ESAIM: PROCEEDINGS, **December 2009, Vol. 29, p. 28-42**
F. Coquel, Y. Maday, S. Müller, M. Postel and Q. H. Tran, Editors

ADAPTIVE MULTIRESOLUTION OR ADAPTIVE MESH REFINEMENT? A CASE STUDY FOR 2D EULER EQUATIONS

Ralf Deiterding[1], Margarete O. Domingues[2,3], Sônia M. Gomes[4], Olivier Roussel[3,4] and Kai Schneider[53,5]

Résumé. Nous présentons des simulations adaptatives multirésolution (MR) des équations d'Euler compressibles bi-dimensionnelles pour un problème de Riemann classique. Les résultats sont comparés en précision et en efficacité – temps CPU et place mémoire – avec ceux obtenus par la méthode volumes finis sur la grille la plus fine. Pour le même cas-test, nous présentons les calculs obtenus à l'aide de la méthode AMR (Adaptive Mesh Refinement) en imposant les mêmes critères de précision. Les résultats ainsi obtenus sont comparés en termes d'effort de calcul et de compression mémoire, en utilisant des pas de temps globaux puis locaux. De ces résultats préliminaires, nous concluons que les techniques multirésolution présentent des gains en termes de temps CPU et de place mémoire supérieurs à ceux de la méthode AMR.

Abstract. We present adaptive multiresolution (MR) computations of the two-dimensional compressible Euler equations for a classical Riemann problem. The results are then compared with respect to accuracy and computational efficiency, in terms of CPU time and memory requirements, with the corresponding finite volume scheme on a regular grid. For the same test case, we also perform computations using adaptive mesh refinement (AMR) imposing similar accuracy requirements. The results thus obtained are compared in terms of computational overhead and compression of the computational grid, using in addition either local or global time stepping strategies. We preliminarily conclude that the multiresolution techniques yield improved memory compression and gain in CPU time with respect to the adaptive mesh refinement method.

Introduction

Adaptive discretization methods for solving nonlinear PDEs have a long tradition and can be tracked back to the late seventies [11]. Adaptivity is motivated by the huge computational complexity of real world problems which involve typically a multitude of dynamically active spatial and temporal scales. Introducing adaptivity

[1] Computer Science and Mathematics Division, Oak Ridge National Laboratory, P.O. Box 2008 MS-6367, Oak Ridge, TN 37831, United States (e-mail : `deiterdingr@ornl.gov`)

[2] Laboratório Associado de Computação e Matemática Aplicada (LAC), Coordenadoria dos Laboratórios Associados (CTE), Instituto Nacional de Pesquisas Espaciais (INPE), Av. dos Astronautas 1758, 12227-010 São José dos Campos, São Paulo, Brazil (e-mail : `margarete@lac.inpe.br`)

[3] Laboratoire de Modélisation en Mécanique et Procédés Propres (M2P2), CNRS, Universités d'Aix-Marseille et Ecole Centrale Marseille, 38 rue F. Joliot-Curie, 13451 Marseille Cedex 20, France (e-mail : `o_roussel@yahoo.fr`)

[4] Universidade Estadual de Campinas (UNICAMP), IMECC, Caixa Postal 6065, 13083-970 Campinas, São Paulo, Brazil (e-mail : `soniag@ime.unicamp.br`)

[5] Centre de Mathématiques et d'Informatique (CMI), Université de Provence, 39 rue F. Joliot-Curie, 13453 Marseille Cedex 13, France (e-mail : `kschneid@cmi.univ-mrs.fr`)

can be understood in the sense that the computational effort is concentrated at locations and time instants where it is necessary to insure a given numerical accuracy, while efforts may be significantly reduced elsewhere.

Adaptive methods are in many cases more competitive than schemes on regular fine grids, in particular for solutions of nonlinear PDEs exhibiting a non-uniformly distributed regularity. Essential ingredients of fully adaptive schemes are first reliable error estimators for the solution, e.g., Richardson ideas of extrapolation, adjoint problems, gradient based approaches. For evolutionary problems, a major task is the time evolution of the grid and a reliable prediction of the grid for the next time step.

Adaptivity has however its price. A significant effort has to be made on programming data structures, which are usually based on graded trees, hash-tables or multi-domains. Moreover, the computational cost per cell is significantly increased. Hence, an adaptive method is only efficient when the data compression is large enough to compensate the additional computational cost per cell. Fortunately, for problems exhibiting local discontinuities or steep gradients, adaptive computations are always faster than fine grid computations.

Adaptive finite element methods have a long history, in particular for elliptic problems. Moving grid techniques have been applied successfully to combustion problems [34]. A-posteriori error estimators have also been studied for a long time to improve the grid, since the early work of Babuška and Rheinboldt [2]. However, adjoint problems have to be solved which are more expensive than the original PDEs [3].

The main challenge is to estimate theoretically and control the error of adaptive schemes with respect to the exact solution, or at least with respect to the same numerical scheme on an underlying uniform grid. Self adaptive methods are preferred as they automatically adjust to the solution.

Recently, multiresolution (MR) techniques have become popular for hyperbolic conservation laws going back to the seminal work of Harten [31] in the context of finite volume schemes and cell-average MR analysis. Starting point is a finite volume scheme for hyperbolic conservation laws on a regular grid. Subsequently a discrete multiresolution analysis is used to avoid expensive flux computations in smooth regions, first without reducing memory requirements, e.g for 1D hyperbolic conservation laws (Harten [31]), 1D conservation laws with viscosity (Bihari [9]), 2D hyperbolic conservations laws (Bihari and Harten [10]), 2D compressible Euler equations (Chiavassa and Donat [13]), 2D hyperbolic conservation laws with curvilinear patches (Dahmen *et al* [18]) and unstructured meshes (Abgrall and Harten [1], Cohen *et al* [15]). A fully adaptive version, still in the context of 1D and 2D hyperbolic conservation laws, has been developed to reduce also memory requirements (Gottschlich-Müller and Müller [30], Kaibara and Gomes [35], Cohen *et al* [16]). This algorithm has been extended to the 3D case and to parabolic PDEs (Roussel *et al* [48], Roussel and Schneider [46]), and more recently to self–adaptive global and local time–steppings (Müller and Stiriba [41], Domingues *et al* [26–28]). Therewith the solution is represented and computed on a dynamically evolving automatically adapted grid. Different strategies have been proposed to evaluate the flux without requiring a full knowledge of fine grid cell-average values.

The MR approach has also been used in other contexts. For instance, the Sparse Point Representation (SPR) method was the first fully adaptive MR scheme, introduced by Hölmstrom [32,33] in the context of finite differences and point-value MR analysis, leading to both CPU time and memory reduction. In the SPR method, the wavelet coefficients are used as regularity indicators to create locally refined grids, on which the numerical solution is represented and the finite difference discretization of the operators is performed. Applications of the SPR method have been published in [25,44]. Concerning Discontinuous Galerkin methods, they have been applied to hyperbolic conservation laws in [12] using Haar wavelet indicators to decide where to refine or coarsen the meshes.

These publications reveal that the multiresolution concept has been applied by several groups with success to different stiff problems. For comprehensive literature about the subject, we refer to the books of Cohen and Müller [14,40].

