi_y + \eta_x)^2. \tag{2.35}$$

A conformal map satisfies the Cauchy–Riemann equations ($\xi_x = \eta_y, \xi_y = -\eta_x$), which gives $h_1 = 0$. The second term measures the deviation from 'Lagrangianness', the degree to which the mesh moves along with the physical fluid:

$$h_2 := \|\mathbf{v} - \dot{\mathbf{x}}\|^2, \tag{2.36}$$

where $\mathbf{v}$ is the fluid's velocity. The third term is volume control again:

$$h_3 := \omega J^{\alpha}, \tag{2.37}$$

where α is an additional parameter, which they leave unspecified, and ω is a monitor function based on solution gradients. When $\epsilon_2 = \epsilon_3 = 0$ the resulting mesh is equal to Winslow's equipotential mesh (2.25).

The Euler–Lagrange equations for the minimization of functional (2.34) are time-dependent and coupled with the physical PDEs, Euler's equations of gas dynamics. The equations are simplified such that only one of the coordinates is dynamically adapted, which makes the examples a lot simpler.

2.3.2.6 *Harmonic maps*

Harmonic maps are critical points of an energy functional. Intuitively, this means that the obtained mesh map minimizes some mesh energy, thus making it the most appropriate for the specified goal. An important feature of harmonic maps is that their existence and uniqueness can be proved in any dimension, when certain conditions are satisfied. Their invertibility is *not* guaranteed for higher than two dimensions, though. We will get back to this at the end of this section.

A firm mathematical basis for harmonic map theory was developed since the 1950s, and harmonic maps were first applied by Dvinsky [Dvi91] for goal-based mesh adaptation. Winslow's equipotential generator (2.24) is a basic harmonic map generator.

In general, a harmonic map $\boldsymbol{\xi}(\mathbf{x})$ maps between two Riemannian manifolds M and N with Riemannian metrics g_{ij} and $h_{\alpha\beta}$, and local coordinates x^i and ξ^{α}, respectively. The energy associated with this map is then defined as:

$$E(\boldsymbol{\xi}) := \int_M |\nabla \xi|^2 \,\mathrm{d}M = \frac{1}{2}\int_M g^{ij}(\mathbf{x}) \frac{\partial \xi^{\alpha}}{\partial x^i} \frac{\partial \xi^{\beta}}{\partial x^j} h_{\alpha\beta}(\boldsymbol{\xi}(\mathbf{x}))\sqrt{g}\,\mathrm{d}\mathbf{x}, \tag{2.38}$$

where we use the standard summation convention.[2] The map is harmonic if it is a critical point of this energy functional. Alternatively, it can be found as the solution to the energy's Euler–Lagrange equations.

Dvinsky applies harmonic maps to the same two-dimensional planar domains from the previous subsections, i.e., $M = \Omega$, $N = \Omega_c$. The computational domain is Euclidean, hence its metric is diagonal: $h_{\alpha\beta} = \delta_{\alpha\beta}$. The key point for adaptation is now the following. The domain M may be planar, but we are still free to choose any metric G on it. Dvinsky's harmonic mapping functional now simplifies to:

$$I_{\text{hrm}} := \int_{\Omega} \left[(\nabla\xi)^T G^{-1} (\nabla\xi) + (\nabla\eta)^T G^{-1} (\nabla\eta) \right] \sqrt{g}\, \mathrm{d}\mathbf{x}, \tag{2.39}$$

whose minimizer is the desired mesh map. The freedom to choose the metric G on the physical domain provides the means to prescribe a nonuniform, i.e., adaptive mesh. Dvinsky's choice of monitor functions is described in Section 2.5.1 and involves solution gradients, i.e., aimed at mesh adaptation. Brackbill [Bra93] uses harmonic mapping in combination with a functional for mesh alignment, but we note that the adaptivity functional is incorrectly fitted within the framework of harmonic mappings (see below). Li et al. [LTZ01] devise a method based on harmonic maps, which—for their choice of arc length-type monitor—is actually similar to the direct variant of Winslow, which we will discuss in Section 2.3.2.8. Recently, Di et al. [DLT08] employ the same approach, but now with a multigrid solver for the generator equations, applied to three-dimensional multi-phase flows.

The above authors use convex domains and warn for the complicated form of the inverse formulation when nonconvex domain would have to be handled. Chacón and Lapenta [CL06] *do* derive a compact formulation of this inverse map. They use a Jacobian-free Newton–Krylov approach to solve the discretized equations, with a multigrid preconditioner to handle the ill-conditionedness of the system. The adaptivity is tested on prescribed solution, but the same authors have experimented with coupled solution and mesh equations in 1D [LC06]. Recently, the authors seem to have switched to Monge–Kantorovich optimization for mesh adaptation, which we discuss in Section 2.3.2.10.

Existence, uniqueness and invertibility

When the target domain has nonpositive curvature and a convex boundary ∂N, there exists a unique harmonic map $\boldsymbol{\xi} : M \to N$. This holds in any dimension. Moreover, when M and N are two-dimensional domains this map is a diffeomorphism, i.e., its inverse exists and both are differentiable.

The above is the Hamilton–Schoen–Yau (HSY) theorem due to Hamilton [Ham75] and Schoen and Yau [SY78].

[2] The Einstein summation convention is notational shorthand for a sum of terms. When an index appears twice in a single term, once as subscript, once as superscript, this term is summed over all possible values for that index. The index generally concerns coordinates, hence attains the values $1, \ldots, n$, where n = dim(space).

As mentioned above, the manifold N is the planar computational domain, so it has zero curvature. Besides, it has a convex boundary, so both conditions are met and the harmonic map is indeed suitable for mesh generation and adaptation.

When also the manifold M is planar, namely the physical domain $M = \Omega$, an older version of the above theorem exists, which is due to Rado [Rad26]. Harmonic maps between two planes simply yield two coordinate maps (for ξ and η) that are both harmonic functions. Harmonic functions are C^2 functions that are solutions to Laplace's equation.

Brackbill [Bra93] was inspired by the above foundations of harmonic maps and claimed that Winslow's method is a special case of a harmonic map. This is not true, though. To equate the integrands in both energy functionals (2.30) and (2.39), the following should hold:

$$\frac{1}{\omega} I = \sqrt{g} G^{-1}.$$

Hence, at least:

$$\det\left(\frac{1}{\omega} I\right) = \det(\sqrt{g} G^{-1}) \quad \Longrightarrow \quad \frac{1}{\omega^2} = \frac{g}{g} \quad \Longrightarrow \quad \omega = 1.$$

In other words: Winslow's method only yields a harmonic map if there is no adaptivity ($\omega = 1$), i.e., when it is the Laplace mesh generator (2.24). The claim that the HSY-theorem applies to Winslow's method is thus not valid. An alternative proof for the invertibility of the obtained map is given by Clément et al. [CHS96], which we discuss in Section 2.3.2.8.

Liao [Lia91] critically noted how the invertibility of harmonic mappings does not extend to three dimensions. This led to the development of the deformation method [LA92, LJL99], whose theoretical basis *does* extend to higher dimensions. The deformation method is studied in Section 2.3.3.3.

2.3.2.7 *Huang, Ren and Russell's MMPDE approach*

In 1994, Huang, Ren and Russell [HRR94] presented an adaptive moving mesh method in 1D, based on moving mesh PDEs (MMPDEs) that strive to achieve the equidistribution principle. Later, they extended some of the better MMPDEs to two dimensions, now motivated from the theory of harmonic maps, which results in gradient flow equations [HR97a, HR99]. In 2001 Huang describes the practical aspects of the actual implementation [Hua01a]. Subsequent work of the same author concentrates on proper monitor functions, which we will discuss in Section 2.5.3.

In the MMPDE approach, the mesh map is explicitly time-dependent, i.e., $x(\xi, t)$. Several one-dimensional MMPDEs are proposed; one that lies very close to equidistribution (2.21) is MMPDE5:

$$\dot{x} = \frac{1}{\tau} \frac{\partial}{\partial \xi} \left(\omega \frac{\partial x}{\partial \xi} \right), \tag{2.40}$$

where $\tau > 0$ is a time-relaxation parameter to fit the speed of mesh movement approximately to the typical physical time scales (see page 43). Clearly, the mesh points are moved towards regions of large ω and the mesh speed is zero when exact equidistribution is attained.

In the general, multidimensional gradient-flow formulation [HR99], the mesh functional is generalized to:

$$I := \int_\Omega \left[(\nabla\xi)^T G_1^{-1} (\nabla\xi) + (\nabla\eta)^T G_2^{-1} (\nabla\eta) \right] \mathrm{d}\mathbf{x}, \tag{2.41}$$

where the symmetric positive definite matrices G_1 and G_2 are the monitor functions. When $G_1 = G_2 = G/\sqrt{g}$ it is a genuine energy functional (2.39), resulting in a harmonic map, but many other choices have been considered. The functional derivatives $-\delta I/\delta\xi$ and $-\delta I/\delta\eta$ are the directions in which I descends the fastest. The Euler–Lagrange equations yields the functional derivatives for (2.41):

$$\frac{\delta I}{\delta \xi} = -\nabla \cdot \left(G_1^{-1} \nabla \xi \right), \quad \frac{\delta I}{\delta \eta} = -\nabla \cdot \left(G_2^{-1} \nabla \eta \right). \tag{2.42}$$

These functional derivatives define the movement of the mesh points:

$$\frac{\partial \xi}{\partial t} = -\frac{P_1}{\tau} \frac{\delta I}{\delta \xi}, \quad \frac{\partial \eta}{\partial t} = -\frac{P_2}{\tau} \frac{\delta I}{\delta \eta}, \tag{2.43}$$

where P_1 and P_2 are operators with positive spectrum that allows one to change the descent directions and τ changes the time scale of the mesh equation. For example, limiting the above equations to one dimension, $P = (\frac{\omega}{\xi_x})^2 I$ gives the above one-dimensional MMPDE5 (2.40), where I is the identity operator. Other choices for P result in other MMPDEs by Huang et al.

Mackenzie and coworkers have employed MMPDEs for a wide range of applications. Amongst these are phase-field equations modeling fluid solidification and other state transitions [MR02, BMR06], and the Hamilton–Jacobi equations modeling front propagation [MN07].

The adaptive mesh generators in the previous sections are sometimes called *quasi-static* generators, since they do not contain a mesh speed $\dot{x}$. They *are* time-dependent, though, since the monitor functions—or more generally, the mesh functionals—are solution-dependent, hence time-dependent. Section 2.4 describes how these generators combine easily with the PDE solver.

2.3.2.8 *Direct variant of 'variable diffusion'*

Ceniceros and Hou [CH01] have formulated a direct (covariant) method that closely resembles Winslow's variable diffusion method. They start from the following observation. Given a solution $q(x)$, define the solution on the underlying reference mesh as $v(\xi) := q(x(\xi))$. A good mesh map should make this solution 'better-behaved', for example by making the new gradient v_ξ

smoother than the original q_x. They derive the Euler–Lagrange equations and after some simplifications end up with the generator equations:

$$\nabla_\xi \cdot (G\nabla_\xi x) = 0, \quad \nabla_\xi \cdot (G\nabla_\xi y) = 0, \tag{2.44}$$

where $G = \omega I$. We will discuss the monitor function ω in Section 2.5.1. Later work, mentioned below, also uses more general directional monitor matrices for G (Section 2.5.2).

Note that the generator equations (2.44) are a lot simpler than Winslow's inverted equations (2.29). Extension to three dimensions is straightforward by including a similar generator equation for the z coordinate. Still, these direct equations *are* nonlinear and coupled, through the monitor function $\omega(\mathbf{x}, q, \nabla q, \ldots)$, which may contain the solution q and its derivatives.

Ceniceros and Hou solve the generator equations by an MMPDE similar to (2.40):

$$x_\tau = \nabla_\xi \cdot (G\nabla_\xi x), \quad y_\tau = \nabla_\xi \cdot (G\nabla_\xi y), \tag{2.45}$$

where τ is an artificial time, over which the coordinates tend to their steady state solution that satisfies (2.44). They decouple the MMPDE from the transformed physical PDEs, an approach that we describe in Section 2.4.2. One of the test problems concerns inviscid Boussinesq convection and is handled very well by the adaptive method. Others solve the generator equation (2.44) by a Jacobi or Gauss–Seidel iteration of the linearized system. Note that the former is equivalent to a forward time central space discretization of the MMPDE (2.45) when $\Delta\tau = 1$. A Gauss–Seidel iteration converges somewhat faster and has proved reliable. We give more details on this discretization in the following chapters.

Extensions of the method

The direct method was adopted by Tang and Tang [TT03] and Van Dam and Zegeling [vDZ06, vDZ09] and Zegeling et al. [ZdBT05]. Tang and Tang have devised a proper framework for solving hyperbolic PDEs from conservation laws (e.g., gas dynamics) that employs this direct moving mesh approach. This includes a well-designed conservative solution interpolation, which we discuss in Section 2.4.2. Other applications include incompressible flow [TLK08], resistive magnetohydrodynamics (MHD) [Zeg05], ideal MHD [HT07] and the Hamilton–Jacobi equations in 2D and 3D [TTZ03].

Tang [Tan06] included a directional monitor, applied to 2D hydrodynamics (HD). We have adopted this approach and improved the behavior of the monitor functions, as the following chapters will show. We apply the method to 1D magnetohydrodynamics (MHD), 2D HD and 2D MHD, respectively.

We relate the direct formulation to near-equidistribution in Section 2.3.2.9. First, we give proof for the invertibility of the obtained map.

Existence, uniqueness and invertibility

Hagmeijer [Hag94] uses a directional variant of Winslow's method to achieve anisotropic mesh adaptation. Together with Clément and Sweers [CHS96] he

has proved the invertibility of the obtained map in two dimensions. This forms an alternative for the application of the harmonic mapping proof on page 26 to Winslow's generator.

Consider the following problem on the open unit square $S := (0,1) \times (0,1)$. We search a mapping $[s,t] \to [u(s,t), v(s,t)] : \bar{S} \to \bar{S}$ that is the solution to the following system:

$$\begin{cases} \mathcal{L}u = 0 & \text{in } S, \\ \mathcal{L}v = 0 & \text{in } S, \\ u \text{ and } v & \text{are boundary conforming,} \\ \frac{\partial u}{\partial n} = 0 & \text{on the bottom and top boundaries,} \\ \frac{\partial v}{\partial n} = 0 & \text{on the left and right boundaries,} \end{cases} \tag{2.46}$$

where the solutions u and v should lie in the Sobolev space $W^{2,p}(S)$ for $p > 2$ and the operator $\mathcal{L}$ is given by:

$$\mathcal{L} := a_1(s,t)\Big(\frac{\partial}{\partial s}\Big)^2 + a_2(s,t)\Big(\frac{\partial}{\partial t}\Big)^2 + b_1(s,t)\frac{\partial}{\partial s} + b_2(s,t)\frac{\partial}{\partial t}. \tag{2.47}$$

The coefficients—which should not be confused with the physical domain boundaries in (2.13)—satisfy for some $c > 0$ and $\gamma \in (0,1)$:

$$a_i \in C^{0,1}(\bar{S}) \text{ and } a_1 \geq c > 0 \text{ in } \bar{S} \text{ for } i = 1,2, \tag{2.48}$$

$$b_i \in C^{\gamma}(\bar{S}), \text{ for } i = 1,2. \tag{2.49}$$

We now state the main theorem of Clément et al.

Theorem 2.3.1. *The elliptic generator problem (2.46) possesses exactly one solution $u, v \in C^2(\bar{S})$. Moreover $[u,v]$ is a bijection from $\bar{S}$ onto itself and*

$$\det \begin{pmatrix} u_s & u_t \\ v_s & v_t \end{pmatrix} > 0 \text{ on } \bar{S}. \tag{2.50}$$

Proof. The proof forms the main part of the original publication [CHS96]. □

The direct Euler–Lagrange equations (2.44) are a special case of the above problem (2.46). Take $[s,t] := [\xi,\eta]$, $[u,v] := [x,y]$. If we use directional adaptation $G = \text{diag}(\omega_1, \omega_2)$ with $\omega_1, \omega_2 > 0$, then $a_i = \omega_i$ and $b_i = \partial\omega_i/\partial\xi_i$. These coefficients satisfy the conditions (2.48) and (2.49), since the monitor functions ω_i are strictly positive continuous functions with a known minimum value (see Section 2.5).

The same theorem can be applied to the Winslow's original method (2.27) or directional variants thereof. Theorem 2.3.1 does not extend to three dimensions as its proof relies on the Carleman–Hartman–Wintner theorem, which has no 3D equivalent either.

2.3.2.9 *Approximate equidistribution in higher dimensions*

The Winslow formulation (2.27) is equivalent to equidistribution in one dimension, but this is not the case for higher dimensions. Anderson [And90] shows that only under certain assumptions, Winslow's diffusion coefficient is approximately proportional to the cell volume. We make a similar statement for Ceniceros' covariant formulation (2.44).

Theorem 2.3.2. *The mesh map $\mathbf{x}(\boldsymbol{\xi})$ that is the solution to the covariant variational formulation (2.44) in two dimensions only gives equidistribution when R/J is zero, with R as in (2.54).*

Proof. We expand the PDE (2.44) and obtain

$$\nabla_\xi^2 x = -\frac{\nabla_\xi G}{G} \cdot \nabla_\xi x, \tag{2.51a}$$

$$\nabla_\xi^2 y = -\frac{\nabla_\xi G}{G} \cdot \nabla_\xi y. \tag{2.51b}$$

Local cell area is given by the Jacobian, so as a start we now also apply the Laplace operator to it:

$$\nabla_\xi^2 J = \nabla_\xi \cdot \nabla_\xi (x_\xi y_\eta - x_\eta y_\xi). \tag{2.52}$$

Expanding the partial derivatives and substituting (2.51a) and (2.51b) gives an equation of the form:

$$\nabla_\xi^2 J = -\nabla_\xi \bar{G} \cdot \nabla_\xi J - J \nabla_\xi^2 \bar{G} + R, \tag{2.53}$$

where the 'remainder' R is defined as

$$R := 2(\nabla_\xi x_\xi \cdot \nabla_\xi y_\eta - \nabla_\xi x_\eta \cdot \nabla_\xi y_\xi) \tag{2.54}$$

and $\bar{G} := \log G$. We also use $\bar{J} := \log J$ and expanding its Laplacian yields a simplified form of (2.53):

$$\nabla_\xi^2 (\bar{J} + \bar{G}) + \nabla_\xi \bar{J} \cdot \nabla_\xi (\bar{J} + \bar{G}) = \frac{R}{J}. \tag{2.55}$$

When the right hand side term is assumed to be small or even ignored, a solution to (2.55) is given by

$$\nabla_\xi (\bar{J} + \bar{G}) = 0, \tag{2.56}$$

hence $\bar{J} + \bar{G}$ = constant or in the original quantities for the Jacobian and monitor function:

$$\nabla_\xi \cdot (G \nabla_\xi x_i) = 0 \text{ for } i = 1, 2 \text{ and therefore } \frac{R}{J} = 0 \quad \Longrightarrow \quad JG = c \tag{2.57}$$

for some constant c. This final result shows that when $R/J = 0$, Ceniceros' method leads to equidistribution in two dimensions. □

Interpretation of R/J

The right hand side of (2.54) can be rewritten as:

$$\begin{aligned} R &= 2\,(\nabla_\xi x_\xi \cdot \nabla_\xi y_\eta - \nabla_\xi x_\eta \cdot \nabla_\xi y_\xi) \\ &= 2\,(x_{\xi\xi} y_{\eta\xi} - x_{\eta\xi} y_{\xi\xi} + x_{\xi\eta} y_{\eta\eta} - x_{\eta\eta} y_{\xi\eta}) \\ &= 2\,(\det(\nabla_\xi \mathbf{x}_\xi) + \det(\nabla_\xi \mathbf{x}_\eta)). \end{aligned}$$

Hence, R contains the sum of the Jacobian determinants of the two tangent vectors $\mathbf{x}_\xi$ and $\mathbf{x}_\eta$. The change in these vectors defines a contraction or expansion of the coordinates $\mathbf{x}$ and is also related to the curvature of the coordinate lines. Only when this change is significantly smaller than the local cell area, (near-)equidistribution is achieved. Experiments show that in the refined areas, equidistribution is almost achieved, but in stretched areas, the local value for ωJ may sometimes deviate by almost a factor two from the exact value.

2.3.2.10 *Monge–Kantorovich optimization*

The latest approach for adaptive mesh generation is Monge–Kantorovich optimization. The key difference of this method with the other methods discussed, is the following. Instead of optimizing some *combination* of adaptation criteria and mesh quality measures, the Monge–Kantorovich approach *enforces* local equidistribution and then optimizes some mesh quality measure under this constraint. Monge's mapping problem [Mon81] and Kantorovich's associated optimization problem [Kan42] go back a long time. Only recently, Budd and Williams [BW06] employed it for mesh adaptation using relaxation. Delzanno et al. [DCF+08] solve the full nonlinear system instead of using relaxation.

Monge–Kantorovich optimization aims to find a mapping that satisfies for any set $A_c \subset \Omega_c$ (which maps to $A := \{\mathbf{x}(\boldsymbol{\xi}) \,|\, \boldsymbol{\xi} \in A_c\} \subset \Omega$):

$$\int_{A_c} \mathrm{d}\boldsymbol{\xi} = \int_A \omega(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}, \quad \text{i.e.,} \quad \omega J = 1, \tag{2.58}$$

and—under these constraints—minimizes the point displacement:

$$\int_{\Omega_c} \frac{\|\mathbf{x} - \boldsymbol{\xi}\|_2^2}{2}\,\mathrm{d}\boldsymbol{\xi}. \tag{2.59}$$

The constrained minimization problem can be put into variational form by including the equidistribution constraint with a local Lagrange multiplier:

$$I_{\mathrm{MK}} := \int_{\Omega_c} \frac{\|\mathbf{x} - \boldsymbol{\xi}\|_2^2}{2} + \lambda(\boldsymbol{\xi})(\omega(\mathbf{x})J - 1)\,\mathrm{d}\boldsymbol{\xi}. \tag{2.60}$$

The Euler–Lagrange equations imply $\mathbf{x} - \boldsymbol{\xi} = \nabla\lambda$, i.e., the map $\mathbf{x}(\boldsymbol{\xi}) := \boldsymbol{\xi} + \nabla_\xi \Phi$ is a gradient map. Inserting this into equidistribution relation (2.58) yields the *Monge–Ampère* equation to the displacement potential Φ:

$$\nabla_\xi^2 \Phi + H(\Phi) = \frac{1}{\omega(\mathbf{x}, t)} - 1, \tag{2.61}$$

where H denotes the determinant of the Hessian matrix:

$$H(\Phi) := \frac{\partial^2 \Phi}{\partial \xi^2} \frac{\partial^2 \Phi}{\partial \eta^2} - \left(\frac{\partial^2 \Phi}{\partial \xi \partial \eta} \right)^2. \tag{2.62}$$

Budd and Williams solve their Monge–Ampère equation by approximation. Using temporal relaxation, an approximate potential tends to the exact solution over time. The Monge–Ampère equation in its relaxed form is parabolic (PMA), from which a convex potential and thus Jacobian positivity can be proved. This approach is similar to the relaxation approach that yields the MMPDEs (Section 2.3.2.7). In fact, the PMA equation simplifies to one of the MMPDEs in 1D.

Delzanno et al. solve (2.61) directly with a Newton–Krylov method that is sped up with multigrid. They also compare their results with the deformation method. As expected, the deformation method results in distorted meshes for some of the more difficult problems, as no quality control is applied.

The question remains whether such an accurate solution is truly necessary. Example problem 3 in [DCF+08] takes 4.9 seconds of CPU time for a 64×64 mesh with their method. We produced an equally smooth mesh in less than one second using the 'direct Winslow' method from Section 2.3.2.8. This does not maintain equidistribution everywhere, though (see Section 2.3.2.9). Interestingly, the refined regions (with large monitor values) approximate equidistribution the best. Monge–Kantorovich optimization is probably too costly in time-dependent problems with many evolving features, but the exact equidistribution *and* mesh quality control is a definite advantage for problems with singular solutions, for example.

2.3.3 *Velocity-based mesh adaptation*

Velocity-based methods obtain a time-dependent mapping $\mathbf{x}(\boldsymbol{\xi}, t)$ by defining the mesh velocity

$$\dot{\mathbf{x}} := \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \frac{\partial \mathbf{x}}{\partial t}. \tag{2.63}$$

The equation for the mesh velocity is solved in conjunction with the physical PDEs and t is actual time, or—in the case of steady problems—t is an artificial time over which the mesh converges to the desired distribution. The coupling of the mesh equations with physical PDEs is discussed in Section 2.4. We now give a classification of the best-known velocity-based approaches.

2.3.3.1 *The Lagrangian approach: or method of characteristics*

The Lagrangian formulation of physical PDEs moves the coordinates $\mathbf{x}$ with the characteristic velocities of the physical problem. This approach is very important in the analysis of PDEs (see, e.g., [CH62]). For a two-dimensional hyperbolic PDE:

$$\frac{\partial q}{\partial t} + \frac{\partial f(q)}{\partial x} + \frac{\partial g(q)}{\partial y} = 0, \tag{2.64}$$

where q is the solution variable, the characteristic velocity is given by:

$$\mathbf{v} := \left[\frac{\mathrm{d}f(q)}{\mathrm{d}q}, \frac{\mathrm{d}g(q)}{\mathrm{d}q}\right]^T. \tag{2.65}$$

The Lagrangian coordinate lines are then given by:

$$\dot{\mathbf{x}} = \mathbf{v}. \tag{2.66}$$

If we now write the PDE with reference to the computational coordinates $\boldsymbol{\xi}$, the convective flux terms cancel exactly against the mesh velocities, leaving the ODE:

$$\frac{\mathrm{d}q}{\mathrm{d}t} = 0. \tag{2.67}$$

The solution q is known *exactly* for all times t if the initial solution is given. It is constant along each integral curve of (2.66).

Note that this is mainly useful for local analysis of the solution. On the entire domain the mesh map (2.66) can quickly become stretched, skewed or even degenerate. Take, for example, the one-dimensional inviscid Burgers' equation with an initial sine wave solution. The characteristics are defined at $t = 0$ by $\frac{\mathrm{d}x}{\mathrm{d}t} = q(x, 0) = \sin(x|_{t=0})$, so the peaks of the sine yield characteristics that will 'overtake' the characteristics of the smaller sine values after some time. As a result the strict ordering of the mesh points is violated. The requirement that the mesh points are 'tied' to the flow for ever makes the method of characteristics unsuitable for mesh adaptation in nonlinear flow problems. The arbitrary Lagrangian Eulerian (ALE) approach relaxes this requirement: the mesh velocities may mimic the flow, but need not follow it forever. We will discuss the ALE approach further in Section 2.4.1.

Also note that the method of characteristics does not involve any mesh refinement in the usual sense: there is *no* monitor function that can steer mesh points, e.g., to sharp solution gradients.

Harten and Hyman [HH83] average the characteristics for one-dimensional *systems* of PDEs, based on the eigendecomposition of the solution on each mesh cell. They prevent collapses of mesh points by regularization at each time step.

2.3.3.2 *Moving Finite Elements*

Miller and Miller [MM81, Mil81] have introduced the moving finite element (MFE) method for generating adaptive meshes. The idea is to minimize the residual of a solution to the physical PDE not only over a space of basis functions, but also over a space of functions related to mesh velocity.

Consider a general PDE:

$$\frac{\partial q}{\partial t} = \mathcal{L}q, \tag{2.68}$$

for some spatial differential operator $\mathcal{L}$. The residual of a solution q_h is then defined by:

$$\mathcal{R}(q_h) := \left\| \frac{\partial q_h}{\partial t} - \mathcal{L}(q_h) \right\|_2, \tag{2.69}$$

with the standard L^2-norm over the entire domain.

Normally, solutions are written as a superposition of basis functions $q_h := \sum q_i \phi_i$, where i ranges over the number of nodes N. If now the coordinates of the nodes are time-dependent, the basis functions above those nodes also become time-dependent. By the chain rule we have:

$$\frac{\partial q_h}{\partial t} = \sum_{i=1}^{N} \Big[\frac{\partial q_i}{\partial t} \phi_i + \dot{\mathbf{x}} \cdot \psi_i \Big], \tag{2.70}$$

where the $\psi_i := \nabla_{\mathbf{x}_i} q_h$ are n-dimensional (here $n = 2$) *secondary-type basis functions*, reflecting the solution gradient with reference to each node. For more details on the resulting augmented weak formulation and its solution, we refer to the excellent monograph by Baines [Bai94]. In the same work, Baines also shows that for a first-order PDE, the discretized velocities in the MFE method approximate the velocities from the method of characteristics, as discussed in the previous section. For a nonlinear PDE the characteristics may cross at some point, which will cause the finite element solution to 'overturn'.

The solution to the PDE (2.68) now depends on an extra set of basis functions $\{\psi_i\}$ for which additional equations (weak formulations) are needed. Miller and Miller use the equivalence of weak formulations with minimization of the residual functional (2.69), i.e., their approach comes down to finding the following minimum:

$$\min_{\dot{\mathbf{x}}, \dot{q}} I_{\text{GWMFE}} := \min_{\dot{\mathbf{x}}, \dot{q}} \int_{\Omega} (\dot{q} - \nabla q \cdot \dot{\mathbf{x}} - \mathcal{L}q)^2 w + P^2 \, \mathrm{d}\mathbf{x}, \tag{2.71}$$

where $w > 0$ is a weight function and P^2 is a regularizing penalty term that we will discuss later. In the original method $w = 1$, but in the gradient-weighted MFE method $w = 1/\sqrt{1 + |\nabla q|^2}$.

When the standard basis functions ϕ_i are piecewise linear, the secondary-type basis functions ψ_i are discontinuous. This forms a problem when the spatial differential operator $\mathcal{L}$ contains second-order derivatives. In this case, one generally employs 'mollification': in a small region with radius δ around the nodes the original hat functions are made twice differentiable. Next, the integrals from the weak formulation are evaluated in the limit for $\delta \to 0$.

The gradient-weighted MFE (GWMFE) method was introduced by Carlson and Miller [CM98a, CM98b] to reduce the effect of minimization in steep parts of the solution. This makes the method more robust. The weighted formulation is also more natural in terms of normals and tangents to the solution gradient, see, e.g., Baines and the references therein. A disadvantage of the weighted formulation is that it no longer maintains the conservation property when operator $\mathcal{L}$ describes conservation laws (cf. [Bai94]).

The motivation for GWMFE gradually shifted to a geometrical-mechanical viewpoint [Mil97], which led to the String GWMFE (SGWMFE) for systems of PDEs. For a scalar PDE (2.68), GWMFE and SGWMFE are equivalent. The geometrical-mechanical interpretation comes down to the following. The solution q forms a manifold over the domain with coordinates $\mathbf{x}$. The partial

time derivative q_t denotes vertical displacement of this manifold. Instead, the *normal* displacement with reference to the manifold is used, i.e., multiply (2.70) by $1/\sqrt{1+|\nabla q|^2}$. The resulting equation describes the balance between 'viscous drag forces' and 'applied forces' on the manifold. This balance drives the movement of mesh points. Now, if solution $\mathbf{q}$ is a vector, this balance equation and the normal vector is defined separately for each component. Since all components of $\mathbf{q}$ are generally defined on the same mesh, i.e., the same set of nodes, the resulting forces are summed. The SGWMFE method does not separate the equations: they are formulated over a manifold that is $(p+n)$-dimensional, where p is the number of PDEs/components in $\mathbf{q}$. The projection onto the normal direction is done in this higher-dimensional space, and the resulting forces directly prescribe the mesh point movement.

Wacher and coworkers [WSM05, WS07] have further developed the SGWMFE method. Their results for the Gray–Scott reaction–diffusion and shallow water equations show that it is not necessarily better or worse than GWMFE, but its specification for systems seems to be more elegant. A comparison with a basic MMPDE (Section 2.3.2.7) in one dimension shows that for well-chosen parameters they perform equally well on average [Wac05].

Regularization measures

The unknowns in the minimization problem (2.71) are both the solution coefficient q_i and the mesh point coordinates $\mathbf{x}_i$ at each of the N nodes, which can be combined in a vector of length $N \cdot (n+1)$:

$$\mathbf{Y} := [q_1, \dot{x}_1, \dot{y}_1, \ldots, q_i, \dot{x}_i, \dot{y}_i, \ldots]^T, \quad i \leq N.$$

The finite element formulation of the problem requires that the numerical error, i.e., the integrand in (2.71), is orthogonal to the space of test functions. This results in the *nonlinear* discretized finite element system:

$$M(\mathbf{Y})\dot{\mathbf{Y}} = \mathbf{F}(\mathbf{Y}), \tag{2.72}$$

where both the extended mass matrix M and the vector $\mathbf{F}$ (which normally is the stiffness matrix times $\mathbf{Y}$) are nonlinear in the unknown vector $\mathbf{Y}$.

A potential problem is the collapse of two nodes (mesh points), because two of the diagonal blocks in the mass matrix then become singular. Also, when two edges between three nodes form a straight line, the middle point has no force operating on it, again rendering the mass matrix singular. Miller and Miller added a penalty term $P^2 := \sum(\varepsilon|x_i - x_{i-1}|)^2$ in 1D to the residual term as some form of internodal viscosity. New test problems and extension to multiple dimensions led to several alternate forms of the internodal viscosity. Although understanding of these penalties clearly has improved, experience is still required to properly choose their form and parameters, see, e.g., [ZB92].

2.3.3.3 *Deformation method*

In 1992, Liao and Anderson [LA92] proposed a new way of adaptive mesh generation, motivated by problems in extending harmonic maps and Winslow's

variable diffusion to three dimensions. Over the following years, Liao and coworkers further developed the deformation method. This method yields a mesh map whose Jacobian determinant can be prescribed exactly, namely:

$$J = \det\nabla_\xi \mathbf{x}(\boldsymbol{\xi}, t) = \frac{1}{\omega(\mathbf{x}(\boldsymbol{\xi}, t))}, \quad \text{where} \quad \mathbf{x} \in \Omega \subset \mathbb{R}^n,\ t > 0. \tag{2.73}$$

It achieves equidistribution in any number of dimensions for all times.

The main theorem is inspired by Moser and Dacorogna's work [Mos65, DM90] on diffeomorphisms on manifolds with prescribed Jacobian. The coordinate map is an automorphism, e.g., on the unit cube. If the physical domain is different from the computational domain ($\Omega \neq \Omega_c$), an additional transformation can be included, e.g., a linear scaling on rectangular domains or a curvilinear transformation around an airfoil. Liao et al. [LPS94] show that this does not invalidate the original proof.

The existence of a map that satisfies the equidistribution property (2.73) is valid for any number of dimensions, the proof is by construction. We discuss a time-dependent map here, for the static case we refer to the original publications.

Suppose we have a monitor function $\omega > 0$ that is 'normalized' such that at each time it satisfies:

$$\int_\Omega (\omega(\mathbf{x}, t) - 1)\, \mathrm{d}\mathbf{x} = 0. \tag{2.74}$$

The idea is to evolve the mesh map $\mathbf{x}(\boldsymbol{\xi}, t)$ according to a well-chosen velocity field $\mathbf{v}(\boldsymbol{\xi}, t)$. The first step is to find a field that satisfies:

$$\nabla_\xi \cdot \mathbf{v}(\boldsymbol{\xi}, t) = -\frac{\partial}{\partial t}\omega(\boldsymbol{\xi}, t). \tag{2.75}$$

This is a scalar equation for the unknown vector-valued function $\mathbf{v}$, so it is underdetermined. A vector field can always be written as the sum of the gradient of a potential and the curl of some other vector potential. Liao et al. [LJL99] neglect the second term, because they impose zero curl ($\nabla_\xi \times \mathbf{v} = 0$) in order to allow points to move along the boundaries. Substituting this form of $\mathbf{v}$ in (2.75) gives:

$$\mathbf{v} := \nabla_\xi a \quad \Longrightarrow \quad \nabla_\xi^2 a = -\frac{\partial}{\partial t}\omega. \tag{2.76}$$

Now the velocity field is the solution to a Poisson equation, completed by Neumann boundary conditions that keep the points on the boundary:

$$\frac{\partial a}{\partial n} = 0.$$

The second step is to solve the ODE system:

$$\dot{\mathbf{x}} = \frac{\mathbf{v}(\mathbf{x}, t)}{\omega(\mathbf{x})}, \text{ for} \quad t > 0,\ \mathbf{x}(\boldsymbol{\xi}, 0) = \mathbf{x}_0(\boldsymbol{\xi}). \tag{2.77}$$

These two steps will make sure that the resulting mesh satisfies equidistribution, as we will show in Theorem 2.3.3. Interestingly, though, they are also equivalent with the GCL-based approach (see next section), since the substitution of (2.77) in (2.75) yields $\nabla_\xi \cdot (\omega \dot{\mathbf{x}}) = -\omega_t$, which is (2.85).
We now state Liao's main theorem, reformulated in our notation.

Theorem 2.3.3 (Equidistribution for deformation in n dimensions)**.** *Let $\mathbf{x}_0$ be an initial mesh map that satisfies $J(\mathbf{x}_0) = 1/\omega(\mathbf{x}_0(\boldsymbol{\xi}), 0)$. The time-dependent mesh map obtained from (2.76) and (2.77) satisfies $J = 1/\omega(\mathbf{x}, t)$ for all $t > 0$.*

Proof. We only need to prove $\frac{\mathrm{d}}{\mathrm{d}t}(J(\mathbf{x})\omega(\mathbf{x}, t)) = 0$ for all $t > 0$.

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}(J\omega) &= \omega\frac{\mathrm{d}}{\mathrm{d}t}(J) + J\frac{\mathrm{d}}{\mathrm{d}t}(\omega(\mathbf{x}, t)) \\ &= \omega\frac{\mathrm{d}}{\mathrm{d}t}(J) + J(\dot{\mathbf{x}} \cdot \nabla\omega + \frac{\partial}{\partial t}\omega(\mathbf{x}, t)) \\ &= \omega\frac{\mathrm{d}}{\mathrm{d}t}(J) + J\Big(\frac{\mathbf{v}}{\omega} \cdot \nabla\omega + \frac{\partial}{\partial t}\omega(\mathbf{x}, t)\Big). \end{aligned} \tag{2.78}$$

The time derivative of the Jacobian matrix can be derived using (2.77):

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{J} = \frac{\mathrm{d}}{\mathrm{d}t}\nabla_\xi \mathbf{x}(\boldsymbol{\xi}, t) = \nabla_\xi\Big(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(\boldsymbol{\xi}, t)\Big) = \nabla_\xi\Big(\frac{\mathbf{v}}{\omega}\Big) = \nabla\Big(\frac{\mathbf{v}}{\omega}\Big)\nabla_\xi \mathbf{x}, \tag{2.79}$$

which is a matrix differential equation, for which holds in general:

$$\frac{\mathrm{d}}{\mathrm{d}t}X(t) = A(t)X(t) \quad \Longrightarrow \quad \frac{\mathrm{d}}{\mathrm{d}t}\det(X(t)) = \mathrm{trace}(A(t))\det(X(t)).$$

We can now introduce the determinant in (2.79):

$$\frac{\mathrm{d}}{\mathrm{d}t}J = \frac{\mathrm{d}}{\mathrm{d}t}\det(\mathcal{J}) = \Big(\nabla \cdot \Big(\frac{\mathbf{v}}{\omega}\Big)\Big)J = \Big(\frac{1}{\omega}\nabla \cdot \mathbf{v} + \mathbf{v} \cdot \nabla\Big(\frac{1}{\omega}\Big)\Big)J. \tag{2.80}$$

Finally, the definition of the velocity field (2.75) implies $\nabla \cdot \mathbf{v}(\mathbf{x}, t) = -\frac{\partial}{\partial t}\omega(\mathbf{x}, t)$, so that (2.78) simplifies to:

$$\begin{aligned}\frac{1}{J}\frac{\mathrm{d}}{\mathrm{d}t}(J\omega) &= \Big(-\frac{\partial}{\partial t}\omega + \omega\mathbf{v} \cdot \nabla\Big(\frac{1}{\omega}\Big)\Big) \\ &\quad + \Big(\frac{\mathbf{v}}{\omega} \cdot \nabla\omega + \frac{\partial}{\partial t}\omega\Big) \\ &= 0.\end{aligned}$$

□

It is easy to see that in one dimension the above theorem reduces to (2.19) by proving (2.22).

