ADAPTIVE MULTIRESOLUTION METHODS

Margarete O. Domingues¹, Sônia M. Gomes², Olivier Roussel³ and Kai Schneider³,⁴

Abstract. These lecture notes present adaptive multiresolution schemes for evolutionary PDEs in Cartesian geometries. The discretization schemes are based either on finite volume or finite difference schemes. The concept of multiresolution analyses including Harten’s approach for point and cell averages is described in some detail. Then the sparse point representation method is discussed. Different strategies for adaptive time-stepping, like local scale dependent time stepping and time step control are presented. Numerous numerical examples in one, two and three space dimensions validate the adaptive schemes and illustrate the accuracy and the gain in computational efficiency in terms of CPU time and memory requirements. Another aspect, modeling of turbulent flows using multiresolution decompositions, the so-called Coherent Vortex Simulation ansatz is also described and examples are given for computations of three-dimensional weakly compressible mixing layers. Most of the material is assembled and adapted from previous publications [21, 23–25, 57, 58].

Contents

1. Introduction 2
2. Multiresolution representation 4
  2.1. Harten’s approach 5
  2.2. Multiresolution analyses for cell averages on uniform dyadic grids 12
  2.3. The lifting methodology 21
3. Sparse Point Representation (SPR) method 24
  3.1. Adaptive block representation (ABR) 25
  3.2. Operations on Block-Structured Grids 31
  3.3. Numerical Examples 32
4. FV/MR Method 36
  4.1. Finite volume discretization on uniform grids 36
  4.2. Conservative fully adaptive multiresolution scheme 39

¹ 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)
² Universidade Estadual de Campinas (UNICAMP), IMECC, Caixa Postal 6065, 13083-970 Campinas, São Paulo, Brazil (e-mail: soniag@ime.unicamp.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 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)

© EDP Sciences, SMAI 2002
1. Introduction

Adaptive numerical techniques are necessary tools to solve systems of nonlinear partial differential equations (PDEs) without wasting computational resources while preserving the accuracy of the solution.

PDEs naturally arise from mathematical modelling of chemical-physical problems encountered in many applications, like in meteorology or chemical industry. In turbulent reactive or non-reactive flows, for instance, the solutions of these PDEs usually exhibit several magnitudes of active spatial and temporal scales. Typically these scales are not uniformly distributed in the space–time domain, efficient numerical discretizations could take advantage of this property. Space-time adaptivity allows thus to reduce the computational complexity with respect to uniform discretizations, while controlling the accuracy of the adaptive discretization.

Different approaches have been investigated to define adaptive space discretizations, some emerge from ad hoc criteria, others call for more sophisticated a posteriori error estimators using control strategies by solving computational expensive adjoint problems [5,64]. Adaptive mesh refinement methods introduced by Berger and Oliger [7] are now widely used for many applications using structured or unstructured grids, see e.g. [6,44]. However, the data compression rate is high where the solution is almost constant, but remains low where the solution is smooth.

More recently, multiresolution based schemes (MR) have been first developed by Harten [34,35] for conservation laws. Now they are known to yield an appropriate framework to construct adaptive schemes for hyperbolic conservation laws. Harten’s approach has then been extended and further developed in different directions by Cohen et al [16], Kaibara and Gomes [40], Chiavassa and Donat [10], Müller [46], Roussel et al [55,57]. The main idea of the MR method is to use a multiresolution data representation. The decay of the MR coefficients yields information on local regularity of the solution. Therewith the truncation error can be estimated and coarser grids can be used in regions where this error is small and the solution is smooth. An adaptive grid can be introduced by suitable thresholding of the multiresolution representation where only significant coefficients are retained. Hence a given discretization on a uniform mesh can be accelerated as the number of costly flux evaluations is significantly reduced, without contaminating the accuracy of the discretization. The memory requirements could also be reduced, for example using a dynamic tree data structure. An overview of the different MR methods can be found, e.g. in the books of Cohen [11] and Müller [46].
A main drawback of most of these space–adaptive methods, which mostly employ explicit or semi–explicit time discretizations, is that the finest spatial grid size imposes a small time step in order to fulfill the stability criterion of the time scheme. Hence, for extensive grid refinement with a huge number of refinement levels, a very small size of the time step is implied.

To overcome this difficulty different strategies have been pursued to introduce adaptive time stepping for space adaptive discretizations of PDEs. Osher and Sanders introduced local time stepping for one dimensional scalar conservation laws where the space discretization is non–uniform but fixed [50]. Extensions of this approach have been presented in Dawson and Kirby [20] for second order Runge–Kutta schemes using a predictor-corrector type local time stepping, which has been further improved by Tang and Warnecke [66]. Space-time mesh refinement for the one-dimensional wave equation based on the conservation of a discrete energy is proposed in [17, 18]. In [26] a local time stepping algorithm for discontinuous Galerkin methods is presented. Each element uses its optimal time step given by the local stability condition without requiring synchronization between the elements.

In the context of adaptive multiresolution and wavelet methods, Bacry et al [4] first introduced a scale-dependent time step. They applied this method to linear parabolic equations and to the Burgers equation. A stability analysis of this scheme has been conducted for the heat equation in [9]. It is shown that the adaptive time stepping strategy does not affect the stability of the scheme.

More recently, Müller and Stiriba [48] presented a fully adaptive multiresolution finite volume scheme with a locally varying time stepping. For time discretization one stage methods, either explicit or implicit Euler schemes are used. A linear combination, leading to a Crank-Nicholson scheme, yields second order accuracy. Applications for one dimensional conservation laws are discussed to illustrate the efficiency and accuracy of the scheme. A pure space–time Galerkin approach for viscous Burgers equation where the time axis is treated like a space direction has been introduced by Alam et al [2]. Results for one space dimension look promising, however the extension of this method to higher dimensions seems questionable as it could be expensive in memory storage.

The aim of the lecture notes is to present in a concise way adaptive multiresolution methods for evolutionary PDEs. In addition to space adaptivity we show that local scale-dependent time stepping can yield additional speed-up. The idea is hereby to introduce at large scales larger time steps without violating the stability condition of the explicit time scheme, which results in less flux evaluation due to larger time steps.

Most of the presented material is extracted and based on journal publications [21, 23–25, 57, 58].

In the following three kinds of adaptive strategies are considered to speed up adaptive multiresolution schemes.

a. **Space adaptivity (MR):** A second order finite volume scheme (FV) is applied on dynamically adapted grids. We consider space adaptivity in the multiresolution (MR) context of wavelet analysis for cell averages. The main idea of such a MR scheme is to use the decay of the wavelet coefficients to obtain information on local regularity of the solution. Therewith, coarser grids can be used in regions where these coefficients are small and the solution is smooth, while fine grids are used where the coefficients are significant and the solution has strong variations. For an efficient MR representation, we adopt a data structure which is organized as a dynamic graded tree, as proposed in [57].

b. **Time step control (CTS):** The time integration is performed with variable time steps, where the time step size selection is based on estimated local truncation errors obtained by the computation of two solutions with embedded ODE solvers of different orders.

c. **Local time stepping (LTS):** The time evolution uses scale-dependent time steps. Instead of evolving the solution with a single time step $\Delta t$ on all grid cells, computational work is saved if the solution is integrated with different time steps, according to each cell scale: if $\Delta t$ is used for the cells in the finest level, then a double time step $2\Delta t$ is used in the coarser level with double spacing. Required missing values in ghost cells are interpolated in intermediary time levels.

In [30] Ferm and Löstedt considered such kinds of adaptive strategies for hyperbolic problems in one space dimension. However, instead of MR space adaptivity, there the grid adaptation is based on the control of local discretization errors in space, which are estimated by comparing the space discretization on two different grids.
To simplify the data structure, the spatial grids are dynamically refined or coarsened in blocks of grids, in the spirit of the so-called Adaptive Mesh Refinement method [7].

An analysis of the performance of the following adaptive multiresolution schemes is presented:

- MR: adaptive multiresolution scheme with the same constant time step at each every scale;
- MR/CTS: MR scheme with time step control, but the same time step at each every scale;
- MR/LTS: MR scheme with scale dependent local time stepping, but remaining constant in time;
- MR/CTS/LTS: MR scheme with time step control and local time stepping.

We also review the Sparse Point Representation method (SPR) which is concerned with adaptive finite difference schemes using interpolating multiresolution analysis [23, 39, 52].

Application of the different adaptive schemes are given for various equations in one, two and three space dimensions. We demonstrate their accuracy and computational efficiency in terms of CPU time and memory requirements.

Finally, we present a modeling part using the multiresolution representation to reduce the number of degrees of freedom for computing turbulent flows.

The Coherent Vortex Simulation (CVS), originally introduced by Farge & Schneider [27, 28], is described here for weakly compressible flows. For an overview on multiresolution techniques for computational fluid dynamics we refer the reader to a recent review of Schneider & Vasilyev [61].

The lecture notes are organized as follows. After describing the multiresolution representation, including Harten’s approaches for point values and cell averages, we describe sparse point representations.

Then the coupling of finite volume schemes with adaptive multiresolution strategies is described starting with the discretization on uniform grids. Dynamical tree data structures are discussed, conservative flux evaluations are described and an error analysis is performed for linear equations which yields an estimate for the threshold.

Different time adaptive strategies are then discussed and their coupling with the multiresolution scheme is detailed.

In the applications’ section we present various adaptive computations of different equations. We show results for convection-diffusion equations, for the compressible Euler equations, for reaction–diffusion equations in one, two and three space dimensions and also CVS computations of turbulent mixing layers.

For numerous cases the accuracy, CPU time and memory compression are discussed. Finally, conclusions are drawn and some perspectives for future investigations are discussed.

2. MULTiresOLUTION REPRESENTATION

In a discrete multiscale framework, data are represented in different scale levels, and the main tools are appropriate transformations relating the information $f_J$ at the finest scale level $J$ to the lower ones, and vice versa. After a transformation the output contains the information $f_0$ in the coarsest level, and $d_j$ that keeps the details between a scale level $j$ and the next upper level $j + 1$. Schematically, we have

$$f_J \leftrightarrow f^R_J = (f_0, d_0, \ldots, d_{J-1}).$$

The MR direct and inverse transforms are usually associated to biorthogonal multiresolution representations of functional spaces of the form

$$V_J = V_0 + \sum_{j=0}^{J} W_j$$

$$\sum_k f_{J,k} \Phi_{J,k} = \sum_k f_{0,k} \Phi_{0,k} + \sum_j \sum_k d_{j,k} \Psi_{j,k},$$
where the coefficients are interpreted as the inner products with dual basis

\[ f_{j,k} = \langle u, \Phi_{j,k}^* \rangle, \quad d_{j,k} = \langle f, \Psi_{j,k}^* \rangle. \]

In the classical approach, the content of \( f_j \) comes from local informations at uniformly sampled sites of a Cartesian grid \( X_j \), and the inter-level transformations are relied on convolutions with low and high pass filters. We consider here the construction of multiresolution analyses using two other different approaches: the formulation by A. Harten [1, 35], and the lifting methodology proposed in [65].

Harten’s general framework for the construction of multiresolution representations of data is based on the concepts of discretization, restriction and prediction operators. This framework is described in Section 2.1, where discretization by point values and cell averages are considered for uniform grids on the interval. As an illustration in non-Cartesian setting, we use Harten’s approach to describe some examples of multiresolution analyses for triangular geometry.

In the lifting methodology the idea is to start from a simple framework, Haar-like wavelets for instance, and use the lifting scheme (or dual lifting) [65] to obtain a new biorthogonal multiresolution analysis with better performance, for instance, with higher degree of polynomial cancellation. The basic ideas of this methodology are described in Section 2.3.

2.1. Harten’s approach

In [1, 35], a general framework for the construction of multiresolution representations of data is presented, which is based on the concepts of discretization, restriction and prediction operators.

To illustrate this methodology, we consider the case of discretization by point values and cell averages for uniform grids in the unit interval \( \Omega = [0, 1] \). This methodology can be used in more general contexts, for grids in higher dimensions and irregular geometry.

2.1.1. Multiresolution analyses for point values on uniform dyadic grids

Let us start by considering in the unit interval the hierarchy of uniform dyadic grids

\[ X_j = \{ x_{j,k} = k2^{-j}, \quad k = 0, \ldots, 2^j \} \]

with spacing \( 2^{-j} \). As indicated in Figure 1, this means that to go from \( X_j \) to a more refined grid \( X_{j+1} \) we add to \( X_j \) the new midpoints \( (2k+1)2^{-j-1} \) between the old points \( k2^{-j} \) and \( (k+1)2^{-j} \), halving the step size.

\[ \begin{align*}
X_4 & \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \\
X_3 & \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \\
X_2 & \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \\
X_1 & \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \\
X_0 & \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \\
\end{align*} \]

**Figure 1.** Dyadic grids on the interval.

In this geometry, at each scale level \( j \), the discretization operator \( D_j : f \rightarrow f_j \) associates to a continuous function \( f(x) \) on the interval \([0, 1]\) a vector \( f_j \) formed by its sample values \( f_{j,k} = f(x_{j,k}) \) at the points in \( X_j \).

For interchange of information between consecutive levels \( j \) and \( j+1 \), restriction and prediction operators are required

\[ P_{j+1 \rightarrow j} : f_{j+1} \rightarrow f_j \]

\[ P_{j \rightarrow j+1} : f_j \rightarrow \tilde{f}_{j+1} \]
Restriction (or decimation) is the operation that goes from level \( j + 1 \) to \( j \) by just doing decimation

\[
f_{j,k} = f_{j+1, 2k}.
\]

Prediction is a more subtle operation that, from the knowledge of its sample values at \( X_j \), gives estimated values \( \tilde{f}_{j+1, 2k+1} \approx f_{j+1, 2k+1} \) at the new odd locations in \( X_{j+1} \).

**Interpolating predictions**

The main properties required for the interpolating prediction operator are:

- **Localization**: the computation of \( \tilde{f}_{j+1, 2k+1} \) only requires the samples \( f_{j,m} \) on a close neighborhood of \( x_{j+1,2k+1} \).

- **Reproduction of polynomials**: the prediction is exact for polynomials of a prescribed degree.

A particular interest emerges regarding predictions by polynomial Lagrange interpolation. The most simple example is given by linear interpolation

\[
\tilde{f}_{j+1, 2k+1} = \frac{f_{j,k} + f_{j,k+1}}{2}.
\]

Generally, for \( M = 2L, L > 1 \), the idea is to define \( \tilde{f}_{j+1, 2k+1} \) by the evaluation at \( x_{j+1,2k+1} \) of the polynomial \( p(x) \) of degree \( M-1 \) that interpolates \( f_{j,m} \) at the \( M \) points \( x_{j,m} \in X_j \) that are as close as possible to \( x_{j+1,2k+1} \). Far from the boundary of the interval, and for even \( M \), we are in the classical iterative subdivision scheme by Dubuc-Deslauriers, where the interpolation stencils are centered around \( x_{j+1,2k+1} \), that is, \( x_{j,m}, -M/2 \leq m \leq M/2 \). It is clear that such procedure will reproduce polynomials of degree \( M-1 \).

As an example, consider the case of cubic interpolation (\( M = 4 \)). In the middle of the interval, the formula reads

\[
\tilde{f}_{j+1, 2k+1} = \frac{9}{16}(f_{j,k} + f_{j,k+1}) - \frac{1}{16}(f_{j,k-1} + f_{j,k+2}), \quad 0 < k < 2^j.
\]

Close to the left boundary, at \( x_{j+1,1} \), we chose the interpolation stencil \( x_{j,m}, m = 0, 1, 2 \) and 3, resulting

\[
\tilde{f}_{j+1,1} = \frac{5}{16}f_{j,0} + \frac{15}{16}f_{j,2} - \frac{5}{16}f_{j,3} + \frac{1}{16}f_{j,4},
\]

and on the right, at \( x_{j+1,2^{j+1}-1} \), we chose the interpolation stencil \( x_{j,2^j-m}, m = 0, 1, 2 \) and 3, and reverse the formula to get

\[
\tilde{f}_{j+1,2^{j+1}-1} = \frac{1}{16}f_{j,2j-3} - \frac{5}{16}f_{j,2j-2} + \frac{15}{16}f_{j,2j-1} + \frac{5}{16}f_{j,2j}.
\]

**MR transformations**

To construct multiresolution analyses for point values, the difference of information between a scale level \( j \) and the next upper level \( j + 1 \) occurs at the new points \( x_{j+1, 2k+1} \in \Lambda_j = X_{j+1} \setminus X_j \). Therefore, to measure these details, wavelet coefficients \( d_{j} \) are defined as the prediction errors at such locations, as indicated in Figure 3.

If we do this \( J \) times, from level \( J - 1 \) to 0, we obtain the details \( d_j \), and the signal \( f_0 \), at the coarsest level, as indicated in Figure 4. In Figure 5 the structure of the reverse process the process is shown, which starts from \( f_0 \) and \( d_j, j = 0, \ldots, J - 1 \), and recovers \( f_J \).

**Analysis - \( f_j \rightarrow f_j^{MR} = (f_0, d_0, \ldots, d_{J-1}) \). For \( j = J-1, \ldots, 0 \), given \( f_{j+1} \):**

1. Do restriction: \( P_{j+1-j} : f_{j+1} \rightarrow f_j \)
2. Do prediction: \( P_{j-j+1} : f_j \rightarrow \tilde{f}_j \)
3. Compute prediction errors: \( d_{j,k} = f_{j+1, 2k+1} - \tilde{f}_{j+1, 2k+1} \)
Figure 2. Prediction by linear (left side) and cubic interpolation (right side). The operation insert approximated values in-between the old values.

Figure 3. Wavelet coefficient as the error in the linear prediction case.

Figure 4. Structure of the analysis algorithm.

Synthesis - $f_{MR}^{j} = (f_0, d_0, \ldots, d_{J-1}) \rightarrow f_J$. For $j = 0, \ldots, J - 1$, given $f_j$, $d_j$:

1. Do prediction: $P_{j\rightarrow j+1}: f_j \rightarrow \tilde{f}_{j+1}$
2. Compute $f_{j+1}$
   
   $f_{j+1,2k} = f_{j,k}$
   
   $f_{j+1,2k+1} = d_{j,k} + \tilde{f}_{j+1,2k+1}$
Local regularity wavelet indicator

As interpolation errors, wavelet coefficients contain information about the local regularity of the analyzed function \( f(x) \). For instance, denoting by \( I_{j,k} \) the smallest interval containing the interpolation stencil used to compute \( f_{j+1,2k+1} \), the approximation theory by polynomial interpolation states that, for \( 1 \leq s \leq M-1 \), there exists a constant \( K = K(s, M) \) such that the estimation

\[
|d_{j,k}| \leq K 2^{-(s+1)j} \max_{\xi \in I_{j,k}} |f^{(s+1)}(\xi)|
\]

holds for functions \( f \in C^s(I_{j,k}) \), having bounded derivative of order \( s + 1 \).

This result can be used to estimate the degree of regularity of a function at a certain location by analyzing the decay rate of the magnitude of the wavelet coefficients associated to points in such zone. As an example, consider the function

\[
f(x) = \begin{cases} 
8.1 e^{1/4} e^{-|x-1/2|}, & x \leq 1/4, \\
9 e^{-|x-1/2|}, & 1/4 \leq x \leq 3/4, \\
e^{-|x-1/2|} (16x^2 - 24x + 18), & x \geq 3/4,
\end{cases}
\]

which presents discontinuities at \( \xi = 1/4 \), of \( f' \) at \( \xi = 1/2 \), and of \( f'' \) at \( \xi = 3/4 \). Figure 6 shows the plot of \( f(x) \) on the left side. On the right side, the position of the significant wavelet coefficients \( |d_{j,k}| \leq 5 \times 10^{-4} \) are plotted on the plane position \( x \) scale, which clearly indicates the location of the singularities. Now consider sufficiently localized \( \delta \) neighborhoods of the singularities \( \mu \), and define \( A_j(\xi) = \max\{|d_{j,k}| : |x_{j+1,2k+1} - \xi| < \delta\} \). Figure 7 shows the plots of \( \log_2 A_j(\xi) \), which have slopes approximately \( \alpha = -0.0004, -1.0149 \) and \(-1.9538 \) for \( \xi = 1/4, 1/2 \) and 3/4, respectively, confirming the expected values \( \alpha = 0, -1 \) and -2.

Functional context. For discretization by point values, we can define multiresolution functional decompositions like (1-2). It is clear that

\[
f_{j,k} = f(x_{j,k}) = \langle f, \Phi_{j,k}^* \rangle,
\]

where \( \Phi_{j,k}^* \) is the delta distribution at \( x_{j,k} \). For the definition of the dual wavelets \( \Psi_{j,k}^* \) such that

\[
d_{j,k} = f_{j+1,2k+1} - f_{j+1,2k+1} = \langle f, \Psi_{j,k}^* \rangle
\]

it implies that

\[
\Psi_{j,k}^* = \Phi_{j,2k+1}^* - \sum_m h_m^k \Phi_{j,m}^*,
\]

where \( h_m^k \) are the interpolating prediction coefficients, such that

\[
f_{j+1,2k+1} = \sum_m h_m^k f_{j,m}.
\]
The function \( f(x) \) (left side) and the position of the significant wavelet coefficients \( |d_{j,k}| \leq 5 \times 10^{-4} \).

