ESAIM: PROCEEDINGS, December 2011, Vol. 34, p. 97-150
E. Cancès, V. Louvet and M. Massot, Editors

BLOCK-STRUCTURED ADAPTIVE MESH REFINEMENT - THEORY, IMPLEMENTATION AND APPLICATION

Ralf Deiterding[1]

Abstract. Structured adaptive mesh refinement (SAMR) techniques can enable cutting-edge simulations of problems governed by conservation laws. Focusing on the strictly hyperbolic case, these notes explain all algorithmic and mathematical details of a technically relevant implementation tailored for distributed memory computers. An overview of the background of commonly used finite volume discretizations for gas dynamics is included and typical benchmarks to quantify accuracy and performance of the dynamically adaptive code are discussed. Large-scale simulations of shock-induced realistic combustion in non-Cartesian geometry and shock-driven fluid-structure interaction with fully coupled dynamic boundary motion demonstrate the applicability of the discussed techniques for complex scenarios.

Contents

[1] Oak Ridge National Laboratory, P.O. Box 2008 MS-6367, Oak Ridge, TN 37831, USA, E-mail: deiterdingr@ornl.gov

Introduction

The most important discretization approach for conservation laws is the finite volume method, which is constructed particularly to account properly for the discontinuous solutions inherent to hyperbolic equations. Today, numerous high-resolution shock-capturing schemes are available for those that provide proper upwinding based on the local characteristic information, are non-oscillatory across discontinuities and achieve higher order in regions where the solution is smooth. The canonical set of equations considered are usually the Euler equations for a single polytropic gas and computational examples are typically one-, occasionally two-dimensional. The provided computer codes have in general a few hundred lines and utilize a uniform Cartesian mesh on a single-processor system.

Predictive computational science, involving hyperbolic (sub-)problems, however, usually requires highly resolved results in typically three space dimensions. The computational costs increase dramatically mandating the use of parallel high-performance computing systems and the application of mesh adaptation on-the-fly. When parallelism and dynamic mesh modification need to be combined in a single implementation, one is faced with a considerable increase in algorithmic and software complexity. In here, we consider primarily the block-structured adaptive mesh refinement (SAMR) method for hyperbolic conservation laws [14,16] including its parallelization and application to numerous realistic scientific computing problems.

We start in Section 1 with a specification of the problem class, the introduction of the finite volume method, and the description of the most important classes of high-resolution shock-capturing schemes. Two frequently used upwind schemes for the Euler equations are given. We contrast the SAMR approach to other mesh adaptation techniques and give a brief overview of freely available SAMR software. In Section 2, we define the SAMR method exactly. The description is topologically accurate and intended as an unambiguous algorithmic basis for an error-free implementation. All necessary sub-algorithms are detailed. We describe the rigorous domain decomposition approach chosen for parallelization in our own SAMR framework AMROC [31] and discuss its basic software design, as one example for implementing the described algorithms with object-oriented concepts. Three typical SAMR test cases for Euler equations are given to verify and benchmark the software. Finally, Section 3 is devoted to the utilization of SAMR techniques for large-scale and technically relevant computations. We describe a straightforward level-set-based approach for considering complex, moving boundaries. The primary application of the embedded boundary method is fluid-structure interaction (FSI) simulation and we sketch the algorithmic and software extensions implemented in our FSI system Virtual Test Facility [41] to enable this problem class. Further on, we describe well-resolved simulations of shock-induced combustion with detailed chemistry, for which mesh adaptation is particularly efficient. Necessary extensions of the previously described standard upwind schemes are also included, as examples for realistic hyperbolic systems and to ensure reproducibility of the given results.

1. Fundamentals

1.1. Hyperbolic conservation laws

In the following we are concerned with the construction of advanced adaptive finite volume methods for hyperbolic conservation laws. In Cartesian coordinates such conservation laws have the structure

$$\frac{\partial}{\partial t}\mathbf{q}(\mathbf{x},t) + \sum_{n=1}^{d} \frac{\partial}{\partial x_n}\mathbf{f}_n(\mathbf{q}(\mathbf{x},t)) = \mathbf{s}(\mathbf{q}(\mathbf{x},t))\,, \quad \mathbf{x}\in\mathbb{R}^d\,, \quad t>0\,. \tag{1}$$

Herein, $t \in \mathbb{R}_0^+$ denotes the time and $\mathbf{x} = (x_1,\ldots,x_d)^T \in \mathbb{R}^d$ denotes a point in Cartesian coordinates. The vector-valued mapping $\mathbf{q} = \mathbf{q}(\mathbf{x},t)$ from $D := \{(\mathbf{x},t) \in \mathbb{R}^d \times \mathbb{R}_0^+\}$ into the space of *admissible* states $S \subset \mathbb{R}^M$ is called *vector of state*. The components of the vector of states are physical meaningful quantities, like mass, momentum or energy, that have to be conserved because of fundamental physical principles. The functions $\mathbf{f}_n(\mathbf{q})$ are called *flux functions*, $\mathbf{s}(\mathbf{q})$ is a *source term*.

Definition 1. (Hyperbolicity). *Let $\mathbf{A}_n(\mathbf{q}) = \partial \mathbf{f}_n(\mathbf{q})/\partial \mathbf{q}$ denote the Jacobian matrix of flux function $\mathbf{f}_n(\mathbf{q})$. System (1) is called hyperbolic, if the matrix $\mathbf{A}(\mathbf{q},\nu) = \nu_1 \mathbf{A}_1(\mathbf{q}) + \cdots + \nu_d \mathbf{A}_d(\mathbf{q})$ has M real eigenvalues $\lambda_1(\mathbf{q},\nu) \le ... \le \lambda_M(\mathbf{q},\nu)$ and M linear independent right eigenvectors $\mathbf{r}_m(\mathbf{q},\nu)$, $m = 1,\dots,M$ defined by $\mathbf{A}(\mathbf{q},\nu)\,\mathbf{r}_m(\mathbf{q},\nu) = \lambda_m(\mathbf{q},\nu)\,\mathbf{r}_m(\mathbf{q},\nu)$ for all admissible states $\mathbf{q} \in S$ and $\nu = (\nu_1,\dots,\nu_d) \in \mathbb{R}^d$ with $|\nu_1| + \cdots + |\nu_d| > 0$.*

The theory of hyperbolic conservation laws is very well established, cf. [49, 66, 72, 79], and it is well known that in the general case of nonlinear flux functions classical solutions are guaranteed to exist only for small times. Even continuously differentiable initial data may be steepened to discontinuities, cf. [72, 79], and an integral formulation with less differentiability is required instead of Eq. (1) to define weak solutions, e.g.,

$$\int\limits_{\Omega} \mathbf{q}(\mathbf{x}, t+\Delta t)\, d\mathbf{x} - \int\limits_{\Omega} \mathbf{q}(\mathbf{x}, t)\, d\mathbf{x} + \sum_{n=1}^{d} \int\limits_{t}^{t+\Delta t} \int\limits_{\partial\Omega} \mathbf{f}_n(\mathbf{q}(\mathbf{o}, t))\, \sigma_n(\mathbf{o})\, d\mathbf{o}\, dt = \int\limits_{t}^{t+\Delta t} \int\limits_{\Omega} \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, d\mathbf{x}\ , \tag{2}$$

cf. [108]. Herein, σ_n denotes the n-th component of $\mathbf{n}$, the outward unit normal vector of $\partial\Omega$, the boundary of the problem domain Ω.

1.2. Finite volume methods

Finite volume (FV) methods are tailored for problems with discontinuities and are built on Eq. (2). Numerous text books are available nowadays, cf. [49, 59, 66, 72, 111] and we restrict the description to the basic concepts. Without loss of generality we assume $d = 2$ in the following.

1.2.1. *Discretization*

Let the computational domain, D, be discretized with a rectangular grid with mesh widths $\Delta x_1, \Delta x_2$ in each coordinate direction and a time step Δt. The discrete mesh points are then defined by $(x_1^j, x_2^k) := \left(\left(j+\frac{1}{2}\right)\Delta x_1, \left(k+\frac{1}{2}\right)\Delta x_2\right)$, $j,k \in \mathbb{Z}$. Further on, it is useful to define $x_1^{j-1/2} := x_1^j - \frac{\Delta x_1}{2}$, $j \in \mathbb{Z}$ and $x_2^{k-1/2} := x_2^k - \frac{\Delta x_2}{2}$, $k \in \mathbb{Z}$. Discrete time values are defined by $t_i := i\Delta t$, $i \in \mathbb{N}_0$ and we denote the value in the discrete point (x_1^j, x_2^k, t_i) by $\mathbf{Q}_{jk}^i$. We define a rectangular computational cell C_{jk} around each mesh point (x_1^j, x_2^k). The domain of cell C_{jk} is $I_{jk} = [x_1^{j-1/2}, x_1^{j+1/2}] \times [x_2^{k-1/2}, x_2^{k+1/2}]$. We use I_{jk} and the discrete time interval $[t_i, t_{i+1}[$ as integration domain in the integral form (2) and obtain

$$\int\limits_{I_{jk}} \mathbf{q}(\mathbf{x}, t_{i+1})\, d\mathbf{x} - \int\limits_{I_{jk}} \mathbf{q}(\mathbf{x}, t_i)\, d\mathbf{x} + \sum_{n=1}^{d} \int\limits_{t_i}^{t_{i+1}} \int\limits_{\partial I_{jk}} \mathbf{f}_n(\mathbf{q}(\mathbf{o}, t))\, \sigma_n(\mathbf{o})\, d\mathbf{o}\, dt = \int\limits_{t_i}^{t_{i+1}} \int\limits_{I_{jk}} \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, d\mathbf{x}\, dt\ . \tag{3}$$

Within each computational cell C_{jk} the value $\mathbf{Q}_{jk}(t)$ is an approximation to the exact cell average value

$$\mathbf{Q}_{jk}(t) \approx \frac{1}{|I_{jk}|} \int\limits_{I_{jk}} \mathbf{q}(\mathbf{x}, t)\, d\mathbf{x}\ . \tag{4}$$

By employing the approximated values $\mathbf{Q}_{jk}(t)$ instead of $\mathbf{q}(\mathbf{x}, t)$ as argument for $\mathbf{s}(\mathbf{q}(\mathbf{x}, t))$ a natural approximation to the cell average of the source term function is found immediately:

$$\mathbf{s}(\mathbf{Q}_{jk}(t)) \approx \frac{1}{|I_{jk}|} \int\limits_{I_{jk}} \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, d\mathbf{x} \tag{5}$$

Furthermore, we define numerical flux functions $\mathbf{F}^n$ at the sides of C_{jk} by

$$\mathbf{F}^{1,\pm 1/2}_{jk}(\mathbf{Q}(t)) \approx \frac{1}{\Delta x_2} \int\limits_{x_2^{k-1/2}}^{x_2^{k+1/2}} \mathbf{f}_1(\mathbf{q}(x_1^{j\pm 1/2}, x_2, t))\, dx_2 \;, \quad \mathbf{F}^{2,\pm 1/2}_{jk}(\mathbf{Q}(t)) \approx \frac{1}{\Delta x_1} \int\limits_{x_1^{j-1/2}}^{x_1^{j+1/2}} \mathbf{f}_2(\mathbf{q}(x_1, x_2^{k\pm 1/2}, t))\, dx_1 \;. \tag{6}$$

We insert these approximations into Eq. (3) and divide by $|I_{jk}|$. We obtain

$$\mathbf{Q}_{jk}(t_{i+1}) = \mathbf{Q}_{jk}(t_i) - \sum_{n=1}^{d} \frac{1}{\Delta x_n} \int\limits_{t_i}^{t_{i+1}} \left(\mathbf{F}^{n,+1/2}_{jk}(\mathbf{Q}(t)) - \mathbf{F}^{n,-1/2}_{jk}(\mathbf{Q}(t)) \right) dt + \int\limits_{t_i}^{t_{i+1}} \mathbf{s}(\mathbf{Q}_{jk}(t))\, dt \;. \tag{7}$$

If the Euler Method is used to approximate all time integrals of Eq. (7), the time-explicit scheme

$$\mathbf{Q}^{i+1}_{jk} = \mathbf{Q}^{i}_{jk} - \sum_{n=1}^{d} \frac{\Delta t}{\Delta x_n} \left(\mathbf{F}^{n,+1/2}_{jk}(\mathbf{Q}^i) - \mathbf{F}^{n,-1/2}_{jk}(\mathbf{Q}^i) \right) + \Delta t\, \mathbf{s}(\mathbf{Q}^{i}_{jk}) \tag{8}$$

is derived. For $\mathbf{s} \equiv 0$ the scheme is discretely conservative, i.e., $\sum_{j,k\in\mathbb{Z}} \mathbf{Q}^{i+1}_{jk} = \sum_{j,k\in\mathbb{Z}} \mathbf{Q}^{i}_{jk}$. As written here, the scheme (8) is just first-order accurate, yet *high-resolution* methods are available that use the method of lines and spatial inter- and extrapolation in the approximation of $\mathbf{F}^n$ to achieve at least second-order accurate in smooth solution regions and resolve discontinuities sharply (cf. Section 1.4).

1.2.2. *The method of fractional steps*

In many applications it is convenient to apply the time-operator splitting approach or method of fractional steps [62] to numerically decouple the source term $\mathbf{s}(\mathbf{q})$ from the partial differential equation. The homogeneous partial differential equation

$$\frac{\partial \mathbf{q}}{\partial t} + \sum_{n=1}^{d} \frac{\partial}{\partial x_n} \mathbf{f}_n(\mathbf{q}) = 0 \;, \;\; \text{IC: } \mathbf{Q}^i \overset{\Delta t}{\Longrightarrow} \tilde{\mathbf{Q}}^{i+1} \tag{9}$$

and the ordinary differential equation

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{s}(\mathbf{q}) \;, \;\; \text{IC: } \tilde{\mathbf{Q}}^{i+1} \overset{\Delta t}{\Longrightarrow} \mathbf{Q}^{i+1} \tag{10}$$

are solved successively with the result of the preceding step as initial condition (IC). If we denote the discrete solution operator of (9) by $\mathcal{H}^{(\Delta t)}$ and the discrete operator of (10) by $\mathcal{S}^{(\Delta t)}$, the entire splitting scheme (9-10) reads

$$\mathbf{Q}^{i+1} = \mathcal{S}^{(\Delta t)} \mathcal{H}^{(\Delta t)}(\mathbf{Q}^i) \;. \tag{11}$$

A second-order accurate alternative is

$$\mathbf{Q}^{i+1} = \mathcal{S}^{(\frac{1}{2}\Delta t)} \mathcal{H}^{(\Delta t)} \mathcal{S}^{(\frac{1}{2}\Delta t)}(\mathbf{Q}^i) \;, \tag{12}$$

which is called *Strang splitting* [110]. The idea of operator splitting can also be applied to the solution of sub-problem (9), i.e., to the homogeneous operator $\mathcal{H}^{(\Delta t)}$. A simple dimensional splitting scheme in two space dimensions is

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial}{\partial x_1} \mathbf{f}_1(\mathbf{q}) = 0 \;, \;\; \text{IC: } \mathbf{Q}^i \overset{\Delta t}{\Longrightarrow} \tilde{\mathbf{Q}}^{1/2} \;, \quad \frac{\partial \mathbf{q}}{\partial t} + \frac{\partial}{\partial x_2} \mathbf{f}_2(\mathbf{q}) = 0 \;, \;\; \text{IC: } \tilde{\mathbf{Q}}^{1/2} \overset{\Delta t}{\Longrightarrow} \mathbf{Q}^{i+1} \;. \tag{13}$$

By denoting the dimensional steps by $\mathcal{X}_1^{(\Delta t)}$ and $\mathcal{X}_2^{(\Delta t)}$ scheme (13) is written in analogy to scheme (11) as $\mathbf{Q}^{i+1} = \mathcal{X}_2^{(\Delta t)}\mathcal{X}_1^{(\Delta t)}(\mathbf{Q}^i)$. Like the standard *Godunov splitting*, Eq. (11), the latter scheme is first-order accurate in time if the solution operators $\mathcal{X}_n^{(\Delta t)}$ are at least first-order accurate [111]. A second-order accurate scheme (provided that the operators $\mathcal{X}_n^{(\Delta t)}$ are at least second-order) is $\mathbf{Q}^{i+1} = \mathcal{X}_1^{(\frac{1}{2}\Delta t)}\mathcal{X}_2^{(\Delta t)}\mathcal{X}_1^{(\frac{1}{2}\Delta t)}(\mathbf{Q}^i)$. Dimensional splitting is a simple and efficient means of extending (high-resolution) schemes, that originally have been developed in one space dimension, to multiple dimensions. Therefore, we restrict the following presentation of numerical schemes for the hydrodynamic transport to the one-dimensional case.

1.3. **Upwind schemes**

High-resolution finite volumes schemes are usually built on first-order accurate upwind methods that utilize characteristic information. In order to introduce the idea of upwinding and to supply the basis of the flux-difference splitting methods (see Section 1.3.3) we study the linear case first.

1.3.1. *Linear upwind scheme*

In the case of a linear hyperbolic equation

$$\frac{\partial}{\partial t}\mathbf{q}(x,t) + \mathbf{A}\frac{\partial}{\partial x}\mathbf{q}(x,t) = \mathbf{0}\,, \quad x \in \mathbb{R}\,, \quad t > 0 \tag{14}$$

a finite volume scheme can easily be built by considering the analytic solution of the Cauchy problem between two constant states $\mathbf{q}_L$ and $\mathbf{q}_R$ (also denoted as *Riemann Problem*). Assuming (for simplicity) that $\mathbf{A}$ has M *distinct* real eigenvalues $\lambda_1 < \cdots < \lambda_M$ with M linear independent right eigenvectors $\mathbf{r}_m$, $m = 1,\ldots,M$, the exact solution of the Riemann Problem reads

$$\mathbf{q}(x,t) = \mathbf{q}_L + \sum_{\lambda_m < x/t} a_m \mathbf{r}_m = \mathbf{q}_R - \sum_{\lambda_m \geq x/t} a_m \mathbf{r}_m = \sum_{\lambda_m \geq x/t} \delta_m \mathbf{r}_m + \sum_{\lambda_m < x/t} \beta_m \mathbf{r}_m\,, \tag{15}$$

where

$$\mathbf{q}_L = \sum_{m=1}^{M} \delta_m \mathbf{r}_m\,,\ \mathbf{q}_R = \sum_{m=1}^{M} \beta_m \mathbf{r}_m \Rightarrow \mathbf{q}_R - \mathbf{q}_L = \sum_{m=1}^{M} (\beta_m - \delta_m)\mathbf{r}_m = \sum_{m=1}^{M} a_m \mathbf{r}_m\,. \tag{16}$$

The complete derivation can be found for instance in [72]. Obviously, the solution of (14) is self-similar, i.e., $\mathbf{q}(x,t) \equiv \mathbf{v}(x/t)$. In particular, $\mathbf{q}(0,t) = const.$ holds true for $t \in \mathbb{R}^+$. Hence, the flux $\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R) := \mathbf{f}(\mathbf{q}(0,t)) = \mathbf{A}\mathbf{q}(0,t)$ can easily be evaluated for all times t for the exact solution (15) as

$$\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R) = \mathbf{A}\mathbf{q}_L + \sum_{\lambda_m<0} a_m\lambda_m\mathbf{r}_m = \mathbf{A}\mathbf{q}_R - \sum_{\lambda_m\geq 0} a_m\lambda_m\mathbf{r}_m = \sum_{\lambda_m\geq 0} \delta_m\lambda_m\mathbf{r}_m + \sum_{\lambda_m<0} \beta_m\lambda_m\mathbf{r}_m\,. \tag{17}$$

We introduce the notations

$$\begin{array}{lll} \mathbf{\Lambda}^+ := \mathrm{diag}(\lambda_1^+,\ldots,\lambda_M^+), & \lambda_m^+ = \max(\lambda_m,0) = \frac{1}{2}(\lambda_m + |\lambda_m|) & \text{for all } m = 1,...,M\,, \\ \mathbf{\Lambda}^- := \mathrm{diag}(\lambda_1^-,\ldots,\lambda_M^-), & \lambda_m^- = \min(\lambda_m,0) = \frac{1}{2}(\lambda_m - |\lambda_m|) & \text{for all } m = 1,...,M \end{array} \tag{18}$$

and $\mathbf{A}^+ := \mathbf{R}\,\mathbf{\Lambda}^+\,\mathbf{R}^{-1}$, $\mathbf{A}^- := \mathbf{R}\,\mathbf{\Lambda}^-\,\mathbf{R}^{-1}$ with $\mathbf{A} = \mathbf{A}^+ + \mathbf{A}^-$,$|\mathbf{A}| = \mathbf{A}^+ - \mathbf{A}^-$ and write expression (17) with these definitions in short as

$$\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R) = \mathbf{A}\mathbf{q}_L + \mathbf{A}^-\Delta\mathbf{q} = \mathbf{A}\mathbf{q}_R - \mathbf{A}^+\Delta\mathbf{q} = \mathbf{A}^+\mathbf{q}_L + \mathbf{A}^-\mathbf{q}_R\,. \tag{19}$$

Further on, summation yields the useful expression

$$\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R) = \frac{1}{2}\left(\mathbf{A}\mathbf{q}_L + \mathbf{A}\mathbf{q}_R - |\mathbf{A}|\Delta\mathbf{q}\right)\,. \tag{20}$$

A numerical scheme for Eq. (14) that naturally considers the characteristic information can be constructed by assuming a FV discretization as introduced in Section 1.2 with cell values $\mathbf{Q}_j^i, j \in \mathbb{Z}, i \in \mathbb{R}_0^+$ and by solving the Riemann initial-value problem between two neighboring cells in every time step. We choose $\mathbf{Q}_j^i$ as $\mathbf{q}_L$ and $\mathbf{Q}_{j+1}^i$ as $\mathbf{q}_R$ and introduce the notation $\Delta\mathbf{Q}_{j+1/2}^i = \mathbf{Q}_{j+1}^i - \mathbf{Q}_j^i$. The numerical flux function then reads

$$\mathbf{F}(\mathbf{Q}^i) = \mathbf{F}(\mathbf{Q}_j^i, \mathbf{Q}_{j+1}^i) = \mathbf{A}\mathbf{Q}_j^i + \mathbf{A}^+\Delta\mathbf{Q}_{j+1/2}^i = \mathbf{A}\mathbf{Q}_{j+1}^i - \mathbf{A}^-\Delta\mathbf{Q}_{j+1/2}^i \,. \tag{21}$$

We insert these numerical fluxes into Eq. (8) and obtain the FV upwind scheme for linear systems

$$\mathbf{Q}_j^{i+1} = \mathbf{Q}_j^i - \frac{\Delta t}{\Delta x}\left(\mathbf{F}(\mathbf{Q}_j^i, \mathbf{Q}_{j+1}^i) - \mathbf{F}(\mathbf{Q}_{j-1}^i, \mathbf{Q}_j^i)\right) = \mathbf{Q}_j^i - \frac{\Delta t}{\Delta x}\left(\mathbf{A}^-\Delta\mathbf{Q}_{j+1/2}^i + \mathbf{A}^+\Delta\mathbf{Q}_{j-1/2}^i\right) \,. \tag{22}$$

Scheme (22) is *Godunov's Method* for Eq. (14) and of first-order accuracy [72]. The linear upwind scheme (22) is stable under the Courant-Friedrichs-Levy (CFL) condition [49, 72, 111]

$$C_{\text{CFL}}^{lin} := |\lambda_m|\frac{\Delta t}{\Delta x} \le 1 \,, \quad \text{for all } m = 1, \ldots, M. \tag{23}$$

In general, Godunov's Method requires the exact solution $\mathbf{q}(x,t)$ of the Riemann Problem (RP) between $\mathbf{q}_L$ and $\mathbf{q}_R$, at least for $x = 0$. Yet, the values used are only approximations on a finite grid. Satisfying results can often also be obtained if the intermediate Riemann problems are solved approximately. Schemes utilizing an approximate Riemann solver within Godunov's Method are said to be of *Godunov-type*.

In case of nonlinear hyperbolic systems even the approximate solution of the RP can be a very challenging task, especially for complex equations of state. Therefore, methods for nonlinear systems usually avoid the evaluation of the intermediate state and try to approximate the flux at $x = 0$ directly on the basis of upwind directions of the neighboring values $\mathbf{q}_L$ and $\mathbf{q}_R$. Two different approaches here are *flux-vector splitting* (FVS) and *flux-difference splitting* (FDS). FDS methods are of Godunov-type. They utilize a suitable linearization of $\mathbf{f}(\mathbf{q})$ on the basis of $\mathbf{q}_L$ and $\mathbf{q}_R$ and solve the linear RP as described. FVS methods are simpler and identify upwind directions separately for $\mathbf{q}_L$ and $\mathbf{q}_R$.

1.3.2. *Flux-vector splitting approach*

The FVS approach requires a splitting of $\mathbf{f}(\mathbf{q})$ into two components $\mathbf{f}^+(\mathbf{q})$ and $\mathbf{f}^-(\mathbf{q})$, such that the equation

$$\mathbf{f}(\mathbf{q}) = \mathbf{f}^+(\mathbf{q}) + \mathbf{f}^-(\mathbf{q}) \tag{24}$$

is satisfied under the restriction that the eigenvalues $\hat{\lambda}_m^+$ and $\hat{\lambda}_m^-$ of the split Jacobian matrices

$$\hat{\mathbf{A}}^+(\mathbf{q}) = \frac{\partial \mathbf{f}^+(\mathbf{q})}{\partial \mathbf{q}} \,, \quad \hat{\mathbf{A}}^-(\mathbf{q}) = \frac{\partial \mathbf{f}^-(\mathbf{q})}{\partial \mathbf{q}} \tag{25}$$

fulfill the conditions $\hat{\lambda}_m^+ \ge 0$ and $\hat{\lambda}_m^- \le 0$ for all $m = 1, \ldots, M$. Further on, the splitting is required to reproduce regular upwinding, i.e.,

$$\begin{array}{lllllllll} \mathbf{f}^+(\mathbf{q}) & = & \mathbf{f}(\mathbf{q})\,, & \mathbf{f}^-(\mathbf{q}) & = & \mathbf{0} & \text{if} & \lambda_m \ge 0 & \text{for all} \quad m = 1, \ldots, M \,, \\ \mathbf{f}^+(\mathbf{q}) & = & \mathbf{0}\,, & \mathbf{f}^-(\mathbf{q}) & = & \mathbf{f}(\mathbf{q}) & \text{if} & \lambda_m \le 0 & \text{for all} \quad m = 1, \ldots, M \,. \end{array} \tag{26}$$

The FVS approach then approximates the unknown intermediate flux $\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R)$ by

$$\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R) = \mathbf{f}^+(\mathbf{q}_L) + \mathbf{f}^-(\mathbf{q}_R) \,. \tag{27}$$

1.3.3. *Flux-difference splitting approach*

The FDS approach uses an approximate Riemann solver to calculate an approximation to the unknown intermediate flux $\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R)$. Instead of the RP between $\mathbf{q}_L$ and $\mathbf{q}_R$ for the original, possibly nonlinear equation, a RP with the same initial data for the modified conservation law

$$\frac{\partial \bar{\mathbf{q}}}{\partial t} + \frac{\partial \bar{\mathbf{f}}(\bar{\mathbf{q}})}{\partial x} = \mathbf{0} \tag{28}$$

with a linear flux function $\bar{\mathbf{f}}(\bar{\mathbf{q}}) = \hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)\bar{\mathbf{q}}$ is solved. Herein, $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ denotes a suitable constant Jacobian, chosen with respect to the initial data. The RP for this modified linear conservation law can easily be solved exactly (see Section 1.3.1), but care must be taken in the approximation of the intermediate flux $\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R)$. The obvious choice $\bar{\mathbf{F}}(\mathbf{q}_L, \mathbf{q}_R)$ according to Eq. (17) leads to a scheme that is inconsistent with the original conservation law. The scheme would satisfy a different discrete conservation property and hence converge toward the wrong weak solution. Using Eqs. (16-17) plus a conservation argument, cf. [32], yields

$$\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R) = \mathbf{f}(\mathbf{q}_L) + \sum_{\hat{\lambda}_m < 0} a_m \hat{\lambda}_m \hat{\mathbf{r}}_m = \mathbf{f}(\mathbf{q}_R) - \sum_{\hat{\lambda}_m \geq 0} a_m \hat{\lambda}_m \hat{\mathbf{r}}_m \ , \tag{29}$$

or in terms of the notations of Section 1.3.1

$$\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R) = \mathbf{f}(\mathbf{q}_L) + \hat{\mathbf{A}}^- \Delta \mathbf{q} = \mathbf{f}(\mathbf{q}_R) - \hat{\mathbf{A}}^+ \Delta \mathbf{q} \tag{30}$$

$$= \frac{1}{2}\left(\mathbf{f}(\mathbf{q}_L) + \mathbf{f}(\mathbf{q}_R) - |\hat{\mathbf{A}}|\Delta \mathbf{q}\right) . \tag{31}$$

If these flux approximations are used within scheme (22), the update formula becomes

$$\mathbf{Q}_j^{i+1} = \mathbf{Q}_j^i - \frac{\Delta t}{\Delta x}\left(\hat{\mathbf{A}}^-(\mathbf{Q}_j^i, \mathbf{Q}_{j+1}^i)\Delta \mathbf{Q}_{j+\frac{1}{2}}^i + \hat{\mathbf{A}}^+(\mathbf{Q}_{j-1}^i, \mathbf{Q}_j^i)\Delta \mathbf{Q}_{j-\frac{1}{2}}^i\right) . \tag{32}$$

A necessary stability condition for scheme (32) is

$$C_{\text{CFL}}^{nl} := \max_{j \in \mathbb{Z}} |\hat{\lambda}_{m,j+\frac{1}{2}}| \frac{\Delta t}{\Delta x} \leq 1 \ , \quad \text{for all } m = 1, \ldots, M. \tag{33}$$

The difficult task in the FDS approach is the derivation of a constant matrix $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ for each RP that approximates the original Jacobian appropriately. Roe suggested the following three properties for such a matrix [98]: (i) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ is diagonalizable with real eigenvalues, (ii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R) \to \dfrac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}$ smoothly as $\mathbf{q}_L, \mathbf{q}_R \to \mathbf{q}$, (iii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)\Delta \mathbf{q} = \mathbf{f}(\mathbf{q}_R) - \mathbf{f}(\mathbf{q}_L)$. The third property ensures the conservation of the resulting scheme and is just another form of Eq. (30). Obviously, scheme (32) can be implemented without explicit evaluation of $\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R)$. It suffices to calculate the *fluctuations*

$$\hat{\mathbf{A}}^-(\mathbf{q}_L, \mathbf{q}_R)\Delta \mathbf{q} = \sum_{\hat{\lambda}_m < 0} a_m \hat{\lambda}_m \hat{\mathbf{r}}_m \ , \quad \hat{\mathbf{A}}^+(\mathbf{q}_L, \mathbf{q}_R)\Delta \mathbf{q} = \sum_{\hat{\lambda}_m \geq 0} a_m \hat{\lambda}_m \hat{\mathbf{r}}_m \ . \tag{34}$$

1.4. **High-resolution methods**

The basic shock-capturing upwind schemes described previously are just of first order approximation accuracy. While this allows the construction of very reliable, monotonity-preserving schemes, first-order accurate schemes are computationally rather inefficient. Even when dynamic mesh adaptation is available, a high-resolution approach should be employed that achieves at least second-order accuracy in smooth solution regions by proper interpolation and gives an oscillation-free extrapolation near discontinuities.