The blockstructured adaptive mesh refinement technique (AMR or SAMR) for hyperbolic partial differential equations has been pioneered by Berger and Oliger [5,8]. While the first approach utilized rotated refinement grids that required complicated conservative interpolation operations, AMR denotes today especially the simplified variant of Berger and Collela [6] that allows only refinement patches aligned to the coarse grid mesh.

The striking efficiency of this algorithm, in particular for instationary supersonic gas dynamical problems, has been demonstrated by Bell *et al* [4]. Several implementations of the AMR method for single processor computers [7,17,29] and parallel systems [4,19,36,45] have been presented; variants utilizing simplified data structures have also been proposed [39,43]. Our interest in here is in comparing MR with the fundamental AMR method proposed in [6]. Such benchmarks are still missing and the current paper can be seen as a first contribution step towards a direct comparison.

The outline of the paper is the following: first, we present the governing Euler equations together with their finite volume discretization. Then, we briefly summarize the adaptive mesh refinement and multiresolution strategies. Numerical results for a classical Riemann test problem are presented in Section 2, followed by preliminary conclusions.

1. Numerical Methods

1.1. Finite Volume discretization of the Euler equations

The compressible Euler equations can be written in the following form,

$$\frac{\partial Q}{\partial t} + \nabla \cdot f(Q) = 0, \tag{1}$$

with $Q = (\rho, \rho\vec{v}, \rho e)^T$, where $\rho = \rho(\vec{x}, t)$ is the density, $\vec{v} = \vec{v}(\vec{x}, t)$ is the vector velocity with components (v_1, v_2), and $e = e(\vec{x}, t)$ is the energy per unit of mass, which are functions of time t and position $\vec{x} = (x_1, x_2)$. The flux function $f = (f_1, f_2)^T$ is given by

$$f_1 = \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ (\rho e + p) v_1 \end{pmatrix} \quad f_2 = \begin{pmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ (\rho e + p) v_2 \end{pmatrix},$$

where $p = p(\vec{x}, t)$ denotes the pressure. The system is completed by an equation of state for a calorically ideal gas

$$p = \rho R T = (\gamma - 1)\, \rho \left(e - \frac{|\vec{v}|^2}{2} \right), \tag{2}$$

where $T = T(\vec{x}, t)$ is the temperature, γ the specific heat ratio and R the universal gas constant. In dimensionless form, we obtain the same system of equations, but the equation of state becomes $p = \frac{\rho T}{\gamma M^2}$, where M denotes the Mach number. For the present applications, the physical parameters are $M = 1$ and $\gamma = 1.4$. The considered domain is the unit square $[0,1] \times [0,1]$, with free-slip boundary conditions.

As reference discretization, we consider the numerical solution represented by the vector $\bar{Q}$ of the approximated cell-averages $Q_{i,j}$

$$\bar{Q}_{i,j} = \frac{1}{|\Omega_{i,j}|} \int_{\Omega_{i,j}} Q(x_1, x_2)\, dx_1\, dx_2$$

on cells $\Omega_{i,j}$ of a uniform grid Ω. For the space discretization, a finite volume (FV) method is chosen, which results in the following system of ODEs

$$\frac{d\bar{Q}}{dt} = -F(\bar{Q}), \tag{3}$$

where $F(\bar{Q})$ denotes the vector of the numerical flux function.

For the time integration, approximate solutions $\bar{Q}^n$ at a sequence of time instants t^n are obtained using explicit ODE solvers. The time step to go from t^n to $t^{n+1} = t_n + \Delta t^n$ is set by the CFL condition.

1.2. Adaptive Mesh Refinement Method

The AMR method [4, 6, 8] follows a patch-oriented refinement approach. Instead of replacing single cells by finer ones, non-overlapping rectangular subgrids $G_{l,m}$ define the domain of an entire level $l = 0, \ldots, L$ by $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$. As the construction of refinement proceeds recursively, a hierarchy of subgrids successively contained within the next coarser level domain is created (Figure 1). Note that the recursive nature of the algorithm allows only the addition of one new level in each refinement operation. In contrast to cell-based mesh adaptation techniques, the patch-based approach does not require special coarsening operations; subgrids are simply removed from the hierarchy. The coarsest possible resolution is thereby restricted to the level 0 grid. Usually, it is assumed that all mesh widths on level l are r_l-times finer than on the level $l-1$, i.e. $\Delta t_l = \Delta t_{l-1}/r_l$ and $\Delta x_{n,l} = \Delta x_{n,l-1}/r_l$, with $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$, which ensures that a time-explicit finite volume scheme (in principle) remains stable under a CFL-type condition on all levels of the hierarchy.

The numerical update is applied on the level l by calling a single-grid routine implementing the finite volume scheme in a loop over all the subgrids $G_{l,m}$. The regularity of the input data allows a straightforward implementation of the scheme and further permits optimizations to take advantage of high-level caches, pipelining, etc. New refinement grids are initialized by interpolating the vector of conservative quantities Q from the next coarser level. However, data in cells already refined is copied directly from the previous refinement patches. *Ghost* or *halo* cells around each patch are used to decouple the subgrids computationally. Ghost cells outside of the root domain G_0 are used to implement physical boundary conditions. Ghost cells in G_l have a unique interior cell analogue and are set by copying the data value from the patch where the interior cell is contained (synchronization). For $l > 0$, internal boundaries can also be used. If recursive time step refinement is employed, ghost cells at the internal refinement boundaries on the level l are set by time-space interpolation from the two previously calculated time steps of level $l-1$. Otherwise, spatial interpolation from the level $l-1$ is sufficient.

```
AdvanceLevel($l$)
 Repeat $r_l$ times
    Set ghost cells of $Q^l(t)$
    If time to regrid
       Regrid($l$)
    UpdateLevel($l$)
    If level $l+1$ exists
       Set ghost cells of $Q^l(t+\Delta t_l)$
       AdvanceLevel($l+1$)
       Average $Q^{l+1}(t+\Delta t_l)$ onto
          $Q^l(t+\Delta t_l)$
       Flux correction of $Q^l(t+\Delta t_l)$
    $t := t + \Delta t_l$
```

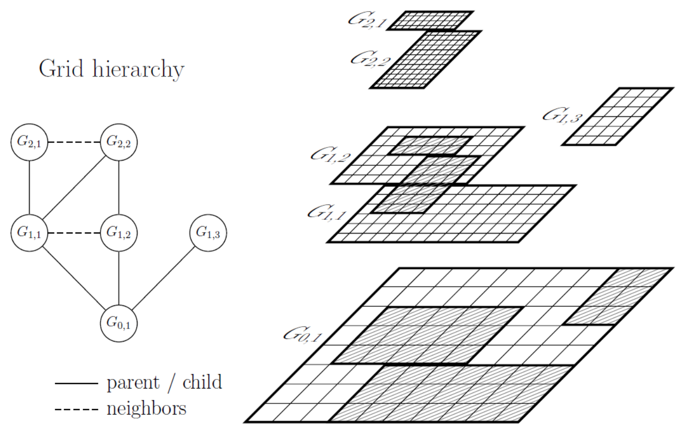

FIGURE 1. Recursive AMR algorithm and typical hierarchy of rectangular subgrids.