For instance, at locations in the center of the interval, \( h_m^k = \{1/2, 1/2\} \) for linear interpolation, and \( h_m^k = \{-1/16, 9/16, 9/16, -1/16\} \) for cubic interpolation.

The definition of the primal basic functions \( \Phi_{j,k} \) and \( \Psi_{j,k} \) relies on the convergence of the prediction operator. This means that for any starting sequence \( s_j = (s_{j,k}) \), there exist a continuous function \( s_j(x) \) such that \( s_j(x_{\ell,k}) = s_{\ell,k} \), for \( \ell \geq j \), where \( s_{\ell} \) is defined by interactive subdivision \( s_{\ell} = P_{\ell-1-\ell}s_{\ell-1} \). This property holds for predictions by polynomial Lagrange interpolation of degree \( M - 1 \). Let the basic function \( \Phi_{j,k}(x) \) be defined by the iterative subdivision scheme of the delta-sequence \( s_{j,m} = \delta(k - m) \), which means that they satisfy the interpolating property

\[
\Phi_{j,k}(x_{j,m}) = \delta(k - m).
\]

The scaling relation

\[
\Phi_{j,k}(x) = \sum_m c_k(m) \Phi_{j+1,m}(x),
\]

where \( c_k(m) = \Phi_{j,k}(x_{j+1,m}) \), holds as a consequence of the fact that the functions in both sides coincide at \( X_{j+1} \). Observe that the scaling coefficients are \( c_k(2m) = \delta(m - k) \), and \( c_k(2m + 1) = h_k^m \). Figure 8 displays the scaling functions \( \Phi_{j,k}(x) \) for linear and cubic interpolations, at level \( j = 5 \), at the center of the interval. The interpolating scaling functions at level \( j = 5 \), interacting with the left boundary, for cubic predictions, are shown in Figure 9. On the right side, they are obtained by reversing those of the left side.
Figure 8. Interpolating scaling functions for linear (left side) and cubic predictions (right side), at level $j = 5$.

Figure 9. Interpolating scaling functions at level $j = 5$, interacting with the left boundary, for cubic predictions.

By defining the functional subspaces

$$V_j = \text{span}\{\Phi_{j,k}(x), \; k = 0, \cdots, 2^j\} \subset C[0,1],$$

it is clear that $V_j \subset V_{j+1}$. For functions $f \in C[0,1]$ consider the sequence of approximations

$$f_j(x) = \sum_k f_{j,k}\Phi_{j,k}(x),$$

which correspond to an interpolation operators of $f$ in $V_j$. It can be proved that

$$\|f_j\|_{\infty} \leq C\|f\|_{\infty}$$

$$\|f - f_j\|_{\infty} \to 0, \text{ as } J \to \infty$$
Consider $\Psi_{j,k}(x) = \Phi_{j+1,2k+1}(x)$, the scaling functions associated with the points $x_{j+1,2k+1} \in \Lambda_j$, and the space

$$W_j = \text{span}\{\Psi_{j,k}(x), \ k = 0, \ldots, 2^j - 1\},$$

which is formed by those functions in $V_{j+1}$ that vanish on $X_j$. Therefore, it results that $V_{j+1}$ is the direct sum of the spaces $V_j$ and $W_j$. And since the difference between two consecutive reconstructions $f_{j+1}(x) - f_j(x)$ vanishes at $X_j$ and coincides with $d_{j,k} = f_{j+1,2k+1} - f_{j+1,2k+1}$ at $\Lambda_j$, then the two-level direct-sum representation holds

$$V_{j+1} = V_j + W_j$$

$$\sum_k f_{j+1,k}\Phi_{j+1,k} = \sum_k f_{j,k}\Phi_{j,k} + \sum_k d_{j,k}\Psi_{j,k}.$$

Figure 10 illustrates the two-level decomposition for interpolating multiresolution analysis for $j = 4$, associated to linear predition.

Interactively applying this decomposition, the MR direct (analysis) and inverse (synthesis) transforms are associated to the multiresolution representation

$$V_J = V_0 + \sum_{j=0}^J W_j$$

$$\sum_k f_{j,k}\Phi_{j,k}(x) = \sum_k f_{0,k}\Phi_{0,k}(x) + \sum_{j=0}^J \sum_k d_{j,k}\Psi_{j,k}(x).$$

Data compression. Given the point values $f_J$ on the finest level $J$, after the analysis step of the MR transform, they can be represented as

$$f_J^{MR} = (f_0, d_0, \ldots, d_{J-1}).$$
Given threshold parameters $\epsilon_j$, data compression is obtained by the thresholding operation

$$d_{j,k}^\epsilon = \begin{cases} 0 & \text{if } |d_{j,k}| \leq \epsilon_j \\ d_{j,k} & \text{else} \end{cases}$$

The two main properties of the prediction operator, namely, localization and polynomial reproduction, have an immediate consequence on the size of the wavelet coefficients in smooth regions. For instance, for functions as the one in Figure 6, it is expected that the compressed multiresolution representation

$$f_{j}^{MR,\epsilon} = (f_0^\epsilon, d_0^\epsilon, \ldots, d_{J-1}^\epsilon)$$

has a reduced number of non-vanishing terms.

Stability. There is another important key aspect in wavelet compression: after the application of the inverse MR transform to the compressed data $f_{j}^{MR,\epsilon}$, how the reconstructed values $f_j$ compare to the original ones? Note that a perturbation in the wavelet coefficients at level $j$ is transmitted to higher levels by successive predictions. Could such perturbation be amplified by this process without control?

In the context of convergent subdivision schemes, we may use the expansion

$$f_{j}(x) - f_{j}(x) = \sum_{j=0}^{J-1} \sum_{k} (d_{j,k} - d_{j,k}^\epsilon) \Psi_{j,k}(x)$$

to obtain the stability result

$$\|f_{j} - f_{j}^\epsilon\|_\infty \leq \sum_{j=0}^{J-1} \epsilon_j \|\Psi_{j,k}\|_\infty \leq C \sum_{j=0}^{J-1} \epsilon_j$$

2.2. Multiresolution analyses for cell averages on uniform dyadic grids

The dyadic grids $X^j$ form a hierarchy of partitions of the interval $[0, 1] = \cup_k \Omega_{j,k}$ by disjoint cells $\Omega_{j,k} = (k2^{-j}, (k+1)2^{-j})$, $k = 0, \ldots, 2^j - 1$, as illustrated in Figure 11. Consider now the setting where the discretization operator $D_j : f \rightarrow f_j$ is defined for absolutely integrable functions $f$ in $[0, 1]$ by the cell averages on the partition at level $j$

$$f_{j,k} = 2^j \int_{\Omega_{j,k}} f(x) \, dx.$$
Since each cell $\Omega_{j,k}$ is formed by two children cells, of equal size, at level $j + 1$, that is,

$$\Omega_{j,k} = \Omega_{j+1,2k} \cup \Omega_{j+1,2k+1},$$

it is clear that the restriction operation $P_{j+1 \rightarrow j} : f_{j+1} \rightarrow f_{j}$, that gives the exact cell averages at level $j$ from the corresponding values at level $j + 1$, is defined by

$$f_{j,k} = \frac{1}{2}[f_{j+1,2k} + f_{j+1,2k+1}].$$

Cell-averages predictions. Now the purpose of the prediction operation $P_{j \rightarrow j+1} : f_{j} \rightarrow \tilde{f}_{j+1}$ is to estimate the cell averages at level $j + 1$ from the knowledge of the corresponding values at the coarser level $j$. The simplest way for predicting cell averages is by constant approximation, using the cell average of the mother cell $\Omega_{j,k}$ to predict the cell averages of its children cells $\Omega_{j+1,2k}$ and $\Omega_{j+1,2k+1}$

$$\tilde{f}_{j+1,2k} = \tilde{f}_{j+1,2k+1} = f_{j,k},$$

which is exact for constant functions. If we want exact reproduction of higher order polynomials, in addition to the cell average of the mother cell $\Omega_{j,k}$, we must also use cell averages of some neighbour cells. For instance, by including the cell averages of the two closest neighbours $\Omega_{j,k+1}$ and $\Omega_{j,k-1}$, for $1 \leq k \leq 2^j - 2$, the predicted values

$$\tilde{f}_{j+1,2k+1} = f_{j,k} + \frac{1}{8}[f_{j,k+1} - f_{j,k-1}],$$

$$\tilde{f}_{j+1,2k} = f_{j,k} - \frac{1}{8}[f_{j,k+1} - f_{j,k-1}]$$

result to be exact for quadratic polynomials. The restriction and prediction operations are illustrated in Figure 12.
Similarly, on the first cell of the left side we have

\[
\tilde{f}_{j+1,1} = \frac{11}{8} f_{j,0} - \frac{1}{2} f_{j,1} + \frac{1}{8} f_{j,2},
\]

\[
\tilde{f}_{j+1,0} = \frac{5}{8} f_{j,0} + \frac{1}{2} f_{j,1} - \frac{1}{8} f_{j,2},
\]

and on the right side

\[
\tilde{f}_{j+1,2^{j+1}+1} = -\frac{1}{8} f_{j,2^{j+1}-3} + \frac{1}{2} f_{j,2^{j+1}-2} + \frac{5}{8} f_{j,2^{j+1}-1},
\]

\[
\tilde{f}_{j+1,2^{j+1}+2} = -\frac{1}{8} f_{j,2^{j+1}-3} - \frac{1}{2} f_{j,2^{j+1}-2} + \frac{11}{8} f_{j,2^{j+1}-1},
\]

It can be verified that this prediction operator is derived from cell average polynomial interpolation in the sense that

\[
\tilde{f}_{j+1,2k+1} = 2^{j+1} \int_{\Omega_{j+1,2k+1}} p(x) \, dx
\]

\[
\tilde{f}_{j+1,2k} = 2^{j+1} \int_{\Omega_{j+1,2k}} p(x) \, dx
\]

where \( p(x) \) is the quadratic polynomial whose cell averages on the three stencil cells \( \Omega_{j,m} \) coincide with \( f_{j,m} \)

\[
f_{j,m} = 2^j \int_{\Omega_{j,m}} p(x) \, dx.
\]

This procedure may be extended to derive higher order predictions, using larger stencils which include \( \Omega_{j,k} \) and \( 2L \) closest neighbour cells \( \Omega_{j,k}, L > 1 \). Firstly, define \( p(x) \) as the polynomial of degree \( 2L \) whose cell averages on the stencil cells \( \Omega_{j,m} \) coincide with \( f_{j,m} \), and then define \( \tilde{f}_{j+1,2k} \) and \( \tilde{f}_{j+1,2k+1} \), as the cell average of \( p(x) \) on the children cells \( \Omega_{j+1,2k} \) and \( \Omega_{j+1,2k+1} \). It is clear that such procedure will reproduce polynomials of degree \( M = 2L \).

MR transformations. As in the case of discretization by point values, to construct multiresolution analyses for cell averages we may keep the details between a scale level \( j \) and the next upper level \( j+1 \) as the prediction error at one of the children cells. For instance, \( d_{j,k} = f_{j+1,2k+1} - \tilde{f}_{j+1,2k+1} \).

Given \( f_j \) and \( d_j \), \( f_{j+1} \) may be recovered, as indicated in the next algorithms.

**Analysis** - \( f_j \rightarrow f_j^{MR} = (f_0, d_0, \ldots, d_{j-1}) \).

- For \( j = J - 1, \ldots, 0 \), given \( f_{j+1} \):
  - Do restriction: \( P_{j+1-j} : f_{j+1} \rightarrow f_j \)
  - Do prediction: \( P_{j-j+1} : f_j \rightarrow \tilde{f}_{j+1} \)
  - Compute prediction errors: \( d_{j,k} = f_{j+1,2k+1} - \tilde{f}_{j+1,2k+1} \)

**Synthesis** - \( f_j^{MR} = (f_0, d_0, \ldots, d_{j-1}) \rightarrow f_j \).

- For \( j = 0, \ldots, J - 1 \), given \( f_j, d_j \):
  - Do prediction: \( P_{j-j+1} : f_j \rightarrow \tilde{f}_{j+1} \)
  - Compute \( f_{j+1} \):
    \[
    f_{j+1,2k+1} = d_{j,k} + \tilde{f}_{j+1,2k+1}
    \]
    \[
    f_{j+1,2k} = 2 d_{j,k} - f_{j+1,2k+1}
    \]
Figure 13. Cell-average scaling functions for constant (left side) and quadratic predictions (right side), at level $j = 5$.

Figure 14. Cell-average scaling functions at level $j = 5$, interacting with the left boundary, for quadratic predictions.

Functional context. In the present case of cell average discretization, we can also define multiresolution decompositions like (1-2). It is clear that

$$f_{j,k} = 2^j \int_{\Omega_{j,k}} f(x) dx = \langle f, \Phi_{j,k} \rangle,$$

where $\Phi_{j,k} = 2^j \chi_{\Omega_{j,k}}$. Here $\chi_C$ is the characteristic function of the set $C$, whose values are 1 for $x \in C$ and 0 otherwise. The scaling relation holds

$$\Phi_{j,k}^* = \frac{1}{2}[\Phi_{j+1,2k} + \Phi_{j+1,2k+1}].$$

It is also clear that the dual wavelets $\Psi_{j,k}^*$ such that

$$d_{j,k} = f_{j+1,2k+1} - \tilde{f}_{j+1,2k+1} = \langle f, \Psi_{j,k}^* \rangle$$

are given by

$$\Psi_{j,k}^* = 2^{j+1} \chi_{\Omega_{j+1,2k+1}} - 2^j \sum_{m} h_{m}^k \chi_{C_{j,m}}$$

$$= \Phi_{j+1,2k+1} - \sum_{m} h_{m}^k \Phi_{j,m}^*.$$
where \( h_m^k \) are the prediction coefficients, such that

\[
\tilde{f}_{j+1, 2k+1} = \sum_m h_m^k f_{j,m}.
\]

The definition of the primal basic functions \( \Phi_{j,k} \) and \( \Psi_{j,k} \) relies on the convergence of the prediction operator. A cell-average prediction is convergent if for any starting sequence of \( s_j = (s_{j,k}) \), there exists an integrable function \( s_j(x) \) such that its cell averages at levels \( \ell \geq j \) coincide with the output of the iterative cell-average subdivision \( s_t = P_{\ell - 1 - \ell s_{\ell - 1}} \). If this property holds, average-interpolating scaling functions \( \Phi_{j,k}(x) \) may be defined by cell-average subdivision of delta sequences \( s_{j,m} = \delta(k - m) \).

In the functional spaces

\[
V_j = \text{span} \{ \Phi_{j,k}(x), k = 0, \ldots, 2^j - 1 \},
\]

the expansion

\[
f_j(x) = \sum_{k=0}^{2^j-1} f_{j,k} \Phi_{j,k}(x) \in V_j
\]
corresponds to the cell-average reconstruction of an integrable function \( f \), in the sense that \( D_j f_j(x) = f_j \), i.e., the cell averages of \( f \) and \( f_j \) coincide at level \( j \).

Using the same arguments of the interpolating setting, we obtain scaling relations

\[
\Phi_{j,k}(x) = \sum_m c_k(m) \Phi_{j+1, m}(x),
\]

where \( c_k(m) \) are the cell averages of \( \Phi_{j,k} \) at level \( j + 1 \), which are obtained from the prediction of the delta-sequence \( \delta(k - m) \).

Figure 13 displays the scaling functions \( \Phi_{j,k}(x) \) for constant and quadratic cell-average interpolations, at level \( j = 5 \), at the center of the interval. The scaling functions interacting with the left boundary, for quadratic prediction, are shown in Figure 14.

Having that \( V_j \subset V_{j+1} \), now we examine complementary spaces \( W_j \) such that \( V_{j+1} = V_j + W_j \), and wavelets \( \Psi_{j,k} \) that span these differences. As such, the multiresolution expansion of \( \Psi_{j,k} \) is expected to correspond to zero cell averages at level \( j \), and wavelet coefficients \( d_{j,m} = \delta(k - m) \). Therefore, it should be given by the expression

\[
\Psi_{j,k}(x) = \Phi_{j+1,2k+1}(x) - \Phi_{j+1,2k}(x).
\]

For the particular case of prediction by constant approximation, \( \Phi_{j,k} = \chi_{\Omega_{j,k}} \) and the spaces \( V_j \) are formed by piecewise constant functions, generate a multiresolution analysis of \( L^2(\Omega) \) such that \( V_j \subset V_{j+1}, \cup_{j \geq 0} V_j \) is dense in \( L^2(\Omega) \) and \( \{ \Phi_{j,k}, k = 0, \ldots, 2^j - 1 \} \) is an orthogonal basis for \( V_j \).

The expansion

\[
f_j(x) = \sum_k f_{j,k} \Phi_{j,k}(x)
\]
corresponds to the orthogonal projection of \( f \) on \( V_j \). The wavelet spaces \( W_j = \text{span} \{ \Psi_{j,k}, k = 0, \ldots, 2^j - 1 \} \) are the orthogonal complement of \( V_j \) in \( V_{j+1} \), and \( \Psi_{j,k}(x) \) are the known Haar wavelets.

For general cell-average predictions by polynomials of higher degree \( M = 2L \), the spaces \( V_j \) also generate a multiresolution analysis of \( L^1(\Omega) \), such that \( V_j \subset V_{j+1}, \cup_{j \geq 0} V_j \) is dense in \( L^1[0,1] \). The sequence of approximations

\[
f_j(x) = \sum_k f_{j,k} \Phi_{j,k}(x)
\]
corresponds to the biorthogonal projections of \( f \) on \( V_j \) and
For a comprehensive reference on the convergence properties of MR approximations, we refer to [11].

Local regularity wavelet indicator

In the cell-average context, the wavelet coefficients can also be used as local regularity indicators. Assume that the function $f(x)$ has $C^s$-smoothness, for some $s \leq M$, within the interval $I_{j,k}$ supporting $\Psi_{j,k}^*$ (i.e. $I_{j,k}$ contains the stencil cells used in the prediction of $\tilde{f}_{j+1,2k+1}$). Using classical results of local polynomial approximation by polynomials, and the polynomial cancellation property, the wavelet coefficient can be estimated by

$$
|d_{j,k}| \leq \inf_{q \in \Pi_{M-1}} ||f - q||_{L^\infty(I_{j,k})} ||\Psi_{j,k}^*||_{L^1} \\
\leq C 2^{-sj} ||f||_{C^s(I_{j,k})}
$$

where the facts $||\Psi_{j,k}^*|| = 1$, and that the size of $I_{j,k}$ is $O(2^{-j})$ have been used. Therefore, for high accurate predictions, the fast decay of of wavelet coefficients is ensured in smooth regions.

Data compression. Given the cell averages $f_J$ on the finest level $J$, after the iteration of the MR transform, they can be represented as

$$f^*_{JMR} = (f_0, d_0, \ldots, d_{J-1}).$$

Data compression is obtained after a truncation step

$$d^*_{j,k} = \begin{cases} 
0 & \text{if } |d_{j,k}| \leq \epsilon_j \\
\text{else} 
\end{cases}$$

Figure 15 shows the plot of functions $f(x)$ on the top side, having different regularity pattern. The position of their significant wavelet coefficients, on the bottom side, illustrates the property of local regularity characterization.
Stability. We pose again the same stability question: after applying the inverse MR transform to the compressed data to $f^\text{MR, } \epsilon$, how do the reconstructed values $f_J$ compare to the original ones? Could the thresholding perturbation be amplified by iterative application of the cell-average prediction?

For convergent subdivision schemes we may use the expansion

$$f_J(x) - f'_J(x) = \sum_{j=0}^{J-1} \sum_k (d_{j,k} - d'_{j,k}) \Psi_{j,k}(x)$$

Since the support of $\Psi_{j,k}$ is uniformly localized around $C_{j,k}$, such that $||\Psi_{j,k}||_{L^1} \leq C 2^{-j}$, the $L^1$ thresholding error at level $j$ is

$$\left| \int_0^1 \sum_{k=0}^{2^j-1} (d_{j,k} - d'_{j,k}) \Psi_{j,k}(x) \, dx \right| \leq \epsilon_j \sum_{k=0}^{2^j-1} ||\Psi_{j,k}||_{L^1} \leq C \epsilon_j$$

Consequently, the $L^1$-stability of thresholding error overall the scale levels is estimated by

$$||f_J - f'_J||_{L^1} \leq C \sum_{m=0}^{J-1} \epsilon_j.$$ 

If $\epsilon_{J-1} = \epsilon$, and $\epsilon_j = \frac{1}{2} \epsilon_{j+1} = \epsilon 2^{j-J}$, we conclude that the thresholding error is uniformly bounded by $C \epsilon$, up to a change in the constant $C$.

Cartesian grids in higher dimensions.