1.4.1. *TVD schemes*

Since strictly monotone scheme are restricted to first order [57], the slightly relaxed concept of *total variation diminishing* (TVD) discretizations provides good guidance for the construction of high-resolution methods.

Definition 2. (TVD property). *A scheme $\mathbf{Q}_j^{l+1} = \mathcal{H}^{(\Delta t)}(\mathbf{Q}^i; j)$ is called total variation diminishing (TVD), if the property $TV(\mathbf{Q}^{i+1}) \le TV(\mathbf{Q}^i)$ is satisfied for all discrete sequences $\mathbf{Q}^i$. Herein, $TV(\mathbf{Q})$ denotes the discrete total variation, which is defined by $TV(\mathbf{Q}^i) := \sum_{j\in\mathbb{Z}} |\mathbf{Q}_{j+1}^i - \mathbf{Q}_j^i|$.*

TVD schemes maintain the property that no new extrema in x can be created. Local minima are non-decreasing, while local maxima are non-increasing, which is termed *monotonicity-preserving* [55]. Relaxed concepts are employed for the construction of (Weighted) Essentially Non-Oscillatory (WENO/ENO) schemes [105], the Flux-Corrected Transport (FCT) approach [88], and the Piecewise Parabolic Method (PPM) [28]. The book by Laney [68] gives an excellent overview of higher order schemes.

1.4.2. *MUSCL-Hancock method*

A method for the practical construction of second-order five-point TVD schemes, especially on the basis of Godunov-type upwind schemes, has been developed by Van Leer [116]. Profound descriptions of this technique, which is also called the *variable extrapolation approach* or *slope limiting* can be found in the books of Toro [111] and Hirsch [59]. In here, we just give the basic formulas.

In the MUSCL (Monotone Upwind Schemes for Conservation Laws) variable extrapolation method, the cell-wise constant approximation Q_j^i is replaced by a linear or quadratic interpolation $\tilde{Q}_j(x)$, $x \in [x_{j-1/2}, x_{j+1/2}]$ utilizing the three values Q_{j-1}^i, Q_j^i and Q_{j+1}^i. At the boundaries of cell j, the one-sided extrapolated and limited values read

$$\tilde{Q}^L_{j+\frac{1}{2}} = Q_j^n + \frac{1}{4}\left[(1-\omega)\,\Phi^+_{j-\frac{1}{2}}\Delta_{j-\frac{1}{2}} + (1+\omega)\,\Phi^-_{j+\frac{1}{2}}\Delta_{j+\frac{1}{2}}\right] , \tag{35}$$

$$\tilde{Q}^R_{j-\frac{1}{2}} = Q_j^n - \frac{1}{4}\left[(1-\omega)\,\Phi^-_{j+\frac{1}{2}}\Delta_{j+\frac{1}{2}} + (1+\omega)\,\Phi^+_{j-\frac{1}{2}}\Delta_{j-\frac{1}{2}}\right] , \tag{36}$$

with $\Delta_{j-1/2} = Q_j^n - Q_{j-1}^n$, $\Delta_{j+1/2} = Q_{j+1}^n - Q_j^n$. Note that this interpolation is second-order accurate for $\omega = 1/3$ and first-order accurate for all other values [59]; discrete conservation is preserved only for $\omega = 0$. In Eqs. (35-36), the factors $\Phi^{\pm}_{j\mp1/2}$ are

$$\Phi^+_{j-\frac{1}{2}} := \Phi\left(r^+_{j-\frac{1}{2}}\right) , \quad \Phi^-_{j+\frac{1}{2}} := \Phi\left(r^-_{j+\frac{1}{2}}\right) \quad \text{with} \quad r^+_{j-\frac{1}{2}} := \frac{\Delta_{j+\frac{1}{2}}}{\Delta_{j-\frac{1}{2}}} , \quad r^-_{j+\frac{1}{2}} := \frac{\Delta_{j-\frac{1}{2}}}{\Delta_{j+\frac{1}{2}}} , \tag{37}$$

where the choice of the *slope limiters* function, Φ, to ensure the TVD property is the crucial step. Very reliable are the *Minmod*-type limiter

$$\Phi(r) = \max(0, \min(r, 1)) , \tag{38}$$

or the less diffusive *Van Albada*-type limiter [115]

$$\Phi(r) = \max\left(0, \frac{r^2 + r}{1 + r^2}\right) . \tag{39}$$

The utilization of interpolated values in the numerical flux approximation of Eq. (22) alone gives a scheme that is usually unstable. A stable, second-order accurate scheme requires a higher-order temporal integration. Very effective for MUSCL-type methods is the Hancock approach [118], which constructs extrapolated values first and evolves these values by $\Delta t/2$ with a midpoint rule, i.e.,

$$\bar{Q}^L_{j+\frac{1}{2}} = \tilde{Q}^L_{j+\frac{1}{2}} - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(f(\tilde{Q}^L_{j+\frac{1}{2}}) - f(\tilde{Q}^R_{j-\frac{1}{2}})\right) , \quad \bar{Q}^R_{j-\frac{1}{2}} = \tilde{Q}^R_{j-\frac{1}{2}} - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(f(\tilde{Q}^L_{j+\frac{1}{2}}) - f(\tilde{Q}^R_{j-\frac{1}{2}})\right) , \tag{40}$$

before they are utilized for the flux approximation, for instance, at $x_{j+1/2}$ as $F(\bar{Q}^L_{j+1/2}, \bar{Q}^R_{j+1/2})$. For the proof of the TVD property of the MUSCL-Hancock approach, see [111]. Note that the theoretical basis is strictly sound only for scalar problems with $\mathbf{s} \equiv \mathbf{0}$.

1.4.3. *Wave Propagation Method*

The Wave Propagation Method is a second-order accurate extension for FDS schemes that utilize the fluctuations (34) instead of the numerical fluxes, see [69, 73–75]. Utilizing the *waves* $\mathcal{W}_m := a_m \hat{\mathbf{r}}_m$ and introducing the notation

$$\mathcal{A}^-\Delta = \sum_{\hat{\lambda}_m<0} \hat{\lambda}_m \, \mathcal{W}_m \ , \quad \mathcal{A}^+\Delta = \sum_{\hat{\lambda}_m \ge 0} \hat{\lambda}_m \, \mathcal{W}_m \ . \tag{41}$$

instead of (34), the one-dimensional Wave Propagation Method reads

$$\mathbf{Q}^{l+1} = \mathbf{Q}^l_j - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^-\Delta_{j+\frac12} + \mathcal{A}^+\Delta_{j-\frac12}\right) - \frac{\Delta t}{\Delta x}\left(\tilde{\mathbf{F}}_{j+\frac12} - \tilde{\mathbf{F}}_{j-\frac12}\right) \ . \tag{42}$$

Herein, $\tilde{\mathbf{F}}_{j\pm1/2}$ denote additional terms that are necessary to achieve second-order accuracy in smooth regions of the solution. The basic second-order scheme of the Wave Propagation Method is the Lax-Wendroff scheme. At each cell interface, $\tilde{\mathbf{F}}_{j+1/2}$ is uniquely defined by the difference between the second-order Lax-Wendroff flux and the first-order upwind flux (32). This difference reads

$$\tilde{\mathbf{F}}_{j+\frac12} = \frac12 |\mathcal{A}| \left(1 - \frac{\Delta t}{\Delta x}|\mathcal{A}|\right) \Delta_{j+\frac12} = \frac12 \sum_{m=1}^{M} |\hat{\lambda}^m_{j+\frac12}| \left(1 - \frac{\Delta t}{\Delta x}\right) |\hat{\lambda}^m_{j+\frac12}| \, \mathcal{W}^m_{j+\frac12} \ . \tag{43}$$

In order to achieve a TVD scheme, limited waves $\tilde{\mathcal{W}}^m_{j+1/2}$ are used instead of the original waves. The wave limiting is calculated by $\tilde{\mathcal{W}}^m_{j+\frac12} = \Phi(\Theta^m_{j+\frac12})\,\mathcal{W}^m_{j+\frac12}$ with

$$\Theta^m_{j+\frac12} = \begin{cases} a^m_{j-\frac12}/a^m_{j+\frac12} \ , & \hat{\lambda}^m_{j+\frac12} \ge 0 \ , \\ a^m_{j+\frac32}/a^m_{j+\frac12} \ , & \hat{\lambda}^m_{j+\frac12} < 0 \ . \end{cases} \tag{44}$$

Typical slope limiter functions, such as (38) or (39), can also be applied as wave limiters [73]. Setting $\tilde{\mathcal{A}}^\pm\Delta_{j\pm1/2,k} := \mathcal{A}^\pm\Delta_{j\pm1/2,k} + \tilde{\mathbf{F}}_{j\pm1/2,k}$ and denoting the limited fluctuations in the x_2-direction by $\tilde{\mathcal{B}}^\pm\Delta_{j,k\pm1/2}$, a truly two-dimensional Wave Propagation Method can be formulated as [69]

$$\begin{aligned}\mathbf{Q}^{i+1}_{jk} = \mathbf{Q}^i_{jk} - \frac{\Delta t}{\Delta x}\Bigg(&\tilde{\mathcal{A}}^-\Delta_{j+\frac12,k} - \frac12\frac{\Delta t}{\Delta y}\left[\mathcal{A}^-\tilde{\mathcal{B}}^-\Delta_{j+1,k+\frac12} + \mathcal{A}^-\tilde{\mathcal{B}}^+\Delta_{j+1,k-\frac12}\right] + \\ &\tilde{\mathcal{A}}^+\Delta_{j-\frac12,k} - \frac12\frac{\Delta t}{\Delta y}\left[\mathcal{A}^+\tilde{\mathcal{B}}^-\Delta_{j-1,k+\frac12} + \mathcal{A}^+\tilde{\mathcal{B}}^+\Delta_{j-1,k-\frac12}\right]\Bigg) \\ -\frac{\Delta t}{\Delta y}\Bigg(&\tilde{\mathcal{B}}^-\Delta_{j,k+\frac12} - \frac12\frac{\Delta t}{\Delta x}\left[\mathcal{B}^-\tilde{\mathcal{A}}^-\Delta_{j+\frac12,k+1} + \mathcal{B}^-\tilde{\mathcal{A}}^+\Delta_{j-\frac12,k+1}\right] + \\ &\tilde{\mathcal{B}}^+\Delta_{j,k-\frac12} - \frac12\frac{\Delta t}{\Delta x}\left[\mathcal{B}^+\tilde{\mathcal{A}}^-\Delta_{j+\frac12,k-1} + \mathcal{B}^+\tilde{\mathcal{A}}^+\Delta_{j-\frac12,k-1}\right]\Bigg) \ ,\end{aligned} \tag{45}$$

with stability condition

$$C^{wp}_{\text{CFL}} := \max_{j,k\in\mathbb{Z}} \left\{ |\hat{\lambda}_{m,j+\frac12,k}|\frac{\Delta t}{\Delta x_1}, |\hat{\lambda}_{m,j,k+\frac12}|\frac{\Delta t}{\Delta x_2} \right\} \le 1 \ , \quad \text{for all } m = 1,\ldots,M. \tag{46}$$

In combination with a a *linearized* Riemann solver (cf. Section 1.5.3), Eq. (45) gives a second-order accurate scheme for two-dimensional nonlinear systems. A canonical problem of the original Wave Propagation Method is that the wave limiting approach used can produce serious overshoots in derived quantities, such as the hydrodynamic pressure in case of Euler equations, especially when physical units are used. If a system in conservation form is solved, usage of the robust slope-limited MUSCL-Hancock approach and application of the multi-dimensional correction terms of (29) can be combined by computing the second-order accurate fluctuations in the normal direction, e.g.,

$$\tilde{\mathcal{A}}^- \Delta_{j+1/2,k} := \mathbf{F}(\bar{\mathbf{Q}}^L_{j+1/2,k}, \bar{\mathbf{Q}}^R_{j+1/2,k}) - \mathbf{f}(\mathbf{Q}^i_{jk}) \tag{47}$$

according to (30). We have found that the resulting scheme is very reliable, yet preserves the multi-dimensional accuracy of Eq. (45) entirely.

1.5. **Euler equations**

As a practically relevant example for a nonlinear hyperbolic conservation law without source term we consider the Euler equations of gas dynamics. Most numerical methods have been initially proposed for the Euler equations and we provide two first-order accurate upwind operators $\mathcal{X}_1^{(\Delta t)}$ in here from which the operators for the other spatial directions can easily be derived by canonically exchanging the velocities u_n and the index n.

1.5.1. *Governing equations for an ideal gas*

Using the vector of state $\mathbf{q} = [\rho_1, \rho u_1, \ldots, \rho u_d, \rho E]^T$ and the flux functions

$$\mathbf{f}_n(\mathbf{q}) = [\rho u_n, \rho u_1 u_n + \delta_{1n} p, \ldots, \rho u_d u_n + \delta_{dn} p, u_n(\rho E + p)]^T \quad \text{for } n = 1, \ldots, d, \tag{48}$$

we write the governing equations for the flow of an inviscid ideal gas in terms of the notations of Section 1.1 as

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ \rho u_1 \\ \vdots \\ \rho u_d \\ \rho E \end{bmatrix} + \sum_{n=1}^{d} \frac{\partial}{\partial x_n}\begin{bmatrix} \rho u_n \\ \rho u_1 u_n + \delta_{1n} p \\ \vdots \\ \rho u_d u_n + \delta_{dn} p \\ u_n(\rho E + p) \end{bmatrix} = 0\,. \tag{49}$$

Herein, $\rho > 0$ is the density, $E > 0$ is the specific total energy, and we denote the n-th component of the velocity vector $\mathbf{u} = (u_1, \ldots, u_d)^T$ by u_n, and δ_{jn} is the Kronecker-Symbol. The total specific enthalpy is $H := E + \frac{p}{\rho}$, and internal specific energy and enthalpy can be computed as $e = E - \frac{1}{2}\mathbf{u}^2$ and $h = H - \frac{1}{2}\mathbf{u}^2$, respectively. In order to close system (49), an equation of state for the hydrostatic pressure p is required. For a polytropic ideal gas, the equation of state is the explicit expression

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho \mathbf{u}^2\right), \tag{50}$$

with γ denoting the adiabatic constant. For any ideal gas, the speed of sound is given by

$$c^2 = \frac{\partial p}{\partial \rho} = \gamma \frac{p}{\rho}\,. \tag{51}$$

Using Eqs. (50-51), it is a straightforward exercise to verify that the Jacobian $\mathbf{A}_n = \partial \mathbf{f}_n / \partial \mathbf{q}$ of the flux function (48) has a complete eigendecomposition with $d + 2$ real eigenvalues and that system (49) thereby satisfies Def. 1 for all admissible states. The eigenvalues of $\mathbf{A}_n$ are $\lambda_1 = u_n - c$, $\lambda_2 = \cdots = \lambda_{d+1} = u_n$, and $\lambda_{d+2} = u_n + c$.

1.5.2. *The Van Leer flux vector splitting for Euler equations*

As a practically relevant example for a flux-vector splitting scheme (cf. Section 1.3.2), we provide the FVS by Van Leer [117]. A complete derivation can also be found, for instance, in [68]. The Van Leer FVS has been derived specifically for the Euler equations of a polytropic gas. It circumvents typical problems of simpler splittings, adds comparably little numerical diffusion, and additionally, it is very robust. For instance, in the x_1-direction the split functions read

$$\mathbf{f}_1^{\pm}(\mathbf{q}) = \pm\frac{\rho}{4c}(u_1 \pm c)^2 \left[1, \frac{(\gamma-1)u_1 \pm 2c}{\gamma}, u_2, \ldots, u_d, \frac{((\gamma-1)u_1 \pm 2c)^2}{2(\gamma^2-1)} + \frac{\mathbf{u}^2 - u_1^2}{2}\right]^T . \tag{52}$$

Equation (52) is explicitly constructed for $-c \le u_1 \le c$. For $|u_1| > c$, the relations (26) have to be applied. Note that the necessary stability condition is

$$C_{\mathrm{CFL}}^{VL} := \max_{j\in\mathbb{Z}} \left[(|u_{1,j}| + c_j)\,\Pi_j\right] \frac{\Delta t}{\Delta x} \le 1 \quad \text{with} \quad \Pi_j = \begin{cases} \dfrac{\gamma+3}{2\gamma + u_{1,j}(3-\gamma)/c_j} & \text{if } |u_{1,j}| < c_j \;, \\ 1 & \text{otherwise .} \end{cases} \tag{53}$$

1.5.3. *The approximate Riemann solver of Roe for Euler equations*

The archetype of a flux-difference splitting method for a nonlinear hyperbolic system (cf. Section 1.3.3) is the approximate Riemann solver of Roe for the Euler equations of a polytropic gas, Eqs. (49-50). At each cell interface, a locally constant Jacobian $\hat{\mathbf{A}}_n(\mathbf{q}_L, \mathbf{q}_R)$ is sought, enforcing the properties (i-iii) given on page 103. The challenging property is (iii) and using the definitions $\Delta v := v_r - v_l$ and the specific average

$$\hat{v} := \frac{\sqrt{\rho_l}v_l + \sqrt{\rho_r}v_r}{\sqrt{\rho_l} + \sqrt{\rho_r}} \tag{54}$$

for u_n and H, the average of the density $\hat{\rho} := \sqrt{\rho_l \rho_r}$, and the averaged speed of sound

$$\hat{c} := \left((\gamma-1)(\hat{H} - \frac{1}{2}\hat{\mathbf{u}}^2)\right)^{1/2} , \tag{55}$$

the Roe scheme for Euler equations evaluates the waves $\mathcal{W}_m := a_m \hat{\mathbf{r}}_m$ (cf. Eqs. (32-34)), for instance, in the x_1-direction as

$$a_{1,d+2} = \frac{\Delta p \mp \hat{\rho}\hat{c}\Delta u_1}{2\hat{c}^2} \;,\; \hat{\mathbf{r}}_{1,d+2} = \left[1, \hat{u}_1 \mp \hat{c}, \hat{u}_2, \ldots, \hat{u}_d, \hat{H} \mp \hat{u}_1\hat{c}\right]^T , \tag{56}$$

$$a_2 = \Delta\rho - \frac{\Delta p}{\hat{c}^2} \;,\; \hat{\mathbf{r}}_2 = \left[1, \hat{u}_1, \hat{u}_2, \ldots \hat{u}_d, \frac{\hat{\mathbf{u}}^2}{2}\right]^T , \tag{57}$$

$$a_{n+1} = \hat{\rho}\Delta u_n \;,\; \hat{\mathbf{r}}_{n+1} = [0, \ldots, 0, \delta_{2n}, \ldots, \delta_{dn}, \hat{u}_n]^T \quad \text{for } n = 2, \ldots, d, \tag{58}$$

with the eigenvalues $\hat{\lambda}_{1,d+2} = \hat{u}_1 \mp \hat{c}$ and $\hat{\lambda}_{n+1} = \hat{u}_1$ for $n = 1, \ldots, d$. A comprehensive derivation is given in [111]. The approximate Riemann solver of Roe adds very little numerical diffusion and is thereby an excellent building block for high-resolution schemes; however, it does not satisfy a maximum principle and can produce states with $\rho \le 0$ or $E \le 0$ near vacuum that are not admissible. Further on, a linearized Riemann solver by construction neglects smooth rarefaction waves intrinsic to the Euler equations that can occur instead of discontinuous shock waves in the genuinely nonlinear characteristic fields associated to the eigenvalues $\lambda_{1,d+2}$. The linearized inter-cell flux approximation becomes incorrect only for the special case of a transonic rarefaction wave, yet the resulting entropy violation needs to be avoided to make the solver generally applicable. Commonly used entropy corrections are the identification of transonic rarefaction fans by the Lax entropy conditions and subsequent explicit flux correction [56] or the addition of an appropriate amount of numerical viscosity near sonic points [55]. The latter can be accomplished effectively by replacing $\hat{\lambda}_m$ in the flux approximation, Eq. (29), by

$$|\bar{\lambda}_m| = \begin{cases} |\hat{\lambda}_m| \, , & |\hat{\lambda}_m| \geq 2\eta \, , \\ |\hat{\lambda}_m^2|/(4\eta) + \eta \, , & |\hat{\lambda}_m| < 2\eta \, . \end{cases} \tag{59}$$

A natural choice for the parameter η for Euler equations is $\eta = \frac{1}{2}(|u_{1,r} - u_{1,l}| + |c_r - c_l|)$, cf. [101]. In one space dimension, Eq. (59) only needs to be applied to $m = 1, d + 2$; two- and three-dimensional simulations with strong grid-aligned shocks require a suitable extension of Eq. (59) to all other fields, cf. Section 3.3.3.

1.6. **Meshes and adaptation**

Solutions of the Euler equations, Eq. (49), often involve a wide range of different scales. A common requirement is usually to increase resolution along discontinuities, while in smooth solution regions resolution can be reduced. Efficient implementations of the described FV schemes therefore have to utilize non-uniform grids. Various techniques to adapt the discretization dynamically to the solution have been developed during the last two decades.

1.6.1. *Unstructured approach*

Unstructured triangulations offer superior geometrical flexibility. The coordinates of all vertices have to be stored explicitly and the basic discretization is intrinsically non-uniform (cf. Fig. 1). Consequently, existing implementations can relatively easily be supplemented with dynamical adaptation. Cells that have been flagged for refinement are simply replaced by finer ones and the numerical solution is advanced on the entire grid simultaneously. A coarsening step is necessary to recombine fine cells. For time-explicit FV schemes this simple strategy can be inefficient as it requires a global time step to satisfy a CFL-type condition for the smallest cell. Further on, unstructured triangulations are usually implemented with cell-based data structures that store all neighborhood relationships explicitly. Without complicate re-numbering and re-arrangement of the vector of cells based on the mesh topology the memory access during computation is highly irregular and the performance on vector or super-scalar computers rather poor. Implementations on parallel computers with distributed memory have to solve complex load-balancing problems on the fly. In particular, appropriate synchronization regions (overlaps) with respect to the numerical stencil are difficult to compute. Freely available generic C++-libraries that support unstructured meshes on distributed memory machines are GrAL[1] (Grid Algorithms Library) by Berti [19] and DUNE[2] by Bastian et al. [9].

1.6.2. *Structured approach*

If geometric flexibility is only of secondary interest, the numerical scheme can be formulated on a logically rectangular (not necessarily Cartesian) mesh. Rectangular meshes allow optimizations that moderate some of the technical complexities of unstructured refinement techniques. A structured refinement strategy replaces or overlays a single coarse cell by a regular refinement block of r^d cells. For simplicity, the refinement factor r is often fixed and all successively generated refinement blocks can be accessed efficiently by utilizing a regular data tree (see Fig. 2). The data tree avoids explicit storage of parent- or child-relations and the use of a global integer coordinate system (cf. Section 2.4.2) allows an easy evaluation of neighborhood relationships.

In the case of time-explicit FV schemes the construction of time-space interpolated internal boundary conditions can be implemented with moderate expense allowing a successive time step refinement with factor r. A disadvantage of all types of structured refinement is that hanging nodes along the coarse-fine interfaces are unavoidable (cf. left sketch of Fig. 2).[3]

[1]http://gral.berlios.de

[2]www.dune-project.org

[3]A conforming closure is possible if unstructured cells are employed. Such a hybrid refinement strategy is used in the UG multigrid package of Bastian & Wittum [8]. Note that hybrid discretization techniques require implementations of the numerical

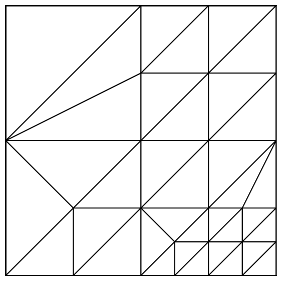


FIGURE 1. Unstructured refinement strategy.

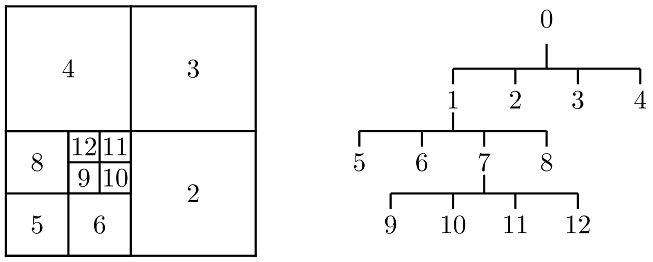


FIGURE 2. Mesh and corresponding local quad-tree of a structured mesh refinement strategy ($r = 2$, $d = 2$).

A recent freely available FV program for distributed memory machines that allows arbitrary refinement factors is the NASA code PARAMESH[4] by MacNeice et al. [78] that has also been used as the basis for the Flash code[5]. Another notable implementation of cell-based structured mesh refinement is the RAGE code [48].

Although the structured approach uses the available computer memory better than the unstructured technique, consecutive memory blocks of r^d cells are usually not large enough to fill the vector pipelines of modern super-computers satisfactory. Furthermore, a large number of small refinement blocks requires a large synchronization overhead. If the numerical scheme is implemented with ghost cells, the waste due to overlapping neighboring ghost cells may be considerable (see [86] for a detailed discussion). Such overlap can be eliminated completely if refinement blocks of arbitrary size are considered.

1.6.3. *Block-structured adaptive mesh refinement*

The structured adaptive mesh refinement technique (SAMR) for hyperbolic partial differential equations has been pioneered by Berger & Oliger [12, 16]. While the first approach utilized rotated refinement grids that required complicated conservative interpolation operations, SAMR denotes today especially the simplified variant of Berger & Collela [14] that only allows refinement patches aligned to the coarse-grid mesh. The efficiency of this simplified variant, in particular on vector and super-scalar computers, was demonstrated by Bell et al. [11].

Instead of replacing single cells by finer ones the SAMR method follows a patch-wise refinement strategy. Cells being flagged by various error indicators are clustered with a special algorithm (cf. Section 2.1.10) into rectangular boxes of appropriate size. They describe refinement regions geometrically and subgrids with the same refinement factor in all space directions and also in time are generated according to them. Refined subgrids are derived recursively from coarse-level grids and an entire hierarchy of successively embedded grid patches is thereby constructed, cf. Fig. 3. Like in the structured approach, only the implementation of the numerical scheme on a single rectangular grid is required. The adaptive algorithm calls this application-dependent routine automatically. Further on, it uses interpolation functions to transfer cell values between refined subgrids and their coarser parents appropriately. It is important to note, that refined grids overlay the coarser subgrids from which they have been created. The numerical solution on a particular level is first of all advanced independently. Values of cells covered by refined subgrids are overwritten by averaged fine-grid values subsequently. The superfluous work on the coarse grid is deemed negligible compared to the computational costs for integrating the superimposed fine grids. Unlike the refinement technique of Section 1.6.2, that only allows for one parent cell, the SAMR method requires a general data tree as arbitrary parent- and child-relations need to be considered. In Fig. 3, this generality is expressed by grid $G_{2,2}$ that overlays two parents.

---

scheme on unstructured and logically rectangular meshes and are usually more complex that an accurate modification of the numerical stencil at hanging nodes.