Applications

The deformation method was demonstrated with prescribed monitoring functions, i.e., without any physical PDEs, in the aforementioned papers. The steady Euler equations were solved in [LJL99]. The monitor function is based on pressure gradients:

$$\omega = C_1(1 + C_2\nabla p),$$

where C_2 is set manually to fit the problem and C_1 follows from the normalization requirement (2.74). The resulting meshes are smooth and well adapted. In [LLdlP02] a time-dependent circular implosion problem is considered. Recently, the deformation method is also used in image registration, some first results are presented in [LX06]. Image registration is widely applied to medical scans of deforming tissues, see, e.g., Hsieh et al. [HCL+08]. All of these applications make no fundamental improvements to the original deformation method.

Just like the location-based methods, the deformation method still bears the risk of mesh tangling. The positive Jacobian is guaranteed for the analytic map, but in solving the discretized equations mesh distortion *could* occur, see, e.g., [CJ04] and [DCF+08]. The deformation method does not offer any control over mesh qualities such as smoothness and orthogonality. In this sense, the variational approaches and Monge–Kantorovich optimization (Section 2.3.2.10) are more robust. Both methods involve two steps to obtain the adaptive mesh: first, a vector potential is obtained iteratively, next the new mesh results from (a time-integration of) the gradient field of that potential. For actual flow computations this may be quite expensive. It is not always necessary to strive for exact equidistribution.

2.3.3.4 *Geometric Conservation Law*

Thomas and Lombard [TL79] were the first to recognize the importance of a geometric conservation law (GCL) for conservative flow models. Their context is a domain with moving boundary, but it equally applies to moving mesh approaches in general. Consider an arbitrary volume element $A(t) = \{\mathbf{x}(\boldsymbol{\xi}, t) \,|\, \boldsymbol{\xi} \in A_c\}$ mapped from some reference element $A_c \subset \Omega_c$. The change of its area over a certain time is equal to the area swept by its boundary:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{A(t)} \mathrm{d}\mathbf{x} = \int_{\partial A(t)} \dot{\mathbf{x}} \cdot \mathbf{n}\, \mathrm{d}S. \tag{2.81}$$

A change of coordinates to $\boldsymbol{\xi}$ introduces the Jacobian and application of the divergence theorem to the right hand side then yields:

$$\int_{A_c} \frac{\mathrm{d}}{\mathrm{d}t} J\, \mathrm{d}\boldsymbol{\xi} = \int_{A_c} (\nabla \cdot \dot{\mathbf{x}}) J\, \mathrm{d}\boldsymbol{\xi}. \tag{2.82}$$

This holds for arbitrary volume elements, so we can formulate the differential form of the GCL:

$$\nabla \cdot \dot{\mathbf{x}} = \frac{1}{J}\frac{\mathrm{d}J}{\mathrm{d}t}. \tag{2.83}$$

The GCL holds for any smooth mesh map, but its discrete variant (DGCL) is not trivially satisfied. Its differential form allows for coupling it with the physical PDEs; we discuss this further in the context of ALE methods in Section 2.4.1.

Most research on the GCL is motivated by free surface problems and moving boundaries around objects. Still, it applies equally to solution adaptive meshes. Cao et al. [CHR02] have based a moving mesh method on it. The size of a volume element should be inversely proportional to the monitor function, so they change the GCL (2.83) into:

$$\nabla \cdot \dot{\mathbf{x}} = -\frac{1}{\omega}\frac{\mathrm{d}\omega}{\mathrm{d}t}. \tag{2.84}$$

The combination of (2.83) and (2.84) exactly satisfies multidimensional equidistribution (2.23).

It is more practical to get rid of the total derivative by means of the chain rule:

$$\nabla \cdot (\omega \dot{\mathbf{x}}) + \frac{\partial \omega}{\partial t} = 0. \tag{2.85}$$

This scalar equation has no unique solution, but by the classical decomposition theorem of Helmholtz a continuously differentiable vector field can be written as the orthogonal sum of the gradient field of a scalar potential and the curl of another vector field, so we can additionally specify the curl of the mesh velocity. Cao et al. let

$$\nabla \times w(\dot{\mathbf{x}} - \mathbf{u}) = 0, \tag{2.86}$$

where $w > 0$ is some weight function and $\mathbf{u}$ is a velocity field that $\dot{\mathbf{x}}$ should align with. They prove that due to orthogonality of the divergence and curl in (2.85) and (2.86), the unique solution can also be found from the minimization of the following functional, which combines equidistribution and alignment:

$$I_{\mathrm{GCL}}(\dot{\mathbf{x}}) := \int_\Omega \left| \nabla \cdot (\omega \dot{\mathbf{x}}) + \frac{\partial \omega}{\partial t} \right|^2 + \left(\frac{\omega}{w} \right)^2 |\nabla \times w \cdot (\dot{\mathbf{x}} - \mathbf{u})|^2 \, \mathrm{d}\mathbf{x}, \tag{2.87}$$

completed by the boundary condition:

$$\dot{\mathbf{x}} \cdot \mathbf{n} = 0 \quad \text{on } \partial\Omega. \tag{2.88}$$

The above GCL-based method is a generalization of several others. The Lagrangian method (Section 2.3.3.1) results when the alignment field is set to the physical flow's velocity and adaptivity is disabled ($\omega = 1$).

The MFE methods (Section 2.3.3.2) use a very different functional (2.71), but similar to the GCL approach, it obtains the mesh by a minimization over $\dot{\mathbf{x}}$ and has a strong relation with the Lagrangian approach.

The deformation method (Section 2.3.3.3), finally, results when no particular alignment field is used, i.e., $\mathbf{u} = \mathbf{0}$, and $w = \omega$. Instead, $w = 1$ is a better choice, as it produces a mesh that is free of rotation ($\nabla \times \dot{\mathbf{x}} = \mathbf{0}$).

Cao et al. consider several approaches for finding the minimum of the functional (2.87) and find a direct finite-element minimization of the functional most suitable. The velocity field $\dot{\mathbf{x}}$ is determined at several intermediate time steps in $[t_n, t_{n+1}]$, after which $\dot{\mathbf{x}}$ is integrated by Runge-Kutta steps.

They test their method on prescribed monitor functions, i.e., no underlying physical PDEs. The discrete Jacobian value at each cell differs at most 30% with the value prescribed by exact equidistribution. The experiments also show that the GCL-based approach has the same disadvantages as all velocity-based approaches: when flowing along with the physical flow ($\mathbf{u} \neq \mathbf{0}$), mesh cells can become very skewed. Unless a good intuition for the velocity field $\mathbf{u}$ exists, they recommend setting it to zero. The same authors developed the location-based MMPDEs (Section 2.3.2.7) in which the smoothing effect of the elliptic Euler-Lagrange equation applies to the mesh points $\mathbf{x}$ directly, instead of to the mesh velocities $\dot{\mathbf{x}}$; in the end they seem to prefer the MMPDE to the GCL approach.

Time-dependent applications

Recently, the GCL-based approach has gained renewed attention in work by Baines and coworkers [BHJ05, BHJJ06]. They use it to solve general time-dependent PDEs (2.68) on a time-dependent domain $\Omega(t)$. The solution-dependent monitor function is rescaled to a time-dependent monitor $\omega(q,t)$ such that it satisfies the equidistribution principle:

$$\omega_{\text{avg}}(q(\mathbf{x},t),t) \cdot |A(t)| = c_A, \tag{2.89}$$

where $|A(t)|$ is the area of a mesh cell that moves in time and ω_{avg} is the averaged monitor on that cell and c_A is time-*independent*. This is a generalized form of equidistribution principle (2.20). Also note that the monitor function now has an explicit dependence on the solution q, as opposed to the original form $\omega(\mathbf{x})$. Baines et al. solve the mesh movement by finite elements, so they use a weak form, now called the *scale-invariant conservation principle*:

$$\int_{\Omega(t)} \psi(\mathbf{x},t)\omega(q(\mathbf{x},t),t)\,\mathrm{d}\mathbf{x} = c_0(\psi), \tag{2.90}$$

where the test function $\psi(\mathbf{x},t)$ has local support $\Delta\Omega$ and the constant $c_0(\psi)$ is again time-independent.

The time derivative of the monitor ω can be expressed in terms of the solution:

$$\frac{\partial \omega}{\partial t} = \frac{\partial \omega}{\partial q}\frac{\partial q}{\partial t}. \tag{2.91}$$

The conservative form of the GCL (2.85) now gets coupled to the physical PDE:

$$\nabla \cdot (\omega \dot{\mathbf{x}}) = -\frac{\partial \omega}{\partial q}\mathcal{L}(q). \tag{2.92}$$

Just like Huang and coworkers the authors use an irrotational mesh velocity ($\dot{\mathbf{x}} = \nabla\phi$). The original PDE is transformed into ALE-form (2.94), which we

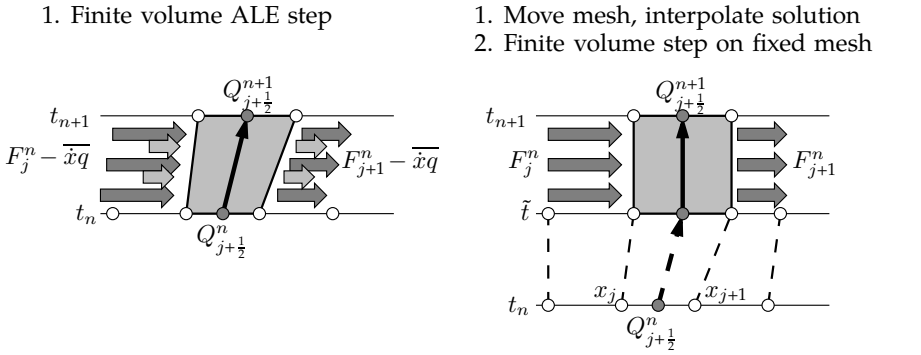

Figure 2.6 ALE formulation versus solution interpolation for finite volumes in 1D. The ALE formulation of a PDE incorporates the mesh velocity into the flux.

discuss in Section 2.4.1. We refer to the original publication [BHJJ06] for the weak forms of the mesh velocity PDE (2.91) and physical PDE (2.94) and their finite element solutions.

The results for compressible fluid problems [WBG05] appear very smeared, but this is apparently due to the first-order HLLC-solver. Recent work on phase-change problems [BHJM09] shows well-adapted meshes and second-order accurate results.

2.4 *R*-REFINEMENT: COUPLING WITH PHYSICAL PDES

The solution of time-dependent PDEs using any of the preceding moving mesh approaches requires the mesh equations and physical PDEs to be combined in one way or another. The moving mesh cells move 'underneath' the solution manifold on the physical domain, so the discrete solution values at the nodes need to be updated accordingly. Two approaches can be distinguished. The first incorporates the mesh velocity into the physical PDEs. The time integration of the new PDEs applies the change in solution due to physical differential operators and the mesh movement in one step. The second approach moves the mesh one step while 'freezing' the physical time and then interpolates the solution onto the new mesh nodes. The second step then 'freezes' the nonuniform mesh points and integrates the original physical PDEs for one time step. The following two sections discuss these approaches. Tang [Tan05] calls them 'interpolation-free' and 'interpolation' approaches, respectively, in his review paper. Figure 2.6 shows the two approaches schematically for a finite volumes approach in one dimension.

2.4.1 *Transformed physical PDEs*

The mesh velocity is now incorporated into the physical PDEs. The velocity-based approaches in Section 2.3.3 have a mesh velocity $\dot{\mathbf{x}}$ by definition. The location-based approaches in Section 2.3.2 solve their generator equations for

$\mathbf{x}$ in an iterative way. The difference between the old and new points could be used as an approximate velocity:

$$\dot{\mathbf{x}} \approx \frac{\mathbf{x}^{\text{new}} - \mathbf{x}^{\text{old}}}{\Delta t}, \tag{2.93}$$

although this is less common.

Again consider the general PDE (2.68). Using the chain rule for the time derivative of solution $q(\mathbf{x}(\boldsymbol{\xi}, t), t)$, we get the Lagrangian form of this PDE:

$$\dot{q} - \nabla q \cdot \dot{\mathbf{x}} = \mathcal{L}q. \tag{2.94}$$

This approach is called an arbitrary Lagrangian Eulerian (ALE) method. The method is not fully Lagrangian, because the mesh velocity is not by definition equal to the flow characteristics (as in Section 2.3.3.1). It is also not Eulerian, because it uses a moving reference frame.

Coupled vs. uncoupled

Most velocity-based approaches and the ones using MMPDEs combine the mesh velocity equations with the physical PDEs, i.e.:

$$\begin{cases} \dot{q} = \nabla q \cdot \dot{\mathbf{x}} + \mathcal{L}q, \\ \dot{\mathbf{x}} = \dots, \end{cases} \tag{2.95}$$

where the right hand side for $\dot{\mathbf{x}}$ depends on the chosen moving mesh approach. By the method of lines (MOL), spatial derivatives are first discretized using finite differences or finite elements. This results in a system of ordinary differential equations (ODEs) for the time integration. The natural time scale of the mesh movement may be very different from the one of the physical solution, which makes the coupled nonlinear ODE system extremely stiff.

The same can happen for the location based methods. Consider for example one of the equidistribution approaches. When discretized, this forms an algebraic equation for the mesh point locations. Together with the ODEs for the physical solution, this forms a system of differential algebraic equations (DAE). More specifically, an index-2 DAE system as Li et al. [LPR99] show. This means that the discretized equidistribution equations need to be differentiated twice before fitting into one ODE system with the physical PDEs. Again, a very stiff system is the result.

The stiffness of the coupled system can be reduced by a relaxation of the time scale. In the first design of their MMPDEs, Huang et al. [HRR94] already included a scalar time relaxation parameter τ. For example, their (one-dimensional) MMPDE6 is relaxed as follows:

$$\frac{\partial^2 \dot{x}}{\partial \xi^2} = -\frac{\partial}{\partial \xi}\left(\omega \frac{\partial x}{\partial \xi}\right) \xrightarrow[\text{relaxation}]{\text{time}} \frac{\partial^2 \dot{x}}{\partial \xi^2} = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(\omega \frac{\partial x}{\partial \xi}\right). \tag{2.96}$$

Originally, the authors claim that the numerical methods are not very sensitive to choice of τ. Recently, though, Soheili and Stockie [SS08] show that for

solutions that have a natural time scale which changes significantly over time, a constant value for τ can result in bad mesh adaptation. When τ is increased, the stiffness decreases, but the adaptation becomes less sharp (the mesh points may in fact 'lag behind'). When τ is decreased, the results are more accurate, but the system becomes stiffer. They propose a time dependent relaxation parameter $\tau(t)$ which is specific for the chosen MMPDE. This new approach is very effective, although it is still specific for self-similar blow-up problems only.

Contrary to their coupled approach in one dimension, Huang et al. decouple the mesh and physical PDEs for two-dimensional problems [HR99]. The two systems are solved alternately, where (2.93) defines the mesh velocity for use in the ALE form (2.94). An argument for the relaxation of mesh generator equations is the fact that the mesh point locations need not be solved up to the same accuracy as is desirable for the physical PDEs.

Discrete Geometric Conservation Law

Decoupled or not, in the transformed ALE formulation of PDEs the stability of numerical solvers may not be guaranteed. For finite volume methods, the crucial point is the time integration of the fluxes across the moving cell edges. Farhat and coworkers have investigated the effects of satisfying a discrete geometric conservation law (DGCL) on a method's stability for several types of PDEs. For the nonlinear Euler equations they prove that the satisfaction of a DGCL is a necessary and sufficient condition for a moving grid numerical scheme to maintain the nonlinear *stability* of its fixed grid counterpart [FGG01]. They also prove, though, that satisfaction of a DGCL is neither a necessary nor a sufficient condition for a scheme to maintain the order of time-*accuracy* of its fixed-grid counterpart [GGF03].

Consider a system of PDEs in conservation form, i.e., $\mathcal{L}\mathbf{q} = \nabla \cdot \mathcal{F}(\mathbf{q})$, where $\mathcal{F}$ is the multidimensional flux tensor. The ALE formulation (2.94) can be brought back close to conservative form by multiplying it by the Jacobian and rewriting it to:

$$(\dot{J}q) + J\nabla \cdot (\mathcal{F}(\mathbf{q}) - \dot{\mathbf{x}}\mathbf{q}) = 0. \tag{2.97}$$

Integrating this over a computational cell A_c and using the Jacobians to transfer to the associated physical cell $A_{j,k}$ yields:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{A_{j,k}(t)} \mathbf{q}\,\mathrm{d}\mathbf{x} + \int_{\partial A_{j,k}(t)} (\mathcal{F}(\mathbf{q}) - \dot{\mathbf{x}}\mathbf{q}) \cdot \mathbf{n}(t)\,\mathrm{d}s = 0. \tag{2.98}$$

Notice that the second integral for the fluxes across the edges is now time-dependent. The time-discretization has to include some way of averaging three quantities: edge lengths $|\partial A_{j,k}(t)|$, edge normals $\mathbf{n}(t)$ and velocity of the edge $\dot{\mathbf{x}}(t)$. These averages are marked by lines above them hereafter.

A DGCL is *specific* for the discretization scheme used. It follows from the requirement that a constant solution should be reproduced exactly by the

discretized scheme. Consider for example, the Euler–Backward scheme:

$$\left|A_{j,k}^{n+1}\right|\mathbf{q}_{j,k}^{n+1} = \left|A_{j,k}^{n}\right|\mathbf{q}_{j,k}^{n} - \Delta t \sum_{l=1}^{4} \left|\overline{\partial A}_{j,k,l}\right| \Phi(\mathbf{q}_{j,k}^{n+1}, \mathbf{q}_{j,k,l}^{n+1}, \overline{\mathbf{n}}_{j,k,l}, \overline{\dot{\mathbf{x}} \cdot \mathbf{n}}_{j,k,l}), \quad (2.99)$$

where Φ defines the numerical flux of the second integrand of (2.98), which is integrated across all four edges of the cell. The DGCL follows after some rewriting (cf. [FGG01]):

$$\left|A_{j,k}^{n+1}\right| - \left|A_{j,k}^{n+1}\right| = \Delta t \sum_{l=1}^{4} \left|\overline{\partial A}_{j,k,l}\right| \overline{\dot{\mathbf{x}} \cdot \mathbf{n}}_{j,k,l}. \quad (2.100)$$

Notice how this looks like a discretized version of the original GCL (2.81). Again, this may be different for other discretization schemes. The choice for the averaged quantities determines whether the scheme violates its DGCL or not. If so, spurious oscillations may occur, yet the scheme may still maintain its order of time-accuracy. Recently, Mackenzie and Mekwi [MM07] investigated a θ-time integration method on contracting and expanding one-dimensional domains, for which they use an adaptive parameter θ to satisfy the DGCL, thus achieving unconditional stability and asymptotic second-order accuracy.

2.4.2 *Transformed solution*

The stiff, combined system can be avoided by solving the mesh equations separately from the physical PDEs and not incorporating the mesh velocity into them. The new mesh locations are computed first, after which the discrete solution values are interpolated onto the new mesh. Next, a finite volume step evolves the solution while keeping the mesh points fixed. This is shown in the right scheme in Figure 2.6.

For systems of conservation laws, the interpolation step should maintain the conservation property. Tang and Tang [TT03] presented a conservative interpolation based on mesh velocities. Later Han and Tang proposed a slightly modified formulation based on a geometrical interpretation. Both maintain global conservation in the following discrete sense:

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

We refer to Chapters 3 and 4 for a full specification of the interpolation.

The subsequent time integration step—e.g., by finite elements or finite volumes—does not need to be modified for the mesh movement whatsoever. It just advances the solution one time step on a nonuniform mesh. If the interpolation step is similar to the finite volume step, the *decoupled* ALE approach and the interpolation approach are practically equal. The conservative interpolation (see (3.13) and (4.32)) is in fact a finite volume step with the artificial flux $\dot{\mathbf{x}} \cdot \nabla \mathbf{q}$, which is exactly the correction flux in the ALE formulation.

2.5 *R*-REFINEMENT: CONTROL BY MONITOR FUNCTIONS

The generator equations of the various approaches for adaptive mesh movement are diverse, as we showed in Section 2.3. Most of them have some solution-dependent weight function in common, though, which is generally called a monitor function.

A loose classification of monitor functions could be 'practical–theoretical'. Many publications on solution adaptivity define monitor functions in terms of solution gradients or curvatures. These monitors are generally applied to time-dependent PDEs where solutions are unknown and error analysis is complicated. The more theoretical studies of monitor functions define them in terms of a posteriori error estimates. Generally, these monitors are applied to prescribed solutions. The error estimate is then an interpolation error, which makes analysis more feasible.

2.5.1 *Monitoring based on solution gradients*

Arc length monitoring

The unit of arc length of a scalar solution in one dimension is given by its first derivative: $\mathrm{d}s := \sqrt{\mathrm{d}x^2 + \mathrm{d}y^2} = \sqrt{1 + (\mathrm{d}y/\mathrm{d}x)^2}\mathrm{d}x$. The arc length(-based) monitor is named after it:

$$\omega(q) := \sqrt{1 + \alpha q_x^2}. \tag{2.101}$$

The floor value of 1 prevents mesh cells of infinite size and the parameter α has to be chosen by the user depending on the behavior of the solution q. The form shown above is very basic, but variants of it are used in the majority of publications on solution adaptivity.

Second-order derivatives

Strictly speaking, it is the *curvature* or second derivative of a function that forms the main numerical error of linear approximations. It is logical to think of a curvature-based monitor function, e.g.,

$$\omega(q) := \sqrt[4]{\alpha + q_{xx}^2}. \tag{2.102}$$

This is the monitor that Blom and Verwer [BV89] compare to an arc length monitor like (2.101). The results for the curvature monitor are indeed more accurate, for some examples significantly so. However, the ODEs for the semi-discretized coupled Lagrangian system (see page 43) are very hard to solve numerically. The Newton iteration repeatedly fails to converge, after which a restart with significantly smaller time steps needs to be made. The coordinate lines for the curvature monitor are often oscillatory, in spite of the applied mesh smoothing, whereas the arc length monitor shows smooth mesh movement for all cases.

We have also experienced that the smallest oscillations in a numerical solution immediately attract mesh points with a curvature monitor, after which

the oscillations can grow even more. A monitor based on solution gradients does not suffer from this problem.

For a shock wave, or jump in the solution the arc length monitor will yield refinement *in* the shock, whereas the curvature monitor will yield refinement at the *start and end* of the shock. In practice—especially with monitor filtering (Section 2.5.5)—these regions will overlap. Therefore, first order derivatives are more viable monitor components for solution adaptivity in flow simulations.

The combination of both arc length and curvature monitors is also used, for example by Dvinsky [Dvi91] and Tang et al. [TTZ03]. Di et al. [DLTZ07] add the curvature of a level set function to the monitor in the context of two-phase flows. This is not motivated by truncation errors, though, but by the need to refine at places where the interface between the two media develops features such as corners and cusps.

Blow-up problems can be treated in a fundamentally different way. Budd et al. [BHR96] study blow-up problems where the solution—despite being unknown—has a known built-in scaling-invariance. They choose the monitor function such that the resulting MMPDE remains invariant under the same scaling, namely:

$$\omega(q) := q^{p-1}, \tag{2.103}$$

where p comes from the source term in the original PDE: $q_t - q_{xx} = q^p$. If the natural time scale of the MMPDE to reach equidistribution is now at least smaller than the natural time scale of the evolution of the solution structure, then the mesh can keep up even during the faster and faster growth of the implosion. An arc length or curvature monitor would stop adapting after a certain time.

Automated balancing of monitor values

The standard arc length monitor performs well on solutions with only one distinct feature, but for a range of solution features that differ in size and over time it quickly falls short. As mentioned before, the necessity to properly choose parameter α for each new problem is inconvenient. More importantly, for unsteady problems, the solution features will often change over time, to such an extent that the original choice for α becomes unsuitable.

Beckett and Mackenzie [BM00] were the first to propose a solution-dependent, hence time-dependent parameter $\alpha(q)$. Both Huang [Hua01a] and Van Dam and Zegeling [vDZ06] include a dimensionless parameter $\beta \in (0, 1)$ that defines the relative amount of refinement. The balanced monitor function takes the following form in one dimension for a scalar equation:

$$\omega(\phi) := \frac{(1-\beta)}{\beta}\alpha(\phi) + \phi, \tag{2.104}$$

where the monitor component ϕ is generally, but not necessarily, a solution gradient and the time-dependent scaling parameter $\alpha(\phi)$ is given by the av-

erage value:

$$\alpha(\phi) := \langle\phi\rangle = \int_{\Omega_c} \phi \, \mathrm{d}\xi. \tag{2.105}$$

We now show that β defines the relative amount of refinement initiated by the monitor component ϕ. We integrate the monitor function (2.104) over the entire reference domain:

$$\int_{\Omega_c} \omega \, \mathrm{d}\xi = \int_{\Omega_c} \frac{1-\beta}{\beta} \langle\phi\rangle \, \mathrm{d}\xi + \int_{\Omega_c} \phi \, \mathrm{d}\xi.$$

The parameter β is a constant, hence:

$$\begin{aligned} \beta \int_{\Omega_c} \omega \, \mathrm{d}\xi &= (1-\beta) \int_{\Omega_c} \langle\phi\rangle \, \mathrm{d}\xi + \beta \int_{\Omega_c} \phi \, \mathrm{d}\xi \\ &= (1-\beta) \int_{\Omega_c} \phi \, \mathrm{d}\xi + \beta \int_{\Omega_c} \phi \, \mathrm{d}\xi. \end{aligned} \tag{2.106}$$

Finally, we see that β is the ratio of the integral over the nonzero values of ϕ and the total monitor function:

$$\beta = \frac{\int_{\Omega_c} \phi \, \mathrm{d}\xi}{\int_{\Omega_c} \omega \, \mathrm{d}\xi}. \tag{2.107}$$

By equidistribution, ω is inversely proportional to the mesh density $1/J$. Therefore, the ratio (2.107) is an indicator for the relative amount of mesh points in regions of large ϕ.

We have found that the standard summation of multiple components:

$$\omega := \sum_{p=1}^{P} \omega(\phi_p)$$

can lead to unexpected and unsatisfactory results. The problem is that a monitor component with a very small average value and at the same time very large maximum value will dominate all other monitor components. It makes sense to make sure that the values for all monitor components are in the same range before summing them:

$$\omega(\phi) := \frac{(1-\beta)\langle\phi\rangle + \beta\phi}{(1-\beta)\langle\phi\rangle + \beta \max_{\Omega_c} \phi}. \tag{2.108}$$

Note that for a single monitor component ($P = 1$) the original form (2.104) and this new balanced form are equivalent, due to scaling invariance of the monitor function: the adaptive mesh that is the solution to one of the equidistribution-based generator equations for a given monitor ω is the same

as the mesh for that monitor multiplied by a constant: $c \cdot \omega$. We elaborate on this monitor balancing in Chapter 4.

Extension to multiple dimensions is straightforward, either by taking the norm of the gradient vector (nondirectional) or by taking respective spatial derivative in each direction (directional), see Section 2.5.2.

2.5.2 *Directional monitoring*

The monitor functions discussed so far are all scalar functions, which for large monitor values results in attraction of mesh points from all surrounding directions. In view of the truncation errors it is sensible to align mesh lines with user-defined vector fields. Besides, different degrees of adaptivity in different direction is beneficial. Section 4.4.4 contains a motivating example for this. The term 'directional' means the same as 'anisotropic', which is often used in research on h-refinement. Similarly, 'non-directional' and 'isotropic' have the same meaning.

Mesh alignment

Giannakopoulos and Engel [GE88] are the first to give a variational formulation that involves mesh alignment. Brackbill [Bra93] uses the same approach, but with a solution-adaptive weight. The alignment or directional functional is now:

$$I_{\mathrm{d}} := \int_{\Omega} \frac{1}{\omega} \left((\mathbf{a} \times \nabla\xi)^2 + (\mathbf{b} \times \nabla\eta)^2 \right) \mathrm{d}\mathbf{x}, \tag{2.109}$$

where $\mathbf{a}$ and $\mathbf{b}$ are user-defined vector fields. In two dimensions, a cross product is defined as follows:

$$\begin{bmatrix} a_1 \\ a_2 \\ 0 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix} := \begin{bmatrix} 0 \\ 0 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}. \tag{2.110}$$

The square of a vector is just its 2-norm squared: $\mathbf{a}^2 := \|\mathbf{a}\|_2^2 = \mathbf{a} \cdot \mathbf{a}$. The solution to the associated Euler–Lagrange equations is the mesh map for which the cross products vanish when the contravariant normal $\nabla\xi$ aligns with vector field $\mathbf{a}$ and similar for the other direction. Figure 2.4 depicts this normal vector of the iso-lines of ξ. In a direct, covariant setting, the logically horizontal edges ($\mathbf{x}_\xi$, η = constant) will align with $\mathbf{b}_\perp$, the vector field that is perpendicular to $\mathbf{b}$. In a three-dimensional setting where $\mathbf{c}$ is the third vector field, $\mathbf{x}_\xi$ aligns with $\mathbf{b} \times \mathbf{c}$, $\mathbf{x}_\eta$ aligns with $\mathbf{a} \times \mathbf{c}$ and $\mathbf{x}_\zeta$ aligns with $\mathbf{a} \times \mathbf{b}$.

When the *direction* of $\nabla\xi$ aligns perfectly with its vector field ($(\mathbf{a} \times \nabla\xi)^2 = 0$), the directional functional (2.109) is minimized with reference to ξ no matter what the *size* of that same normal vector $\nabla\xi$ is. Hence, the system has no unique solution. Therefore, alignment is generally combined with the weighted smoothness functional (2.30):

$$I_{\text{Brackbill}} := (1 - \lambda) I_{\mathrm{s}} + \lambda I_{\mathrm{d}}, \tag{2.111}$$

for some constant $\lambda \in (0, 1)$. Brackbill demonstrates the combined functional on unsteady ideal magnetohydrodynamics in two dimensions. The alignment

is defined by the normalized magnetic field and its orthogonal counterpart: $\mathbf{a} := \mathbf{B}$, $\mathbf{b} := \mathbf{B}_\perp$. As the two fields are perpendicular, the resulting mesh also has a good orthogonality, but this does not hold for arbitrary fields $\mathbf{a}$, $\mathbf{b}$ (and $\mathbf{c}$). Glasser et al. [GLK05] give some examples of this for prescribed vector fields. To limit the mesh skewness, a third functional can always be included, as Brackbill and Saltzmann did, see (2.33).

Finally, note that the choice of vector fields requires some advance knowledge on the solution. In the case of Brackbill's magnetic reconnection example, the magnetic field was more 'horizontal' than 'vertical', so alignment of $\nabla\xi$—as opposed to $\nabla\eta$—with it was the right choice. For a more vertically directed magnetic field, the choice for $\mathbf{a}$ and $\mathbf{b}$ would have to be switched, to prevent strong rotation of mesh cells.

Directionality

The concept of *directionality* is more general than *alignment*. It involves monitor matrices that are not equivalent to a scalar monitor: $G \neq \omega I$. Cao et al. [CHR99] have investigated the effect of directional monitoring in terms of the matrix's eigendecomposition.

Consider the functional (2.41), which incorporates the MMPDE, harmonic mapping and variable diffusion approaches. For the moment, take $G_1 = G_2 = G$. This monitor matrix can be written in terms of its eigenvalues and normalized (orthogonal) eigenvectors:

$$G = \lambda_1 \mathbf{v}_1^T \mathbf{v}_1 + \lambda_2 \mathbf{v}_2^T \mathbf{v}_2. \tag{2.112}$$

The Euler–Lagrange equations in a $\mathbf{v}_1, \mathbf{v}_2$-reference frame are of the following form:

$$\frac{\partial^2 \xi}{\partial \mathbf{v}_1^2} + \left(\frac{\lambda_1}{\lambda_2}\right) \frac{\partial^2 \xi}{\partial \mathbf{v}_2^2} + \text{'2D effects'} = \frac{1}{\lambda_1} \frac{\partial \lambda_1}{\partial \mathbf{v}_1} \frac{\partial^2 \xi}{\partial \mathbf{v}_1^2}, \tag{2.113a}$$

$$\frac{\partial^2 \eta}{\partial \mathbf{v}_1^2} + \left(\frac{\lambda_1}{\lambda_2}\right) \frac{\partial^2 \eta}{\partial \mathbf{v}_2^2} + \text{'2D effects'} = \frac{1}{\lambda_1} \frac{\partial \lambda_1}{\partial \mathbf{v}_1} \frac{\partial^2 \eta}{\partial \mathbf{v}_1^2}, \tag{2.113b}$$

where the directional derivatives are defined as $\partial/\partial \mathbf{v}_i := \mathbf{v}_i \cdot \nabla$, and the authors collect the not so well understood terms in the '2D effects'. We refer to the original paper [CHR99] for more details. Using Green's function, the authors show the role of the source terms in the above equations. An increase of λ_1 in the direction of $\mathbf{v}_1$ followed by a decrease results in compression of both ξ and η in the same direction $\mathbf{v}_1$. The same holds for variations in λ_2 in the $\mathbf{v}_2$ direction. The mesh refinement will definitely have preference for these two directions, but no strict alignment of mesh lines occurs. Also, the mesh cells may become very skew, since both ξ and η are compressed equally. The second term in (2.113a) tends to space coordinate lines equally, hence it has a strong smoothing effect on the mesh when λ_1/λ_2 is large.

A monitor matrix inspired by the arc length monitor is defined by its eigendecomposition (2.112) with:

$$\begin{cases} \mathbf{v}_1 := \mathbf{v} = \nabla q/|\nabla q|, & \mathbf{v}_2 := \mathbf{v}_\perp, \\ \lambda_1 := \sqrt{1+|\nabla q|^2}, & \lambda_2 := \text{a function of } \lambda_1. \end{cases} \tag{2.114}$$

Winslow's non-directional mesh adaptation (Section 2.3.2.3)has $\lambda_2 = \lambda_1 = \omega$ and it has no preference for any direction. This results in perfectly symmetric meshes for symmetric functions ω. A harmonic mapping (Section 2.3.2.6) results for $\lambda_2 = 1/\lambda_1$. The interplay of the elliptic terms and the other two-dimensional effects is so much intricate that the resulting mesh remains hard to predict, though.

Recently, Huang [Hua06] provided a new motivation for the equidistribution and alignment conditions in terms of monitor matrices, which he introduced in [Hua01b]. Firstly, the singular value decomposition (SVD) of the Jacobian matrix $\mathcal{J}$ controls the change in *size*, *shape* and *orientation* of local cells by the mesh map. The first two are dictated by the singular values σ_i, the latter by the left SVD vectors $\mathbf{u}_i$. Secondly, the eigenvalues of the matrix $\mathcal{J}^{-T}\mathcal{J}^{-1}$ are equal to σ_i^{-2} and its eigenvectors are the same $\mathbf{u}_i$. Therefore, finding (the eigendecomposition of) $\mathcal{J}^{-T}\mathcal{J}^{-1}$ gives total control over the mesh adaptation. Finally, it turns out that satisfying the following relation allows one to control the adaptation entirely through a monitor matrix M:

$$\mathcal{J}^{-T}\mathcal{J}^{-1} = \Big(\frac{\sigma}{|\Omega_c|}\Big)^{-\frac{2}{n}} M(x), \tag{2.115}$$

where M is $n \times n$ symmetric and positive definite and

$$\sigma := \int_\Omega \rho(\mathbf{x})\,\mathrm{d}\mathbf{x}, \quad \rho := \sqrt{\det(M)}.$$

The control over size is through a multi-dimensional generalization of equidistribution, obtained by taking the determinant of (2.115):

$$\rho J = \frac{\sigma}{|\Omega_c|} = \text{constant}. \tag{2.116}$$

Note that for Winslow's nondirectional method $\rho = \omega$. Next, relation (2.115) is equivalent with $\mathcal{J}^T M \mathcal{J} = (\sigma/|\Omega_c|)^{\frac{2}{n}} I$, in other words: all its eigenvalues are equal. This is only the case when their arithmetic mean is equal to their geometric mean, or in terms of the matrix itself:

$$\frac{1}{n}\mathrm{tr}\big(\mathcal{J}^T M \mathcal{J}\big) = \det\big(\mathcal{J}^T M \mathcal{J}\big)^{\frac{1}{n}}. \tag{2.117}$$

Huang shows that this *alignment condition* is equivalent to $\mathcal{J}^{-T}\mathcal{J}^{-1} = \theta(\mathbf{x})M(\mathbf{x})$ for some scalar function θ, in other words: the eigenvectors and ratios of eigenvalues of $\mathcal{J}^{-T}\mathcal{J}^{-1}$ are completely determined by the monitor matrix M.

Therefore, the equidistribution and alignment conditions (2.116) and (2.117) are necessary and sufficient for total control over size, shape and orientation of mesh elements.

Now that the role of the monitor matrix M is clear, its definition based on error estimates is discussed in the next section.

2.5.3 *Error-based monitoring*

Huang [Hua06] has performed a thorough analysis of local interpolation error estimates, generalizing over the smoothness of the functions, the order of interpolation, dimensionality of the domain and the p-norm. Basically, the resulting monitor matrix can be based on gradients:

$$M := r_{\text{ani},1}(\rho)\Big[I + \frac{1}{\alpha^2_{\text{ani},1}}\nabla q \nabla q^T\Big], \tag{2.118}$$

where the scaling function $r_{\text{ani},1}$ of the determinant $\rho := \sqrt{\det(M)}$ enables equidistribution (2.116) and $\alpha^2_{\text{ani},1}$ is similar to the solution-dependent $\alpha(\phi)$ in (2.105). Alternatively, the monitor matrix can be based on second-order derivatives:

$$M := r_{\text{ani},2}(\rho)\Big[I + \frac{1}{\alpha^2_{\text{ani},2}}|H(q)|\Big], \tag{2.119}$$

where $H(q)$ is the Hessian matrix with second-order derivatives of solution variable q.

The above two monitors are anisotropic (directional), which greatly improves the results when the solution has strong anisotropic features. Alternatively, isotropic (nondirectional) variants of the above monitors can be used (see also [HS03]):

$$M := \rho^{\frac{1}{n}} I, \tag{2.120}$$

where the determinant ρ contains either the norm of the gradient or Hessian, or even higher-order derivatives.

The numerical results for linear interpolation are best when the Hessian-based monitor is used, but only for prescribed solutions $q(\mathbf{x}, t)$. For actual PDEs, no comparison between gradient-based and Hessian-based monitors is shown, but at least the anisotropic monitor functions always outperform the isotropic ones.

2.5.4 *Mesh qualities*

The elliptic generator equations have a built-in smoothness that tends to produce good meshes. However, the stronger the adaptivity becomes, and the more anisotropic it is, the more stretching and skewing is applied to the mesh. It is possible to add mesh quality measures to the variational minimization problems that compete with the alignment and adaptivity measures.

Brackbill and Saltzman originally included an orthogonality measure (Section 2.3.2.5), which we repeat here for completeness:

$$I'_{\mathrm{o}} := \int_{\Omega} (\nabla\xi \cdot \nabla\eta)^2 J^3 \,\mathrm{d}\mathbf{x}. \tag{2.121}$$

The alignment condition (2.117) can be used as a measure for skewness when no alignment is prescribed ($M := I$). It is not exactly satisfied, but the ratio of the left and right hand sides should be as close as possible to the minimal value of one:

$$I_{\mathrm{sk}} := \int_{\Omega} \left[\frac{\mathrm{tr}(\mathcal{J}^{-1}\mathcal{J}^{-T})}{n \det(\mathcal{J}^{-1}\mathcal{J}^{-T})^{\frac{1}{n}}} \right]^{\frac{2}{2(n-1)}} \mathrm{d}\mathbf{x}, \tag{2.122}$$

We have found that for finite volume solvers, the possible skewness of mesh cells is not that big a problem. The net fluxes through cell edges are namely always projected onto the edge normal. The stretching factor between neighboring cells appears more important, which is controlled by balanced monitoring and monitor filtering.