In higher dimensions, multiresolution schemes for Cartesian grids can be obtained by tensor products of 1D schemes. For instance, in 2D geometry, consider a hierarchy of Cartesian meshes

$$G_j = \{ \Omega_{j,\gamma} \}, \quad \gamma = (k, m) \in S_j, \quad \Omega = \bigcup_{\gamma \in S_j} \Omega_{j,\gamma}, \quad ||\Omega_{j,\gamma}|| \sim 2^{-2j},$$

where $\Omega_{j,\gamma} \in G_j$ is the union of four cells in $G_{j+1}$ (see Figure 16).
For the prediction to be exact for quadratic polynomials at the children cells of $\Omega_j$, $(k,m)$, the neighbors $\Omega_{j,(k-1,m)}$, $\Omega_{j,(k+1,m)}$, $\Omega_{j,(k,m-1)}$ and $\Omega_{j,(k,m+1)}$ are considered, the formula reads

$$
\tilde{f}_{j+1}(2k+1,2m) = f_{j+1}(2k,2m) - \tilde{f}_{j+1}(2k+1,2m) = f_{j+1}(2k+1,2m) - \tilde{f}_{j+1}(2k,2m) = f_{j+1}(2k+1,2m) - \tilde{f}_{j+1}(2k,2m)
$$

Now, three wavelet coefficients have to be considered and the two-level transformations are $f_{j+1} \leftrightarrow \{f_j, d_j^{(1)}, d_j^{(2)}, d_j^{(3)}\}$, where

$$
d_j^{(1)}(k,m) = f_{j+1}(2k,2m) - \tilde{f}_{j+1}(2k,2m)
$$

$$
d_j^{(2)}(k,m) = f_{j+1}(2k+1,2m) - \tilde{f}_{j+1}(2k+1,2m)
$$

$$
d_j^{(3)}(k,m) = f_{j+1}(2k+1,2m+1) - \tilde{f}_{j+1}(2k+1,2m+1)
$$

General geometries. Harten’s approach can be extended to discretizations based on general geometries, as indicated in [1]. For instance, consider a hierarchy of triangular meshes

$$
C_j = \{T_j^\gamma\}, \Omega = \bigcup_{\gamma \in \mathbb{S}} T_j^\gamma, \quad |T_j^\gamma| \sim 2^{-2j},
$$

such that $T_j^\gamma$ is the union of four child triangles $C_j^\gamma = \{T_{\mu}^{j+1} \mid \mu = (\gamma, i), i = 0, 1, 2, 3\}$ at level $j + 1$
Given the cell averages \( f_j = (f_{j,\gamma}) \), where

\[
  f_{j,\gamma} = \frac{1}{|T_j|} \int_{T_j} u(x) dx
\]

the restriction operator \( P_{j+1 \rightarrow j} : f_{j+1} \rightarrow f_j \) is straightforward

\[
  f_{j,\gamma} = \frac{1}{|T_j|} \sum_{\mu \in C_j^{(\gamma)}} |T_{j+1}^{(\mu)}| f_{j+1,\mu}
\]

For the definition of the prediction operator \( P_{j \rightarrow j+1} : f_j \rightarrow \tilde{f}_{j+1} \), the easiest way is by piecewise constant approximation

\[
  \tilde{f}_{j+1,\mu} = f_{j,\gamma}, \quad \mu \in C_j^{(\gamma)},
\]

which gives the three Haar wavelet coefficients

\[
  d_{j,\gamma}^{(i)} = f_{j+1,\mu} - \tilde{f}_{j+1,\mu}, \quad \mu = (\gamma, i) \in C_j^{(\gamma)}, i = 1, 2, 3
\]

as indicated in Figure 18

For higher accurate predictions, the suggestion in [13] is

\[
  \tilde{f}_{j+1,\mu} = \begin{cases} 
  f_{j,\gamma} + \frac{1}{6}[f_{j,\gamma}(2) - f_{j,\gamma}(1)] & \mu = (\gamma, 0) \\
  f_{j,\gamma} + \frac{1}{6}[f_{j,\gamma}(1) - f_{j,\gamma}(2)] & \mu = (\gamma, 1) \\
  f_{j,\gamma} + \frac{1}{6}[f_{j,\gamma}(3) - f_{j,\gamma}(1)] & \mu = (\gamma, 2) \\
  f_{j,\gamma} + \frac{1}{6}[f_{j,\gamma}(1) - f_{j,\gamma}(3)] & \mu = (\gamma, 3)
\end{cases}
\]

which holds for equilateral triangular partitions, where the triangles \( T_{\gamma(i)}^{(j)} \), \( i = 1, 2, 3 \) denote the three neighbors of \( T_j^{(j)} \). This prediction results to be convergent, and exact for first order polynomials.
2.3. The lifting methodology

The basic idea in the lifting methodology is to start from a simple framework, Haar-like wavelets for instance, and use the lifting scheme (or dual lifting) [65] to obtain a new multiresolution analysis with better performance. The block diagram of Figure 19 illustrates the lifting procedure in terms of two operations: prediction $P$ and update $U$. Suppose that the input data $u_{j+1}$ is subjected to a given MR scheme to produce coarse values $u_j$ and details $d_j$. The prediction acts on the coarse data to modify the detail coefficients

$$d_j^{new} = d_j - Pu_j.$$ 

Then, the operation $U$ is applied to the details in order to modify the coarse data

$$u_j^{new} = u_j + Ud_j^{new}.$$ 

Once the forward lifting transform is defined, its inverse can be easily constructed by reversing the operations. Given the input $\{u_j^{new}, d_j^{new}\}$, first recover $\{u_j, d_j\}$

$$u_j = u_j^{new} - Ud_j^{new}$$ $$d_j = d_j^{new} + Pu_j$$

and then apply the original inverse MR transform to get $u_{j+1}$.

Lifting provides an efficient methodology for custom-design construction of wavelets on arbitrary topological domains, e.g. for wavelets on the sphere [62]. To give a basic idea of this methodology, we describe some examples to show how MR transforms for point values and cell averages can be formulated using lifting tools, starting from very simple schemes.

2.3.1. Lifting the interpolating wavelets

Multiresolution for point values can be formulated in the lifting context. Consider the simplest MR scheme (lazy MR) where the analysis step just split the data in two sets of even and odd indexes

$$\begin{align*}
(even_j, odd_j) & := \text{Split}(u_{j+1}) \\
\begin{array}{c}
u_{j,k} = u_{j+1,2k} \\
d_{j,k} = u_{j+1,2k+1}
\end{array}
\end{align*}$$

and the synthesis corresponds to merging operation

$$\begin{align*}
u_{j+1} & := \text{Merge}(even_j, odd_j) \\
u_{j+1,2k} = u_{j,k} & \quad u_{j+1,2k+1} = d_{j,k}
\end{align*}$$

Figure 19. Lifting scheme
If an interpolating prediction $\mathcal{P} = P_{j \rightarrow j+1}$ is introduced, the lazy wavelet coefficients are modified
\[
d_{j,k}^\text{new} = \text{odd}_{j,k} = \mathcal{P}(\text{even})_{j,k}
\]
\[
d_{j,k}^\text{new} = u_{j+1,2k+1} - \tilde{u}_{j+1,2k+1}
\]
which coincides with the result given by Harten’s approach. Figure 20 illustrates this new interpretation of the interpolating MR inter-level operations in the lifting context.

In interpolating MR contexts, one disadvantage is that the average of the information $f_j$ is not conserved within the scale levels. Using the lifting methodology, an update operation may be defined to modify the scaling coefficients in order to preserve averages
\[
\frac{1}{2^j} \sum_k u_{j,k}^\text{new} = \frac{1}{2^{j+1}} \sum_k u_{j+1,k}
\]

For instance, given the interpolating MR scheme where
\[
d_{j,k} = u_{j+1,2k+1} - \frac{u_{j,k} + u_{j,k+1}}{2},
\]
Thus, we obtain the desired property
\[
\sum_k u_{j,k}^{\text{new}} = \sum_k u_{j+1,2k+1} - \sum_k u_{j+1,2k},
\]
which means that the multiresolution spaces do not modify, \(V_j^{\text{new}} = V_j\). Furthermore, since the wavelet coefficients are not modified, the dual wavelets \(\Psi_j^{\ast}(x)\), such that \(d_{j,k} = \langle u, \Psi_j^{\ast}(x) \rangle\), are also preserved. However, the modification of the scaling coefficients implies that new dual scaling functions \(\Phi_j^{\ast,\text{new}}(x)\) such that
\[
\langle u_{j,k}^{\text{new}}, \Phi_j^{\ast,\text{new}}(x) \rangle = \langle u, \Phi_j^{\ast}(x) \rangle + \frac{1}{4}[\Psi_{j,k-1}(x) + \Psi_{j,k}(x)].
\]

The new primal wavelets \(\Psi_{j,k}^{\text{new}}(x)\) appear. This means that the effect of updating is the creation of a new factorization of the scaling spaces
\[
V_{j+1} = V_j + W_j^{\text{new}}.
\]

For instance, in the example presented in 21, it is clear that
\[
\Phi_{j,k}^{\ast,\text{new}}(x) = \Phi_j^{\ast}(x) + \frac{1}{4}[\Psi_{j,k-1}(x) + \Psi_{j,k}(x)].
\]

The new primal wavelets \(\Psi_{j,k}^{\text{new}}(x)\) are the functions in \(V_{j+1}\) whose point values at \(X_{j+1}\) are obtained by the updated synthesis of zero values in \(X_j\) and wavelet coefficients \(d_{j,m} = \delta(k-m)\), that is \(\cdots, 0, -1/4, 1, -1/4, 0, \cdots\). Consequently, they are the continuous piecewise linear functions as indicated in Figure 22.

**Figure 22.** Updated primal wavelets \(\Psi_{j,k}^{\text{new}}(x)\)

### 2.3.2. Dual lifting the Haar wavelets

Suppose that for the applications of our interest, starting with cell average data, we are not interested in modifying the coarse values, since we want to keep the cell averages at all levels. Therefore, we skip the updating step and just modify the wavelet coefficients. If the original scheme is of Haar type, which is only consistent for constant functions, the purpose is to obtain a new MR scheme with higher order of consistency. This kind of procedure is known as dual lifting transform.

As described in Section 2.2, Haar wavelets have one vanish moment, i.e., the Haar wavelet coefficients vanish only for constant polynomials. The dual lifting scheme can be used to construct new MR settings connected to dual scaling functions \(\{\Phi_{j,k}^{\text{new}}, \Phi_{j,k}^{\ast,\text{new}}\}\) such that the corresponding wavelet coefficients vanish for higher degree polynomials. The basic idea is to keep the dual scaling functions \(\Phi_{j,k}^{\ast,\text{new}} = \Phi_j^{\ast}\), but modify the other basic
functions accordingly to the new requirements. This means that in the forward MR transform, the scaling coefficients \( u_{j,k} = < u, \phi^*_{j,k} > \) remain the same, but the Haar wavelet coefficients \( d_{j,k} \) are updated using the scaling coefficients \( u_{j,m} \) corresponding to neighboring cells \( \Omega_{j,m}, \; m \in S(k) \)

\[
\begin{align*}
\tilde{d}_{j,k}^{\text{new}} &= d_{j,k}^{\text{Haar}} - \sum_{m \in S(k)} c_{m}^k u_{j,m}
\end{align*}
\]

which leads to the new dual wavelets

\[
\psi_{j,k}^{\text{new}} = \psi_{j,k}^{*} - \sum_{m \in S(k)} c_{m}^k \phi_{j,m}^{*}
\]

The coefficients \( c_{m}^k \) are determined in order to kill the desired polynomials \( q \)

\[
\sum_{m \in S(k)} c_{m}^k < q, \phi_{j,m}^{*} > = < q, \psi_{j,k}^{*} >.
\]

Since the scaling coefficients are not modified, the dual scaling functions are preserved, such that \( u_{j,k}^{\text{new}} = < u, \phi_{j,k}^{*} > \). The modification of the wavelet coefficients implies new dual wavelets \( \psi_{j,k}^{\text{new}}(x) \) (dual lifting) such that

\[
\tilde{d}_{j,k}^{\text{new}} = < u, \psi_{j,k}^{\text{new}} >,
\]

new primal scaling functions \( \Phi_{j,k}^{\text{new}} \), and new primal wavelets \( \Psi_{j,k}^{\text{new}} \). Consequently, the inclusion of the prediction leads to new MR spaces

\[
V_{j+1}^{\text{new}} = V_{j}^{\text{new}} + W_{j}^{\text{new}}
\]

For example, we consider the dual lifting of the Haar wavelets in order to cancel quadratic polynomials

\[
\begin{align*}
\tilde{d}_{j,k}^{\text{new}} &= d_{j,k}^{\text{Haar}} - \frac{1}{8}[u_{j,k+1} - u_{j,k-1}] \\
&= u_{j+1,2k+1} - [u_{j,k} + \frac{1}{8}(u_{j,k+1} - u_{j,k-1})]
\end{align*}
\]

which coincides with the third order scheme derived in Section 2.2 using Harten’s approach. We refer to [19], where it is shown that the classical wavelet transforms, defined in terms of filter banks, can be decomposed into a finite sequence of lifting steps.

3. Sparse Point Representation (SPR) method

The SPR method, introduced by [38,39], is an adaptive finite difference strategy that combines the simplicity and accuracy of traditional finite difference schemes with the ability of wavelet coefficients in the characterization of local regularity of functions. The idea is to represent the functions by the point values corresponding to their significant wavelet coefficients. Typically, few points are found in each time step, the grid being coarse in smooth regions, and refined close to irregularities. At each point, spatial derivatives are discretized by uniform finite differences, using step size proportional to the point local scale. Eventually, stencils not present in the grid are approximated from coarser scales by using an interpolating subdivision scheme.

Other wavelet adaptive methods have been proposed with many similarities to the SPR method. For instance, the filter bank method by [69] and the second generation wavelet collocation method by [67] may be considered as generalizations of the SPR method. As it is well discussed in [69], one of the main attractive aspects of such class of adaptive wavelet solvers is that they can be separated into two basic parts: the representation part and the operator part. The operator part is performed by finite differences on uniform grids which may be chosen by
considering stability and consistency criteria. Therefore, they can be beneficiary of the considerable advances achieved in this area. The representation part is formulated in the context of wavelet data compression by means of a simple thresholding operation. This is a more recent topic, but a rigorous study of the effectiveness of such kind of nonlinear approximation has already being established. We refer to [11] for a substantial survey on this subject.

This kind of separation of the solver into independent parts makes it general: it is simple to change the differential equation, the order of the finite difference method, the boundary conditions, the wavelet transform etc. Consequently, it fits well into the object oriented programming philosophy. However, there is one major weaknesses of the SPR method. Namely, the overhead involved in accessing or interpolating neighboring stencils to compute finite differences at scattered grid points. One possible remedy is to decrease the overhead by losing some grid sparsity. The suggestion in [38] is to use an adaptive block representation (ABR). The computational domain is formed by non-overlapping blocks. Each block is a uniform grid, but the step size may change from one block to another. In the automatic construction of such block-grids, the wavelet coefficients are also used as the main tool to decide whether a block needs to be refined or may be coarsened. In order to avoid undesirable search communications between blocks during the computation of derivatives, layers of auxiliary neighboring points are added around each block.

3.1. Adaptive block representation (ABR)

In this section we establish a general framework for the application of the ABR method in 2D. The algorithms and data structure are formulated by using abstract concepts borrowed from quaternary trees. With this procedure, we expect to improve the understanding of the method and help the process of its computational and data structure are formulated by using abstract concepts borrowed from quaternary trees. With this point of view, new blocks at level $j+1$ are obtained, firstly by dyadic refinement, and then by splitting into four parts

$$B_{\mu}^j \rightarrow S(B_{\mu}^j) = \{B_{\mu_0}^{j+1}, B_{\mu_1}^{j+1}, B_{\mu_2}^{j+1}, B_{\mu_3}^{j+1}\}.$$ 

If $\mu$ is the origin of the starting block, then the origins of the new blocks are $\mu_{00} = \mu$, $\mu_{01} = \mu + (0, \frac{N_x}{2} h_y^j)$, $\mu_{10} = \mu + (\frac{N_x}{2} h_x^j, 0)$, and $\mu_{11} = \mu + (\frac{N_x}{2} h_x^j, \frac{N_y}{2} h_y^j)$. Using the quad-tree terminology, the elements of the set $S(B_{\mu}^j)$ are called the children of the node $B_{\mu}^j$. $B_{\mu_0}^{j+1}$ is the mother of $B_{\mu_1}^{j+1}$, $B_{\mu_1}^{j+1}$ is son of $B_{\mu}^j$, and $B_{\mu_3}^{j+1}$ is brother of $B_{\mu_2}^{j+1}$. One generation of a quad-tree block structure is illustrated in Figure 23.
By defining \( B^0_\mu = X^0 \) as the root of the tree, and performing \( J \) generations, a complete quad–tree having \( J + 1 \) levels is obtained. At each level \( 0 \leq j \leq J \) there exist \( 2^{2j} \) blocks (nodes). Let us denote by \( \mathcal{I}^j \) the set of block origin points at level \( j \).

Incomplete quad–trees occur when at intermediary levels some nodes have no children. This leads to the concept of tree leaves: leaves are nodes that do not have children.

In a complete block quad–tree, the leaves correspond to the blocks \( B^J_\mu \) at the last level. The union of these blocks constitutes the uniform grid at the finest scale

\[
X^J = \bigcup_{\mu \in \mathcal{I}^J} B^J_\mu.
\]

In an incomplete block quad–tree, the leaves at intermediary levels \( 0 < j < J \) correspond to blocks where the refinement process has been interrupted. Let \( \Lambda^j \subset \mathcal{I}^j \) be the set of origin points associated to the leaf-blocks at level \( j \). The union of such blocks forms a block-structured grid

\[
\mathcal{M} = \bigcup_{j=1}^{J} \bigcup_{\mu \in \Lambda^j} B^j_\mu.
\]

The diagram in Figure 24 illustrates a 4-level incomplete tree and its corresponding grid.

**Adaptive Construction of Block–Grids**

For a given block-structured grid \( \mathcal{M} \), we shall denote by \( f^\mathcal{M} \) the collection of point values of a given function \( f(x, y) \) represented at \( \mathcal{M} \). According to the tree structure of \( \mathcal{M} \), \( f^\mathcal{M} \) can be organized as a vector whose components are \( N_x \times N_y \) matrices containing the data corresponding to the point values at the leaf-blocks \( B^j_\mu \).

The purpose of the ABR technique is to obtain representations \( \{ \mathcal{M}, f^\mathcal{M} \} \) as sparse as possible. This means that a small total number of blocks is found in an incomplete quad-tree \( \mathcal{M} \), the leaves in lower levels (big scales) corresponding to smooth regions and those at higher levels (small scales) corresponding to irregularity regions. In the construction of a such adaptive representation, the main tool is a local regularity indicator \( i(B) \) to decide, at each generation, whether a given block should be a leaf, \( i(B) = 0 \), or not, \( i(B) = 1 \). Next, we shall describe
how wavelet coefficients in an interpolatory multi-resolution analysis can be used in the definition of \(i(B)\). In the wavelet literature, there are several examples of multi-resolution analyses that may be used as well. In fact, some of them offer convenient aspects, like shorter filters for the same order of polynomial cancellation (zero moments for the dual wavelets), as described in [69], or additional zero moments for the primal wavelets, like in the modified lifting scheme adopted by [67].

**Wavelet Indicators**

Given a generic block \(B_{j}^{\mu}\) at level \(j\), define the set \(\tilde{B}_{j}^{\mu}\) as the completion of \(B_{j}^{\mu}\) by the inclusion of extra points in the right and upper lateral lines. That is,

\[
\tilde{B}_{j}^{\mu} = \{ \gamma = \mu + (kh_{x}^{j}, \ell h_{y}^{j}), 0 \leq k \leq N_{x}, 0 \leq \ell \leq N_{y} \}.
\]

Consider the rectangular grid \(\mathcal{R}^{j} = \tilde{B}_{j}^{\mu}\) and its dyadic refinement \(\mathcal{R}^{j+1}\) given by

\[
\mathcal{R}^{j+1} = \{ \gamma = \mu + (kh_{x}^{j+1}, \ell h_{y}^{j+1}), 0 \leq k \leq 2N_{x}, 0 \leq \ell \leq 2N_{y} \}.
\]

Note that \(\mathcal{R}^{j+1}\) can also be expressed as

\[
\mathcal{R}^{j+1} = \bigcup_{n=0}^{3} \tilde{B}_{j+1}^{\mu_{n}},
\]

where \(\tilde{B}_{j+1}^{\mu_{n}}\) are the completion of the children of \(B_{j}^{\mu}\).