[4]http://sourceforge.net/projects/paramesh. Generic library that supports parallel time-explicit and implicit FV methods.

[5]http://flash.uchicago.edu/website/home. Parallel adaptive code for solving the magneto-hydrodynamic equations with self-gravitation.

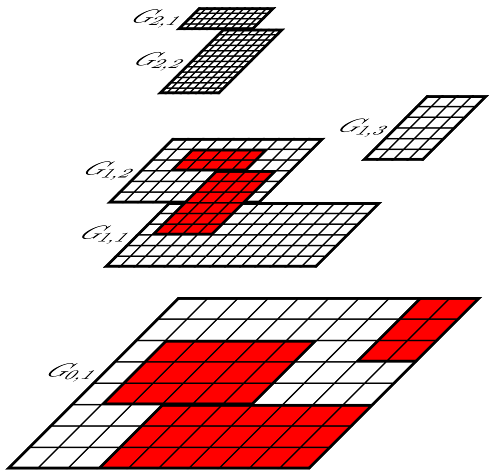


Figure 3. The SAMR method creates a hierarchy of rectangular subgrids.

Replacing coarse-cell values by averaged fine-grid values modifies the numerical stencil on the coarse grid. In general the important property of conservation is lost. A flux correction replacing the coarse-grid flux at the affected side of a neighboring cell by accumulated fine-grid fluxes is necessary to ensure conservation. In the SAMR method, this *conservative fixup* is usually implemented as a correction pass. Like in the structured approach, described in the previous subsection, hanging nodes additionally have to be considered in two and three space dimensions. The correction procedure is explained in detail in Section 2.1.6.

Various implementations of the SAMR method for single-processor computers have been developed [29,46,86]. Freely available are the two-dimensional codes AMRClaw[6] by Berger & LeVeque [15] and Amrita[7] by Quirk. The extension of the SAMR approach to parallel computers with shared memory is rather straightforward: It suffices to parallelize the loop updating the grids, cf. [11]. On distributed memory machines, however, communication costs cannot be neglected and parallelization strategies are significantly more complex. Freely available SAMR software systems, that all support three-dimensional FV computations and MPI-based distributed memory parallelism are the very mature SAMRAI[8] (Structured Adaptive Mesh Refinement Application Infrastructure) [60], Chombo[9] by Colella et al., and our own software system AMROC[10] (Adaptive Mesh Refinement in Object-oriented C++) [32, 33]. Notable are further the Overture system[11] by Henshaw et al. [23], that provides tools for solving PDEs on overlapping SAMR meshes, the BoxLib[12] collection of C++ classes by Bell et al. [97], and the Uintah[13] SAMR code for the simulation of accidental fires and explosions.

[6]http://www.clawpack.org. Serial Fortran 77 code for the explicit two-dimensional Wave Propagation Method [74].

[7]http://www.amrita-cfd.org. Supports serial two-dimensional explicit FV schemes. Embedded boundary algorithm for complex geometry representation available.

[8]https://computation.llnl.gov/casc/SAMRAI. Generic parallel framework that supports explicit FV schemes directly, implicit methods through an interface to the Hypre package (https://computation.llnl.gov/casc/linear_solvers). Mapped geometry and some embedded boundary support.

[9]https://seesar.lbl.gov/anag/chombo. Redesign of BoxLib, both parallel explicit and implicit FV algorithms demonstrated at large scale. Some embedded boundary support.

[10]V2.0 available at http://www.cacr.caltech.edu/asc. Generic parallel support for explicit FV schemes, embedded boundaries and fluid-structure coupling. Large number of provided schemes. Implicit multigrid algorithms prototyped but not distributed yet. V1.0 is still available at http://amroc.sourceforge.net but demonstrates just the Wave Propagation Method (cf. Section 1.4.3) in parallel on strictly Cartesian meshes.

[11]https://computation.llnl.gov/casc/Overture. Explicit and implicit schemes supported on geometrically complex overlapping adaptive meshes. Special mesh generation tools provided.

[12]https://ccse.lbl.gov/Software. Explicit and implicit FV schemes supported but codes are not available to the public anymore.

[13]http://www.uintah.utah.edu. Implements explicit ICE scheme and Material Point Method to allow for fluid-structure coupling.

2. SAMR for hyperbolic problems

2.1. **Serial algorithm**

In this chapter we define the SAMR method exactly. Like in Section 1.2 we concentrate (without loss of generality) on the two-dimensional case. For simplicity, we assume that the numerical scheme is a conservative time-explicit FV method in two space dimensions, based on the update formula (8) with $s \equiv 0$ and $d = 2$, i.e.,

$$\mathbf{Q}_{jk}^{i+1} = \mathbf{Q}_{jk}^{i} - \frac{\Delta t}{\Delta x_1}\left(\mathbf{F}^1_{j+\frac{1}{2},k}(\mathbf{Q}^i) - \mathbf{F}^1_{j-\frac{1}{2},k}(\mathbf{Q}^i)\right) - \frac{\Delta t}{\Delta x_2}\left(\mathbf{F}^2_{j,k+\frac{1}{2}}(\mathbf{Q}^i) - \mathbf{F}^2_{j,k-\frac{1}{2}}(\mathbf{Q}^i)\right), \tag{60}$$

that is formulated on a rectangular Cartesian grid G. We denote the update operator (60) by $\mathcal{H}^{(\cdot)}$ and assume that its implementation requires $s \geq 1$ auxiliary cells (ghost cells) around G to define discrete boundary conditions.

2.1.1. *The grid hierarchy*

Let the SAMR hierarchy consist of a sequence of levels $l = 0, \dots, l_{\max}$. Analogous to Section 1.2.1 we define a discretization of the computational domain on each level l with successively finer mesh widths $\Delta x_{n,l}$, $n = 1, \dots, d$ with $d = 2$ and a refined time step Δt_l. All mesh widths of Level $l > 0$ are set to be r_l-times smaller than those of level $l-1$. With $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$ we define $\Delta t_l := \Delta t_{l-1}/r_l$ and $\Delta x_{n,l} := \Delta x_{n,l-1}/r_l$ for all $n = 1, \dots, d$. Therefore, the ratios

$$\frac{\Delta t_l}{\Delta x_{n,l}} = \frac{\Delta t_{l-1}}{\Delta x_{n,l-1}} = \cdots = \frac{\Delta t_0}{\Delta x_{n,0}} \quad \text{for all } n = 1, \dots, d \tag{61}$$

remain constant on all levels and a time-explicit FV scheme can expected to be stable under a CFL-condition on all grids of the hierarchy [14].

2.1.2. *Topology*

With the notations of Section 1.2.1 we define the domain of the m-th grid on level l by

$$G_{l,m} \;:=]x_{1,l}^{j-1/2}, x_{1,l}^{j+\mu_1-1/2}[\times]x_{2,l}^{k-1/2}, x_{2,l}^{k+\mu_2-1/2}[\;. \tag{62}$$

It has $\mu_1 \cdot \mu_2$ FV cells and corresponds to the interior grid of Fig. 4. The boundary of $G_{l,m}$ is $\partial G_{l,m}$ and $\bar{G}_{l,m} = G_{l,m} \cup \partial G_{l,m}$ is its hull. With a total number of M_l grids on level l the domain of the entire level is

$$G_l := \bigcup_{m=1}^{M_l} G_{l,m} \quad \text{with} \quad G_{l,m} \cap G_{l,n} = \emptyset \;\text{ for } m \neq n \;, \tag{63}$$

where we have assumed that the grids $G_{l,m}$ do not overlap. The problem domain $G_0 = \bigcup_m G_{0,m}$ does not need to be a single grid. In order to specify the setting of ghost cell values exactly and to derive geometric relations between the grids of different levels we introduce enlarged grid domains $G^\sigma_{l,m}$ that extend $G_{l,m}$ at all sides by $\sigma > 0$ additional cells, i.e.

$$G^\sigma_{l,m} \;:=]x_{1,l}^{j-\sigma-1/2}, x_{1,l}^{j+\mu_1+\sigma-1/2}[\times]x_{2,l}^{k-\sigma-1/2}, x_{2,l}^{k+\mu_2+\sigma-1/2}[\;. \tag{64}$$

Analogous to G_l we denote the enlarged level domain $\bigcup_m G^\sigma_{l,m}$ by G^σ_l. For $\sigma = s$, expression (64) yields the domain required for the numerical update on grid $G_{l,m}$. We denote the additional necessary ghost cell region $G^s_{l,m} \backslash \bar{G}_{l,m}$ by $\tilde{G}^s_{l,m}$. Note that values $\sigma \neq s$ are also used in the following description to express geometric inter-level relations. For instance, only *properly nested* refinements will be allowed that satisfy the equation

$$G^{r_l}_l \cap G_{l-1} = G^{r_l}_l \cap G_0 \tag{65}$$

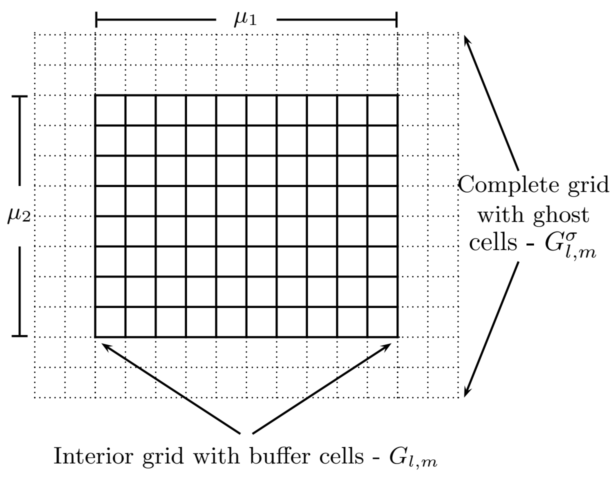

μ_1
μ_2
Complete grid with ghost cells - $G^\sigma_{l,m}$
Interior grid with buffer cells - $G_{l,m}$

FIGURE 4. Parts of a refinement grid $G_{l,m}$.

for all $l > 0$ (cf. Fig. 3). Condition (65) assures that internal cells of level l can abut only internal cells of the levels $l-1$ and $l+1$.

2.1.3. *Grid-based data*

The notation $\mathbf{Q}(G^s_{l,m}, x^j_{1,l}, x^k_{2,l})$ denotes the approximations of the vector of state that are defined on all discrete points $(x^j_{1,l}, x^k_{2,l})$, $j, k \in \mathbb{Z}$, which satisfy $(x^j_{1,l}, x^k_{2,l}) \in G^s_{l,m}$. In general, some points $(x^j_{1,l}, x^k_{2,l})$ will be contained in multiple extended grids $G^s_{l,m}$. We assume that the data values associated to such points are equal in all sets $\mathbf{Q}(G^s_{l,m}, \cdot, \cdot)$. Under this assumption, we define the vector of state on level l as the union of all grid-based data sets $\mathbf{Q}(\cdot, x^j_{1,l}, x^k_{2,l})$ by

$$\mathbf{Q}^l := \bigcup_{m=1}^{M_l} \mathbf{Q}(G^s_{l,m}, x^j_{1,l}, x^k_{2,l}) \ . \tag{66}$$

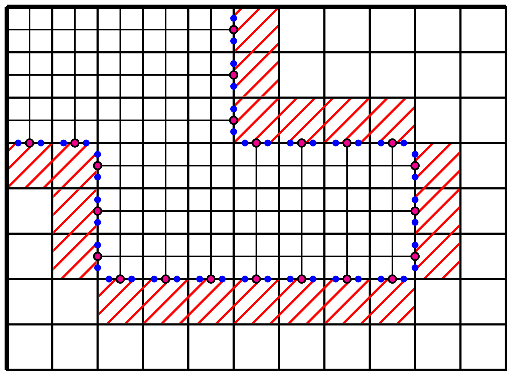
The notations $\mathbf{F}^n(\bar{G}_{l,m}, \cdot, \cdot)$, $n = 1, \ldots, d$ denote the numerical fluxes on the edges of $\bar{G}_{l,m}$. While $\mathbf{F}^1(\bar{G}_{l,m}, x^{j+1/2}_{1,l}, x^k_{2,l})$ is used for the discrete fluxes in the x_1-direction, $\mathbf{F}^2(\bar{G}_{l,m}, x^j_{1,l}, x^{k+1/2}_{2,l})$ denotes the flux approximations in the x_2-direction. Analogous to (66), the numerical fluxes on level l are defined by

$$\mathbf{F}^{n,l} := \bigcup_{m=1}^{M_l} \mathbf{F}^n(\bar{G}_{l,m}, \cdot, \cdot) \ . \tag{67}$$

The notations $\delta\mathbf{F}^n(\partial G_{l,m}, \cdot, \cdot)$, $n = 1, \ldots, d$ are only used on levels with $l > 0$. They denote correction terms associated to the numerical fluxes of level $l-1$ that are defined on the boundary of $G_{l,m}$. The set of correction terms for $\mathbf{F}^{1,l-1}$ is $\delta\mathbf{F}^1(\partial G_{l,m}, x^{j+1/2}_{1,l-1}, x^k_{2,l-1})$ and the corrections for $\mathbf{F}^{2,l-1}$ are stored in $\delta\mathbf{F}^2(\partial G_{l,m}, x^j_{1,l-1}, x^{k+1/2}_{2,l-1})$. The correction terms on level l are $\delta\mathbf{F}^{n,l} := \bigcup_{m=1}^{M_l} \delta\mathbf{F}^n(\partial G_{l,m}, \cdot, \cdot)$. The values of correction terms are only required on lower-dimensional domains $\partial G_l \backslash \partial G_0$ where a fine level $l > 0$ abuts the next coarser level. Since the geometric location of the data values in the different sets has now been explained in detail, we neglect the point information in the following.

2.1.4. *Numerical update*

Suppose all cell values $\mathbf{Q}^l$ are set appropriately, a whole level l is updated by applying the solution operator $\mathcal{H}^{(\cdot)}$ to all grids on level l in a simple loop:



▨ Cells to correct • $\mathbf{F}^{n,l}$ • $\mathbf{F}^{n,l+1}$ ∘ $\delta\mathbf{F}^{n,l+1}$

FIGURE 5. Usage of fine- instead of coarse-grid fluxes to integrate cells abutting a fine grid. Cells needing correction are shaded. The circles mark the locations of the relevant fluxes $\mathbf{F}^{n,l}$, $\mathbf{F}^{n,l+1}$ and of the correction terms $\delta\mathbf{F}^{n,l+1}$.

```
For all $m = 1$ To $M_l$ Do
    $\mathbf{Q}(G^s_{l,m},t)\ ,\mathbf{F}^n(\bar{G}_{l,m},t) \overset{\mathcal{H}^{(\Delta t_l)}}{\longrightarrow} \mathbf{Q}(G_{l,m},t+\Delta t_l)$
```

The loop involves the grid-wise update of the flux approximations $\mathbf{F}^{n,l}$.

2.1.5. *Conservative averaging*

When two levels l and $l+1$ reach the same discrete time, the fine-level values are projected onto the coarser level since the fine-level approximation can be assumed to be more accurate. Each interior cell value of level l in $G_l \cap G_{l+1}$ is replaced by the conservative average of the r^2_{l+1} internal cells of level $l+1$ that overlay it. The value $\mathbf{Q}^l_{jk}$ of cell (j,k) is replaced by

$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota} \,. \tag{68}$$

Note that the application of the projection formula (68) generally results in a conservation error because numerical fluxes between the fine- and the coarse-level domain are not considered. A special flux correction is applied along coarse-fine boundaries to account for this issue. The correction has to be applied in all coarse-level cells abutting a higher-level refinement region.

2.1.6. *Conservative flux correction*

When the update formula (60) is applied to the cells of level l contained in $(G^{r_{l+1}}_{l+1}\backslash G_{l+1}) \cap G_l$, we have to replace the coarse flux approximation with all modified neighboring cells by the sum of all overlying fine-level fluxes [14]. Note that condition (65) ensures that only cells of level l have to be corrected. Figure 5 shows these cells for a particular refinement. As an example, we consider the cell (j,k). The correct update for $\mathbf{Q}^l_{jk}$ is

$$\begin{aligned}\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) = \mathbf{Q}^l_{jk}(t) - \frac{\Delta t_l}{\Delta x_{1,l}} &\left(\mathbf{F}^{1,l}_{j+\frac{1}{2},k} - \frac{1}{r^2_{l+1}} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{F}^{1,l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1}) \right) \\ - \frac{\Delta t_l}{\Delta x_{2,l}} &\left(\mathbf{F}^{2,l}_{j,k+\frac{1}{2}} - \mathbf{F}^{2,l}_{j,k-\frac{1}{2}} \right) .\end{aligned} \tag{69}$$

In order to replace $\mathbf{Q}^l_{jk}(t+\Delta t_l)$ (calculated with the unaltered scheme (60)) by $\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l)$, we use the correction procedure proposed in [14] that avoids the modification of the numerical scheme. After the update on level l we initialize the correction term $\delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k}$, which belongs to the fine-level boundary but is associated to the point $(x^{j-1/2}_{1,l}, x^k_{2,l})$, by

$$\delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k} := -\mathbf{F}^{1,l}_{j-\frac{1}{2},k} \; . \tag{70}$$

During the r_{l+1} update steps of level $l+1$ we accumulate all necessary fine-level fluxes, i.e.

$$\delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k} := \delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k} + \frac{1}{r^2_{l+1}} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{F}^{1,l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1}) \; . \tag{71}$$

When the integration of the fine level is complete, the correction is applied by modifying $\mathbf{Q}^l_{jk}(t+\Delta t_l)$ by

$$\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) := \mathbf{Q}^l_{jk}(t+\Delta t_l) + \frac{\Delta t_l}{\Delta x_{1,l}}\,\delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k} \; . \tag{72}$$

To avoid the usage of the numerical fluxes of the entire level, we combine the numerical update and the computation of the correction terms in a single loop:

```
update_level($l$)
  For all $m = 1$ To $M_l$ Do
      $\mathbf{Q}(G^s_{l,m}, t)\ ,\mathbf{F}^n(\bar{G}_{l,m}, t) \overset{\mathcal{H}^{(\Delta t_l)}}{\longrightarrow} \mathbf{Q}(G_{l,m}, t+\Delta t_l)$
      If level $l > 0$
          Add $\mathbf{F}^n(\partial G_{l,m}, t)$ to $\delta\mathbf{F}^{n,l}$
      If level $l+1$ exists
          Init $\delta\mathbf{F}^{n,l+1}$ with $\mathbf{F}^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t)$
```

Algorithm 1. Numerical update of $\mathbf{Q}$ and calculation of correction terms on level l.

2.1.7. *Boundary conditions*

Three different types of boundary conditions have to be considered in the SAMR method to set the values of $\mathbf{Q}(\tilde{G}^s_{l,m})$ in the ghost cell region $\tilde{G}^s_{l,m} := G^s_{l,m}\backslash\bar{G}_{l,m}$. Cells in

$$\tilde{P}^s_{l,m} = \tilde{G}^s_{l,m}\backslash G_0 \tag{73}$$

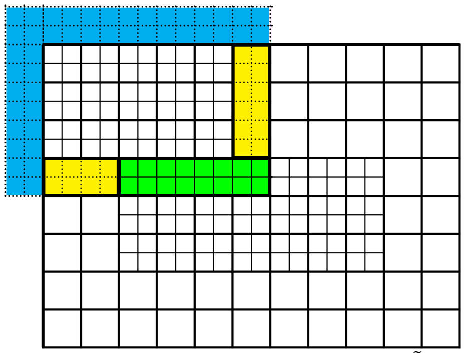
are auxiliary cells outside of the physical domain. Their values are used to implement *physical* boundary conditions. Cells in

$$\tilde{S}^s_{l,m} = \tilde{G}^s_{l,m} \cap G_l \tag{74}$$

have a unique interior cell analogue and are set by copying the data value from the data set of the grid, where the interior cell is contained. We call the overwriting of ghost cell values with internal cell values *synchronization.* It assures the validity of the equal data assumption necessary for the definitions of Section 2.1.3. On the root level no further boundary conditions need to be considered. Yet, for $l > 0$ also *internal* boundaries can occur. In the SAMR method ghost cells in the domain

$$\tilde{I}^s_{l,m} = \tilde{G}^s_{l,m}\backslash(\tilde{S}^s_{l,m} \cup \tilde{P}^s_{l,m}) \tag{75}$$

are used to set internal Dirichlet boundary conditions by time-space interpolation, where $G_{l,m}$ abuts G_{l-1}. The SAMR method is usually implemented with simple linear interpolation operations [14]. For instance, for the



🟩 Synchronization with G_l - $\tilde{S}^s_{l,m} = \tilde{G}^s_{l,m} \cap G_l$
🟦 Physical boundary conditions - $\tilde{P}^s_{l,m} = \tilde{G}^s_{l,m} \backslash G_0$
🟨 Interpolation from G_{l-1} - $\tilde{I}^s_{l,m} = \tilde{G}^s_{l,m} \backslash (\tilde{S}^s_{l,m} \cup \tilde{P}^s_{l,m})$

FIGURE 7. Ghost cell regions of a refinement grid $G_{l,m}$.

fine-level cell (v, w) of Fig. 6, a frequently used bilinear spatial interpolation reads

$$\check{\mathbf{Q}}^l_{vw} := (1-f_1)(1-f_2)\, \mathbf{Q}^{l-1}_{j-1,k-1} + f_1(1-f_2)\, \mathbf{Q}^{l-1}_{j,k-1} + (1-f_1)f_2\, \mathbf{Q}^{l-1}_{j-1,k} + f_1 f_2\, \mathbf{Q}^{l-1}_{jk} \tag{76}$$

with factors

$$f_1 := \frac{x^v_{1,l} - x^{j-1}_{1,l-1}}{\Delta x_{1,l-1}} \ , \quad f_2 := \frac{x^w_{2,l} - x^{k-1}_{2,l-1}}{\Delta x_{2,l-1}} \ . \tag{77}$$

The interpolation (76) is followed by a linear time-interpolation to supply suitable internal boundary conditions at discrete time steps that do not exist on level $l-1$, i.e.,

$$\tilde{\mathbf{Q}}^l(t+\kappa\Delta t_l) := \left(1 - \frac{\kappa}{r_l}\right) \check{\mathbf{Q}}^l(t) + \frac{\kappa}{r_l}\, \check{\mathbf{Q}}^l(t+\Delta t_{l-1}) \quad \text{for } \kappa = 0, \ldots r_l - 1 \ . \tag{78}$$

Figure 7 displays all types of boundary conditions for levels with $l > 0$. The setting of all ghost cell values on level l requires just a loop over all subgrids and the application of the three types of boundary conditions. Since the domains $\tilde{P}^s_{l,m}$, $\tilde{S}^s_{l,m}$, $\tilde{I}^s_{l,m}$ do not overlap, the order is arbitrary.

The interpolation formula (76) is also employed to initialize $\mathbf{Q}^l$ from coarser data in new refinement regions during the regridding procedure. The maximal domain of the space-interpolation (76) is G^ν_l with $\nu = (\lfloor s/r_l \rfloor + 1)\, r_l$.[14] Since the condition $\lfloor s/r_l \rfloor + 1 \le s$ is satisfied for all $r_l \ge 2$, $s \ge 1$ the interpolation domain G^ν_l is always fully contained within G^s_{l-1}, yet in general exceeds the interior grid domain G_{l-1}. Consequently, time-space interpolation on level l requires the previous ghost cell setting for $\mathbf{Q}^{l-1}$ on the entire domain at both time steps t and $t + \Delta t_{l-1}$.

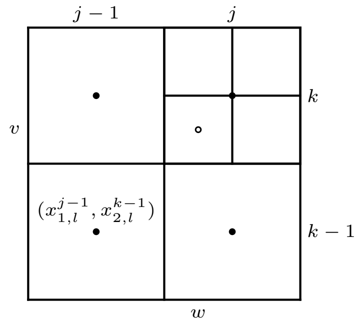


FIGURE 6. Cells used for interpolating data into the fine-level cell (v, w).

[14] $\lfloor . \rfloor$ denotes the Gauss-function which rounds off to the next integer.

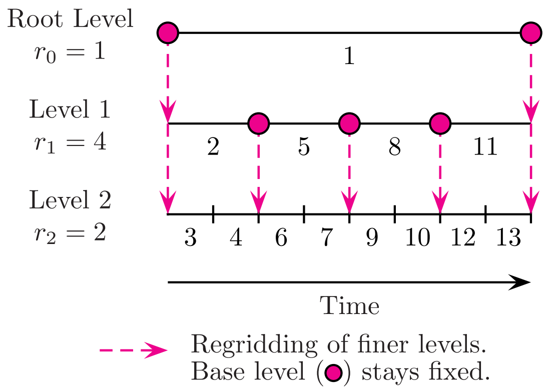


FIGURE 8. Recursive integration order of SAMR.

2.1.8. *The recursive algorithm*

Interpolation in time during the r_l time steps of level l necessitates the availability of the coarse-level values $\mathbf{Q}^{l-1}(t)$ and $\mathbf{Q}^{l-1}(t+\Delta t_{l-1})$. Therefore, the numerical update must be calculated first on level $l-1$. Further on, the ghost cell values of $\mathbf{Q}^{l-1}(t+\Delta t_{l-1})$ must be set before advancing level l. On the other hand, we want to replace coarse-level values successively with the highest level approximation available. A recursive algorithm is most appropriate to achieve these purposes. The basic SAMR algorithm is formulated in Algorithm 2. Except the function `regrid(`l`)`, that modifies the grid hierarchy, all elements of Algorithm 2 have already been explained. Note that the setting of the boundary values of $\mathbf{Q}^l(t)$ at the beginning of `advance_level(`l`)` is mandatory. Although boundary values of coarser levels have already been set before advancing the next finer level, a further application of the boundary conditions is necessary to take changes due to restriction and flux correction into account. An example for the temporal integration order of the numerical solution on a three level hierarchy is shown in Fig. 8. The arrows denote regridding of finer levels. The level at which the regridding procedure is initiated (marked by the circles) stays fixed. The recursive integration order of Algorithm 2 can be started by calling `advance_level(0)` on the root level.

```
advance_level($l$)
  Repeat $r_l$ times
      Set ghost cells of $\mathbf{Q}^l(t)$
      If time to regrid
          regrid($l$)
      update_level($l$)
      If level $l+1$ exists
          Set ghost cells of $\mathbf{Q}^l(t+\Delta t_l)$
          advance_level($l+1$)
          Average $\mathbf{Q}^{l+1}(t+\Delta t_l)$ onto $\mathbf{Q}^l(t+\Delta t_l)$
          Correct $\mathbf{Q}^l(t+\Delta t_l)$ with $\delta\mathbf{F}^{n,l+1}$
      $t := t+\Delta t_l$
```

ALGORITHM 2. The basic recursive SAMR algorithm.

2.1.9. *Grid generation*

A level l initiates the creation of new refinement grids based upon the data of all levels ι that satisfy $\iota \geq l$. Level l by itself is not modified. To consider the nesting condition (65) already in the grid generation, the regridding procedure starts at the highest level available that allows further refinement. We denote its level number by l_c, which satisfies the condition $0 \le l_c < l_{\max}$.

Special indicators (see Section 2.3) are used to flag cells for refinement. Grid-based integer data sets $N^\iota := \bigcup_m N(G_{\iota,m}, x^j_{1,\iota}, x^k_{2,\iota})$ are used to store these flags. Additional buffer cells are marked around each flagged cell. In order to ensure that a feature, which has caused the flagging, remains within the refinement region until the next regridding, the size of the buffer zone b must satisfy the relation $b \ge \kappa_r$. Herein, κ_r denotes the number of time steps between two regridding procedures. To minimize the influence of internal boundary conditions on the solution, $b > \kappa_r$ should be used. A buffer zone of two cells is typical for the standard strategy of regridding in every time step (cf. Fig. 8).

A special *clustering* algorithm (cf. Section 2.1.10) is employed to create new refinement grids $\breve{G}_{\iota+1,m} \subset G_0$ on the basis of N^ι. This algorithm generates successively smaller grids until the ratio between flagged and all cells in every new grid $\breve{G}_{\iota+1,m}$ is above a prescribed threshold $0 < \eta_{tol} < 1$. As usual, we use $\breve{G}_\iota := \bigcup_m \breve{G}_{\iota,m}$. In order to ensure that the previously generated new refinement grids of the next finer level are fully contained in $\breve{G}_{\iota+1}$, all cells in N^ι below $\breve{G}_{\iota+2}$ are also flagged before creating the buffer zone. Before the new grids $\breve{G}_{\iota+1}$ can be used to replace $G_{\iota+1}$, the validity of the nesting condition (65) has to be enforced. In Algorithm 3, we evaluate the invalid region for level $\iota+1$ by calculating the complement $C\breve{G}_\iota := G_0 \backslash \breve{G}_\iota$ of the next coarse-level domain $\breve{G}_\iota$ in G_0 and by enlarging $C\breve{G}_\iota$ by one additional cell, i.e. $C\breve{G}^1_\iota$. The operation $\breve{G}_{\iota+1} := \breve{G}_{\iota+1} \backslash C\breve{G}^1_\iota$ then eliminates all regions violating (65) from the new level domain $\breve{G}_{\iota+1}$. Note that Algorithm 3 can create only one new level above l_c; however, all levels above l could be removed.

```
regrid($l$) - Regrid all levels $\iota > l$
  For $\iota = l_c$ Downto $l$ Do
      Flag $N^\iota$ according to $\mathbf{Q}^\iota(t)$
      If level $\iota+1$ exists?
          Flag $N^\iota$ below $\breve{G}_{\iota+2}$
      Flag buffer zone on $N^\iota$
      Generate $\breve{G}_{\iota+1}$ from $N^\iota$
  $\breve{G}_l := G_l$
  For $\iota = l$ To $l_c$ Do
      $C\breve{G}_\iota := G_0 \backslash \breve{G}_\iota$, $\breve{G}_{\iota+1} := \breve{G}_{\iota+1} \backslash C\breve{G}^1_\iota$
  recompose($l$)
```

Algorithm 3. The regridding procedure.