2.5.5 *Monitor filtering*

Monitor filtering—or monitor smoothing—is the application of a low-pass filter to the discrete monitor values in order to spread the mesh refinement somewhat. In two dimensions, the following Gaussian filter is common:

$$\begin{aligned}\omega_{j,k}^{\mathrm{filter}} := \frac{1}{4}\,\omega_{j,k} + \frac{1}{8}\,(\omega_{j-1,k} + \omega_{j+1,k} + \omega_{j,k-1} + \omega_{j,k+1}) \\ + \frac{1}{16}(\omega_{j-1,k-1} + \omega_{j+1,k-1} + \omega_{j-1,k+1} + \omega_{j+1,k+1}).\end{aligned} \tag{2.123}$$

The regions that require mesh refinement are often very thin, for example across shock lines. Adjacent mesh cells can easily be a factor of $\mathcal{O}(10)$ larger, according to equidistribution. In other words: the stretching factor of adjacent mesh cells $\Delta x_{j+1}/\Delta x_j$ can be very large (or very small, evidently). For robustness it is better to have a smooth transition between the large cells and the extremely fine cells, which is implicitly accomplished by filtering the monitor values. The filter (2.123) is generally applied several times.

Veldman and Rinzema [VR92] show that the accuracy of a discrete solution is not very well predicted by the local truncation error. In that context they also show how large stretching factors result in a strongly stiff problem that easily becomes unstable. Verwer et al. [VBFZ89] keep the stretching ratio bounded:

$$\frac{\kappa}{\kappa+1} \leq \frac{\Delta x_{i+1}}{\Delta x_i} \leq \frac{\kappa+1}{\kappa},$$

where κ is user-defined, e.g., $\kappa = 2$. They do so by a spatial smoothing of the discrete monitor values through diffusion. Huang and Russell [HR97b] do the same in two dimensions: an artificial diffusion term $\Delta_\xi \omega$ that smoothens the monitor. Its effect is related to the above discrete filter (2.123).

Many authors report the use of monitor filtering, e.g., [BMRS01, HZZ02, TTZ03]. However, the increased smoothness of our balanced monitor functions makes the filtering less necessary; we limit it typically to 0, 1 or 2 applications.

2.6 LOCAL OR H-REFINEMENT

r-Refinement—or adaptive mesh refinement (AMR)—is often compared with h-refinement, as both have the same goal—solution-adaptive improvement of resolution to achieve higher accuracy—and both can be integrated with finite volume- or element-based solvers. This section provides no complete overview of all research that involves h-refinement, but it does identify the limitations and advantages of both h- and r-refinement. The following three subsections discuss the same three aspects of h-refinement as we did for r-refinement in Sections 2.3–2.5. As the (dis)advantages of both methods are mostly disjoint, the suggestion for a hybrid combination of the two is obvious.

2.6.1 *Refinement and coarsening*

The formulation of local refinement itself is fairly straightforward, as opposed to the diverse approaches for mesh movement in Section 2.3. A mesh cell can be split into a number of smaller cells, generally two in each direction. A rectangular cell (in two dimensions) thus gets divided into four smaller cells. On unstructured meshes composed of triangles the same is possible.

Unsteady physical problems have changing flow phenomena over time, so locally refined cells may need to be brought back to their original state. This is called *coarsening* of mesh cells. Both the refinement and coarsening can be applied recursively, so the locally refined cells form a nested set of discretization levels. The initial mesh could be called the 'level 0' mesh. Most implementations of h-refinement allow the user to set a maximal refinement level, which puts an upper limit to the total number of mesh cells. Due to the recursive nature of the refinement, this number may become unexpectedly large anyway, resulting in a very long running time. Proper refinement criteria should prevent this.

Berger and coworkers have done extensive research on AMR. They allow for arbitrary rectangular regions in the domain to have increased refinement-levels, as long as the rectangles remain properly nested [BC89]. Van der Holst and Keppens [vdHK07] call this *patch-based* refinement levels. On the opposite side, the most structured form of nested refinement is when a cell is split into 2^n cells, such that a perfectly nested and symmetric tree results. A hybrid form that is block-structured, but allows for anisotropic refinement and incomplete grid families (equivalent to hanging nodes) combines the best of both approaches with an increased efficiency of up to 30% on example problems. Trompert [Tro94] has investigated isotropic local grid refinement for various problems. Systems of linear PDEs allow for proper error analysis and it turns out that there can indeed be a big difference between local error

estimates (e.g., second-order derivatives) and global errors. When applied to unsteady ground water flow PDEs, a curvature monitor is used [TVB93].

The virtually unlimited possible accuracy makes local refinement an extremely powerful method, especially for detecting local phenomena. Mesh movement, on the other hand, aligns with flow features and thereby reduces dispersive errors, e.g., near wave fronts.

2.6.2 *Embedding local refinement into the physical PDE solver*

The subdivision of mesh cells is extremely pragmatic and intuitive. The incorporation of this refinement into a physical PDE solver requires some additional measures, though. Firstly, the addressing of discrete cells and solution values can no longer be done through a standard matrix-like data structure, but this is fairly easily remedied with nested data structures.

Secondly, in the case of flow problems flux functions need to be evaluated across cell edges (across faces in three dimensions), but neighboring cells will often have different levels of refinement. The total flux from the large cell is spread over its neighboring smaller cells. In the opposite direction, all flux values from the smaller cells are summed up to one total flux into the larger cell. Generally, neighboring cells are allowed to differ by at most one refinement level, such that no extremely large differences in local resolution occur. This is called the proper nesting condition.

Thirdly, and strongly related to the previous point, the finer cells at higher levels of refinement require a smaller time step, dictated by the stability criterion. As a result, local time stepping is an obvious choice. For one time step at some coarse level, multiple time steps are made at the finer level. In order to maintain global conservation for conservative flow problems, the total flux into the coarser cell is corrected with the more accurate fluxes over the smaller time steps from the finer neighboring cells. Local time stepping greatly improves the efficiency of the solver and it can also be applied to r-refinement methods (see, e.g., Tan et al. [TZHT04]).

2.6.3 *Refinement criteria*

Refinement and coarsening should be initiated by some error indicator exceeding a given tolerance value, as opposed to the relative monitor values used for r-refinement. In the equidistribution-related approaches in Section 2.3, only the *relative* differences in monitor values was important for mesh adaptions: the mesh was invariant under monitor scaling. For h-refinement, the error indicator is used at each cell independently and at all refinement levels in the same way.

A well-known error indicator is obtained through Richardson interpolation. Solutions values are averaged to a one level coarser mesh and then advanced one time step. The result is compared to the solution value obtained by first advancing and *then* averaging. Bell et al. [BBSW94] additionally include a spatial error measure to capture stationary features such as contact discontinuities, as these are hardly detected by the Richardson estimate.

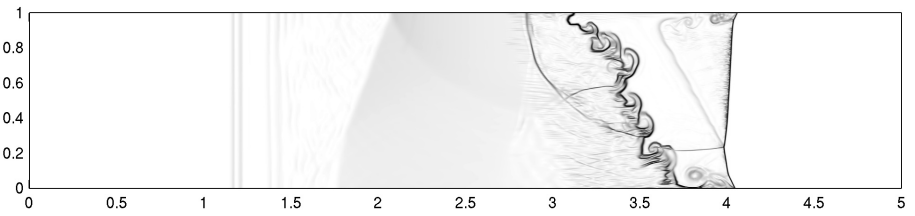

Figure 2.7 Schlieren plot for the density in the HD22RM problem.

In finite element methods, the inherent a posteriori error estimates could be used, but this depends on the complexity of the physical PDEs. For example, for the unsteady Euler equations no good error estimates are available, so Van der Vegt and Van der Ven use equidistribution-like weights based on solution variables and total pressure loss [vdVvdV98] complemented by a local vorticity sensor and a grid anisotropy sensor [vdVvdV02].

2.6.4 *hr-refinement*

We believe that the hybrid combination of h- and r-refinement could be extremely powerful. The alignment of mesh cells with shock lines and other flow features greatly reduces local truncation errors, but in some cases the fixed number of mesh cells in r-refinement methods is too big a limitation.

The need for hr-refinement: an example

We consider the two-dimensional HD22RM example problem [vdHK07] from hydrodynamics, details on the physics and the solver used can be found in Chapter 4. The domain is a long tube of size $[0,5]\times[0,1]$. At the initial time, a vertical Mach-10 shock wave is located at $x = 1/10$ and ahead of it lies a contact discontinuity (CD) at an angle of $\pi/3$ with tenfold lower density:

$$[\rho,\mathbf{v},p] = \begin{cases} [\ \ 8,\ 8.25,\ 0,\ 0,\ 116.5], & \text{if } x \le 0.1, \\ [\ 1.4,\ \ \ 0,\ 0,\ \ \ 0,\ \ \ 1], & \text{if } x > 0.1 \text{ and } \tan(\pi/3)(x-1) \le y, \\ [0.14,\ \ \ 0,\ 0,\ \ \ 0,\ \ \ 1], & \text{otherwise.} \end{cases}$$

The shock moves rightward and hits the contact discontinuity after some time. It then gains speed in the lower-density medium and reflects off the top and bottom walls. A more detailed discussion is given in the original paper. Figure 2.7 shows the overall solution at $t = 0.3$. The rightmost structure is the original shock with its reflections and the distinct line of features left of it is the perturbed CD. All of these phenomena are well captured by adaptive methods (both r- and h-). Moreover, the passing shock deposits additional vorticity on the unstable CD, which produces Richtmyer–Meshkov instabilities over time. These are small rotational phenomena that can occur everywhere along the CD. Here lies the problem: in the horizontal direction mesh cells have concentrated near the CD (and shock), but vertically the points are needed everywhere along the CD. The result is a more-or-less

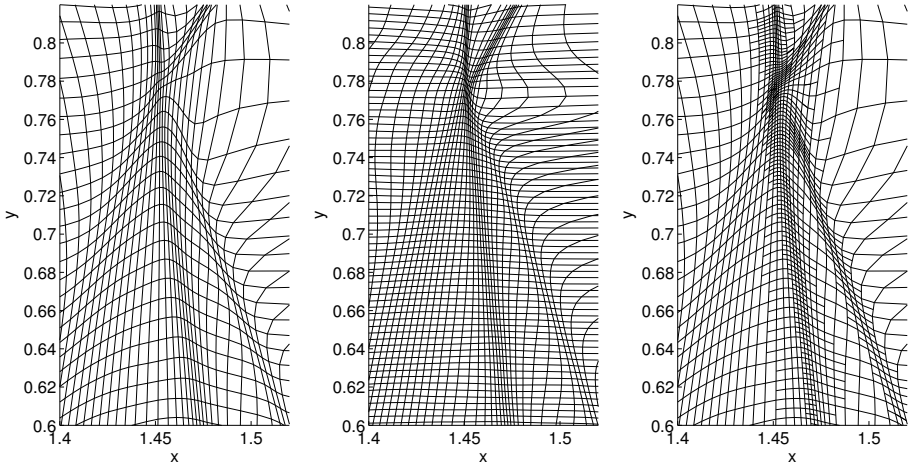

Figure 2.8 The need for hr-refinement. Mesh details for the HD22RM problem at $t = 0.12$. Left: total mesh size is 200×100 for the 5×1-sized domain. Middle: total mesh size is 200×300. Right: r-refinement from left diagram (200×100) with h-refinement where needed.

uniform mesh in the y-direction. The spatial resolution is generally too low to properly capture the small instabilities. In r-refinement (moving mesh) methods the only option is to start with a large amount of mesh points in the y-direction. Figure 2.8 shows a detail of a 200×100 mesh at an earlier time, where the horizontal resolution in the shock and CD is much larger than the vertical resolution. The middle diagram shows the expensive alternative: a mesh with three times as many points in the y-direction only. Combined with h-refinement, the cells that are aligned with the CD and shock could additionally be *locally* refined, to obtain the same effective resolution in the y-direction as was already obtained in the x-direction. The rightmost diagram shows this hr-combination where the base grid is the same as for the r-refinement in the leftmost diagram and on top of that only in the CD and shocks local h-refinement has occurred.

Related work

Combinations of h- and r-refinement have been investigated occasionally in the past. Arney and Flaherty [AF90] employ a basic moving mesh approach that identifies clusters of cells with large errors and moves them along with the solution features. Subsequently, these cells can be refined recursively. According to their experiments, the main gain comes from r-refinement.

Capon and Jimack [CJ96] use a two-dimensional mesh movement approach inspired by equidistribution on an unstructured mesh. The combination with local refinement yields results that are equally accurate as without local refinement, but at lower costs, as the initial mesh needs to have less points. In the resulting meshes, the refined areas are very similar for both methods, but

the less demanding regions in the domain have significantly less mesh cells for the hr-combination.

Lang et al. [LCHR03] use the MMPDE approach (Section 2.3.2.7) for mesh movement on an unstructured mesh, combined with a finite element solver. Next, they apply local refinement, based on an estimate of the local discretization error. The authors show convincing results of the hr-combination, especially for unsteady problems. The distinguishing difference between pure h- and hr-refinement is the continuous coarsening and refining in the former method to keep up with the moving flow features, whereas the latter method keeps the total number of mesh cells approximately constant over time.

The latter two approaches consider unstructured meshes. Morrell et al. [MSB07, Mor07] use moving quadrilateral meshes that are refined one level in an anisotropic way. Hanging nodes are allowed. They report a performance gain of a factor 9 for several examples. The most sensitive point appears to be the possibility of oscillations when a shock-like structure passes the interface between cells of different refinement levels.

2.7 INGREDIENTS OF OUR ADAPTIVE METHOD

The great variety of mesh adaptation approaches discussed in this chapter may be somewhat overwhelming. We now list the approaches that form the ingredients of our one- and two-dimensional solvers in the following chapters.

We use the direct variant of Winslow's method (Section 2.3.2.8), with directional monitor functions (Section 2.5.2). The monitor uses two types of monitor balancing (within and across components, see page 48) and the components are usually solution gradients, and sometimes derived physical quantities (see Section 4.4.3). The mesh generator equations and the physical PDEs are solved alternately, combined with conservative solution interpolation (Section 2.4.2).

One-dimensional Mesh Movement, Multidimensional Magnetohydrodynamics

3

The adaptive mesh movement is demonstrated on one-dimensional (1D) domains in this chapter[1]. As 1D mesh movement is pretty well understood, an additional challenge is posed by the application of quasi-two- and three-dimensional magnetohydrodynamics (MHD). Solutions to these models have various interesting features that the monitor function should detect automatically. This chapter has three main contributions. Firstly, it shows how adaptive mesh movement can be easily combined with a PDE solver, in this case a higher-order finite volume method. Secondly, it demonstrates the effectiveness of a smooth monitor function that needs little manual fine-tuning. Thirdly, it shows how the mesh adaptation captures subtle phenomena such as critical solutions and also led to the discovery of a physical staircasing phenomenon in the oscillating plasma sheet problem.

3.1 INTRODUCTION

Many interesting phenomena in plasma fluid dynamics can be described within the framework of magnetohydrodynamics (MHD). Numerical studies in plasma flows frequently involve simulations with highly varying spatial and temporal scales. As a consequence, numerical methods on uniform meshes are inefficient, since a very large number of mesh points is needed to resolve the spatial structures, such as shocks, contact discontinuities, shear layers, or current sheets. For the efficient study of these phenomena we need adaptive mesh methods which automatically track and spatially resolve all of these structures. The problems considered here come from previous work by Tóth et al. [TO96, TKB98], Keppens [Kep04], Torrilhon [Tor03], and Zegeling et al. [ZK01].

Huang et al. [HRR94] prescribe mesh movement by a moving mesh PDE (MMPDE), which is solved simultaneously with the physical PDEs for one-dimensional models. Although this avoids solution interpolation, the coupled system may be hard to solve due to differences in time scales and desired error tolerances. For two-dimensional models the MMPDE and physical PDEs are often decoupled and solved in an alternating way, as Huang and Russell [HR99] show. Stockie et al. [SMR00] also use an MMPDE-based decoupled approach for solving one-dimensional hyperbolic systems of conservation laws. It is similarly based on the equidistribution principle that follows from a variational formulation of mesh energy minimization. Tang

[1] An earlier version of this chapter appeared as [vDZ06]: 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.

et al. [TT03] extend this approach to two-dimensional domains, but use a stationary description for the mesh movement, hence a decoupled approach by definition. Their monitor function still needs parameterization for each new problem by hand, though. In this chapter, we take the latter approach, with an improved monitor function. Zegeling et al. [ZdBT05] have recently used a similar method for two-dimensional hydrodynamic problems. The smoothness of mesh distribution is important for decreasing interpolation errors in the decoupled approach. The most powerful means for this is a good choice of monitor function. Huang has done much research on different monitor functions [CHR99, with Cao and Russell], monitor quality [Hua01a], and mesh quality [Hua05]. Error analysis quickly becomes complicated for moving mesh methods, but Beckett and Mackenzie [BM00, BMRS01] have done some convergence studies for these methods. Tang [Tan05] recently presented an interesting overview paper on moving mesh methods for computational fluid dynamics. Zegeling et al. [ZK01] also employ a moving mesh method, but it is fully coupled and solved using the method of lines and an implicit time solver. Although their mesh movement is fairly sophisticated, ensuring mesh-consistency and smooth mesh movement, it still needs manually-set adaptivity parameters. Furthermore, an artificial diffusion term is added in order to handle discontinuities in the physical solution. To avoid these artificial terms, we use a high-resolution finite volume method with MUSCL-type flux-limiters as proposed by Van Leer [vL79].

The layout of this chapter is as follows. In the next section we present the full set of MHD equations and their physical meaning. In Section 3.3 we describe the adaptive moving mesh method, based on the equidistribution principle, including a conservative solution interpolation. This is followed by details on the high-resolution finite volume method. Special attention is also given to a more sophisticated monitor function. Numerical experiments are presented in Section 3.4. Not only accuracy is considered, but also computational efficiency, in comparison with uniform methods. Also, some experiments compare r-refinement with h-refinement. Besides, interesting physical aspects of MHD are studied, such as pseudo-convergence to incorrect critical solutions, propagation of Alfvén waves, and high speed magnetosonic effects. In Section 3.5 we give conclusions and suggestions for improvement.

3.2 THE EQUATIONS OF MAGNETOHYDRODYNAMICS

The MHD equations govern the dynamics of a charge-neutral ionized gas, or 'plasma'. Just as the conservative Euler equations provide a continuum description for a compressible gas, the MHD equations express the basic physical conservation laws a plasma must obey. Because plasma dynamics are influenced by magnetic fields through the Lorentz-force, the needed additions in going from hydrodynamic to magnetohydrodynamic behavior consist of a vector equation for the magnetic field evolution and extra terms in the Euler system that quantify the magnetic force and energy density.

Using the conservative variables density ρ, momentum density $\mathbf{m} := \rho\mathbf{v}$

(with velocity $\mathbf{v}$), magnetic field $\mathbf{B}$, and total energy density e, the ideal MHD equations can be written as follows (cf. [BW88], [TO96], [TKB98]):
Conservation of mass:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{m} = 0. \tag{3.1}$$

Conservation of momentum:

$$\frac{\partial \mathbf{m}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B}) + \nabla p_{tot} = 0. \tag{3.2}$$

Magnetic field induction:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v}) = 0. \tag{3.3}$$

Conservation of energy:

$$\frac{\partial e}{\partial t} + \nabla \cdot (e\mathbf{v} + \mathbf{v}p_{tot} - \mathbf{B}\mathbf{B} \cdot \mathbf{v}) = 0. \tag{3.4}$$

Hereafter, we will abstract from the above four quantities by introducing the solution vector $\mathbf{q}(\mathbf{x}, t)$, where $\mathbf{x} := [x, y]$ or $\mathbf{x} := [x, y, z]$. In (3.2) and (3.4) the total pressure p_{tot} consists of both a thermal and a magnetic contribution, as given by:

$$p_{tot} = p + \frac{B^2}{2}, \text{ where } p = (\gamma - 1)(e - \rho\frac{v^2}{2} - \frac{B^2}{2}) \tag{3.5}$$

is the thermal pressure ($B^2 := \mathbf{B} \cdot \mathbf{B}$). The adiabatic constant γ is the ratio of specific heats of the plasma. This set of equations must be solved in conjunction with an important condition on the magnetic field $\mathbf{B}$, namely the non-existence of magnetic 'charge' or monopoles. Mathematically, it is easily demonstrated that this property can be imposed as an initial condition alone, since

$$\nabla \cdot \mathbf{B}|_{t=0} = 0 \Longrightarrow \nabla \cdot \mathbf{B}|_{t \geq 0} = 0. \tag{3.6}$$

In multi-dimensional numerical MHD, the combined spatio-temporal discretization may not always ensure this conservation of the solenoidal character of the vector magnetic field. Note that in our 1D applications this solenoidal property is satisfied automatically by construction (see below).

3.2.1 *Derivation of 1.5D and 1.75D models*

If we restrict the MHD model (3.1)–(3.6) to variations in one spatial dimension x, i.e., $\partial \mathbf{q}/\partial y = \mathbf{0}$, with possibly non-vanishing y-components for the vector quantities, we obtain a 5-component PDE system in 1D, which is sometimes referred to as '1.5D'. If we also include possibly non-vanishing z-components of the vector quantities, but still keep $\partial/\partial z = 0$ for the flux, we obtain a 7-component PDE system in 1D, which is sometimes referred to as '1.75D'. This system is formally written as

$$\frac{\partial}{\partial t}\mathbf{q} + \frac{\partial}{\partial x}\mathbf{f}(\mathbf{q}) = 0, \quad x \in [x_L, x_R], \quad t > 0. \tag{3.7}$$

Here, $\mathbf{q} = [\rho, m_1, m_2, m_3, B_2, B_3, e]^T$ is the vector of conserved variables (m_1, m_2 , m_3 are now the x-, y- and z-components of the momentum vector and B_2 and B_3 denote the y- and z-component of the magnetic induction), with the flux-vector $f = (f_1, \ldots, f_7)^T$ given by

$$\begin{aligned}
f_1 &= m_1, \\
f_2 &= \frac{m_1^2}{\rho} - \bar{B_1}^2 + (\gamma - 1)e \\
&\quad -(\gamma - 1)\frac{m_1^2 + m_2^2 + m_3^2}{2\rho} + (2 - \gamma)\frac{\bar{B_1}^2 + B_2^2 + B_3^2}{2}, \\
f_3 &= \frac{m_1 m_2}{\rho} - \bar{B_1} B_2, \\
f_4 &= \frac{m_1 m_3}{\rho} - \bar{B_1} B_3, \\
f_5 &= B_2 \frac{m_1}{\rho} - \bar{B_1} \frac{m_2}{\rho}, \\
f_6 &= B_3 \frac{m_1}{\rho} - \bar{B_1} \frac{m_3}{\rho}, \\
f_7 &= \frac{m_1}{\rho}\left(\gamma e - (\gamma - 1)\frac{m_1^2 + m_2^2 + m_3^2}{2\rho} + (2 - \gamma)\frac{\bar{B_1}^2 + B_2^2 + B_3^2}{2}\right) \\
&\quad -\bar{B_1}\left(\bar{B_1}\frac{m_1}{\rho} + B_2\frac{m_2}{\rho} + B_3\frac{m_3}{\rho}\right).
\end{aligned} \tag{3.8}$$

The first component of the magnetic field vector is kept at a constant value $\bar{B_1}$. The vanishing divergence of the magnetic field is thereby trivially satisfied in this model situation. The remaining set of 7 PDEs given by (3.7) constitutes the physical model used for the '1.75D shock tube' simulation found hereafter. Furthermore, several 1.5D simulations are shown; these are again described by (3.7), where f_4 and f_6 drop out of the flux formulae, as well as all terms involving m_3 and B_3. Keppens [Kep04] also derives these two models and solves two shock tube problems on uniform meshes.

We first indicate how this system is further manipulated and discretized to solve alternately for the adaptive mesh with its corresponding solution.

3.2.2 *Eigen-structure for MHD*

The eigenvalues of the flux Jacobian $\mathbf{f}_q := \partial\mathbf{f}/\partial\mathbf{q}$ represent the speeds at which the various waves of an MHD Riemann solution move. In 1.5D these are an entropy wave (v_1), two fast ($v_1 \pm c_f$) and two slow ($v_1 \pm c_s$) magnetosonic waves, where

$$c_{f,s}^2 = \frac{1}{2}\left(\frac{\gamma p + B^2}{\rho} \pm \sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - 4\frac{\gamma p}{\rho}\frac{\bar{B_1}^2}{\rho}}\right). \tag{3.9}$$

For the full system of MHD equations, i.e., 1.75D, two additional eigenvalues represent Alfvén waves with speed $v_1 \pm c_a$, where $c_a = B_1/\sqrt{\rho}$. In general, the

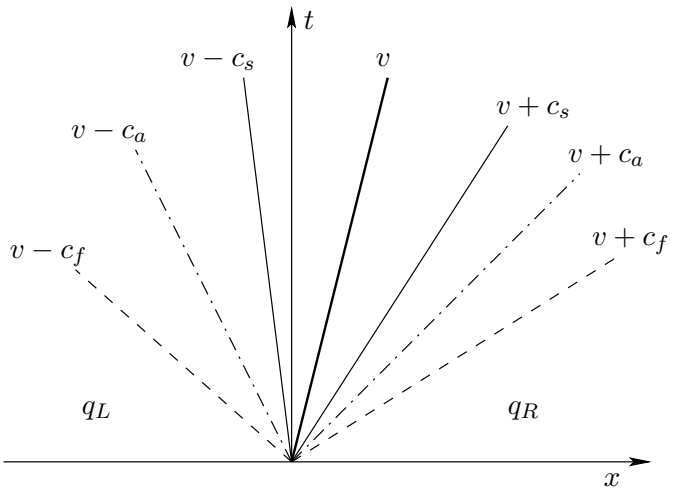

Figure 3.1 Wave structure of a 1.75D MHD Riemann problem.

following ordering holds: $c_s \leq c_a \leq c_f$. Figure 3.1 shows the wave structure of a 1.75D MHD Riemann problem.

3.3 THE MOVING MESH METHOD

This section describes the moving mesh finite volume approach as introduced by Tang et al. [TT03]. For increased robustness, we use a more sophisticated monitor function, originally proposed by Beckett et al. [BM00]. This combination yields a powerful solver that tracks and resolves both small, local and large solution gradients automatically. No parameter adaptation by hand using prior knowledge on the eventual shape of the solution is necessary. Hence, the solver can be quickly applied to problems from entirely different application areas.

The numerical algorithm is shown below. The symbol $\mathbf{Q}_{j+1/2}$ represents the numerical solution for $\mathbf{q}$, as will be introduced in (3.11). Each time step consists of a mesh moving step and a physical PDE solving step. The next two sections describe these separate steps. Finally, Section 3.3.3 deals with monitor functions in more detail.

3.3.1 *Mesh adaptation in 1D*

The solution of the MHD equations, denoted by $\mathbf{q} \in \mathbb{R}^m$, is defined on the physical domain $\Omega_p := [x_L, x_R] \subset \mathbb{R}$ with coordinate x. Introducing a fixed computational domain $\Omega_c := [0, 1] \subset \mathbb{R}$, with coordinate ξ, a coordinate transformation, or one-to-one mesh map, is defined by:

$$x := x(\xi), \quad \xi \in \Omega_c,$$

Algorithm 1 MMFVSOLVE – 1D moving mesh finite volume PDE solver.

Generate an initial uniform mesh: $x_j^0 = x_L + j \cdot \frac{x_R - x_L}{N}$, $j = 0, \ldots, N$.
Compute initial values $\mathbf{Q}_{j+1/2}^0$ based on cell average of $\mathbf{q}(x, 0)$.
while $t_n < T$ **do**
  repeat
    $\nu = 0$; $x_j^{[0]} = x_j^n$; $\mathbf{Q}_{j+1/2}^{[0]} = \mathbf{Q}_{j+1/2}^n$, $j = 0, \ldots, N$.
    Move grid $\left\{x_j^{[\nu]}\right\}$ to $\left\{x_j^{[\nu+1]}\right\}$, using a Gauss-Seidel iteration (3.12).
    Compute the solution $\left\{\mathbf{Q}_{j+1/2}^{[\nu+1]}\right\}$ on the new mesh, using (3.13).
  until $\nu \geq \nu_{\max}$ or $\left\|x^{[\nu+1]} - x^{[\nu]}\right\| \leq \epsilon$
  Compute $\mathbf{Q}^{n+1}$ using high-resolution finite volumes (3.14).
end while

or its inverse

$$\xi := \xi(x), \quad x \in \Omega_p.$$

In a variational approach, finding the most appropriate mesh map $x(\xi)$ for some solution profile is equivalent to finding a ξ that minimizes a mesh energy functional $\mathcal{E}(\xi)$. A simple, but effective, mesh energy is:

$$\mathcal{E}(\xi) = \frac{1}{2} \int_{\Omega_p} \xi_x^2 \, \frac{1}{\omega} \, \mathrm{d}x,$$

where $\omega > 0$ is a monitor function, which will be considered in more detail in Section 3.3.3. In general, ω is defined in terms of spatial derivatives of $\mathbf{q}$. In a variational formulation (cf. [TWM85]), minimization of the mesh energy yields the Euler–Lagrange equation:

$$\left(\frac{1}{\omega}\xi_x\right)_x = 0.$$

This is equivalent to the equidistribution principle in 1D, ωx_ξ = constant, or:

$$(\omega x_\xi)_\xi = 0. \tag{3.10}$$

Now that the adaptive mesh is implicitly prescribed, a numerical algorithm can be set up, that determines the new mesh and updates the solution on it.

Domain discretization

To facilitate differential operators with stencils up to size 5, a domain discretization as depicted in Figure 3.2 is used. The domain $\Omega_p := [x_L, x_R]$ is discretized using $N+1$ mesh points, with two additional mesh points on both sides outside Ω_p. The computational domain Ω_c is discretized with $N+1$ uniform coordinates $\xi_j = j/N$ $(0 \leq j \leq N)$.

As the finite volume solver uses cell averaged solution values, the discrete solution $\mathbf{Q}_{j+1/2}$ is defined on the cell center:

$$\mathbf{Q}_{j+\frac{1}{2}} \approx \frac{1}{\Delta x_{j+1/2}} \int_{x_j}^{x_{j+1}} \mathbf{q}(x) \, \mathrm{d}x, \quad 0 \leq j \leq N-1, \tag{3.11}$$

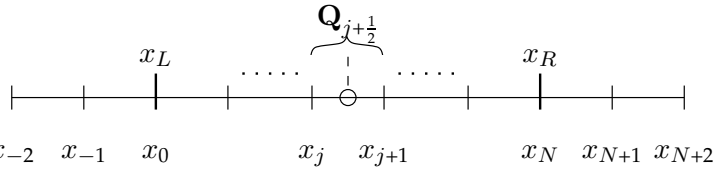

Figure 3.2 The discretized spatial domain with 'beyond-boundary'-points.

where the local cell size, or mesh width is given by

$$\Delta x_{j+1/2} := x_{j+1} - x_j.$$

Mesh redistribution

For every time $t > 0$, the new mesh should satisfy the redistribution equation (3.10). Using central differences for $(x_\xi)_{j+1/2}$, and inserting the current solution and monitor values yields a linear system in $[x_1, \ldots, x_{N-1}]^T$, which is solved with a Gauss–Seidel (GS) iteration:

$$x_j^{[\nu+1]} = \frac{\omega\left(\mathbf{Q}_{j-1/2}^{[\nu]}\right) x_{j-1}^{[\nu+1]} + \omega\left(\mathbf{Q}_{j+1/2}^{[\nu]}\right) x_{j+1}^{[\nu]}}{\omega\left(\mathbf{Q}_{j-1/2}^{[\nu]}\right) + \omega\left(\mathbf{Q}_{j+1/2}^{[\nu]}\right)}, \tag{3.12}$$

where $x_j^{[\nu+1]}$ ($\nu = 0, 1, \ldots$) denotes the updated mesh point. Typically a mere three to five steps are performed before the mesh adaptation is considered appropriate ($\nu_{\max} = 3$ to $5, \epsilon = 10^{-6}$). In most cases the $\nu_{\max}$-bound is reached before the ϵ-bound. Each mesh moving step also involves a solution interpolation, as described hereafter. The small number of GS steps keeps the costs of this interpolation low. Many, more advanced solvers exist, but accuracy of mesh movement is not the most critical aspect here. Tang et al. give proof [TT03] of the preservation of monotonic order of $x^{[\nu]}$:

$$x_{j+1}^{[\nu]} > x_j^{[\nu]} \Longrightarrow x_{j+1}^{[\nu+1]} > x_j^{[\nu+1]}, \quad 0 \le j \le N,$$

or, equivalently: x_ξ is strictly monotonically increasing. This is desirable, since otherwise mesh points might collapse and solution gradients could blow up.

Solution updating on the new mesh

In each redistribution step, mesh points x are moved to a new location $\tilde{x}$. Also, the solution $\mathbf{Q}$ needs to be updated on the new mesh, yielding $\tilde{\mathbf{Q}}$. A conservative interpolation technique is used, to maintain physically correct solutions.

Assuming that the difference $c(x)$ between the old mesh x and new mesh $\tilde{x} := x - c(x)$ is small, using a perturbation method eventually yields the interpolation relation:

$$\mathbf{Q}_{j+1/2}^{[\nu+1]} = \frac{\left(x_{j+1}^{[\nu]} - x_j^{[\nu]}\right) \mathbf{Q}_{j+1/2}^{[\nu]} - \left((c\mathbf{Q})_{j+1}^{[\nu]} - (c\mathbf{Q})_j^{[\nu]}\right)}{x_{j+1}^{[\nu+1]} - x_j^{[\nu+1]}}, \tag{3.13}$$

where the upwinding (cf. Van Leer [vL77a, Eq. (12)]) numerical fluxes are approximated by:

$$(c\mathbf{Q})_j = \frac{c_j}{2}\left(\mathbf{Q}_j^+ + \mathbf{Q}_j^-\right) - \frac{|c_j|}{2}\left(\mathbf{Q}_j^+ - \mathbf{Q}_j^-\right).$$

This method uses the 'wave speed' $c_j^{[\nu]} = x_j^{[\nu]} - x_j^{[\nu+1]}$, and $\mathbf{Q}_j^+$ and $\mathbf{Q}_j^-$, which approximate $\mathbf{Q}_j$ at a cell edge, are defined by (3.18). The interpolation relation (3.13) is in a conservative flux-differencing form, hence the interpolation satisfies the following conservation property:

$$\sum_j \Delta\tilde{x}_{j+\frac{1}{2}}\tilde{\mathbf{Q}}_{j+\frac{1}{2}} = \sum_j \Delta x_{j+\frac{1}{2}}\mathbf{Q}_{j+\frac{1}{2}}.$$

The updating of the solution is preceded by a single mesh redistribution step; the combination of the two forms the body of the GS iteration.

3.3.2 *Finite volume solver for physical PDEs*

On the redistributed mesh, the physical PDEs can be solved by any PDE solver that accepts nonuniform discretizations. We use a second order finite volume method. In the following, the mesh at time t_n is given by $x^n := x^{[\nu+1]}$.

One-dimensional hyperbolic systems of conservation laws are described by the PDE system in (3.7). Integrating the PDE over the control volume $[t_n, t_{n+1}\rangle \times \left[x_j^n, x_{j+1}^n\right]$ leads to the following explicit finite volume method (we improve the time integration in (3.20)):

$$\mathbf{Q}_{j+1/2}^{n+1} = \mathbf{Q}_{j+\frac{1}{2}}^n - \frac{t_{n+1}-t_n}{x_{j+1}^n - x_j^n}\left(\mathbf{F}_{j+1}^n - \mathbf{F}_j^n\right) \tag{3.14}$$

$$=: \mathbf{Q}_{j+\frac{1}{2}}^n + \Delta t^n\ L_{j+\frac{1}{2}}(\mathbf{Q}^n), \tag{3.15}$$

where the cell average $\mathbf{Q}_{j+1/2}^{n+1}$ is defined in (3.11) and $\mathbf{F}_j^n$ is some numerical flux satisfying

$$\mathbf{F}_j^n = \mathbf{F}\left(\mathbf{Q}_j^{n,-}, \mathbf{Q}_j^{n,+}\right), \quad \mathbf{F}(\mathbf{Q},\mathbf{Q}) = \mathbf{f}(\mathbf{Q}). \tag{3.16}$$

We use a *local* Lax–Friedrichs (LF) flux

$$\mathbf{F}(\mathbf{Q}_a, \mathbf{Q}_b) = \frac{1}{2}\left[\mathbf{f}(\mathbf{Q}_a) + \mathbf{f}(\mathbf{Q}_b) - \max_{\mathbf{Q}\in\{\mathbf{Q}_a,\mathbf{Q}_b\}}\{|\mathbf{f}_q|\}(\mathbf{Q}_b - \mathbf{Q}_a)\right], \tag{3.17}$$

where the largest absolute eigenvalues of the Jacobian $\mathbf{f}_q := \partial\mathbf{f}/\partial\mathbf{q}$ are used. Local LF is less diffusive than normal LF, since it locally limits the numerical viscosity instead of having a uniform viscosity on the entire domain.

To determine flux values at cell boundaries, the solution values $\mathbf{Q}_j$ are approximated using values from the cell centers at both the left and right side. In (3.16), $\mathbf{Q}_j^{n,\pm}$ are defined using the initial reconstruction technique:

$$\mathbf{Q}_j^{n,\pm} = \mathbf{Q}_{j\pm1/2} + \frac{1}{2}\left(x_j^n - x_{j\pm1}^n\right)\tilde{\mathbf{S}}_{j\pm1/2}, \tag{3.18}$$

where $\tilde{\mathbf{S}}_{j+1/2}$ is an approximation of the slope $\mathbf{q}_x$ at $x^n_{j+1/2}$, defined by:

$$\tilde{\mathbf{S}}_{j+1/2} = \left(\text{sign}\left(\tilde{\mathbf{S}}^+_{j+1/2}\right) + \text{sign}\left(\tilde{\mathbf{S}}^-_{j+1/2}\right)\right) \frac{\left|\tilde{\mathbf{S}}^+_{j+1/2}\tilde{\mathbf{S}}^-_{j+1/2}\right|}{\left|\tilde{\mathbf{S}}^+_{j+1/2}\right| + \left|\tilde{\mathbf{S}}^-_{j+1/2}\right|}, \tag{3.19}$$

with

$$\tilde{\mathbf{S}}^+_{j+1/2} = \frac{\mathbf{Q}^n_{j+3/2} - \mathbf{Q}^n_{j+1/2}}{x^n_{j+3/2} - x^n_{j+1/2}}, \quad \tilde{\mathbf{S}}^-_{j+1/2} = \frac{\mathbf{Q}^n_{j+1/2} - \mathbf{Q}^n_{j-1/2}}{x^n_{j+1/2} - x^n_{j-1/2}}.$$

The above is a MUSCL-type method, where the slope approximation (3.19) uses a monotonicity preserving slope limiter as formulated by Van Leer [vL77b, Eq. (67)]. Note that both the reconstruction and slope limiting is performed for each component of the solution vector $\mathbf{Q}$ separately.

To obtain a higher accuracy in the time range, the standard *one-step* finite volume formulation (3.15) is replaced by a second-order Runge-Kutta scheme:

$$\begin{aligned} \mathbf{Q}^*_{j+1/2} &= \mathbf{Q}^n_{j+\frac{1}{2}} + \Delta t^n \; L_{j+\frac{1}{2}}(\mathbf{Q}^n), \\ \mathbf{Q}^{n+1}_{j+1/2} &= \frac{1}{2}\left(\mathbf{Q}^n_{j+\frac{1}{2}} + \mathbf{Q}^*_{j+\frac{1}{2}} + \Delta t^n \; L_{j+\frac{1}{2}}(\mathbf{Q}^*)\right). \end{aligned} \tag{3.20}$$

In a method with changing mesh widths, the stability criterion for the time step is extra important. The standard CFL limit reads

$$\left|\frac{\mathbf{f}_q \Delta t}{\Delta x}\right| \leq 1, \quad \forall \Delta t, \, \Delta x, \text{ eigenvalues of } \mathbf{f}_q. \tag{3.21}$$

To enforce higher accuracy, the Courant number will here be limited by a parameter $\mathcal{C}$, thereby limiting the time step to:

$$\Delta t^n \leq \mathcal{C} \; \min_j \frac{\Delta x_{j+1/2}}{\left|\mathbf{f}_q(\mathbf{Q}^n_{j+1/2})\right|}, \tag{3.22}$$

where $0 < \mathcal{C} \leq 1$. Notice how we determine the limit on the time step locally, instead of using, e.g., $\Delta t^n \leq \min_j \Delta x_{j+1/2} / \max_j |\mathbf{f}_q(\mathbf{Q}^n_{j+1/2})|$.