Let \(f^{j}\) be the matrix containing the values of a given function \(f(x, y)\) at the grid points in \(\mathcal{R}^{j}\)

\[
f_{k,\ell}^{j} = f(\mu + (kh_{x}^{j}, \ell h_{y}^{j})).
\]

In wavelet analysis, transformation algorithms relating \(f^{j+1}\), \(f^{j}\) and the wavelet coefficients \(d^{j}\), containing the difference of information between two consecutive levels, play a crucial role. For discretization by means of points values, the wavelet coefficients are usually defined in terms of interpolation error [11]. Values \(f_{2k,2\ell+1}^{j+1}\), \(f_{2k+1,2\ell}^{j+1}\), and \(f_{2k,2\ell+1}^{j+1}\), at the new midpoints \(\gamma_{k,\ell}^{(a),j} \in \mathcal{R}^{j+1} \setminus \mathcal{R}^{j}\),

\[
\gamma_{k,\ell}^{(1),j} = \mu + (2kh_{x}^{j+1}, (2\ell + 1)h_{y}^{j+1}),
\]

\[
\gamma_{k,\ell}^{(2),j} = \mu + ((2k + 1)h_{x}^{j+1}, 2\ell h_{y}^{j+1}),
\]

\[
\gamma_{k,\ell}^{(3),j} = \mu + ((2k + 1)h_{x}^{j+1}, (2\ell + 1)h_{y}^{j+1}),
\]

are computed by polynomial Lagrange interpolation from the values \(f^{j}\) at the coarser grid. Wavelet coefficients are then defined as the differences between the known function values \(f_{2k,2\ell+1}^{j+1}\), \(f_{2k+1,2\ell}^{j+1}\), and \(f_{2k,2\ell+1}^{j+1}\) and the
values predicted by the interpolation procedure. Precisely,
\[
\begin{align*}
\tilde{f}_{k,\ell}^{j} &= f_{2k,2\ell}^{j+1} \\
d_{k,\ell}^{(1),j} &= f_{2k,2\ell+1}^{j+1} - \tilde{f}_{2k,2\ell+1}^{j+1}, \\
d_{k,\ell}^{(2),j} &= f_{2k+1,2\ell}^{j+1} - \tilde{f}_{2k+1,2\ell}^{j+1}, \\
d_{k,\ell}^{(3),j} &= f_{2k,2\ell+1}^{j+1} - \tilde{f}_{2k,2\ell+1}^{j+1}.
\end{align*}
\]

In case of rectangular 2D grids, interpolation can be expressed in terms of 1D algorithms. For instance, \(\tilde{f}_{2k+1,2\ell}^{j+1}\) is obtained from \(f_{k,\ell}^{j}\) by one-dimensional interpolation along the \(l\)-direction, \(\tilde{f}_{2k+1,2\ell}^{j+1}\) is obtained from \(f_{k,\ell}^{j}\) by one-dimensional interpolation along the \(k\)-direction. Finally, \(\tilde{f}_{2k,2\ell+1}^{j+1}\) is obtained from \(\tilde{f}_{2k+1,2\ell}^{j+1}\) by applying the one-dimensional interpolatory scheme in the \(l\)-direction (or, equivalently, by one-dimensional interpolation of the \(\tilde{f}_{2k,2\ell+1}^{j+1}\) values in the \(k\)-direction). The general 1D interpolation formula reads
\[
\tilde{f}_{2\ell+1}^{j+1} = \sum_{q} p_{\ell,q} f_{\ell+q}^{j},
\]
in which \(p_{\ell,q}\) are the Lagrange interpolation weights. In the interior of the grid, central interpolation is used for which the weights do not depend on the location, i.e., \(p_{k,q} = p_{q}\). One-sided interpolation is required close to the boundaries, leading to weights that do depend on the location \(k\) [38].

As interpolation errors, wavelet coefficients are good indicators of local smoothness. This fact leads to the definition of a set index \(i_{e}(\mathcal{B})\) which is based on the significance of the wavelet coefficients associated to the block \(\mathcal{B}\), as compared to a certain given threshold \(\epsilon\). By definition, \(i_{e}(\mathcal{B}_{0}) = 1\). Then, at level \(j > 0\), for each block \(\mathcal{B}_{\mu}^{j}\) having set index equal to one, the children set indexes \(i_{e}(\mathcal{B}_{\mu+1}^{j+1})\) are computed according to the following strategy. First we perform the one-level wavelet transform, as described before, and consider \(\mathcal{D}(\mathcal{B}_{\mu+1}^{j+1})\) as the set of those wavelet coefficients \(d_{k,\ell}^{(\alpha),j}\) which are associated to points \(\gamma_{k,\ell}^{(\alpha),j}\) in \(\mathcal{B}_{\mu+1}^{j+1}\). If all wavelet coefficients in \(\mathcal{D}(\mathcal{B}_{\mu+1}^{j+1})\) are not significant, this means that the function is smooth in this region. Consequently, the block does not need to be refined, and it will be a leaf-tree. On the other hand, if there exists at least one significant wavelet coefficient in the block, this means that the function is not represented there with the prescribed accuracy, and refinement is needed. Therefore, according to these principles, the set index \(i_{e}\) is defined by
\[
i_{e}(\mathcal{B}_{\mu+1}^{j+1}) = \begin{cases} 
0 & \text{if } |d_{k,\ell}^{(\alpha),j}| < \epsilon \forall d_{k,\ell}^{(\alpha),j} \in \mathcal{D}(\mathcal{B}_{\mu+1}^{j+1}) \\
1 & \text{otherwise}.
\end{cases}
\]

Following this strategy, the adaptive grid construction ends when, at a certain level, all the analyzed blocks are leaves.

**Numerical Examples**

The first example is for the spike function
\[
f(x, y) = 3 \exp^{-2500.0((x-0.3)^2+(y-0.3)^2)} + \text{sen}(2\pi x) + \text{sen}(2\pi y),
\]
with a singularity point at \((x, y) = (0.3, 0.3)\), as illustrated in Figure 25. For this function, the ABR grids \(\mathcal{M}_{\epsilon}\) are represented in Figure 26. The wavelet coefficients are for cubic polynomial interpolation \((M = 4)\), the blocks have \(N_{x} = N_{y} = 32\) and \(\epsilon = 10^{-3}, 10^{-4}\) and \(10^{-5}\). As \(\epsilon\) becomes smaller, more refined blocks are tended to concentrate near the singularity region. As indicated in Table 1, the number of blocks grows as \(\epsilon\).
becomes smaller. However, if the number of points in the ABR grid $\mathcal{M}_\epsilon$ is compared to number of points in the uniformed mesh $\mathcal{X}^\ell$, in which $\ell$ is the finest scale level present in $\mathcal{M}_\ell$, then the ratio $\#(\mathcal{M}_\epsilon)/\#(\mathcal{X}^\ell)$ reduces very quickly as $\epsilon$ decreases.

$$f(x, y) = 1 - \tanh(25x + 5(y - 1)),$$
that exhibit abrupt changes close to the line $25x + 5(y - 1) = 0$, as displayed in Figure 27. Therefore, the most refined blocks in the ABR grids $\mathcal{M}_\epsilon$ are expected to be placed around the oblique-front region, as $\epsilon$ decreases. This behavior is illustrated in Figure 28 for the same parameters of the previous example. An indication of the efficiency of the ABR method for the oblique-front function can be obtained from the data in Table 2. Their comparison with the corresponding results in Table 1 implies that, given an accuracy $\epsilon$, the spike ABR meshes $\mathcal{M}_\epsilon$ need higher resolution levels and more blocks than in the oblique-front case. Despite of that, the efficiency ratio is better for the spike function.

![Figure 27. Oblique-front function.](image)

**Table 2. Efficiency in ABR for the oblique-front**

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$\sharp$ blocks</th>
<th>$\sharp(M_\epsilon)$</th>
<th>$\sharp(X^2)$</th>
<th>$\sharp(M_\epsilon)/\sharp(X^2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-5}$</td>
<td>44</td>
<td>45056</td>
<td>262144</td>
<td>0.17</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>19</td>
<td>19456</td>
<td>65536</td>
<td>0.30</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>10</td>
<td>10240</td>
<td>16384</td>
<td>0.60</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>4</td>
<td>4096</td>
<td>4096</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Fixing $\epsilon = 10^{-3}$ and the block size $32 \times 32$, ABR grids $\mathcal{M}_{10^{-3}}$ for the oblique front are presented in Figure 29 (a) and (b), for different interpolation order. The number of blocks occurring with linear interpolation ($M = 2$) is more than twice than with cubic interpolation ($M = 4$). Fixing $M = 4$ and reducing the block size to $N_x = N_y = 16$, Figure 29 (c) indicates that the number of blocks in $\mathcal{M}_{10^{-3}}$ increases. However, despite of the smaller block size, the total number of points also increases. Similar behavior also occurs for different truncation parameter $\epsilon$.  

![Figure 28. ABR for the oblique-front function](image)
3.2. Operations on Block-Structured Grids

For the application of ABR method in the numerical solution of partial differential equations, there are some typical operations involved: grid refinement and coarsening, functional operations and differentiation.

Refining and Coarsening

The automatic adaptation of the grid structure during time evolution requires a simple procedure for refining or coarsening the grids. For block-structured grids, these operations are easily defined in the context of quad-trees by means of tree extension or reduction. In a general sense, if $\mathcal{A}$ and $\tilde{\mathcal{A}}$ are two quad-trees such as $\mathcal{A} \subset \tilde{\mathcal{A}}$, then $\tilde{\mathcal{A}}$ is an extension of $\mathcal{A}$, or $\mathcal{A}$ is a reduction of $\tilde{\mathcal{A}}$. Specifically, the extension and reduction operations of interest are described next.

- **Tree Extension (Grid Refinement):** $\mathcal{A} \xrightarrow{E} \tilde{\mathcal{A}}$

  An extended tree $\tilde{\mathcal{A}}$ may be obtained by adding a new block generation to all leaves of the tree $\mathcal{A}$. Figure 30 shows an example of a block structured grid $\mathcal{M}$ and its corresponding extension $\tilde{\mathcal{M}}$. Given the representation $(\mathcal{M}, f^\mathcal{M})$, the functional values $f^{\tilde{\mathcal{M}}}$ at the extended grid $\tilde{\mathcal{M}}$ may be not available. In such case, each missing value in $f^{\tilde{\mathcal{M}}}$ is obtained by the interpolatory refinement scheme.

- **Tree Reduction (Grid Coarsening):** $\tilde{\mathcal{A}} \xrightarrow{T} \mathcal{A}$

  As done for the grid construction, the reduction operator $T$ is characterized by a regularity indicator $i_x(B)$. All leaves in $\tilde{\mathcal{A}}$, whose three brothers are also leaves, are tested. If one of these leaves and their brothers have $i_x = 0$, these nodes are removed from the tree structure. Consequently, their mother becomes a new leaf-node. When this particular case happens, the leaf-structure changes. Therefore, the reduction process must be executed again.

Functional Operations

Operations between two functions represented in block-structured grids are straightforward point-wise evaluations if their grids coincide. Otherwise, it is necessary to extend both grids in order to get representations in
a common grid.

**Differentiation**

The idea is to use finite difference operators with uniform spacing in each block. For blocks at level \( j \), partial derivatives in the x-direction are discretized with spacing \( h_{xj} \), and in the y-direction with \( h_{yj} \). For points close to block boundaries, stencil information from neighboring blocks may be required. To avoid demanding search procedures, the process of block construction should consider the addition of needed extra rows and columns around the block boundaries. For instance, in the particular case of fourth order discretization of derivatives by a central scheme, two extra rows or columns are necessary at each side of the blocks. The importance in maintaining the block independence, by adding auxiliary extra rows or columns, is crucial for implementations of the method on parallel computers.

**Application to Evolution Problems**

The concepts in the ABR method are particularly suitable for adaptive solvers to evolution partial differential equations. For the applications of this lecture note, we shall consider equations of the form

\[
\frac{\partial U}{\partial t} = L U
\]

where \( L(U) \) is a differential operator acting on spatial variables. The solution \( U = U(x, y, t) \) is searched for \((x, y) \in [0, 1] \times [0, 1] \) and \( t > 0 \), augmented with initial and boundary conditions.

Suppose that at time \( t_n = n \Delta t \) a sparse block–grid representation \((M^n, U^n)\) for the approximate solution is given, in which \( U^n \) is formed by the numerical solution values at an adaptive mesh \( M^n \).

In the next time step, the representation \((M^{n+1}, U^{n+1})\) is obtained after the following procedure.

1. **Extension:** \((M^n, U^n) \xrightarrow{E} (M^{n+1}, U^{n+1})\)
   
   The grid \( M^n \) may be not suitable for the solution at the next time step, since regions of smoothness or irregularities of the solution may change from one step to the next one. Therefore, before doing time evolution, the representation of the solution should be extended to a grid \( M^{n+1} \), which is expected to contain \( M^{n+1} \). This refinement stage is very important. In the applications of this paper, the simple one level refinement described previously is adopted.

2. **Time evolution:** \((M^{n+1}, U^{n+1}) \xrightarrow{L_a} (M^{n+1}, \tilde{U}^{n+1})\)

   A discrete evolution operator \( L_a \) is applied. The action of \( L_a \) includes the discretization of the partial derivatives in \( L \) by uniform finite difference schemes, adapted to each block in \( M^{n+1} \), the discretization in time by some explicit ODE solver, and the enforcement of boundary conditions.

3. **Truncation:** \((M^{n+1}, \tilde{U}^{n+1}) \xrightarrow{T} (M^{n+1}, U^{n+1})\)

   Finally, a thresholding operation is applied, in order to unrefine those blocks in \( M^{n+1} \) that are unnecessary for an accurate representation of \( U^{n+1} \).

3.3. **Numerical Examples**

A solver that performs these tasks is implemented using a \( C++ \) object oriented paradigm programming. All the simulations are for 4-th order interpolation, finite differences and Runge-Kutta ODE solver. The time step is dynamically chosen according to the smallest scale \( h_{\text{min}}^{n} \) present in the extended adaptive mesh \( M^{n+} \), such that \( h_{\text{min}}^{n}/\Delta t^{n} = \lambda \).

**Oblique Front**

Let us consider the advection-diffusion equation

\[
\frac{\partial U}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial U}{\partial y} - \frac{\partial^2 U}{\partial x^2} - \frac{\partial^2 U}{\partial y^2} = F
\]

for \( t \geq 0, (x, y) \in [0, 1] \times [0, 1] \). Given the forcing term

\[
F(x, y, t) - \beta \text{sech}^2 \left( \frac{z}{\alpha} \right) - 2(\alpha^2 + \beta^2) \text{sech}^2 \left( \frac{z}{\alpha} \right) \tanh \left( \frac{z}{\alpha} \right).
\]
where \( z = 25(x - t) + 5(y - 1) \), and appropriate boundary conditions, the exact solution is

\[
U = 1 - \tanh(z).
\]

It describes a propagating steep front moving to the right. Figure 31 exhibits some typical features for the numerical solution and the corresponding extended adaptive block-grids. The parameters for this simulation are \( \epsilon = 10^{-2} \), \( N_x = N_y = 16 \) and \( \lambda = 10^{-3} \). The Figure 32 shows how the number of blocks in the adaptive mesh evolves during the simulation. The maximum number of blocks occurs when the front crosses the center of the region.

Figure 31. Oblique front
Figure 32. Number of blocks in the adaptive meshes for the evolution of the oblique front.

**Sharp Transition Layers**

The following results are for the Burgers’ equation

$$\frac{\partial U}{\partial t} + U \frac{\partial U}{\partial x} + U \frac{\partial U}{\partial y} - \mu \frac{\partial^2 U}{\partial x^2} - \mu \frac{\partial^2 U}{\partial y^2} = 0,$$

(8)

$$t \geq 0, \ (x,y) \in [0,1] \times [0,1] \text{ and } \mu = 10^{-2},$$

with periodic boundary conditions and initial data

$$U(x,y,0) = \text{sen}(2\pi x) \text{sen}(2\pi y).$$

Since negative and positive features move on opposite directions, this example shows sharp transition layers, as time evolves. In the present simulation, the parameters are $\epsilon = 10^{-5}$, $N_x = N_y = 32$ and $\lambda = 5 \times 10^{-2}$. Figure 33 exhibits some typical features of the numerical solution and the corresponding extended adaptive grids. Initially, the function is smooth and the mesh is totally uniform with 16 blocks. At $t \approx 0.9$, the application of the ABR method produces an irregular mesh with refined blocks close to the sharp transition regions. Figure 34 shows how the number of blocks in the adaptive meshes increases according to the steepness of the solution during the simulation.
Figure 33. Sharp transition layer formation. Solution and adaptive block meshes at different times.

Figure 34. Number of blocks in the adaptive meshes for the evolution of the sharp transition layer.
4. FV/MR Method

4.1. Finite volume discretization on uniform grids

We consider parabolic conservation laws in Cartesian geometry for \((x,t) \in \Omega \times [0, +\infty), \Omega \subset \mathbb{R}^d\), of the form

\[
\frac{\partial u}{\partial t} = \mathcal{D}(u, \nabla u),
\]

with initial condition \(u(x,0) = u_0(x)\), and appropriate boundary conditions. We shall consider with operators of type \(\mathcal{D}(u, \nabla u) = -\nabla \cdot F(u, \nabla u) + S(u)\) formed by divergence and source terms. For the applications of this lecture note, the flux function contains advective and diffusive parts of the form \(F(u, \nabla u) = f(u) - \nu \nabla u\), with constant diffusion coefficient \(\nu \geq 0\). The numerical model has two basic aspects: the spatial and the temporal discretizations.

For the spatial discretization, in the classical finite volume formulation, we consider the computational domain composed by the union of cells \(\{\Omega_i\}_{i=1}^N\). In each cell \(\Omega_i\), with boundary \(\partial \Omega_i\), external normal \(\eta_i\) and volume \(|\Omega_i|\). We integrate Equation 9 using of quadrature formula to get

\[
\frac{d\bar{u}_i}{dt} = \bar{D}_i(\bar{U}(t)),
\]

where \(\bar{U}(t) = (\bar{u}_i(t))\) contains the cell-averages of the numerical solution in the computational mesh at instant \(t\), such that

\[
\bar{u}_i(t) \approx \frac{1}{|\Omega_i|} \int_{\Omega_i} u(x, t) \, dx,
\]

and

\[
\bar{D}_i(\bar{U}(t)) \approx -\frac{1}{|\Omega_i|} \int_{\partial \Omega_i} F(u, \nabla u) \cdot \eta_i \, dx + \bar{S}_i(u).
\]

A variety of methods can be distinguished in the literature, mainly, by the way the numerical flux is defined for the approximation of the flux contribution. Typically, advective and diffusive terms are approximated differently. For the advective part, we use either second-order centered or upwind AUSM+ schemes [43] whereas, for the diffusive part, we always choose a second-order centered scheme. The source term is approximated by \(\bar{S}_i \approx S(u_i)\).

After space discretization by means of the finite volume scheme, the result is a system of ordinary equations

\[
\frac{d\bar{U}}{dt}(t) = \bar{D}(\bar{U}(t)),
\]

where we assume that the action of the operator \(\bar{D}\) also includes the enforcement of boundary conditions. For a sequence of discrete time values \(t_n = n\Delta t\), let \(\bar{U}^n\) be the approximation at \(t_n\). At the next time step, \(\bar{U}^{n+1}\) is obtained by the application of a discrete evolution operator \(\mathbf{E} = \mathbf{E}(\Delta t)\) such that

\[
\bar{U}^{n+1} = \mathbf{E}\bar{U}^n,
\]

where \(\mathbf{E}\) includes the application of the spatial differential operator \(\bar{D}\) and the discretization in time by means of some ODE solver. For the application of the present lecture note we use the second order Runge-Kutta (RK2) scheme

\[
\bar{U}^{n+1} = \bar{U}^n + \frac{\Delta t}{2} \left[ \bar{D}(\bar{U}^n) + \bar{D}(\bar{U}^n + \Delta t \bar{D}(\bar{U}^n)) \right].
\]

For illustration, it follows the advective flux \(f\) and the source term \(S\) for three different test-cases presented in the numerical, which yields simple models for viscous fluid dynamics and combustion problems:

1D convection-diffusion equation:: \(f(u) = cu\), and \(S(u) = 0\), where \((c > 0)\);
1D viscous Burgers equation: \(f(u) = \frac{u^2}{2}\) and \(S(u) = 0\);
reaction-diffusion equation: $f(u) = 0$ and $S(u) = \frac{\beta}{2}(1-u) \exp^{\frac{\beta(1-u)}{\alpha(1-u)+1}}$, where ($\alpha > 0$, $\beta > 0$).

To perform the discretization, we use a classical finite volume formulation in the standard conservative form. In the general case, let us consider the computational domain $\Omega$ in dimension $d$ with an arbitrary shape, and let us partition it into cells $(\Omega_i)_{i \in \Lambda}$, $\Lambda = \{1, \ldots, i_{\text{max}}\}$. We then denote $\bar{q}_i(t)$ the cell-average value of a given quantity $q$ on $\Omega_i$ at instant $t$,

$$\bar{q}_i(t) = \frac{1}{|\Omega_i|} \int_{\Omega_i} q(x,t) \, dx$$

where $|\Omega_i| = \int_{\Omega_i} \, dx$ is the volume of the cell. Integrating on $\Omega_i$ yields

$$\int_{\Omega_i} \frac{\partial u}{\partial t} (x,t) dx = \int_{\Omega_i} D(u(x,t), \nabla u(x,t)) \, dx$$

i.e.,

$$\frac{\partial \bar{u}_i}{\partial t}(t) = \tilde{D}_i(t)$$

(13)

Applying the divergence theorem, we get

$$\tilde{D}_i(t) = -\frac{1}{|\Omega_i|} \int_{\partial \Omega_i} F(u(x,t), \nabla u(x,t)) \cdot \sigma_i(x) \, dx + \bar{S}_i(t)$$

(14)

where $\sigma_i(x)$ denotes the outer normal vector to $\Omega_i$.