The reinitialization of the hierarchy is done in `recompose(l)`. In particular, grid-based auxiliary data $\breve{\mathbf{Q}}(\breve{G}_\iota, t)$ are necessary to reorganize the grid-based data of the vector of state. Cells in newly refined regions $\breve{G}_\iota \backslash G_\iota$ are initialized by interpolation; values of cells in $\breve{G}_\iota \cap G_\iota$ are copied. Since interpolation requires the previous reorganization of $\mathbf{Q}^{\iota-1}(t)$ (including an update of ghost cell values), recomposition starts on level $l+1$.

```
recompose($l$) - Reorganize all levels $\iota > l$
  For $\iota = l+1$ To $l_c+1$ Do
      Interpolate $\mathbf{Q}^{\iota-1}(t)$ onto $\breve{\mathbf{Q}}^\iota(t)$
      Copy $\mathbf{Q}^\iota(t)$ onto $\breve{\mathbf{Q}}^\iota(t)$
      Set ghost cells of $\breve{\mathbf{Q}}^\iota(t)$
      $\mathbf{Q}^\iota(t) := \breve{\mathbf{Q}}^\iota(t)$, $G_\iota := \breve{G}_\iota$
```

Algorithm 4. Serial recomposition.

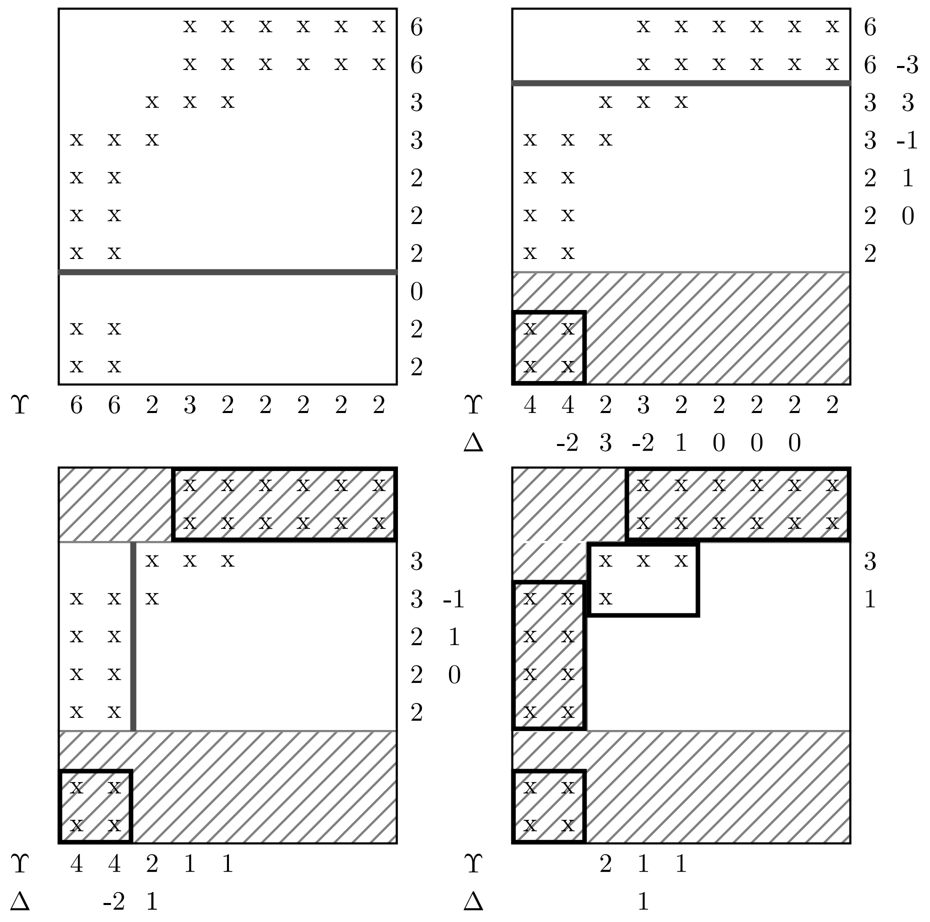


Figure 9. Grid generation by signatures [11].

2.1.10. *Clustering by signatures*

We use the algorithm proposed by Bell et al. [11] to cluster flagged cells into new Grids $\breve{G}_{\iota+1,m}$ (see also [13, 17]). This method, inspired by techniques used in image detection, counts the number of flagged cells in each row and column on the entire domain of N^{ι}. The sums Υ are called *signatures*. First, cuts into new boxes are placed on all edges where Υ is equal to zero (upper left picture of Fig. 9). In the second step, cuts are placed at zero crossings of the discrete second derivative $\Delta = \Upsilon_{\nu+1} - 2\,\Upsilon_{\nu} + \Upsilon_{\nu-1}$. The algorithm starts with the steepest zero crossing and uses recursively weaker ones, until the ratio between flagged and all cells in every new grid is above the prescribed threshold value η_{tol} (upper right and lower left picture of Fig. 9). Typical values for η_{tol} are between 0.7 and 0.9.

2.2. **Parallel algorithm**

The computationally most expensive operation of Algorithm 2 is the numerical update in `update_level(`l`)`. The update loop over all subgrids $G^s_{l,m}$ can be parallelized in a straightforward way by computing the update of different grids on different computing nodes. A time-explicit scheme only requires the synchronization of $\mathbf{Q}^l(t)$ before the grid-based data is distributed and this operation is already part of the boundary update in the basic algorithm. Hence, the efficient usage of parallel computers with shared memory is straightforward. Only an estimation of the necessary work on each subgrid is necessary to split the loop in `update_level(`l`)` in a

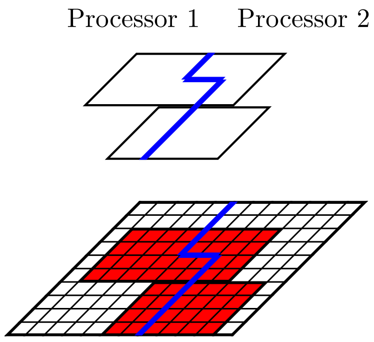


Figure 10. Splitting of refinement grids due to distribution based on the root level.

load-balancing manner. However, this simple strategy is not practicable on parallel computers with distributed memory. The computing nodes of distributed memory architectures usually do not have enough memory to store the complete data of large-scale problems and the hierarchical data itself must be distributed among the available nodes. Unfortunately, the order of partitioning objectives is not clear: (i) the recursive algorithm (in principle) requires load balanced work on each level, (ii) existing subgrids should not be split, (iii) synchronization and inter-level communication costs should be small, and (iv) the repartitioning overhead during regridding should be minimal. In the following, we describe the rigorous domain decomposition approach that was chosen in our own implementation AMROC [31,33] that satisfies all partitioning objectives, although with some compromises in (i) and (ii). The main idea has been proposed by Parashar & Browne [91,92] and was implemented in the hierarchical data structures of the DAGH library[15]. An alternative approach has been pursued by Rendleman et al. [97] in Berkeley-Lab-AMR. They used a partitioning strategy based on the level-wise application of a Knapsack algorithm that satisfies objectives (i) and (ii) almost perfectly but neglects (iii) and (iv) entirely.

2.2.1. *Decomposition of the hierarchy*

We assume a parallel machine with P identical processing nodes. The core idea of the chosen rigorous domain decomposition approach is to partition first of all the computational domain. The root domain G_0 is split into P non-overlapping portions G_0^p, $p = 1, \dots, P$ by

$$G_0 = \bigcup_{p=1}^{P} G_0^p \quad \text{with} \quad G_0^p \cap G_0^q = \emptyset \;\; \text{for } p \neq q \;, \tag{79}$$

which are defined as usual as the union of new non-overlapping grids $G_{l,m}^p$ by

$$G_0^p := \bigcup_{m=1}^{M_0^p} G_{0,m}^p \; . \tag{80}$$

The key idea now is that all higher level domains G_l are required to follow the decomposition of the root level:

$$G_l^p := G_l \cap G_0^p \tag{81}$$

Condition (81) can cause the splitting of a subgrid $G_{l,m}$ into multiple subgrids $G_{l,\kappa}^p$ on different processors. Although the merging of subgrids $G_{l,\kappa}^p$ on processor p is allowed, the total number of grids in $\bigcup_p G_l^p$ usually exceeds the number of grids in G_l, i.e., $\sum_p M_l^p > M_l$.[16] Under requirement (81) we estimate the work on an

[15]http://userweb.cs.utexas.edu/users/dagh. Mere C++ prototypical data structures, no schemes included.

[16]For the example of Fig. 10 we have $M_1^p = 2$, $M_1^q = 1$ and $M_1 = 2$.

arbitrary rectangular sub-domain $\Omega \subset G_0$ by

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right] . \tag{82}$$

Herein, $\mathcal{N}_l(G)$ is the total number of FV cells of level l that are completely contained in $\bar{G}$. The product in (82) is used to consider the time step refinement. A nearly equal distribution of the work necessitates

$$\mathcal{L}^p := \frac{P \cdot \mathcal{W}(G_0^p)}{\mathcal{W}(G_0)} \approx 1 \quad \text{for all } p = 1, \ldots, P . \tag{83}$$

The creation of a load-balanced decomposition G_0^p requires an appropriate partitioning algorithm.

Note that the reduction of the communication overhead is already considered in condition (81) in a natural way. Together with the use of synchronized ghost cells this condition allows a strictly local execution of most SAMR operations. In particular, no major modification of Algorithm 2 is necessary, which simplifies the practical implementation.

2.2.2. *Boundary conditions*

Our domain decomposition technique involves a slight increase in complexity for ghost cell synchronization. In the parallel algorithm, the synchronization domain of a decomposed grid $G_{l,m}^{s,p}$ on node p is divided into the local domain

$$\tilde{S}_{l,m}^{s,p} = \tilde{G}_{l,m}^{s,p} \cap G_l^p , \tag{84}$$

and the parallel domains

$$\tilde{S}_{l,m}^{s,q} = \tilde{G}_{l,m}^{s,p} \cap G_l^q , \quad q = 1, \ldots, P , \quad q \neq p . \tag{85}$$

While the cell values in $\tilde{S}_{l,m}^{s,p}$ can be copied from interior cells, which are locally available on p, the setting of cells in $\tilde{S}_{l,m}^{s,q}$ requires communication with node q, on which the interior cells originally reside.

The setting of physical and internal boundaries remains strictly local. Analogous to Section 2.1.7, the domain for the spatial interpolation, $G_l^{\nu,p}$, is fully contained in $G_{l-1}^{s,p}$, the local domain of the next coarser level. Since the parallel synchronization of $\mathbf{Q}^{l-1}(t)$ and $\mathbf{Q}^{l-1}(t + \Delta t_{l-1})$ is guaranteed by the SAMR algorithm itself, the parallel synchronization of level l itself is the only communication operation necessary to set the ghost cells on this level.

2.2.3. *Numerical update and flux correction*

The function `update_level(l)` does not involve any parallel overhead. Apparently, the new vector of state $\mathbf{Q}(G_{l,m}^p, t + \Delta t_l)$ on each grid $G_{l,m}^p$ and the fluxes $\mathbf{F}^n(\bar{G}_{l,m}^p, t)$ can be computed strictly local on the basis of $\mathbf{Q}(G_{l,m}^{s,p})$, but also the computation of the correction terms does not require communication.

To illustrate this, we assume a parallel border in Fig. 11 at $j - \frac{1}{2}$. Let cell (j,k) be contained in G_l^q and let cell (v,w) be contained in G_{l+1}^p. Then, the necessary correction term $\delta\mathbf{F}_{j-1/2,k}^{1,l+1}$ resides on node p as it is assigned to the fine level. The initialization of this term in (70) requires the coarse-grid flux $\mathbf{F}_{j-1/2,k}^{1,l}$. This flux is available on node p, because the basic SAMR strategy ensures that below (v,w) an interior coarse-level cell $(j-1,k)$ exists, having $\mathbf{F}_{j-1/2,k}^{1,l}$ as flux into a ghost cell (j,k). On the other hand, $\mathbf{F}_{j-1/2,k}^{1,l}$ is also computed on q, where (j,k) is interior and $(j-1,k)$ is a ghost cell. Since the ghost cells have been synchronized before the numerical update, the same boundary flux is calculated on both nodes (cf. level l in Fig. 11). The fine-grid fluxes $\mathbf{F}_{v+1/2,w+\iota}^{1,l+1}$ are only available on p, because no abutting interior fine-grid cell exists on q. Since the correction term $\delta\mathbf{F}_{j-1/2,k}^{1,l+1}$ is also stored on p, the summation in (71) remains local (cf. level $l+1$ of node p in Fig. 11).

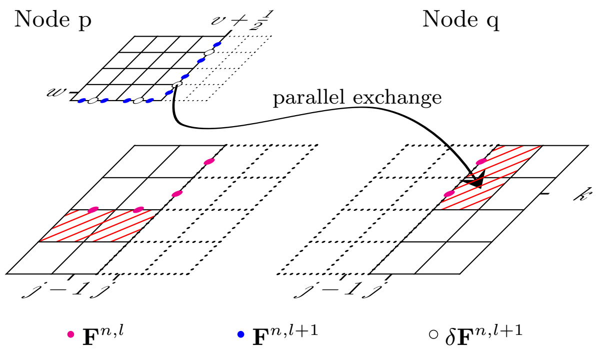


FIGURE 11. Parallelized conservative flux correction. Cells needing correction are shaded. The circles mark the locations of the relevant fluxes $\mathbf{F}^{n,l}$, $\mathbf{F}^{n,l+1}$ and of the correction terms $\delta\mathbf{F}^{n,l+1}$.

The only operation of the flux correction that necessarily requires communication is the application of correction terms in Eq. (72). In our example, the term $\delta\mathbf{F}^{1,l+1}_{j-1/2,k}$ of node p has to be added to the value of the interior cell (j,k), which is stored on node q. Our practical implementation only allows the setting of ghost cell values from interior cells. We achieve the parallel exchange by employing auxiliary grid-based *cell-centered* data $\mathbf{H}(G^{1,p}_{l,m}, x^{j^*}_{1,l}, x^{k^*}_{2,l})$ that has an overlap of one cell. $\mathbf{H}^l$ is initialized with zero everywhere. On node p, the value $\delta\mathbf{F}^{1,l+1}_{j-1/2,k}$ is copied into the interior cell $(j-1,k)$. $\mathbf{H}^l$ is then synchronized and all values are shifted by one cell to the right. On node q, this technique transfers $\delta\mathbf{F}^{1,l+1}_{j-1/2,k}$ from the ghost cell $(j-1,k)$ to the interior cell (j,k), where it can be applied. The simultaneous application of this trick to all correction terms $\delta\mathbf{F}^{n,l+1}$ reduces the parallel overhead for the entire procedure to two synchronization operations per space direction. The transfer and application of $\delta\mathbf{F}^{1,l+1}_{j-1/2,k}$ via $\mathbf{H}^l$ to the interior cell (j,k) is expressed by the black arrow in Fig. 11.

Finally, we remark that the strict locality of the inter-level averaging (68) follows directly from condition (81). This property avoids the expensive parallel communication of volume data during the averaging operation.

2.2.4. *Parallel grid generation*

Analogous to Algorithm 2 the regridding procedure formulated in Algorithm 3 is hardly affected by parallelization. The flagging of cells on each level can be done locally. If an error estimation criterion is used (cf. Section 2.3), the computation of auxiliary time steps involves parallel boundary synchronization, yet it does not alter Algorithm 3 itself. The only difficult task in the creation of $\breve{G}_{\iota+1}$ from N^ι in Algorithm 3 is the clustering.

Two possibilities exist for running the cluster algorithm in parallel. The cluster algorithm could be executed strictly local on $N(G^p_\iota)$ or it could be executed on the data of the entire level $N(G_\iota)$. Note that both options can be guaranteed to give an identical result only for a clustering threshold of $\eta_{tol} = 1$. For $\eta_{tol} < 1$ the algorithm has some freedom in combining flagged and non-flagged cells leading to slightly different results for both approaches. Naively implemented, the second option would require a global concatenation of all data sets $N(G^p_\iota)$ to $N(G_\iota)$, which is prohibitively expensive for large-scale problems. A truly parallel cluster algorithm has recently been presented by Gunney et al. [52]; however, we have opted to accept some ambiguities in parallel for $\eta_{tol} < 1$ for the sake of simplicity in our approach. We execute the cluster algorithm strictly local and communicate only the result $\breve{G}^p_{\iota+1}$ to obtain the global list $\breve{G}_{\iota+1} = \bigcup_p \breve{G}^p_{\iota+1}$. The global list $\breve{G}_{\iota+1}$ is mandatory to ensure the correct proper nesting of the new hierarchy. To consider the buffer zone before local clustering, we use extended grid-based data $\tilde{N}^\iota := \bigcup_m N(G^b_{\iota,m}, x^j_{1,\iota}, x^k_{2,\iota})$ instead of N^ι. Herein, b is the size of the buffer region (see Section 2.1.9). By synchronizing $\tilde{N}^\iota$ before creating the buffer zone we ensure that all interior cells are flagged correctly.

```
recompose($l$) - Reorganize all levels
  Generate $G_0^p$ from $\{G_0, ..., G_l, \breve{G}_{l+1}, ..., \breve{G}_{l_c+1}\}$
  For $\iota = 0$ To $l_c + 1$ Do
      If $\iota > l$
          $\breve{G}_\iota^p := \breve{G}_\iota \cap G_0^p$
          Interpolate $\mathbf{Q}^{\iota-1}(t)$ onto $\breve{\mathbf{Q}}^\iota(t)$
      else
          $\breve{G}_\iota^p := G_\iota \cap G_0^p$
          If $\iota > 0$
              Copy $\delta\mathbf{F}^{n,\iota}$ onto $\delta\breve{\mathbf{F}}^{n,\iota}$
              $\delta\mathbf{F}^{n,\iota} := \delta\breve{\mathbf{F}}^{n,\iota}$
      If $\iota \geq l$ then $\kappa_\iota = 0$ else $\kappa_\iota = 1$
      For $\kappa = 0$ To $\kappa_\iota$ Do
          Copy $\mathbf{Q}^\iota(t + \kappa\Delta t_\iota)$ onto $\breve{\mathbf{Q}}^\iota(t + \kappa\Delta t_\iota)$
          Set ghost cells of $\breve{\mathbf{Q}}^\iota(t + \kappa\Delta t_\iota)$
          $\mathbf{Q}^\iota(t + \kappa\Delta t_\iota) := \breve{\mathbf{Q}}^\iota(t + \kappa\Delta t_\iota)$
      $G_\iota^p := \breve{G}_\iota^p$, $G_\iota := \bigcup_p G_\iota^p$
```

Algorithm 5. Parallel recomposition. Executed on each node $p = 1, \ldots, P$.

The main changes in the regridding procedure are in `recompose(`l`)`. Instead of Algorithm 4 we apply Algorithm 5. Due to our distribution strategy, we have now to consider a complete reorganization of the entire hierarchy, even for a regridding at a higher level. In Fig. 8, this corresponds to the three regridding operations initiated by level 1. Particularly, the whole relevant data of levels with $\iota \leq l$ has to be copied. Like the synchronization operation, these copy operations are partially local and parallel. For levels with $\iota < l$ the relevant data is $\mathbf{Q}^\iota(t)$, $\mathbf{Q}^\iota(t+\Delta t_\iota)$ and $\delta\mathbf{F}^{n,\iota}$, for level l we have to copy $\mathbf{Q}^l(t)$ and $\delta\mathbf{F}^{n,l}$. The initialization of a level with $\iota > l$ is in principle identical to Algorithm 4. As explained in the previous section, the interpolation is a strictly local operation, provided that the next coarser level has already been reorganized. The copy operation is a combination of local and parallel copy.

2.2.5. *Partitioning*

It is evident that the overall efficiency of the chosen parallelization strategy depends especially on the first step of `recompose(`l`)`, the partitioning algorithm. This algorithm has to create a load-balanced domain decomposition for the new hierarchy, which consists for $\iota \leq l$ of unchanged level domains G_ι and for $\iota > l$ of new domains $\breve{G}_\iota$. The algorithm has to meet several requirements: It must balance the estimated workload, while the amount of data, that has to be synchronized during the numerical solution procedure, should be as small as possible. A slight change of the grid hierarchy should involve a moderate data redistribution. Execution of the partitioning algorithm should be fast as it is carried out on-the-fly.

Distribution strategies based on space-filling curves give a good compromise between these partially competing requirements. A space-filling curve defines a continuous mapping from $[0, 1]$ onto $[0, 1]^d$, $d \geq 2$, cf. [100]. As such curves can be constructed recursively, they are locality-preserving and therefore avoid an excessive redistribution overhead. Further on, the surface is small, which reduces synchronization costs. By applying the mapping of a space-filling curve to the discrete index space of the root level, the root level cells become ordered. This sequence can easily be split into portions of equal size yielding load-balanced new distributions G_0^p. The computational time necessary for distribution can be decreased if neighboring cells with the same workload are concatenated. In this case, generalized space-filling must be employed [91, 92].

In our case, computation of a generalized space-filling curve is done directly in the cell indices, which requires an index domain satisfying $(2^\nu)^d$ with $\nu \in \mathbb{N}$. In most cases, the space-filling curve thereby exceeds the

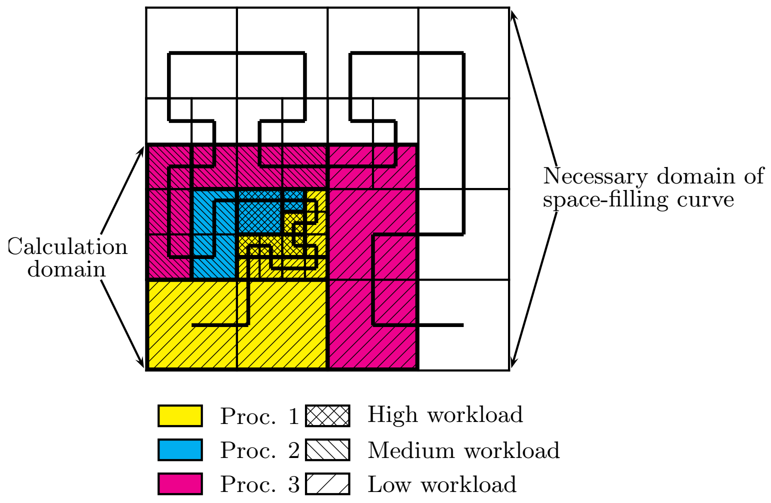


Figure 12. A generalization of Hilbert's space-filling curve is used to distribute grid blocks. The domain of the space-filling curve exceeds the calculation domain as the number of cells in the x_1- and x_2-direction are not of the same power of 2.

computational domain (see Fig. 12); however, the possibly resultant disconnect of some processor sub-domains and thereby increased communication costs are assumed to be negligible.

2.3. Refinement indicators

Rigourously derived error estimations for finite volume schemes for conservation laws are only available for scalar equations [67] and are usually not particularly sharp. Their efficiency is rather modest and practitioners still employ primarily heuristic and physically motivated indicators instead. In the following, we describe two frequently used refinement indicators that we apply to selected scalar quantities, e.g., to some components of the vector of state or additionally evaluated derived quantities.

2.3.1. *Scaled gradients*

An adaptation along discontinuities can easily be achieved by evaluating gradients multiplied by the step size (aka *scaled gradients*) in all directions. Cell (j,k) is flagged for refinement if any of the relations

$$|w(\mathbf{Q}_{j+1,k}) - w(\mathbf{Q}_{jk})| > \epsilon_w\;,\;\; |w(\mathbf{Q}_{j,k+1}) - w(\mathbf{Q}_{jk})| > \epsilon_w\;,\;\; |w(\mathbf{Q}_{j+1,k+1}) - w(\mathbf{Q}_{jk})| > \epsilon_w \tag{86}$$

is satisfied for an arbitrary scalar quantity w, which is derived from the vector of state $\mathbf{Q}^l(t)$. The constant ϵ_w denotes a prescribed refinement limit.

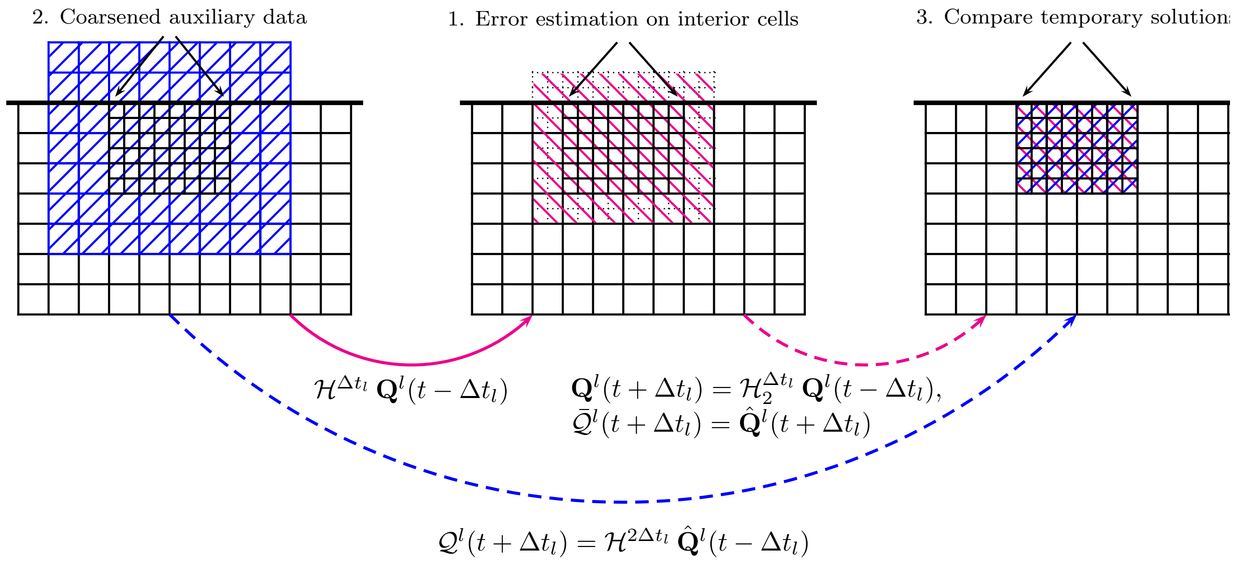
2.3.2. *Heuristic error estimation*

A simple adaptation criterion for regions of smooth solutions is the heuristic estimation of the local truncation error by Richardson extrapolation [12,14,16]. The local truncation error of a difference scheme of order o satisfies

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}^{(\Delta t)}(\mathbf{q}(\cdot, t)) = \mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})\;. \tag{87}$$

If $\mathbf{q}$ is sufficiently smooth, we have for the local error at $t+\Delta t$, after two time steps with Δt,

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}_2^{(\Delta t)}(\mathbf{q}(\cdot, t-\Delta t)) = 2\,\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})\;, \tag{88}$$



2. Coarsened auxiliary data
1. Error estimation on interior cells
3. Compare temporary solution
$\mathcal{H}^{\Delta t_l}\,\mathbf{Q}^l(t-\Delta t_l)$
$\mathbf{Q}^l(t+\Delta t_l)=\mathcal{H}_2^{\Delta t_l}\,\mathbf{Q}^l(t-\Delta t_l),$
$\bar{\mathcal{Q}}^l(t+\Delta t_l)=\hat{\mathbf{Q}}^l(t+\Delta t_l)$
$\mathcal{Q}^l(t+\Delta t_l)=\mathcal{H}^{2\Delta t_l}\,\hat{\mathbf{Q}}^l(t-\Delta t_l)$

FIGURE 13. Utilization of Richardson extrapolation to estimate the error on a refinement subgrid.

and for the local error at $t+\Delta t$, after one time step with $2\Delta t$,

$$\mathbf{q}(\mathbf{x},t+\Delta t)-\mathcal{H}^{(2\Delta t)}(\mathbf{q}(\cdot,t-\Delta t))=2^{o+1}\mathbf{C}\Delta t^{o+1}+O(\Delta t^{o+2})\ . \tag{89}$$

Subtracting (88) from (89) one obtains the relation

$$\mathcal{H}_2^{(\Delta t)}(\mathbf{q}(\cdot,t-\Delta t))-\mathcal{H}^{(2\Delta t)}(\mathbf{q}(\cdot,t-\Delta t))=(2^{o+1}-2)\mathbf{C}\Delta t^{o+1}+O(\Delta t^{o+2})\ , \tag{90}$$

which can be employed to approximate the leading-order term $\mathbf{C}\Delta t^{o+1}$ of the local error at $t+\Delta t$. The implementation of a criterion based on (90) requires a discrete solution $\mathcal{Q}^l$ defined on a mesh two times coarser than the mesh of level l, cf. Fig. 13. With $y_{1,l}^j=(2j+1)\Delta x_{1,l}$ and $y_{1,l}^k=(2k+1)\Delta x_{2,l}$, $j,k\in\mathbb{Z}$ we therefore introduce

$$\mathcal{Q}^l:=\bigcup_m \mathcal{Q}(G_{l,m}^s,y_{1,l}^j,y_{2,l}^k)\ . \tag{91}$$

A coarse approximation $\mathcal{Q}^l(t+\Delta t_l)$ is obtained by restriction of $\mathbf{Q}^l(t-\Delta t_l)$, i.e., $\mathcal{Q}^l(t-\Delta t_l):=\hat{\mathbf{Q}}^l(t-\Delta t_l)$, and taking a time step of $2\Delta t_l$ (cf. blue, dashed arrow of Fig. 13). Finally, a second coarsened solution $\bar{\mathcal{Q}}^l$ is derived by advancing $\mathbf{Q}^l(t)$ tentatively by Δt_l and restriction of $\mathbf{Q}^l(t+\Delta t_l)$, i.e., $\bar{\mathcal{Q}}^l(t+\Delta t_l):=\hat{\mathbf{Q}}^l(t+\Delta t_l)$ (cf. pink arrows of Fig. 13). The difference

$$\tau_{jk}^w:=\frac{|w(\bar{\mathcal{Q}}_{jk}^l(t+\Delta t_l))-w(\mathcal{Q}_{jk}^l(t+\Delta t_l))|}{2^{o+1}-2}>\eta_w \tag{92}$$

is an approximation to the leading-order term of the local error of quantity w. If the latter inequality is satisfied, all four cells below the coarsened cell (j,k) are flagged for refinement. Note that usage of (92) and the choice of η_w can be rather cumbersome for computations in physical units. In such scenarios, we have obtained best results with the criterion

$$\frac{\tau_{jk}^w}{\max(|w(\mathcal{Q}_{jk}^l(t+\Delta t_l))|,S_w)}>\eta_w^r \tag{93}$$

that additionally considers a scaling parameter S_w and combines absolute and relative error.