The characteristic of the AMR algorithm is that refinement patches overlay coarser grid data structures, instead of being embedded, again avoiding data fragmentation. Values of cells covered by finer subgrids are subsequently overwritten by averaged fine grid values, which, in general, would lead to a loss of conservation on the coarser mesh. A remedy to this problem is to replace the coarse grid numerical fluxes at refinement boundaries with the sum of fine grid fluxes along the corresponding coarse cell boundary. Details about this flux correction can be found in [6, 20, 21]. The basic recursive AMR algorithm is formulated in Figure 1. New refinement grids on all the higher levels are created by calling `Regrid()` at a given level l. The level l by itself is not modified. To consider the nesting of the level domains already in the grid generation, `Regrid()` starts at the highest level to be refined and proceeds down to $l+1$. After evaluating the refinement indicators and flagging cells for refinement, a special clustering algorithm [4] is used to create new refinement patches until the ratio between flagged and all cells in every new subgrid is above a prescribed threshold $0 < \eta_{tol} \leq 1$.

In the present paper, all the AMR computations have been carried out using the AMROC (Adaptive Mesh Refinement in Object-oriented C++) system that implements the Berger-Collela AMR algorithm [6] generically in C++ and is free of charge for scientific use [19,23]. At the present time, the AMR core of AMROC consists of approximately $46,000$ lines of code in C++ and approximately $6,000$ lines for visualization and data conversion. Although AMROC permits large-scale MPI-parallel AMR computations in all three space dimensions [22,42], the present investigation is restricted to 2D and uses only the *serial* algorithm of the software. The employed patch-based finite volume discretization is a standard unsplit shock-capturing MUSCL scheme with a Minmod limiter and an AUSMDV flux-vector splitting numerical flux function [52]. The time integration is based on the MUSCL-Hancock approach [20,50]. For sufficiently smooth solutions, the method is of second-order accuracy in space and time. Similarly to the AMR inter-level transfer operations (interpolation, averaging), the employed finite volume update routine was coded in Fortran-77 and all the codes were compiled with standard compiler optimizations (-O3 with loop unrolling, inlining, etc.) using the GNU compiler suite on the benchmark system.

As refinement indicators, scaled gradient criteria of the form

$$|w(Q_{j+1,k}) - w(Q_{jk})| > \epsilon_w \ , \ |w(Q_{j,k+1}) - w(Q_{jk})| > \epsilon_w \ , \ |w(Q_{j+1,k+1}) - w(Q_{jk})| > \epsilon_w \tag{4}$$

were applied to density and pressure, where $\epsilon_\rho = \epsilon_p = 0.05$. As it is common practice [6], a layer of one additional cell width was also tagged for the refinement around each refinement flag to ensure that the flagged feature does not leave the refinement region during the next time step. Furthermore, AMROC allows for the additional application of a heuristic local error indicator based on a Richardson estimation [6,8] that compares an auxiliary twice coarser solution $\bar{\mathcal{Q}}$ with a coarsened solution $\mathcal{Q}$ of $\mathbf{Q}$ at $t + \Delta t$. The difference

$$\frac{|w(\bar{\mathcal{Q}}_{jk}(t+\Delta t)) - w(\mathcal{Q}_{jk}(t+\Delta t))|}{2^{o+1}-2} > \eta_w$$

is an approximation to the leading-order term of the local error on quantity w. Multiple refinement indicators are combined by successive boolean OR operations. For the shock dominated example of Section 2, however, scaled gradient and Richardson error estimation criteria were found to give virtually identical grid refinement and the benchmarked computations only utilized the former.

1.3. Adaptive Multiresolution Method

The adaptive MR scheme belongs to a class of adaptive hybrid methods which are formed by two basic parts: the operational part and the representation part. The operational part consists of an accurate and stable discretization of the partial differential operators. In the representation part, wavelet tools are employed for the multiresolution analysis of the discrete information. The principle of the MR setting is to represent a set of function data as values on a coarser grid plus a series of differences at different levels of nested grids. The information at consecutive scale levels are related by inter-level transformations: projection and prediction operators. The wavelet coefficients are defined as prediction errors, and they retain the detail information when going from a coarse to a finer grid [48].

In MR schemes for the adaptive numerical solution of PDEs, the main idea is to use the decay of the wavelet coefficients to obtain information on local regularity of the solution. Adaptive MR representations are obtained by stopping the refinement in a cell at a certain scale level where the wavelet coefficients are non-significant. In particular, these coefficients are small in regions where the solution is smooth and significant close to irregularities. In the finite volume context, the natural representation framework is the multiresolution analysis based on cell-averages. Instead of using the cell-average representation on the uniform fine grid, the MR scheme computes the numerical solution represented by its cell-averages on an adaptive sparse grid, which is formed by the cells whose wavelet coefficients are significant.

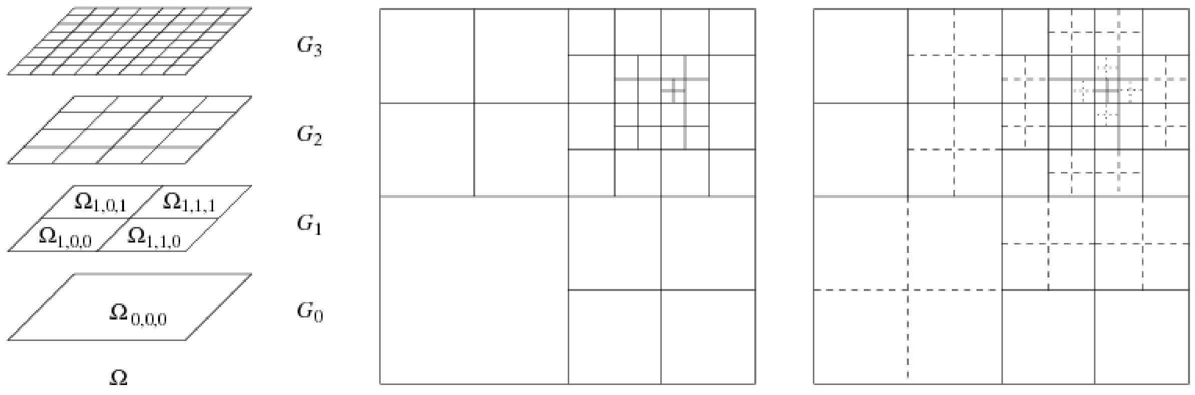

Figure 2. Left: set of nested dyadic grids G_l for $0 \leq l \leq 3$. Center: Sketch of a 2D tree structure. Right: corresponding sketch with its leaves (*plain*) and virtual leaves (*dashed*).

An efficient way to store the reduced MR data is to use a tree data structure, where grid adaptivity is related with an incomplete tree, and where the refinement may be interrupted at intermediate scale levels. This means that, using the tree terminology, a MR grid is formed by *leaves*, which are nodes without children (Figure 2).

For the time evolution of the solution, three basic steps are considered: *refinement*, *evolution* and *coarsening*. The refinement operator is a precautionary measure to account for possible translation or creation of finer scales in the solution between two subsequent time steps. Since the regions of smoothness or irregularities of the solution may change with time, the MR grid at t^n may not be convenient anymore at the next time step t^{n+1}. Therefore, before doing the time evolution, the representation of the solution should be interpolated onto an extended grid that is expected to be a refinement of the adaptive grid at t^n, and to contain the adaptive grid at t^{n+1}.