3.3.3 *A sophisticated monitor function*

An often seen, most basic choice for controlling adaptivity is the arc length-type monitor function for scalar solutions q:

$$\omega(q) = \sqrt{1 + \alpha(\partial q/\partial x)^2}, \tag{3.23}$$

where the adaptivity parameter α controls the amount of adaptivity. The value 1 is set as floor on the monitor function to prevent *all* points from concentrating in just the steep parts of the solution. This type of monitor has two problems. Firstly, α is problem dependent; a problem from gas dynamics might require an α of an entirely different order than a problem

from hyperbolic macroscopic traffic flow models. Secondly, α is a constant, whereas the solution profile might change significantly through time. The chosen α based on the solution at the initial time may be far from optimal at some point of time $t > 0$.

From now on we will use the term 'critical' for parts of the domain where refinement is especially necessary. For the monitor function (3.23), 'critical' is equivalent to 'steep', because of the first-order derivative. In general, higher order derivatives may be used as well.

To overcome the before-mentioned disadvantages, Beckett and Mackenzie [BM00] introduce a more sophisticated monitor function, which we schematically define as:

$$\omega(q) = \alpha(q) + \phi(q).$$

It has a solution dependent floor value $\alpha(q)$, where $\alpha(q)$ is defined as an average value of some function $\phi(q)$. Most often, ϕ will contain solution gradients. Huang [Hua01a] generalizes this monitor function with a parameter β that controls the ratio of points in critical parts. Here, we furthermore generalize to PDE *systems*, i.e., when $\mathbf{q}$ has $m > 1$ components. We define the monitor function $\omega(\mathbf{q}) \in (\Omega_p \times \mathbb{R}^{\geq 0} \to \mathbb{R}^m) \to \mathbb{R}^{>0}$ as:

$$\omega(\mathbf{q}) = \sum_{p=1}^{m} \left[(1-\beta)\alpha_p(\mathbf{q}) + \beta \left| \frac{\partial q_p}{\partial \xi} \right|^{1/2} \right], \tag{3.24}$$

where

$$\alpha_p(\mathbf{q}) = \int_{\Omega_c} \left| \frac{\partial q_p}{\partial \xi} \right|^{1/2} d\xi. \tag{3.25}$$

The critical regions are now identified by the computational derivative $\partial\mathbf{q}/\partial\xi$, which is smoother than the physical derivative $\partial\mathbf{q}/\partial x$. The solution dependent $\alpha(\mathbf{q})$ averages this derivative for each component q_p separately. Finally, the m monitor values for all components are summed. Although β is still a user-defined parameter, we found $\beta = 0.8$ a suitable value for a range of different problems and keep it fixed at that for all numerical experiments in the next section.

Following the approach of Huang [Hua01a], it can be shown that for monitor (3.24), β is indeed the ratio of points in critical parts:

$$\beta = \frac{\int_{\Omega_p} \beta\phi \, \mathrm{d}x}{\int_{\Omega_p} (1-\beta)\langle\phi\rangle + \beta\phi \, \mathrm{d}x}, \tag{3.26}$$

where $\langle\phi\rangle$ is the averaged ϕ, i.e., $\alpha(q)$. Hence, for our fixed choice of $\beta = 0.8$, approximately 80% of the mesh points is positioned in critical parts of the domain.

Another technique to prevent the mesh points from being moved too brusquely, when some local gradient changes rapidly, is to smoothen the monitor function. This is done by applying a low-pass filter, possibly multiple times:

$$\omega_{j+1/2}^{\text{smooth}} \longleftarrow \frac{1}{4}\left(\omega_{j+\frac{3}{2}} + 2\omega_{j+1/2} + \omega_{j-1/2}\right), \tag{3.27}$$

where $\omega_{j+1/2} := \omega\left(\mathbf{Q}_{j+1/2}\right)$. Even with the sophisticated monitor (3.24) we found a single application of this smoothing operator to be beneficial and sufficient.

3.4 NUMERICAL EXPERIMENTS

The moving mesh method is now used on a selection of problems from magnetohydrodynamics. Although still in one spatial dimension, these problems have five ($m = 5$) or even seven ($m = 7$) model equations, and consequently exhibit a range of shocks, rarefaction waves and contact discontinuities (at most m). Some problems were also used by Zegeling and Keppens [ZK01] for testing their adaptive method of lines approach, which is a fully-coupled moving mesh method.

The numerical results are compared to a reference solution. Solutions to the shock tube problems (Sections 3.4.1, 3.4.2 and 3.4.3) were obtained with the exact Riemann solver by Torrilhon [Tor02]. The shear Alfvén problem (Section 3.4.4) is compared to a 2500 points adaptive solution. For all problems, solutions by the widely-used Versatile Advection Code [Tót96] (VAC, see `http://www.phys.uu.nl/~toth`) are also considered.

A *discrete* L_1 norm is used for an error measure on adaptive meshes:

$$\mathbf{E}_{L_1} = \sum_{j=1}^{N} \Delta x_j \left| \mathbf{Q}_{j+1/2} - \mathbf{q}(x_{j+1/2}) \right|, \tag{3.28}$$

which is an approximation to the area between the numerical and exact solution profile. Note that $\mathbf{E}$ is still a vector in $\mathbb{R}^m$, error measures may later pick out single components from the solution, e.g., density, or sum them. In addition to observing the numerical errors (3.28), we have also checked some physical properties of the computed solution, such as conservation and positivity of solution components.

Most problems have homogeneous Neumann boundary conditions, unless stated otherwise. We expand solution values to the two ghost cells on the left and right by copying the first value inside the domain (i.e., $\mathbf{Q}_{-3/2} = \mathbf{Q}_{-1/2} = \mathbf{Q}_{1/2}$ at the left). All experiments keep the CFL number at 0.5 for increased accuracy, although values up to 1.2 did not result in instability yet. We used a Pentium M 1.8GHz notebook for all experiments.

3.4.1 *MHD shock tube in 1.5D: computational efficiency*

One-dimensional shock tube problems are Riemann problems, where an imaginary tube contains plasmas in two different states, separated by a diaphragm. At $t = 0$ the diaphragm opens and the left and right state start to interact. In hydrodynamics, Sod's shock tube is the best known example problem. Here, we consider the classical MHD shock tube in 1.5D, initially described by Brio and Wu [BW88], which now is widely considered a benchmark problem for MHD simulations.

The problem is set up in the domain $[0, 1]$, with the discontinuity at $x = 0.5$. We simulate for times $t \in [0, 0.1]$. The plasma is initially at rest ($\mathbf{m} = \mathbf{0}$), with $\gamma = 2$ and $\bar{B}_1 = 0.75$. The problem is *co-planar*, i.e., $B_{2,L} = -B_{2,R}$. The difference in density and pressure between the two states is: $\rho_L = 8\rho_R = 1$, and $p_L = 10p_R = 1$. In conservation form, the initial conditions are:

$$\begin{aligned} [\rho, m_1, m_2, B_2, e]_L &= [\quad 1,\ 0,\ 0,\quad 1,\ 1.78125], \quad \text{if } x \leq 0.5, \\ [\rho, m_1, m_2, B_2, e]_R &= [0.125,\ 0,\ 0,\ -1,\ 0.88125], \quad \text{if } x > 0.5, \end{aligned} \tag{3.29}$$

where subscripts L and R denote the left and right state. Homogeneous Neumann boundary conditions are used for all components.

3.4.1.1 *Numerical results*

First, we compare our moving mesh method to the same method with a uniform mesh. Also, a finite volume solver from the VAC package is considered; we use it with a similar TVD Lax–Friedrichs flux and Van Leer flux limiter for a fair comparison. We sum all five components of E_{L_1} to study the overall error. Figure 3.3 shows the density and v_1 component of the velocity on the first row. The moving mesh and uniform solutions have $N = 250$ mesh points, and the uniform VAC solution has $N = 1500$ (both take equal running time approximately). The bottom left diagram focuses on the middle three waves.

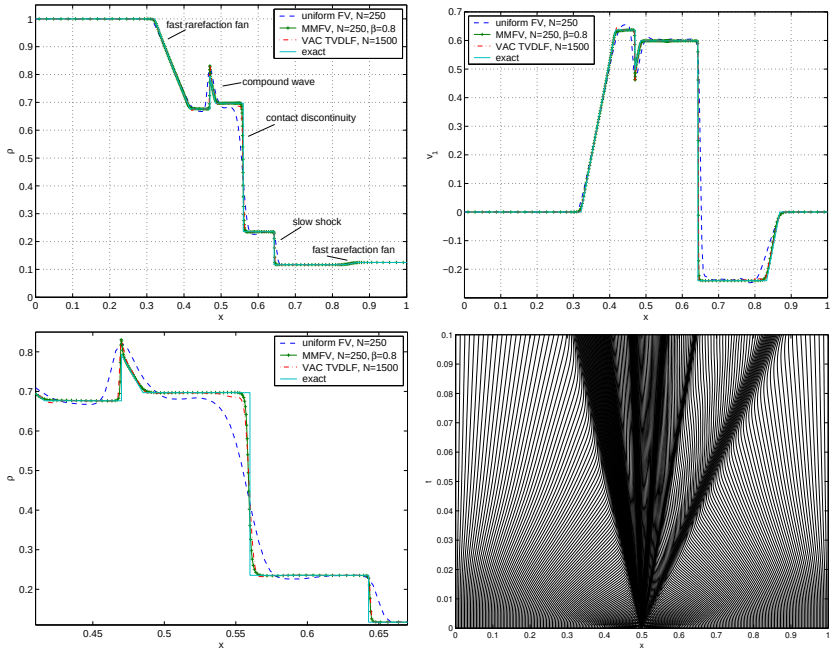

Figure 3.3 Solutions to the Brio and Wu problem.

The uniform solution is quite diffusive, a known property of Lax–Friedrichs-type methods. VAC with 1500 points has a much higher resolution, especially

at the compound wave. Our moving mesh result is slightly more accurate for all shocks. Its overall error is $9.1 \cdot 10^{-3}$, and the VAC result has an overall error of $1.3 \cdot 10^{-2}$. The top right diagram shows the v_1 component of the velocity. It is very accurate and does not suffer from the dispersive effects observed by Zegeling and Keppens [ZK01, Fig. 2]. Finally, the bottom right diagram shows the mesh movement through time. Note how the rightmost fast rarefaction fan is also properly detected.

Computational efficiency

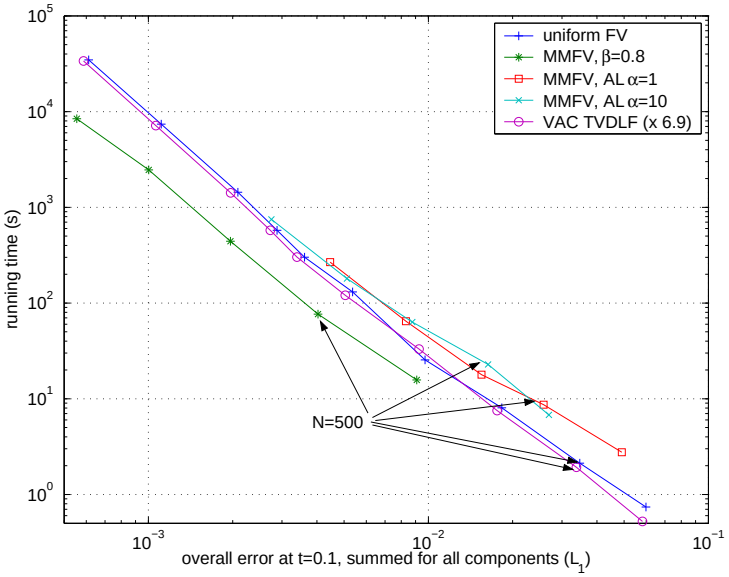

Figure 3.4 Computational efficiency for the Brio and Wu problem.

The increased accuracy comes at a price: the mesh movement and conservative interpolation take about 50% of the total running time. The amount of mesh points needed is seriously smaller, though, so the moving mesh method should be more efficient on the whole. To test this, the Brio and Wu problem was solved with N = 250, 500, 1000, 2000, 4000, 6000, 8000, 12,000, 24,000, and 48,000. The error was determined using (3.28) and summing all five components, as we are interested in overall accuracy here.

The diagram in Figure 3.4 sets out these errors on the horizontal axis and the running time on the vertical axis. The moving mesh method is used with monitor (3.24), and twice with the arc length-type monitor (α = 1 and 10). Since VAC programs are in Fortran and our method runs in MATLAB, the VAC timings are normalized using

$$t_{\text{VAC,norm}} = t_{\text{VAC}} \cdot t_{\text{unif,N=N}_{\text{norm}}} / t_{\text{VAC,N=N}_{\text{norm}}},$$

where we normalized with the $N_{\text{norm}} = 8000$ measurements. The uniform lines for MATLAB and VAC now almost coincide for all N, which justifies the normalization. The arc length runs are never efficient enough to beat uniform solutions. With the advanced monitor function, the moving mesh method becomes a lot more efficient. The gain factor in running time, compared to uniform runs, is approximately 3. A possible improvement could be to adapt the mesh every $k > 1$ time steps. This will reduce the mesh movement and interpolation costs. In their study of efficiency of h-refinement, Keppens et al. have an average of 20% additional costs for mesh adaptation in 1D [KNTG03]. Also note how for the same amount of points (e.g., $N = 500$) the obtained accuracy differs (by almost a factor 10 between adaptive and uniform runs). When computer memory is an issue, adaptive methods can still compute accurate solutions with relatively small discrete solution vectors. This becomes a definite advantage in higher dimensional simulations.

r-Refinement vs. h-refinement

In this research we perform mesh adaptation by points movement (r-refinement). An alternative is local mesh refinement (h-refinement), where mesh cells are split into smaller cells or merged again. An adaptive version of the previously used VAC package exists: AMRVAC [KNTG03, vdHK07]. It uses L mesh levels, where level 1 is the initial uniform mesh. We used AMRVAC with a refinement ratio of 2 on each level (i.e., splitting a cell into two equal pieces), hence the maximal mesh refinement is 2^{L-1}.

The advantage of h-refinement is its simplicity. The disadvantage is that the eventual number of mesh points is unknown, which can lead to unexpectedly long running times. Besides, good results require proper knowledge of the parameters by which the user controls refinement (initial mesh size, number of refinement levels, refinement ratios, and the tolerance level for deciding on local refinement). Our method is virtually free of user-defined parameters and is problem independent. For proper choices of refinement levels and tolerance, AMRVAC also produces good results.

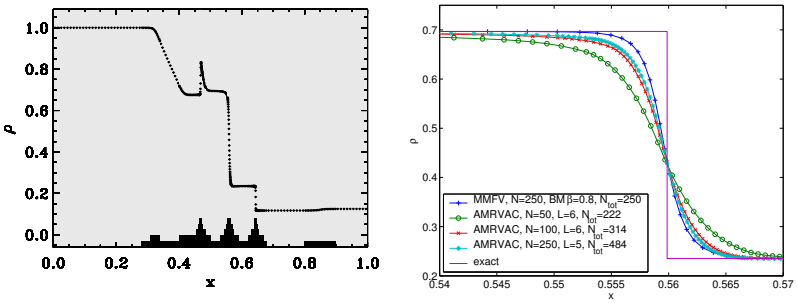

Figure 3.5 The Brio and Wu problem solved using h-refinement. Left: AMRVAC solution for the density. Black bars represent mesh refinement levels. Right: Comparison of r-refinement with h-refinement, detail at contact discontinuity.

We ran the Brio and Wu problem (3.29) again. Figure 3.5 shows the AMRVAC results. The left diagram shows the density, notice how all shocks are represented properly on the maximal refinement level. Also, both rarefaction waves are properly detected and refined. We used $N_{\text{initial}} = 100$ and six mesh levels, with tolerance $\epsilon_{\text{tol}} = 0.002$. All solution components are used equally in the error estimate for deciding on local refinement. The final mesh contains 314 mesh points and has an overall error of $1.4 \cdot 10^{-2}$. Running time is 5.1 seconds, which would roughly scale to 35 seconds in MATLAB. Our $N = 250$ result reaches a smaller error in 16 seconds.

The right diagram in Figure 3.5 compares three AMRVAC results with our $N = 250$ result. The smallest mesh cell in our experiment is 13 times smaller than in the original uniform mesh. This can be compared to five refinement levels ($2^{5-1} = 16$). The biggest mesh cells are between two and five times larger than the initial uniform cells. As AMRVAC does not coarsen its initial mesh, we start AMRVAC also with smaller mesh sizes ($N = 50$ and 100). The diagram also shows the final number of points; only for $N = 50$ this is less than 250. Table 3.1 summarizes the results and lists the overall errors.

Method	Initial N	L	ϵ_{tol}	Final N	Running time (+ MATLAB equiv.)	Overall error
MMFV	250	–	–	250	16.5 (16.5)	0.0091
AMRVAC:	50	6	0.005	222	2.4 (16.5)	0.0254
TVDLF,	100	6	0.002	314	5.1 (35.2)	0.0135
Van Leer	250	5	0.0005	484	7.2 (49.7)	0.0102

Table 3.1 Brio and Wu problem solved with r- and h-refinement.

AMRVAC contains more powerful methods as well: MUSCL-type solvers that use problem-specific Riemann solvers, and more sophisticated limiters. We used TVDLF with a Van Leer limiter for an equal comparison with our solver.

3.4.2 *MHD shock tube in 1.75D: physical energy loss*

The 1.5D Brio and Wu shock tube of the previous section can be extended to 1.75D. Keppens [Kep04] describes a problem where all seven MHD waves show up. The problem is set up in the domain $[0, 1]$, with the discontinuity at $x = 0.35$. We simulate for times $t \in [0, 0.08]$. The plasma has $\gamma = 5/3$ and $\bar{B}_1 = 1$. In primitive form, the initial conditions are:

$$\begin{aligned} [\rho, v_1, v_2, v_3, B_2, B_3, p]_L &= [0.5,\ 0,\ 1,\ 0.1,\ 2.5,\ 0,\ 0.1], \quad \text{if } x \leq 0.35, \\ [\rho, v_1, v_2, v_3, B_2, B_3, p]_R &= [0.1,\ 0,\ 0,\ \ 0,\ \ 2,\ 0,\ \ 0], \quad \text{if } x > 0.35. \end{aligned} \tag{3.30}$$

3.4.2.1 *Numerical results*

We use $N = 250$ mesh points again. The left diagram in Figure 3.6 shows the density and the v_3 component of the velocity. Note how the Alfvén signals do not change the density. Similarly, the contact discontinuity, the fast rarefaction

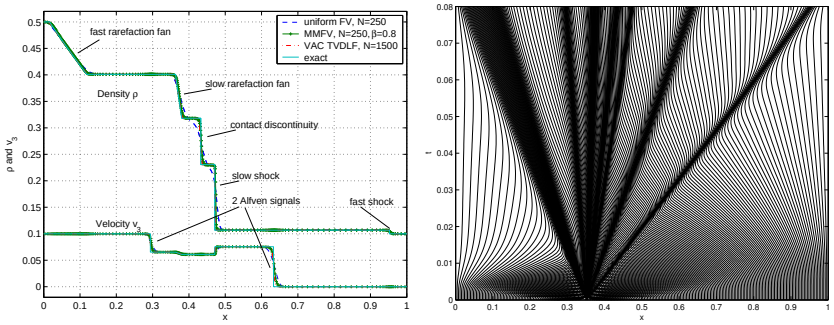

Figure 3.6 Solution to Keppens' 1.75D shock tube problem. Left: density and v_3 component of the velocity. Right: mesh history.

and the fast shock are not reflected in v_3. Still, the monitor is based on *all* solution components. Indeed, the right diagram in Figure 3.6 shows that the mesh movement captures all seven structures in a balanced way. The seven waves are directly related to the eigen-structure of the 1.75D MHD system, as depicted in Figure 3.1.

Throughout rarefaction fans, the entropy $s = c_v \log(p/\rho^\gamma)$ should remain constant. We have verified that this is indeed the case here. Also, for increasing N, the second-order accuracy of the finite volume solver in smooth regions was confirmed.

A final physical check here is the conservation of solution components. Mass conservation is satisfied, but energy conservation is not. Between $t = 0$ and $t = 0.08$, a constant decrease of energy yields a total loss of 0.2. This is exactly right though: at the left boundary the only nonzero part of the energy flux (3.8) is $\bar{B}_1 B_2 v_2 = 2.5$, whereas at the right boundary it is zero. Integrated over time, this should indeed cause a total energy loss of $0.08 \cdot 2.5 = 0.2$.

3.4.3 *Regular and critical solutions*

We now consider a more general 1.75D shock tube problem described by Torrilhon [Tor03] to investigate multiple possible solutions. The problem is set up in the domain $x \in [-1, 1.5]$ with the discontinuity at $x = 0$. We simulate for times $t \in [0, 0.4]$. In primitive form, the initial conditions are:

$$\begin{aligned} [\rho, v_1, v_2, v_3, B_2, B_3, p]_L &= [\ 1,\ 0,\ 0,\ 0,\quad 1,\quad 0,\quad 1], \quad \text{if } x \le 0, \\ [\rho, v_1, v_2, v_3, B_2, B_3, p]_R &= [0.2,\ 0,\ 0,\ 0,\ \cos\theta,\ \sin\theta,\ 0.2], \quad \text{if } x > 0. \end{aligned} \tag{3.31}$$

The problem is *non-planar* if the angle θ between the transversal parts (i.e., $[B_2, B_3]^T$) of $\mathbf{B}_L$ and $\mathbf{B}_R$ is not a multiple of π. Torrilhon describes how θ affects the possibility of multiple solutions. Regular r-solutions consist only of shocks or contact discontinuities, whereas critical c-solutions can also contain non-regular waves, such as compound waves. For critical choices of θ, both an r- and c-solution are analytically correct simultaneously; $\theta = \pi$ is such a choice. In the Brio and Wu example indeed the irregular compound

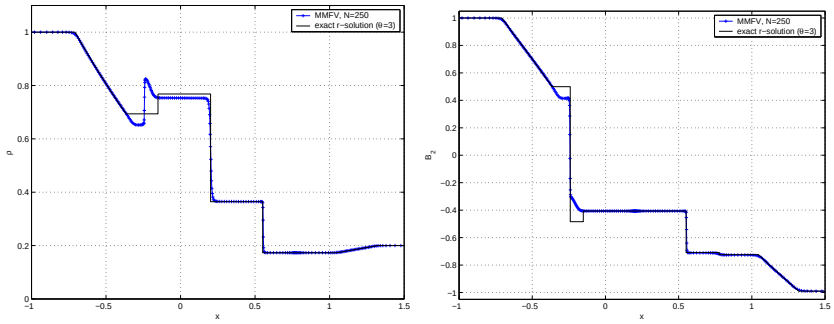

Figure 3.7 Solution to the almost co-planar problem (3.31) with $\theta = 3$. The adaptive solution uses $N = 250$ mesh points, and suffers from pseudo-convergence towards an incorrect c-solution.

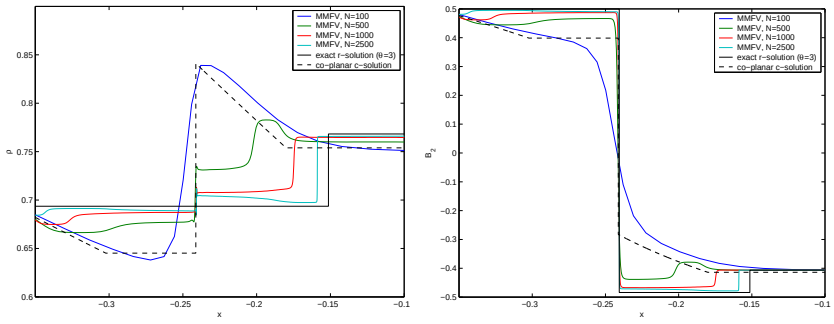

Figure 3.8 Convergence to correct solution for the almost co-planar problem (3.31, with $\theta = 3$).

wave from the c-solution showed up. It depends on the amount of numerical diffusion whether a PDE solver will converge to the r-solution.

3.4.3.1 *Numerical results*

We now consider the *almost* co-planar case $\theta = 3$. Analytically, this has only one r-solution. However, the numerical solution is attracted towards the nearby critical solution for $\theta = \pi$. Figure 3.7 shows the density and the B_2 component of the magnetic field. The solutions resemble the one to the Brio and Wu problem (a c-solution), but are clearly different from the correct r-solution here.

Increasing the number of mesh points results in smaller mesh cells, hence less numerical diffusion. We study, for increasing N, the convergence of our numerical solution towards the correct r-solution, just as Torrilhon [Tor03, Sec. 4.2.1] does. Figure 3.8 shows the density and the B_2 component of the magnetic field at $[-0.35, -0.1]$ for N up to 2500. The dashed line represents the co-planar c-solution to which the $N = 100$ solution clearly is attracted.

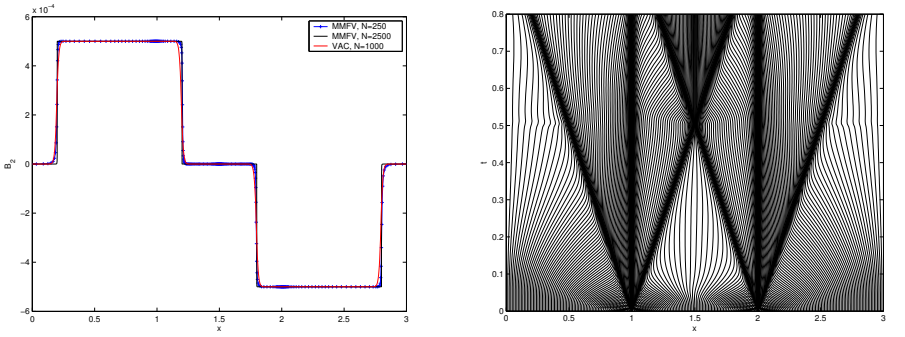

Figure 3.9 Moving mesh solution to shear Alfvén problem at $t = 0.8$, with $N = 250$ mesh points. Left: Transverse component B_2 of magnetic field. Right: mesh history.

For larger values of N, the solutions converge towards the solid black line of the correct r-solution. At $N = 1000$ the solution is about as good as the uniform $N = 20,000$ solution by Torrilhon: a considerable improvement.

3.4.4 *Shear Alfvén waves in 1.5D*

This test problem was described by Stone and Norman [SN92] and also used by Tóth and Odstrčil [TO96] for their evaluation of different discretization schemes. A homogeneous, uniformly magnetized plasma state is perturbed with a localized velocity pulse transverse ($v_2 := (m_2/\rho) \neq 0$) to the horizontal (x-direction) magnetic field. This evolves into two oppositely traveling Alfvén waves that have associated $v_2 := m_2/\rho$ and B_2 perturbations.

The problem is set up in the domain $x \in [0, 3]$, with the velocity pulse on $x \in [1, 2]$. We simulate for times $t \in [0, 0.8]$. The plasma has an adiabatic constant $\gamma = 1.4$, and $\bar{B}_1 = 1$. In conservation form, the initial conditions are:

$$\begin{aligned} [\rho, m_1, m_2, B_2, e] &= [1,\ 0,\ 10^{-3},\ 0,\ 0.5000005025], \quad \text{for } x \in [1, 2], \\ [\rho, m_1, m_2, B_2, e] &= [1,\ 0,\ \ \ \ 0,\ 0,\ 0.5000000025], \quad \text{elsewhere}, \end{aligned} \tag{3.32}$$

where the difference in total energy e is only caused by the difference in v_2. In primitive form, all quantities but v_2 are constant. Homogeneous Neumann boundary conditions are used for all components.

When considering linear effects, only v_2 and B_2 will be perturbed, and all other primitive quantities should remain constant. Quadratic terms in the flux for m_1, however, cause nonlinear effects in the density and energy. Furthermore, thermal pressure should always be positive.

3.4.4.1 *Numerical experiments*

Figure 3.9 shows the B_2 component of the magnetic induction at $t = 0.8$ from both the adaptive mesh solution and the reference solution. In the right diagram, the mesh history is shown. Again, the $N = 250$ adaptive solution compares favorably with the 1000 point VAC solution. Analytic computation

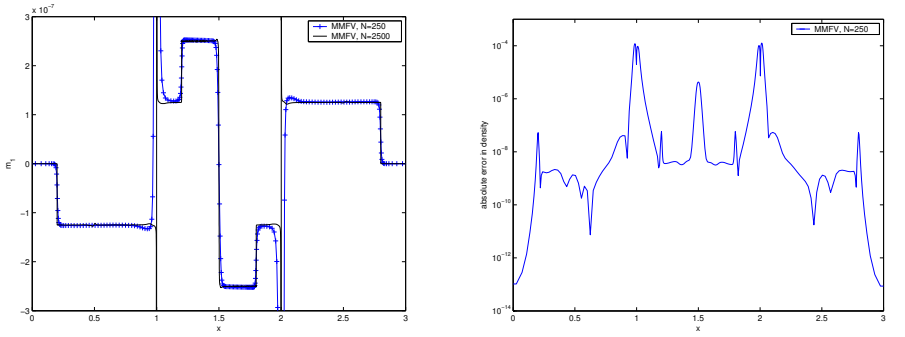

Figure 3.10 Nonlinear effects in the shear Alfvén problem. Left: Nonlinear effects in m_1. Right: Local errors in density

of the exact solution is more complicated than with the shock tube problems, because of the interacting right- and left-going waves. An $N = 2500$ adaptive solution shows the sharp profile here.

The mesh history reveals that some intermediate structures were captured too, although those are not in the B_2 (nor v_2) component. A closer look at the almost zero momentum shows levels slightly off from 0. These are caused by the nonlinear terms in the m_1 flux. The left diagram in Figure 3.10 shows multiple levels, instead of a constant value of 0. Not only do the physical equations justify these levels; changing the number of mesh points to 100 or 1000 results in the same levels. Furthermore, when changing the initial velocity perturbation from 10^{-3} to 10^{-6} changes the momentum offset from $O(10^{-7})$ to $O(10^{-13})$; clearly a quadratic effect.

The right diagram in Figure 3.10 shows the absolute, local errors in the density for the $N = 250$ solution, obtained by subtracting the 2500 points reference solution from it. At $x = 1$ and $x = 2$, local errors are the largest, at 10^{-4}. Elsewhere, errors are very small, $O(10^{-8})$, compared to VAC ($O(10^{-3})$) and the adaptive method of lines ($O(10^{-6})$, cf. Zegeling and Keppens [ZK01, Fig. 4]).

3.4.5 *Oscillating plasma sheet in 1.5D: fast wave effects*

To investigate the necessity of an implicit solver, Tóth et al. [TKB98] set up a problem that leads to a very strict CFL limit, i.e., very small time steps. A plasma sheet is surrounded by a vacuum which is modeled by a low density, low pressure plasma. At the left and right boundaries are perfectly conducting walls with reflective boundary conditions.

The problem is set up on the domain $x \in [0, 1]$, with the plasma sheet on $x \in [0.45, 0.55]$. We simulate for times $t \in [0, 2]$. The plasma has $\gamma = 1.4$, and

$\bar{B}_1 = 0$. In primitive form, the initial conditions are:

$$[\rho, m_1, m_2, B_2, p] = \begin{cases} [10^{-3},\ 0,\ 0,\ 1.1,\quad 10^{-4}], & \text{for } x \in [0, 0.45], \\ [\quad 1,\ 0,\ 0,\ 0.6,\ 0.3201], & \text{for } x \in [0.45, 0.55], \\ [10^{-3},\ 0,\ 0,\ 1.0,\quad 10^{-4}], & \text{for } x \in [0.55, 1]. \end{cases} \tag{3.33}$$

In the plasma sheet, the total pressure $p_{\text{tot}} = p + B^2/2 = 0.5001$ is in balance with the pressure in the 'vacuum' at the right, and is about 10% less than in the 'vacuum' at the left. Therefore the sheet will start to move rightward until the changing pressure imbalance reverses the movement leftward. Because of conservation of magnetic flux in the left and right 'vacuum', this will result in an ongoing oscillation of the sheet.

A reflective wall means zero flux for all components except for the ones orthogonal to the boundary, hence only m_1 is nonzero here. The zero fluxes can not be obtained by setting the values in the ghost cells outside the domain to zero. As fluxes are computed at cell edges, and solution values are set on cell centers, interpolation will yield slightly nonzero flux values on the boundaries. Instead, we make the m_1 values asymmetric around the two boundaries (i.e., $\mathbf{Q}_{-j-1/2} = -\mathbf{Q}_{j+1/2}$ at the left, see also Figure 3.2), and impose an exactly zero flux for all but the first momentum equation on the two boundaries (i.e., $F_0 = F_N = 0$, except for the second component of the flux vector F). Now, total mass, magnetic field and energy are conserved numerically up to machine precision.

3.4.5.1 *Numerical experiments*

We first look at the slow oscillation that should set in. The oscillating sheet can be approximated by a point mass with total mass $M = 0.1$ at distance $L_0 = 0.5$ from the walls with some equilibrium value B_0 for the magnetic field. The point mass oscillates around this equilibrium, driven by the difference in magnetic pressure between the left and right half. By conservation of magnetic flux (3.3), the total magnetic flux in the equilibrium and at an extremal position are equal:

$$\begin{aligned} B_L(L_0 + \Delta L) &= (B_0 - \Delta B)(L_0 + \Delta L) &&= B_0 L_0, \\ B_R(L_0 - \Delta L) &= (B_0 + \Delta B)(L_0 - \Delta L) &&= B_0 L_0. \end{aligned}$$

A linear approximation gives: $\Delta B/B_0 \approx \Delta L/L_0$. Describing the oscillation as $x(t) = L_0 + \Delta L \sin(\omega t)$, and differentiating twice gives $x''(t) = -\Delta L \omega^2 \sin(\omega t)$. Inserting this into $F = Mx''$ for the rightmost extremum gives: $-M\Delta L = B_L^2/2 - B_R^2/2 = -2B_0^2/L_0\Delta L$. The oscillation is now characterized by its frequency and amplitude:

$$\omega \approx \sqrt{\frac{2B_0^2}{ML_0}}, \text{ and } \Delta L \approx (\Delta B/B_0)L_0.$$

We estimate $B_0 \approx 0.5 \cdot 1.1 + 0.5 \cdot 1 = 1.05$ and $\Delta B \approx 0.1$. This yields $\omega \approx 6.64$, i.e., the period $T \approx 0.946$. The maximum of the total momentum Mv_1 is

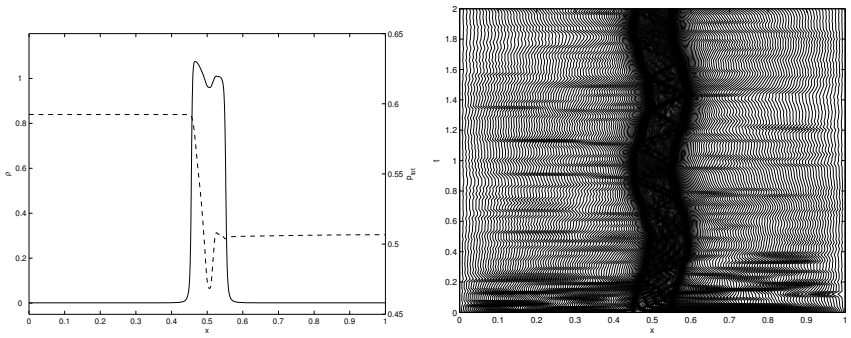

Figure 3.11 Oscillating plasma sheet. Left: density (solid line) and total pressure (dashed line) at $t = 1$. Right: mesh history over $t \in [0, 2]$.

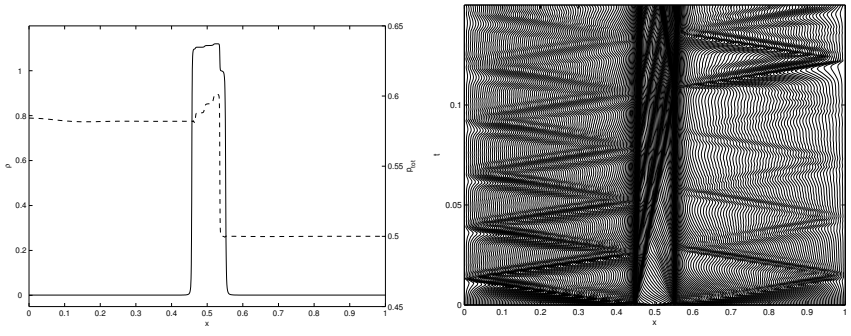

Figure 3.12 Oscillating plasma sheet, initial details. Left: density (solid line) and total pressure (dashed line) at $t = 0.1$. Right: mesh history over $t \in [0, 0.15]$.

$M\omega\Delta L \approx 0.158$. Our numerical experiments yield a period of $T = 0.942$ and a momentum amplitude of 0.15. This is quite accurate, considering the simplistic approximation sketched above.

A simulation up to $t = 2$ with $N = 250$ mesh points takes about 25000 time steps, and only 220 seconds to run, with the CFL number limited to 0.5. The right diagram in Figure 3.11 clearly shows how the adaptive mesh captures the oscillation. The left diagram shows the solution profiles of the density ρ and total pressure p_{tot} at $t = 1$. The oscillation is driven by the imbalance in magnetic (and hence total) pressure. In the diagram the sheet is moving rightward, because of high pressure at the left. The solution profile is much less diffused than in the results by Tóth et al. [TKB98, Fig. 3] and Zegeling et al. [ZK01, Fig. 5].

We now focus on fast waves in the solution and simulate for early times $t \in [0, 0.15]$. The right diagram in Figure 3.12 shows the mesh history in more detail for early times. Within the sheet, additional waves are tracked repeatedly. They are initiated by a wave that continuously moves through

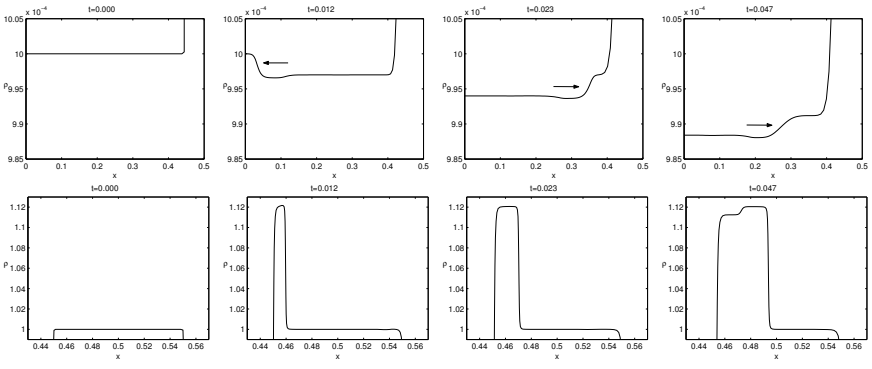

Figure 3.13 Oscillating plasma sheet, physical staircasing. Density profiles at $t = 0$, $t = 0.012$, $t = 0.023$, and $t = 0.047$. Top row: movement of a fast magnetosonic wave through the left 'vacuum'. Bottom row: staircase formation in the high density sheet.

the 'vacuum' between the left wall and the left edge of the sheet; it touches the sheet for the first time at $t \approx 0.026$. The left diagram again shows the density and total pressure, for $t = 0.1$. Both show 'physical staircasing' on top of their profile, initiated by three touches of the fast wave. Notice that a similar fast wave moves through the 'vacuum' at the right. The wave is less strong and hence causes hardly any 'staircasing' at first.