2.4. **Design of SAMR software**

The explanation of the SAMR method in the previous sections forms the basis for the design of our object-oriented framework AMROC [31]. AMROC currently consists of approximately 46,000 lines of code (loc) in C++ plus approximately 6,000 loc for visualization and data conversion. Computationally expensive single-grid operations (numerical update, prolongation, etc.) are written in Fortran. The same implementation approach is chosen in Berkeley-Lab-AMR that consists of approximately 50,000 loc [97]. Although both packages are written independently of the spatial dimension whenever possible, the drastic increase in complexity compared to the serial two-dimensional Fortran 77 code AMRClaw with approximately 8,500 loc due to the support of distributed memory parallelism is apparent. A salient feature of AMROC, however, is the realization of object-oriented framework concepts on all levels of software design. This allows for effective code re-use in implementing parallel SAMR algorithms and extensive capabilities for customization through subclass derivation.

2.4.1. *Three-level design*

In block-structured dynamically adaptive codes three abstraction levels can be identified. At the top level, a particular physical simulation problem is formulated by providing a numerical scheme, by setting boundary and initial conditions, and by specifying prolongation and restriction methods for the inter-level transfer operations. Characteristic for block-structured methods is that at this level only single-patch routines operating on $\mathbf{Q}(G^s_{l,m})$ have to be provided. In AMROC, SAMR implementation classes call the single-patch routines through abstract class interfaces. For a fully implemented SAMR algorithm, the system is used as an application framework invoked by a generic main program.

Classes implementing SAMR algorithms and their auxiliary components operating on and manipulating complex hierarchical data exactly along the lines of Sections 2.1 and 2.2 make up the second level. A comparison of typical SAMR algorithms, e.g., the Berger-Colella technique for time-explicit finite volume schemes with geometric adaptive multigrid methods for implicit discretizations, reveals that the SAMR auxiliary components show great similarity and can easily be re-used. In AMROC, components such as the flagging of cells for refinement depending on various criteria (cf. Section 2.3), the clustering of flagged cells into rectangular regions (cf. Section 2.1.10), inter-level data transfer (Eqs. (68) and (76)) and flux correction (cf. Section 2.1.6) reside in clearly separated classes. This is highlighted in Fig. 14 that displays the most important AMROC classes and their relationships in Unified Modeling Language (UML) notation [21] for the purely Cartesian case. The recursive SAMR algorithm for hyperbolic problems is realized here in the central class HypSAMRSolver; all other classes are generic, enabling the utilization of AMROC as a software framework for the efficient implementation of different SAMR algorithms, typically implemented in new central SAMRSolver-classes. The middle level operates mainly on grid-based hierarchical data structures that are supplied by the base level.

2.4.2. *The hierarchical data structures*

The base level is divided into elementary functionality for single grid patches and the implementation of various lists that store these patches hierarchically. A common design for the base level (see also [97]) involves a Box-class that specifies a single rectangular box in global integer index space. Methods for geometric operations on boxes like $\cap$, $\cup$, $\setminus$ and the enlargement operation $G^\sigma_{l,m}$ are available. The implementation of these operations can be simplified significantly if a global integer coordinate system is employed. All coordinates in the description in Section 2.1 can be mapped uniquely into this integer coordinate system by replacing the mesh widths $\Delta x_{n,l}$, $l = 0, \ldots, l_{\max}$ by increasing integers, i.e.,

$$\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa \quad \text{for all } n = 1, \ldots, d \; . \tag{94}$$

Further on, we use a similar mapping to denote the discrete time steps by a unique positive integer. The use of integer coordinates eliminates round-off errors completely [11] and speeds up the execution.

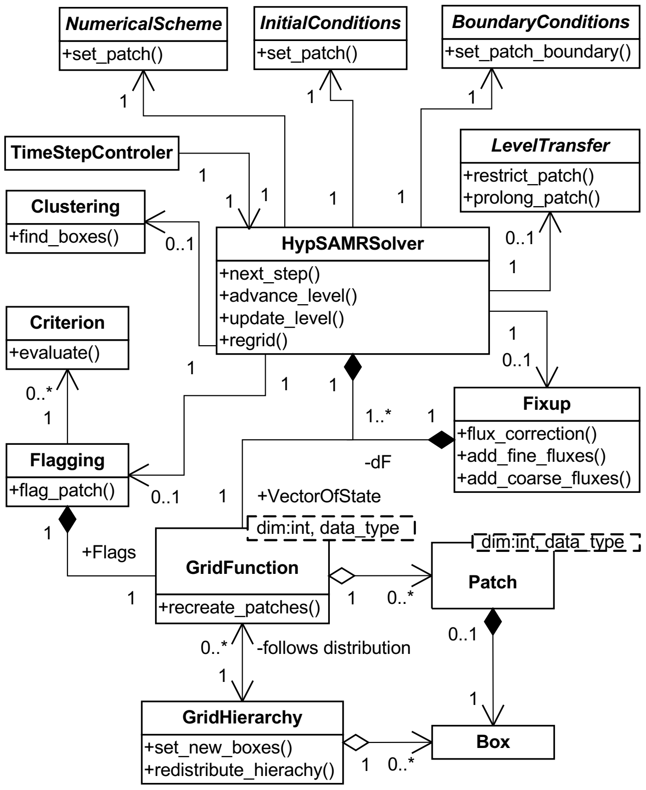


Figure 14. UML diagram for the most important AMROC classes implementing explicit Cartesian SAMR.

A Patch-class adds consecutive data storage to a Box representing a rectangle $G_{l,m}$ in the global integer index space. In AMROC, the geometrical description of all refinement areas is stored in hierarchical lists of Box-objects inside a single GridHierarchy. GridHierarchy holds the global list G_l and the local one, G_l^p, that stores each processor's local contribution. Grid-based data are allocated locally with respect to G_l^p inside the GridFunction-class. For each grid $G_{l,m}^p$, GridFunction allocates a Patch-object. The type of data used is a template parameter for GridFunction. The GridFunction-class also considers parameters for assigning data to different mesh locations (cell centers, edges, vertices), differently sized ghost cell regions, and the reduction of grids to lower-dimensional slices thereby allowing the representation of $\mathbf{Q}^l$, $\delta\mathbf{F}^{n,l+1}$, and N^l. The usage of a templatized base class for all kind of hierarchical grid-based data exploits the commonality in organizing rectangular data blocks independent of their storage type and reduces the implementation work without sacrificing computational performance significantly.

The function `recompose()`, described in pseudo-code in Algorithm 5, is therefore implemented as an interplay between a single GridHierarchy- and multiple GridFunction-objects. When the SAMRSolver-object calls the GridHierarchy method `redistribute_hierachy()` with new lists $\breve{G}_l$, the space-filling curve partitioner (cf. Section 2.2.5) is called and new GridBoxLists G_l and G_l^p are created. `redistribute_hierachy()` then calls the GridFunction method `recreate_patches()` of each GridFunction-object to initiate redistribution of the patch data. This corresponds to the partially local and parallel copy-operations in Algorithm 5. Further on, GridFunction implements the setting of boundary conditions for $\mathbf{Q}^l$. In this case, each GridDataBlock allocates extended data $\mathbf{Q}(G_{l,m}^{s,p})$ and stores detailed topological information on $P_{l,m}^s$, $I_{l,m}^s$, $S_{l,m}^{s,p}$ and $S_{l,m}^{s,q}$. GridFunction sets ghost cells in $P_{l,m}^s$ by a call to the user-defined physical boundary routine. Cells in $I_{l,m}^s$ are set by applying the interpolation function to $\mathbf{Q}^{l-1}$.

TABLE 1. Refinement after the last time step of the test problem of Section 2.5.2 for four computations with an increasing number of refinement levels with AMROC's DAGH and the original DAGH, refinement factors $r_{1,2,3,4} = 2$.

	$l_{\max}$	Level 0	Level 1	Level 2	Level 3	Level 4
AMROC's DAGH grids/cells	1	43/22500	145/38696			
	2	42/22500	110/48708	283/83688		
	3	36/22500	78/54796	245/109476	582/165784	
	4	41/22500	88/56404	233/123756	476/220540	1017/294828
Original DAGH grids/cells	1	238/22500	125/41312			
	2	494/22500	435/48832	190/105216		
	3	695/22500	650/55088	462/133696	185/297984	
	4	875/22500	822/57296	677/149952	428/349184	196/897024

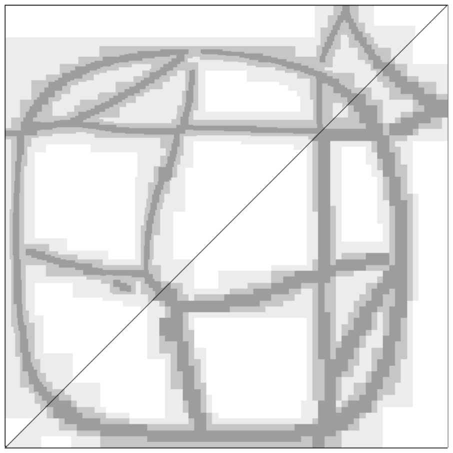


FIGURE 15. Comparison of the refinement grids of the four-level solution with AMROC's DAGH (top left) and the original DAGH (bottom right).

2.4.3. *AMROC's DAGH package*

The design of the hierarchical data structures in AMROC is based on the DAGH (Distributive Adaptive Grid Hierarchies) package by Parashar & Browne [92] that itself was intended as software framework for SAMR algorithms. However, the large complexity of SAMR algorithms and their auxiliary components makes framework concepts at higher design levels (see above) more effective. Further on, a complete redesign of parts of the DAGH package was necessary to allow the SAMR algorithm as it was described previously. AMROC's version of DAGH implements GridFunction- and GridHierarchy-classes that are much more general and allow a more efficient adaptation than the original DAGH package. The GridFunction-class of the original DAGH package is restricted to grids that are aligned to the base mesh coarsened by a factor of 2, i.e.,

$$G^\star_{l,m} := \,](2j-1)\Delta x_{1,0}, (2j+\mu^\star_1-1)\Delta x_{1,0}[\, \times \,](2k-1)\Delta x_{2,0}, (2k+\mu^\star_2-1)\Delta x_{2,0}[\ . \qquad (95)$$

In general, we have $G_l \subseteq G^\star_l$, yet for $l > 0$ typically $G_l \subset G^\star_l$ is satisfied. Therefore, the original DAGH usually refines more cells than required. The limitation in DAGH follows from the simplifying assumption that two grids on neighboring levels only can be connected by a 1 : 1 relation. A coarse-level grid may only have one child and a fine-level grid has exactly one parent. If this assumption is violated, the coarse-level grids are split. Consequently, the maximal number of grids on all levels is equal. This reduces the recomposition overhead on higher levels, but leads to an increasing number of unnecessarily refined cells when advancing the numerical solution. As the entire computational time is usually dominated by the numerical update, AMROC's DAGH is a significant improvement over the original package.

Figure 15 displays the level domains (indicated by different gray scales) used by AMROC's DAGH and the original DAGH of a four-level solution for the two-dimensional spherical shock wave test problem, described in depth in Section 2.5.2, at $t_{\rm end}$. Table 1 shows the number of grids and cells for a constant refinement factor of 2. All solutions have been computed with Hilbert's space-filling curve on 7 processors. Obviously, the simplified data structures perform well if just one or two refinement levels are used, yet for deeper, more realistic hierarchies, the drastic improvement by allowing arbitrary SAMR grids becomes apparent.

Additional new features in AMROC, compared to the original DAGH, are level-dependent refinement factors, periodic boundary conditions, a restart option from memory for explicit schemes with automatic time step adjustment and a restart feature for a variable number of computing nodes.

2.5. **Computational examples**

In this sub-section, we discuss some typical SAMR benchmarks for the Euler equations, expression (49), in two and three space dimensions. The adiabatic constant of Eq. (50) is set to the value commonly used for air ($\gamma = 1.4$) in all computations.

2.5.1. *Accuracy verification*

Our first two-dimensional test is tailored to verify the order of accuracy of the SAMR implementation. Utilizing a first-order accurate interpolation operation (cf. Section 2.1.7), a method of overall second order accuracy can be expected. We use second-order accurate dimensional splitting as explained in Section 1.2.2 and apply the Van Leer FVS detailed in Section 1.5.2. The MUSCL-Hancock variable extrapolation technique with MinMod-limiter (cf. Section 1.4.2) is used to construct a second-order accurate high-resolution scheme. We use the computational domain $\Omega = [-1, 1]^2$ with periodic boundary conditions at all sides. As initial density distribution, the Gaussian function

$$\rho_0(x_1, x_2) = 1 + \exp\left(-\frac{x_1^2 + x_2^2}{r^2}\right) , \tag{96}$$

with $r = 1/4$ is applied. The velocity field is initialized with the constant values $u_{1,0} = u_{2,0} = 1$ and the pressure with $p_0 = 1$ everywhere, giving constant velocity advection of the unperturbed shape along the diagonal. Using $t_{\rm end} = 2$ as final time replicates the initial conditions and the evaluation of error norms becomes straightforward on uniform grids. In here, we use the L_1-error, that is evaluated in the SAMR case as the sum of the errors on the domain G_l of level l without higher refinement. Denoting by $l_{\rm max}$ the highest level available, the norm calculation reads

$$L_1(\rho) = L_1(\Delta x_{n,l_{\rm max}}, G_{l_{\rm max}}) + \sum_{l=0}^{l_{\rm max}-1} L_1(\Delta x_{n,l}, G_l \setminus G_{l+1}), \tag{97}$$

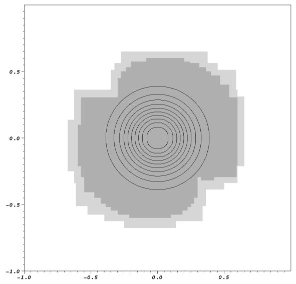
where

$$L_1(\Delta x_n, \cdot) = \sum_{j,k}^{l} |\rho_{jk} - \rho_0(x_{1,l}^j, x_{2,l}^k)| \Delta x_{1,l} \Delta x_{2,l} \tag{98}$$

is the L_1-error norm on a sub-domain of level l. The results for computations on uniform grids with N^2 cells and corresponding computations using two levels of mesh adaptation with $r_{1,2} = 2$ are given in Table 2. The refinement in all adaptive computations is based on the estimation of the absolute local error, Eq. (92), with threshold $\eta_\rho = 5 \cdot 10^{-5}$. The clustering tolerance is set to $\eta_{tol} = 95\%$ and the buffer zone width to $b = 2$ (cf. Section 2.1.9). The graphic right of Table 2 shows the domains of refinement for the adaptive case with base grid 160^2 at $t_{\rm end}$. As it can be expected, the absolute error of the SAMR computations is slightly larger than in the unigrid case; however, a rate of convergence indicating a second-order accurate method, even in the SAMR case, can be confirmed. Additionally, the functioning of the conservative flux correction (fixup), cf. Section 2.1.6, is

TABLE 2. Accuracy verification test case: L_1-error norms of density ρ and order of convergence. Right: Final refinement domains.

N	Unigrid		SAMR - fixup			SAMR - no fixup		
	Error	Order	Error	Order	$\Delta\rho$	Error	Order	$\Delta\rho$
20	0.109464							
40	0.042394	1.369						
80	0.014082	1.590	0.015948		0	0.015960		2e-5
160	0.004929	1.514	0.005267	1.598	0	0.005305	1.589	2e-5
320	0.001461	1.754	0.001565	1.751	0	0.001638	1.695	-1e-5
640	0.000418	1.805	0.000515	1.603	0	0.000600	1.449	-6e-5



verified. If the fixup is deactivated, the density accumulated throughout the domain experiences a mass loss $\Delta\rho$ that increases the error visibly and affects the rate of convergence. Note that this computation used the strictly conservative interpolation

$$\check{\mathbf{Q}}^l_{v,w} := \mathbf{Q}^{l-1}_{jk} + f_1(\mathbf{Q}^{l-1}_{j+1,k} - \mathbf{Q}^{l-1}_{j-1,k}) + f_2(\mathbf{Q}^{l-1}_{j,k+1} - \mathbf{Q}^{l-1}_{j,k-1}) \tag{99}$$

with

$$f_1 = \frac{x^v_{1,l} - x^j_{1,l-1}}{2\Delta x_{1,l-1}} \;, \quad f_2 = \frac{x^w_{2,l} - x^k_{2,l-1}}{2\Delta x_{2,l-1}} \;, \tag{100}$$

instead of (76), which is not monotonicity-preserving and thereby only applicable to smooth problems. For problems with discontinuities, usage of (76) is recommended.

2.5.2. *Parallel benchmark*

In order to evaluate our chosen parallelization strategy, we use a circular two-dimensional shock wave expansion problem. The used scheme $\mathcal{H}^{(\cdot)}$ is the second-order accurate Wave Propagation Method (cf. Section 1.4.3) with the approximate Riemann solver of Roe (cf. Section 1.5.3) with Minmod-limiter, thereby allowing a direct comparison to the serial SAMR code AMRClaw (coded in Fortran 77). The domain is $\Omega = [0,1]^2$ and ideally reflective wall boundary conditions are used at all sides (cf. [32] for a discussion of typical boundary conditions for Euler equations). A circle of uniform high density and pressure, $p_s = 5$ and $\rho_s = 5$, respectively, of radius $r = 0.3$ is positioned at $(0.4, 0.4)$. The ambient density and pressure are $\rho_0 = 1$ and $p_0 = 1$, and the velocities are $u_{1,0} = u_{2,0} = 0$ everywhere. After a few time steps, the initial discontinuity separates into a rapidly expanding discontinuous shock wave, a following slower contact discontinuity and a collapsing smooth rarefaction wave.

The computations use a base grid of 150^2 cells, and a two-level SAMR refinement with the factors $r_1 = 2$ and $r_2 = 4$ is applied. As adaptation criteria, the scaled gradient of the density with $\epsilon_\rho = 0.4$ and the absolute error threshold $\eta_\rho = 0.1$ are used. The clustering tolerance is $\eta_{tol} = 70\%$ and the buffer zone width is $b = 2$. About 200 root level grid time steps with $C_{\mathrm{CFL}} \approx 0.8$ to $t_{\mathrm{end}} = 0.5$ were computed. A repartitioning of the hierarchy is done only at root level time steps, cf. Section 2.2.4. A standard Linux-Beowulf-cluster of Pentium-III-1 GHz CPUs connected with Fast Ethernet (effective bandwidth $\approx$ 40 MB) was used for the benchmarks. Exemplary results on eight processing nodes are shown in Fig. 16. While the AMROC computation on one node required 152 min, the execution time decreased to 13.9 min on 16 nodes. Ta-

TABLE 3. Parallel benchmark: compute times on P nodes.

Task [%]	P=1	P=2	P=4	P=8	P=16
Update by $\mathcal{H}^{(\cdot)}$	86.6	83.4	76.7	64.1	51.9
Flux correction	1.2	1.6	3.0	7.9	10.7
Boundary setting	3.5	5.7	10.1	15.6	18.3
Recomposition	5.5	6.1	7.4	9.9	14.0
Misc.	4.9	3.2	2.8	2.5	5.1
Time [min]	**151.9**	**79.2**	**43.4**	**23.3**	**13.9**
Efficiency [%]	**100.0**	**95.9**	**87.5**	**81.5**	**68.3**

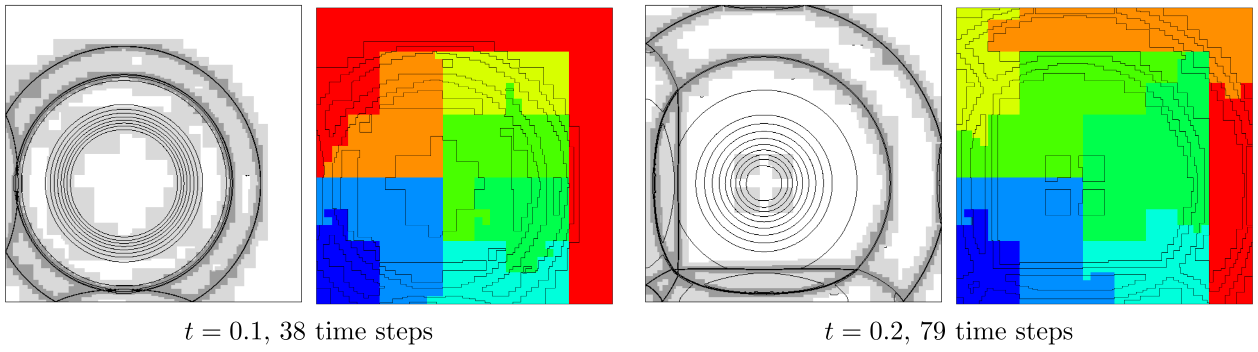


FIGURE 16. Parallel benchmark: circular Riemann problem in an enclosed box. Isolines of density on two refinement levels (indicated by gray scales) and distribution to eight nodes (indicated by different colors).

ble 3 shows a breakdown of the computational time for the most important SAMR operations. Parallel efficiency is decent, but levels off for larger node count, which is to be expected given the conventional network used and the moderate size of the test problem. For one node, the fractions spent in different parts of the code are in good agreements with the results in [14] and our overall mixed C++/Fortran 77 code is about 14.5% slower on one node than AMRClaw, which is a small prize to pay for the generality of parallelization and dimension independence.

2.5.3. *Three-dimensional refinement benchmark*

As an example for a suitable large-scale three-dimensional SAMR benchmark, we discuss results for a Sedov problem [103]. Similar point explosions are prototypical for supernovae and are commonly used to test the hydrodynamics of astro-physical simulation codes. The detailed configuration was suggested as a hyperbolic benchmark for SAMR implementations at the 2003 Chicago workshop on AMR methods,[17] and the shown AMROC results have been obtained for this purpose on the provided benchmark machine, an SGI Altix 3000 shared memory system.

The equations solved are again the Euler equations of a polytropic gas and we utilize the second-order accurate Wave Propagation Method, but the approximate Riemann solver is now blended between the HLL and the Roe scheme for robustness (cf. Section 3.3.5). The computational domain is $\Omega = [-1,1]^3$ with outflow boundary conditions at all sides. The ambient density is set to $\rho_0 = 1$, the ambient pressure to $p_0 = 10^{-5}$, and all velocities are zero everywhere. The explosion is initiated by increasing the energy (and thereby the pressure) uniformly in a sphere of radius r centered in the origin. The integral energy of the entire sphere is set to 1, which yields $e = \left(\frac{4}{3}\pi r^3\right)^{-1}$ for the specific internal energy and $p = (\gamma-1)e$ with $\gamma = 1.4$ for the pressure. The sphere radius is set to be four cell widths of the highest available resolution, i.e., $r = 4\Delta x_{l_{\max}}$. The refinement criterion is

$$\frac{\max\{w_{j\pm1,k,m}, w_{j,k\pm1,m}, w_{j,k,m\pm1}\}}{\min\{w_{j\pm1,k,m}, w_{j,k\pm1,m}, w_{j,k,m\pm1}\}} - 1 > 0.1 \tag{101}$$

applied to density and pressure. The used SAMR base grid is 32^3 and computations are carried out for $l_{\max} = \{2,\ldots,5\}$, all refined by a factor $r_l = 2$, which is equivalent to unigrid computations on 128^3, 256^3, 512^3, and 1024^3 meshes. Defined as part of the benchmark, the cluster tolerance is $\eta_{tol} = 85\%$ and the buffer zone width is $b = 1$. The final time of all computations is $t_{\text{end}} = 0.05$.

Details of the mesh refinement for all adaptive computations for AMROC are given in Table 4. The simulations were run on eight cores of the SGI Altix 3000 server, where communication between cores was still done

[17]http://flash.uchicago.edu/amr2003/bsession.html

Table 4. Three-dimensional refinement benchmark: number of grids and cells for the SAMR simulation and compute times on 8 cores of an SGI Altix 3000 server. Note that the overall compute times for the two highest resolved unigrid configurations were estimated by timing only a small number of actual time steps.

l	$l_{\max} = 2$		$l_{\max} = 3$		$l_{\max} = 4$		$l_{\max} = 5$	
	Grids	Cells	Grids	Cells	Grids	Cells	Grids	Cells
0	28	32,768	28	32,768	33	32,768	34	32,768
1	8	32,768	14	32,768	20	32,768	20	32,768
2	63	115,408	49	116,920	43	125,680	50	125,144
3			324	398,112	420	555,744	193	572,768
4					1405	1,487,312	1,498	1,795,048
5							5,266	5,871,128
$\sum$		180,944		580,568		2,234,272		8,429,624

Time steps	15	27	52	115
SAMR [min]	0.5	5.1	73.0	2,100
Uni [min]	5.4	160	5,000	180,000
Savings	11	31	69	86

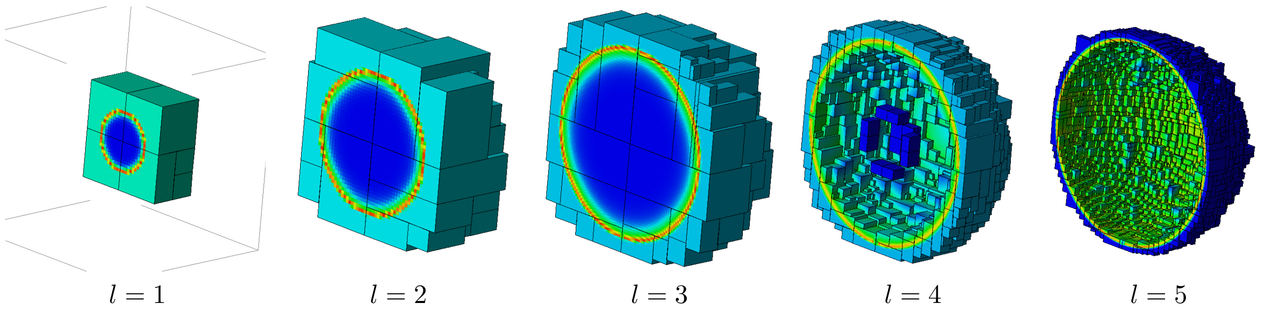


Figure 17. Color plot of the density shown on the subgrids of all five adaptation levels for the case $l_{\max} = 5$.

through MPI. Figure 17 shows the refinement grids for the highest resolved case with $l_{\max} = 5$. The successively smaller subgrids are visible, and from Table 4 it can be inferred that the average subgrid had roughly 1,000 to 2,000 cells, which is significantly larger than in a structured but cell-based refinement approach (cf. Section 1.6.2). Table 4 also shows the time steps taken (the variations are due to the resolution-dependent initial conditions) and the wall clock time to complete the computation. For comparison, the compute times for equivalent unigrid computations (uni) is also given and used to evaluate the savings factor due to the usage of SAMR. Note that the savings factor is smaller than the ratio of cells in the SAMR computation versus the unigrid case as the mesh (re-)organization costs (see also Table 3) increase for deeper refinement hierarchies.