Then, the time evolution operator is applied on the leaves of the extended grid. To compute fluxes between leaves of different levels, we also add *virtual leaves* (Figure 2, right side). Conservativity is ensured by the fact that the fluxes are always computed on the higher level, the value being reported on the leaves of a lower level [48].

Finally, a wavelet thresholding operation (coarsening) is applied in order to unrefine the cells in the extended grid that are unnecessary for an accurate representation of the solution at t^{n+1}.

In order to save CPU time, instead of evolving the solution with a single time step on all grid cells, the solution may be integrated with a different time step for each level. For the MR/LT scheme, the time step Δt^n at the finest scale level L is determined by the CFL condition. The principle is then to evolve the cells at lower levels $0 \leq \ell < L$ with larger time steps $\Delta t^n_\ell = 2^{L-\ell}\Delta t^n$. Required missing values in ghost cells are interpolated at intermediate time levels.

For the MR method used in the present paper, the reference FV scheme of the operational part uses a 2nd order MUSCL with an AUSM+ flux vector splitting scheme [38] and the van Albada limiter [51]. For the time evolution, an explicit second-order Runge-Kutta (RK2) scheme is used, and the cell-average multiresolution analysis corresponds to a prediction operator based on a third order polynomial interpolation on the cell-aerages. For further details on the adaptive MR scheme, we refer to [48], and to [26] for its MR/LT version.

2. Numerical Results

In this section, the main parameters used in both methods are described. Then, the results of the simulations are presented and discussed.

2.1. **Initial condition and parameters**

The case study chosen here is a typical Riemann problem for 2D gas dynamics treated e.g. in [37], and initially discussed in [49,53]. The initial data are constant in each quadrant (Figure 3), and the values are given in Table 1. This test case corresponds to the configuration #5 in [37]. This classical test case only involves contact discontinuities and generates motion in opposite directions.

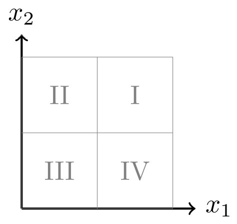

FIGURE 3. Sketch of the initial condition

TABLE 1. Initial Values for the Lax-Liu configuration #5 [37]

Variables	Domain position			
	I	II	III	IV
Density(ρ)	1.00	2.00	1.00	3.00
Presure (p)	1.00	1.00	1.00	1.00
Velocity component (v_1)	-0.75	-0.75	0.75	0.75
Velocity component (v_2)	-0.50	0.50	0.50	-0.50

As detailed previously, both mesh refinement methods use enhanced AUSM-type numerical flux functions with comparable second-order accurate reconstruction and limiting. For both methods, the expected CFL number is 0.45, and a series of computations with maximal level $L = 8, 9$ and 10 are performed, respectively corresponding to 256×256, 512×512, and 1024×1024 cells on the finest uniform grid. Adaptive computations are performed first without, then with a local time stepping for both AMR and MR schemes.

For the MR method, the refinement factors are always dyadic, i.e., $r_l = 2$, and leaves are allowed in every level $0 \leq l \leq L$. A threshold analysis has been performed, and the optimal values of the tolerance ϵ are 0.01 for $L = 8$, 0.008 for $L = 9$, and 0.005 for $L = 10$. For details on the way to determine the optimal value of ϵ, we refer to [48].

The basic coarse grid for the AMR method is a 128×128 grid, i.e. $7 \leq l \leq L$, and the threshold coefficients for refinement $\epsilon_\rho = \epsilon_p = 0.05$ were used. Only computations with refinement factors $r_l = 2$ are compared to the MR method. Additionally, an option was added to AMROC to always restrict the repeat-loop in the recursive AMR algorithm of Figure 1 to one iteration, deactivating the utilization of hierarchical (or local) time step refinement, thereby enabling comparisons with the MR method without local time stepping. Since these measures influence the computational efficiency of the AMR method, we quantify the resulting performance penalties for the $L = 10$ case (1931 time steps in unigrid mode) in Table 2.

Table 2 provides a breakdown of AMROC's overall CPU time into the most important operations of the AMR algorithm with and without local time steps (LT), and when a refinement factor of 4 is used instead of 2 on the highest level. *Integration* denotes the block-based routine of the finite volume scheme; the other profiled operations are sub-tasks of the refinement algorithm (cf. Section 1.2). Further on, counters were added to the code to compute the number of cells updated, $\sum \mathcal{C}$, and the number of updated leaf cells, $\sum \mathcal{L}$, which are not covered by further refinement.

Table 2. Breakdown of the computational costs into the most important operations and integration performance with and without local time stepping to solve the benchmark problem with AMROC.

	uni	no LT		with LT	
r_l	1	2,4	2,2,2	2,4	2,2,2
Integration [%]	98.4	70.3	48.5	86.2	59.0
Interpolation [%]	-	4.2	3.8	4.2	4.6
Flux correction [%]	-	2.7	5.1	1.1	3.6
Flagging [%]	-	4.1	7.0	1.1	4.4
Regridding [%]	-	11.1	17.2	4.1	14.9
Clustering [%]	-	6.0	16.9	1.9	12.3
Misc [%]	1.1	1.6	1.5	1.4	1.2
CPU time [s]	4531	1249	1721	865	1017
$\sum \mathcal{C}$ [M]	2025	363	326	311	231
$\sum \mathcal{L}$ [M]	2025	336	252	305	202
Int. perf. [k/s]	454	413	390	417	385

Compared to the standard application situation ($r_l = 2, 4$ with LT) the CPU time has roughly doubled in the worst case scenario ($r_l = 2, 2, 2$, no LT). Since $\sum \mathcal{C}$ is only slightly larger in the latter case, this increase is obviously due to higher costs for orchestrating the mesh adaptation. It is worth pointing out that for straightforward finite volume schemes, as the one used here, the core idea of the AMR method of employing larger than necessary data patches to benefit from cache coherence and super-scalar processing units during the finite volume update works particularly well. To quantify this effect, the integration performance, the ratio of $\sum \mathcal{C}$ and the total time spent in *Integration* alone has been calculated. Unsurprisingly, the adaptive computations using less and larger refinement patches ($r_l = 2, 4$) are considerably closer to the unigrid integration performance.

In summary, the investigation of Table 2 illustrates that the performance of the AMR method does not solely depend on the number of updated total or leaf cells. Enforcing a minimal overall cell count by employing very deep refinement hierarchies or using a grid generation efficiency close to 1 usually results in a loss of computational performance. The specific choices of using a base mesh of 128×128 and a clustering efficiency of $\eta_{tol} = 80\,\%$ for the present study are quasi-optimal settings with respect to overall compute time, which was verified in additional computations (not shown here).

2.2. Visualization of the final solution

In Figure 5, the isolines of density for the final solution are plotted together with the adaptive grid. We observe that both solutions fit well with their respective reference solution given in Figure 4. The grids on the right side show that both methods adapt well to the discontinuities and steep gradients of the density. However, for the MR method, the lowest level reached is lower than 7, which is the level of the coarsest grid the AMR method. This results in a better compression for the MR method, especially when the maximal level is $L = 8$.