Conservativity in the flux computation is ensured if and only if, for two adjacent cells $\Omega_i$ and $\Omega_{i+1}$, the outgoing flux from $\Omega_i$ to $\Omega_{i+1}$ balances with the one from $\Omega_{i+1}$ to $\Omega_i$. In the next subsections, we will describe the time integration and space discretization schemes applied to (13).

Due to the adaptive space discretization, the grid is changing in time, and therefore we first discretized in time and then in space. Here we use an explicit second-order accurate Runge-Kutta (RK2) scheme. Denoting by $\Delta t$ the time step and by $\bar{u}_i^n = \bar{u}_i(t^n)$, where $t^n = n \Delta t$, the RK2 scheme used here has the form

$$\begin{align*}
\bar{u}_i^{n+\frac{1}{2}} &= \bar{u}_i^n + \Delta t \tilde{D}_i^n \\
\bar{u}_i^{n+1} &= \frac{1}{2} \left[ \bar{u}_i^n + \bar{u}_i^{n+\frac{1}{2}} + \Delta t \tilde{D}_i^{n+\frac{1}{2}} \right]
\end{align*}$$

(15)

Denoting by $\bar{u}^n$ the vector $(\bar{u}_i^n)_{i \in \Lambda}$, the discrete time evolution operator $\hat{E}(\Delta t)$ is defined by

$$\bar{u}^{n+1} = \hat{E}(\Delta t) \cdot \bar{u}^n$$

(16)

where

$$\hat{E}(\Delta t) = \mathbf{I} + \frac{\Delta t}{2} \left[ \tilde{D} + \tilde{D}(\mathbf{I} + \Delta t \tilde{D}) \right]$$

and $\mathbf{I}$ denotes the identity operator. The discretization of the operator $\tilde{D}$ is described in the following subsection.

We now consider a fixed time $t^n$ and, in the following, the superscript $n$ is omitted everywhere. For the 1D case, $\Omega_i$ is a segment $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ with step size $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. Eq. (14) becomes

$$\tilde{D}_i = -\frac{1}{\Delta x_i} \left( F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}} \right) + \bar{S}_i$$

(17)

Advective and diffusive terms are approximated differently. For the advective part, we use Roe’s scheme [53] with a second-order ENO interpolation, whereas, for the diffusive part, we choose a second-order accurate centered scheme. Bihari [8] showed that the resulting global scheme, which is non-linear, is second-order accurate in space.
\[
\bar{F}_{i+\frac{1}{2}} = f^R \left( \bar{u}_{i+\frac{1}{2}}^-, \bar{u}_{i+\frac{1}{2}}^+ \right) - \nu \frac{\bar{u}_{i+1} - \bar{u}_i}{\Delta x_{i+\frac{1}{2}}} 
\]

where \( \Delta x_{i+\frac{1}{2}} = \frac{1}{2}(\Delta x_i + \Delta x_{i+1}) \)

The term \( f^R \) denotes, for the advective part, Roe’s approximate solution to the Riemann problem given the left (-) and right (+) values of \( u \). Its scalar version is given by

\[
f^R(u^-, u^+) = \frac{1}{2} \left[ f(u^-) + f(u^+) - |a(u^-, u^+)| (u^+ - u^-) \right] \quad (19)
\]

where

\[
a(u^-, u^+) = \begin{cases} 
\frac{f(u^+) - f(u^-)}{u^+ - u^-} & \text{if } u^- \neq u^+ \\
 f'(u^-) & \text{if } u^- = u^+
\end{cases}
\]

The left and right terms, \( \bar{u}_{i+\frac{1}{2}}^- \) and \( \bar{u}_{i+\frac{1}{2}}^+ \) respectively, are computed using a second-order ENO interpolation

\[
\bar{u}_{i+\frac{1}{2}}^- = \bar{u}_i + \frac{1}{2} M \left( \bar{u}_{i+1} - \bar{u}_i - \bar{u}_{i-1} \right) \\
\bar{u}_{i+\frac{1}{2}}^+ = \bar{u}_{i+1} + \frac{1}{2} M \left( \bar{u}_{i+2} - \bar{u}_{i+1} - \bar{u}_i \right) \quad (20)
\]

where \( M \) is the Min-Mod limiter, which chooses the minimal slope between the left and right sides, i.e.

\[
M(a, b) = \begin{cases} 
a & \text{if } |a| \leq |b| \\
b & \text{if } |a| > |b|
\end{cases}
\]

The source term is approximated by \( \bar{S}_i \approx S(\bar{u}_i) \). For a general non-linear source term, this choice yields a second-order accuracy.

Extension to higher dimension in Cartesian geometries is performed through a tensor product approach. For the 2D case, \( \Omega_{i,j} \) is a rectangle with a volume of size \( |\Omega_{i,j}| = \Delta x_i \Delta y_j \). Eq. (13) can be written as

\[
\frac{\partial \bar{u}_{i,j}^2}{\partial t}(t) = \bar{D}_{i,j}(t) \quad (21)
\]

where

\[
\bar{D}_{i,j} = -\frac{1}{\Delta x_i} \left( \bar{F}_{i+\frac{1}{2},j} - \bar{F}_{i-\frac{1}{2},j} \right) - \frac{1}{\Delta y_j} \left( \bar{F}_{i,j+\frac{1}{2}} - \bar{F}_{i,j-\frac{1}{2}} \right) + \bar{S}_{i,j}.
\]

The same numerical flux as in the 1D case is applied in each direction.

\[
\bar{F}_{i+\frac{1}{2},j} = f^R \left( \bar{u}_{i+\frac{1}{2},j}, \bar{u}_{i+\frac{1}{2},j}^+ \right) - \nu \frac{\bar{u}_{i+1,j} - \bar{u}_{i,j}}{\Delta x_{i+\frac{1}{2}}} \quad (22)
\]

\[
\bar{F}_{i,j+\frac{1}{2}} = f^R \left( \bar{u}_{i,j+\frac{1}{2}}, \bar{u}_{i,j+\frac{1}{2}}^+ \right) - \nu \frac{\bar{u}_{i,j+1} - \bar{u}_{i,j}}{\Delta y_{j+\frac{1}{2}}}
\]

where \( \Delta x_{i+\frac{1}{2}} = \frac{1}{2}(\Delta x_i + \Delta x_{i+1}) \) and \( \Delta y_{j+\frac{1}{2}} = \frac{1}{2}(\Delta y_j + \Delta y_{j+1}) \).

Analogously, for the 3D case, \( \Omega_{i,j,k} \) is a rectangle parallelepiped with a volume of size \( |\Omega_{i,j,k}| = \Delta x_i \Delta y_j \Delta z_k \). Hence we get

\[
\frac{\partial \bar{u}_{i,j,k}}{\partial t}(t) = \bar{D}_{i,j,k}(t) \quad (23)
\]
where

\[ \bar{D}_{i,j,k} = \frac{1}{\Delta x_i} \left( \bar{F}_{i+\frac{1}{2},j,k} - \bar{F}_{i-\frac{1}{2},j,k} \right) - \frac{1}{\Delta y_j} \left( \bar{F}_{i,j+\frac{1}{2},k} - \bar{F}_{i,j-\frac{1}{2},k} \right) \]
\[ - \frac{1}{\Delta z_k} \left( \bar{F}_{i,j,k+\frac{1}{2}} - \bar{F}_{i,j,k-\frac{1}{2}} \right) + \bar{S}_{i,j,k}. \]

The fluxes are in this case

\[ \bar{F}_{i+\frac{1}{2},j,k} = f_R \left( \bar{u}_{i+\frac{1}{2},j,k} - \bar{u}_{i-\frac{1}{2},j,k} \right) - \nu \frac{\bar{u}_{i+1,j,k} - \bar{u}_{i,j,k}}{\Delta x_i} \]
\[ \bar{F}_{i,j+\frac{1}{2},k} = f_R \left( \bar{u}_{i,j+\frac{1}{2},k} - \bar{u}_{i,j-\frac{1}{2},k} \right) - \nu \frac{\bar{u}_{i,j+1,k} - \bar{u}_{i,j,k}}{\Delta y_j}, \quad (24) \]
\[ \bar{F}_{i,j,k+\frac{1}{2}} = f_R \left( \bar{u}_{i,j,k+\frac{1}{2}} - \bar{u}_{i,j,k-\frac{1}{2}} \right) - \nu \frac{\bar{u}_{i,j,k+1} - \bar{u}_{i,j,k}}{\Delta z_k} \]

where \( \Delta x_i = \frac{1}{2}(\Delta x_i + \Delta x_{i+1}), \Delta y_j = \frac{1}{2}(\Delta y_j + \Delta y_{j+1}), \) and \( \Delta z_k = \frac{1}{2}(\Delta z_k + \Delta z_{k+1}). \)

4.2. Conservative fully adaptive multiresolution scheme

The principle of the multiresolution analysis is to represent a set of data given on a fine grid as values on a coarser grid plus a series of differences at different levels of nested dyadic grids. In fact, they constitute an ensemble where each grid is twice finer than the previous one. The differences contain the information of the solution when going from a coarse to a finer grid. In particular, these coefficients are small in regions where the solution is smooth. The data structure needs to be organized as a dynamic graded tree if one wants to compress data, while still being able to navigate through it.

4.3. Dynamic graded tree

In the wavelet terminology, a graded tree structure corresponds to the adaptive approximation. Its difference with the classical non-linear approximation is that the connectivity in the tree structure is always ensured. In other words, no hole is admitted inside the tree. DeVore [22] showed that the difference between both approximations is negligible in terms of required nodes.

**Figure 35.** Example of graded tree data structure in 1D for \( s = 1, s' = 2 \)

Following [14], we first introduce a terminology to define the tree structure.

- **The root** is the basis of the tree;
- A **node** is an element of the tree. Here, every cell, when existing, can be considered as a node;
- A **parent** node has \( 2^d \) **children** nodes, \( d \) being the space dimension of the problem;
- The children nodes of the same parent are called **brothers**;
- A given node has nearest neighbors in each direction, called the **nearest cousins**. The brothers can also be considered as nearest cousins;
• Given a child node, the nearest cousins of the parent node are called the nearest uncles;
• A node is called a leaf when it has no children;
• In order to compute the ingoing and outgoing fluxes of a given leaf, we need its nearest cousins. When one of them is not existing, it is created as virtual leaf. A virtual leaf is not considered as an existing node and is only used for flux computations. As a consequence, no time evolution is made on it.

Fig. 35 illustrates the graded tree structure in 1D. The standard nodes are represented by a thin line, the leaves by a bold line, the virtual leaves by a dotted line.

A dynamic tree is a tree which changes in time. When needed, some nodes can be added or removed. To remain graded, it must respect the following conditions:
• When a child is created, all its brothers are also created in the same time;
• A given node has always its $s$ nearest uncles in each direction, diagonal included. When not existing, create them as nodes;
• A given node has always its $s'$ nearest cousins in each direction. When not existing, create them as virtual leaves;

As a consequence, a node can be removed only if all its brothers can also be removed, and if it is not the nearest uncle of an existing node. The number of nearest cousins $s'$ depends on the accuracy of the space discretization. For a second-order TVD accurate scheme, as the one described in the previous section, a five-point space scheme is applied for each dimension. Therefore we have $s' = 2$. In addition, the number of nearest uncles $s$ depends on the multiresolution accuracy.

4.4. Conservative flux computation

To illustrate the conservative flux computation, we first consider a 1D leaf $\Omega_{l+1,2i+1}$ whose cousins in the right direction $\Omega_{l+1,2i+2}$ and $\Omega_{l+1,2i+3}$ are virtual. Therefore their father $\Omega_{l,i+1}$ is a leaf.

Figure 36. Ingoing and outgoing flux computation in 1D for two different levels.

As shown in Fig. 36, the outgoing flux from $\Omega_{l+1,2i+1}$ in the right direction $F_{l+1,2i+1\rightarrow l+1,2i+2}$ is not balanced with the outgoing flux from $\Omega_{l,i+1}$ in the left direction $F_{l,i+1\rightarrow l,i}$. Of course, we could directly compute the outgoing fluxes from $\Omega_{l+1,2i+1}$ to $\Omega_{l,i+1}$ in 1D, but such a computation cannot be extended to higher dimensions, as we can see in Fig. 37.

Figure 37. Ingoing and outgoing flux computation in 2D for two different levels.

So we decided to compute only the ones at the level $l + 1$ and to set the ingoing flux on the leaf of level $l$ equal to the sum of the outgoing fluxes on the leaves of level $l + 1$, i.e.

$$F_{l,i,j\rightarrow l,i+1,j} = F_{l+1,2i+1,2j\rightarrow l+1,2i+2,2j} + F_{l+1,2i+2,2j+1\rightarrow l+i,2i+2,2j+1}$$
This choice ensures a strict conservativity in the flux computation between cells of different levels, without increasing significantly the number of costly flux evaluations.

4.5. Error analysis

The global error between the cell-average values of the exact solution at the level L, denoted by $\bar{u}^L_{ex}$, and those of the multiresolution computation with a maximum level $L$, denoted by $\bar{u}^L_{MR}$, can be decomposed into two errors

$$||\bar{u}^L_{ex} - \bar{u}^L_{MR}|| \leq ||\bar{u}^L_{ex} - \bar{u}^L_{FV}|| + ||\bar{u}^L_{FV} - \bar{u}^L_{MR}||$$

(25)

where $|| \cdot ||$ denotes e.g. the $L^1$, $L^2$, or $L^\infty$ norms. The first error on the right-hand side, called discretization error is the one of the finite volume scheme on the finest grid of level $L$. It can be bounded by

$$||\bar{u}^L_{ex} - \bar{u}^L_{FV}|| \leq C 2^{-\alpha L}, \ C > 0$$

(26)

where $\alpha$ is the convergence order of the finite volume scheme. In the present case, as we use second-order accurate schemes in time and space, we have $\alpha = 2$.

For the second error, called perturbation error, Cohen et al [14] showed that, if the details on a level $l$ are deleted when smaller than a prescribed tolerance $\epsilon_l$, if the discrete time evolution operator $\bar{E}$ is contractive in the chosen norm, and if the tolerance $\epsilon_l$ at the level $l$ is set to

$$\epsilon_l = 2^{d(l-L)} \epsilon$$

where $d$ is the space dimension, then the difference between finite volume solution on the fine grid and the solution obtained by multiresolution accumulates in time and verifies

$$||\bar{u}^L_{FV} - \bar{u}^L_{MR}|| \leq C n \epsilon, \ C > 0$$

(27)

where $n$ denotes the number of time steps. At a fixed time $T = n \Delta t$, this leads to

$$||\bar{u}^L_{FV} - \bar{u}^L_{MR}|| \leq C \frac{T}{\Delta t} \epsilon, \ C > 0.$$  

For the linear convection-diffusion equation with the numerical scheme defined above, the time step $\Delta t$, following (??), must verify

$$\Delta t \leq \frac{\Delta x^2}{4\nu + c\Delta x}$$

Denoting $X$ the size of the domain and $\Delta x$ the smallest space step, we have $\Delta x = X \, 2^{-L}$, from which we deduce that

$$\Delta t = C \frac{\Delta x^2}{4\nu + c\Delta x} = C \frac{X^2 \, 2^{-2L}}{4\nu + cX \, 2^{-L}} , \ 0 < C < 1.$$  

If we want the perturbation error to be of the same order as the discretization error, we need that

$$\frac{\epsilon}{\Delta t} \propto 2^{-\alpha L}$$

i.e.

$$\epsilon 2^{2L} (4\nu + cX \, 2^{-L}) \propto 2^{-\alpha L}$$

Defining the Peclet number $Pe = cX\nu^{-1}$, the previous condition can be rewritten as

$$\epsilon \propto \frac{2^{-(\alpha+1)L}}{Pe + 2^{L}T^2}$$

(28)
For the inviscid case (i.e. $\nu = 0$ or $Pe \to +\infty$), (28) is equivalent to the result obtained by Cohen et al [14], i.e. $\epsilon \propto 2^{-(\alpha+1)L}$. In the numerical computations in section 6.4.6, the so-called reference tolerance will be set to

$$\epsilon_R = C \frac{2^{-(\alpha+1)L}}{Pe + 2^{L+2}}$$

(29)

To choose an acceptable value for the factor $C$, a series of computations with different tolerances will be necessary, as shown in subsection 6.1.1.

5. Time Adaptive Strategies for the Multiresolution Scheme

In the following we consider three time adaptive strategies for the MR scheme.

5.1. MR/CTS scheme

For ODE simulations, instead of using a fixed time step $\Delta t$ chosen a priori, it can be advantageous to have a technique that automatically adjusts its size dynamically. The time step $\Delta t$ must be chosen sufficiently small to satisfy a required precision of the computed results, denoted by $\delta_{\text{desired}}$, but it must be sufficiently large to avoid unnecessary computational work. Typically, if $\bar{U}(1)$ is the approximation of $\bar{U}(t + \Delta t)$ developments of the local truncation errors of the form

$$\bar{U}(t + \Delta t) - \bar{U}(1) = C\Delta t^{p+1} + O(\Delta t^{p+2})$$

(30)

could be used to find the step size required to attain a specified accuracy. However, since the leading constant $C$ is not known a priori, practical error estimates are necessary. To estimate the local truncation error, the idea is to apply two embedded ODE solvers, one with order $p$ and the other with order $p + 1$. If $\bar{U}(2)$ is the approximation of $\bar{U}(t + \Delta t)$ generated by the method of order $p + 1$, then, for sufficiently small $\Delta t$ we have,

$$\bar{U}(1) - \bar{U}(2) \approx C_1 \Delta t^{p+1} - C_2 \Delta t^{p+2} \approx C_1 \Delta t^{p+1}.$$

This yields the estimate

$$C_1 \approx \frac{\bar{U}(1) - \bar{U}(2)}{\Delta t^{p+1}}.$$

Hence the step size required to maintain the local truncation error of the first scheme below $\delta_{\text{desired}}$ has the form $\Delta t_{\text{new}} = \Delta t\xi$, where

$$\xi = \left[\frac{\delta_{\text{desired}}}{|\bar{U}(1) - \bar{U}(2)|}\right]^{1/(p+1)}.$$

If we want to prevent the time step of varying too abruptly or to be sure that $\Delta t_{\text{new}}$ in fact will produce an error less than $\delta_{\text{desired}}$, some care is needed. In the present implementation, we can not go back to the previous time step once the solution at the new time step is computed due to the low storage memory model we are using. Hence we decided to limit the increase of the time step by introducing a so-called safety factor ($S$). The new time step $(\Delta t)_{\text{new}}$ is chosen such that

$$\frac{S}{2} \leq \frac{(\Delta t)_{\text{new}} - (\Delta t)}{(\Delta t)} \leq \frac{S}{2}.$$

This method is typically used for ordinary differential equations to avoid bad choices of the time step. For memory reasons we can not go back in the evolution once we have computed the solution at new the time step, as e.g. proposed by [30]. Using a more stringent limiter or a safety factor the choice of non admissible time steps can be avoided. The drawback of the limiter is that, in case that the initial time step is far from the ideal
time step CPU time could be wasted as the time step can not be increased sufficiently fast. To overcome this, we define \( S = S(t) \) with an exponential decay during the first time steps, i.e.,

\[
S(t) = (S_0 - S_{\text{min}}) \exp \left( -\frac{t}{\Delta t} \right) + S_{\text{min}}.
\]

The behavior of the limiter \( S(t) \) for \( t = 0 \) is the maximal allowed variation \( S_0 \) and, for \( t \to \infty \), it is \( S_{\text{min}} \), where \( S_{\text{min}} < S_0 \). In the present paper we use \( S_0 = 0.1 \) and \( S_{\text{min}} = 0.01 \) for all case studies presented. This means that we allow 10% of variation of the time step in the initial time step and after few iterations we allow only 1%.

For the applications of the present paper, we adopt the Runge-Kutta Fehlberg 2(3), where the second order scheme is the one defined in (??). Given \( \bar{U}^* \) computed as in (??), then the third order scheme computes one more stage to obtain

\[
\bar{U}^{**} = \frac{1}{4} \left[ 3 \bar{U}^n + \bar{U}^* - \Delta t F(\bar{U}^*) \right],
\]

\[
\bar{U}^{n+1} = \frac{1}{3} \left[ \bar{U}^n + 2 \bar{U}^{**} - 2 \Delta t F(\bar{U}^{**}) \right].
\]

In the MR/CTS setting, \( \bar{U} \) is replaced by \( \bar{U}_{\text{MR}} \), and \( F(\bar{U}) \) should be taken as \( F_{\text{MR}}(\bar{U}_{\text{MR}}) \).