3. Complex hyperbolic SAMR applications

3.1. Non-Cartesian boundaries

Our previous description of schemes and algorithms was restricted to the purely Cartesian case, yet technically relevant flow computations usually take place in non-Cartesian domains. If a suitable geometric mapping is available, topologically structured grids can still be applied and the SAMR algorithms of Section 2 can be used without modification. Only the single-grid discretization and the grid-wise applied inter-level transfer operators need to consider the mapping. The utilization of structured but non-Cartesian meshes is, for instance, common for solving flow problems around bodies for which the resolution of the viscous boundary layer is important, cf. [121]. A refinement of this approach is the overlapping mesh (aka *Chimera*) technique [83] that uses boundary-aligned but structured meshes near bodies and Cartesian grids in the far-field. Data transfer in an overlap area is done via (usually non-conservative) interpolation operations. The approach can also be combined with structured mesh adaptation [58] and is very capable for simulating moving objects in viscous flow. However, for inviscid flows governed by Euler equations or very slow viscous flows immersed boundary approaches are generally more efficient. Immersed boundary techniques utilize a strictly Cartesian mesh on which non-Cartesian boundary conditions are enforced (see also [99] and [114]). Strictly local mesh generation is inherently part of the approach and cumbersome global meshing operations are avoided.

One differentiates between cut-cell techniques that resolve the embedded boundary sharply and techniques that diffuse the boundary location within one cell [84]. A typical cut-cell mesh in two space dimensions with linear boundary approximation in each cell is depicted in Fig. 18. The approach considers the accurate discrete cell area and edge lengths and is strictly conservative. A canonical problem are very small cells that would constrain a global CFL-conditions severely. Cell merging [94], as depicted in Fig. 18, is a common solution approach to this issue; elaborate flux redistribution techniques have also been proposed, cf. [18,93]. The boundary representation in a cut-cell techniques can be explicit [3], but can also be implicit and based on a scalar level set function [85,87].

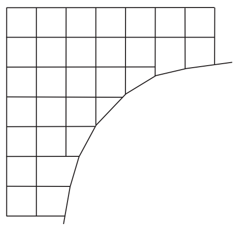


FIGURE 18. Cut-cell mesh.

3.2. An embedded boundary method

Due to the involved complexity we have decided against a cut-cell technique and opted to implement a generic diffused boundary representation technique in AMROC V2.0. The boundary geometry is mapped onto the Cartesian mesh by employing a scalar level set function φ that stores the signed distance to the boundary surface. The boundary itself is represented by the isosurface at $\varphi = 0$ [104]. Geometrically complex moving boundaries are then considered within the Cartesian upwind scheme outlined above by using some of the finite volume cells as ghost cells to enforce immersed boundary conditions, cf. [45]. Utilization of a differentiable level set function allows the evaluation of the boundary outer normal in every mesh point as $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$ [89]. We consider a cell as interior if the distance in the cell *midpoint* is positive and it is treated as exterior otherwise. The numerical stencil by itself is not modified, which causes a diffusion of the boundary location throughout the method and results in an overall non-conservative scheme. The boundary undergoes a staircase approximation but by refining the embedded boundary, typically up to the highest available resolution, we alleviate these problems effectively. A refinement criterion based on $\varphi = 0$ has been implemented to ensure highest level refinement when required (see also Fig. 26).

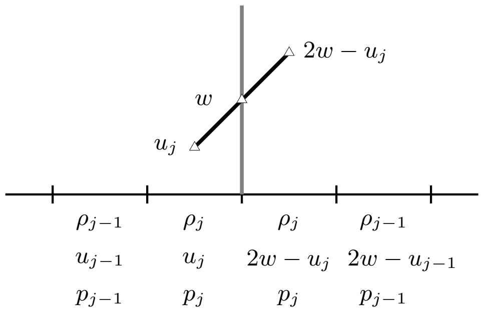
For the inviscid Euler equations, Eq. (49), the boundary condition at a rigid wall moving with velocity $\mathbf{w}$ is $\mathbf{u}\cdot\mathbf{n} = \mathbf{w}\cdot\mathbf{n}$. Enforcing the latter with ghost cells, in which the discrete values are located in the cell centers, requires the mirroring of the primitive values ρ, $\mathbf{u}$, p across the embedded boundary. The normal velocity in the ghost cells is set to $(2\mathbf{w}\cdot\mathbf{n} - \mathbf{u}\cdot\mathbf{n})\mathbf{n}$, while the mirrored tangential velocity remains unmodified. The construction of the velocity vector within the ghost cells therefore reads

$$\mathbf{u}' = (2\mathbf{w}\cdot\mathbf{n} - \mathbf{u}\cdot\mathbf{n})\mathbf{n} + (\mathbf{u}\cdot\mathbf{t})\mathbf{t} = 2\left((\mathbf{w}-\mathbf{u})\cdot\mathbf{n}\right)\mathbf{n} + \mathbf{u} \tag{102}$$

with $\mathbf{t}$ denoting the boundary tangential. This construction of discrete values in ghost cells (indicated by gray) for our model is depicted in one space dimension in Fig. 19.

The utilization of mirrored ghost cell values in a ghost cell center $\mathbf{x}$ requires the calculation of spatially interpolated values in the point

$$\tilde{\mathbf{x}} = \mathbf{x} + 2\varphi\mathbf{n} \tag{103}$$



$2w - u_j$
w
u_j
ρ_{j-1} ρ_j ρ_j ρ_{j-1}
u_{j-1} u_j $2w - u_j$ $2w - u_{j-1}$
p_{j-1} p_j p_j p_{j-1}

FIGURE 19. Moving wall boundary conditions for Euler equations in one space dimension.

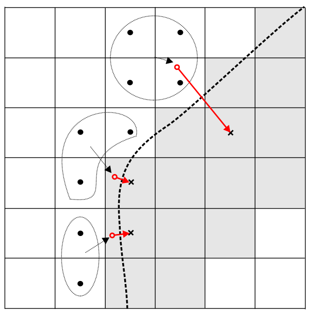


FIGURE 20. Interpolation from interior cells to ghost cells (gray).

from neighboring interior cells. For instance in two space dimensions, we employ the monotonicity-preserving bilinear interpolation Eq. (76) to interpolate values at locations $\tilde{\mathbf{x}}$ within the rectangle formed by four neighboring cell midpoints. Directly near the boundary the number of interpolants in Eq. (76) needs to be decreased, cf. Fig. 20. The interpolation locations according to Eq. (103) are indicated in Fig. 20 by the origins of the red arrows.

3.2.1. *Verification*

As a verification test for the proposed embedded boundary scheme, we consider a Mach reflection problem in physical quantities. A planar two-dimensional shock wave in nitrogen impinges on a ramp inserted into the shocktube at angle $\alpha = 43^o$ and creates a growing transitional Mach reflection pattern. We model the ramp with a level set function and embedded wall boundary conditions, as described in the previous section. For comparison, the computation is repeated in a strictly Cartesian domain, yet the shock wave is now rotated.

The shocktube is filled with nitrogen ($\gamma = 1.4$) at rest ($u_{1,0} = u_{2,0} = 0$) at room temperature (273 K) and ambient pressure $p_0 = 2$ kPa, leading to a density of $\rho_0 = 0.11450\,\mathrm{kg/m}^3$ and speed of sound $c_0 = 494.51\,\mathrm{m/s}$. The shock wave travels at Mach number $M = 2.38$, which leads to the constant state $\rho_s = 0.03642\,\mathrm{kg/m}^3$, $u_{n,s} = 805.16\,\mathrm{m/s}$, $u_{t,s} = 0$, and $p_s = 12.827$ kPa behind the shock. Herein, $u_{n,s}$ and $u_{t,s}$ denote the component of the velocity vector normal and tangential to the shock front, respectively. For the simulations with embedded boundary we use the computational domain $[-4\,\mathrm{mm}, 22\,\mathrm{mm}] \times [0, 22\,\mathrm{mm}]$; in the strictly Cartesian case the domain is $[-4\,\mathrm{mm}, 32\,\mathrm{mm}] \times [0, 16\,\mathrm{mm}]$. The shock front is always initially located at $(0, 0)$ with the shock propagating to the right. In the embedded boundary case, we use constant inflow boundary conditions at the left boundary, reflective wall boundary conditions at the bottom, and outflow boundary conditions elsewhere. In the strictly Cartesian case, constant inflow boundary conditions are used at the left boundary, at the bottom for $x_1 \le 0$ and at the upper boundary behind the shock wave. For $x_1 > 0$ and $x_2 = 0$, reflective wall boundary conditions are used. Outflow boundary conditions are applied at the right and at the upper boundary ahead of the shock wave, requiring the implementation of a time-dependent boundary condition based on the constant shock velocity in the x_1-direction, $v_1 = Mc_0 \cos\alpha$.

We simulate both problems with the Roe scheme of Section 1.5.3, where we use the MUSCL-Hancock extrapolation technique with MinMod-limiter plus the second-order multi-dimensional correction terms of the Wave Propagation scheme (cf. last paragraph of Section 1.4.3). In both cases, we use three additional levels of SAMR with refinement factors $r_0 = 2$, $r_{1,2} = 4$, where adaptation is based on $\eta_\rho = 10^{-3}$ and additionally $\varphi = 0$ in the embedded boundary case. The strictly Cartesian computation uses a base mesh of 360×160 cells; the one with embedded boundary method employs a base mesh of 390×330 cells. The finest resolution along the ramp boundary is $\Delta x_1 = 3.125\mu$m and $\Delta x_e = 2.849\mu$m, respectively. The strictly Cartesian run required 20,768 finest level time steps, the one with embedded boundary method 33,984. The results at $t_{\mathrm{end}} = 17.03\,\mu$s are shown in Fig 21. The qualitative agreement of the Mach reflection patterns is excellent, with all discontinuities predicted at the same locations. An enlargement of just the triple point pattern is shown in Fig. 22. The

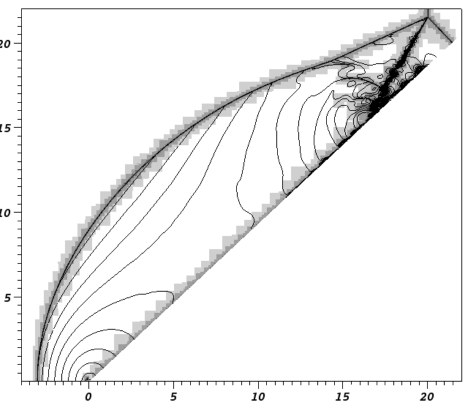
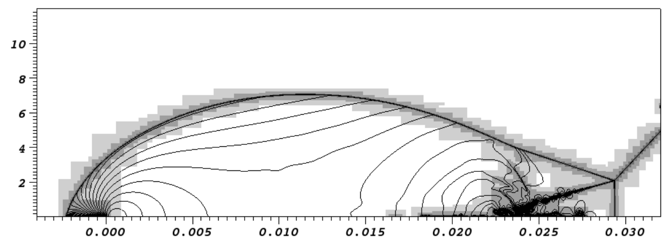



Figure 21. SAMR Simulation of a transitional Mach reflection pattern. Left: embedded boundary method, right: strictly Cartesian setup. Contour lines of ρ on domains of refinement (gray).

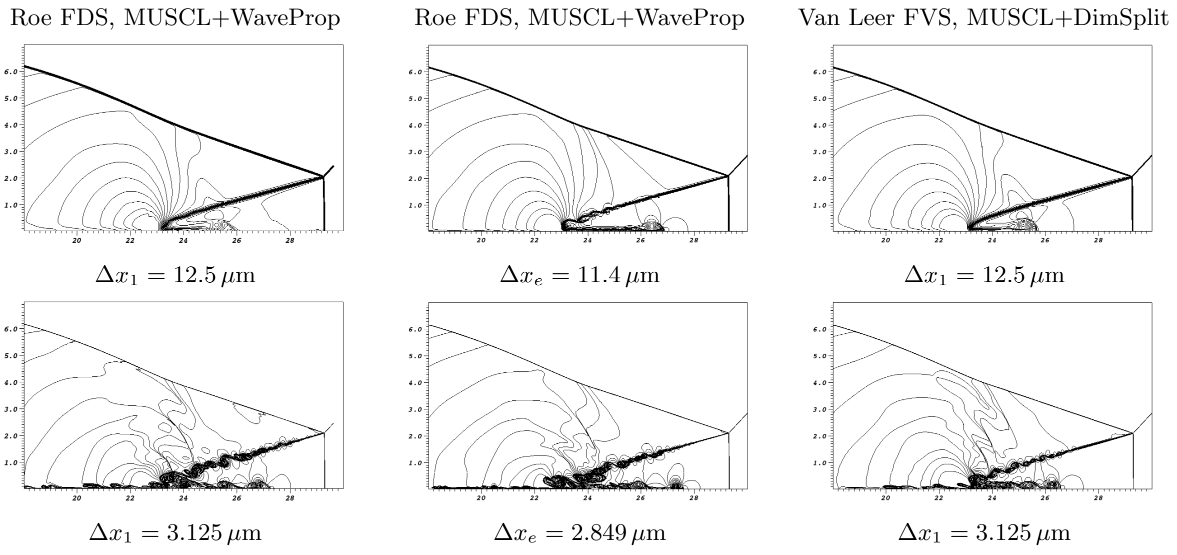


Figure 22. Enlargement of the Mach reflection pattern using two (upper row) and three (lower row) additional level of SAMR refinement. Contour lines of the density shown.

upper row shows less refined results computed without using the finest level. The right column shows results for the strictly Cartesian SAMR setup but using the Van Leer FVS (MUSCL reconstruction, MinMod-mimiter) with dimensional splitting. As can be inferred particularly from the vortices along the slip line (the contact discontinuity emanating from the triple point onto the ramp boundary), which is Kelvin-Helmholtz unstable for Euler equations, a change of the numerical scheme has larger influence on the solution than utilizing our embedded boundary method. Although the proposed boundary representation technique is not strictly conservative, test cases (not included here) exhibit first-order convergence in the L_1-error norm, which makes our approach well applicable to technically relevant computations. Validation results that compare SAMR simulations using the proposed embedded boundary technique directly to experimental results can be found for instance in [71] and [20].

3.2.2. *Implementation*

We have implemented the described embedded boundary method largely independent of a specific single-grid FV scheme within the framework design of AMROC V2.0 [41]. Figure 23 shows the most important classes

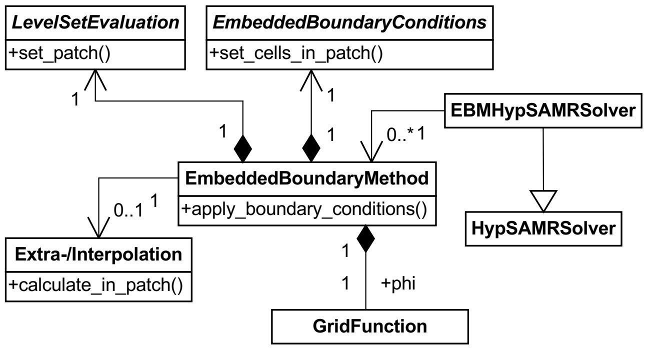


FIGURE 23. Class structure extension of Fig. 14 for level-set-based embedded boundary methods.

that have been added. An abstract class LevelSetEvaluation is provided to evaluate the scalar GridFunction φ patch-wise; EmbeddedBoundaryConditions allows the specification of the detailed boundary value modification. Multiple EmbeddedBoundaryMethods can also be considered and are incorporated with minimal implementation overhead into the existing algorithms of the SAMRSolver-class for hyperbolic problems, HypSAMRSolver, through the derived class EBMHypSAMRSolver. The only operation, that had to be extended, was that of applying physical boundary conditions.

3.3. **Shock-induced combustion**

Dynamic mesh adaptation is often vital for the efficient simulation of chemically reacting flows [37, 96]. Chemistry is modeled as a source term $\mathbf{s}(\mathbf{q})$ in the conservation law, Eq. (1), and the accurate representation of the interaction between nonlinear source and hydrodynamic transport can require very fine local resolutions in space and in time, making the SAMR approach very suitable.

3.3.1. *Euler equations for gas-mixtures*

Chemically reacting flows of premixed inviscid gases are modeled with extended Euler equations. In here, we use a symmetric model of K reactive species with vector of state $\mathbf{q} = (\rho_1, \dots, \rho_K, \rho u_1, \dots, \rho u_d, \rho E)^T$. The partial density of the i-th species is denoted by ρ_i, where $i = 1, \dots, K$. The total density of the mixture is $\rho = \sum_{i=1}^K \rho_i$. The ratios $Y_i = \rho_i/\rho$ are called mass fractions. In analogy to (48), the flux functions of the multi-component model are

$$\mathbf{f}_n(\mathbf{q}) = [\rho_1 u_n, \dots, \rho_K u_n, \rho u_1 u_n + \delta_{1n} p, \dots, \rho u_d u_n + \delta_{dn} p, u_n(\rho E + p)]^T \quad \text{for } n = 1, \dots, d. \tag{104}$$

It is assumed that all species are ideal gases in thermal equilibrium, for which the same temperature T can be used in the partial pressures as $p_i = \mathcal{R} T \rho_i / W_i$, with $\mathcal{R}$ denoting the universal constant and W_i the molecular weight, respectively. According to Dalton's law the total pressure is given by

$$p = \sum_{i=1}^K p_i = \mathcal{R} T \sum_{i=1}^K \frac{\rho_i}{W_i} \ . \tag{105}$$

For the representation of realistic chemical reactions, it is vital to consider temperature-dependent specific heats $c_{pi}(T)$. The specific enthalpies are written as

$$h_i(T) = h_i^0 + \int_{T^0}^T c_{pi}(\sigma) d\sigma \ , \tag{106}$$

with h_i^0 called the heat of formation at the reference temperature T^0. For the enthalpy of the mixture $h = \sum_{i=1}^K Y_i\, h_i(T)$ holds true. Inserting this into the thermodynamic relation $\rho h - \rho e - p = 0$ and inserting Eq. (105) for p yields

$$\sum_{i=1}^{K} \rho_i\, h_i(T) - \rho e - \mathcal{R}T \sum_{i=1}^{K} \frac{\rho_i}{W_i} = 0 \ . \tag{107}$$

Equation (107) has a unique temperature solution [32]. Unfortunately, a closed form of the inverse can only be derived under simplifying assumptions (e.g. $c_{pi} = const.$ for all i in case of polytropic gases) and the iterative computation of T from Eq. (107) (in our case by Newton iteration) is required whenever the pressure p has to be evaluated. Analogous to the case of a single polytropic gas the *frozen* speed of sound is given by $c^2 = \gamma\, p/\rho$. It may be calculated by applying the basic relations $c_p = \sum Y_i\, c_{pi}$, $W = (\sum Y_i/W_i)^{-1}$ and $\gamma = \frac{c_p}{c_p - \mathcal{R}/W}$.

3.3.2. *Reactive source terms*

We write the chemical production of a single species as product of its chemical production rate in molar concentration per unit volume $\dot\omega_i$ and its constant molecular weight W_i. The source term $\mathbf{s}(\mathbf{q})$ then reads

$$\mathbf{s}(\mathbf{q}) = [W_1\, \dot\omega_1, \ldots, W_K\, \dot\omega_K, 0, \ldots, 0, 0]^T \ . \tag{108}$$

The chemical production rates $\dot\omega_i = \dot\omega_i(\rho_1, \ldots, \rho_K, T)$ are derived from a reaction mechanism that consists of J chemical reactions

$$\sum_{i=1}^{K} \nu_{ji}^f S_i \rightleftharpoons \sum_{i=1}^{K} \nu_{ji}^r S_i \ , \quad j = 1, \ldots, J, \tag{109}$$

where ν_{ji}^f and ν_{ji}^r are the stoichiometric coefficients of species S_i, appearing as a reactant and as a product. Note that for a large number of species the majority of the coefficients ν_{ji}^f, ν_{ji}^r is usually zero in most reactions. The entire molar production rate of species S_i per unit volume is then given by

$$\dot\omega_i = \sum_{j=1}^{J} (\nu_{ji}^r - \nu_{ji}^f) \left[k_j^f \prod_{l=1}^{K} \left(\frac{\rho_l}{W_l} \right)^{\nu_{jl}^f} - k_j^r \prod_{l=1}^{K} \left(\frac{\rho_l}{W_l} \right)^{\nu_{jl}^r} \right], \ i = 1, \ldots, K \ , \tag{110}$$

with $k_j^f(T)$ and $k_j^r(T)$ denoting the forward and backward reaction rate of each chemical reaction. The reaction rates are calculated by the Arrhenius law $k_j^{f/r}(T) = A_j^{f/r} T^{\beta_j^{f/r}} \exp(-E_j^{f/r}/\mathcal{R}T)$. The computations shown in here were all produced with the same hydrogen-oxygen reaction mechanism proposed within a larger hydrocarbon mechanism [119]. The employed subset consists of 34 elementary reactions and considers the nine species H, O, OH, H_2, O_2, H_2O, HO_2, H_2O_2 and Ar. The computationally expensive reaction rate expressions (110) are evaluated by a loop-free, reaction-mechanism-specific Fortran 77 function, which is produced by a source code generator built on top of the Chemkin-II library [65].

3.3.3. *Used schemes*

Chemically reactive flows are ideal candidates for the method of fractional steps introduced in Section 1.2.2. The fractional step approach allows the separate numerical integration of the homogeneous Euler equations with an explicit FV method and the application of a time-implicit discretization for the typically stiff ODE resulting from the chemical sources according to (10) separately in each grid cell. In here, we employ a semi-implicit Rosenbrock-Wanner method of fourth order with automatic step-size adjustment [64].

The used FV upwind operator is an extension of the linearized Riemann solver of Roe (cf. Section 1.5.3) to multiple thermally perfect species [35, 37]. In analogy to Eqs. (56-58), the waves $\mathcal{W}_m := a_m \hat{\mathbf{r}}_m$ are computed as [51]

$$a_{1,K+d+1} = \frac{\Delta p \mp \hat\rho \hat c \Delta u_1}{2\hat c^2} \ , \ \hat{\mathbf{r}}_{1,K+d+1} = \left[\hat Y_1, \ldots, \hat Y_K, \hat u_1 \mp \hat c, \hat u_2, \ldots, \hat u_d, \hat H \mp \hat u_1 \hat c \right]^T ,$$

$$a_{i+1} = \Delta\rho_i - \hat{Y}_i \frac{\Delta p}{\hat{c}^2}\ ,\ \hat{\mathbf{r}}_{i+1} = \left[\delta_{1i}, \dots, \delta_{Ki}, \hat{u}_1, \hat{u}_2, \dots, \hat{u}_d, \hat{\mathbf{u}}^2 - \frac{\hat{\phi}_i}{\hat{\gamma}-1}\right]^T ,$$

$$a_{K+n} = \hat{\rho}\Delta u_n\ ,\ \hat{\mathbf{r}}_{K+n} = [0, \dots, 0, 0, \delta_{2n}, \dots, \delta_{dn}, \hat{u}_n]^T \quad \text{for } n = 2, \dots, d.$$

As before, $\hat{v}$ denotes the Roe average, Eq. (54). Specific to the thermally perfect model are the averages

$$\hat{\gamma} := \frac{\hat{c}_p}{\hat{c}_v} \quad \text{with} \quad \hat{c}_{\{p/v\}} = \sum_{i=1}^{K} \hat{Y}_i \hat{c}_{\{p/v\}i}\ , \quad \hat{c}_{\{p/v\}i} = \frac{1}{T_r - T_l} \int_{T_l}^{T_r} c_{\{p,v\}i}(\tau)\, d\tau$$

and

$$\hat{\phi}_i := (\hat{\gamma}-1)\left(\frac{\hat{\mathbf{u}}^2}{2} - \hat{h}_i\right) + \hat{\gamma}\,\frac{\mathcal{R}}{W_i}\,\hat{T}\ , \quad \hat{c} := \left((\hat{\gamma}-1)(\hat{H} - \hat{\mathbf{u}}^2) + \sum_{i=1}^{K} \hat{Y}_i \hat{\phi}_i\right)^{1/2} ,$$

where the linearized eigenvalues now read $\hat{\lambda}_{1,K+d+1} = \hat{u}_1 \mp \hat{c}$ and $\hat{\lambda}_{i+1} = \hat{\lambda}_{K+n} = \hat{u}_1$ for $i = 1, \dots, K$ and $n = 2, \dots, d$. For the detailed derivation, see [32].

Since we are interested in the simulation of combustion, induced by strong shock waves, the approximate Riemann solver also needs to be stabilized against the so called *carbuncle phenomenon*, a multi-dimensional numerical crossflow instability that occurs at strong grid-aligned shocks or detonation waves [95]. The carbuncle phenomenon can be avoided completely by applying Eq. (59) to all characteristic fields and evaluating η in a multi-dimensional way. We utilize the "H-correction" by Sanders et al. [101] for this purpose. For instance, in the x_2-direction between the cells (j,k) and $(j,k+1)$ it takes the form

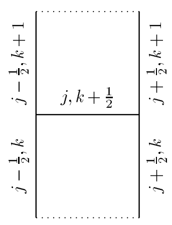


FIGURE 24. H-correction.

$$\tilde{\eta}_{j,k+\frac{1}{2}} = \max\left\{\eta_{j+\frac{1}{2},k}, \eta_{j-\frac{1}{2},k}, \eta_{j,k+\frac{1}{2}}, \eta_{j-\frac{1}{2},k+1}, \eta_{j+\frac{1}{2},k+1}\right\} \tag{111}$$

in the two-dimensional case (see Fig. 24). As mentioned in Section 1.5.3, Roe-type schemes are not guaranteed to be positivity-preserving. To ensure positivity of the mass fractions Y_i we apply the correction [70]

$$\mathbf{F}_i^{\star} = \mathbf{F}_\rho \cdot \begin{cases} Y_i^l\ , & \mathbf{F}_\rho \geq 0\ , \\ Y_i^r\ , & \mathbf{F}_\rho < 0\ . \end{cases} \tag{112}$$

with $\mathbf{F}_\rho := \sum_{i=1}^{K} \mathbf{F}_i$ after evaluating the numerical flux according to Eq. (29). Finally, the linearized Riemann solver is extended to a high-resolution multi-dimensional method with the MUSCL-Hancock reconstruction technique (using primarily limiter (39)) and dimensional splitting (cf. Section 1.2.2). We recommend to apply the MUSCL extrapolation to ρ, p, Y_i and ρu_n and to derive a thermodynamically consistent extrapolated vector of state from these [32].

3.3.4. *Shock-induced combustion around a sphere*

As an example for combustion in hypersonic flow, we consider the shock-induced ignition of a combustible mixture by the bow shock ahead of a supersonic projectile [36]. If the speed of the projectile is sufficiently large, the temperature in the stagnation point will exceed the autoignition temperature of the mixture and combustion occurs. In here, we consider a test case suggested by Hung [61] of a small spherical projectile traveling with $v_I = 2170.6\,\mathrm{m/s}$ through a hydrogen-oxygen-argon mixture (molar ratios 2:1:7) at $p_0 = 6.67\,\mathrm{kPa}$ and $T_0 = 298\,\mathrm{K}$. The simplicity of initial conditions and setup, steadiness of the solution, and moderate resolution requirements make this an excellent verification case for numerical methods.

The computation is carried out under the assumption of cylindrical symmetry in the frame of reference of the projectile. A two-dimensional domain of $[-4.0\,\mathrm{mm}, 13.5\,\mathrm{mm}] \times [0, 10.0\,\mathrm{mm}]$ is used, where the axis of symmetry is aligned with the x_1-axis. The non-differential terms arising additionally in the Euler equations under cylindrical coordinate transformation are considered in the originally two-dimensional Cartesian method in a splitting approach, in which the arising ODE is integrated with a 2-stage Runge-Kutta scheme at the end of

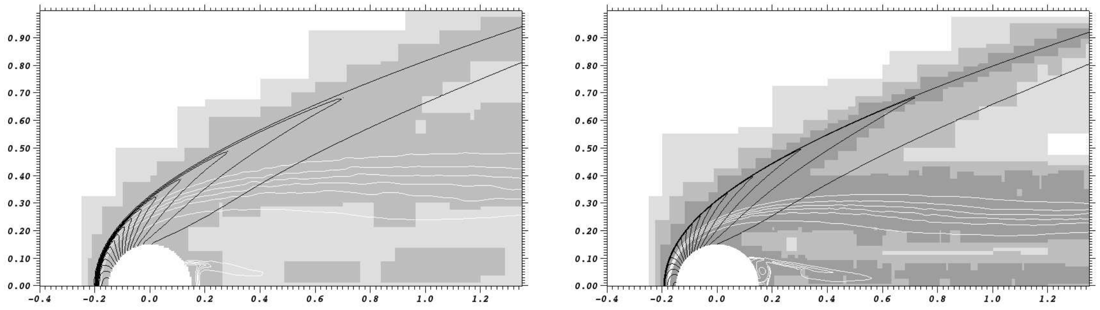


Figure 25. Iso-contours of p (black) and Y_{H_2} (white) on domains of different refinement levels (gray) at $t = 350\,\mu$s for a 3-level (left) and a 4-level computation (right).

Table 5. Refinement indicator values used for projectile-induced combustion simulations.