2.3. Error, speed-up, memory compression and overhead

The goal of an adaptive computation is to obtain the solution with a significant gain in CPU time and memory, while preserving the accuracy of the corresponding FV scheme on the regular finest grid. To assess the quality of an adaptive simulation, the discrete L_1-error is computed, using as reference the FV solution with the same space-time schemes on a 2048×2048 uniform grid, i.e. using $L = 11$ levels. Figure 4 shows the isolines of density for such reference solutions. We observe that both results differ slightly, which is likely due to the fact that the numerical fluxes used in MR and AMR method are not exactly identical.

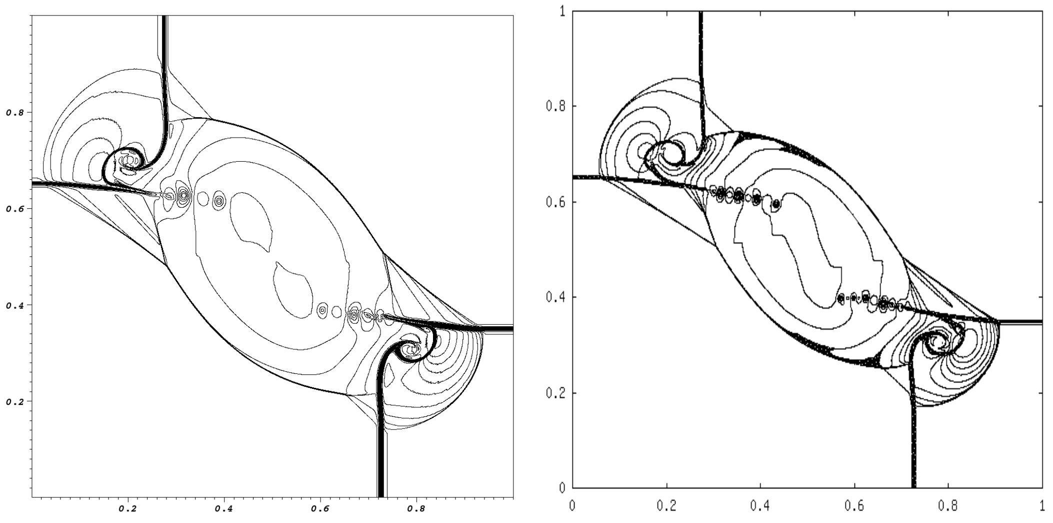

FIGURE 4. Isolines of density $\rho = 1.0, \cdots, 4.0$ every 0.1 for the fine-grid reference solution at $t = 0.3$, obtained with $CFL = 0.45$ and $L = 11$ levels, using the FV algorithms of the AMR (left) and MR schemes (right).

For the adaptive AMR case, the error is evaluated as the sum of the L_1-error norms on the domain Ω_l without higher refinement, i.e.,

$$L_1^e(Q) = L_1^e(\Delta x_L, \Omega_L) + \sum_{l=0}^{L-1} L_1^e(\Delta x_l, \Omega_l \setminus \Omega_{l+1}),$$

where

$$L_1^e(\Delta x, \Omega) = \sum_{i,j} |\mathbf{Q}_{(i,j)} - \mathbf{Q}^r_{(i,j)}| \Delta x^2,$$

denotes the L_1-norm on the domain Ω, and where $\mathbf{Q}^r_{i,j}$ denotes the projection of the reference solution from the 2048×2048 uniform mesh down to the desired mesh with step size Δx.

For the MR method, the adaptive solution is recursively projected up to the desired finest uniform grid of level L with a step size Δx_L. The goal is to obtain $\tilde{\mathbf{Q}}_{(i,j)}$ using the third order cell-average interpolation. Then the discrete error is evaluated on the domain Ω as

$$L_1^e(Q) = \sum_{i,j} |\tilde{\mathbf{Q}}_{(i,j)} - \mathbf{Q}^r_{(i,j)}| \Delta x_L^2,$$

where $\mathbf{Q}^r_{(i,j)}$ denotes the projection of the reference solution from the level $L = 11$ down to the desired level L.

Special counters were implemented to evaluate the performance of the two codes. The CPU time compression rate is defined as the ratio between the CPU time required to compute the final solution using the adaptive method and the one required to compute the same solution using the fine-grid method. In an adaptive simulation, an *average memory requirement* is defined as