To study the formation of the 'physical staircase', Figure 3.13 shows four snapshots in time of the density profile. The top row shows the left 'vacuum' part and the bottom row shows the high density plasma sheet. Not the entire plasma sheet starts to oscillate at once: first only the left edge of the sheet slowly moves rightward. This leads to an increased density shock on top of the sheet that moves towards the right edge. The bottom diagrams show this expanding shock wave. Only when it touches the right edge, the entire sheet is in oscillation (not shown). In the meantime, other movement takes place as well. As the diagrams in the top row of Figure 3.13 show, a fast magnetosonic wave moves between the left wall and the left edge of the plasma sheet. The wave is reflected on both sides because of the reflective wall and the high density in the plasma sheet. The fast wave speed should be equal to $v_1 + c_f$, with c_f as defined in (3.9). As $v_1 = 0$ here, this yields $c_f \approx 34.8$ in the 'vacuum' at the left, and $c_f \approx 31.6$ in the 'vacuum' at the right. The wave speed at the left in our numerical simulation is equal to 34.85, which is very accurate. The first staircase formation is about to occur in the third column of Figure 3.13: the fast wave will soon touch the sheet edge. In the fourth column, the fast wave has almost completed its second period, and in the meantime the first step in the staircase has properly formed. This process will continue forever, although left-moving waves will start to interact after $t \approx 0.11$. We stress that the observed staircasing is definitely physical and should not be confused with numerical staircasing sometimes seen in finite volume methods. The recurring interactions between the fast wave and the

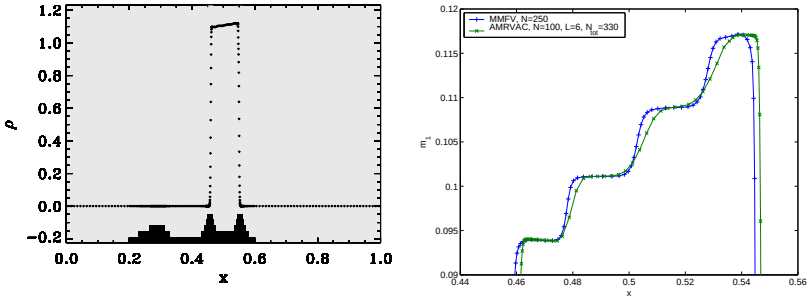

Figure 3.14 AMRVAC results for the staircasing effect in the oscillating plasma sheet problem. Left: AMRVAC solution for the density. Black bars represent mesh refinement levels. Right: Comparison of r-refinement with h-refinement, detail of m_1 component of moment in the plasma sheet.

plasma sheet are in fact repeated, distinct shock tube problems which change density and momentum levels in steps. Local shock tube experiments near the plasma's edge have also confirmed this.

Both Tóth et al. [TKB98] and Zegeling et al. [ZK01] have not shown the above fast wave effects. A probable explanation is their use of implicit time solvers, which take too big time steps to properly capture the fast waves. We also tested an explicit AMRVAC solution. Having seen that the staircasing is mainly visible in the m_1 component of the momentum, we base the refinement on m_1 by 80% and on the density by 20%. Again we use the TVDLF solver with a Van Leer limiter. The initial mesh has $N = 100$. The refinement tolerance ϵ_{tol} had to be lowered to 0.0005. Figure 3.14 shows the results. The left diagram shows the AMRVAC solution for the density. Notice how the refinement has properly detected the fast magnetosonic wave in the left vacuum. The right diagram focuses on the staircase formation in m_1 within the plasma sheet. It compares AMRVAC and our MMFV result. AMRVAC seems more diffused, and the refinement could be better at the stair steps. Running time was 37 seconds (Fortran), our MMFV run took 36 seconds (MATLAB).

3.5 CONCLUSIONS

Adaptive methods for solving PDE systems are a commonly used technique to increase numerical accuracy and save computing costs. Often, the adaptive methods are manually finetuned for the specific problem under consideration. A truly robust adaptive method should adapt itself to each new problem considered, without additional fine-tuning. In this chapter we considered such a method. Using a sophisticated monitor function, conservative solution interpolation and a robust finite volume solver, the method is suitable for any nonlinear system of hyperbolic PDEs based on conservation laws, where numerical conservation is guaranteed. After earlier successful application to

hyperbolic traffic flow PDEs [vD02] and problems from gas dynamics, we now used the method on a selection of problems from MHD.

Each of the example problems has one or more interesting physical features that were accurately tracked by the adaptive method. The 1.75D shock wave problem showed automatic and balanced refinement for all individual solution components, thanks to the monitor function used. The study of regular and critical solutions showed how nearby critical solutions are a strong attractor for numerical solutions. The use of our adaptive method shows convergence to the correct solution with 20 times fewer mesh points than for a uniform method. The shear Alfvén problem showed correct tracking and propagation of Alfvén waves. Moreover, nonlinear effects in the flux terms were accurately computed, with average errors of $O(10^{-8})$. Finally, the oscillating plasma sheet problem challenged the method because of the severe limit on the time step. Even after a large number of time steps, the important parts in the solution are tracked by our adaptive method. The oscillation that should set in is correctly represented. Moreover, the high speed magnetosonic waves in the two 'vacuum' parts turn out to cause a 'physical staircasing' in the plasma sheet. Although this effect can be explained from the physical formulas, it had not been studied before. The use of an adaptive method increased the accuracy sufficiently to let these effects show up noticeably in the numerical results.

The Brio and Wu shock tube problem was used to benchmark our adaptive method. The gain with respect to a uniform method is at least a factor three. For two- or higher-dimensional models this gain factor counts exponentially. The overall accuracy of the finite volume method is first order, due to first order accuracy of the method at discontinuities. Focusing on smooth parts, however, correctly shows the second order nature of the method. Also, a short comparison with h-refinement shows that our r-refinement method can reach smaller errors more efficiently.

Although Lax–Friedrichs-type methods are known for their numerical viscosity, the combination of a *local* Lax–Friedrichs flux in combination with a moving mesh yields very accurate results, with still good computational performance. We will extend the use of this robust adaptive technique to higher-dimensional models. The use of higher-order solvers, and a more accurate solution interpolation step during mesh moving, are possible future improvements during that process.

Acknowledgments

We wish to thank Rony Keppens at the FOM Institute for Plasma Physics Rijnhuizen, for his valuable help with the physical aspects of ideal MHD and assistance in using the VAC and AMRVAC packages. We are also grateful to Manuel Torrilhon at the Hong Kong University of Science and Technology, for kindly providing exact solutions to the shock tube problems in Section 3.4. Huazhong Tang at Peking University gave some additional details on the numerical scheme used in [TT03].

Balanced Monitoring of Flow Phenomena

4

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 chapter[1] we generalize our monitor function from the previous chapter 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.

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.

4.1 INTRODUCTION

Adaptive mesh methods improve local resolution of numerical solvers and, as a result, improve the performance of them. Results are significantly sharper than those obtained by using a uniform mesh with more mesh points. True gain in performance is only obtained, 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 chapter presents such a balanced monitoring, combined with a powerful finite volume solver, applied to hydrodynamical problems.

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. An overview of the aspects involved in this method was given in Section 2.6. In one of our experiments (Section 4.5) we will make a comparison between our r-refinement results and h-refinement results produced by AMRVAC [NK02, vdHK07].

r-Refinement or *(adaptive) moving mesh refinement* moves mesh points towards regions that need refinement based on some criterion. The number

[1] This chapter will appear as [vDZ09]: A. van Dam and P.A. Zegeling. *Balanced monitoring of flow phenomena in moving mesh methods.* To appear in Commun. Comput. Phys., 2009.

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. This chapter deals with r-refinement only and shows that it can achieve great levels of detail. The previous chapter contains a detailed overview of the wide range of approaches possible for r-refinement.

We employ a variational formulation of mesh adaptation, an approach which has become well-known over the past five decades. Tang and Tang [TT03] 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 [Tan06] and Zegeling et al [ZdBT05], and to magnetohydrodynamics (MHD), e.g., by Han and Tang [HT07], Tan [Tan07], Van Dam and Zegeling [vDZ06] and Zegeling [Zeg05]. 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. [DLTZ05] and the Hamilton–Jacobi equations by Tang et al. [TTZ03]. A similar method, but now using direct minimization of the mesh functional has been used for reactive flows in 2D by Azarenok and Tang [AT05] and multi-phase fluids in 3D by Di et al. [DLT08].

Inspired by earlier work by Beckett and Mackenzie [BM00] and Huang [Hua01a], we formulated an adaptive monitor function that makes manual fine-tuning unnecessary in the previous chapter. 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 [Hua06].

Brackbill has done similar research on the combination of several functionals in the minimization process (see Section 4.3.3.1). Besides combining mesh quality functionals [BS82, with Saltzman] he also combined an alignment functional with a solution adaptivity functional in order to obtain directional control [Bra93]. Directional monitor functions have been widely used ever since, for example by Glasser et al. [GLK05] in a more analytical context. Tang [Tan06] applies a directional monitor function to the two-dimensional Euler equations and Tan [Tan07] 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.

This chapter is organized as follows. In Section 4.2 we briefly recall the physics behind compressible gas flow and mention some relevant flow quantities. Next, we give a detailed description of our moving mesh finite volume solver in Section 4.3. The first part concerns the finite volume solver with HLLC fluxes, non-uniform solution reconstruction and slope limiting. The

second part concerns the mesh movement, its history and our current algorithm. Section 4.4 contains the main contribution of this chapter: a *balanced monitor function* to capture various flow phenomena. Section 4.5 contains three example problems that were already partly used in the preceding sections and are then further explored. In Section 4.6 we summarize our findings and give some recommendations for further research.

4.2 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. We defer the latter problem to the next chapter.

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.

4.2.1 *Physical model*

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

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ E\end{bmatrix} + \nabla\cdot\begin{bmatrix}\rho\mathbf{v}\\ \rho\mathbf{v}\mathbf{v}+pI\\ (E+p)\mathbf{v}\end{bmatrix} = \begin{bmatrix}0\\ \mathbf{0}\\ 0\end{bmatrix}, \tag{4.1}$$

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 (4.1) 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.

4.2.2 *Relevant flow features*

Analysis of shock waves and other features in compressible gas flow is easiest in a one-dimensional setting. For completeness, Figure 4.1 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

$$p = (\gamma-1)(E - \frac{1}{2}\rho\mathbf{v}\cdot\mathbf{v}) \tag{4.2}$$

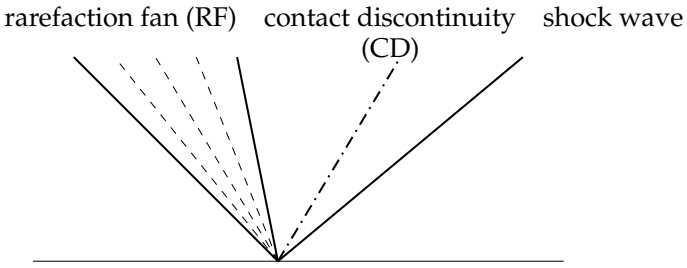

Figure 4.1 Elementary wave structure of the hyperbolic Euler equations for one-dimensional compressible gas flow. The three wave types form the building blocks of wave interactions in two dimensions.

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

$$S = \log\left(\frac{p}{\rho^\gamma}\right), \tag{4.3}$$

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. [SRCG93] give a classification of fifteen different solutions for two-dimensional Riemann problems, which was later corrected by Lax and Liu [LL98] 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 4.5 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.

4.3 A 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 4.3.2 and the mesh movement in Section 4.3.3.

4.3.1 *Physical problem description*

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

$$\frac{\partial}{\partial t}\mathbf{q} + \frac{\partial}{\partial x}\mathbf{f}(\mathbf{q}) + \frac{\partial}{\partial y}\mathbf{g}(\mathbf{q}) = \mathbf{0}, \quad \mathbf{q}([x,y]^T,t) \in \mathbb{R}^M. \tag{4.4}$$

The Euler equations for compressible gas dynamics (4.1) 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.

The domain Ω is defined and discretized as follows:

$$\Omega := [a_1,b_1]\times[a_2,b_2] = \bigcup_{j=0,\ldots,N_x-1,\ k=0,\ldots,N_y-1} A_{j+\frac{1}{2},k+\frac{1}{2}}, \tag{4.5}$$

$$A_{j+\frac{1}{2},k+\frac{1}{2}} := \text{quadrilateral cell with corners } \mathbf{x}_{j+c,k+d},\ c,d\in\{0,1\}, \tag{4.6}$$

$$\mathbf{x}_{j,k} := [x_{j,k},y_{j,k}]^T \quad \text{for} \quad j=-2,\ldots,N_x+2, \quad \text{and} \quad k=-2,\ldots,N_y+2. \tag{4.7}$$

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.

4.3.2 *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:

$$\mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} \approx \iint\limits_{A_{j+\frac{1}{2},k+\frac{1}{2}}} \mathbf{q}([x,y],t_n)\,\mathrm{d}x\,\mathrm{d}y/|A^n_{j+\frac{1}{2},k+\frac{1}{2}}|, \tag{4.8}$$

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 (4.4) leads to the well-known finite volume discretization:

$$\begin{aligned}\mathbf{Q}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} = \mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} - \frac{\Delta t_n}{|A^n_{j+\frac{1}{2},k+\frac{1}{2}}|}\Big(&|h^n_{j+1,k+\frac{1}{2}}|\breve{\mathbf{F}}_{j+1,k+\frac{1}{2}} - |h^n_{j,k+\frac{1}{2}}|\breve{\mathbf{F}}_{j,k+\frac{1}{2}} \\ &+|h^n_{j+\frac{1}{2},k+1}|\breve{\mathbf{G}}_{j+\frac{1}{2},k+1} - |h^n_{j+\frac{1}{2},k}|\breve{\mathbf{G}}_{j+\frac{1}{2},k}\Big) \\ =: \mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t_n L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^n),&\end{aligned} \tag{4.9}$$

where $\breve{\mathbf{F}}$ and $\breve{\mathbf{G}}$ 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 $|h_{j,k+\frac{1}{2}}|$.

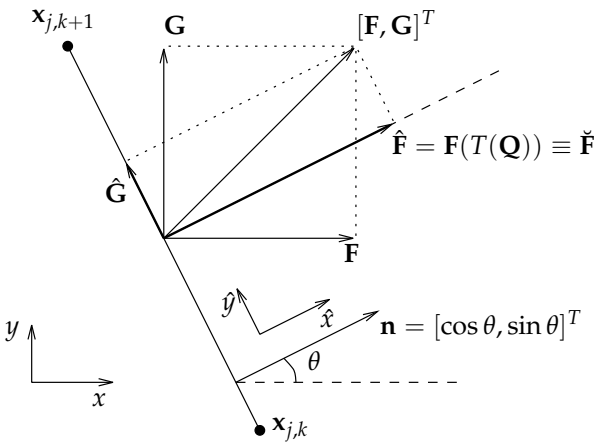

Figure 4.2 Rotation of the flux tensor into the (logically vertical) edge's normal reference frame. For ease of notation, $\mathbf{F}$ denotes the tensor $[\mathbf{F}, \mathbf{0}]$, and similar for $\mathbf{G}$, $\hat{\mathbf{F}}$ and $\hat{\mathbf{G}}$.

Figure 4.2 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 $[\mathbf{F}, \mathbf{G}]$ with the edge's normal, but here we can instead exploit the rotational invariance of the Euler equations:

$$\check{\mathbf{F}}(\mathbf{Q}) = \cos(\theta)\mathbf{F}(\mathbf{Q}) + \sin(\theta)\mathbf{G}(\mathbf{Q}) = T^{-1}(\mathbf{F}(T(\mathbf{Q}))), \tag{4.10}$$

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

$$T(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T^{-1}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.11}$$

Equation (4.10) describes how $\mathbf{Q}$ is first rotated over an angle θ into a new reference frame as $\hat{\mathbf{Q}} = T(\mathbf{Q})$ where the new $\hat{x}$-direction aligns with the edge's normal. Now, only $\mathbf{F}(T(\mathbf{Q}))$ needs to be evaluated, since flux $\mathbf{G}(T(\mathbf{Q}))$ 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 4.3.2.2. 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.

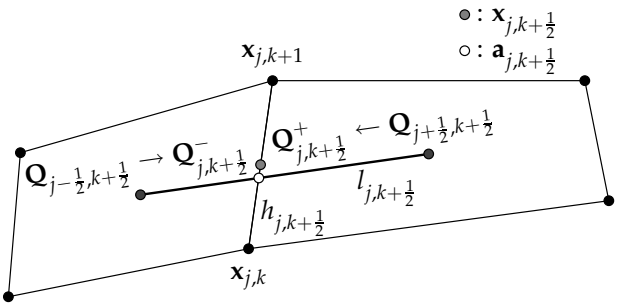

Figure 4.3 Solution reconstruction on a nonuniform mesh.

4.3.2.1 *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 [vL77a, vL77b], 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 Figure 4.3. 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:

$$\mathbf{a}^n_{j,k+\frac{1}{2}} := l^n_{j,k+\frac{1}{2}} \cap h^n_{j,k+\frac{1}{2}} \approx (\mathbf{x}^n_{j,k} + \mathbf{x}^n_{j,k+1})/2 =: \mathbf{x}^n_{j,k+\frac{1}{2}}. \tag{4.12}$$

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

$$\mathbf{Q}^{n,\pm}_{j,k+\frac{1}{2}} = \mathbf{Q}^n_{j\pm\frac{1}{2},k+\frac{1}{2}} \mp \|\mathbf{x}^n_{j,k+\frac{1}{2}} - \mathbf{x}^n_{j\pm\frac{1}{2},k+\frac{1}{2}}\|_2 \, \bar{\mathbf{S}}^n_{j\pm\frac{1}{2},k+\frac{1}{2}}, \tag{4.13}$$

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

$$\bar{\mathbf{S}}^n_{j+\frac{1}{2},k+\frac{1}{2}} = \phi(\mathbf{S}^n_{j+1,k+\frac{1}{2}}/\mathbf{S}^n_{j,k+\frac{1}{2}})\mathbf{S}^n_{j,k+\frac{1}{2}}, \tag{4.14}$$

$$\mathbf{S}^n_{j,k+\frac{1}{2}} = \frac{\mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} - \mathbf{Q}^n_{j-\frac{1}{2},k+\frac{1}{2}}}{\|\mathbf{x}^n_{j+\frac{1}{2},k+\frac{1}{2}} - \mathbf{x}^n_{j-\frac{1}{2},k+\frac{1}{2}}\|}, \tag{4.15}$$

where ϕ is the Van Leer limiter:

$$\phi(r) = \frac{r + |r|}{1 + r}. \tag{4.16}$$

Other well-known slope limiters, such as the Woodward or Koren limiters may be used instead. Note that the reconstruction (4.13) 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 [Hub99] and Berger et al. [BAM05]. In the latter, this even involves solving linear programming problems at each cell interface. Our experience is that our quasi-one-dimensional approach (4.13) proves robust for a wide range of problems.

4.3.2.2 *Approximate Riemann solvers*

The numerical fluxes in (4.9) are supposed to be averaged at the edge over the time interval $[t_n, t_{n+1}]$. The reconstructed solution values

$$\mathbf{Q}_L := \mathbf{Q}^{n,-}_{j,k+\frac{1}{2}} \text{ and } \mathbf{Q}_R := \mathbf{Q}^{n,+}_{j,k+\frac{1}{2}} \tag{4.17}$$

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 [Rus61]—flux averages the left and right fluxes and adds a stabilizing numerical diffusion locally:

$$\mathbf{F}^{\text{llf}}(\mathbf{Q}_L, \mathbf{Q}_R) = \frac{1}{2}(\mathbf{f}(\mathbf{Q}_L) + \mathbf{f}(\mathbf{Q}_R)) - \frac{1}{2} \max_{K \in \{L,R\}} |\lambda_{\max}(\mathbf{Q}_K)|(\mathbf{Q}_R - \mathbf{Q}_L), \tag{4.18}$$

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 [vDZ06]. 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 4.5.1.1 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 (4.18). 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. [HLL83] proposed a new approximate Riemann solver to obtain Godunov-type fluxes, which is now widely known as the HLL Riemann

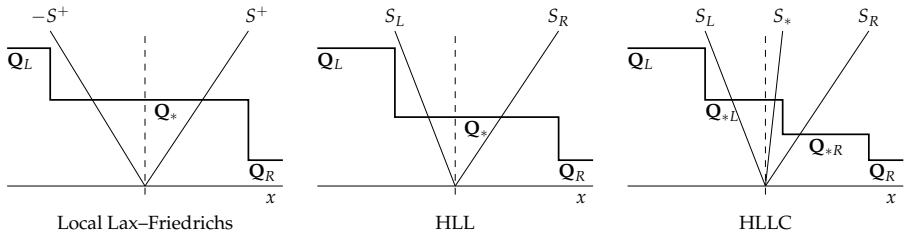

Figure 4.4 Approximate Riemann solvers: Local Lax–Friedrichs, HLL and HLLC. The difference lies in the number of different wave speeds that are used. In between waves, the solution is approximated by a constant state.

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 [Tor99].

Toro et al. [TSS94] 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 Figure 4.4.

The local Riemann problem $(\mathbf{Q}_L, \mathbf{Q}_R)$ is one-dimensional due to the rotated reference frame (see Section 4.3.2). 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:

$$\mathbf{Q}_j^{\text{hllc}} = \begin{cases} \mathbf{Q}_L & \text{if } \tilde{x}/t \le S_L, \\ \mathbf{Q}_{*L} & \text{if } S_L \le \tilde{x}/t \le S_*, \\ \mathbf{Q}_{*R} & \text{if } S_* \le \tilde{x}/t \le S_R, \\ \mathbf{Q}_R & \text{if } S_R \le \tilde{x}/t. \end{cases} \tag{4.19}$$

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:

$$\mathbf{Q}_{*K} = \rho_K \left(\frac{S_K - u_K}{S_K - S_*} \right) \begin{bmatrix} 1 \\ S_* \\ v_K \\ \frac{E_K}{\rho_K} + (S_* - u_K)\left(S_* + \frac{p_K}{\rho_K (S_K - u_K)} \right) \end{bmatrix} \quad \text{for } K \in \{L, R\}. \tag{4.20}$$

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

$$\mathbf{F}^{\text{hllc}}_{j+\frac{1}{2}} = \begin{cases} \mathbf{F}_L & \text{if } 0 \le S_L, \\ \mathbf{F}_{*L} = \mathbf{F}_L + S_L(\mathbf{Q}_{*L} - \mathbf{Q}_L) & \text{if } S_L \le 0 \le S_*, \\ \mathbf{F}_{*R} = \mathbf{F}_R + S_R(\mathbf{Q}_{*R} - \mathbf{Q}_R) & \text{if } S_* \le 0 \le S_R, \\ \mathbf{F}_R & \text{if } S_R \le 0. \end{cases} \tag{4.21}$$

The left- and rightmost wave speeds are chosen as follows:

$$S_L = \min\{(u-c)_L, (u-c)_R\} \quad \text{and} \quad S_R = \max\{(u+c)_L, (u+c)_R\}, \tag{4.22}$$

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:

$$S_* = \frac{p_R - p_L + \rho_L u_L(S_L - u_L) - \rho_R u_R(S_R - u_R)}{\rho_L(S_L - u_L) - \rho_R(S_R - u_R)}. \tag{4.23}$$

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.

4.3.2.3 *Two-step explicit time integration*

The PDEs (4.4) are integrated in time by Heun's predictor–corrector method, which has second-order accuracy. Using the notation from (4.9), we now use two FV-steps:

$$\begin{aligned} \mathbf{Q}^*_{j+\frac{1}{2},k+\frac{1}{2}} &= \mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t_n L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^n), \\ \mathbf{Q}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} &= \mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t_n \frac{L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^n) + L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^*)}{2}, \end{aligned} \tag{4.24}$$

where the $L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q})$ operator computes the discretized 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:

$$\Delta\hat{x}^n_{j+\frac{1}{2},k+\frac{1}{2}} := \frac{|A^n_{j+\frac{1}{2},k+\frac{1}{2}}|}{\max\{|h^n_{j,k+\frac{1}{2}}|,\ |h^n_{j+1,k+\frac{1}{2}}|\}}, \quad \Delta\hat{y}^n_{j+\frac{1}{2},k+\frac{1}{2}} := \frac{|A^n_{j+\frac{1}{2},k+\frac{1}{2}}|}{\max\{|h^n_{j+\frac{1}{2},k}|,\ |h^n_{j+\frac{1}{2},k+1}|\}}. \tag{4.25}$$

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

$$\Delta t^n \le \mathcal{C} \min_{j,k} \frac{\min\{\Delta\hat{x}^n_{j+\frac{1}{2},k+\frac{1}{2}},\ \Delta\hat{y}^n_{j+\frac{1}{2},k+\frac{1}{2}}\}}{\max\{|\lambda_{1,\max}(\mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}})|,\ |\lambda_{1,\max}(\mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}})|\}}, \tag{4.26}$$

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, although it requires precise bookkeeping and correction of fluxes to maintain the conservation property over all time steps. Tan et al. [TZHT04] have done so and report performance improvements by a factor 2 to 3.

4.3.3 *Adaptive moving mesh method*

The problem domain is discretized 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.

4.3.3.1 *Background*

The following background information supports the presentation of the algorithm; a more in-depth discussion of these methods was given in Section 2.3.2.

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 on mesh adaptation is by Winslow [Win81] in which he generalizes the elliptic mesh generator to a solution-adaptive form:

$$\nabla \cdot (D \nabla \xi) = 0, \quad \nabla \cdot (D \nabla \eta) = 0. \tag{4.27}$$

Here the 'diffusion' coefficient $D > 0$ could depend on the solution gradient. Instead of inverting (4.27) to obtain $\mathbf{x}(\boldsymbol{\xi})$, Ceniceros and Hou [CH01] formulate the elliptic generator directly in the covariant form:

$$\nabla_\xi \cdot (G \nabla_\xi x) = 0, \quad \nabla_\xi \cdot (G \nabla_\xi y) = 0, \tag{4.28}$$

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 G are still scalar functions (take $G = gI$), but later work by, e.g., Cao et al. [CHR99] and Tang [Tan06] proposes to make 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.[2]

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 (4.28) is the minimizer of the 'energy' functional

$$E(\mathbf{x}(\boldsymbol{\xi})) = \frac{1}{2} \iint_{\Omega_c} \left[\nabla_\xi^T G \nabla_\xi x + \nabla_\xi^T G \nabla_\xi y \right] \, \mathrm{d}\xi \, \mathrm{d}\eta. \tag{4.29}$$

[2] Huang [CHR99] even proposes nonzero elements off the diagonal, but we do not consider that here.

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 G.

Brackbill and Saltzman [BS82] 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 (4.45) in Section 4.4 helps keeping the mesh smooth. Still, we do see advantages in orthogonality monitors in future work.

4.3.3.2 *Algorithm*

The mesh movement algorithm is similar to the one set out by Tang and Tang [TT03]. We propose a much more versatile and robust monitor function, though, which will be described in Section 4.4.

The mesh movement equations (4.28) are solved separately from the physical PDEs (4.4). 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 2 summarizes it all.

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

$n \leftarrow 0$; $t_0 \leftarrow 0$
Generate an initial uniform mesh $\mathbf{x}^0_{j,k}$.
Compute initial values $\mathbf{Q}^0_{j+1/2,k+1/2}$.
while $t_n < T$ **do**
  $\nu \leftarrow 0$; $\mathbf{x}^{[0]}_{j,k} \leftarrow \mathbf{x}^n_{j,k}$; $\mathbf{Q}^{[0]}_{j+1/2,k+1/2} \leftarrow \mathbf{Q}^n_{j+1/2,k+1/2}$.
  repeat
    Evaluate monitor function (4.45) and filter it (Section 4.4.5).
    Move mesh $\mathbf{x}^{[\nu]}_{j,k}$ to $\mathbf{x}^{[\nu+1]}_{j,k}$, by Gauss–Seidel of (4.30).
    Conservative interpolation of $\mathbf{Q}^{[\nu+1]}_{j+1/2,k+1/2}$ by (4.32)
    (or re-initialize at $t = t_0$).
    $\nu \leftarrow \nu + 1$
  until $\nu \geq \nu_{\max}$ or $\left\|\mathbf{x}^{[\nu]} - \mathbf{x}^{[\nu-1]}\right\|_{\text{rel}} \leq \epsilon$
  Fix new mesh $\mathbf{x}^n \leftarrow \mathbf{x}^{[\nu]}$ and solution $\mathbf{Q}^n \leftarrow \mathbf{Q}^{[\nu]}$.
  $t_{n+1} \leftarrow t_n + \Delta t_n$ by CFL criterion (4.26).
  Compute $\mathbf{Q}^{n+1}$ using finite volumes (Section 4.3.2).
  $\mathbf{x}^{n+1} \leftarrow \mathbf{x}^n$.
  $n \leftarrow n + 1$.
end while

Mesh adaptation

We combine the moving mesh PDEs (4.28) with a directional[3] monitor function $G = \mathrm{diag}(\omega^{(1)}, \omega^{(2)})$. All gradients are discretized 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:

$$\mathbf{x}_{j,k}^{[\nu+1]} = \frac{\dfrac{\omega^{(1)}_{j-\frac{1}{2},k}\mathbf{x}^{[\nu+1]}_{j-1,k} + \omega^{(1)}_{j+\frac{1}{2},k}\mathbf{x}^{[\nu]}_{j+1,k}}{(\Delta\xi)^2} + \dfrac{\omega^{(2)}_{j,k-\frac{1}{2}}\mathbf{x}^{[\nu+1]}_{j,k-1} + \omega^{(2)}_{j,k+\frac{1}{2}}\mathbf{x}^{[\nu]}_{j,k+1}}{(\Delta\eta)^2}}{(\omega^{(1)}_{j-\frac{1}{2},k} + \omega^{(1)}_{j+\frac{1}{2},k} + \omega^{(2)}_{j,k-\frac{1}{2}} + \omega^{(2)}_{j,k+\frac{1}{2}})}. \tag{4.30}$$

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 $x_{0,k} = a_1$, $y_{0,k} = y_{1,k}$ in (4.30) 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 [TT03], which considers the velocity of the mesh points as an artificial flux. Han and Tang [HT07] formulate an alternative geometrical approach, which may be slightly more accurate. Both maintain global conservation in the following way:

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

Zhang [Zha06] devises a new approach based on L^2-projection, where solution conservation is even preserved in each cell. We employ the first method, though, because of our good experiences with it in the past.

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

$$\mathbf{c}_{j,k} := \mathbf{x}^{[\nu]}_{j,k} - \mathbf{x}^{[\nu+1]}_{j,k}. \tag{4.31}$$

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

$$\begin{aligned}\left|A^{[\nu+1]}_{j+\frac{1}{2},k+\frac{1}{2}}\right| \mathbf{Q}^{[\nu+1]}_{j+\frac{1}{2},k+\frac{1}{2}} = &\left|A^{[\nu]}_{j+\frac{1}{2},k+\frac{1}{2}}\right| \mathbf{Q}^{[\nu]}_{j+\frac{1}{2},k+\frac{1}{2}} \\ &- \left((\widehat{c_n\mathbf{Q}})_{j+1,k+\frac{1}{2}}|h_{j+1,k+\frac{1}{2}}| - (\widehat{c_n\mathbf{Q}})_{j,k+\frac{1}{2}}|h_{j,k+\frac{1}{2}}|\right) \\ &- \left((\widehat{c_n\mathbf{Q}})_{j+\frac{1}{2},k+1}|h_{j+\frac{1}{2},k+1}| - (\widehat{c_n\mathbf{Q}})_{j+\frac{1}{2},k}|h_{j+\frac{1}{2},k}|\right).\end{aligned} \tag{4.32}$$

[3] See Section 4.4.4. The direction indices (1), (2) are in superscript here, to avoid confusion with point indices j, k.

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

$$\widehat{c_n \mathbf{Q}}_{j,k+\frac{1}{2}} = (c_n^+ \mathbf{Q})^-_{j,k+\frac{1}{2}} + (c_n^- \mathbf{Q})^+_{j,k+\frac{1}{2}}, \tag{4.33}$$

where the two MUSCL-type solution reconstructions $\mathbf{Q}^-$ and $\mathbf{Q}^+$ are again defined by (4.13). The upwind choice is defined by:

$$c_n^\pm = \frac{c_n \pm |c_n|}{2} \tag{4.34}$$

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

$$(c_n)_{j,k+\frac{1}{2}} = \frac{\mathbf{c}_{j,k} + \mathbf{c}_{j,k+1}}{2} \cdot \mathbf{n}_{j,k+\frac{1}{2}}, \tag{4.35}$$

$$\mathbf{n}_{j,k+\frac{1}{2}} = \frac{[y_{j,k+1} - y_{j,k},\ -(x_{j,k+1} - x_{j,k})]^T}{\|\mathbf{x}_{j,k+1} - \mathbf{x}_{j,k}\|}. \tag{4.36}$$

Degeneracy of the mesh

The solution of the mesh PDEs (4.28) is a mesh map $\mathbf{x}(\boldsymbol{\xi})$, which is *unique* and *regular* as long as the monitor matrix G is diagonal and strictly positive. This result is a special case of the proof by Clément et al. [CHS96], which we describe in Section 2.3.2.8. In other words: the mesh map has a strictly positive Jacobian $J := \det((\nabla_\xi \mathbf{x})^T)$, so mesh cells can not collapse in the exact solution.

The nonlinear mesh PDEs (4.28) 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. One might put a 'barrier' on the movement of the mesh nodes, as we describe in Section A.1.2, but this was not necessary for any of the results presented in this chapter.

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

4.4 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.

4.4.1 *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:

$$\omega = \sqrt{1 + \alpha \|\nabla q\|^2}. \tag{4.37}$$

The assumption of a scalar solution q is for simplicity here and will be extended to vector-valued solutions $\mathbf{q}$ in Section 4.4.3. 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.

4.4.2 *An adaptive monitor function*

The disadvantages of the adaptivity parameter α in the AL monitor (4.37) mentioned in the previous section, can be summarized as follows: α is not dimensionless, nor scaling invariant, nor time-dependent. Beckett and Mackenzie (BM) [BM00] 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 (4.37) 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:

$$\omega(q) = 1 + \frac{\|\nabla q\|^{\frac{1}{m}}}{\alpha(q)}, \text{ where } \alpha(q) = \iint_{\Omega} \|\nabla q\|^{\frac{1}{m}} \, \mathrm{d}\mathbf{x}, \quad m > 0. \tag{4.38}$$

The next section will generalize this to the case where the solution is a vector $\mathbf{q}$, i.e., as in (4.4). 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 [CH01]. 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 (4.29).

The above monitor assigns approximately half of the mesh points to 'important' areas (see for example [Hua01a]). 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 (4.39)) that gives the user generic refinement control. All experiments in Section 4.5 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.

4.4.3 *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 $\phi_{i,p}(\mathbf{q})$:

$$\omega_i(\mathbf{q}) = \sum_{p=1}^{P} \omega_{i,p}(\mathbf{q}) = \sum_{p=1}^{P} \left[(1-\beta) + \frac{\beta}{\alpha_{i,p}(\mathbf{q})} \phi_{i,p}(\mathbf{q}) \right], \quad i \in \{1,2\}, \tag{4.39}$$

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

$$\alpha_{i,p}(\mathbf{q}) = \max[\iint_{\Omega_c} \phi_{i,p}(\mathbf{q}) \, \mathrm{d}\boldsymbol{\xi}, \ \epsilon]. \tag{4.40}$$

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 4.4.3.1.

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

$$\phi_{i,p}(\mathbf{q}) = \left| \frac{\partial q_p}{\partial \xi_i} \right|^{1/m}, \quad p = 1, \ldots, M, \text{ i.e., } P = M. \tag{4.41}$$

Other monitor components will be used in Sections 4.4.3.2 and 4.5.

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.

4.4.3.1 *Component imbalance*

The adaptive monitor function (4.39) 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 (4.42) below, the components were divided by the *average* as in (4.43). 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 (4.39,4.40,4.41)). Starting from the AL monitor (4.37) a possible way of balancing monitor components

would be standard normalization by dividing by the maximum component value:

$$\omega_p(\mathbf{q}) = 1 + \frac{\phi_p(\mathbf{q})}{\max_\Omega \phi_p(\mathbf{q})} \equiv 1 + \frac{\phi_p(\mathbf{q})}{\mathcal{M}_p(\mathbf{q})} \qquad \in [1,2]. \tag{4.42}$$

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 (4.39) uses the average value $\alpha_p(\mathbf{q})$ for normalization:

$$\omega_p(\mathbf{q}) = 1 + \frac{\phi_p(\mathbf{q})}{\int\int_{\Omega_c} \phi_p(\mathbf{q}) \mathrm{d}\boldsymbol{\xi}} \equiv \frac{\alpha_p(\mathbf{q}) + \phi_p(\mathbf{q})}{\alpha_p(\mathbf{q})} \qquad \in \left[1, 1 + \frac{\mathcal{M}_p(\mathbf{q})}{\alpha_p(\mathbf{q})}\right]. \tag{4.43}$$

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 (4.43):

$$\omega_p(\mathbf{q}) = \frac{(\alpha_p(\mathbf{q}) + \phi_p(\mathbf{q}))/\alpha_p(\mathbf{q})}{\max\limits_\Omega[(\alpha_p(\mathbf{q}) + \phi_p(\mathbf{q}))/\alpha_p(\mathbf{q})]} \equiv \frac{\alpha_p(\mathbf{q}) + \phi_p(\mathbf{q})}{\alpha_p(\mathbf{q}) + \mathcal{M}_p(\mathbf{q})} \qquad \in \left[\frac{\alpha_p(\mathbf{q})}{\alpha_p(\mathbf{q}) + \mathcal{M}_p(\mathbf{q})}, 1\right] \subseteq \langle 0, 1]. \tag{4.44}$$

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

$$\omega_i(\mathbf{q}) = \sum_{p=1}^{P} \omega_{i,p}(\mathbf{q}) = \sum_{p=1}^{P} \left[\frac{(1-\beta)\alpha_{i,p} + \beta\,\phi_{i,p}(\mathbf{q})}{(1-\beta)\alpha_{i,p} + \beta\mathcal{M}_{i,p}(\mathbf{q})}\right], \quad i \in \{1,2\}. \tag{4.45}$$

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 (4.42) 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 (4.42) in a 1D MHD setting [vDZ06] and therefore discard it here. Thirdly, for a single component $\phi_p(\mathbf{q})$, variants (4.43) and (4.44) are completely equivalent. The latter is obtained by dividing the former by its maximum value and scalar multiplication of a monitor function has no effect on the mesh refinement. Hence, the new form (4.44) is consistent with (4.43) 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

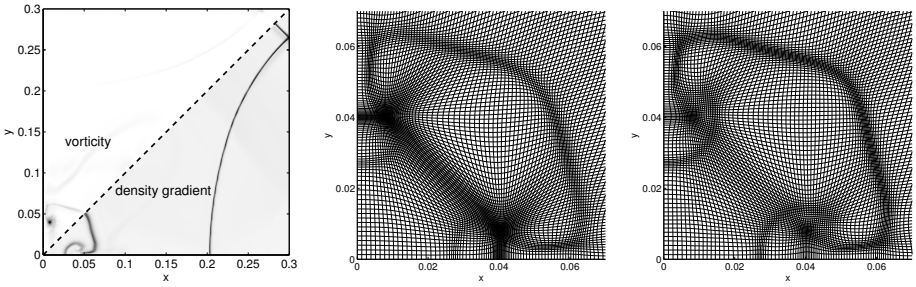

Figure 4.5 Solving the imbalance between monitor components in the HD22IMPDIAG example. Left: solution components at $t = 0.4$, upper triangular half shows the vorticity, lower half shows the norm of the density gradient. Middle: mesh detail using the unbalanced monitor function (4.39). Right: mesh detail using the balanced monitor function (4.45).

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.

4.4.3.2 *Proof of concept*

The necessity for replacing variant (4.43) is illustrated by the HD22IMPDIAG example problem (full specification in Section 4.5.1). We only include the density gradient and vorticity in the monitor summation:

$$\phi_{i,1}(\mathbf{q}) = \left| \frac{\partial \rho}{\partial \xi_i} \right|, \quad \phi_{i,2}(\mathbf{q}) = \|\nabla \times \mathbf{v}\|, \quad i \in \{1, 2\}. \tag{4.46}$$

The left diagram in Figure 4.5 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 Figure 4.5 shows the bottom left part of the domain. The result is a bad mesh for the unbalanced monitor variant (4.39): 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 (4.45) properly balances the two monitor components, resulting in a high quality adaptive mesh.