Y_i	$S_{Y_i} \cdot 10^{-4}$	$\eta^r_{Y_i} \cdot 10^{-4}$
O_2	10.0	4.0
H_2O	5.8	3.0
H	0.2	10.0
O	1.4	10.0
OH	2.3	10.0
H_2	1.3	4.0

$\epsilon_\rho = 0.02\,\mathrm{kg\,m^{-3}}$, $\epsilon_p = 16\,\mathrm{kPa}$

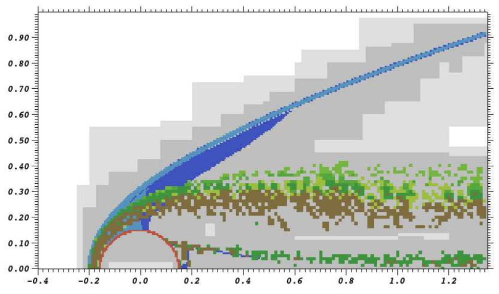


Figure 26. Refinement indicators on $l = 2$ at $t = 350\,\mu$s. Blue: ϵ_ρ, light blue: ϵ_p, green shades: $\eta^r_{Y_i}$, red: embedded boundary.

$\mathcal{S}^{(\cdot)}$, cf. [76]. The midpoint of the sphere is located at $(0, 0)$ and the radius is 1.5 mm. While outflow boundary conditions are applied at the upper and the right boundary, inflow boundary conditions are used at the left domain side. The inflow velocity is ramped with constant acceleration from zero to v_I during the first $10\,\mu$s of the simulation. A steady flow situation of a shock-induced combustion front separating gradually from the expanding bow shock develops quickly. We simulate $400\,\mu$s which corresponds to a flight of 85.7 cm distance.

We discuss dynamically adaptive simulations on a base mesh of 70×40 cells. We employ a physically motivated combination of the refinement indicators described in Section 2.3. The scaled gradients of total density ρ and total hydrodynamic pressure p are used to achieve adaptation to the bow shock; the combustion front is captured by applying the relative error criterion (93) to the mass fractions Y_i of the most relevant chemical species. Figure 25 shows a comparison between a 3-level computation with $r_{0,1} = 2$ and a 4-level computation with $r_{0,1} = 2$ and $r_2 = 4$ at $t = 350\,\mu$s. As it is characteristic for chemically reactive flows with hydrodynamic discontinuities, the bow shock around the projectile is captured at the same location in both computations, yet the combustion zone is visibly more diffused and appears at a slightly different position in the coarser computation. A visualization of the type of refinement criteria leading to the highest level refinement in the right graphic of Fig. 25 is depicted in Fig. 26. At the time shown, this computation uses 223,068 cells on the finest level and 255,914 total, where a uniformly refined computation would use 716,800 cells.

3.3.5. *Detonation defraction*

In order to demonstrate that the SAMR approach is also well suited for cutting-edge combustion simulations (see also [24]), we briefly present exemplary results for a two-dimensional hydrogen-oxygen detonation propagating out of a tube into unconfinement [34]. The simulation reproduces the critical width for square

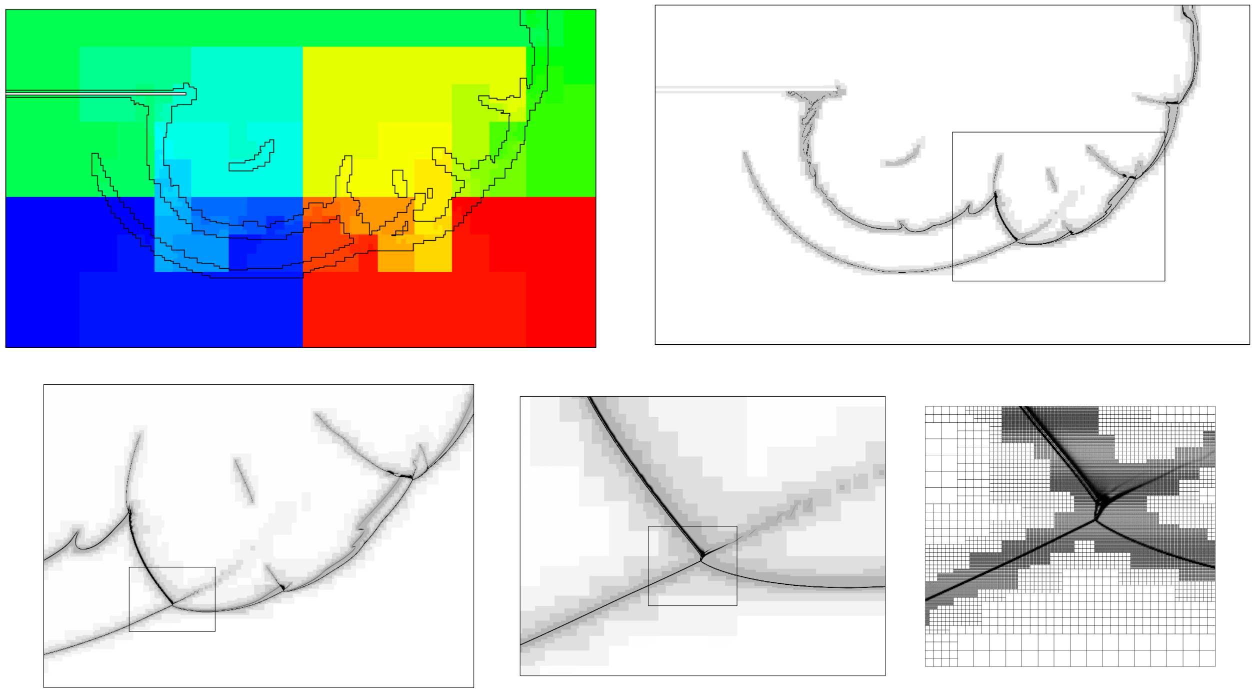


FIGURE 27. Final distribution to 48 nodes and density distribution on four refinement levels, $t = 240\,\mu$s after the detonation has left the tube simulated. Multiple zooms are necessary to display the finite volume cells.

tubes and is in perfect qualitative agreement with experimental results. The computation required ~ 3850 h CPU (~ 80 h wall time) on a cluster of 48 Athlon-1.4 GHz single-processor nodes. As detonation simulations require an extraordinarily high local resolution to capture the influence of the chemical kinetics correctly, the computation benefits remarkably from dynamic mesh adaptation. The graphics in Fig. 27 display the solution on the refinement levels $240\,\mu$s after the detonation has left the tube (730 root level time steps with $C_{CFL} \approx 0.8$, one half of the domain was simulated) and the enormous efficiency of the refinement is apparent. The base grid used 508×288 cells and four levels of refinement with $r_{1,2,3} = 2$, $r_4 = 4$, which corresponds to ~ 150 M cells, yet at the time step displayed, the simulation uses less than 3.0 M cells on all levels.

A crucial modification of the approximate Riemann solver of Section 3.3.3 to enable this detonation defraction simulation is to replace the numerical flux at a few cell interfaces near the edge of the tube with the strictly positive but highly diffusive HLL flux [43] when the detonation is exiting. Due to shock wave diffraction, a near-vacuum situation occurs below the tip that causes the linearized Riemann solver to fail (cf. Section 1.5.3). We have found that this switching can effectively and reliably be accomplished by evaluating the intermediate states $\mathbf{q}_l^\star = \mathbf{q}_l + \mathcal{W}_1$ and $\mathbf{q}_r^\star = \mathbf{q}_r - \mathcal{W}_{K+d+1}$ in the linearized Riemann problem and to use the HLL flux if $\rho_{l/r}^\star \le 0$ or $e_{l/r}^\star \le 0$.

3.4. **Fluid-structure interaction**

The final type of complex SAMR applications considered in here is the simulation of shock-driven fluid-structure interaction (FSI) problems. The objective is to dynamically couple an explicit, shock-capturing FV scheme to an explicit solid mechanics solver. Large structural deformations, fracture and even fragmentation, might have to be considered. Our approach to this problem is to employ the level-set-based embedded boundary method of Section 3.2 in combination with SAMR for this problem class [26, 39, 40, 42].

3.4.1. *Coupling approach*

Fluid-structure interaction is assumed to take place only at the evolving interface $\mathcal{I}$ between fluid and solid and is implemented numerically by exchanging boundary data after consecutive time steps. In the case of inviscid flows, the boundary conditions along $\mathcal{I}$ are

$$v_n = u_n \ , \quad \sigma_{nn} = p \ , \tag{113}$$

with v_n and u_n denoting the velocity in the normal direction in solid and fluid, respectively. The solid stress tensor is denoted by σ, σ_{nn} are the normal stresses, and p is the fluid pressure. We accomplish the coupling between fluid and solid solver with the simple fractional step method:

```
$u_n := v_n(t)|_{\mathcal{I}}$
update_fluid( $\Delta t$ )
$\sigma_{nn} := p(t+\Delta t)|_{\mathcal{I}}$
update_solid( $\Delta t$ )
$t := t + \Delta t$
```

Algorithm 6. Basic fluid-structure coupling algorithm.

The adaptive fluid solver with embedded boundary capability receives the velocities and the discrete geometry of the solid surface, while only the hydrostatic pressure is communicated back to the solid solver as a force acting on the solid's exterior [77, 109]. As the inviscid Euler equations can not impose any shear on the solid structure, cf. [5, 44], the fluid pressure is sufficient to prescribe the entire stress tensor on the solid boundary. We have implemented this algorithm with an ad-hoc partitioning into dedicated fluid and solid processes that communicate to exchange the data along a triangulated interface $\mathcal{I}$, cf. [42].

3.4.2. *Fluid-structure coupling with SAMR*

While the implementation of a loosely coupled FSI method is straightforward with conventional solvers with consecutive time update, the utilization of the recursive SAMR method is non-apparent. In our SAMR-FSI software system [41], called *Virtual Test Facility* (VTF), we treat the fluid-solid interface $\mathcal{I}$ as a discontinuity that is a-priori refined at least up to a coupling level $l_{\rm fsi}$. The resolution at level $l_{\rm fsi}$ has to be sufficiently fine to ensure an accurate wave transmission between fluid and structure, but might not necessarily be the highest level of refinement. We formulate the corresponding extension of Algorithm 2 in Algorithm 7, where some of the steps explicitly formulated in Algorithm 2 are suppressed for clarity. The extended version of `advance_level()` calls the routine `level_set_generation()` to evaluate the signed distance φ for the actual level l based on the currently available interface $\mathcal{I}$. Together with the recent solid velocity on the interface $\mathbf{v}|_{\mathcal{I}}$, the discrete vector of state in the fluid $\mathbf{Q}$ is updated for the entire level. The method then proceeds recursively to higher levels and utilizes the (more accurate) data from the next higher level to correct cells overlaid by refinement. If level l is the coupling level $l_{\rm fsi}$, we use an updated fluid data field to evaluate the pressure on the discrete vertices of $\mathcal{I}$, which is sent to the solid and to receive updated mesh positions and nodal velocities. The recursive order of the SAMR algorithm automatically ensures that updated interface mesh information is available at later time steps on coarser levels and to adjust the grids on level $l_{\rm fsi}$.

In order to achieve a proper matching of communication operations, we start the cycle by posting a receive-message in the routine `fluid_step()` (which does one fluid time step on level 0) before entering into the SAMR recursion. The routine `fluid_step()` below highlights a straightforward automatic time step adjustment for the SAMR method coupled to a solid solver. During one root level time step at level 0, the time steps on all levels remain fixed and are calculated in advance by employing the refinement factor with respect to the root level $R_l = \prod_{\iota=0}^{l} r_l$. The root level time step $\Delta\tau_F$ itself is taken to be the minimum of the stable time step estimations from all levels and a corresponding time step $\Delta\tau_S$ in the solid. We define $\Delta\tau_S$ as a multiple of the stable time step estimation in the solid solver with respect to the communication frequency $R_{l_{\rm fsi}}$ in one fluid root level step and an additional factor K that allows sub-iterations in the solid solver in case of considerably

```
advance_level($l$)
  Repeat $r_l$ times
    If time to regrid
      regrid($l$)
    level_set_generation($\varphi^l$, $\mathcal{I}$)
    update_fluid_level($\mathbf{Q}^l$, $\varphi^l$, $\mathbf{v}|_\mathcal{I}$, $\Delta t_l$)
    If level $l+1$ exists
      advance_level($l+1$)
      Correct $\mathbf{Q}^l(t+\Delta t_l)$ with $\mathbf{Q}^{l+1}(t+\Delta t_l)$
    If $l = l_{\rm fsi}$
      send_interface_data($p(t+\Delta t_l)|_\mathcal{I}$)
      receive_interface_data($\mathcal{I}$, $\mathbf{v}|_\mathcal{I}$)
    $t := t+\Delta t_l$
```

Algorithm 7. Fluid-structure coupling with SAMR.

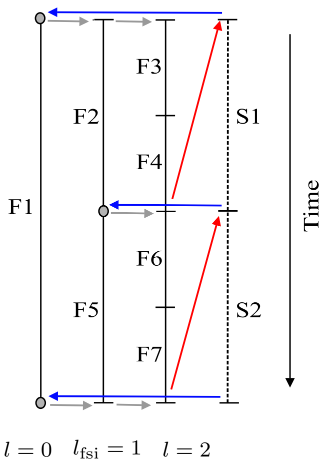


Figure 28. Data exchange between Algorithm 7 and an explicit solid dynamics solver.

smaller solid time steps. The solid update algorithm used to advance the solid by one fluid root level step is given below.

The data exchange between `solid_step()` and `fluid_step()` is visualized in Fig. 28 for an exemplary SAMR hierarchy with two additional levels with $r_{1,2} = 2$. Figure 28 pictures the recursion in the SAMR method by numbering the fluid update steps (F) according to the order determined by `advance_level()`. The order of the solid update steps (S) on the other hand is strictly linear. The red arrows correspond to the sending of the interface pressures $p|_\mathcal{I}$ from fluid to solid at the end of each time step on level $l_{\rm fsi}$. The blue arrows visualize the sending of the interface mesh $\mathcal{I}$ and its nodal velocities $\mathbf{v}|_\mathcal{I}$ after each solid update. The modification of refinement meshes is indicated in Fig. 28 by the gray arrows; the initiating base level, that remains fixed throughout the regridding operation, is indicated by the gray circles.

```
solid_step()
  $\Delta\tau_S := \min(K \cdot R_{l_{\rm fsi}} \cdot$ stable_solid_step(), $\Delta\tau_F)$
  Repeat $R_{l_{\rm fsi}}$ times
    $t_e := t + \Delta\tau_S/R_{l_{\rm fsi}}$, $\Delta t := \Delta\tau_S/(KR_{l_{\rm fsi}})$
    While $t < t_e$
      send_interface_data($\mathcal{I}(t)$, $\vec{u}^S|_\mathcal{I}(t)$)
      receive_interface_data($p^F|_\mathcal{I}$)
      update_solid($p^F|_\mathcal{I}$, $\Delta t$)
      $t := t + \Delta t$
      $\Delta t := \min($stable_solid_step(), $t_e - t)$
```

```
fluid_step()
  $\Delta\tau_F := \min_l(R_l \cdot$ stable_fluid_step($l$), $\Delta\tau_S)$
  $\Delta t_l := \Delta\tau_F/R_l$ for $l = 0, \cdots, l_{\max}$
  receive_interface_data($\mathcal{I}$, $\vec{u}^S|_\mathcal{I}$)
  advance_level(0)
```

Algorithms 8 (left) and 9. Implementation of time stepping when using the recursive SAMR method with FSI coupling.

3.4.3. *Implementation*

The incorporation of the algorithms described above into the AMROC framework is relatively straightforward. Utilizing the design for general embedded boundary methods sketched in Sec. 3.2.2, we have implemented `fluid_step()` and the fluid-structure coupled version of `advance_level()` in a class CoupledHypSAMRSolver derived from EBMHypSAMRSolver (cf. Fig. 29). CoupledHypSAMRSolver interpolates the pressure values $p|_\mathcal{I}$ along the surface mesh and communicates them to CoupledSolidSolver through the coupling module InterSolverCommunication. CoupledHypSAMRSolver receives an updated interface mesh $\mathcal{I}$ that it passes to the

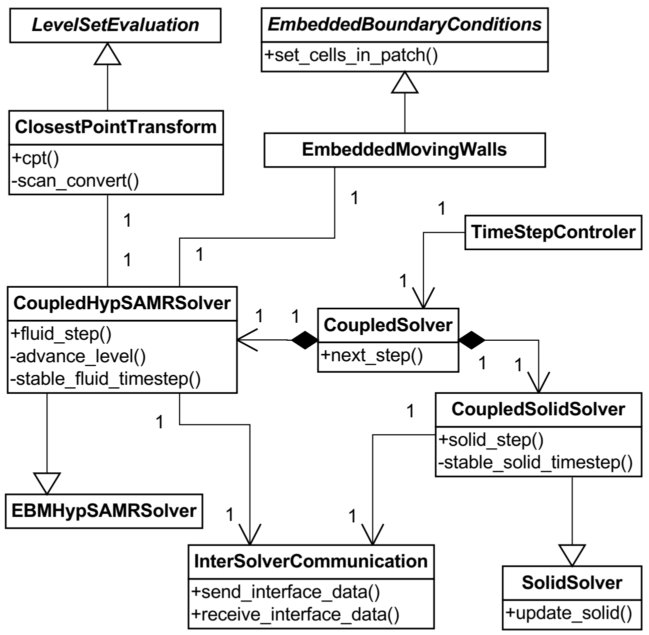


Figure 29. Class structure of the fluid-structure coupling method realized as a concrete embedded boundary method in AMROC, see Fig. 23 for base classes.

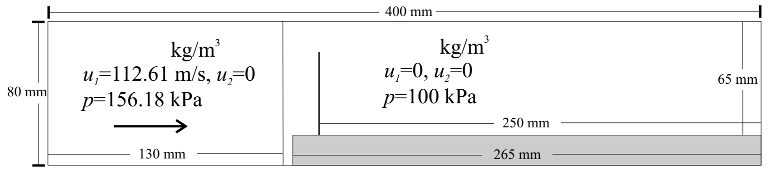
ClosestPointTransform, which is made available as a concrete class based on LevelSetEvaluation. ClosestPoint-Transform contains the implementation of a specially developed algorithm for efficient computation of distance functions [82] that has linear computational complexity both in the number of Cartesian mesh points and the surface triangles considered [42]. Further, CoupledHypSAMRSolver receives updated interface velocities $\mathbf{v}|_{\mathcal{I}}$ to be used in EmbeddedMovingWalls as the necessary concretization of EmbeddedBoundaryConditions. In order to re-use our standard TimeStepControler, we have incorporated CoupledHypSAMRSolver and CoupledSolid-Solver as attributes into a single CoupledSolver that encapsulates the extended method.

3.4.4. *Shock-induced elastic panel deflection*

As a demonstration for our coupling approach, we simulate the quasi two-dimensional verification configuration of a thin-walled steel panel impacted by a planar shock wave in air ($\gamma = 1.4$), proposed by Giordano et al. [47]. The panel has the thickness $h = 1\,\mathrm{mm}$ and extends 50 mm from a mounting with forward-facing step geometry, into which it is firmly clamped. Figure 30 depicts the computational domain and initial conditions. Inflow boundary conditions are applied on the left side, rigid wall boundary conditions anywhere else.

We assume that the fluid domain and the panel extend 5 mm in the x_3-direction. The panel is modeled in the explicit solid mechanics solver DYNA3D[18] [54] simply with ten four-node shell elements. The material is assumed to be linearly elastic with density $\rho_s = 7600\,\mathrm{kg/m^3}$, elasticity modulus $E = 220\,\mathrm{GPa}$ and Poisson ratio $\nu = 0.33$. The panel is embedded into a three-dimensional fluid base mesh of $320 \times 64 \times 2$ cells that allows up to two additional levels of dynamic isotropic refinement (based on φ and scaled gradients of ρ and p) with refinement factors $r_{1,2} = 2$. Beside $u_3 = 0$, all fluid initial conditions are shown in Fig. 30. The fluid solver is the approximate Riemann solver of Roe with MUSCL reconstruction and dimensional splitting. Calculating 19,864 coupled time steps at $l_{\mathrm{fsi}} = 2$ to $t_{\mathrm{end}} = 5.0\,\mathrm{ms}$ required $\sim 450\,\mathrm{h}$ CPU ($\sim 28.2\,\mathrm{h}$ wall time) on sixteen 3.4 GHz processors connected with a GB-Ethernet network. To accommodate for mandatorily smaller time steps in the

[18]Source code (U.S. export controlled) available for licensing fee from http://www.osti.gov/estsc.



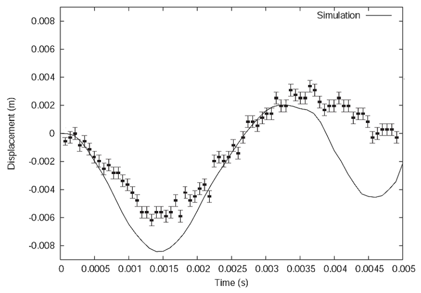


FIGURE 30. Left: geometry setup and fluid initial conditions for the deflecting panel case. Right: panel tip displacement over time in FSI simulation and experiment [47].

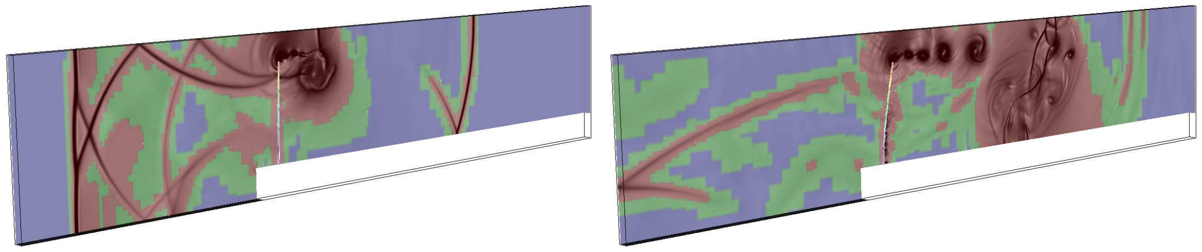


FIGURE 31. Quasi two-dimensional computation of a thin-shell panel hit by a shock wave at $t \approx 0.43$ ms (left) and $t \approx 1.17$ ms (right) after impact. Gray-scale schlieren of fluid density on domains of refinement levels (indicated by color).

solid, $K = 4$ sub-iterations were taken in DYNA3D within one coupled FSI time step. Fifteen processors were dedicated to AMROC, one to the serial DYNA3D code.

Figure 31 visualizes the dynamic bending of the plate strip and the evolving fluid mesh adaptation with two additional levels (indicated by color on the back plane) as the initial shock is partially reflected and increased vortex shedding occurs at the panel tip at later times. In the left snapshot of Fig. 31, the adaptive computation uses $635,264$ cells in 269 subgrids on the finest level. In the right graphic of Fig. 31, $1,295,584$ cells in 305 subgrids are used, which would compare to $1,970,176$ cells in the uniform case. A comparison of the simulated panel tip displacement over time versus the experimental measurements from Giordano et al. [47], shown with error bars, is given in the right graph of Fig. 30. The agreement, especially at later times, is actually better than the computational results in [47], Fig. 10, which is likely due to a significantly finer effective resolution, thanks to the availability of SAMR, in the AMROC fluid solver.

3.4.5. *Modeling of shocks in liquid-gas mixtures*

As final configuration, we discuss briefly the coupled FSI simulation of deforming structures in water due to impinging shock waves created by the explosion of energetic materials [38]. For the fluid solver, we consider a multi-component mixture model based on the volume fractions α^i, with $\sum_{i=1}^m \alpha^i = 1$, that defines the mixture quantities as

$$\rho = \sum_{i=1}^m \alpha^i \rho^i \,, \quad \rho u_n = \sum_{i=1}^m \alpha^i \rho^i u_n^i \,, \quad \rho e = \sum_{i=1}^m \alpha^i \rho^i e^i \,, \quad \frac{p}{\gamma - 1} = \sum_{i=1}^m \frac{\alpha^i p^i}{\gamma^i - 1} \,, \quad \frac{\gamma p_\infty}{\gamma - 1} = \sum_{i=1}^m \frac{\alpha^i \gamma^i p_\infty^i}{\gamma^i - 1} \,, \qquad (114)$$

in which each component satisfies a stiffened gas equation of state of the form $p^i = \left(\gamma^i - 1\right) \rho^i e^i - \gamma^i p_\infty^i$. At this point, several possibilities would exist for deriving different sets of governing transport equations for a two-fluid model, however, we choose to follow the approach of Shyue [106] that supplements system (49) with the two advection equations

$$\frac{\partial}{\partial t}\left(\frac{1}{\gamma-1}\right)+\sum_{n=1}^{d} u_n \frac{\partial}{\partial x_n}\left(\frac{1}{\gamma-1}\right)=0\,,\quad \frac{\partial}{\partial t}\left(\frac{\gamma p_\infty}{\gamma-1}\right)+\sum_{n=1}^{d} u_n \frac{\partial}{\partial x_n}\left(\frac{\gamma p_\infty}{\gamma-1}\right)=0\,. \tag{115}$$

Abgrall [1] proved that a multi-component continuum scheme needs to satisfy Eq. (115.1) in the discrete sense to prevent unphysical oscillations at material boundaries. Although different scheme alterations are possible to satisfy this requirement, cf. [2], the utilization of (115) in the governing equations and therefore direct discretization together with (49) is the simplest remedy to the problem, cf. [106, 107].

Since the equations (115) are not in conservation form, we use the Wave Propagation approach (Section 1.4.3) to discretize the system (49), (115). In here, we use the HLLC[19] scheme by Toro et al. [112] that is tailored specifically for the Euler equations and approximates the RP (here x_1-direction) with three discontinuous jumps by

$$\mathbf{q}^{HLLC}(x_1,t)=\begin{cases}\mathbf{q}_l\,, & x_1 < s_l\,t,\\ \mathbf{q}_l^\star\,, & s_l\,t \le x_1 < s^\star\,t,\\ \mathbf{q}_r^\star\,, & s^\star\,t \le x_1 \le s_r\,t,\\ \mathbf{q}_r\,, & x_1 > s_r\,t,\end{cases} \tag{116}$$

For the wave speeds $s_{l/r}$ we use the estimations $s_l=\min\{u_{1,l}-c_l, u_{1,r}-c_r\}$, $s_r=\max\{u_{1,l}+c_l, u_{1,r}+c_r\}$ suggested by Davis [30] and $s^\star$ is given in the HLLC approach by

$$s^\star=\frac{p_r-p_l+s_l u_{1,l}(s_l-u_{1,l})-\rho_r u_{1,r}(s_r-u_{1,r})}{\rho_l(s_l-u_{1,l})-\rho_r(s_r-u_{1,r})}\,. \tag{117}$$

Conservation arguments and consideration of the structure of the RP for Euler equations lead to the specification of the unknown solution values as

$$\mathbf{q}_k^\star=\left[\eta,\eta s^\star,\eta u_2,\eta\left[\frac{(\rho E)_k}{\rho_k}+(s^\star-u_{1,k})\left(s_k+\frac{p_k}{\rho_k(s_k-u_{1,k})}\right)\right],\frac{1}{\gamma_k-1},\frac{\gamma_k p_{\infty,k}}{\gamma_k-1}\right]^T,\quad \eta=\rho_k\frac{s_k-u_{1,k}}{s_k-s^\star} \tag{118}$$

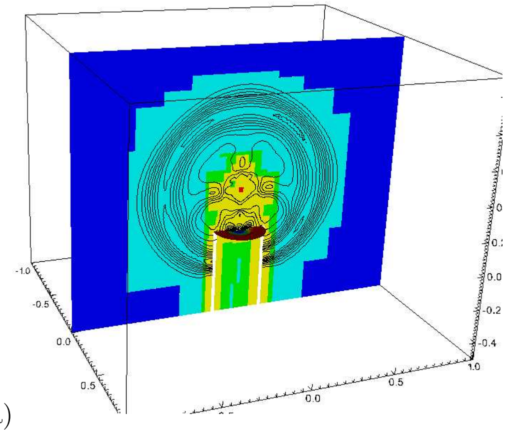
for $k=\{l,r\}$, cf. [111]. Knowledge of the intermediate state then allows the direct evaluation of the waves as $\mathcal{W}_1=\mathbf{q}_l^\star-\mathbf{q}_l$, $\mathcal{W}_2=\mathbf{q}_r^\star-\mathbf{q}_l^\star$, $\mathcal{W}_3=\mathbf{q}_r-\mathbf{q}_r^\star$ and by setting $\lambda_1=s_l$, $\lambda_2=s^\star$, $\lambda_3=s_r$ the fluctuations in the x_1-direction are defined as $\mathcal{A}^-\Delta=\sum_{\lambda_\nu<0}\lambda_\nu\mathcal{W}_\nu$, $\mathcal{A}^+\Delta=\sum_{\lambda_\nu\ge 0}\lambda_\nu\mathcal{W}_\nu$ for $\nu=\{1,2,3\}$.

Note that the robustness and positivity-preservation of the HLLC approach are essential for obtaining reliable simulation results when multiple fluids with disparate material properties are involved as it is the case in the configuration described below.