### 5.2. MR/LTS scheme

Assuming that \( \Delta t \) is the time step used to evolve the cells on the finest scale level \( L \), the main principle in the MR/LTS algorithm is to evolve the cells at lower level \( 0 \leq \ell < L \) with time step \( 2^{L-\ell} \Delta t \). Consequently, if \( \bar{U}^n_{\text{MR/LTS}} \) represents the numerical solution at \( t^n = n \Delta t \) on the adaptive grid \( \Gamma^n = \Gamma^n_L \), then one complete time cycle of the local time stepping evolution operator evolves the solution from \( t^n \) to \( t^{n+2^L} \). This adaptive time strategy combined with the multiresolution scheme gives the fully adaptive time cycle

\[
\bar{U}^n_{\text{MR/LTS}} = (T(\epsilon) E_{\text{MR/LTS}} R) \bar{U}^n_{\text{MR/LTS}},
\]

where \( E_{\text{MR/LTS}} \) denotes the evolution operator using the adaptive MR discretization combined with the scale-dependent time stepping. Recall that \( T(\epsilon) \) corresponds to the coarsening (thresholding), and \( R \) to the refinement operations. For more details of the implementation of the MR/LTS scheme, we refer to [24].

### 5.3. MR/CTS/LTS scheme

The MR/CTS/LTS scheme combines the two previous time adaptive strategies as follows:

- First, the MR/CTS strategy is applied just to determine a step size \( \Delta t \) required to attain a specified accuracy with a global time stepping.
- Second, the MR/LTS cycle is computed using the obtained step size \( \Delta t \) for the evolution of the cell averages in the finest scale level.
- Third another MR/CTS time step is done to adjust the time step, and so on.

This technique hence allows to change the time step size during the time evolution to control the truncation error in time and to benefit from the local local time stepping to reduce further the computational cost. Nevertheless, we should mention that the local time stepping implies large time cycles for many refinement levels. The size of the time cycle is increasing with the number of refinement levels. Therefore a wide range of adaptive scales in MR/CTS/LTS implies that the time step control becomes less efficient to adjust \( \Delta t \) rapidly.
6. FV-MR APPLICATIONS

In this section, we present some applications of the MR methods to convection-diffusion, reaction-diffusion, Euler equations and CVS modeling.

6.1. Convection-diffusion equations

6.1.1. In one space dimension

We consider the linear convection-diffusion equation for 
\[
\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad (x, t) \in [-1, 1] \times [0, +\infty), \tag{31}
\]
with boundary values \(u(-1,t) = 1\) and \(u(1,t) = 0\). All the numerical solutions of this section are compared with the analytical solution (see e.g. [36])
\[
uex(x, t) = \frac{1}{2} \text{erfc} \left( \frac{x - t}{\sqrt{2 \nu t}} \right), \tag{32}
\]
and the computations start at \(u(x, 0) = u_{\text{ex}}(x, 0.1)\). Setting the Peclet number \(Pe = \frac{1}{\nu}\), the tolerance parameter \(\epsilon\) is chosen according to the formula in [57]
\[
\epsilon = \epsilon_L = 5 \cdot 10^8 \frac{2^{-3L}}{Pe + 2^{L+2}}. \tag{33}
\]

Using \(Pe = 1000\), and \(L = 9\), the plots in Figure 38 represent the stability regions in the \(Re \times \sigma\) plane for the FV, MR and MR/LTS methods obtained by checking the solution at \(t = 0.5\). For all schemes we performed computations for \(\sigma\) ranging from 0.125 to 1.875 in steps of 0.125 and for \(Re\) from 1 to 10 in steps of 1. The stability region for the FV scheme depicted in Figure 38(left) fits well with the theoretical one presented in Figure ??(left). As shown in Figure 38(right), the stability domains for MR and MR/LTS schemes coincide. Moreover, they are quite similar to the one found for the FV scheme.

<table>
<thead>
<tr>
<th>Terms</th>
<th>(L_1) norm</th>
<th>(L_2) norm</th>
<th>(L_\infty) norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(|u_{\text{ex}} - u_{\text{FV}}|)</td>
<td>1.9980 \times 10^{-3}</td>
<td>2.9597 \times 10^{-4}</td>
<td>1.4969 \times 10^{-2}</td>
</tr>
<tr>
<td>(|u_{\text{FV}} - u_{\text{MR}}|)</td>
<td>3.0091 \times 10^{-5}</td>
<td>5.2205 \times 10^{-6}</td>
<td>3.7992 \times 10^{-4}</td>
</tr>
<tr>
<td>(|u_{\text{MR}} - u_{\text{LTS}}|)</td>
<td>4.2655 \times 10^{-5}</td>
<td>8.7415 \times 10^{-6}</td>
<td>9.1523 \times 10^{-4}</td>
</tr>
<tr>
<td>(|u_{\text{MR}} - u_{\text{LTS}}|)</td>
<td>1.9890 \times 10^{-3}</td>
<td>2.9403 \times 10^{-4}</td>
<td>1.5501 \times 10^{-2}</td>
</tr>
</tbody>
</table>

Table 3. Convection-diffusion equation: contributions of the different errors at \(t = 0.5\) for \(\sigma = 0.5\), \(Pe = 1000\), \(\epsilon = 1.2 \times 10^{-3}\) and \(L = 9\).

Table 3 presents comparison results for numerical solutions of the convection-diffusion equation at \(t = 0.5\) obtained with different methods and \(L = 9\) scales. The MR/LTS discretization error \(\|u_{\text{ex}} - u_{\text{LTS}}\|\) can be decomposed into the FV discretization error \(\|u_{\text{ex}} - u_{\text{FV}}\|\), the thresholding error \(\|u_{\text{FV}} - u_{\text{MR}}\|\) and the local time stepping error \(\|u_{\text{MR}} - u_{\text{LTS}}\|\). We can see that the the MR/LTS discretization error is almost identical with the FV discretization error, and the thresholding error is two orders of magnitude smaller.

Table 4 presents errors for different schemes versus the maximal scale \(L\). We observe that MR and MR/LTS computations yield the same second order accuracy of the reference FV computation on a regular grid.

The gain in CPU time of the MR/LTS computation with respect to the MR one is shown in Figure 39 (right) showing the significant gain of CPU time using the MR/LTS method, which increases with the number of levels. Moreover, it can be noticed that almost the same memory is required for both methods (Figure 39, left).
Figure 38. Numerically computed stability regions for the convection-diffusion equation using centered second order FV (left), MR and MR/LTS (right) schemes, with $L = 9$.

Table 4. Convection-diffusion: Error in $L_\infty$ and $L_1$ norm for FV, MR and MR/LTS methods obtained at $t = 0.5$ for $\sigma = 0.5$, $Pe = 1000$, and $L = 9$ to 13.

6.1.2. In two space dimensions

In this part, we study the performances and check the second-order accuracy of the multiresolution scheme with the reference tolerance in the 2D case. Therefore we consider the dimensionless equation for $(x, y, t) \in [-5, 5]^2 \times [0, +\infty)$

$$\frac{\partial u}{\partial t} + V \cdot \nabla u = \frac{1}{Pe} \nabla^2 u$$  \hspace{1cm} (34)

with the initial condition $u(x, y, 0) = u_0(x, y)$. Here we consider a convection in the $x$-direction, i.e. $V = (1, 0)^T$. For the initial condition $u_0(x, y) = \delta(x) \delta(y)$, where $\delta$ denotes the Dirac distribution, we have an analytic solution in an infinite domain

$$u(x, y, t) = \frac{Pe}{4 \pi t} e^{-\frac{Pe}{x^2 + y^2}}$$  \hspace{1cm} (35)

For a Gaussian initial condition, we can change variables $(x \leftarrow x - \tau, t \leftarrow t - \tau$, where $\tau > 0)$. Thus, given the initial condition

$$u_0(x, y) = \frac{Pe}{4 \pi \tau} e^{-\frac{Pe}{x^2 + y^2}}$$
we get the analytic solution

\[ u(x, y, t) = \frac{Pe}{4 \pi (t + \tau)} e^{-Pe \frac{(x - t \tau)^2 + y^2}{4 (t + \tau)}} \]  \hspace{1cm} (36) 

For the numerical computations, the boundaries are taken far enough from the Gaussian bump, so that their influence can be considered as negligible.

Numerical results for a given scale and a given tolerance at Pe=1000. The numerical solution of (34) for an initial Gaussian bump is represented in Fig. 40 for \( Pe = 1000, \epsilon = \epsilon_R, \) and \( L = 8 \) scales, which represents a maximum of \((2^8)^2 = 256^2\) cells. In the figures where the corresponding meshes are plotted, each point represents a leaf. For the initial condition, we set \( \tau = 0.1.\)

We observe the phenomenon of linear propagation of the 2D Gaussian bump in the \( x \)-direction. The diffusion effect is difficult to detect, but we can see that the radius of the smallest circle slightly decreases with time. We also remark that the adaptive mesh follows well the propagation. Nevertheless, although the mesh is well symmetric at the initial condition, it remains symmetric only on the two sides of the \( x \)-axis, whereas it is not in the other direction. This is due to the fact that the advection takes place in the \( x \)-direction.

Dependency of CPU time, memory, and error on maximal level with the reference tolerance at Pe=1000. As the definition of the reference tolerance is independent of the space dimension, we use the same one as in the 1D case, i.e. \( C = 5.10^4.\) We remark here that both CPU and memory compressions are low and stable with \( L \) (around 15 \% for the CPU compression, 10 \% for the memory compression), while the corresponding errors confirm that the computations are well second-order accurate (Fig. 41). This time, as no discontinuity exists in the initial condition and as the equation is linear, no steep gradient exists, which explains that the same percentage of leaves is used whatever \( L \), although more levels are used around the Gaussian bump.

6.2. Diffusion-reaction equations

Another prototype of a non-linear parabolic equation is the reaction-diffusion equation. Here the non-linearity is no more in the advective term, as e.g. for the viscous Burgers equation, but in the source term.
6.2.1. In one space dimension

In its dimensionless form, the reaction-diffusion equation in one space dimension, for \((x,t) \in [0, 20] \times [0, +\infty)\) can be written as,

\[
\frac{\partial u}{\partial t} = \frac{\beta^2 u}{\partial x^2} + S(u) \tag{37}
\]

\[
S(u) = \frac{\beta^2}{2} (1-u) \exp \frac{\beta(1-u)}{\alpha(1-u)-1}. \tag{38}
\]
where $\alpha$ is the temperature ratio and $\beta$ is the dimensionless activation energy (Zeldovich number). We choose as initial condition

$$u_0(x) = \begin{cases} 1 & \text{if } x \leq 1; \\ \exp(1 - x) & \text{if } x > 1. \end{cases}$$  \hspace{1cm} (39)$$

This equation yields a model for a 1D premixed flame propagation where heat and mass diffusivities are equal. The function $u$ is the dimensionless temperature. It varies between 0 and 1. The non-dimensional partial mass of the unburnt gas is $1 - u$. We choose a Neumann condition at the left boundary and a Dirichlet condition at the right boundary.

$$\frac{\partial u}{\partial x}(0,t) = 0$$
$$u(20,t) = 0$$  \hspace{1cm} (40)

For the numerical computation, the parameters are $\alpha = 0.8$ and $\beta = 10$. The dimensionless time goes from $t = 0$ to $t = t_f = 10$.

![Figure 42](image_url)

**Figure 42.** Left: Initial condition for $u$ (dashed) and $S(u)$ (dotted), solution by finite volume method for $u$(plain), solution by multiresolution method for $u$ (circles) and $S(u)$ (dash-dotted) at $t = 10$ for the reaction-diffusion equation, $\alpha = 0.8$, $\beta = 10$, $L = 8$, $\epsilon = 5.10^{-2}$. Right: corresponding tree structure at $t = 10$.

In Fig. 42, we observe the flame propagation in the $x$-direction. The highest level is reached in the region of the reaction zone, i.e. for $x \approx 10$. We can also notice that the multiresolution computation gives the same result as the finite volume one. We then compare the value of the flame velocity, defined by

$$v_f = \int_{\Omega} S \, dx$$  \hspace{1cm} (41)$$

with the asymptotic value given in Peters and Warnatz [51]. We observe that the value of $v_f$ is approximately the same for finite volume and multiresolution computations, for the three different values of the tolerance. All these values are comparable with the asymptotic one. Hence we can conclude that the value $\epsilon = 5.10^{-2}$ is well adapted.

6.2.2. In two space dimensions

In this part, the 2D reaction-diffusion equation is solved for a flame ball initially stretched in one direction. As in the 1D case, heat and mass diffusivities are equal. This test-case was originally proposed in [31]. The
Asymptotic 0.908

<table>
<thead>
<tr>
<th>Method</th>
<th>$v_f$</th>
<th>% CPU</th>
<th>% Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV</td>
<td>0.916</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>MR $\epsilon = 5.10^{-2}$</td>
<td>0.917</td>
<td>36.0%</td>
<td>32.6%</td>
</tr>
<tr>
<td>MR $\epsilon = 10^{-2}$</td>
<td>0.916</td>
<td>54.2%</td>
<td>47.1%</td>
</tr>
<tr>
<td>MR $\epsilon = 10^{-3}$</td>
<td>0.916</td>
<td>79.0%</td>
<td>67.2%</td>
</tr>
</tbody>
</table>

Table 5. Flame velocity, CPU and memory compression for finite volume and multiresolution methods.

The resulting equation in the dimensionless form is

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + S(u)$$

(42)

where $(x, y, t) \in [-20, 20]^2 \times [0, +\infty)$ and $S(u)$ verifies (38). The initial condition is $u(x, y, 0) = u_0(r)$, where $u_0$ verifies (39) and $r^2 = x^2 + y^2$. We perturb the circular initial condition by stretching the circle in one direction and applying a rotation. Therefore, we have

$$r = \sqrt{\frac{X^2}{a^2} + \frac{Y^2}{b^2}}$$

where

$$\begin{cases}
X &= x \cos \theta + y \sin \theta \\
Y &= -x \sin \theta + y \cos \theta
\end{cases}$$

We consider that the reaction takes place in a closed box with adiabatic walls, and hence we choose Neumann conditions on the boundary, i.e.

$$\left.\frac{\partial u}{\partial n}\right|_{\partial \Omega} = 0.$$  

The parameters are the Zeldovich number $\beta = 10$, and the temperature ratio $\alpha = 0.8$. The aspect ratio of the ellipse is given by $a = 2$, $b = 1$, and the rotation angle is $\theta = -\frac{\pi}{6}$. The elapsed time is $t = 10$. For the multiresolution computation, the tolerance is set to $\epsilon = 5.10^{-2}$, like in the 1D case. As in Fröhlich and Schneider [31], we observe a relaxation of the elliptic flame towards a circularly symmetric structure which is then growing in space. The finest resolution would correspond to $(2^8)^2 = 256^2$ cells. On average we only use 6763 out of 256$^2$ = 65536 control volumes, which yields a memory compression of 10.3%. Comparing the elapsed CPU time with the one obtained by the same finite volume scheme on the finest grid, we get a CPU compression of 19.9%.

6.2.3. In three space dimensions

The previous equation is now extended to three dimensions, in order to study the evolution of a 3D flame ball initially stretched in one direction, for equal heat and mass diffusivities. Therefore we consider the dimensionless equation for $(x, y, z, t) \in [-20, 20]^3 \times [0, +\infty)$

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + S(u)$$

(43)

where $S(u)$ verifies (38). The initial condition is now $u(x, y, z, 0) = u_0(r)$, where $u_0$ verifies (39) and $r^2 = x^2 + y^2 + z^2$. The spherical initial condition is stretched in one direction and the same rotation is applied as previously. Therefore we have

$$r = \sqrt{\frac{X^2}{a^2} + \frac{Y^2}{b^2} + \frac{Z^2}{c^2}}$$
where

\[
\begin{align*}
X &= x \cos \theta + y \sin \theta \\
Y &= -x \sin \theta + y \cos \theta \\
Z &= z
\end{align*}
\]

As in the 2D case, we consider that the reaction takes place in a closed box with adiabatic walls, which means that

\[
\frac{\partial u}{\partial n} \bigg|_{\partial \Omega} = 0
\]

The Zeldovich number and the temperature ratio are the same as in the 2D case. The aspect ratio of the ellipsoid is given by \(a = 2, b = 1, c = 1\), and the rotation angle is \(\theta = -\frac{\pi}{3}\). The elapsed time is \(t = 12\). For the multiresolution computation, the tolerance is set to \(\epsilon = 5 \times 10^{-2}\).

We observe, as in the 2D case, a relaxation of the ellipsoidal flame towards a spherically symmetric structure which is then growing in space, which shows that the perturbation is not amplified. The finest resolution would correspond to \((2^7)^3 = 128^3\) cells. On average we only use 39636 out of \(128^3 = 2097152\) control volumes, which yields a memory compression of 1.89%. Comparing the elapsed CPU time with the one obtained by the same finite volume scheme on the finest grid, we get a CPU compression of 7.64%. For splitting flames, we refer to Roussel and Schneider [54].

6.3. Euler equations

6.3.1. 1D: Lax test-case

The compressible Euler equations in 1D could be written in the following form,

\[
\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} = 0,
\]

(44)
Figure 44. Isosurfaces $u = 0.5$ (black), 0.1 (gray) and corresponding mesh at $t = 0$ (left), $t = 6$ (middle), and $t = 12$ (right) for the 3D reaction-diffusion equation.

with,

$$Q = \begin{pmatrix} \rho \\ \rho v \\ \rho e \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} \rho v \\ \rho u^2 + p \\ (pe + p)v \end{pmatrix}$$

where $\rho = \rho(x,t)$ is the density, $v = v(x,t)$ is the velocity in the $x$-direction, $e = e(x,t)$ is the energy per unit of mass and $p = p(x,t)$ is the pressure.

The system is completed by the equation of state for an ideal gas

$$p = \rho r T = (\gamma - 1) \rho \left( e - \frac{u^2}{2} \right). \quad (45)$$

where $T = T(x,t)$ is the temperature, $\gamma = \gamma(x,t)$ is the specific heat ratio and $r$ is the universal gas constant.

In dimensionless form, we obtain the same system of equations, but the equation of state becomes

$$p = \frac{\rho T}{\gamma Ma^2}, \quad (46)$$

where $Ma$ denotes the Mach number.
We compute the Lax test-case in the computational domain $\Omega = [-1, 1]$. The initial condition is given by

$$Q(x, t=0) = \begin{cases} 
(0.445, 0.311, 8.928) & \text{for } x < 0, \\
(0.5, 0, 1.4275) & \text{otherwise.}
\end{cases} \quad (47)$$

Neumann boundary conditions are applied on both sides of the domain. The physical parameters are $Ma = 1$ and $\gamma = 1.4$ and the computations are performed until physical time $t = 0.32$. In this case, a rarefaction wave is moving to the left when $t > 0$, and a contact discontinuity and a shock wave are propagating to the right (cf. Figure 45, bottom, left). More details on this test-case and its exact solution can be found in [42, 70].

The time step $\Delta t$ used in the FV, MR, MR/LTS computations is $\Delta t = 4.343 \times 10^{-4}$ and the threshold is chosen as $\epsilon = 5 \times 10^{-3}$ for the MR methods. For the controlled time step experiments MR/CTS and MR/CTS/LTS we use a required precision $\delta_{\text{desired}} = 0.07$. For MR/CTS we start with $\sigma = \Delta t/\Delta x = 0.7$ and we find a minimum time step $4.343 \times 10^{-4}$. For MR/CTS/LTS with $\sigma = 0.5$ we find $\Delta t = 4.786 \times 10^{-4}$. Therefore MR/CTS/LTS uses a larger time step than MR/CTS to compute the solution.

In Figure 45 we observe that the numerical solutions fit well the exact one. On the contact region all numerical solutions are a slightly smoother, however they detect well the shock region present in the exact solution. As expected, the highest level on the adaptive grids is reached in the vicinity of the discontinuities of the functions and their first derivatives (Figure 45). Let us remark that, for both MR/CTS and MR/CTS/LTS methods, the time step is converging towards a value corresponding to $\sigma = 0.45$.

In Table 6, we separately compare second order and third order methods for the time integration. For the $L^1$ error of density, shown in Table 6, the MR/CTS/LTS and MR/CTS present the best results of the MR methods.

The kinetic energy of MR, MR/LTS, MR/CTS and MR/CTS/LTS, defined by $E = \frac{1}{2} \sum_{i \in \Lambda} \rho_i |\vec{v}_i|^2 h_{x,i}$, is compared with the kinetic energy of the exact solution. In all the cases, it differs not more than 0.1%, as one can observe in Table 6. For the experiments with $L = 10$, the best CPU performance is given by MR/CTS/LTS using 30.30% of FV RK3 CPU time (see Table 6). Moreover, the MR/LTS (RK2) scheme is faster than MR/RK2 one without loosing too much accuracy, and the MR/CTS/LTS scheme yields smaller errors than the MR/LTS one, for approximately the same CPU time. Concerning the memory compression we find approximately the same result whatever the chosen MR method. It proves that the space adaption is largely independent from the chosen method for time adaption. Nevertheless, the LTS methods require a little more memory due to the fact that a leaf cell can only be removed when the time cycle is finish on its level.