4.4.4 *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

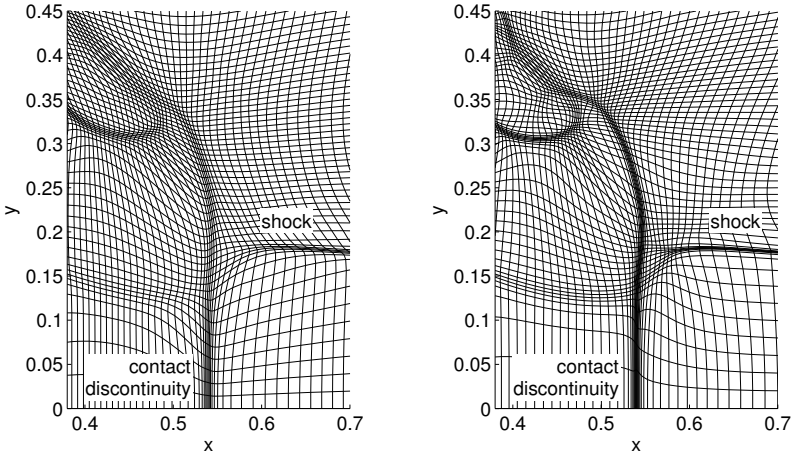

Figure 4.6 Nondirectional (left) and directional (right) mesh adaptation for the HD22CONF11 example problem. Close-up of mesh near $(0.53, 0.2)$.

multiple solution features from different directions meet, since there is less competition in attracting points.

In the mesh PDEs (4.28) the monitor matrix 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 (4.45), i.e., the second form below:

$$G_{\text{nondir}} = \begin{bmatrix} \omega & 0 \\ 0 & \omega \end{bmatrix}, \quad G_{\text{dir}} = \begin{bmatrix} \omega_1 & 0 \\ 0 & \omega_2 \end{bmatrix}. \tag{4.47}$$

The HD22CONF11 example problem (full specification in Section 4.5.2) illustrates the improved mesh for a directional monitor. Figure 4.6 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.

4.4.5 *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

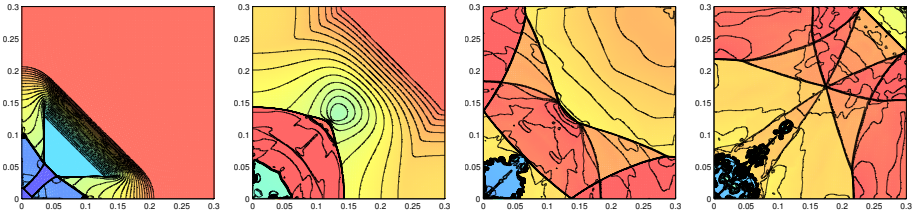

Figure 4.7 Time evolution of the HD22IMPDIAG problem. The colors and 50 contours show the density (range $[.125, 1.15]$). The snapshots are at $t = 0.05$, 0.175, 0.7 and 1.5.

widely-used Gaussian filter, typically 2 to 5 times:

$$\omega_{j,k}^{(i),\text{filter}} := \Big(4\omega_{j,k}^{(i)} + 2\cdot(\omega_{j-1,k}^{(i)} + \omega_{j+1,k}^{(i)} + \omega_{j,k-1}^{(i)} + \omega_{j,k+1}^{(i)}) + \omega_{j-1,k-1}^{(i)} + \omega_{j+1,k-1}^{(i)} + \omega_{j-1,k+1}^{(i)} + \omega_{j+1,k+1}^{(i)}\Big)/16 \quad (4.48)$$

for $i = 1$ and 2 independently.

4.5 EXPERIMENTS

4.5.1 HD22IMPDIAG: *a symmetric implosion with jets*

The implosion problem (HD22IMPDIAG) by Hui et al. [HLL99]—also extensively studied by Liska and Wendroff [LW03]—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.

Figure 4.7 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. [HVBDE03]. However, quantitative analysis is very complicated and rarely seen. We will study the sharpness of the jet front and its velocity in Section 4.5.1.3.

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

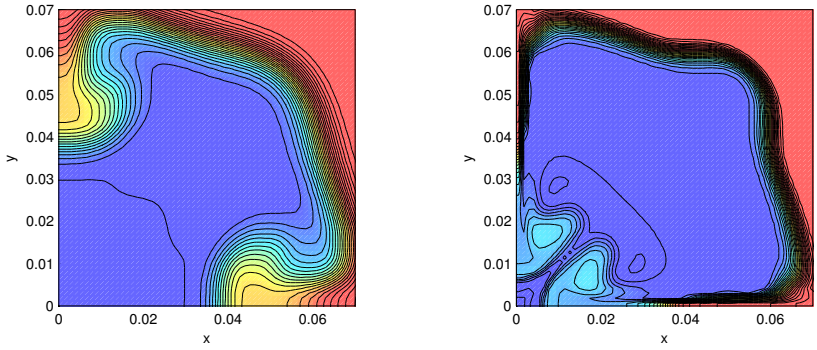

Figure 4.8 Non-adaptive local Lax–Friedrichs versus HLLC flux. Close-up of the HD22IMPDIAG problem on a uniform 250×250 mesh at $t = 0.45$. The colors and contours show the density. HLLC is much less diffusive.

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 4.3.2.2. Next, we will illustrate the monitor component balancing from Section 4.4.3. Finally, we will focus on details of the jets and the formation of instabilities.

4.5.1.1 *Local Lax–Friedrichs and HLLC*

The LLF and HLLC fluxes from Section 4.3.2.2 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.

Figure 4.8 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 4.5.1.3). 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.

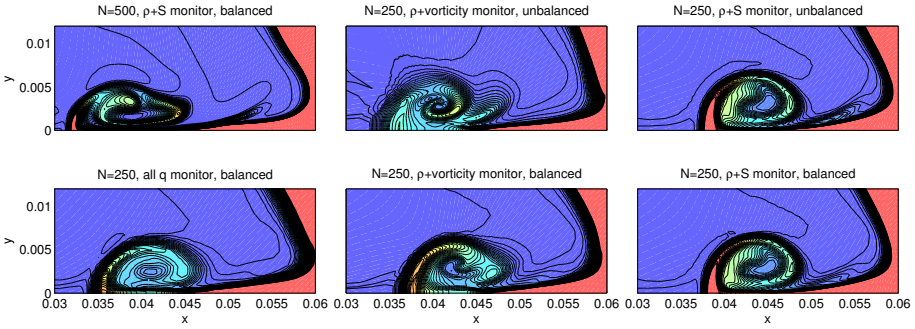

Figure 4.9 Detail of one of the jets in the HD22IMPDIAG problem. The top left diagram shows a high-resolution version ($N = 500$, adaptive). The other five show the results for adaptive meshes ($N = 250$) with several choices of monitor components, unbalanced versus unbalanced, i.e., Equation (4.43) versus (4.44). The colors and 50 contours depict density (range $[0.58, 1.1]$).

4.5.1.2 *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 (4.39) with the balanced variant (4.45) 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 Figure 4.9. The other five diagrams show the results of 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., (4.41) and (4.45). 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 Equation (4.46) in Section 4.4.3.2. However, there is only large vorticity within the back of the jet head. The mesh in the first diagram in Figure 4.10 shows the strongly localized refinement in the two points with high vorticity. The top middle diagram in Figure 4.9 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.

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

$$\phi_{i,1}(\mathbf{q}) = \left| \frac{\partial \rho}{\partial \xi_i} \right|, \quad \phi_{i,2}(\mathbf{q}) = \left| \frac{\partial S}{\partial \xi_i} \right|, \quad i \in \{1, 2\}. \tag{4.49}$$

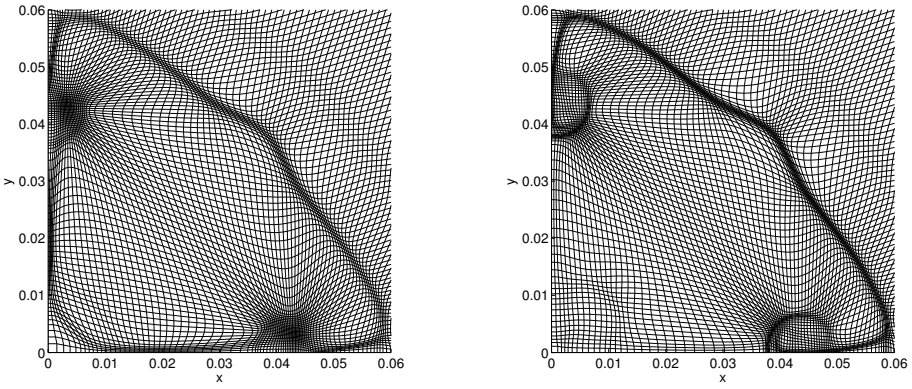

Figure 4.10 Monitor components: vorticity versus entropy gradients. Adaptive mesh details for the HD22IMPDIAG problem at $t = 0.15$. Left: 250×250, ρ+vorticity monitor, right: 250×250, $\rho + S$ monitor.

The 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.

Figure 4.10 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 Figure 4.5.

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.

4.5.1.3 *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. Figure 4.11 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 [SGT+08], with third-order spatial reconstruction and Roe's Riemann solver on a uniform 1000×1000 mesh. All other lines are by our

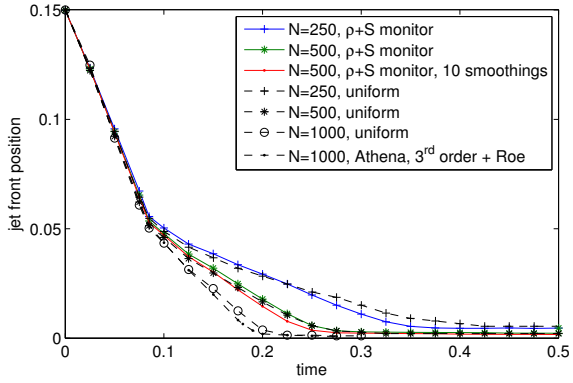

Figure 4.11 Position of the jet front in the HD22IMPDIAG problem for several numerical methods.

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 $t \approx 0.85$ 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. Figure 4.12 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

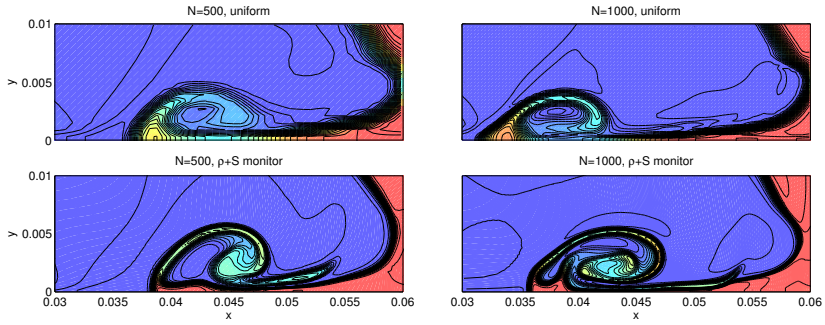

Figure 4.12 Detail of the horizontal jet in the HD22IMPDIAG problem at $t = 0.125$. The two uniform runs ($N = 500$ and $N = 1000$) are both expectedly more diffusive than the $N = 500$ adaptive run.

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 (4.26) 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.

4.5.1.4 *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. Figure 4.13 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 [NK02, vdHK07] for an approximately equal amount of mesh cells and identical finite volume solver (HLLC+Van Leer). The middle 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 right 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 ap-

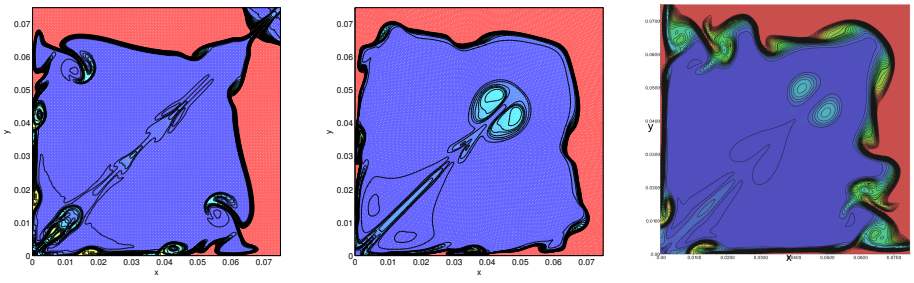

Figure 4.13 Formation of Kelvin–Helmholtz instabilities in the HD22IMPDIAG problem at $t = 0.5$. Left diagram: Athena with third-order Roe solver $N = 1000$, middle diagram: moving mesh with HLLC $N = 500$, right diagram: AMRVAC with HLLC $N = 100$ and 4 refinement levels (colors do not correspond exactly). For Athena, the jet has already moved beyond the top right corner.

proximately 300000 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 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 4.5.1.3. 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. Figure 4.14 shows two detailed views of the adaptive mesh at an early time $t = 0.5$, i.e., as in Figure 4.13, 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

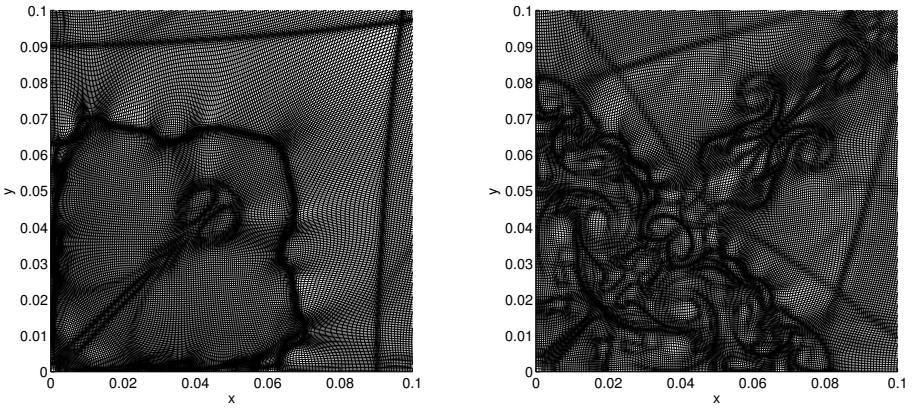

Figure 4.14 Moving mesh details ($N = 500$) showing Richtmyer–Meshkov instabilities in the HD22IMPDIAG problem. Left diagram: $t = 0.5$, right diagram: $t = 2.25$.

any change of parameters; both diagrams were taken from the same run.

4.5.2 HD22CONF11: *A Riemann problem with spirals*

The HD22CONF11 problem is one of the nineteen different Riemann problems classified by Lax and Liu [LL98]. 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 Figure 4.6 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 4.4.4 for further details.

4.5.3 HD22DMR: *Double Mach reflection*

The double Mach reflection problem (DMR) by Woodward and Colella [WC84] 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 $x = 1/6$,

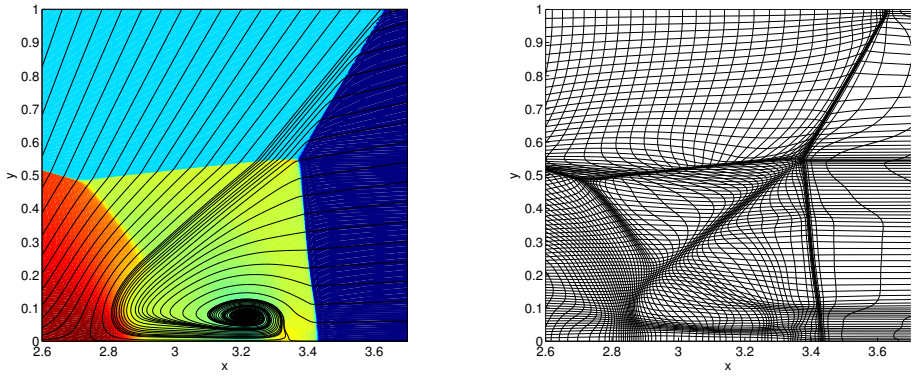

Figure 4.15 Left: Pressure detail for the double Mach reflection problem at t=0.25. The black lines are streamlines of the self-similar velocity field. Right: Adaptive mesh detail for the same problem. Total mesh size is 160×80 and the density-entropy monitor was used.

which gives the following initial conditions:

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

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. [HVBDE03] have analyzed the wall-jetting effect that occurs here. The left diagram in Figure 4.15 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 Figure 4.15 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.

4.6 CONCLUSIONS

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 4.4.3.2 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 4.5.3) is a good example of this. Entropy is a much more meaningful quantity. In both the DMR and the implosion problem (Section 4.5.1) 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

We wish to thank Rony Keppens from K.U. Leuven for pointing out the implosion problem (Section 4.5.1) and for several useful discussions on flow properties. Also, the results obtained during the visits to K.U.Leuven in 2007 and 2008 formed the main cause to most improvements discussed in this chapter. We also wish to thank Barry Koren from CWI for the useful discussions on approximate Riemann solvers and solution reconstruction.

A Moving Mesh Solver for 2D Magnetohydrodynamics

5

This chapter contains recent work on the application of the developed moving mesh method to two-dimensional ideal magnetohydrodynamics (MHD). In Chapter 3 we already found that the one-dimensional model is very rich in physical features, which makes proper and automatic solution adaptivity indispensable, even more so in two dimensions. Multidimensional MHD imposes an additional constraint on the solver: the magnetic field should be divergence-free. We use the well-known vector potential formulation to ensure this, which requires some additions to the original moving mesh finite volume solver from the previous chapter.

The results presented are a first investigation of the applicability of our mesh adaptation to new models. Another contribution of this chapter are the perfectly periodic and shifted periodic boundaries conditions, which is a definite improvement over the previous chapter where boundary points were kept restricted to the boundary.

5.1 INTRODUCTION

Multidimensional magnetohydrodynamics has been the subject of many research work. Firstly, most high-order numerical solvers developed for gas dynamics can also be extended to handle MHD problems. Tóth and Odstrčil [TO96] compare several methods on a range of problems.

Secondly, the experience of physicists with the MHD laws can lead to model-specific modifications to the numerical solvers. An example of this are the various model formulations that aim to keep the magnetic field divergence-free (see Tóth [Tót00] for an overview). In this chapter we derive the magnetic field from a vector potential, which ensures the divergence-free condition but also requires modifications to the original scheme.

Finally, the small spatial scale at which MHD flow features occur, has motivated the development of many adaptive (mainly h-refinement) solvers. The AMRVAC code by Keppens and coworkers [vdHK07, vdHKM08] has been mentioned before. It offers (parallel) block-AMR and finite volume solver with several approximate Riemann solvers. The divergence property is approximated by adding diffusive source terms to the original PDEs, such that the nonzero $\nabla \cdot \mathbf{B}$ is constantly damped.

The NIRVANA package by Ziegler [Zie98, Zie08] offers similar schemes as AMRVAC, but handles the divergence property exactly by a constrained transport approach on a staggered mesh.

Very few research has been done on moving mesh adaptivity for two-dimensional MHD problems. Zegeling [Zeg05] uses the inverse Winslow

method on resistive MHD problems. Tan [Tan07] uses the same approach and obtains comparable results. The incompressibility and lack of shocks makes this model fairly easy in terms of adaptivity. Han and Tang [HT07] apply the direct version of Winslow's method to ideal MHD. They capture shocks and other sharp flow features well.

This chapter is organized as follows. In Section 5.2 we briefly recall the MHD equations in two dimensions. In Section 5.3 we outline the vector potential formulation and its discretization in a finite volume method. Next, we summarize the modifications we made to the moving mesh solver to support MHD problems. We show some first results in Section 5.5, followed by conclusions in Section 5.6.

5.2 IDEAL MAGNETOHYDRODYNAMICS

The partial differential equations (PDEs) for ideal MHD have the general form of a nonlinear system of hyperbolic PDEs (4.4) as introduced in the previous chapter. The conservative solution variable is defined as $\mathbf{q} := [\rho, \rho\mathbf{v}, \mathbf{B}, E]^T$, containing density, momentum, magnetic field and total energy, respectively. The fully multidimensional conservation laws for ideal MHD were already presented in Equations (3.1)–(3.4). Here we restrict ourselves to two-dimensional domains, i.e., two columns in the flux tensor. The velocity and magnetic field may be three-dimensional, though, which is called '2.5D'. It is similar to the 1.5D and 1.75D models derived in Section 3.2.1. The flux tensor now becomes:

$$[\mathbf{f}(\mathbf{q}), \mathbf{g}(\mathbf{q})] = \mathcal{F}(\mathbf{q}) = \begin{bmatrix} \rho v_1 & \rho v_2 \\ \rho v_1^2 + p_{\text{tot}} - B_1^2 & \rho v_2 v_1 - B_2 B_1 \\ \rho v_1 v_2 - B_1 B_2 & \rho v_2^2 + p_{\text{tot}} - B_2^2 \\ \rho v_1 v_3 - B_1 B_3 & \rho v_2 v_3 - B_3 B_1 \\ 0 & v_2 B_1 - B_2 v_1 \\ v_1 B_2 - B_1 v_2 & 0 \\ v_1 B_3 - B_1 v_3 & v_2 B_3 - B_2 v_3 \\ (E + p_{\text{tot}}) v_1 - B_1 \mathbf{B} \cdot \mathbf{v} & (E + p_{\text{tot}}) v_2 - B_2 \mathbf{B} \cdot \mathbf{v} \end{bmatrix}. \quad (5.1)$$

The eigenstructure for the first flux vector $\mathbf{f}$ was already given in Section 3.2.2. It is similar for $\mathbf{g}$, with all scalar v_1 and B_1 terms replaced by v_2 and B_2.

The two-dimensional moving mesh finite volume solver from the previous chapter can directly solve ideal MHD problems, but an important physical property is then ignored. The build-up of magnetic 'charge' (or: magnetic monopoles) is prohibited, so the divergence of the magnetic field should be zero: $\nabla \cdot \mathbf{B} = 0$, given that $\nabla \cdot \mathbf{B}|_{t=0} = 0$. The finite volume method does guarantee conservation of all conservative variables, but something similar does *not* hold for this divergence-free property of the magnetic field. In the next section, we will describe the vector potential formulation that remedies this problem.

5.3 THE VECTOR POTENTIAL FORMULATION

Several approaches exist to ensure or approximate the divergence-free property of the magnetic field. We refer to Tóth's review paper [Tót00] for a description of seven possible approaches and their accuracy for several test problems.

The rotation of any vector field has zero divergence ($\nabla \cdot \nabla \times \mathbf{v} \equiv 0$), so it is attractive to define the magnetic field as the curl of a potential field. The divergence-free property is then analytically satisfied and for proper discretizations of the derivatives, the property is also maintained up to machine precision. An additional benefit in two dimensions is that the vector field only needs to have a z-component:

$$\boldsymbol{\mathcal{A}} := [0, 0, \mathcal{A}]^T, \tag{5.2}$$

$$\mathbf{B} := \nabla \times \boldsymbol{\mathcal{A}} = [\mathcal{A}_y, -\mathcal{A}_x, 0]^T. \tag{5.3}$$

The magnetic field can now be obtained directly from an appropriate finite difference discretization of $\mathcal{A}_y$ and $\mathcal{A}_x$, so the PDEs for $\mathbf{B}$ are discarded from the original system. Instead, one new equation describes the evolution of the potential $\mathcal{A}$. Inserting (5.3) into the induction equation (3.3) yields:

$$\frac{\partial \mathcal{A}}{\partial t} = v_1 B_2 - v_2 B_1 \tag{5.4}$$

$$= -\Big(v_1 \frac{\partial \mathcal{A}}{\partial x} + v_2 \frac{\partial \mathcal{A}}{\partial y}\Big). \tag{5.5}$$

5.3.1 *Discretization of the vector potential*

The magnetic field components are no longer pure solution variables in $\mathbf{q}$, but derived quantities from (central discretizations of) derivatives of the potential. Some form of mesh staggering will result from this. Han and Tang [HT07] place the vector potential at cell corners: $\mathcal{A}_{j,k}$, we avoid staggering and place them at cell centers instead: $\mathcal{A}_{j+\frac{1}{2},k+\frac{1}{2}}$. This is no problem, as the $\mathbf{B}$-values are only needed at edge centers to evaluate the fluxes.

The discrete divergence at cell centers has to be written into computational derivatives:

$$\begin{aligned}
\nabla \cdot \mathbf{B}\big|_{j+\frac{1}{2},k+\frac{1}{2}} &= \Big[\frac{\partial}{\partial x}\mathcal{A}_y + \frac{\partial}{\partial y}(-\mathcal{A}_x)\Big]\Big|_{j+\frac{1}{2},k+\frac{1}{2}} \\
&= \Big[\Big(\mathcal{A}_\xi \frac{-x_\eta}{J} + \mathcal{A}_\eta \frac{x_\xi}{J}\Big)_\xi \cdot \frac{y_\eta}{J} + \Big(\mathcal{A}_\xi \frac{-x_\eta}{J} + \mathcal{A}_\eta \frac{x_\xi}{J}\Big)_\eta \cdot \frac{-y_\xi}{J} \\
&\quad - \Big(\mathcal{A}_\xi \frac{y_\eta}{J} + \mathcal{A}_\eta \frac{-y_\xi}{J}\Big)_\xi \cdot \frac{-x_\eta}{J} - \Big(\mathcal{A}_\xi \frac{y_\eta}{J} + \mathcal{A}_\eta \frac{-y_\xi}{J}\Big)_\eta \cdot \frac{x_\xi}{J}\Big]\Big|_{j+\frac{1}{2},k+\frac{1}{2}}.
\end{aligned} \tag{5.6}$$

For all derivatives, central differences are used. This means that the $\mathcal{A}_\xi$ and $\mathcal{A}_\eta$ between cell corners end up at edge centers $(j+\frac{1}{2}, k)$ and $(j, k+\frac{1}{2})$, respectively. Figure 5.1 compares the cell-centered and cell-cornered variants. In

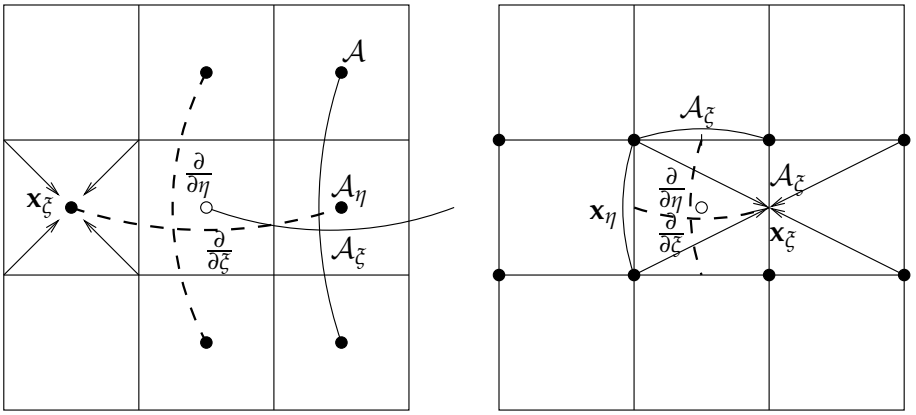

Figure 5.1 Central differences to evaluate $\nabla\cdot\mathbf{B}$ (5.6). Left: vector potential at cell centers. Right: vector potential at cell corners. In both cases, $\nabla\cdot\mathbf{B}$ vanishes at cell centers. A set of arrows averages two central differences. Arches indicate a normal central difference. The dashed lines indicate the outermost $\partial/\partial\xi$ and $\partial/\partial\eta$-derivatives in (5.6).

our cell-centered approach the discretized divergence uses 9 cell centers for $\mathcal{A}$ derivatives and 16 cell corners for $\mathbf{x}$ derivatives. Both satisfy the divergence-free property at cell centers.

In the actual implementation, $\nabla\cdot\mathbf{B}$ does not need to be evaluated. Only the magnetic field at each cell edge is needed for the flux evaluation. This simply follows from a central difference of the potential across each edge. For example, for a vertical edge $(j, k+\frac{1}{2})$:

$$\begin{aligned} B_1\big|_{j,k+\frac{1}{2}} &= \mathcal{A}_y\big|_{j,k+\frac{1}{2}} = \left[\mathcal{A}_\xi\frac{-x_\eta}{J}+\mathcal{A}_\eta\frac{x_\xi}{J}\right]\Big|_{j,k+\frac{1}{2}} \\ &\approx \frac{\mathcal{A}_{j+\frac{1}{2},k+\frac{1}{2}}-\mathcal{A}_{j-\frac{1}{2},k+\frac{1}{2}}}{\Delta\xi}\cdot\frac{-x_\eta}{J}\Big|_{j,k+\frac{1}{2}} \\ &\quad+\frac{\mathcal{A}_{j-\frac{1}{2},k+\frac{3}{2}}-\mathcal{A}_{j-\frac{1}{2},k-\frac{1}{2}}+\mathcal{A}_{j+\frac{1}{2},k+\frac{3}{2}}-\mathcal{A}_{j+\frac{1}{2},k-\frac{1}{2}}}{4\Delta\eta}\cdot\frac{x_\xi}{J}\Big|_{j,k+\frac{1}{2}}, \end{aligned} \tag{5.7}$$

where the coordinate derivatives are approximated in a similar central way:

$$\mathbf{x}_\xi\big|_{j,k+\frac{1}{2}} \approx \frac{\mathbf{x}_{j+1,k}-\mathbf{x}_{j-1,k}+\mathbf{x}_{j+1,k+1}-\mathbf{x}_{j-1,k+1}}{4\Delta\xi}, \tag{5.8}$$

$$\mathbf{x}_\eta\big|_{j,k+\frac{1}{2}} \approx \frac{\mathbf{x}_{j,k+1}-\mathbf{x}_{j,k}}{4\Delta\eta}, \tag{5.9}$$

and similar for B_2 and at the other three edges. The resulting stencil for each edge contains six cells. Had we used a cell-cornered representation of the potential, this stencil size would have been just the same.

5.3.2 *Combination with finite volumes*

The vector potential $\mathcal{A}$ is an auxiliary variable that will yield magnetic field values at cell edges, which are needed for the flux evaluations in a finite volume approach. Besides, a new PDE for $\mathcal{A}$ is included.

The magnetic field variables B_1 and B_2 are now inferred from the vector potential, so they are removed from the conservative solution variables: $\mathbf{q} := [\rho, \rho\mathbf{v}, B_3, E]^T$. The next section discusses the evaluation of the B_3 component, for now we just consider a two-dimensional field without B_3 component. Before, the field $\mathbf{B}$ was transferred to the edges by a limited upwind reconstruction (Section 4.3.2.1). Now it follows from the central difference approximation (5.7).

We first take a basic approach for the time integration of the potential PDE (5.5), namely the second-order Heun scheme and a first-order upwind spatial discretization of $\mathcal{A}_x$ and $\mathcal{A}_y$. A more accurate alternative is to use the ODE form (5.4) and average the right hand side $|\mathbf{v} \times \mathbf{B}|$ between t_n and t_{n+1} (see, e.g., [Tót00] or [HT07]). In this case, the magnetic field PDE has to remain in the original system and its finite volume solution for $\mathbf{B}^{n+1}$ then acts as a predictor value. Algorithm 3 sketches a single time step by finite volumes and the above potential differencing. It is performed on a nonuniform mesh that remains fixed during the time step. The mesh movement takes place just before each finite volume step and is almost identical to the approach in Algorithm 2. Only the interpolation of $\mathcal{A}$ is new. Section 5.4 contains details on this mesh adaptation step.

Algorithm 3 Finite volume step for vector potential for of ideal MHD on a nonuniform (frozen) mesh.

1: $\mathbf{Q}^*_{j+\frac{1}{2},k+\frac{1}{2}} \leftarrow \mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t\, L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^n, \mathbf{B}^n)$ (2^{nd} order MUSCL).
2: $\mathcal{A}^*_{j+\frac{1}{2},k+\frac{1}{2}} \leftarrow \mathcal{A}^n_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t\, L^{\mathcal{A}}_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^n, \mathcal{A}^n)$ (upwind).
3: $\mathbf{B}^*_{j,k+\frac{1}{2}} \leftarrow L^B_{j,k+\frac{1}{2}}(\mathcal{A}^*)$; $\mathbf{B}^*_{j+\frac{1}{2},k} \leftarrow L^B_{j+\frac{1}{2},k}(\mathcal{A}^*)$ by (5.7).

Use the predictor values for the actual time step:

4: $\mathbf{Q}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} \leftarrow \mathbf{Q}^n_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t \left(L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^n, \mathbf{B}^n) + L_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^*, \mathbf{B}^*)\right)/2.$
5: $\mathcal{A}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} \leftarrow \mathcal{A}^n_{j+\frac{1}{2},k+\frac{1}{2}} + \Delta t \left(L^{\mathcal{A}}_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^n, \mathcal{A}^n) + L^{\mathcal{A}}_{j+\frac{1}{2},k+\frac{1}{2}}(\mathbf{Q}^*, \mathcal{A}^*)\right)/2.$
6: $\mathbf{B}^{n+1}_{j,k+\frac{1}{2}} \leftarrow L^B_{j,k+\frac{1}{2}}(\mathcal{A}^{n+1})$; $\mathbf{B}^{n+1}_{j+\frac{1}{2},k} \leftarrow L^B_{j+\frac{1}{2},k}(\mathcal{A}^{n+1})$.

5.3.3 *Vector potential and 2.5D models*

The unique specification of the magnetic field through the vector potential as in (5.3) is only possible due to the two-dimensional nature of the model. In a purely 2D model, the B_3 component is absent, such that the vector potential $\boldsymbol{\mathcal{A}}$ can have zeros as its first two components. In a 2.5D model, there *is* a third magnetic field component. We do not want to give up the simple form of the vector potential, though. The trick is to derive B_1 and B_2 as usual from

(5.3). Next, B_3 is simply included as a conservative solution variable in the original hyperbolic PDEs (remember that the PDEs for B_1 and B_2 have been discarded). The divergence-free property is still maintained:

$$\begin{aligned}\nabla \cdot \mathbf{B} &= \frac{\partial B_1}{\partial x} + \frac{\partial B_2}{\partial y} + \frac{\partial B_3}{\partial z} \\ &= \frac{\partial}{\partial x}\mathcal{A}_y + \frac{\partial}{\partial y}(-\mathcal{A}_x) + 0 = 0,\end{aligned}$$

since all $\partial/\partial z$-derivatives are zero by definition on two-dimensional domains.

5.4 MESH MOVEMENT FOR MAGNETOHYDRODYNAMICS

The new form of the MHD PDEs requires an additional interpolation step of the magnetic potential $\mathcal{A}$ after mesh movement. Besides, additional monitor components than the ones from the previous chapter may be useful for good solution adaptivity. Finally, a perfect representation of periodic domains should greatly improve the robustness and accuracy of the solver for periodic MHD problems.

5.4.1 *Interpolation of the vector potential*

When the mesh points have been moved, all discrete solution values are interpolated onto the new mesh. For the conservative variables in $\mathbf{q}$ the conservative interpolation (4.32) is used again. The vector potential does not need to be conserved. We follow the second-order interpolation approach by Tang et al. [TTZ03], who interpret the interpolation as the solution to a Hamilton–Jacobi-type equation:

$$\tilde{\mathcal{A}}_{j+\frac{1}{2},k+\frac{1}{2}} \approx \mathcal{A}_{j+\frac{1}{2},k+\frac{1}{2}} + \nabla\mathcal{A}|_{j+\frac{1}{2},k+\frac{1}{2}} \cdot (\tilde{\mathbf{x}}_{j+\frac{1}{2},k+\frac{1}{2}} - \mathbf{x}_{j+\frac{1}{2},k+\frac{1}{2}}) \tag{5.10}$$

$$= \mathcal{A}_{j+\frac{1}{2},k+\frac{1}{2}} - c^{\xi}_{j+\frac{1}{2},k+\frac{1}{2}}\mathcal{A}_{\xi}|_{j+\frac{1}{2},k+\frac{1}{2}} - c^{\eta}_{j+\frac{1}{2},k+\frac{1}{2}}\mathcal{A}_{\eta}|_{j+\frac{1}{2},k+\frac{1}{2}}, \tag{5.11}$$

where the velocities c^{ξ} and c^{η} are the mesh displacement vector projected onto the normal directions:

$$c^{\xi} := (\tilde{\mathbf{x}} - \mathbf{x}) \cdot \nabla\xi, \quad \text{and} \quad c^{\eta} := (\tilde{\mathbf{x}} - \mathbf{x}) \cdot \nabla\eta. \tag{5.12}$$

The normal vectors $\nabla\xi$ and $\nabla\eta$ were previously illustrated in Figure 2.4. All derivatives are approximated on cell centers by central differences.

5.4.2 *Solution monitoring in magnetohydrodynamics*

We again employ our new balanced monitor function (4.45) from the previous chapter. The density and entropy gradients remain equally useful as monitor components. Besides, the vorticity of the magnetic field could prove useful. The Orszag–Tang problem is know for the formation of current sheets, where the large currents $\mathbf{j} := \nabla \times \mathbf{B}$ can result in small 'islands' that change the

topology of magnetic field lines. Therefore it may be useful to include a monitor component

$$\phi_{\text{current}} := \|\nabla \times \mathbf{B}\|,$$

such that the mesh refines in these areas.

5.4.3 *Exactly periodic domains*

The boundary treatment of the adaptive mesh in the previous chapter partially fixes boundary points by sliding them on the straight domain edges ($\frac{\partial \mathbf{x}}{\partial n} = 0$ on $\partial\Omega$, i.e., one of the coordinates is fixed). Periodic domains can only be handled exactly if we release this constraint. Discrete solution values on and beyond the boundaries can easily be made periodic, but if the adaptive mesh cells on the facing sides are *different*, then the solution would actually *not* be periodic.

Doubly periodic domains

A doubly periodic domain has periodicity on both the left and right edges and the bottom and top edges. The domain is discretized as before by $N_x \times N_y$ mesh points (see Equation (4.7)), Figure 3.2 shows the numbering of ghost points.

A band-aid solution for rectangular doubly periodic domains would be to make the x-coordinates of mesh points on the bottom and top edges equal to each other, e.g., by averaging $x_{j,0}$ and x_{j,N_y} and similar for the y-coordinates on the left and right edges. This still keeps the straight domain edges, though. Ideally, the edges should not be noticed at all, not by the discrete solution, nor by the mesh movement. After all, a periodic domain is generally just a trick to run simulations on an infinite domain.

We create an exactly periodic domain by deriving monitor values in ghost cells from periodicity, e.g.,

$$\omega^{(i)}_{j,k} = \omega^{(i)}_{j,N_y+k}, \quad \text{for } k = -2, -1 \ \ (\text{and } j = 0, \ldots, N_x, \ i = 1, 2), \tag{5.13}$$

for the bottom boundary and similar for the other three boundaries. The periodic solution values are evaluated in the same way. Notice that this leaves the four 2×2 ghost corners; these are handled at the very end. This allows for monitor filtering (4.48) and mesh point movement (4.30) for both the four corner points and all edge points, i.e., $j = 0, \ldots, N_x$ instead of $j = 1, \ldots, N_x - 1$ and similar for k. The ghost points outside the domain follow in a straightforward way, e.g.,

$$\begin{aligned} x_{j,k} &= x_{j,N_y+k}, \\ y_{j,k} &= y_{j,0} - (y_{j,N_y} - y_{j,N_y+k}), \quad \text{for } k = -2, -1 \ \ (\text{and } j = 0, \ldots, N_x), \end{aligned} \tag{5.14}$$

for the bottom boundary.

Figure 5.2 shows an adaptive mesh for a two-dimensional advection problem. The initial solution is a step-like function on the unit square, with value 1 on an ω-shaped part at the center of the domain and 0 elsewhere. The

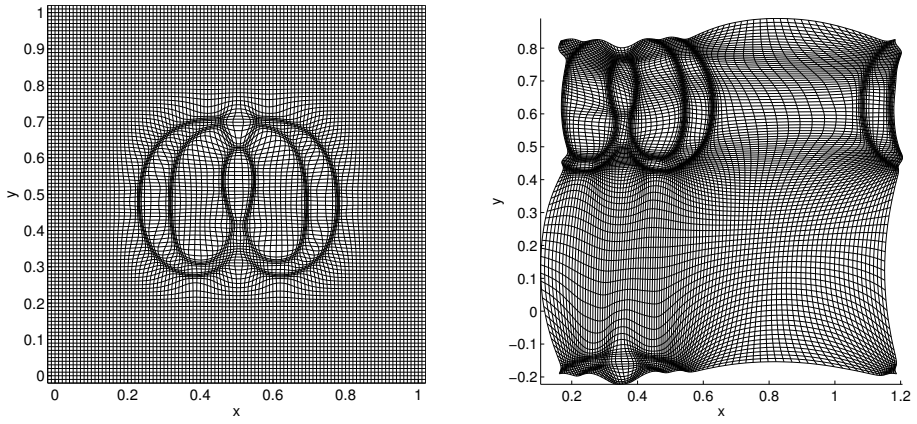

Figure 5.2 Doubly periodic boundaries for an advection problem with constant velocity $[-1, 1]$. Mesh details at $t = 0$ and $t = 0.15$.

advection velocity is $[-1, 1]$. At $t = 0.15$ parts of the solution profile have left the top left corner of the domain and re-entered at the opposite sides. The mesh has properly aligned with the solution curves, in the same way at the boundary as in the interior of the domain.