3.4.6. *Deformation simulation from underwater explosion*

As a practical FSI test for realistic explosion-generated shock waves in water (cf. [38]), we simulate a fluid-structure experiment by Ashani & Ghamsari [6]. A small charge ($m_{C4}=20\,\mathrm{g}$ and $m_{C4}=30\,\mathrm{g}$) of the explosive C4 (energy of $1.34\times$ TNT) is detonated in a water-filled basin at the standoff distances $d=25\,\mathrm{cm}$ or $d=30\,\mathrm{cm}$ above a circular air-backed aluminum plate (exposed radius 85 mm) of thickness $h=3\,\mathrm{mm}$. We model the basin with the fluid domain $[-1\,\mathrm{m},1\,\mathrm{m}]\times[-0.8\,\mathrm{m},0.8\,\mathrm{m}]\times[-1\,\mathrm{m},1\,\mathrm{m}]$. Outflow is assumed at all domain boundaries. In analogy to the experiment, air-backed conditions are represented by inserting a rigid cylinder of radius 150 mm from $x_2=-0.8\,\mathrm{m}$ to $x_2=0$ into the domain. The cylinder is sealed by the test plate of radius 150 mm, discretized with 8148 triangles, which is treated as rigid for $r>85\,\mathrm{mm}$. The material parameters for viscoplastic material behavior of aluminum, that were used in these simulations, are $\rho_s=2719\,\mathrm{kg/m}^3$, Young's modulus of $E=69\,\mathrm{GPa}$, Poisson's ratio $\nu=0.33$, and yield stress $\sigma_y=217.6\,\mathrm{MPa}$. The used solid mechanics solver is the thin shell-element research code SFC by Cirak [27]. The cylinder is filled with air ($\gamma^A=1.4$, $p_\infty^A=0$) at

[19]HLLC:Harten-Lax-van Leer Riemann solver with restored Contact surface


(a)

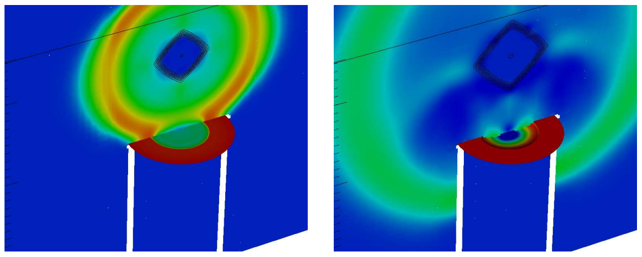

(b) (c)

FIGURE 32. (a) Isolines of p on domains of refinement levels (indicated by color) at $t = 0.31\,\text{ms}$. (b), (c) The plane shows a color plot of p and isolines of α^A, the plate displays the normal vertex velocity at $t = 0.14\,\text{ms}$ and $0.31\,\text{ms}$.

density $\rho^A = 1.29\,\text{kg/m}^3$, the basin with water ($\gamma^W = 7.415$, $p_\infty^W = 296.2\,\text{MPa}$) at $\rho^W = 1027\,\text{kg/m}^3$, which are both initially at rest ($u_n = 0$) and assumed to be at atmospheric pressure $p_0 = 100\,\text{kPA}$. The shock from the explosion is modeled as a spherical energy deposition ($m_{C4} \cdot 6.06\,\text{MJ/kg}$) uniformly distributed over a sphere of radius 5 mm of air at temperature 1500°C located at $(0, d, 0)$.

The fluid domain is discretized with an SAMR base mesh of $50 \times 40 \times 50$ cells. Four additional levels with refinement factors $r_{1,2,3} = 2, r_4 = 4$ are employed. The highest level refinement is static and restricted to the explosion center. Fluid mesh adaptation on all other levels is dynamic and based on φ and the scaled gradient of p. However, refinement at levels 2 and 3 is restricted to the immediate vicinity of the structure and the shock as it impinges onto it. Figure 32(a) depicts a snapshot of the fluid mesh in a plane through the center of the domain for the case $m_{C4} = 20\,\text{g}$, $d = 25\,\text{cm}$. The FSI simulation uses $l_{\text{fsi}} = 3$ with $K = 2$ solid solver sub-steps, and 1296 coupled time steps were computed to reach the final time $t_{\text{end}} = 1\,\text{ms}$.

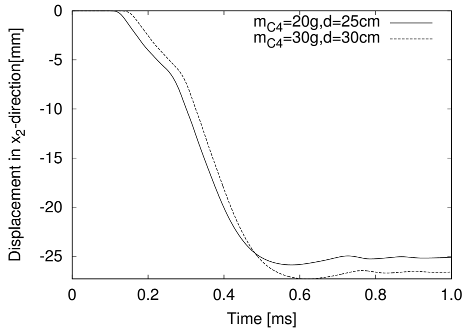


FIGURE 33. Center displacement versus time.

The impact of the spherical shock onto the plate and its partial reflection are visualized in graphics (b) and (c) of Fig. 32, respectively. The induced motion of the exposed part of the test specimen is clearly visible. Figure 33 displays the plate center motion versus time for both cases considered. Note that during the first $\sim 0.2\,\text{ms}$ after the shock impact the deformation occurs with constant velocity since the water near the plates cavitates and does not transmit significant forces onto the plate. In here, we incorporate the effects of cavitation with a simple pressure cutoff model that is implemented by applying the non-conservative energy correction

$$E := \frac{p_c + \gamma p_\infty}{\rho(\gamma - 1)} + \frac{1}{2}\mathbf{u}^T\mathbf{u}\,, \quad \text{for } p < p_c \tag{119}$$

after every fluid time step. Its purpose is to limit all hydrodynamic pressures to a cutoff value p_c, which in here is set to $p_c = -1\,\text{MPa}$.

The computed maximal deflection for the case $m = 20\,\text{g}$, $d = 25\,\text{cm}$ is 25.88 mm; for the case $m_{C4} = 30\,\text{g}$, $d = 30\,\text{cm}$ it is 27.31 mm. Those values compare reasonably well to the experimental measurements of 28.83 mm and 30.09 mm provided in [6], where the differences are primarily due to our rather simplistic modeling of the initial shock wave created by the explosion. Both computations were run on 12 nodes of a parallel cluster with Intel-3.4 GHz-Xeon dual-processors (10 nodes fluid, 2 nodes solid dynamics solver) and required ~ 130 h CPU each (~ 5.4 h wall time).

Outlook

While the present text has been restricted to the purely hyperbolic case for clarity, SAMR techniques are also applicable to elliptic and parabolic problems. As the SAMR approach already uses hierarchical data, it is a small step to geometric multigrid methods [25] that prolong a correction computed on coarser meshes onto finer grids in order to achieve a significant convergence acceleration [22, 53, 113]. An effective multigrid method in the SAMR context then replaces the explicit single-grid FV update with single-grid routines that implement an implicit smoother and defect evaluation. Martin details [80] that for implicit adaptive FV methods exactly the flux correction operation is vital to enforce the required differentiability of the solution along coarse-fine boundaries. Specific to parabolic problems is the additional question of how to utilize hierarchical time step refinement, when a globally coupled, elliptic sub-problem has to be solved in every time step, cf. [4, 10, 81].

A recent area of interest is the construction of SAMR methods that preserve very higher orders of accuracy [7, 90, 122]. The challenges of higher-order SAMR methods, especially for hyperbolic problems, are the construction of higher-order spatial interpolation operations that are monotonicity-preserving but conservative and the realization of higher-order interpolation in time.

Of particular current interest is also the application of SAMR techniques on very large distributed systems [50, 120] and the realization of hybrid distributed/shared memory parallelization, either by combining MPI and OpenMP parallelism [63] or by utilizing accelerators (e.g., graphics processing units) on nodes that communicate via MPI [102]. In contrast to cell-based refinement approaches, the subgrids inherent to SAMR allow for rather coarse-grained shared memory parallelism, making the SAMR approach the prime candidate of adaptive methods for utilizing the new generation of super-computers effectively.

This work was sponsored by the Mathematical, Information, and Computational Sciences Division; Office of Advanced Scientific Computing Research; U.S. Department of Energy (DOE) and was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. This work was also supported by the ASC program of the Department of Energy under subcontract No. B341492 of DOE contract W-7405-ENG-48 and the German DFG high priority research program "Analysis and Numerics of Conservation Laws", grant Ba 840/3-3.

References

[1] R. Abgrall. How to prevent pressure oscillations in multicomponent flow calculations: A quasi conservative approach. *J. Comput. Phys.*, 125:150–160, 1996.

[2] R. Abgrall and S. Karni. Computations of compressible multifluids. *J. Comput. Phys.*, 169:594–523, 2001.

[3] M. J. Aftosmis. Solution adaptive Ccartesian grid methods for aerodynamic flows with complex geometries. Technical Report Lecture Series 1997-2, von Karman Institute for Fluid Dynamics, 1997.

[4] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Comput. Phys.*, 142:1–46, 1998.

[5] M. Arienti, P. Hung, E. Morano, and J. E. Shepherd. A level set approach to Eulerian-Lagrangian coupling. *J. Comput. Phys.*, 185:213–251, 2003.

[6] J. Z. Ashani and A. K. Ghamsari. Theoretical and experimental analysis of plastic response of isotropic circular plates subjected to underwater explosion loading. *Mat.-wiss. u. Werkstofftechn.*, 39(2):171–175, 2008.

[7] M. Barad and P. Colella. A fourth-order accurate local refinement method for Poisson's equation. *J. Comput. Phys.*, 209:1–18, 2005.

[8] P. Bastian. *Parallele adaptive Mehrgitterverfahren.* Teubner Skripten zur Numerik. B. G. Teubner, Stuttgart, 1996.

[9] P. Bastian, M. Blatt, C. Engwer, A. Dedner, R. Klöfkorn, S. P. Kuttanikkad, M. Ohlberger, and O. Sander. The distributed and unified numerics environment (DUNE). In *Proc. of the 19th Symposium on Simulation Technique*, Hannover, 2006.

[10] J. Bell. Block-structured adaptive mesh refinement. Lecture 2. https://ccse.lbl.gov/people/jbb/shortcourse/lecture2.pdf, 2004.

[11] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comp.*, 15(1):127–138, 1994.

[12] M. Berger. *Adaptive mesh refinement for hyperbolic differential equations*. PhD thesis, Stanford University. Report No. STAN-CS-82-924, Aug 1982.

[13] M. Berger. Data structures for adaptive grid generation. *SIAM J. Sci. Stat. Comput.*, 7(3):904–916, 1986.

[14] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1988.

[15] M. Berger and R. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35(6):2298–2316, 1998.

[16] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[17] M. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1278–1286, 1991.

[18] M. J. Berger and C. Helzel. Grid aligned h-box methods for conservation laws in complex geometries. In *Proc. 3rd Intl. Symp. Finite Volumes for Complex Applications*, Porquerolles, June 2002.

[19] G. Berti. GrAL-the grid algorithms library. *Future Generation Computer Systems*, 22(1-2):110–122, 2006.

[20] C. Bond, D. J. Hill, D. I. Meiron, and P. E. Dimotakis. Shock focusing in a planar convergent geometry: experiment and simulation. *J. Fluid Mech.*, 641:297–333, 2009.

[21] G. Booch, J. Rumbaugh, and I. Jacobsen. *The unified modeling language user guide*. Addison-Wesley, Reading, Massachusetts, 1999.

[22] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, 2nd edition, 2001.

[23] D. L. Brown, W. D. Henshaw, and D. J. Quinlan. Overture: An object oriented framework for solving partial differential equations. In *Proc. ISCOPE 1997, appeared in Scientific Computing in Object-Oriented Parallel Environments*, number 1343 in Springer Lecture Notes in Computer Science, 1997.

[24] S. Browne, Z. Liang, R. Deiterding, and J. E. Shepherd. Detonation front structure and the competition for radicals. *Proc. of the Combustion Institute*, 31(2):2445–2453, 2007.

[25] J. Canu and H. Ritzdorf. Adaptive, block-structured multigrid on local memory machines. In W. Hackbuch and G. Wittum, editors, *Adaptive Methods-Algorithms, Theory and Applications*, pages 84–98, Braunschweig/Wiesbaden, January 22-24 1994. Proceedings of the Ninth GAMM-Seminar, Vieweg & Sohn.

[26] F. Cirak, R. Deiterding, and S. P. Mauch. Large-scale fluid-structure interaction simulation of viscoplastic and fracturing thin shells subjected to shocks and detonations. *Computers & Structures*, 85(11-14):1049–1065, 2007.

[27] F. Cirak and M. Ortiz. Fully c^1-conforming subdivision elements for finite deformation thin-shell analysis. *Int. J. Numer. Meth. Engineering*, 51:813–833, 2001.

[28] P. Colella and P. Woodward. The piecewise parabolic method (PPM) for gas-dynamical simulations. *J. Comput. Phys.*, 54:174–201, 1984.

[29] W. Crutchfield and M. L. Welcome. Object-oriented implementation of adaptive mesh refinement algorithms. *J. Scientific Programming*, 2:145–156, 1993.

[30] S. F. Davis. Simplified second-order Godunov-type methods. *SIAM J. Sci. Stat. Comp.*, 9:445–473, 1988.

[31] R. Deiterding. AMROC - Blockstructured Adaptive Mesh Refinement in Object-oriented C++. Available at http://amroc.sourceforge.net.

[32] R. Deiterding. *Parallel adaptive simulation of multi-dimensional detonation structures*. PhD thesis, Brandenburgische Technische Universität Cottbus, Sep 2003.

[33] R. Deiterding. Construction and application of an AMR algorithm for distributed memory computers. In T. Plewa, T. Linde, and V. G. Weirs, editors, *Adaptive Mesh Refinement - Theory and Applications*, volume 41 of *Lecture Notes in Computational Science and Engineering*, pages 361–372. Springer, 2005.

[34] R. Deiterding. Detonation structure simulation with AMROC. In L. T. Yang, editor, *High Performance Computing and Communications 2005*, volume 3726 of *Lecture Notes in Computer Science*, pages 916–927. Springer, 2005.

[35] R. Deiterding. A high-resolution method for realistic detonation structure simulation. In W. Takahashi and T. Tanaka, editors, *Proc. Tenth Int. Conf. Hyperbolic Problems*, volume 1, pages 343–350. Yokohama Publishers, 2006.

[36] R. Deiterding. A parallel adaptive method for simulating shock-induced combustion with detailed chemical kinetics in complex domains. *Computers & Structures*, 87:769–783, 2009.

[37] R. Deiterding and G. Bader. High-resolution simulation of detonations with detailed chemistry. In G. Warnecke, editor, *Analysis and Numerics for Conservation Laws*, pages 69–91. Springer, 2005.

[38] R. Deiterding, F. Cirak, and S. P. Mauch. Efficient fluid-structure interaction simulation of viscoplastic and fracturing thin-shells subjected to underwater shock loading. In S. Hartmann, A. Meister, M. Schäfer, and S. Turek, editors, *Int. Workshop on Fluid-Structure Interaction. Theory, Numerics and Applications, Herrsching am Ammersee 2008*, pages 65–80. kassel university press GmbH, 2009.

[39] R. Deiterding, F. Cirak, S. P. Mauch, and D. I. Meiron. A virtual test facility for simulating detonation-induced fracture of thin flexible shells. In V. N. Alexandrov, G. D. van Albada, P. M. Sloot, and J. Dongarra, editors, *Proc. 6th Int. Conf.*

Computational Science, Reading, UK, May 28-31, 2006, volume 3992 of *Lecture Notes in Computer Science*, pages 122–130. Springer, 2006.

[40] R. Deiterding, F. Cirak, S. P. Mauch, and D. I. Meiron. A virtual test facility for simulating detonation- and shock-induced deformation and fracture of thin flexible shells. *Int. J. Multiscale Computational Engineering*, 5(1):47–63, 2007.

[41] R. Deiterding, R. Radovitzki, S. Mauch, F. Cirak, D. J. Hill, C. Pantano, J. C. Cummings, and D. I. Meiron. Virtual Test Facility: A virtual shock physics facility for simulating the dynamic response of materials. Available at http://www.cacr.caltech.edu/asc.

[42] R. Deiterding, R. Radovitzky, S. P. Mauch, L. Noels, J. C. Cummings, and D. I. Meiron. A virtual test facility for the efficient simulation of solid materials under high energy shock-wave loading. *Engineering with Computers*, 22(3-4):325–347, 2006.

[43] B. Einfeldt, C. D. Munz, P. L. Roe, and B. Sjögreen. On Godunov-type methods near low densities. *J. Comput. Phys.*, 92:273–295, 1991.

[44] R. P. Fedkiw. Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys.*, 175:200–224, 2002.

[45] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492, 1999.

[46] H. Friedel, R. Grauer, and C. Marliani. Adaptive mesh refinement for singular current sheets in incompressible magnetohydrodynamics flows. *J. Comput. Phys.*, 134(1):190–198, 1997.

[47] J. Giordano, G. Jourdan, Y. Burtschell, M. Medale, D. E. Zeitoun, and L. Houas. Shock wave impacts on deforming panel, an application of fluid-structure interaction. *Shock Waves*, 14(1-2):103–110, 2005.

[48] M. Gittings, R. Weaver, M. Clover, T. Betlach, N. Byrne, R. Coker, E. Dendy, R. Hueckstaedt, K. New, R. Oakes, D. Rantal, and R. Stefan. The RAGE radiation-hydrodynamics code. *Comput. Sci. Disc.*, 1, 2008. doi:10.1088/1749-4699/1/1/015005.

[49] E. Godlewski and P.-A. Raviart. *Numerical approximation of hyperbolic systems of conservation laws.* Springer Verlag, New York, 1996.

[50] J. A. Greenough, B. R. de Supinski, R. K. Yates, C. A. Rendleman, D. Skinner, V. Beckner, M. Lijewski, J. Bell, and J. C. Sexton. Performance of a block structured, hierarchical adaptive mesh refinement code on the 64k node IBM BlueGene/L computer. Technical Report LBNL-57500, Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, 2005.

[51] B. Grossmann and P. Cinella. Flux-split algorithms for flows with non-equilibrium chemistry and vibrational relaxation. *J. Comput. Phys.*, 88:131–168, 1990.

[52] B. T. Gunney, A. M. Wissink, and D. A. Hysoma. Parallel clustering algorithms for structured AMR. *J. Parallel and Distributed Computing*, 66(11):1419–1430, 2007.

[53] W. Hackbusch. *Multi-Grid Methods and Applications.* Springer Verlag, Berlin, Heidelberg, 1985.

[54] J. Hallquist and J. I. Lin. A nonlinear explicit three-dimensional finite element code for solid and structural mechanics. Technical Report UCRL-MA-107254, Lawrence Livermore National Laboratory, 2005.

[55] A. Harten. High resolution schemes for hyperbolic conservation laws. *J. Comput. Phys.*, 49:357–393, 1983.

[56] A. Harten and J. M. Hyman. Self-adjusting grid methods for one-dimensional hyperbolic conservation laws. *J. Comput. Phys.*, 50:235–269, 1983.

[57] A. Harten, J. M. Hyman, and P. D. Lax. On finite-difference approximations and entropy conditions for shocks. *Comm. Pure Appl. Math.*, 29:297–322, 1976.

[58] W. D. Henshaw and D. W. Schwendeman. An adaptive numerical scheme for high-speed reactive flow on overlapping grids. *J. Comput. Phys.*, 191:420–447, 2003.

[59] C. Hirsch. *Numerical computation of internal and external flows.* John Wiley & Sons, Chichester, 1988.

[60] R. D. Hornung, A. M. Wissink, and S. H. Kohn. Managing complex data and geometry in parallel structured AMR applications. *Engineering with Computers*, 22:181–195, 2006.

[61] P. Hung. *Algorithms for reaction mechanism reduction and numerical simulation of detonations initiated by projectiles.* PhD thesis, California Institute of Technology, 2003.

[62] N. N. Janenko. *Die Zwischenschrittmethode zur Lösung mehrdimensionaler Probleme der mathematischen Physik.* Springer-Verlag, Berlin, 1969.

[63] H. Jourdon. HERA: A hydrodynamic AMR platform for multi-physics simulation. In T. Plewa, T. Linde, and V. G. Weirs, editors, *Adaptive Mesh Refinement - Theory and Applications*, volume 41 of *Lecture Notes in Computational Science and Engineering*, pages 283–294. Springer, 2005.

[64] P. Kaps and P. Rentrop. Generalized Runge-Kutta methods of order four with stepsize control for stiff ordinary differential equations. *Num. Math.*, 33:55–68, 1979.

[65] R. J. Kee, F. M. Rupley, and J. A. Miller. *Chemkin-II: A Fortran chemical kinetics package for the analysis of gas-phase chemical kinetics.* SAND89-8009, Sandia National Laboratories, Livermore, California, Sep 1989.

[66] D. Kröner. *Numerical schemes for conservation laws.* John Wiley & Sons and B. G. Teubner, New York, Leipzig, 1997.

[67] D. Kröner and M. Ohlberger. A posteriori error estimates for upwind finite volume schemes for nonlinear conservation laws in multi dimensions. *Mathematics of Computation*, 69(229):25–39, 1999.

[68] C. B. Laney. *Computational gasdynamics.* Cambridge University Press, Cambridge, 1998.

[69] J. Langseth and R. LeVeque. A wave propagation method for three dimensional conservation laws. *J. Comput. Phys.*, 165:126–166, 2000.

[70] B. Larrouturou. How to preserve the mass fractions positivity when computing compressible multi-component flows. *J. Comput. Phys.*, 95:59–84, 1991.

[71] S. J. Laurence, R. Deiterding, and H. G. Hornung. Proximal bodies in hypersonic flows. *J. Fluid Mech.*, 590:209–237, 2007.

[72] R. J. LeVeque. *Numerical methods for conservation laws.* Birkhäuser, Basel, 1992.

[73] R. J. LeVeque. Simplified multi-dimensional flux limiter methods. In M. J. Baines and K. W. Morton, editors, *Numerical Methods for Fluid Dynamics 4*, pages 175–190, Oxford, 1993. Clarendon Press.

[74] R. J. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numer. Anal.*, 33(2):627–665, 1996.

[75] R. J. LeVeque. Wave propagation algorithms for multidimensional hyperbolic systems. *J. Comput. Phys.*, 131(2):327–353, 1997.

[76] R. J. LeVeque. *Finite volume methods for hyperbolic problems.* Cambridge University Press, Cambridge, New York, 2002.

[77] R. Löhner, J. Baum, C. Charman, and D. Pelessone. Fluid-structure interaction simulations using parallel computers. volume 2565 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2003.

[78] P. MacNeice, K. M. Olson, C. Mobarry, R. deFainchtein, and C. Packer. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126:330–354, 2000.

[79] A. Majda. *Compressible fluid flow and systems of conservation laws in several space variables.* Applied Mathematical Sciences Vol. 53. Springer-Verlag, New York, 1984.

[80] D. F. Martin. *A cell-centered adaptive projection method for the incompressible Euler equations.* PhD thesis, University of California at Berkeley, 1998.

[81] D. F. Martin and P. Colella. An adaptive cell-centered projection method for the incompressible Euler equations. *J. Comput. Phys.*, 163(2):271–312, 2000.

[82] S. P. Mauch. *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations.* PhD thesis, California Institute of Technology, 2003.

[83] R. L. Meakin. An efficient means of adaptive refinement within systems of overset grids. In *12th AIAA Computational Fluid Dynamics Conference, San Diego*, AIAA-95-1722-CP, 1995.

[84] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[85] S. M. Murman, M. J. Aftosmis, and M. J. Berger. Implicit approaches for moving boundaries in a 3-d Cartesian method. In *41st AIAA Aerospace Science Meeting*, AIAA 2003-1119, 2003.

[86] H. J. Neeman. *Autonomous hierarchical adaptive mesh refinement.* PhD thesis, University of Illinois at Urbana-Champaign, 1996.

[87] R. R. Nourgaliev, T. N. Dinh, and T. G. Theofanus. On capturing of interfaces in multimaterial compressible flows using a level-set-based Cartesian grid method. Technical Report 05/03-1, Center for Risk Studies and Safety, UC Santa Barbara, May 2003.

[88] E. S. Oran and J. P. Boris. *Numerical simulation of reactive flow.* Cambridge Univ. Press, Cambridge, 2nd edition, 2001.

[89] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces.* Applied Mathematical Science Volume 153. Springer, New York, 2003.

[90] C. Pantano, R. Deiterding, D. J. Hill, and D. I. Pullin. A low-numerical dissipation patch-based adaptive mesh refinement method for large-eddy simulation of compressible flows. *J. Comput. Phys.*, 221(1):63–87, 2007.

[91] M. Parashar and J. C. Browne. On partitioning dynamic adaptive grid hierarchies. In *Proc. of the 29th Annual Hawaii Int. Conf. on System Sciences*, Jan 1996.

[92] M. Parashar and J. C. Browne. System engineering for high performance computing software: The HDDA/DAGH infrastructure for implementation of parallel structured adaptive mesh refinement. In *Structured Adaptive Mesh Refinement Grid Methods*, IMA Volumes in Mathematics and its Applications. Springer, 1997.

[93] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome. An adaptive Cartesian grid method for unsteady compressible flows in irregular regions. *J. Comput. Phys.*, 120:287–304, 1999.

[94] J. J. Quirk. An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies. *Computers Fluids*, 23:125–142, 1994.

[95] J. J. Quirk. Godunov-type schemes applied to detonation flows. In J. Buckmaster, editor, *Combustion in high-speed flows: Proc. Workshop on Combustion, Oct 12-14, 1992, Hampton*, pages 575–596, Dordrecht, 1994. Kluwer Acad. Publ.

[96] J. Ray, R. C. Armstrong, C. Safta, B. J. Debusschere, B. A. Allan, and H. N. Najm. Computational frameworks for advanced combustion simulations. In T. Echekki and E. Mastorakos, editors, *Turbulent Combustion Modeling: Advances, Trends and Perspective.* Springer-Verlag, 2011.

[97] C. A. Rendleman, V. E. Beckner, M. Lijewski, W. Crutchfield, and J. B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3:147–157, 2000.

[98] P. L. Roe. Approximate Riemann solvers, parameter vectors and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[99] A. M. Roma, C. S. Perskin, and M. J. Berger. An adaptive version of the immersed boundary method. *J. Comput. Phys.*, 153:509–534, 1999.

[100] H. Sagan. *Space-Filling Curves.* Springer-Verlag, New-York, 1994.

[101] R. Sanders, E. Morano, and M.-C. Druguett. Multidimensional dissipation for upwind schemes: Stability and applications to gas dynamics. *J. Comput. Phys.*, 145:511–537, 1998.

[102] H.-Y. Schive, Y.-C. Tsai, and T. Chiueh. GAMER: a GPU-accelerated adaptive mesh refinement code for astrophysics. *Astrophysical J. Supplement Series*, 186:457–484, 2010.

[103] L. I. Sedov. *Similarity and Dimensional Methods in Mechanics.* Academic, New York, 1959.

[104] J. A. Sethian. *Level set methods and fast marching methods.* Cambridge University Press, Cambridge, New York, 1999.

[105] C.-W. Shu. Essentially non-oscillatory and weigted essentially non-oscillatory schemes for hyperbolic conservation laws. Technical Report CR-97-206253, NASA, 97.

[106] K.-M. Shyue. An efficient shock-capturing algorithm for compressible multicomponent problems. *J. Comput. Phys.*, 142:208–242, 1998.

[107] K.-M. Shyue. A volume-fraction based algorithm for hybrid barotropic and non-barotropic two-fluid flow problems. *Shock Waves*, 15:407–423, 2006.

[108] T. Sonar. *Mehrdimensional ENO-Verfahren.* Teubner Verlag, Stuttgart, 1997.

[109] U. Specht. *Numerische Simulation mechanischer Wellen an Fluid-Festkörper-Mediengrenzen.* Number 398 in VDI Reihe 7. VDU Verlag, Düsseldorf, 2000.

[110] G. Strang. On the construction and comparison of difference schemes. *SIAM J. Num. Anal.*, 5:506–517, 1968.

[111] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics.* Springer-Verlag, Berlin, Heidelberg, 2nd edition, 1999.

[112] E. F. Toro, M. Spruce, and W. Speares. Restoration of the contact surface in the HLL-Riemann solver. *Shock Waves*, 4:25–34, 1994.

[113] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid.* Academic Press, San Antonio, 2001.

[114] Y.-H. Tseng and J. H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *J. Comput. Phys.*, 192:593–623, 2003.

[115] G. D. van Albada, B. van Leer, and W. W. Roberts. A comparative study of computational methods in cosmic gas dynamics. *Astron. Astrophysics*, 108:76–84, 1982.

[116] B. van Leer. Towards the ultimate conservative difference scheme V. A second order sequel to Godunov's method. *J. Comput. Phys.*, 32:101–136, 1979.

[117] B. van Leer. Flux-vector splitting for the euler equations. In *Eighth International Conference on Numerical Methods in Fluid Dynamics*, volume 170 of *Lecture Notes in Physics*, pages 507–512, 1982.

[118] B. van Leer. On the relation between the upwind-differencing schemes of Godunov, Enguist-Osher and Roe. *SIAM J. Sci. Stat. Comp.*, 5(1):1–20, 1985.

[119] C. K. Westbrook. Chemical kinetics of hydrocarbon oxidation in gaseous detonations. *Combust. Flame*, 46:191–210, 1982.

[120] A. Wissink, D. Hysom, and R. Hornung. Enhancing scalability of parallel structured amr calculations. In *Proc. 17th Int. Conf. Supercomputing*, pages 336–347, 2003.

[121] N. K. Yamaleev and M. H. Carpenter. On accuracy of adaptive grid methods for captured shocks. *J. Comput. Phys.*, 181:280–316, 2002.

[122] J. L. Ziegler, R. Deiterding, J. E. Shepherd, and D. I. Pullin. An adaptive high-order hybrid scheme for compressive, viscous flows with detailed chemistry. *J. Comput. Phys.*, submitted.