<table>
<thead>
<tr>
<th>Method</th>
<th>$L^1$ - error on $\rho$ (%)</th>
<th>E error (%)</th>
<th>CPU Time (%)</th>
<th>Memory (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV, RK2</td>
<td>0.0086</td>
<td>0.13</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>MR, RK2</td>
<td>0.0087</td>
<td>0.13</td>
<td>39.13</td>
<td>44.69</td>
</tr>
<tr>
<td>MR/LTS, RK2</td>
<td>0.0091</td>
<td>0.14</td>
<td>30.44</td>
<td>44.51</td>
</tr>
<tr>
<td>FV, RK3</td>
<td>0.0085</td>
<td>0.11</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>MR, RK3</td>
<td>0.0085</td>
<td>0.11</td>
<td>36.36</td>
<td>44.69</td>
</tr>
<tr>
<td>MR/CTS, RK2(3)</td>
<td>0.0084</td>
<td>0.16</td>
<td>36.36</td>
<td>45.01</td>
</tr>
<tr>
<td>MR/CTS/LTS, RK2(3)</td>
<td>0.0083</td>
<td>0.08</td>
<td>30.30</td>
<td>47.49</td>
</tr>
</tbody>
</table>

Table 6. Comparison for the errors and speed-up of the numerical solutions for the 1D Euler equations using the Lax test-case at time $t = 0.32$ with $L = 10$ levels.
6.3.2. 2D: elliptical implosion

As 2D test-case, we study an inviscid implosion phenomenon. In a square box, an elliptic diaphragm separates two regions which contain the same gas, but with different conditions of pressure and temperature. Inside the diaphragm, the pressure is lower than outside. In both regions, the gas is at rest, i.e., the initial velocity vanishes everywhere. On the boundary we impose zero flux boundary conditions.

At $t = 0$, the diaphragm is broken. A shock wave and a contact discontinuity are moving towards the center, while a rarefaction wave is moving in the opposite direction.

To obtain the 2d Euler equations in its conservative form we replace in (9), $u$ by $(\rho, \rho v_1, \rho v_2, \rho e)^T$ which corresponds to the vector of conserved variables. Here $\rho$ is the fluid density, $v_1, v_2$ are the velocity components in $x$ and $y$ direction and $e$ is the energy per unit of mass. The flux function $f = (f_1, f_2)^T$ is given by

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

where the pressure $p$ satisfies

$$p = (\gamma - 1)\rho(e - \frac{1}{2}(v_1^2 + v_2^2)).$$
and $\gamma = 1.4$ denotes the specific heat ratio.

The initial conditions are

$$\rho(r, 0) = \begin{cases} 
1 & \text{if } r \leq r_0 \\
0.125 & \text{if } r > r_0,
\end{cases} \quad (48)$$

$$\rho e(r, 0) = \begin{cases} 
2.5 & \text{if } r \leq r_0 \\
0.25 & \text{if } r > r_0,
\end{cases} \quad (49)$$

$v_1 = v_2 = 0$ and $r_0$ denotes the initial radius.

This initial condition is stretched in one direction and a rotation is applied. The radius thus becomes

$$r = \sqrt{\frac{X^2}{a^2} + \frac{Y^2}{b^2}},$$

with the new coordinates

$$X = x \cos \theta - y \sin \theta,$$
$$Y = -x \sin \theta + y \cos \theta.$$

Figure 46 (left) displays the initial density field with $a = 1/3$, $b = 1$, $\theta = -\pi/3$, $r_0 = 1$, computational domain $\Omega = [-2, 2]^2$, $L = 9$ and $\epsilon = 10^{-2}$. Figure 46(right) corresponds to the respective adaptive grid.

![Initial density field](image1)

![Adaptive grid](image2)

**Figure 46.** Initial condition of the 2D Euler equations: density contours and corresponding adaptive grid for $L=9$.

Figure 47 displays the density field at $t = 0.4$ computed with the FV RK3 and the MR RK3, MR/LTS RK2 and MR/CTS RK 2(3) methods. In Figure 48, the MR/CTS/LTS solution is plotted, together with the corresponding adaptive grid. We observe that all solutions match well the FV RK3 computation on the regular finest grid and that the adaptive grid tracks well the discontinuity and the steep gradients of the solution.

Density cuts are shown in Figure 49. To check for grid convergence we also superpose results of a FV computation using one additional level, i.e. $L = 10$. We find a rather good agreement between both FV computations and hence can conclude that $L = 9$ levels yield a sufficient solution for this problem. Concerning the different adaptive MR computations we also find a good agreement, which is further confirmed by the zoom (Fig. 49, right) into the central region of the computational domain.

In Table 7 we compare the computational efficiency and the precision of the different numerical methods using either second or third order time integration schemes. The reference computations are given by the FV scheme.
on the finest regular grid. For the memory compression we find for all adaptive schemes almost equivalent results, only about 26% of the memory required for the FV computations is used. The best precision of the adaptive computation is obtained with the MR/CTS method followed by the two MR/RK2 and MR/RK3 methods. The MR/LTS computations show slightly larger errors due to the addition interpolation step. Concerning the speed up of the adaptive schemes. Table 7 shows that the MR/CTS/LTS method is 1.5 times faster than the MR/RK3 method. The latter scheme is already 5.5 faster than the FV/RK3 reference computation, which nicely illustrates the additional speed of adaptive and local time stepping.

6.3.3. 2D: Liu-Lax Riemann Problem

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

Initial condition and parameters. The case study chosen here is a typical Riemann problem for 2D gas dynamics treated e.g. in [41], and initially discussed in [63,71]. The initial data are constant in each quadrant (Figure 50),

Figure 47. 2D Euler equations: density contours at $t = 0.4$, $L = 9$, for the FV (top left), MR (top right), MR/CTS (bottom left), and MR/LTS (bottom right) methods, $\epsilon = 2 \cdot 10^{-3}$. 
and the values are given in Table 8. This test case corresponds to the configuration #5 in [41]. This classical test case only involves contact discontinuities and generates motion in opposite directions.

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 $256 \times 256, 512 \times 512$, and $1024 \times 1024$ 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 $\epsilon$ 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 $\epsilon$, we refer to [57].
<table>
<thead>
<tr>
<th>Method</th>
<th>E error (%)</th>
<th>CPU Time (%)</th>
<th>Memory (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV/RK2 (ref.)</td>
<td>0.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>MR/RK2</td>
<td>0.08</td>
<td>28.71</td>
<td>26.55</td>
</tr>
<tr>
<td>MR/LTS, RK2</td>
<td>0.26</td>
<td>28.76</td>
<td>26.26</td>
</tr>
<tr>
<td>FV/RK3 (ref.)</td>
<td>0.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>MR/RK3</td>
<td>0.07</td>
<td>27.21</td>
<td>26.60</td>
</tr>
<tr>
<td>MR/CTS, RK2(3)</td>
<td>0.01</td>
<td>24.33</td>
<td>26.79</td>
</tr>
<tr>
<td>MR/CTS/LTS,RK2(3)</td>
<td>0.15</td>
<td>17.96</td>
<td>26.17</td>
</tr>
</tbody>
</table>

Table 7. Comparison for the numerical solutions of the 2D Euler equations for $t = 0.4$, $L = 9$ and $\epsilon = 2 \cdot 10^{-3}$. The reference FV kinetic energy is $E = 23.18$ for RK2 and $E = 23.91$ for RK3.

Figure 50. Sketch of the initial condition

Table 8. Initial Values for the Lax-Liu configuration #5 [41]

<table>
<thead>
<tr>
<th>Variables</th>
<th>Domain position</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I</td>
</tr>
<tr>
<td>Density ($\rho$)</td>
<td>1.00</td>
</tr>
<tr>
<td>Pressure ($p$)</td>
<td>1.00</td>
</tr>
<tr>
<td>Velocity component ($v_1$)</td>
<td>-0.75</td>
</tr>
<tr>
<td>Velocity component ($v_2$)</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

The basic coarse grid for the AMR method is a $128 \times 128$ 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 ?? 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 9.

Table 9 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. Further on, counters were added to the code to compute the number of cells updated, $\sum C$, and the number of updated leaf cells, $\sum L$, which are not covered by further refinement.

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 $\sum C$ is only slightly larger in the latter case, this increase
Table 9. 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.

<table>
<thead>
<tr>
<th></th>
<th>uni</th>
<th>no LT</th>
<th>with LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_l$</td>
<td>1</td>
<td>2.4</td>
<td>2.2,2</td>
</tr>
<tr>
<td>Integration [%]</td>
<td>98.4</td>
<td>70.3</td>
<td>48.5, 86.2, 59.0</td>
</tr>
<tr>
<td>Interpolation [%]</td>
<td>-</td>
<td>4.2</td>
<td>3.8, 4.2</td>
</tr>
<tr>
<td>Flux correction [%]</td>
<td>-</td>
<td>2.7</td>
<td>5.1, 1.1</td>
</tr>
<tr>
<td>Flagging [%]</td>
<td>-</td>
<td>4.1</td>
<td>7.0, 1.1</td>
</tr>
<tr>
<td>Regridding [%]</td>
<td>-</td>
<td>11.1</td>
<td>17.2, 4.1</td>
</tr>
<tr>
<td>Clustering [%]</td>
<td>-</td>
<td>6.0</td>
<td>16.9, 1.9</td>
</tr>
<tr>
<td>Misc [%]</td>
<td>1.1</td>
<td>1.6</td>
<td>1.5, 1.4</td>
</tr>
<tr>
<td>CPU time [s]</td>
<td>4531</td>
<td>1249</td>
<td>1721, 865, 1017</td>
</tr>
<tr>
<td>$\sum L [M]$</td>
<td>2025</td>
<td>363</td>
<td>326, 311, 231</td>
</tr>
<tr>
<td>$\sum L [M]$</td>
<td>2025</td>
<td>336</td>
<td>252, 305, 202</td>
</tr>
<tr>
<td>Int. perf. [k/s]</td>
<td>454</td>
<td>413</td>
<td>390, 417, 385</td>
</tr>
</tbody>
</table>

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 $\sum L$ 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 9 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 $128 \times 128$ 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).

Visualization of the final solution. In Figure 52, 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 51. 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 $L = 8$.

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 $2048 \times 2048$ uniform grid, i.e. using $L = 11$ levels. Figure 51 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.

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

$$L_1^t(Q) = L_1^t(\Delta x_L, \Omega_L) + \sum_{l=0}^{L-1} L_1^t(\Delta x_l, \Omega_l \setminus \Omega_{l+1}),$$
Figure 51. 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).

where

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

denotes the $L_1$-norm on the domain $\Omega$, and where $Q^r_{(i,j)}$ denotes the projection of the reference solution from the $2048 \times 2048$ uniform mesh down to the desired mesh with step size $\Delta x$.

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

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

where $Q^r_{(i,j)}$ 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

$$\bar{C} = \frac{1}{N_T} \sum_{n=1}^{N_T} C^n,$$

where $N_T$ is the number of performed time steps, and $C^n$ 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

\[ \Gamma_{MR} = \frac{CPU \, Time}{\sum_{n=1}^{N_t} L^n} \]

which denotes the average CPU time spent to evolve the solution in each cell of the computational domain, in each time step. Here \( \sum_{n=1}^{N_t} L^n \) 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

$$\Gamma_{AMR} = \frac{CPU\ Time}{\sum_{n=1}^{N_f} C_n}$$

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

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

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

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 10 and 11, 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 10 and 11.

Table 10. Summary of the results for the MR and the AMR computations.

<table>
<thead>
<tr>
<th>Level</th>
<th>FV</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$L_1^r(\rho)$</td>
<td>$L_1^r(\rho)$</td>
</tr>
<tr>
<td>L=8</td>
<td>3.65</td>
<td>3.68</td>
</tr>
<tr>
<td>L=9</td>
<td>2.23</td>
<td>2.26</td>
</tr>
<tr>
<td>L=10</td>
<td>1.04</td>
<td>1.08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level</th>
<th>FV</th>
<th>AMR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$L_1^r(\rho)$</td>
<td>$L_1^r(\rho)$</td>
</tr>
<tr>
<td>L=8</td>
<td>3.91</td>
<td>3.92</td>
</tr>
<tr>
<td>L=9</td>
<td>2.34</td>
<td>2.37</td>
</tr>
<tr>
<td>L=10</td>
<td>1.22</td>
<td>1.30</td>
</tr>
</tbody>
</table>

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 $128 \times 128$.

As expected from the previously results, 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 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 9 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.

Figures 53 and 54 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
Table 11. Summary of the results for MR/L T and AMR/L T computations

<table>
<thead>
<tr>
<th>Level</th>
<th>FV</th>
<th>MR/L T</th>
<th>AMR/L T</th>
</tr>
</thead>
</table>
|       | \( L_1(\rho) \) | \( L_1^1(\rho) \) | Compression (%) | Overhead per iteration (per leaf) (%) | CPU Memory Grid (%)
| L=8   | 3.65 | 3.74 | 29.36 | 39.07 | 21.70 | 0.32 | 8.0 | 10^{-2} | 10^{-2} |
| 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 |

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

Figure 53. 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.

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.

6.4. Turbulence modeling

The intrinsic multiscale character of turbulent flows in space and time is still a major challenge in computational fluid dynamics. The numerical simulation of fully developed turbulent flows requires turbulence models to reduce the computational complexity as direct approaches are still beyond the frontiers of the available computational resources and hence they will be limited to low Reynolds number flows in the near future. In the above context the compressible turbulent regime is even more challenging than the incompressible one, as in addition to the large range of involved temporal and spatial scales, shocks appear and interact nonlinearly with coherent vortices. Their accurate and efficient numerical simulation becomes more complex.
Different directions for turbulence modeling have been entered upon. The large eddy simulation approach seems to be the most advanced, however reliable simulation of turbulent flow still requires the tuning of ad hoc parameters [45].

In the recent past, multiscale methods for the simulation of turbulent flows have become more and more fashionable, also for compressible flows. A review can be found in the book of Sagaut and coworkers [59]. Wavelet methods in computational fluid dynamics have been reviewed in [61].

The Coherent Vortex Simulation (CVS), which fits into the framework of multiscale methods, has been introduced by Farge, Schneider and coworkers for modeling incompressible turbulent flows [27, 28]. The underlying idea is the decomposition of the flow into coherent and incoherent contributions by means of an orthogonal wavelet filtering of the vorticity field. The coherent flow is then computed deterministically, while the influence of the incoherent background flow is statistically modelled or neglected. In Okamoto et al. [49] it has been shown that for incompressible isotropic turbulence the number of degrees of freedom $N$ required for CVS grows slower with the Reynolds number $Re$, i.e. $N \propto Re^{3.9}$, than for DNS where Kolmogorov type arguments imply $N \propto Re^{4.5}$. This motivates the development of CVS for computing fully developed turbulent flows. Adaptive space discretizations are hereby a key ingredient to be able to benefit from the efficient representation of the coherent flow to be computed in terms of memory and CPU time savings.

Adaptive multiresolution schemes have been developed in the past with the subject to speed up classical numerical discretization methods for PDEs, like finite differences or finite volumes, by locally adapting the grid to the solution which can then result in significant memory reduction. Therewith the costly evaluation of the nonlinear terms can also be reduced without degrading the precision of the solution. Three ingredients of multiresolution analysis are hereby essential: estimation of the local regularity of the solution, thresholding of weak coefficients in the multiresolution representation of the solution and fast interpolation of the solution from the locally refined grid to locally equidistant grids. A comprehensible overview on this rapidly growing topic is given in the book of Müller [47].

The aim of the present section is to present the extension of the CVS method to subsonic compressible turbulent flows using the adaptive multiresolution algorithm originally developed by Roussel and coworkers [57]. The formalism of CVS is adapted to the compressible Navier-Stokes equations which are written in primitive variables. The influence of the wavelet filtering of the conserved quantities is investigated and the choice of the filter tolerance and the normalization of the wavelets is studied.
6.4.1. Navier–Stokes equations for compressible fluid flows

We consider a three-dimensional compressible flow of a Newtonian fluid in the Stokes hypothesis in a domain $\Omega \subset \mathbb{R}^3$. Using Einstein’s summation convention, the balance equations in Cartesian coordinates can be written in the following dimensionless form, see e.g. [37],

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x_j} (\rho u_j),$$

$$\frac{\partial}{\partial t} (\rho u_i) = -\frac{\partial}{\partial x_j} (\rho u_i u_j + p \delta_{i,j} - \tau_{i,j}),$$

$$\frac{\partial}{\partial t} (\rho e) = -\frac{\partial}{\partial x_j} \left( (\rho e + p) u_j - u_i \tau_{i,j} - \lambda \frac{\partial T}{\partial x_j} \right) \tag{50}$$

In the above equations, $\rho$, $p$, $T$ and $e$ denote the dimensionless density, pressure, temperature and specific total energy per unit of mass, respectively; $(u_1, u_2, u_3)^T$ is the dimensionless velocity vector. The components of the dimensionless viscous strain tensor $\tau_{i,j}$ are

$$\tau_{i,j} = \frac{\mu}{Re} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{i,j} \right), \tag{51}$$

where $\mu$ denotes the dimensionless molecular viscosity and $Re$ the Reynolds number. The dimensionless conductivity $\lambda$ is defined by

$$\lambda = \frac{\mu}{(\gamma - 1) Ma^2 Re Pr}, \tag{52}$$

where $\gamma$, $Ma$ and $Pr$ respectively denote the specific heat ratio and the Mach and Prandtl numbers.

The system is completed by an equation of state for a calorically ideal gas

$$p = \frac{\rho T}{\gamma Ma^2}. \tag{53}$$

and suitable initial and boundary conditions.

Assuming the temperature to be larger than 120 K, the molecular viscosity varies with the temperature according to the dimensionless Sutherland law

$$\mu = T_s^\frac{\beta}{\theta} \left( \frac{1}{T} + T_s \right) \tag{54}$$

where $T_s \approx 0.404$.

Denoting by $(x, y, z)$ the three Cartesian directions, this system of equations can be written in the following compact form

$$\frac{\partial U}{\partial t} = -\frac{\partial F}{\partial x} - \frac{\partial G}{\partial y} - \frac{\partial H}{\partial z} \tag{55}$$

where $U = (\rho, \rho u_1, \rho u_2, \rho u_3, \rho e)^T$ denotes the vector of the conservative quantities, and $F$, $G$, $H$ are the flux vectors in the directions $x$, $y$, and $z$, respectively.

6.4.2. Time discretization

The time evolution is performed on the leaves of the tree only. The cell-average values in the other nodes are obtained by projection from the leaves. In order to avoid numerical diffusion, we use an explicit 2-4 Mac Cormack scheme, which is second-order accurate in time, fourth-order in space for the convective terms, and second-order in space for the diffusive terms (see, Gottlieb and Turkel [33]). The time integration is decomposed into two stages, the first stage being decentered in one direction, the second one in the other direction. To avoid
an accumulation of error in one direction, we alternate the directions at each time step. One gets the following scheme to compute the cell averages $\bar{U}^{n+1}$ at time $t^{n+1}$ from the given cell averages $\bar{U}^n$ at time $t^n$

$$
\bar{U}_{l,i,j,k}^{n+1} = \bar{U}_{l,i,j,k}^n + \Delta t \left( \frac{-7 \bar{F}_{l,i,j,k}^n + 8 \bar{F}_{l,i+1,j,k}^n - \bar{F}_{l,i+2,j,k}^n}{6\Delta x_l} \right) + \Delta t \left( \frac{-7 \bar{G}_{l,i,j,k}^n + 8 \bar{G}_{l,i,j+1,k}^n - \bar{G}_{l,i,j+2,k}^n}{6\Delta y_l} \right) + \Delta t \left( \frac{-7 \bar{H}_{l,i,j,k}^n + 8 \bar{H}_{l,i,j+1,k}^n - \bar{H}_{l,i,j+2,k}^n}{6\Delta z_l} \right)
$$

(56)

$$
\bar{U}_{l,i,j,k}^n = \bar{U}_{l,i,j,k}^{n+1} = \frac{\bar{U}_{l,i,j,k}^n + \bar{U}_{l+1,i,j,k}^n}{2} + \Delta t \left( \frac{-7 \bar{F}_{l,i,j,k}^n + 8 \bar{F}_{l,i-1,j,k}^n - \bar{F}_{l,i-2,j,k}^n}{6\Delta x_l} \right) + \Delta t \left( \frac{-7 \bar{G}_{l,i,j,k}^n + 8 \bar{G}_{l,i,j-1,k}^n - \bar{G}_{l,i,j-2,k}^n}{6\Delta y_l} \right) + \Delta t \left( \frac{-7 \bar{H}_{l,i,j,k}^n + 8 \bar{H}_{l,i,j+1,k}^n - \bar{H}_{l,i,j+2,k}^n}{6\Delta z_l} \right)
$$

(57)

where $\Delta x_l = 2^{-l} L_x$, $\Delta y_l = 2^{-l} L_y$, and $\Delta z_l = 2^{-l} L_z$ denote the space steps at a level $l$ in the three directions. Here $L_x$, $L_y$, and $L_z$, denote the lengths of the computational domain in the $x$, $y$, and $z$ directions, respectively.