Now that the corner and edge points are completely free to move, the mesh may actually move away from the original domain. In Figure 5.2 the original domain $[0, 1] \times [0, 1]$ has shifted to approximately $[0.2, 1.2] \times [-0.2, 0.8]$. This shifted mesh can always be brought back to the original domain by taking all coordinates modulo the original domain size, without any change to the solution.

Both the moving mesh and the discrete solution values are perfectly periodic (up to machine precision), so solution features that move across a boundary are not perturbed in any way by this. Clearly, this is desirable, especially for simulations over long times with multiple passes.

Shifted periodic boundaries

A shifted periodic boundary condition states that two facing boundaries are periodic after a given tangential shift. It is useful for a certain class of problems. One-dimensional problems, e.g., the shock tube problems in Section 3.4, can be made quasi-two-dimensional by keeping the initial solution constant in the y-direction. This is a first test for validating two-dimensional codes. A second possible test rotates this initial solution, such that the mesh adaptation and PDE solution becomes fully two-dimensional. Without rotation (or $\theta = \pi/2$), the conditions on the top and bottom boundary can either be periodic or homogeneously Neumann. When the initial shock profile is *not* exactly vertical (parallel with the y-axis), the only accurate boundary condition is a shifted periodic boundary.

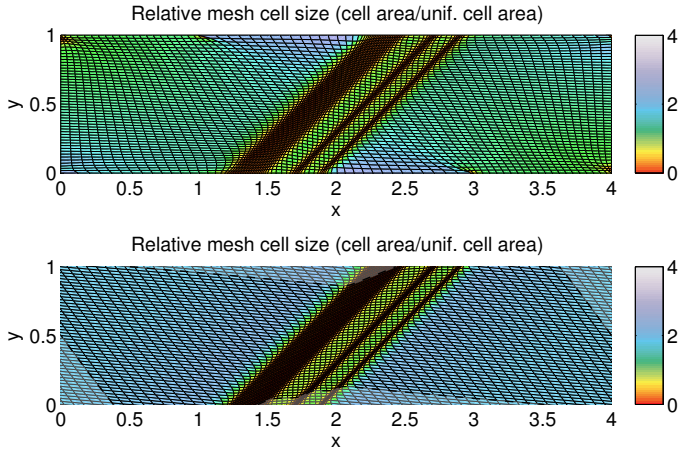

Figure 5.3 Handling the shifted periodic boundary condition for the rotated Sod shock tube. Top diagram: mesh restricted to domain boundary. Bottom diagram: fully periodic mesh with a shift between the bottom and top boundary.

A shock tube problem on a domain $[a_1, b_1] \times [a_2, b_2]$ that has the initial shock line at an angle $\theta \neq \pi/2$ with the horizontal has a horizontal shift of

$$\Delta s := (b_2 - a_2)/\tan(\theta).$$

The only requirement to the initial domain discretization is that Δs should be a multiple of the cell size, such that the index shift

$$\Delta j := \Delta s / N_x \tag{5.15}$$

is a whole number. The original periodic boundary condition (5.14) now changes to:

$$\begin{aligned} x_{j,k} &= x_{j+\Delta j, N_x+k} - \Delta s, \\ y_{j,k} &= y_{j,0} - (y_{j+\Delta j, N_y} - y_{j+\Delta j, N_y+k}), \quad \text{for } k = -2, -1 \ \ (\text{and } j = 0, \ldots, N_x), \end{aligned} \tag{5.16}$$

for the bottom boundary. The top boundary is treated similarly, but note that the solution at the left and right boundaries is generally homogeneously Neumann or Dirichlet. Whenever the shifted index lies beyond the domain boundaries ($j + \Delta j < 0$ or $j + \Delta j > N_x$), the point is mapped back periodically to the other side of the domain.

Motivation

We demonstrate the effect of this special boundary condition with a quasi-2D variant of Sod's classical hydrodynamical shock tube problem [Sod78] rotated by an angle $\theta = \pi/4$:

$$[\rho, v_1, v_2, p] = \begin{cases} [\quad 1,\ 0,\ 0,\quad 1], & \text{if } x - 1.5 \leq y, \\ [0.125,\ 0,\ 0,\ 0.1], & \text{otherwise.} \end{cases} \tag{5.17}$$

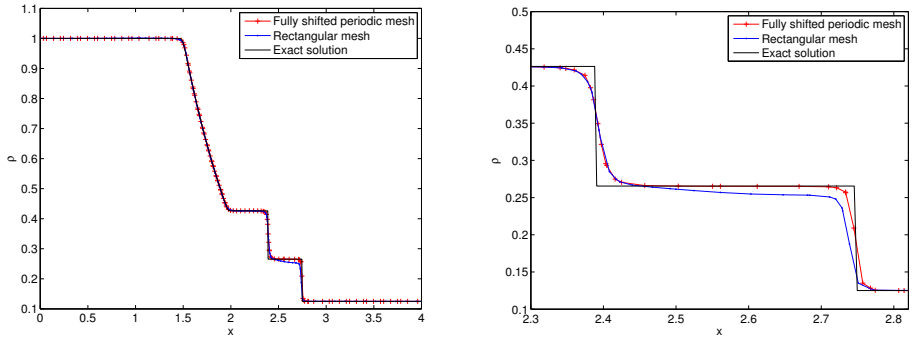

Figure 5.4 Solution profile for the rotated Sod shock tube problem along a slice $y = 0.5$. The fully periodic mesh yields a more accurate solution, particularly after the shock, as shown in the right diagram.

On a domain $[0, 4] \times [0, 1]$ the initial discontinuity goes through the center point $(2, 0.5)$.

First, we restrict the boundary points to the initial rectangular boundary. The shifted periodic boundaries at the top and bottom are approximated by averaging the displacement of each pair of associated boundary points:

$$\tilde{x}_{j,k} := x_{j,k}^{[\nu]} + \left((x_{j,k}^{[\nu+1]} - x_{j,k}^{[\nu]}) + (x_{j+\Delta j, N_y+k}^{[\nu+1]} - x_{j+\Delta j, N_y+k}^{[\nu]}) \right)/2 \text{ for } k = -2, -1, \quad (5.18)$$

at the bottom boundary and in a similar fashion at the top boundary. The first diagram in Figure 5.3 shows the mesh at $t = 0.16$. The colors indicate the area of each cell relative to the initial uniform cell size. The mesh has been refined along the shock lines and the rarefaction fan (RF). However, where the shocks and CD meet the top and bottom edges, mesh cells become more stretched and the resulting errors propagate into the domain. Moreover, the points $[3, 0]$ and $[1, 1]$ remain necessarily fixed (they map to the corners $[4, 1]$ and $[0, 0]$, respectively), which makes them more and more distorted over time.

The second diagram in Figure 5.3 shows the fully shifted periodic mesh. Each pair of sides matches exactly and the solution adaptivity is not affected by any boundary effects. This also yields a better solution. We show the solution along a slice $y = 0.5$ in Figure 5.4. The exact solution along this line is the original one-dimensional solution with the x-coordinates scaled by $\tilde{x} := (x - 2)/\cos(\theta) + 2$. For the restricted mesh, the solution just after the shock at $[x, y] \approx [2.6, 0.5]$ suffers from the boundary perturbation and is less accurate than the solution on the fully periodic mesh. This difference becomes even larger over time.

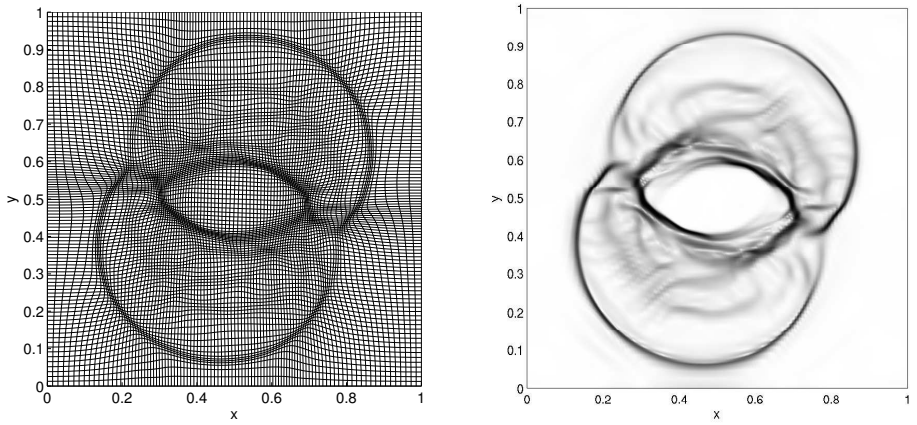

Figure 5.5 The rotor problem at $t = 0.15$. Left: adaptive 100×100 mesh. Right: Schlieren plot of thermal pressure.

5.5 NUMERICAL EXPERIMENTS

5.5.1 *Rotor problem*

This problem was originally proposed by Balsara and Spicer [BS99] and was used by others (see, e.g., [Tót00] and [HT07]) to test MHD solvers. It concerns a rotating disk inside a fluid at rest on the unit square domain with homogeneous Neumann boundary conditions. The disk with radius $r_0 = 0.1$ is placed in the center and has a constant angular velocity $v_0 = 2$. A thin annulus between the disk and the surrounding fluid has outer radius $r_1 = 0.115$ and transfers the variables linearly from the disc to the fluid at rest. The magnetic field and thermal pressure are constant everywhere:

$$[\rho, v_1, v_2, B_1, B_2, p] = \begin{cases} [\quad 10, \quad v_1^{\mathrm{d}}, \quad v_2^{\mathrm{d}}, 5/\sqrt{4\pi}, 0, 1], & \text{if } r \le r_0, \\ [\quad 1, \quad 0, \quad 0, 5/\sqrt{4\pi}, 0, 1], & \text{if } r > r_1, \\ [1+9f, \; f v_1^{\mathrm{d}}, \; f v_2^{\mathrm{d}}, 5/\sqrt{4\pi}, 0, 1], & \text{if } r_0 < r \le r_1, \end{cases} \tag{5.19}$$

where the radius is $r := \|\mathbf{x} - [0.5, 0.5]\|$, the disk's velocity is given by $v_1^{\mathrm{d}} := -v_0(y-0.5)/r_0$, $v_2^{\mathrm{d}} := v_0(x-0.5)/r_0$ and $f := (r_1 - r)/(r_1 - r_0)$ is a linear function from 1 to 0 in the annulus.

Figure 5.5 shows the mesh and a Schlieren plot of the thermal pressure at $t = 0.15$ for a 100×100 mesh. The monitor function uses the kinetic energy as single component. The mesh is well adapted and smooth. The pressure plot shows some perturbations that are unphysical (cf. [HT07, Fig. 7]). Increasing the mesh points, or lowering the CFL limit does not remedy these errors. The first-order scheme for the $\mathcal{A}$ evolution needs to be improved, which will be part of our future research.

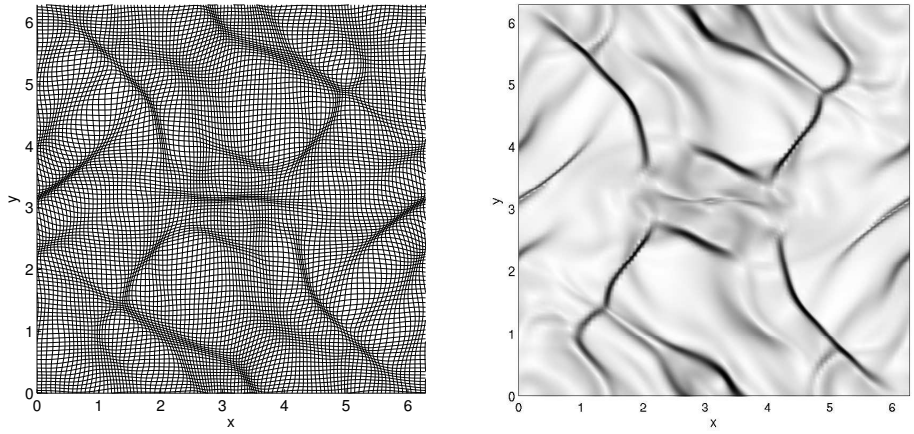

Figure 5.6 The Orszag–Tang problem at $t = 3.1$. Left: adaptive 100×100 mesh. Right: Schlieren plot of density.

5.5.2 *Orszag–Tang vortex problem*

This problem was originally proposed by Orszag and Tang [OT79] for incompressible flows. Picone and Dahlquist [PD91] also used this problem to study the formation of MHD turbulence in the compressible case. The velocity field and magnetic field consist of periodic sine waves with two different wave lengths. The domain is the square $[0, 2\pi] \times [0, 2\pi]$ and is doubly periodic. The initial solution is:

$$\begin{aligned} \rho = 25/9, \quad v_1 = -\sin(y), \quad v_2 = \sin(x), \\ B_1 = -\sin(y), \quad B_2 = \sin(2x), \quad p = 5/3. \end{aligned} \tag{5.20}$$

Thanks to the periodicity of both the solution and the domain, the initial value for the vector potential is straightforward:

$$A = \frac{1}{2}\cos(2x) + \cos(y). \tag{5.21}$$

The solution forms current sheets (with large $\mathbf{j} := \nabla \times \mathbf{B}$) from which shock waves depart. The resulting ongoing interactions produce a complicated pattern. In exact ideal MHD the topology of magnetic field lines should not change. In practice, however, small perturbations in the flow can cause this to happen. In simulations, numerical errors can do this as well; the current sheets may break down and form a series of small 'islands' of closed magnetic field lines. A high order solver and a very fine effective resolution is needed to capture this behavior. Van der Holst et al. [vdHKM08] achieve this for a *relativistic* version of this problem with their h-refinement code, which produces an effective resolution of approximately 2500×2500 cells. The non-relativistic case that we consider here may only form these islands at later

times. The results that we obtain at $t = 3.1$ are in accordance with known solutions (see, e.g., [Tót00]).

Figure 5.6 shows the mesh and a Schlieren plot of the density for a 100×100 mesh at $t = 3.1$. The monitor function includes density gradients, the current strength $\|\mathbf{j}\|$ and B_1 gradients. We included the latter, since it turned out to detect the current sheets at least as good as the current strength itself. Mesh points have focused near shocks and current sheets and the mesh cells vary smoothly.

5.6 CONCLUSIONS

The moving mesh method from the previous chapter turned out to need no adjustments to the point movement nor to the balanced monitor function, as expected. Only the new vector potential $\mathcal{A}$ required an additional interpolation step. The potential formulation of the MHD equations requires some extra bookkeeping of auxiliary variables, and the most sensitive point seems to be the first-order approximation of the PDE for $\mathcal{A}$. Han and Tang [HT07] report good results with a second-order variant, so this will be part of our future research. The solutions that we obtained on relatively coarse 100×100 meshes already does show all main flow features. Once the first-order parts have been made higher-order and more robust, we expect that the strength of mesh adaptation can be further increased. This would yield the effective resolution needed for capturing the islands that may form along the current sheets at later times.

The perfectly periodic boundary conditions turn out to be a valuable addition to our solver. The mesh points on the boundary are completely free to move now, such that flow features that move near or across the boundaries are not perturbed in any sense, which used to be the case for restricted edge points.

Software

A

The two-dimensional solvers in Chapters 4 and 5 have been implemented in Fortran 95. This yields a major performance gain compared to the one-dimensional MATLAB implementation from Chapter 3. The mesh adaptivity and finite volume solver are in separate program modules. The different physical models (advection, hydrodynamics, magnetohydrodynamics) are concisely implemented in separate program modules. The user can define problem-specific initial conditions and boundary conditions, if necessary. Finally, a run is initialized by a parameter file, in which the user specifies when and what data to output, what type of slope limiting is used, the CFL number, etc., and adaptation settings such as monitor function components. The software has a suspend and resume functionality, such that runs can be interrupted and resumed at any later time.

A.1 MISCELLANEOUS ENHANCEMENTS

The essence of a numerical method may lie in its core algorithm, but in the end it is the output by the software that counts. Minor practical choices during the implementation can seriously affect the final outcome. In this section we highlight some program details that proved important for good results.

A.1.1 *Smooth initial discontinuities*

Most problems discussed in this dissertation have some discontinuities in them. The discretization of the initial solution requires extra attention, since the discontinuity seldomly aligns with the—initially uniform—mesh. A too crude discretization can lead to an oscillatory solution. Therefore we perform anti-aliasing on the discretized discontinuity.

The Double Mach reflection (DMR) problem (Section 4.5.3) by Woodward and Colella [WC84] serves as an example. It prescribes an initial shock that starts in $[1/6, 0]$ and stands at an angle of 60 degrees with the bottom boundary. In general, a discontinuity is parameterized by a point $[x_s, y_s]$ on the bottom boundary, an angle θ with the horizontal and the pre- and post-shock solution states $\mathbf{q}_{\text{pre}}$ and $\mathbf{q}_{\text{post}}$. The solution in a point $[x, y]$ is now prescribed by[1]:

$$\mathbf{q}([x, y], t_0) = \begin{cases} \mathbf{q}_{\text{post}} & \text{if } y - y_s > (x - x_s) \tan\theta, \\ \mathbf{q}_{\text{pre}} & \text{otherwise.} \end{cases} \tag{A.1}$$

[1] We assume a rightward motion of the shock. If this is not the case the two solution states should be swapped. We also assume $0 \leq \theta \leq \pi/2$; for larger angles the solution states should again be swapped.

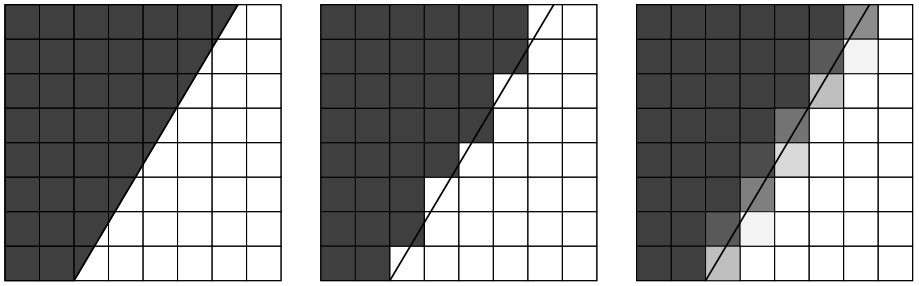

Figure A.1 Anti-aliasing of a discretized discontinuity. Left: Prescribed solution. Middle: 'hit-or-miss' discretization. Right: Anti-aliased discretization.

Figure A.1 shows part of the DMR problem with two possible discretizations. The middle picture shows a straightforward discretization: if the cell center lies ahead of the shock the cell's solution value is set to the pre-shock state, otherwise to the post-shock state. The right picture shows the result of a weighted average between the two states. The weight is equal to the relative area of a cell that is in a certain state.

$$\mathbf{q}|_{A_{j,k}} = r\,\mathbf{q}_{\text{post}} + (1-r)\mathbf{q}_{\text{pre}}, \text{ where}$$
$$r = \frac{|\text{part of } A_{j,k} \text{ in post state}|}{|A_{j,k}|}.$$

The third picture in Figure A.1 may suggest an overly smoothed solution, but remember that it is used for the first mesh adaptation, after which the solution is reinitialized using the exact solution (A.1). These two steps are repeated several times, during which the mesh will align more and more with the angled discontinuity.

We now generalize this approach for a set of n_s shock lines and $n_s + 1$ solution states. Figure A.2 shows this schematically for three shocks. Note that due to the adaptivity of all mesh cells the evaluation of intersections is nontrivial. A shock line is specified by a point $[x_s, y_s]$ on the line and its slope. We assume that on the rectangular domain shocks are sorted from left to right, in other words: all points on one shock line lie right of (or on) the preceding line. This forbids crossing lines and in particular horizontal lines. The following algorithm could be further generalized to support these cases too, but it would become more costly than necessary for our applications.

A.1.1.1 *Algorithm*

For an adaptive mesh, no assumptions can be made about the relative position of two neighboring cells. Cell $(j{+}1, k)$ could lie above or even *left of* cell (j, k) in a heavily rotated mesh. We therefore cannot benefit from already evaluated cell solutions. The solution is initialized on all mesh cells individually by EVALMULTISHOCKSOL. The following procedures use some specified globally available variables for notational convenience.

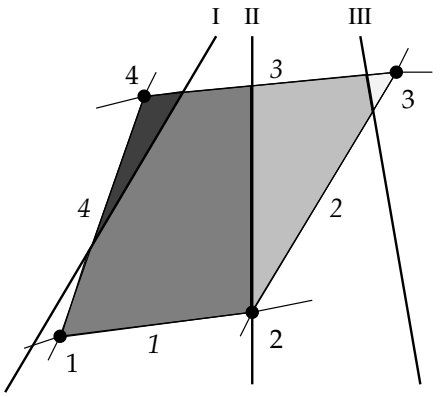

Figure A.2 Multiple shocks on a mesh cell. The Roman numbers denote shocks, the plain Arabic numbers denote cell corners and the italic numbers denote cell edges.

Figure A.3 lists the main procedure EVALMULTISHOCKSOL. The outer loop on line 2 determines the first shock that lies right of or on the mesh cell. In case of an intersection the loop on line 13 takes over to check whether any of the following shocks intersect the current cell as well. For each intersecting shock the relative area between itself and the preceding shock on the current cell is determined. This defines the relative contribution of the shock's left state to the total solution state. Note that for cells that are not intersected, no expensive areas are computed.

The procedure GETSHOCKSTATE in Figure A.4 determines whether a mesh cell lies completely left or right of a given shock or is intersected by it. The four corners are considered by the loop on line 2. We use the following notation for corner indices:

$$j, k, (c) := j + \delta_{c\in\{2,3\}}, k + \delta_{c\in\{3,4\}} \quad c \in \{1, 2, 3, 4\},$$

where δ_P evaluates to 1 if the predicate P is true, and 0 otherwise. Figure A.2 also shows the corner and edge numbering. For each corner, an imaginary horizontal line is drawn through it. The x-coordinate of the point where this line and the shock line intersect, is used to compare the position of the corner relative to the shock line. If the state at all corner points is equal the cell's state is known, since the mesh cell is a polygon.

The procedure GETRELAREA in Figure A.5 forms the final part of the algorithm; it is used by EVALMULTISHOCKSOL. It first uses GETQUADLINEINTERSECTIONS in Figure A.6 to obtain the two intersection points of the shock line and the current mesh cell. It does not consider the case where the line and the cell are disjoint, as it will never be called by EVALMULTISHOCKSOL in such a case. The intersection splits the cell into two parts. The area of one part is evaluated and one of its points is used to determine its position relative to the shock line. There are three cases to handle, depending on how the shock line crosses the cell.

EVALMULTISHOCKSOL(j, k)
Computes average solution value in cell (j, k) *given* n_s *shocks.*
x *:: mesh coordinates,* q *:: solution values,* qstates *::* $n_s + 1$ *solution states.*

```
1  i_s ← 0
2  while i_s < n_s
3     i_s ← i_s + 1      ▷ Consider next shock
4     state ← GETSHOCKSTATE(j, k, i_s)
5     switch state
6     case BEFORE:
7        q(j, k, :) ← qstates(i_s, :)
8        return          ▷ Shocks are sorted, the rest will not intersect.
9     case AFTER:
10       cycle           ▷ Beyond shock i_s, try next one.
11    case INTERSECT:
12       sumarea ← 0
13       while state = INTERSECT ∧ i_s ≤ n_s
14          relarea ← GETRELAREA(j, k, i_s)
15          q(j, k, :) ← q(j, k, :) + (relarea − sumarea) · qstates(i_s, :)
16          sumarea ← sumarea + relarea
17          i_s ← i_s + 1
18          state ← GETSHOCKSTATE(j, k, i_s)
         ▷ Handle the 'right' state of last intersecting shock
19       q(j, k, :) ← q(j, k, :) + (1 − sumarea) · qstates(i_s, :)
20       return          ▷ Shocks are sorted, the rest will not intersect.
21 q(j, k, :) ← qstates(n_s + 1, :) ▷ Point lies beyond last shock: state n_s + 1.
```

Figure A.3 EVALMULTISHOCKSOL computes solution state, averaged across shocks where necessary.

1. The shock line hits the cell in just one corner (line 2). The part's area is set to zero, but the opposite corner is used to check at which side of the shock line it lies, since the intersection point itself lies *on* the shock. The area will be swapped later, on line 34.

2. The shock line cuts the cell through two opposite sides (line 9). The cell is split into two quadrangles, the one that contains corner (j, k) is used to determine the part's area. If the area is nonzero this corner point is also used to determine the part's position relative to the shock, otherwise the opposite point is used (and the area swapped accordingly, see line 27.) If the selected part lies right of the shock, the other of the two parts should have been chosen in the first place, so the area is swapped on line 32.

GETSHOCKSTATE(j, k, i_s)
Computes whether cell (j, k) *lies left (*BEFORE*) or right (*AFTER*) of shock* i_s *or* INTERSECTS *it.*
x *:: mesh coordinates,* shockspecs *:: shock parameters.*

```
1  [x_s, y_s, slope] ← shockspecs_{i_s}
2  for c in [1,2,3,4] ▷ For all corner points...
       ▷ Compute x_(c) such that [x_(c), y_{j,k,(c)}] lies on shock line.
3      if slope = ∞
4          x_(c) ← x_s
5      else
6          x_(c) ← x_s + (y_{j,k,(c)} − y_s)/ slope
7      if x_{j,k,(c)} ≤ x_(c)
8          state ← BEFORE
9      else
10         state ← AFTER
11     if c > 1 ∧ state ≠ prevstate
12         state ← INTERSECT
13         break
14     else
15         prevstate ← state
16 return state
```

Figure A.4 GETSHOCKSTATE determines the position of a cell relative to a shock.

3. The shock line cuts the cell through two adjoining sides (line 16). The triangle that is formed in the corner is used to determine the part's area and is further handled similar to the preceding case.

The procedures GETQUADRANGLEAREA and GETTRIANGLEAREA are straightforward and omitted here.

The intersection point(s) of a shock line and a mesh cell are evaluated by the procedure GETQUADLINEINTERSECTIONS in Figure A.6. It simply considers all four edges and uses GETINTERSECTIONPOINT to get the intersection of each edge and the shock line. The latter procedure is straightforward and is not shown here. When two found intersection points are equal, the latter is discarded and the remaining edges are considered (line 11). This handles the case where the line goes through one of the cell's corners and intersects an opposite edge as well, see for example the second shock in Figure A.2. Note that in actual implementations, round off errors will have to be handled. For example when equality of two points in a cell corner is checked.

In the Double Mach Reflection problem, the top boundary condition is often the exact shock solution. Since this moves in time, the shock specification additionally includes a horizontal speed. It is only necessary to change x_s into $x_s + \text{hspeed} \cdot t$ on lines 4 and 6 in Figure A.4.

GETRELAREA(j, k, i_s)
Computes the relative area of cell (j, k) *that lies left of shock* i_s.
x, y *:: mesh coordinates,* q *:: solution values,* shockspecs *::* n_s *shock parameters.*

```
1  [p1, p2, e1, e2] ← GETLINEQUADINTERSECTIONS(j, k, i_s)
2  if p2 = NOINTERSECTION ∨ p1 = p2) ▷ 'Hit' in just one corner.
3     partarea ← 0
4     switch e1
5     case (1, 4):
6        oppospoint ← (x, y)_{j+1,k+1}
7     case (2, 3):
8        oppospoint ← (x, y)_{j,k}
9  elseif |e2 − e1| = 2 ▷ Two opposing sides: two quadrangles.
10    if e2 = 3 ▷ 'Vertical' intersection.
11       partarea ← GETQUADRANGLEAREA((x, y)_{j,k}, p1, p2, (x, y)_{j,k+1})
12    else ▷ 'Horizontal' intersection.
13       partarea ← GETQUADRANGLEAREA((x, y)_{j,k}, (x, y)_{j+1,k}, p1, p2)
14    partpoint ← (x, y)_{j,k}
15    oppospoint ← (x, y)_{j+1,k+1}
16 else ▷ One triangle.
17    if e1 = 1
18       if e2 = 2      ▷ Bottom right
19          partpoint ← (x, y)_{j+1,k}
20       else           ▷ Bottom left
21          partpoint ← (x, y)_{j,k}
22    elseif e2 = 4     ▷ Top left
23       partpoint ← (x, y)_{j,k+1}
24    else              ▷ Top right
25       partpoint ← (x, y)_{j+1,k+1}
26    partarea ← GETTRIANGLEAREA(p1, partpoint, p2)
27 if partarea = 0
28    partpoint ← oppospoint
29    swaparea ← TRUE
30 else
31    swaparea ← FALSE
32 if partpoint lies right of shock i_s
33    partarea ← cellarea − partarea
34 if swaparea
35    partarea ← cellarea − partarea
36 return partarea / cellarea
```

Figure A.5 GETRELAREA computes the fraction of the left shock state in a mesh cell.

GETQUADLINEINTERSECTIONS(j, k, i_s)
Computes the intersection points (two at most) of shock line i_s and the edges of mesh cell (j, k). Also returns the edge numbers on which the intersections occurred.

```
1  i_p ← 1
2  for e in [1, 2, 3, 4]
3     p_{i_p} ← GETINTERSECTIONPOINT(j, k, i_s, e)
4     if p_{i_p} = NOINTERSECTION
5        cycle        ▷ Try next edge.
6     else
7        e_{i_p} ← e      ▷ Remember p-th edge number.
8        if i_p = 2
9           if p_1 = p_2
10             p_2 ← NOINTERSECTION
11             cycle     ▷ Discard duplicate point in corner, try next edge.
12          else
13             break     ▷ The two intersections have been found.
14       else
15          i_p ← i_p + 1  ▷ Find next intersection.
16 return [p_1, p_2, e_1, e_2]
```

Figure A.6 GETQUADLINEINTERSECTIONS finds intersections of a shock line and a mesh cell.

A.1.2 *Prevention of collapsing mesh cells*

The mesh generator that we employ guarantees a strictly positive Jacobian, i.e., the mesh map is nonsingular, as proved in Section 2.3.2.8 on page 29. This only holds for the exact map, though. The iterative solution of the nonlinear system (4.30) *could* yield collapsing mesh cells if a mesh point is moved beyond its opposite edges.

In practice we rarely witnessed this. The only risky situation turned out to be the imposed time-dependent Dirichlet top boundary in the Double Mach Reflection Problem (Section 4.5.3). Again, the smooth monitor function prevents rapidly changing mesh point locations. Still, one might consider slowing down the Gauss–Seidel iteration by under-relaxation. This slows down the mesh convergence at all times, and is therefore not a good choice. A cheap trick is to perform small time steps in early—often most critical—stages of the simulation by taking the CFL-limit several factors smaller during, say, the first ten time steps. Finally, we devise a check that maintains convexity of all mesh cells, which is a stronger form of 'non-collapsing'. In '99.9%' of our experiments we disabled this check for higher efficiency.

One iteration step of mesh movement gives the new position $\mathbf{x}_n$ for each old mesh point $\mathbf{x}_o$. We now limit $\mathbf{x}_n$ to $\bar{\mathbf{x}}_n$ using the following considerations. Figure A.7 shows how a mesh point $\mathbf{x}_o$ is surrounded by four cells and how it will move into one of them. This is depicted by vector $\mathbf{v}_2 := \mathbf{x}_n - \mathbf{x}_o$. For

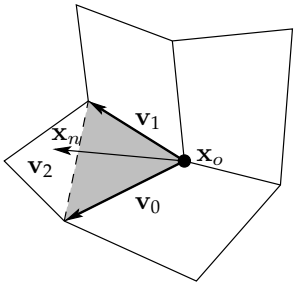

Figure A.7 Protection against mesh collapse by limiting the mesh velocity vector.

one of these neighboring cells we express $\mathbf{v}_2$ in terms of the two edge vectors $\mathbf{v}_0$ and $\mathbf{v}_1$ from $\mathbf{x}_0$:

$$\mathbf{v}_2 = a_0\mathbf{v}_0 + a_1\mathbf{v}_1. \tag{A.2}$$

We solve this equation per coordinate, which gives:

$$a_0 = \frac{v_{2x}v_{1y} - v_{2y}v_{1x}}{v_{0x}v_{1y} - v_{0y}v_{1x}}, \tag{A.3}$$

$$a_1 = \frac{v_{2x}v_{0y} - v_{2y}v_{0x}}{v_{1x}v_{0y} - v_{1y}v_{0x}}. \tag{A.4}$$

If any of the two coefficients is negative, the mesh point has moved into one of the other three neighboring cells. The resulting mesh cell remains convex if the new mesh point $\mathbf{x}_n$ has not moved beyond the diagonal between the tips of vectors $\mathbf{v}_0$ and $\mathbf{v}_1$, see the dashed line in Figure A.7. In terms of the coefficients this is equivalent to:

$$a_0 + a_1 \leq 1. \tag{A.5}$$

The actual mesh point could be limited to stay 'behind' the diagonal:

$$\bar{\mathbf{x}}_n := \mathbf{x}_o + \min\left(1, \frac{\mu}{a_0 + a_1}\right)\mathbf{v}_2, \tag{A.6}$$

where $0 < \mu \leq 1$ determines how close to the diagonal a point may come.

Bibliography

[AF90] David C. Arney and Joseph E. Flaherty. An Adaptive Mesh-Moving and Local Refinement Method for Time-Dependent Partial Differential Equations. *ACM Trans. Math. Softw.*, 16(1):48–71, 1990. (Cited on page 57.)

[And90] Dale A. Anderson. Grid cell volume control with an adaptive grid generator. *Appl. Math. Comput.*, 35(3):209–217, 1990. (Cited on page 31.)

[Ant70] R.A. Anthes. Numerical experiments with a two-dimensional horizontal variable grid. *Monthly Weather Review*, 98(11):810–822, 1970. (Cited on page 1.)

[Art75] A.M. Arthurs. *Calculus of variations*. Routledge and Kegan Paul Boston, 1975. (Cited on page 24.)

[AT05] Boris N. Azarenok and Tao Tang. Second-order Godunov-type scheme for reactive flow calculations on moving meshes. *J. Comput. Phys.*, 206(1):48–80, 2005. (Cited on page 84.)

[Bai94] M.J. Baines. *Moving finite elements*. Oxford University Press, 1994. (Cited on page 35.)

[BAM05] 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. (Cited on page 90.)

[BBSW94] John Bell, Marsha Berger, Jeff Saltzman, and Mike Welcome. Three-Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws. *SIAM J. Sci. Comput.*, 15(1):127–138, 1994. (Cited on page 55.)

[BC89] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, 1989. (Cited on page 54.)

[BHJ05] M.J. Baines, M.E. Hubbard, and P.K. Jimack. A moving mesh finite element algorithm for the adaptive solution of time-dependent partial differential equations with moving boundaries. *Appl. Numer. Math.*, 54(3-4):450–469, 2005. Selected papers from the 16th Chemnitz Finite Element Symposium 2003. (Cited on page 41.)

[BHJJ06] M.J. Baines, M.E. Hubbard, P.K. Jimack, and A.C. Jones. Scale-invariant moving finite elements for nonlinear partial differential equations in two dimensions. *Appl. Numer. Math.*, 56(2):230–252, 2006. (Cited on pages 41 and 42.)

[BHJM09] M.J. Baines, M.E. Hubbard, P.K. Jimack, and R. Mahmood. A Moving-Mesh Finite Element Method and its Application to the Numerical Solution of Phase-Change Problems. *Commun. Comput. Phys.*, 6(3):595–624, 2009. (Cited on pages 3 and 42.)

[BHR96] C.J. Budd, W. Huang, and R.D. Russell. Moving mesh methods for problems with blow-up. *SIAM J. Sci. Comput.*, 17(2):305–327, 1996. (Cited on page 47.)

[BHR09] Chris J. Budd, Weizhang Huang, and Robert D. Russell. Adaptivity with moving grids. *Acta Numerica*, 18:111–241, 2009. (Cited on page 13.)

[BM00] 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. (Cited on pages 6, 47, 60, 63, 68, 84, and 97.)

[BMR06] G. Beckett, J.A. Mackenzie, and M.L. Robertson. An r-adaptive finite element method for the solution of the two-dimensional phase-field equations. *Commun. Comput. Phys*, 1(5):805–826, 2006. (Cited on pages 6, 8, and 28.)

[BMRS01] G. Beckett, J.A. Mackenzie, A. Ramage, and D.M. Sloan. On The Numerical Solution of One-Dimensional PDEs Using Adaptive Methods Based on Equidistribution. *J. Comput. Phys.*, 167:372–392, March 2001. (Cited on pages 54 and 60.)

[Bra93] J. U. Brackbill. An Adaptive Grid with Directional Control. *J. Comput. Phys.*, 108(1):38–50, 1993. (Cited on pages 25, 26, 27, 49, and 84.)

[BS82] J.U. Brackbill and J.S. Saltzman. Adaptive Zoning for Singular Problems in Two Dimensions. *J. Comput. Phys.*, 46:342–368, 1982. (Cited on pages 24, 84, and 94.)

[BS99] Dinshaw S. Balsara and Daniel S. Spicer. A Staggered Mesh Algorithm Using High Order Godunov Fluxes to Ensure Solenoidal Magnetic Fields in Magnetohydrodynamic Simulations. *J. Comput. Phys.*, 149(2):270–292, 1999. (Cited on page 123.)

[BV89] J.G. Blom and J.G. Verwer. On the Use of the Arclength and Curvature Monitor in a Moving-Grid Method which is Based on the Method of Lines. Technical Report NM-N8902, CWI, Amsterdam, 1989. (Cited on page 46.)

[BW88] M. Brio and C.C. Wu. An upwind differencing scheme for the equations of ideal magnetohydrodynamics. *J. Comput. Phys.*, 75:400–422, April 1988. (Cited on pages 61 and 69.)

[BW06] C.J. Budd and J.F. Williams. Parabolic Monge-Ampère methods for blow-up problems in several spatial dimensions. *Journal of Physics A*, 39(19):5425–5444, 2006. (Cited on pages 3, 6, and 32.)

[Car97] Graham F. Carey. *Computational Grids: Generation, Adaptation, and Solution Strategies*. Taylor & Francis, 1997. (Cited on page 12.)

[Cas91] José E. Castillo, editor. *Mathematical Aspects of Numerical Grid Generation*, volume 8 of *Frontiers in applied mathematics*. SIAM, 1991. (Cited on page 12.)

[CH62] Richard Courant and David Hilbert. *Methods of Mathematical Physics*, volume II. Wiley-Interscience, 1962. (Cited on page 33.)

[CH01] Hector D. Ceniceros and Thomas Y. Hou. An efficient dynamically adaptive mesh for potentially singular solutions. *J. Comput. Phys.*, 172(2):609–639, 2001. (Cited on pages 6, 23, 28, 93, and 97.)

[CHR99] Weiming Cao, Weizhang Huang, and Robert D. Russell. A study of monitor functions for two-dimensional adaptive mesh generation. *SIAM J. Sci. Comput.*, 20(6):1978–1994, 1999. (Cited on pages 50, 60, and 93.)

[CHR02] Weiming Cao, Weizhang Huang, and Robert D. Russell. A moving mesh method based on the geometric conservation law. *SIAM J. Sci. Comput.*, 24(1):118–142 (electronic), 2002. (Cited on pages 3 and 40.)

[CHR03] Weiming Cao, Weizhang Huang, and Robert D. Russell. Approaches for generating moving adaptive meshes: location versus velocity. *Appl. Numer. Math.*, 47:121–138, November 2003. (Cited on pages 13 and 14.)

[CHS96] 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. (Cited on pages 27, 29, 30, and 96.)

[CJ96] P.J. Capon and P.K. Jimack. On the Adaptive Finite Element Solution of Partial Differential Equations Using h-r-Refinement. Technical Report 96.13, University of Leeds, school of computer studies, April 1996. (Cited on page 57.)

[CJ04] Maenghyo Cho and Seongki Jun. r-Adaptive mesh generation for shell finite element analysis. *J. Comput. Phys.*, 199(1):291–316, 2004. (Cited on page 39.)

[CL06] L. Chacón and G. Lapenta. A fully implicit, nonlinear adaptive grid strategy. *J. Comput. Phys.*, 212(2):703–717, 2006. (Cited on page 26.)