$$\bar{\mathcal{C}} = \frac{1}{N_I} \sum_{n=1}^{N_I} \mathcal{C}^n$$

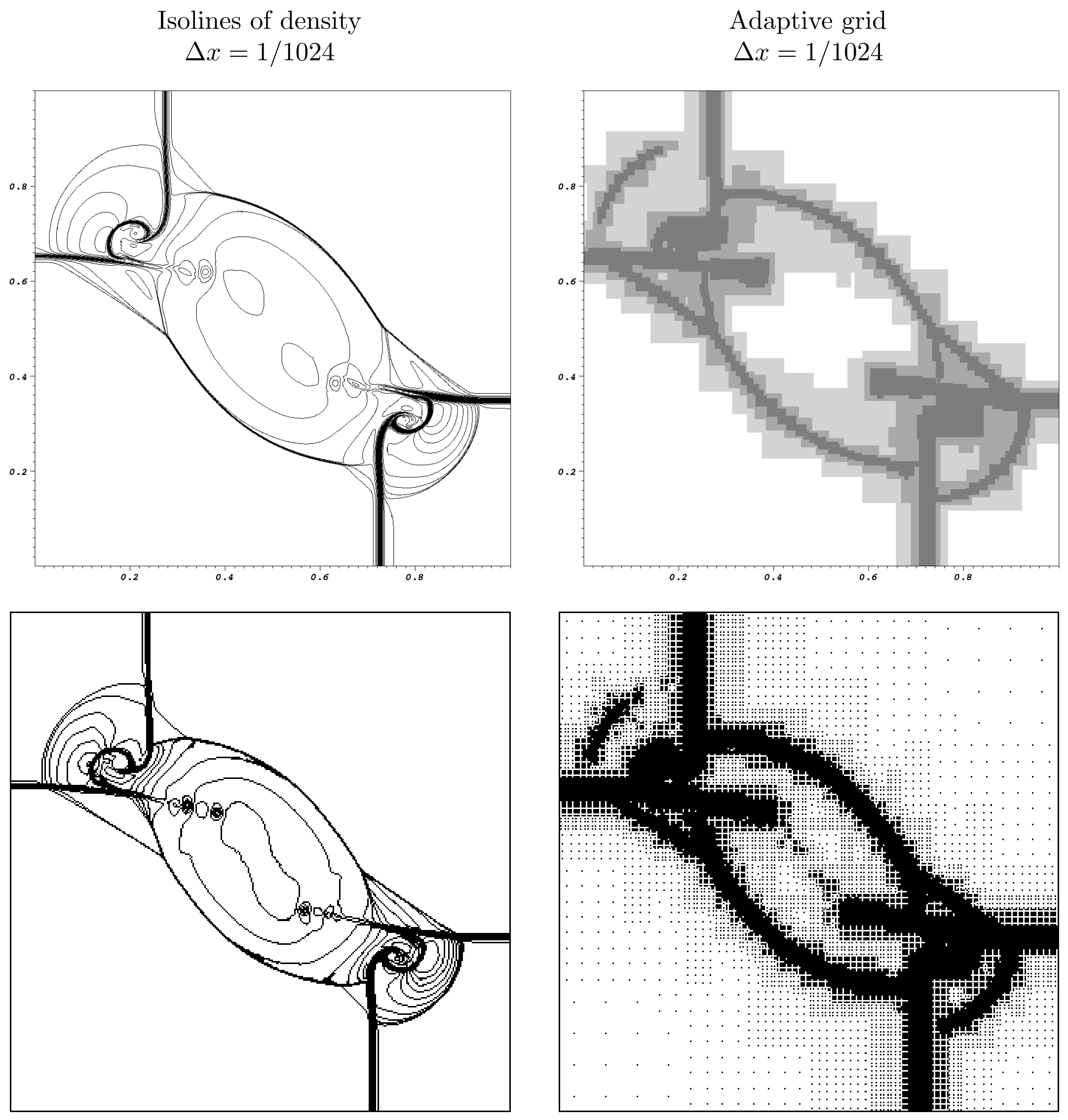

FIGURE 5. Isolines of density (left side) and corresponding adaptive mesh (right side) at $t = 0.3$ with $L = 10$, for the AMR scheme (top), and the MR scheme (bottom).

where N_I is the number of performed time steps, and $\mathcal{C}^n$ denotes the sum of cells of the entire hierarchy at $t = t^n$. Then, the *memory compression* is defined as the ratio of the average memory requirement and the number of cells N_C of the finest uniform grid.

In a FV code, the main contribution to the CPU time is the expensive numerical flux evaluation. One crucial question is to know whether the gain in CPU time due to the reduction of expensive flux computations in adaptive simulations is larger than the additional computational overhead induced by the adaptive algorithm.

To evaluate the overhead of the adaptive computations, we consider the number

$$\Gamma_{MR} = \frac{CPU\ Time}{\sum_{n=1}^{N_I} \mathcal{L}^n}$$

which denotes the average CPU time spent to evolve the solution in each cell of the computational domain, in each time step. Here $\sum_{n=1}^{N_I} \mathcal{L}^n$ denotes the sum of the leaves of the tree during the whole computation. Since the time evolution of the AMR method is performed in every cell of the different adaptive grids, we define

$$\Gamma_{AMR} = \frac{CPU\ Time}{\sum_{n=1}^{N_I} \mathcal{C}^n}$$

where $\sum_{n=1}^{N_I} \mathcal{C}^n$ denotes the sum of the cells of all the adaptive grids during the whole computation.

In adaptive computations, Γ_{MR} and Γ_{AMR} are expected to be larger than Γ_{FV} on the regular finest grid. Hence, the *overhead per iteration and per cell* of an adaptive computation is defined by

$$\bar{\Gamma}_{MR} = \frac{\Gamma_{MR}}{\Gamma_{FV}} - 1 \text{ and } \bar{\Gamma}_{AMR} = \frac{\Gamma_{AMR}}{\Gamma_{FV}} - 1 \, .$$

The average *overhead per iteration* is the *overhead per iteration and per cell* multiplied by the average memory compression for the AMR method, and by the average grid compression - i.e. the average number of leaves divided by the number of cells of the finest grid - for the MR method.

A summary of the MR and AMR results, obtained without and with local time stepping, is assembled in Tables 3 and 4, respectively. All the computations were run on the same double processor workstation.

With the chosen grid adaptation parameters, both adaptive methods give discrete L_1-errors of the same order. Both are comparable with the L_1-error of the FV scheme on the corresponding regular fine grid, as indicated in the second and third columns of Tables 3 and 4.

Table 3. Summary of the results for the MR and the AMR computations.

	FV	MR					
Level	$L_1^e(\rho)$	$L_1^e(\rho)$	Compression (%)			Overhead per iteration	
	$[10^{-2}]$	$[10^{-2}]$	CPU	Memory	Grid	(per leaf)	(%)
L=8	3.65	3.68	33.01	39.64	24.99	0.32	8.0
L=9	2.23	2.26	20.17	23.40	14.57	0.38	5.6
L=10	1.04	1.08	11.93	13.95	8.68	0.38	3.2

	FV	AMR					
Level	$L_1^e(\rho)$	$L_1^e(\rho)$	Compression (%)			Overhead per iteration	
	$[10^{-2}]$	$[10^{-2}]$	CPU	Memory	Grid	(per node)	(%)
L=8	3.91	3.92	82.54	50.79	44.34	0.63	31.75
L=9	2.34	2.37	55.62	28.61	23.02	0.94	27.00
L=10	1.22	1.30	37.98	16.09	12.46	1.36	21.90

NOTE: Computations are performed until $t = 0.3$, with $CFL = 0.45$. For the MR method, the wavelet threshold is $\epsilon = 0.01, 0.008, 0.005$ for $L = 8, 9$ and 10, respectively. For the AMR method, $\epsilon_p = \epsilon_\rho = 0.05$ and the coarsest level is 128×128.

As expected from the results of the previous subsection, we observe that the gain in both CPU time and memory compression is larger using the MR method than using the AMR one. This is particularly true for $L = 8$ levels. For $L = 10$ levels, the difference in terms of memory compression is slightly reduced while the difference

Table 4. Summary of the results for MR/LT and AMR/LT computations

	FV	MR/LT					
Level	$L_1^e(\rho)$	$L_1^e(\rho)$	Compression (%)			Overhead per iteration	
	$[10^{-2}]$	$[10^{-2}]$	CPU	Memory	Grid	(per leaf)	(%)
L=8	3.65	3.74	29.36	39.07	24.70	0.32	8.0
L=9	2.23	2.38	17.04	22.94	14.35	0.39	5.6
L=10	1.04	1.23	9.62	13.64	8.53	0.37	3.1

	FV	AMR/LT					
Level	$L_1^e(\rho)$	$L_1^e(\rho)$	Compression (%)			Overhead per iteration	
	$[10^{-2}]$	$[10^{-2}]$	CPU	Memory	Grid	(per node)	(%)
L=8	3.91	3.92	58.73	38.22	35.00	0.54	20.5
L=9	2.34	2.36	35.39	20.44	18.08	0.73	15.0
L=10	1.22	1.27	22.45	11.40	10.00	0.97	11.0

NOTE: Computations are performed until $t = 0.3$, with $CFL = 0.45$. For the MR/LT method, the wavelet threshold is $\epsilon = 0.01, 0.008, 0.005$ for $L = 8, 9$ and 10, respectively. For the AMR/LT method, $\epsilon_p = \epsilon_\rho = 0.05$ and the coarsest level is 128×128.

in term of CPU time remains large. On the other hand, the speed-up due to local time stepping is larger for the AMR method than for the MR method. Table 2 shows that especially for dyadic refinement factors and no local time stepping the difference between the number of leaf cells and the total cell count used by the AMR method is particularly large. Nevertheless, the gain in CPU time compression is still larger using the MR/LT method than using the AMR/LT method.