[CM98a] Neil N. Carlson and Keith Miller. Design and Application of a Gradient-Weighted Moving Finite Element Code I: in One Dimension. *SIAM Journal on Scientific Computing*, 19(3):728–765, 1998. (Cited on page 35.)

[CM98b] Neil N. Carlson and Keith Miller. Design and Application of a Gradient-Weighted Moving Finite Element Code II: in Two Dimensions. *SIAM J. Sci. Comput.*, 19(3):766–798, 1998. (Cited on page 35.)

[CSR88] J. Castillo, S. Steinberg, and P.J. Roache. Parameter estimation in variational grid generation. *Appl. Math. Comput.*, 28(2):155–177, 1988. (Cited on page 24.)

[dB74] C. de Boor. Good approximation by splines with variable knots. II. In G.A. Watson, editor, *Conference on the Numerical Solution of Differential Equations*, volume 363 of *Lecture Notes in Mathematics*, pages 12–20. Springer Berlin, 1974. (Cited on pages 2 and 20.)

[DCF+08] G.L. Delzanno, L. Chacón, J.M. Finn, Y. Chung, and G. Lapenta. An optimal robust equidistribution method for two-dimensional grid adaptation based on Monge-Kantorovich optimization. *J. Comput. Phys.*, 227(23):9841–9864, December 2008. (Cited on pages 3, 6, 32, 33, and 39.)

[DLT08] Yana Di, Ruo Li, and Tao 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. (Cited on pages 6, 26, and 84.)

[DLTZ05] Yana Di, Ruo Li, Tao Tang, and Pingwen Zhang. Moving mesh finite element methods for the incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 26(3):1036–1056, 2005. (Cited on pages 6, 8, and 84.)

[DLTZ07] Yana Di, Ruo Li, Tao Tang, and Pingwen Zhang. Level Set Calculations for Incompressible Two-Phase Flows on a Dynamically Adaptive Grid. *J.Sc.Comp.*, 31:75–98, May 2007. (Cited on pages 6 and 47.)

[DM90] B. Dacorogna and J. Moser. On a partial differential equation involving the Jacobian determinant. *Ann. Inst. H. Poincare Anal. Non Lineaire*, 7(1):1–26, 1990. (Cited on page 37.)

[Dvi91] Arkady S. Dvinsky. Adaptive grid generation from harmonic maps on Riemannian manifolds. *J. Comput. Phys.*, 95(2):450–476, 1991. (Cited on pages 2, 22, 25, and 47.)

[FGG01] Charbel Farhat, Philippe Geuzaine, and Céline Grandmont. The Discrete Geometric Conservation Law and the Nonlinear Stability of ALE Schemes for the Solution of Flow Problems on Moving Grids. *J. Comput. Phys.*, 174:669–694, December 2001. (Cited on pages 44 and 45.)

[GE88] Antonios E. Giannakopoulos and Antonie J. Engel. Directional control in grid generation. *J. Comput. Phys.*, 74(2):422–439, 1988. (Cited on page 49.)

[GGF03] Philippe Geuzaine, Céline Grandmont, and Charbel Farhat. Design and analysis of ALE schemes with provable second-order time-accuracy for inviscid and viscous flow simulations. *J. Comput. Phys.*, 191:206–227, October 2003. (Cited on page 44.)

[GLK05] 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. (Cited on pages 50 and 84.)

[Hag94] R. Hagmeijer. Grid Adaption Based on Modified Anisotropic Diffusion Equations Formulated in the Parametric Domain. *J. Comput. Phys.*, 115(1):169–183, 1994. (Cited on page 29.)

[Ham75] R.S. Hamilton. *Harmonic Maps of Manifolds with Boundary*, volume 471 of *Lecture Notes in Mathematics*. Springer-Verlag, 1975. (Cited on page 26.)

[HCL+08] C.Y. Hsieh, H. Chen, T.H. Lin, H.Y. Hsiao, M.Y. Chu, G. Liao, and H. Zhong. On the development of a new non-rigid image registration using deformation based grid generation. In *Proceedings of SPIE Medical Imaging*, volume 6914, 2008. (Cited on page 39.)

[HDZ05] Glen A. Hansen, Rod W. Douglass, and Andrew Zardecki. *Mesh enhancement. Selected Elliptic Methods, Foundations and Applications*. Imperial College Press, 2005. (Cited on page 13.)

[HH83] Ami Harten and James M. Hyman. Self adjusting grid methods for one-dimensional hyperbolic conservation laws. *J. Comput. Phys.*, 50(2):235–269, 1983. (Cited on page 34.)

[HLL83] Amiram Harten, Peter D. Lax, and Bram Van Leer. On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws. *SIAM Review*, 25(1):35–61, 1983. (Cited on page 90.)

[HLL99] 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. (Cited on page 102.)

[HR97a] W. Huang and R.D. Russell. A high dimensional moving mesh strategy. *Appl. Numer. Math.*, 26(1):63–76, 1997. (Cited on page 27.)

[HR97b] W. Huang and R.D. Russell. Analysis of moving mesh partial differential equations with spatial smoothing. *SIAM J. Numer. Anal.*, 34:1106–1126, 1997. (Cited on page 53.)

[HR99] Weizhang Huang and Robert D. Russell. Moving Mesh Strategy Based on a Gradient Flow Equation for Two-Dimensional Problems. *SIAM J. Sci. Comput.*, 20(3):998–1015, 1999. (Cited on pages 27, 28, 44, and 59.)

[HRR94] Weizhang Huang, Yuhe Ren, and Robert D. Russell. Moving mesh partial differential equations (MMPDES) based on the equidistribution principle. *SIAM J. Numer. Anal.*, 31(3):709–730, 1994. (Cited on pages 27, 43, and 59.)

[HS02] Bjarte Hægland and Bård Skaflestad. A Survey of Some Methods for Moving Grid and Grid Adaptation. Technical Report 2, Norges Teknisk-Naturvitenskapelige Universitet, 2002. (Cited on page 13.)

[HS03] Weizhang Huang and Weiwei Sun. Variational mesh adaptation. II. Error estimates and monitor functions. *J. Comput. Phys.*, 184(2):619–648, 2003. (Cited on pages 5 and 52.)

[HT07] Jianqiang Han and Huazhong Tang. An adaptive moving mesh method for two-dimensional ideal magnetohydrodynamics. *J. Comput. Phys.*, 220:791–812, Jan 2007. (Cited on pages 6, 29, 84, 95, 114, 115, 117, 123, and 125.)

[Hua01a] Weizhang Huang. Practical aspects of formulation and solution of moving mesh partial differential equations. *J. Comput. Phys.*, 171(2):753–775, 2001. (Cited on pages 27, 47, 60, 68, 84, and 98.)

[Hua01b] Weizhang Huang. Variational Mesh Adaptation: Isotropy and Equidistribution. *J. Comput. Phys.*, 174:903–924, December 2001. (Cited on pages 5 and 51.)

[Hua05] Weizhang Huang. Measuring mesh qualities and application to variational mesh adaptation. *SIAM J. Sci. Comput.*, 26(5):1643–1666, 2005. (Cited on page 60.)

[Hua06] W. Huang. Mathematical principles of anisotropic mesh adaptation. *Commun. Comput. Phys.*, 1:276–310, 2006. (Cited on pages 6, 51, 52, and 84.)

[Hua07] Weizhang Huang. Anisotropic Mesh Adaptation and Movement. In T. Tang and J. Xu, editors, *Adaptive Computations: Theory and Algorithms*. Science Press, Beijing, 2007. Workshop on Adaptive mesh methods, Beijing University, 2005. (Cited on page 13.)

[Hub99] M. E. Hubbard. Multidimensional Slope Limiters for MUSCL-Type Finite Volume Schemes on Unstructured Grids. *J. Comput. Phys.*, 155(1):54–74, October 1999. (Cited on page 90.)

[HVBDE03] 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. (Cited on pages 102 and 110.)

[HZZ02] Weizhang Huang, Li Zheng, and Xiaoyong Zhan. Adaptive moving mesh methods for simulating one-dimensional groundwater problems with sharp moving fronts. *Int. J. Numer. Meth. Eng.*, 54(11):1579–1603, May 2002. (Cited on page 54.)

[Iva04] S.A. Ivanenko. *Selected chapters on grid generation and applications*. Dorodnicyn computing centre of the Russian Academy of Sciences, Moscow, 2004. (ed. Boris N. Azarenok). (Cited on page 13.)

[Kan42] L. V. Kantorovich. On the Translocation of Masses. *Dokl. Akad. Nauk SSSR*, 37:227–229, 1942. (Cited on page 32.)

[Kep04] Rony Keppens. Nonlinear Magnetohydrodynamics: Numerical Concepts. *Fusion Science and Technology*, 45(2T):107–114, March 2004. (Cited on pages 59, 62, and 73.)

[KMvdHC08] R. Keppens, Z. Meliani, B. van der Holst, and F. Casse. Extragalactic jets with helical magnetic fields: relativistic MHD simulations. *Astron. & Astrophys.*, 486:663–678, 2008. (Cited on page 7.)

[KNTG03] R. Keppens, M. Nool, G. Tóth, and J.P. Goedbloed. Adaptive Mesh Refinement for conservative systems: multi-dimensional efficiency evaluation. *Comput. Phys. Commun.*, 153:317–339, July 2003. (Cited on page 72.)

[KS93] P.M. Knupp and S. Steinberg. *The fundamentals of grid generation*. CRC press, 1993. (Cited on pages 12, 23, and 24.)

[LA92] G. Liao and D. Anderson. A new approach to grid generation. *Applicable Analysis*, 44(3):285–298, 1992. (Cited on pages 27 and 36.)

[Lan98] Jens Lang. Adaptive FEM for reaction-diffusion equations. *Appl. Numer. Math.*, 26(1-2):105–116, 1998. (Cited on page 8.)

[LC06] G. Lapenta and L. Chacón. Cost-effectiveness of fully implicit moving mesh adaptation: A practical investigation in 1D. *J. Comput. Phys.*, 219(1):86–103, 2006. (Cited on page 26.)

[LCHR03] Jens Lang, Weiming Cao, Weizhang Huang, and Robert 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. (Cited on pages 7 and 58.)

[Li05] R. Li. On multi-mesh h-adaptive methods. *J. Sc. Comput.*, 24(3):321–341, 2005. (Cited on page 7.)

[Lia91] G. Liao. On Harmonic Maps. In José E. Castillo, editor, *Mathematical Aspects of Numerical Grid Generation*, volume 8 of *Frontiers in Applied Mathematics*, pages 123–130. SIAM, 1991. (Cited on page 27.)

[Lia92] G. Liao. Variational approach to grid generation. *Numerical Methods for Partial Differential Equations*, 8(2):143–147, 1992. (Cited on page 25.)

[Lis96] V. D. Liseikin. The construction of structured adaptive grids—a review. *Comput. Math. Math. Phys.*, 36(1):1–32, 1996. (Cited on page 13.)

[LJL99] Feng Liu, Shanhong Ji, and Guojun Liao. An Adaptive Grid Method and Its Application to Steady Euler Flow Calculations. *SIAM J. Sci. Comput.*, 20(3):811–825, 1999. (Cited on pages 27, 37, and 39.)

[LL98] Peter D. Lax and Xu-Dong Liu. Solution of Two-Dimensional Riemann Problems of Gas Dynamics by Positive Schemes. *SIAM J. Sci. Comput.*, 19(2):319–340, 1998. (Cited on pages 86 and 109.)

[LLdlP02] G. Liao, Zh. Lei, and G.C. de la Pena. Adaptive grids for resolution enhancement. *Shock Waves*, 12(2):153–156, August 2002. (Cited on pages 3 and 39.)

[LPR99] S. Li, L. Petzold, and Y. Ren. Stability of moving mesh systems of partial differential equations. *SIAM J. Sci. Comput.*, 20(2):719–738, 1999. (Cited on page 43.)

[LPS94] G. Liao, T.W. Pan, and J. Su. Numerical grid generator based on Moser's deformation method. *Numerical Methods for Partial Differential Equations*, 10(1):21–31, 1994. (Cited on page 37.)

[LTZ01] Ruo Li, Tao Tang, and Pingwen Zhang. Moving mesh methods in multiple dimensions based on harmonic maps. *J. Comput. Phys.*, 170(2):562–588, 2001. (Cited on page 26.)

[LW03] Richard Liska and Burton 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. (Cited on page 102.)

[LX06] Guojun Liao and Jiaxing Xue. Moving meshes by the deformation method. *J. Comput. Appl. Math.*, 195(1):83–92, 2006. (Cited on page 39.)

[Mil81] K. Miller. Moving finite elements. II. *SIAM J. Numer. Anal.*, 18(6):1033–1057, 1981. (Cited on page 34.)

[Mil97] K. Miller. A geometrical-mechanical interpretation of gradient-weighted moving finite elements. *SIAM J. Numer. Anal.*, 34(1):67–90, 1997. (Cited on page 35.)

[MM81] K. Miller and R.N. Miller. Moving Finite Elements. I. *SIAM J. Numer. Anal.*, 18(6):1019–1032, 1981. (Cited on page 34.)

[MM07] J.A. Mackenzie and W.R. Mekwi. An Analysis of Stability and Convergence of a Finite Difference Discretisation of a Model Parabolic PDE in One Dimension Using a Moving Mesh. *IMA J. of Num. Anal.*, 27(3):507–528, 2007. (Cited on page 45.)

[MN07] J.A. Mackenzie and A. Nicola. A Discontinuous Galerkin Moving Mesh Method for Hamilton–Jacobi Equations. *SIAM J. Sci. Comput.*, 29:2258, 2007. (Cited on pages 6 and 28.)

[Mon81] G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Academie Royale des Sciences*, pages 666–704, 1781. (Cited on page 32.)

[Mor07] J.M. Morrell. *A cell by cell anisotropic adaptive mesh Arbitrary Lagrangian Eulerian method for the numerical solution of the Euler equations*. PhD thesis, Univ. of Reading, 2007. (Cited on page 58.)

[Mos65] J. Moser. On the volume elements on a manifold. *Transactions of the American Mathematical Society*, pages 286–294, 1965. (Cited on page 37.)

[MR02] J.A. Mackenzie and M.L. Robertson. A moving mesh method for the solution of the one-dimensional phase-field equations. *J. Comput. Phys.*, 181(2):526–544, 2002. (Cited on page 28.)

[MSB07] J.M. Morrell, P.K. Sweby, and A. Barlow. A cell by cell anisotropic adaptive mesh ALE scheme for the numerical solution of the Euler equations. *J. Comput. Phys.*, 226(1):1152–1180, 2007. (Cited on pages 7 and 58.)

[NK02] M. Nool and R. Keppens. AMRVAC: a multidimensional grid-adaptive magnetofluid dynamics code. *Comp. Meth. Appl. Math.*, 2:92–109, 2002. (Cited on pages 83 and 107.)

[OT79] Steven A. Orszag and Cha-Mei Tang. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. *J. Fluid Mech.*, 90(01):129–143, 1979. (Cited on page 124.)

[PD91] J.M. Picone and R.B. Dahlburg. Evolution of the Orszag–Tang vortex system in a compressible medium. II. Supersonic flow. *Physics of Fluids B: Plasma Physics*, 3:29, 1991. (Cited on page 124.)

[Rad26] T. Rado. Aufgabe 41. *Jahresber. Deutsche Math.-Verein*, 35:49, 1926. (Cited on page 27.)

[Rus61] V. V. Rusanov. Calculation of Intersection of Non-Steady Shock Waves with Obstacles. *J. Comput. Math. Phys. USSR*, 1:267–279, 1961. (Cited on page 90.)

[SGT+08] James M. Stone, Thomas A. Gardiner, Peter Teuben, John F. Hawley, and Jacob B. Simon. Athena: A New Code for Astrophysical MHD. *The Astrophysical Journal Supplement Series*, 178(1):137–177, 2008. (Cited on page 105.)

[SMR00] John M. Stockie, John A. Mackenzie, and Robert D. Russell. A moving mesh method for one-dimensional hyperbolic conservation laws. *SIAM J. Sci. Comput.*, 22(5):1791–1813 (electronic), 2000. (Cited on page 59.)

[SN92] J.M. Stone and M.L. Norman. ZEUS-2D: A Radiation Magnetohydrodynamics Code for Astrophysical Flows in Two Space Dimensions. II. The Magnetohydrodynamic Algorithms and Tests. *Astrophys. J. Suppl.*, 80:791–818, June 1992. (Cited on page 76.)

[Sod78] Gary A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys.*, 27(1):1–31, 1978. (Cited on page 121.)

[SRCG93] Carsten W. Schulz-Rinne, James P. Collins, and Harland M. Glaz. Numerical Solution of the Riemann Problem for Two-Dimensional Gas Dynamics. *SIAM J. Sci. Comput.*, 14(6):1394–1414, 1993. (Cited on page 86.)

[SS08] Ali Reza Soheili and John M. Stockie. A moving mesh method with variable mesh relaxation time. *Appl. Numer. Math.*, 58(3):249–263, March 2008. (Cited on page 43.)

[SY78] R. Schoen and S.T. Yau. On univalent harmonic maps between surfaces. *Inventiones Mathematicae*, 44(3):265–278, 1978. (Cited on page 26.)

[Tan05] Tao 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*, pages 185–218. American Mathematical Society, 2005. (Cited on pages 13, 42, and 60.)

[Tan06] H.Z. Tang. A moving mesh method for the Euler flow calculations using a directional monitor function. *Commun. Comput. Phys.*, 1:656–676, 2006. (Cited on pages 6, 29, 84, and 93.)

[Tan07] Zhijun Tan. Adaptive moving mesh methods for two-dimensional resistive magneto-hydrodynamic PDE models. *Comput. Fluids*, 36:758–771, May 2007. (Cited on pages 84 and 114.)

[TKB98] G. Tóth, R. Keppens, and M.A. Botchev. Implicit and semi-implicit schemes in the Versatile Advection Code: numerical tests. *Astron. & Astroph.*, 332:1159–1170, April 1998. (Cited on pages 59, 61, 77, 79, and 81.)

[TL79] P.D. Thomas and C.K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA J.*, 17(10):1030–1037, 1979. (Cited on page 39.)

[TLK08] Zhijun Tan, K.M. Lim, and B.C. Khoo. An Adaptive Moving Mesh Method for Two-Dimensional Incompressible Viscous Flows. *Commun. Comput. Phys*, 3(3):679–703, 2008. (Cited on page 29.)

[TO96] Gábor Tóth and Dušan Odstrčil. Comparison of Some Flux Corrected Transport and Total Variation Diminishing Numerical Schemes for Hydrodynamic and Magnetohydrodynamic Problems. *J. Comput. Phys.*, 128:82–100, October 1996. (Cited on pages 59, 61, 76, and 113.)

[Tor99] Eleuterio F. Toro. *Riemann solvers and numerical methods for fluid dynamics- A practical introduction*. Springer-Verlag, 1999. (Cited on page 91.)

[Tor02] M. Torrilhon. Exact solver and uniqueness conditions for Riemann problems of ideal magnetohydrodynamics. Research Report 2002-06, Eidgenössische Technische Hochschule, Seminar für Angewandte Mathematik, Zürich, April 2002. (Cited on page 69.)

[Tor03] M. Torrilhon. Non-uniform convergence of finite volume schemes for Riemann problems of ideal magnetohydrodynamics. *J. Comput. Phys.*, 192:73–94, November 2003. (Cited on pages 59, 74, and 75.)

[Tót96] G. Tóth. A General Code for Modeling MHD Flows on Parallel Computers: Versatile Advection Code. *Astrophysical Letters Communications*, 34:245–+, 1996. (Cited on page 69.)

[Tót00] Gábor Tóth. The $\nabla \cdot B = 0$ Constraint in Shock-Capturing Magnetohydrodynamics Codes. *J. Comput. Phys.*, 161(2):605–652, 2000. (Cited on pages 113, 115, 117, 123, and 125.)

[Tro94] Ron Trompert. *Local Uniform Grid Refinement for Time-dependent Partial Differential Equations*. PhD thesis, Universiteit van Amsterdam, 1994. (Cited on page 54.)

[TSS94] 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. (Cited on page 91.)

[TSW98] Joe F. Thompson, Bharat K. Soni, and Nigel P. Weatherill, editors. *Handbook of Grid Generation*. CRC Press, 1998. (Cited on page 13.)

[TT03] Huazhong Tang and Tao Tang. Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. *SIAM J. Numer. Anal.*, 41(2):487–515, 2003. (Cited on pages 6, 23, 29, 45, 60, 63, 65, 82, 84, 94, and 95.)

[TTM74] Joe F. Thompson, Frank C. Thames, and C. Wayne Mastin. Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *J. Comput. Phys.*, 15(3):299–319, 1974. (Cited on page 23.)

[TTZ03] H. Z. Tang, Tao Tang, and Pingwen Zhang. An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions. *J. Comput. Phys.*, 188(2):543–572, 2003. (Cited on pages 6, 29, 47, 54, 84, and 118.)

[TTZ06] Zhijun Tan, Tao Tang, and Zhengru Zhang. A simple moving mesh method for one- and two-dimensional phase-field equations. *J. Comput. Appl. Math.*, 190(1-2):252–269, 2006. International Conference on Mathematics and its Application. (Cited on page 6.)

[TVB93] R.A. Trompert, J.G. Verwer, and J.G. Blom. Computing brine transport in porous media with an adaptive-grid method. *Internat. J. Numer. Methods Fluids*, 16(1):43–63, 1993. (Cited on page 55.)

[TWM85] Joe F. Thompson, Z.U.A. Warsi, and C. Wayne Mastin. *Numerical grid generation: foundations and applications*. Elsevier North-Holland, Inc., New York, NY, USA, 1985. (Cited on pages 12 and 64.)

[TZHT04] Zhijun Tan, Zhengru Zhang, Yunqing Huang, and Tao Tang. Moving mesh methods with locally varying time steps. *J. Comput. Phys.*, 200:347–367, October 2004. (Cited on pages 55 and 93.)

[vDZ06] 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. (Cited on pages 9, 29, 47, 59, 84, 90, and 99.)

[VBFZ89] J.G. Verwer, J.G. Blom, R.M. Furzeland, and P.A. Zegeling. A moving-grid method for one-dimensional PDEs based on the method of lines. In Joseph E. Flaherty, Pamela J. Paslow, Mark S. Shephard, and John D. Vasilakis, editors, *Adaptive methods for partial differential equations. Proceedings of the Workshop on Adaptive Computational Methods for Partial Differential Equations.*, 1989. (Cited on page 53.)

[vD02] Arthur van Dam. A Moving Mesh Finite Volume Solver for Macroscopic Traffic Flow Models. Master's thesis, Utrecht University, May 2002. (Cited on pages vii and 82.)

[vdHK07] 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. (Cited on pages 7, 54, 56, 72, 83, 107, and 113.)

[vdHKM08] B. van der Holst, R. Keppens, and Z. Meliani. A multidimensional grid-adaptive relativistic magnetofluid code. *Comput. Phys. Commun.*, 179(9):617–627, 2008. (Cited on pages 7, 113, and 124.)

[vdVvdV98] J. J. W. van der Vegt and H. van der Ven. Discontinuous Galerkin Finite Element Method with Anisotropic Local Grid Refinement for Inviscid Compressible Flows. *J. Comput. Phys.*, 141(1):46–77, 1998. (Cited on page 56.)

[vdVvdV02] J. J. W. van der Vegt and H. van der Ven. Space–time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: I. general formulation. *J. Comput. Phys.*, 182(2):546–585, 2002. (Cited on pages 7 and 56.)

[vDZ07] A. van Dam and P.A. Zegeling. Mesh speeds versus solution interpolation – making moving mesh results sharper. In *Proceedings of ICFD'07*, Reading, UK, March 2007. (Cited on page vii.)

[vDZ09] A. van Dam and P.A. Zegeling. Balanced monitoring of flow phenomena in moving mesh methods. To appear in Commun. Comput. Phys. DOI:10.4208/cicp.2009.09.033, 2009. (Cited on pages 9, 23, 29, and 83.)

[vL77a] Bram 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. (Cited on pages 66 and 89.)

[vL77b] Bram van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *J. Comput. Phys.*, 23:276–299, March 1977. (Cited on pages 67 and 89.)

[vL79] Bram van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *J. Comput. Phys.*, 32:101–136, July 1979. (Cited on page 60.)

[VR92] A. E. P. Veldman and K. Rinzema. Playing with nonuniform grids. *Journal of Engineering Mathematics*, 26(1):119–130, February 1992. (Cited on page 53.)

[Wac05] A. Wacher. MMPDE vs. SGWMFE Experiments in one dimension. Technical Report NA-05/21, Oxford University Computing Laboratory, 2005. (Cited on page 36.)

[WBG05] B.V. Wells, M.J. Baines, and P. Glaister. Generation of Arbitrary Lagrangian-Eulerian (ALE) velocities, based on monitor functions, for the solution of compressible fluid equations. *Internat. J. Numer. Methods Fluids*, 47:1375–1381, 2005. (Cited on page 42.)

[WC84] Paul Woodward and Phillip Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *J. Comput. Phys.*, 54:115–173, April 1984. (Cited on pages 3, 109, and 127.)

[Win66] Alan M. Winslow. Numerical Solution of the Quasilinear Poisson Equation in a Nonuniform Triangle Mesh. *J. Comput. Phys.*, 1(2):149–172, November 1966. (Cited on page 21.)

[Win81] A.M. Winslow. Adaptive-mesh zoning by the equipotential method. Technical Report UCID-19062, Lawrence Livermore Laboratory, 1981. (Cited on pages 2, 23, and 93.)

[WS07] A. Wacher and I. Sobey. String Gradient Weighted Moving Finite Elements in Multiple Dimensions with Applications in Two Dimensions. *SIAM J. Sci. Comput.*, 29:459–480, 2007. (Cited on page 36.)

[WSM05] Abigail Wacher, Ian Sobey, and Keith Miller. String gradient weighted moving finite elements. *Internat. J. Numer. Methods Fluids*, 47:1329–1336, 2005. (Cited on page 36.)

[YKL+79] N. N. Yanenko, V. M. Kovenya, V. D. Lisejkin, V. M. Fomin, and E. V. Vorozhtsov. On some methods for the numerical simulation of flows with complex structure. In *Sixth International Conference on Numerical Methods in Fluid Dynamics*, volume 90 of *Lecture Notes in Physics*, pages 565–578. Springer Berlin / Heidelberg, 1979. (Cited on page 25.)

[YLK77] N. Yanenko, V. Lisseikin, and V. Kovenia. The method of the solution of gaz dynamical problems in moving meshes. *Computing Methods in Applied Sciences and Engineering*, II:48–61, 1977. (Cited on page 25.)

[ZB92] P. A. Zegeling and J. G. Blom. An evaluation of the gradient-weighted moving-finite-element method in one space dimension. *J. Comput. Phys.*, 103(2):422–441, 1992. (Cited on page 36.)

[ZdBT05] 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*, pages 419–430. American Mathematical Society, 2005. (Cited on pages 23, 29, 60, and 84.)

[Zeg05] P.A. Zegeling. On Resistive MHD Models with Adaptive Moving Meshes. *Journal of Scientific Computing*, 24(2):263–284, 2005. (Cited on pages 23, 29, 84, and 113.)

[Zeg07] 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, pages 251–296. Science Press, Beijing, 2007. Workshop on Adaptive mesh methods, Beijing University, 2005. (Cited on page 13.)

[Zha06] Z. Zhang. Moving mesh method with conservative interpolation based on L2-projection. *Commun. Comput. Phys.*, 1:930–944, 2006. (Cited on page 95.)

[Zie98] Udo Ziegler. NIRVANA+: An adaptive mesh refinement code for gas dynamics and MHD. *Comput. Phys. Commun.*, 109(2-3):111–134, 1998. (Cited on page 113.)

[Zie08] U. Ziegler. The NIRVANA code: Parallel computational MHD with adaptive mesh refinement. *Computer Physics Communications*, 179(4):227–244, 2008. (Cited on page 113.)

[ZK01] P.A. Zegeling and R. Keppens. Adaptive Method of Lines for Magneto-Hydrodynamic PDE Models. In A. Vande Wouwer, Ph. Saucez, and W.E. Schiesser, editors, *Adaptive Method of Lines*, pages 117–137. Chapman & Hall/CRC Press, 2001. (Cited on pages 59, 60, 69, 71, 77, 79, and 81.)

[ZLI] P.A. Zegeling, I. Lagzi, and F. Izsak. Simulation of the Liesegang precipitation rings using an adaptive grid PDE method. (submitted). (Cited on page 6.)

Index

Samenvatting

Dit proefschrift valt binnen het vakgebied Computational Science, of preciezer: Numerieke Wiskunde. Binnen dit brede gebied richten wij ons op computersimulaties van (veelal) natuurkundige processen, zoals luchtstroming of stroming van magnetische plasma's. De natuurkundige wetten die beschrijven hoe de diverse processen in zo'n luchtstroming verlopen zijn veelal bekend. Wanneer we een echt 'stuk lucht' gaan bekijken en willen voorspellen hoe er draaiingen of drukgolven zullen optreden, is het echter bijzonder lastig—zeg maar gerust ondoenlijk—om met pen en papier al deze natuurwetten te gaan doorrekenen. Zelfs computers doen dit niet exact; we laten ze een benadering berekenen van de echte oplossing. Zo wordt het dus een echte voorspelling die een fout bevat; om deze reden zitten weersvoorspellingen er nog wel eens naast.

Het benaderen van de oplossing gebeurt door het gebied waar je de stroming wilt doorrekenen (het domein) op te delen in kleine hokjes, bijvoorbeeld allemaal vierkantjes.[2] In plaats van op *elk* willekeurig punt in het domein de stroming door te rekenen, berekenen we enkel voor elk hokje de *gemiddelde* waarden van gasdichtheid, snelheid, etcetera. Het opdelen van het domein in hokjes en het doorrekenen van de natuurwetten met gemiddelde waarden hierop wordt discretisatie genoemd.

Hoe groter we de hokjes maken, des te minder er nodig zijn om het hele domein te bedekken en des te sneller de computer klaar is met het rekenwerk. Echter, hoe groter een hokje is, des te grover de gemiddelde waarden zijn en dus ook de fout die we in de voorspelling maken. Het kiezen tussen weinig hokjes (snelle simulaties) en veel kleine hokjes (nauwkeuriger resultaten) is een constant dilemma. Dit brengt ons bij het onderwerp van dit proefschrift. De stroming op een domein zal op bepaalde plaatsen interessant zijn (drukgolven die tegen elkaar botsen, draaiingen door tegengestelde stromingen), maar op diverse andere gebieden is de stroming mogelijk heel kalm. Het zou ideaal zijn om in die interessante gebieden de hokjes veel kleiner te maken en in de kalme gebieden wat groter. We beginnen met allemaal gelijke vierkantjes en trekken vervolgens de kruispunten in dit vierkante rooster richting de interessante gebieden, zodat de vierhoekjes daar kleiner en mogelijk beter vervormd zijn. In de kalme gebieden zijn de vierhoeken dan automatisch meer opgerekt. Het vervormde vierkante rooster wordt een *adaptief rooster* genoemd. Wanneer we de stroming voor langere tijd doorrekenen en het rooster continu mee beweegt met de stroming, heet dit een *bewegend rooster*. De techniek wordt ook wel r-verfijning genoemd, naar *relocatie*.

[2] In dit voorbeeld neem ik een 'plat' domein aan, dus tweedimensionaal en niet driedimensionaal zoals in de echte wereld. Het uitvoeren van tweedimensionale simulaties leert ons veel dat van pas zal komen bij het later uitvoeren van driedimensionale simulaties.

Een cruciaal onderdeel in deze roosterbeweging is hoe er bepaald wordt wat 'interessante' gebieden zijn. Het menselijk oog herkent wel fenomenen als schokgolven en draaiingen, maar hoe kunnen we dit automatiseren? De computer zal aan elke locatie in het rooster een getal toekennen dat aangeeft hoe belangrijk roosterverfijning daar is. Hoe groter het getal, des te kleiner de hokjes daar zullen worden. Dit getal komt uit de zogenaamde monitorfunctie die de oplossing 'monitort'. Deze meet bijvoorbeeld hoe snel de luchtdichtheid ergens verandert of de stromingssnelheid. Een juiste weging van al deze waarden moet ervoor zorgen dat de roosterbeweging gebalanceerd verloopt.

In de navolgende tekst vatten we alle hoofdstukken inhoudelijk samen, met meer gebruik van jargon.

In Hoofdstuk 1 geven we een korte historische introductie van bewegende roostermethoden. We schetsen de meest recente resultaten van anderen op het gebied van roosterverfijning en oplosmethoden voor fysische vergelijkingen, en duiden hoe dit proefschrift zich daartussen positioneert.

In Hoofdstuk 2 geven we een uitgebreid overzicht van de diverse methoden voor roosterbeweging, hun historie en onderlinge verschillen en overeenkomsten. In vergelijking met eerdere overzichtswerken wordt een breed spectrum aan methoden beschouwd en is bovendien het meest recente werk (t/m 2009) toegevoegd. De beschouwing gaat uit van drie onderscheidende factoren: het voorschrift van de roosterbeweging, de combinatie met oplosmethoden voor de fysische vergelijkingen en de aansturing van roosterbeweging door monitorfuncties. Het voorschrijven van roosterbeweging kan globaal gezien op twee verschillende manieren. De locatiegebaseerde methode gebruikt een partiële differentiaalvergelijking (PDV) of bijvoorbeeld een optimalisatieprobleem om de coördinaten, oftewel de puntlocaties, te bepalen. Veel van dit soort methoden zijn te relateren aan het equidistributieprincipe van De Boor en de variabele-diffusie van Winslow. De recent opgekomen toepassing van Monge–Kantorovich optimalisatie is ook locatiegebaseerd en garandeert equidistributie in willekeurig veel dimensies. Er is nog weinig onderzoek gedaan naar de efficiëntie van deze methode voor tijdsafhankelijke PDV-problemen. De snelheidsgebaseerde methode definieert de puntsnelheid, ook middels een PDV en integreert deze vervolgens in de tijd om de roosterpunten naar de juiste locatie te bewegen. De deformatiemethode van Liao garandeert hiermee equidistributie in ruimtes van willekeurig hoge dimensie.

In Hoofdstuk 3 beschrijven we onze eerste eigen toepassing van eendimensionale roosterbeweging op problemen uit de ideale magnetohydrodynamica. Dit fysische model beschrijft de beweging van een geïoniseerd gas (plasma) en bevat dus naast de drie gaswetten van Euler ook termen voor de Lorentzkracht en een extra PDV die inductie van een magnetisch veld beschrijft. Deze extra vergelijkingen brengen ook extra karakteristieke golven met zich mee, waardoor testproblemen zeer rijke stromingsfenomenen kunnen vertonen en roosterbeweging extra belangrijk wordt. De fysische PDV's worden opgelost met eindige volumes en de lokale Lax–Friedrichsflux en is een tweedeorde methode dankzij slope-limiting en MUSCL-type oplossingsreconstructie. De voornaamste verbetering is de monitorfunctie die een oplossingsgradiënt ba-

lanceert middels een automatisch gekozen tijdsafhankelijke constante. Voorheen werd deze handmatig gezet, dus de nieuwe methode voorkomt een hoop giswerk door de gebruiker. Hierdoor kunnen onverwachte fenomenen ontdekt worden. Zo bleek de roosterbeweging bij het 'oscillating plasma sheet' probleem nauwkeurig extra snelle golven op te pikken, en daarmee kwamen we achter de zogenaamde fysische trapvorming. Voorheen had men dit nog niet opgemerkt.

In Hoofdstuk 4 begint de werkelijke uitdaging qua rekenkracht: tweedimensionale roosterbeweging voor compressibele gassen. De volledig nieuw ontwikkelde Fortran 95 code is generiek opgezet voor de brede klasse van hyperbolische stelsels van niet-lineaire PDV's. Voornaamste punt in dit hoofdstuk is een extra balancering van de monitorfunctie. Namelijk, die tussen componenten *onderling*. De monitorfunctie kan bestaan uit meerdere componenten, bijvoorbeeld een dichtheidsgradiënt en de vorticiteit van de stroming. In het voorgaande hoofdstuk werd *binnen één component* al een balancering gebruikt, zodat subtiele fenomenen in de oplossing niet volledig gedomineerd werden door sterke fenomenen. Dit blijkt ook nodig te zijn *tussen meerdere componenten*. Preciezer: wanneer de ratio maximum/gemiddelde voor één component veel groter is dan voor de andere, dan moet dit genormaliseerd worden. Deze operatie is zeer goedkoop, en levert significante verbetering van de roosterbeweging op. Een implosie-probleem laat zien dat ook over zeer lange tijdsintegraties de roosterverfijning goed gebalanceerd blijft, waarbij fysische instabiliteiten, zoals Kelvin–Helmholtz en Richtmyer–Meshkov, scherp gerepresenteerd worden. Een extra verbetering wordt gevormd door de HLLC-numerieke flux, die met slechts 10% extra kosten veel scherpere resultaten geeft ten opzichte van de lokale Lax–Friedrichsflux.

In Hoofdstuk 5 hebben we de methode uit Hoofdstuk 4 uitgebreid naar tweedimensionale ideale magnetohydrodynamica. Was dit in Hoofdstuk 3 nog triviaal, nu speelt een extra fysische eis een belangrijke rol. Het magneetveld moet divergentievrij blijven om magnetische monopolen uit te sluiten en om dit te garanderen gebruiken we de zogenaamde vectorpotentiaalformulering. De PDV voor het magneetveld wordt niet aan de set van hyperbolische PDV's toegevoegd, maar het magneetveld wordt afgeleid als de rotatie van een potentiaalveld. Dit garandeert de divergentie-eis tot op machineprecisie, maar introduceert wel een extra vergelijking voor het potentiaalveld. Deze moet niet alleen opgelost worden, maar vereist ook een extra interpolatiestap in de roosterwegingsfase. De eerste resultaten zijn bemoedigend. De rooster-aanpassing en oplossingsmonitoring werkt nog steeds goed, zoals verwacht. Het meest kritieke onderdeel blijkt de evolutie van de vectorpotentiaal te zijn. Deze is nu van eersteorde nauwkeurigheid, maar voor hogere resolutie is meer stabiliteit nodig en dus minstens een tweedeorde methode.

Over het geheel genomen is dit proefschrift een toepassingsgestuurde verkenning van roosterbeweging en bijbehorende oplossingsmonitoring. De theoretische fundering was al aanwezig, en de generieke oplosmethode en de automatisch (dubbel) gebalanceerde monitorfunctie maken de resulterende methode direct toepasbaar op zeer diverse problemen.

Curriculum Vitae

Arthur van Dam was born in Noordeloos on September 20, 1979. He attended the Rijksscholengemeenschap Wijdschild—later Merewade College—in Gorinchem, where he obtained his Atheneum diploma in 1997.

That same year, he started studying Computational Science at Utrecht University, which ended in an internship at TNO Traffic and Transport in Delft. He graduated in 2002 (cum laude) with a thesis titled '*A Moving Mesh Finite Volume Solver for Macroscopic Traffic Flow Models*'.

In the meantime he had also picked up a studies Computer Science at Utrecht University. He graduated within the Software Technology group in 2004 with a thesis titled '*Extending Dynamic Rules, An Application-Oriented Study Into Stratego's New Dynamic Rules*'.

In October 2004, Arthur returned to mathematics as a Ph.D. student at the Mathematical Institute of Utrecht University under the guidance of dr. P.A. Zegeling. The NWO-funded project '*Adaptive moving mesh methods for higher-dimensional nonlinear hyperbolic conservation laws*' was partly inspired by his master's thesis from 2002. This dissertation contains the majority of the work conducted in this project. During this period, Arthur took courses at the Von Karman Institute in Belgium and the University of Lecce in Italy. He attended seminars and conferences in Zeist, Orlando (FL), Leiden, Amsterdam, Townsville, Reading, Enschede, Eindhoven, Moscow and Delft, at most of these giving talks himself. He was a visiting researcher at the Centre for Plasma Astrophysics of the Katholieke Universiteit Leuven in August 2007 and November 2008, for cooperation with prof. dr. R. Keppens.

At the Mathematical Institute, Arthur taught several undergraduate courses and set up the new form of the master course 'Laboratory Class Scientific Computing'. He also organized the Numerical Mathematics Colloquium in the period 2005–2009. In 2006 and 2007 he was co-organizer of the PhDays events.

And what have you got? At the end of the day.
What have you got? To take away?

— Dire Straits, Private Investigations