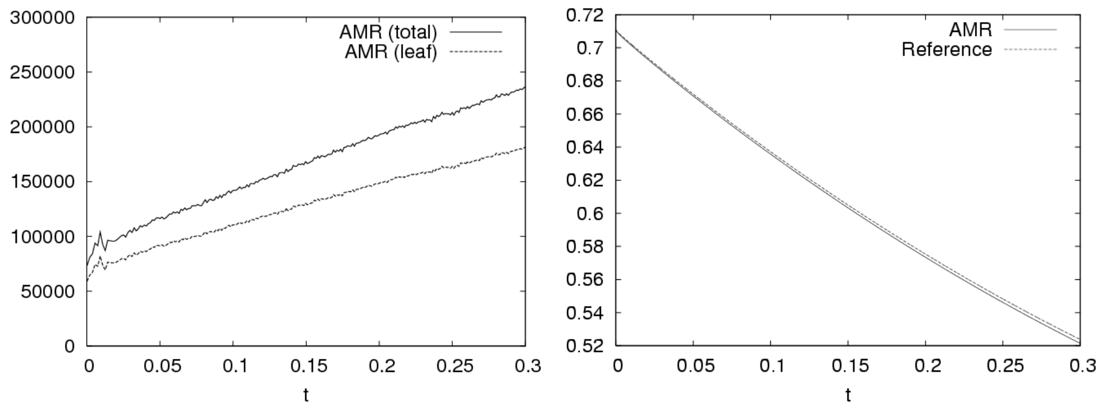

Figure 6. Time evolution of the number of used cells (left) and of the total kinetic energy (right) with $r_l = 2, 2, 2$ for the AMR method using $L = 10$ levels, together with the reference computation on the 2048^2 mesh.

Figures 6 and 7 show the time evolution of the number of used cells and the total kinetic energy for both method. They show that the kinetic energy curves match well with the reference solution, which confirms the accuracy of the method and the good grid convergence obtained on $L = 10$ scales. Naturally, the MR method requires less cells than the AMR one during computation.

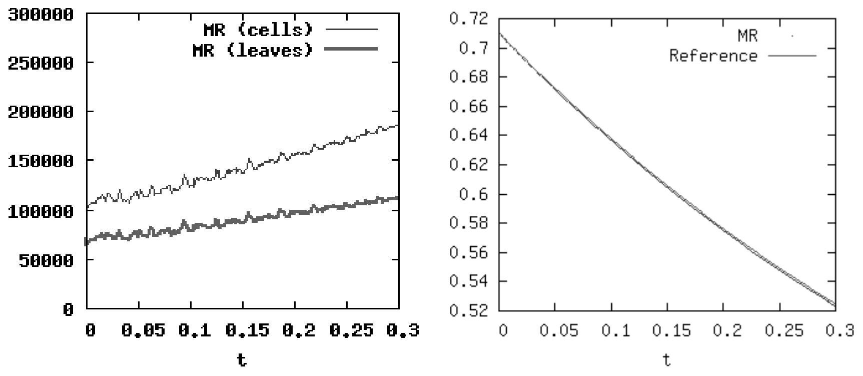

Figure 7. Time evolution of the number of used cells (left) and of the total kinetic energy (right) for the MR method using $L = 10$ levels, together with the reference computation on the 2048^2 mesh.

Conclusion

In the present paper, adaptive computations of the two-dimensional compressible Euler equations for a classical Riemann problem are presented. The goal is to compare, for the same accuracy, the efficiency in terms of CPU time and memory compression of two adaptive methods: the adaptive multiresolution (MR) method and the adaptive mesh refinement (AMR) method, first with a global time step, then with a scale-dependent local time step. Both methods are based on an explicit finite volume method on an adaptive grid, with second order schemes in space and time. The main difference is in the way the adaptive grid is stored: a graded tree data structure for the MR method and a series of regular data blocks on the different levels for the AMR method. The other main difference is the error estimator. It is based on the details, or wavelet coefficients, between two consecutive levels of the adaptive grid for the MR method, whereas it relies on scaled gradient criteria based on pressure and density for the AMR method.

For both methods, the threshold coefficients were chosen to lead approximately to the same accuracy. In the present paper, it is shown that, for this Riemann test-case, the MR method presents larger compression rates and larger gains in CPU time than the AMR method, using either global or local time stepping strategies. The improved compression rates observed for the MR method are a direct result of the patch-based refinement technique, used in the AMR method, that accepts a larger number of total cells to avoid data fragmentation. The objective of the AMR method is to sustain as much of the integration performance of an optimal Cartesian unigrid finite volume code on a dynamically adaptive mesh, however, the better CPU time compression rates of the MR show that there are limitations to this approach.

An advantage of the AMR method is that, nowadays, optimized libraries are available. Thus, the change of the finite volume scheme becomes straightforward and reduces largely to modifying the patch-based numerical update routine. In case of AMROC, most users just need to employ function interfaces that are literally identical to the ones typically employed in Cartesian unigrid codes; the AMR library orchestrates refinement and parallelization in a transparent way and calls the user-specified single grid routines as required (cf. [20, 24]).

Multiresolution methods have a rigorous and potentially more accurate regularity analysis [14, 40], while for AMR methods rigorous error estimators are not available. Therefore, threshold values of AMR have to be tuned for a given problem, whereas in MR, in principle, the threshold is independent of the problem. This was shown in previous papers for different physical problems: flame balls [48], flame ball-vortex interaction [46] and weakly compressible turbulent flows [47].

As perspectives, we plan to compare the efficiency and accuracy of both adaptive methods for three-dimensional problems, amongst others for the compressible Navier-Stokes equations, in order to evaluate these preliminary results for more complex configurations.

Acknowledgments

M.O. Domingues thankfully acknowledges financial support from the ANR project "M2TFP", École Centrale de Marseille (ECM), Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP), and CNPq - the Brazilian Research Council, Brazil. S. Gomes thankfully acknowledges financial support from ECM, CNPq and FAPESP. O. Roussel and K. Schneider acknowledge financial support from the French–German DFG–CNRS Research Program "LES and CVS of Complex Flows". K. Schneider thanks the ANR project "M2TFP" for financial support. O. Roussel also thankfully acknowledges FAPESP for financial support to visit Unicamp, when this paper was concluded. We are grateful to Dominique Fougère and Varlei E. Menconni for their helpful computational assistance.

References

[1] R. Abgrall and A. Harten. Multiresolution representation in unstructured meshes. *SIAM J. Numer. Anal.*, 35(6):2128–2146, 1998.
[2] I. Babuska and W.C. Rheinboldt. A posteriori error estimates for the finite element method. *Int. J. Numer. Meth. Engng.*, 12:1597–1615, 1978.
[3] R. Becker and R. Rannacher. An optimal control approach to a-posteriori error estimation. In A. Iserles, R. Becker, R. Rannacher, and P. G. Ciarlet, editors, *Acta Numerica*. Cambridge University Press, 2001.
[4] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comput.*, 15(1):127–138, 1994.
[5] M. Berger. *Adaptive Mesh Refinement for Hyperbolic Differential Equations*. PhD thesis, Stanford University. Report No. STAN-CS-82-924, Aug 1982.
[6] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1988.
[7] M. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35(6):2298–2316, 1998.
[8] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.
[9] B. L. Bihari. Multiresolution schemes for conservation laws with viscosity. *J. Comput. Phys.*, 123:207–225, 1997.
[10] B. L. Bihari and A. Harten. Multiresolution schemes for the numerical solution of 2-D conservation laws I. *SIAM J. Sci. Comput.*, 18(2):315–354, 1996.
[11] A. Brandt. Multilevel adaptive solutions to boundary value problems. *Math. Comp*, 31:333–390, 1977.
[12] J. D. Calle, P. R. Devloo, and S. M. Gomes. Wavelets and adaptive grids for the discontinuous Galerkin method. *Numer. Algor.*, 39:143–158, 2005.
[13] G. Chiavassa and R. Donat. Point value multi-scale algorithms for 2D compressible flow. *SIAM J. Sci. Comput.*, 23(3):805–823, 2001.
[14] A. Cohen. Wavelet methods in numerical analysis. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of Numerical Analysis*, volume VII. Elsevier, Amsterdam, 2000.
[15] A. Cohen, N. Dyn, S. M. Kaber, and M. Postel. Multiresolution finite volume schemes on triangles. *J. Comput. Phys.*, 161:264–286, 2000.
[16] A. Cohen, S. M. Kaber, S. Müller, and M. Postel. Fully adaptive multiresolution finite volume schemes for conservation laws. *Math. Comp.*, 72:183–225, 2003.
[17] W. Y. Crutchfield and M. L. Welcome. Object-oriented implementation of adaptive mesh refinement algorithms. *J. Scientific Programming*, 2:145–156, 1993.
[18] W. Dahmen, B. Gottschlich-Müller, and S. Müller. Multiresolution schemes for conservation laws. *Numer. Math.*, 88(3):399–443, 2001.
[19] R. Deiterding. AMROC - Blockstructured Adaptive Mesh Refinement in Object-oriented C++. http://amroc.sourceforge.net.
[20] R. Deiterding. *Parallel adaptive simulation of multi-dimensional detonation structures*. PhD thesis, Brandenburgische Technische Universität Cottbus, Sep 2003.
[21] R. Deiterding. *Adaptive Mesh Refinement - Theory and Applications*, volume 41 of *Lecture Notes in Computational Science and Engineering*, chapter Construction and application of an AMR algorithm for distributed memory computers, pages 361–372. Springer, 2005.
[22] R. Deiterding. A parallel adaptive method for simulating shock-induced combustion with detailed chemical kinetics in complex domains. *Comp. Struct.*, 87:769–783, 2009.
[23] 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.
[24] 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.

[25] M. O. Domingues, S. M. Gomes, and L. M. A Diaz. Adaptive wavelet representation and differenciation on block-structured grids. *Appl. Numer. Math.*, 47:421–437, 2003.

[26] M. O. Domingues, S.M. Gomes, O. Roussel, and K. Schneider. An adaptive multiresolution scheme with local time stepping for evolutionary PDEs. *J. Comput. Phys.*, 227:3758–3780, 2008.

[27] M. O. Domingues, S. M. Gomes, O. Roussel, and K. Schneider. Space-time adaptive multiresolution methods for hyperbolic conservation laws: Applications to compressible Euler equations. *Appl. Numer. Math.*, 2009. in press.

[28] M. O. Domingues, O. Roussel, and K. Schneider. An adaptive multiresolution method for parabolic pdes with time-step control. *Int. J. Numer. Meth. Engng.*, 78:652–670, 2009.

[29] 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.

[30] B. Gottschlich-Müller and S. Müller. Adaptive finite volume schemes for conservation laws based on local multiresolution techniques. In M. Fey et al., editors, *Hyperbolic problems: Theory, numerics, applications, Vol. I (Zürich, 1998)*, volume 129 of Int. Ser. Numer. Math., pages 385–394. Birkhäuser, Basel, 1999.

[31] A. Harten. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. *Comm. Pure Appl. Math.*, 48:1305–1342, 1995.

[32] M. Holmström. *Wavelet Based Methods for Time Dependent PDEs*. PhD thesis, Uppsala University, 1997.

[33] M. Holmström. Solving hyperbolic PDEs using interpolating wavelets. *SIAM J. Sci. Comput.*, 21(2):405–420, 1999.

[34] J. M. Hyman, S. Li, and L. R. Petzold. An adaptive moving mesh method with static rezoning for partial differential equations. *Comp. Math. Appl.*, 46(10-11):1511–1524, 2003.

[35] M. Kaibara and S. M. Gomes. A fully adaptive multiresolution scheme for shock computations. In E.F. Toro, editor, *Godunov Methods: Theory and Applications*, volume . Klumer Academic/Plenum Publishers, 2000.

[36] S. R. Kohn and S. B. Baden. A parallel software infrastructure for structured adaptive mesh methods. In *Proc. of the Conf. on Supercomputing '95*, December 1995.

[37] P. D. Lax and X. D. Liu. Solution of two-dimensional Riemann problems of gas dynamics by positive schemes. *SIAM J. Sci. Comput.*, 19(2):319–340, 1998.

[38] M. S. Liou. A sequel to AUSM, part II: AUSM+-up for all speeds. *J. Comput. Phys.*, 214:137–170, 2006.

[39] 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.

[40] S. Müller. *Adaptive multiscale schemes for conservation laws*, volume 27 of Lectures Notes in Computational Science and Engineering. Springer, Heidelberg, 2003.

[41] S. Müller and Y. Stiriba. Fully adaptive multiscale schemes for conservation laws employing locally varying time stepping. *J. Sci. Comput.*, 30(3):493–531, 2007.

[42] 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.*, 221:63–87, 2007.

[43] 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.

[44] P. Pinho, M. O Domingues, P. J. Ferreira, S. M. Gomes, A. Gomide, and J. R. Pereira. Interpolating wavelets and adaptive finite difference schemes solving Maxwell's equations: gridding effect. *IEEE Transactions in Magnetics*, 43:1013–1022, 2007.

[45] 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, 2000.

[46] O. Roussel and K. Schneider. An adaptive multiresolution method for combustion problems: application to flame ball - vortex interaction. *Comp. Fluids*, 34(7):817–831, 2005.

[47] O. Roussel and K. Schneider. Coherent vortex simulation of weakly compressible turbulent mixing layers using adaptive multiresolution methods. *submitted to J. Comput. Phys., in revision*, 2009.

[48] O. Roussel, K. Schneider, A. Tsigulin, and H. Bockhorn. A conservative fully adaptive multiresolution algorithm for parabolic PDEs. *J. Comput. Phys.*, 188(2):493–523, 2003.

[49] C. W. Schulz-Rinne, J. P. Collis, and H. M. Glaz. Numerical solution of the Riemann problem for two-dimensional gas dynamics. *SIAM J. Sci. Comput.*, 14:1394–1414, 1993.

[50] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics*. Springer, 1997.

[51] 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, 1982.

[52] Y. Wada and M. S. Liou. An accurate and robust flux splitting scheme for shock and contact discontinuities. *SIAM J. Sci. Comput.*, 18(3):633–657, 1997.

[53] T. Zhang and Y. Zheng. Conjecture on the structure of solutions the Riemann problem for two-dimensional gas dynamics systems. *SIAM J. Math. Anal.*, 21:593–630, 1990.