For the computation of the diffusive terms, we do not need to use a decentered scheme. Here the diffusive terms are approximated the same way as if we were using a second-order Runge-Kutta-Heun method in time, together with a second-order centered scheme in space.

6.4.3. Principle of Coherent Vortex Simulation

The CVS of incompressible turbulent flows is based on the decomposition of the vorticity $\omega = \nabla \times u$ into coherent and incoherent parts using thresholding of the wavelet coefficients. The corresponding coherent and incoherent velocity fields are then obtained by applying Biot-Savart’s kernel. In the CVS method, the evolution of the coherent flow is then computed deterministically in a dynamically adapted wavelet basis and the influence of the incoherent components is statistically modelled or simply neglected [27, 28, 60].

For compressible flows, the situation is different, since both vortical and potential components are present. Here we decided to decompose the conservative variables $U = (\rho, \rho u_1, \rho u_2, \rho u_3, \rho e)$ into a biorthogonal wavelet series by applying the cell-average multiresolution transform previously described. In [56], we applied the same biorthogonal wavelet decomposition to incompressible isotropic turbulence to decompose the flow into coherent and incoherent parts. We concluded that the biorthogonal decomposition yield reasonable results compared to the orthogonal wavelet decomposition typically used for CVS, which justifies the application of the former to compressible turbulence. A decomposition of the conservative variables into coherent and incoherent components is then obtained by decomposing the conservative variables into wavelet coefficients, applying a thresholding and reconstructing the coherent and incoherent contributions from the strong and weak coefficients, respectively.

First the dimensionless density and pressure are decomposed into

$$
\rho = \rho_C + \rho_I, \quad p = p_C + p_I,
$$

(58)

where $\rho_C$ and $p_C$ respectively denote the coherent part of the density and pressure fields, while $\rho_I$ and $p_I$ denote the corresponding incoherent parts.
Then the other remaining variables, i.e. the velocity components $u_1$, $u_2$, $u_3$, the temperature $T$ and energy $e$, are decomposed using the Favre averaging technique [29], i.e. density weighted, as done in RANS and LES of compressible flows to simplify modeling [45]. This technique avoids the introduction of unresolved terms into the equation of mass conservation and trilinear terms into the momentum equation. For a quantity $\varphi$ corresponding to one of these remaining variables, we obtain the following decomposition,

$$\varphi = \varphi_C + \varphi_I,$$

where $\varphi_C = \frac{(\rho \varphi)_C}{(\rho)_C}$.

For sake of clarity, we use the notation $\varphi_C = \bar{\varphi}$. Retaining only the coherent contributions of the conservative variables we obtain the filtered compressible Navier-Stokes equations which describe the flow evolution of the coherent flow

$$\frac{\partial \bar{\rho}}{\partial t} = -\frac{\partial}{\partial x_j}(\bar{\rho} \bar{u}_j)$$

$$\frac{\partial}{\partial t}(\bar{\rho} \bar{u}_i) = -\frac{\partial}{\partial x_j}(\bar{\rho} \bar{u}_i \bar{u}_j + \bar{\rho} \delta_{i,j} - \bar{\tau}_{i,j} + A_{i,j})$$

$$\frac{\partial}{\partial t}(\bar{\rho} \bar{e}) = -\frac{\partial}{\partial x_j}\left((\bar{\rho} \bar{e} + \bar{p}) \bar{u}_j - \bar{u}_i \bar{\tau}_{i,j} - \lambda \frac{\partial \bar{T}}{\partial x_j} + B_j\right)$$

$$\bar{p} = \frac{\bar{\rho} \bar{T}}{\gamma M a^2}$$

where the terms

$$A_{i,j} = \bar{\rho} (\bar{u}_i \bar{u}_j - \bar{u}_i \bar{u}_j) - \bar{\tau}_{i,j} + \bar{\tau}_{i,j}$$

$$B_j = \bar{\rho} (\bar{u}_j \bar{e} - \bar{u}_j \bar{e}) + \bar{u}_j \bar{p} - \bar{u}_j \bar{p} - \bar{u}_j \bar{\tau}_{i,j} + \bar{u}_j \bar{\tau}_{i,j} - \lambda \frac{\partial \bar{T}}{\partial x_j} + \lambda \frac{\partial \bar{T}}{\partial x_j}$$

constitute the influence of the incoherent contributions. In the current paper, these terms are completely negleted. We also use the approximation $\bar{\rho} = \mu(\bar{T})$ and $\bar{\lambda} = \lambda(\bar{T})$. Note that modeling of the incoherent contributions, similar to LES approaches, has been proposed in [32], which allows for larger choices of the threshold and thus higher compression rates. A review of the different modeling strategies can be found in [61].

6.4.4. Choice of the thresholding rule

In the CVS of incompressible turbulence, the vorticity field is decomposed into orthogonal wavelets and filtered using a fixed threshold $\epsilon_0$ based on the enstrophy, i.e. on the $L^2$-norm of vorticity.

In most papers on adaptive multiresolution methods [12,15,47,57], the conserved quantities are filtered using furthermore biorthogonal wavelets which are normalized with respect to the $L^1$-norm, which is well adapted to the finite volume discretization, i.e. $\| \psi_{\mu,i,j,k} \|_{L^1} = 1$. In the adaptive multiresolution method (see e.g. [57]) typically detail coefficients $d_{\mu,i,j,k} = (u, \psi_{\mu,i,j,k})$ of a conserved quantity are removed if

$$| d_{\mu,i,j,k} | < 2^{d(L-L)} \epsilon$$

where $d$ denotes the space dimension (here $d = 3$), $\epsilon$ the threshold, and $L$ is the finest level present in the computations. The above choice of the threshold takes the $L^1$-normalization of the wavelets into account. The computations using this $L^1$ based thresholding rule are denoted by "Norm #1" results.

To investigate the influence of the thresholding rule and to compare with the classical CVS scheme which uses $L^2$-normalization of the wavelets and filters vorticity, which corresponds to the anti-symmetric part of the velocity gradient tensor, we consider two additional thresholding rules:
• one using the $L^2$-normalization of the wavelets for which we obtain the thresholding rule

$$
| \hat{\mu}_{l,i,j,k}^\mu | < 2^{d(l-L)/2} \epsilon
$$

(63)
denoted in the following "Norm #2" results,

• and one based on the $H^1$-normalization of the wavelets for which we obtain the thresholding rule

$$
| \hat{\mu}_{l,i,j,k}^\mu | < 2^{(d-2)(l-L)/2} \epsilon
$$

(64)
denoted in the following "Norm #3" results. The $H^1$ thresholding rule is motivated by the fact that here the filtering is adapted to the gradients, which mimics filtering of vorticity. However in addition to vortical regions, strain-dominated regions are also retained in the coherent velocity field.

To apply the threshold to the vector valued function $U$ of the dimensionless conservative quantities, we divide each component of the detail vector by the mean-value of the corresponding dimensionless quantity in the whole computational domain. Concerning the velocity vector $(u_1, u_2, u_3)$, to avoid anisotropy in the threshold of the details, we compute only one detail for the velocity

$$
\hat{d}_{l,i,j,k}^\mu (u) = \sqrt{\left( \hat{d}_{l,i,j,k}^{\mu,1} \right)^2 (u_1) + \left( \hat{d}_{l,i,j,k}^{\mu,2} \right)^2 (u_2) + \left( \hat{d}_{l,i,j,k}^{\mu,3} \right)^2 (u_3)}
$$

and we divide this component of the detail vector by the mean-value of the velocity in the whole computational domain.

Details of all components of $U$ are removed only if all components are below the given threshold. The incoherent part $U_I$ corresponds hence to the conservative quantities for which, for a given index $\{l, i, j, k, \mu\}$, all details are below the threshold. The coherent part $U_C$ corresponds to the remainder. This choice implies that only one tree data structure is necessary to store all coherent components of the conserved variables. By construction we have $U = U_C + U_I$, however the sum is not orthogonal due to the use of biorthogonal wavelets instead of orthogonal ones (see e.g. [56] for a discussion on this topic).

6.4.5. Summary of the CVS algorithm

In the following, we briefly summarize the CVS algorithm. For more details on the general adaptive MR algorithm, we refer to [57]. First, depending on the initial condition, an initial graded tree is created. Then, given the graded tree structure, a time evolution is made on the leaves. Finally, details are computed by multiresolution transform, in order to remesh the tree.

Denoting by $\hat{E}(\Delta t)$ the discrete time evolution operator, the global algorithm can schematically be summarized by

$$
\bar{U}^{n+1} = \bar{M}^{-1} \cdot \bar{T}(\epsilon) \cdot \bar{M} \cdot \bar{E}(\Delta t) \cdot \bar{U}^n
$$

(65)

where $\bar{M}$ is the multiresolution transform operator, and $\bar{T}(\epsilon)$ is the threshold operator with tolerance $\epsilon$. The threshold operator depends on the choice of the thresholding rule.

6.4.6. Application to weakly compressible mixing layers

As test case for the CVS method we consider in the following a three-dimensional weakly compressible temporally developing turbulent mixing layer. For a complete description of this test-case in the LES framework, we refer the reader to [3, 68].

The CVS results are compared with a DNS reference computation, using a finite volume scheme on the finest regular grid. We study the impact of the different thresholding rules, the choice of the threshold, and the influence of the Reynolds number to assess the precision and efficiency of CVS.
6.4.7. Flow configuration of the mixing layer

We initialize the test-case by setting two layers of a fluid stacked one upon the other one, each of them with the same velocity norm but opposed directions.

\[
\begin{align*}
\text{Figure 55. Flow configuration: domain and initial basic flow } \mathbf{u}_0 \text{ of the three-dimensional mixing layer.}
\end{align*}
\]

For every computation with \( L = 7 \) scales, the computational domain is a three-dimensional cube \( \Omega = [-30, 30]^3 \) with sidelingth \( L_X = L_Y = L_Z = 60 \), and the final time of all computations corresponds to \( t = 80 \). We set periodic boundary conditions for the \( x - \) and \( y - \)direction and Neumann conditions are imposed in the \( z - \)direction, i.e. on the top and bottom boundaries. The Prandtl and Mach numbers are set to 0.71 and 0.3 respectively, whereas the specific heat ratio \( \gamma \) equals 1.4. The CFL number is set to 0.4 and the maximal resolution is \( 128^3 \), which corresponds to \( L = 7 \) scales. Then, in section 4.5, \( L = 8 \) scales are considered, i.e. a maximal resolution of \( 256^3 \).

In this test-case, the initial dimensionless velocity profile is

\[
\mathbf{u}_0(x) = \begin{pmatrix} \tanh(z) \\ 0 \\ 0 \end{pmatrix}
\]

and the initial dimensionless density and temperature profiles are constant and set to 1.

A sinusoidal perturbation of the form

\[
u'(x) = A(z) \left( \frac{1}{2} \cos \frac{2\pi(x+y)}{L} + \frac{1}{2} \cos \frac{2\pi(x-y)}{L} + \sum_{k=1}^{b-1} \cos \frac{2k+1\pi x}{L} \right)
\]

is superimposed to the initial velocity in the streamwise direction, where

\[
A(z) = \frac{1}{b \cosh^2(z)}
\]

is the amplitude of the perturbation, \( L = 60 \) is the length of the computational domain, and \( b \) denotes the maximal number of modes used for the computation, \( b = 3 \) in this case. The \( \cosh^2(z) \) term is added to limit
the perturbation to the junction of the two stacked-up fluid layers. This way, we spare unnecessary waste of CPU resources. Here, we have added only one oblique mode disturbance, so that we can distinguish more easily the structures. The same amount of intensity is given to each of the three modes.

6.4.8. Time evolution of a mixing layer

For the DNS reference computation, we use the finite volume scheme on the regular grid with resolution $128^3$. The Reynolds number based on the initial velocity and half the initial layer thickness is set to $Re = 100$. Figure 56 shows the time evolution of the mixing layer. At $t \approx 19$, the Kelvin-Helmholtz instability generates four rollers in the streamwise direction (Figure 56, left side). The number of vortices at the initial stage depends on the number of modes $b$ used in the initial perturbation and of the size of the domain $\Omega$. At the beginning, the mixing layer remains approximately two-dimensional, since we have introduced only one oblique mode. Later on, the vortices begin to pair so that, at $t \approx 37$, we observe two vortex pairings (Figure 56, center). At $t \approx 78$, these two pairings are finished and three-dimensional structures appear, generated by the oblique mode between the two remaining vortices (Figure 56, right side). For longer computational times, assuming that the domain is large enough, these two vortices would pair again, thus leading to only one vortex.

6.4.9. Choice of the thresholding rule

In this section, we perform a series of CVS computations for $Re = 100$ using the three different thresholding rules (Norms #1, #2 and #3) previously presented, and compare the results with the reference DNS computation on the finest grid, using the same numerical scheme. For each thresholding rule, we perform first several computations with different tolerances to determine its optimal value $\epsilon_{opt}$. To assess which thresholding rule yields the best results, we compare then the energy spectra, the energy and enstrophy evolutions, as well as the corresponding computational resources (CPU time and memory) spent. Once we have the optimal value for each norm, we compare those optimal computations and decide which norm shows better results.

6.4.10. Results using the $L^1$-normalization (Norm #1)

In Table 12, we show the results of the CVS computations for the Norm #1 together with the DNS reference run. We observe that the CVS computations require much less CPU time (between 26 and 40 %) and memory (between 26 and 43 %) than DNS, except for a very small value of $\epsilon = 0.01$. For the latter we see that the results of CVS are almost identical with the DNS run, as energy and enstrophy is almost perfectly retained. Memory is still reduced (< 60%) but the CPU time is almost doubled with respect to DNS. The time evolution of energy shown in Figure 57 (left side) illustrates that, for all considered values of $\epsilon$, the CVS result matches well with the DNS reference solution. For the evolution of the enstrophy (Fig. 57, right side), we see however that the dynamics differs from the DNS reference, which shows that the numerical dissipation of the CVS is not sufficient.

The spectral distribution of energy in the streamwise direction is given in Fig. 58. For $\epsilon = 0.2$ and 0.1, the CVS results agree well with the DNS reference in the large scales, but still differ in the small scales.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\epsilon$</th>
<th>CPU time</th>
<th>% CPU</th>
<th>% Mem</th>
<th>% $E$</th>
<th>% $Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td></td>
<td>7d 6h</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>MR 0.3</td>
<td></td>
<td>2d 2h</td>
<td>28.74 %</td>
<td>25.96 %</td>
<td>99.83 %</td>
<td>83.19 %</td>
</tr>
<tr>
<td>MR 0.25</td>
<td></td>
<td>2d 14h</td>
<td>35.63 %</td>
<td>29.48 %</td>
<td>99.92 %</td>
<td>83.80 %</td>
</tr>
<tr>
<td>MR 0.2</td>
<td></td>
<td>2d 15h</td>
<td>36.21 %</td>
<td>33.44 %</td>
<td>99.92 %</td>
<td>84.85 %</td>
</tr>
<tr>
<td>MR 0.1</td>
<td></td>
<td>3d 4h</td>
<td>43.68 %</td>
<td>42.91 %</td>
<td>99.89 %</td>
<td>94.18 %</td>
</tr>
<tr>
<td>MR 0.01</td>
<td></td>
<td>13d 20h</td>
<td>190.80 %</td>
<td>75.93 %</td>
<td>99.98 %</td>
<td>99.99 %</td>
</tr>
</tbody>
</table>

Table 12. Results obtained with the Norm #1. Percentages of CPU time (required on a Pentium IV, 2.5 GHz), memory compression, and errors in comparison with the DNS solution for kinetic energy $E$ and enstrophy $Z$ at $t = 80$ with different $\epsilon$ and for $N = 128^3$. 

<table>
<thead>
<tr>
<th>Method</th>
<th>$\epsilon$</th>
<th>CPU time</th>
<th>% CPU</th>
<th>% Mem</th>
<th>% $E$</th>
<th>% $Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td></td>
<td>7d 6h</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>MR 0.3</td>
<td></td>
<td>2d 2h</td>
<td>28.74 %</td>
<td>25.96 %</td>
<td>99.83 %</td>
<td>83.19 %</td>
</tr>
<tr>
<td>MR 0.25</td>
<td></td>
<td>2d 14h</td>
<td>35.63 %</td>
<td>29.48 %</td>
<td>99.92 %</td>
<td>83.80 %</td>
</tr>
<tr>
<td>MR 0.2</td>
<td></td>
<td>2d 15h</td>
<td>36.21 %</td>
<td>33.44 %</td>
<td>99.92 %</td>
<td>84.85 %</td>
</tr>
<tr>
<td>MR 0.1</td>
<td></td>
<td>3d 4h</td>
<td>43.68 %</td>
<td>42.91 %</td>
<td>99.89 %</td>
<td>94.18 %</td>
</tr>
<tr>
<td>MR 0.01</td>
<td></td>
<td>13d 20h</td>
<td>190.80 %</td>
<td>75.93 %</td>
<td>99.98 %</td>
<td>99.99 %</td>
</tr>
</tbody>
</table>

Table 12. Results obtained with the Norm #1. Percentages of CPU time (required on a Pentium IV, 2.5 GHz), memory compression, and errors in comparison with the DNS solution for kinetic energy $E$ and enstrophy $Z$ at $t = 80$ with different $\epsilon$ and for $N = 128^3$. 

The spectral distribution of energy in the streamwise direction is given in Fig. 58. For $\epsilon = 0.2$ and 0.1, the CVS results agree well with the DNS reference in the large scales, but still differ in the small scales.
Figure 56. Time evolution of a weakly compressible mixing layer for $Re = 100$. First row: isolines of vorticity in the plane $y = 0$ for the DNS solution. Second row: same isolines for the CVS computation with $\epsilon = 0.03$ and Norm #3. Third row: Corresponding isosurfaces of vorticity $||\omega|| = 0.5$ (black) and $||\omega|| = 0.25$ (gray). Fourth row: Adaptive meshes for the CVS computation. The time instants are $t = 19$ (left), $t = 37$ (center) and $t = 78$ (right).
6.4.11. Results using the $L^2$-normalization (Norm #2)

Table 13 shows the results of the CVS computations for the Norm #2, together with the DNS reference computation. The CVS computations exhibit a significant reduction of CPU time and memory requirements with respect to the reference DNS computation, while the final kinetic energy and enstrophy are reasonably well predicted.

Figure 59 shows the time evolution of the kinetic energy and the enstrophy for the Norm #2. The evolution of the enstrophy for the CVS computations (Fig. 59, right side) follows well the DNS reference computation.
Table 13. Results obtained with the Norm #2. Percentages of CPU time (required on a Pentium IV, 2.5 GHz), memory compression, and errors in comparison with the DNS solution for the kinetic energy \( E \) and the enstrophy \( Z \) at \( t = 80 \) with different \( \varepsilon \) and for \( N = 128^3 \).

<table>
<thead>
<tr>
<th>Method</th>
<th>( \varepsilon )</th>
<th>CPU time</th>
<th>% CPU</th>
<th>% Mem</th>
<th>% ( E )</th>
<th>% ( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td></td>
<td>7d 6h</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>MR</td>
<td>0.5</td>
<td>7h 57 min</td>
<td>4.58 %</td>
<td>5.44 %</td>
<td>97.84 %</td>
<td>89.35 %</td>
</tr>
<tr>
<td>MR</td>
<td>0.08</td>
<td>2d 18h</td>
<td>37.93 %</td>
<td>30.55 %</td>
<td>99.96 %</td>
<td>88.17 %</td>
</tr>
<tr>
<td>MR</td>
<td>0.07</td>
<td>3d 5h</td>
<td>44.25 %</td>
<td>32.73 %</td>
<td>99.90 %</td>
<td>92.46 %</td>
</tr>
</tbody>
</table>

Figure 59. Results obtained with the Norm #2. Kinetic energy (left), and enstrophy (right) of the DNS and several tolerance values for the CVS computations.

until \( t \approx 30 \). For longer times, we observe a significant error growth. In particular for \( \varepsilon = 0.08 \), the CVS under predicts the enstrophy, the difference with respect to DNS becoming larger than 7\% at \( t = 80 \). The evolution of the kinetic energy (Fig. 59, left side) presents, as for the Norm #1 results, a good agreement with the reference DNS computation, whatever the tolerance.

Figure 60. Results obtained with the Norm #2. Energy spectra in the streamwise direction at \( t = 80 \) for \( \varepsilon = 0.08 \) (left) and \( \varepsilon = 0.07 \) (right).

Concerning the final streamwise energy spectra (Figure 60), it is shown that the agreement between the CVS computations and the reference DNS is significantly improved when compared to the results of the Norm #1. For the large scales, i.e. when \( k < 6 \), the CVS spectra almost coincide with the DNS. However, for \( k > 6 \), the differences becomes visible.
Still, the slight difference between CVS and DNS in the large scales \((k < 6)\) involves a larger energy loss than the apparently larger energy difference in the smaller ones \((k > 6)\), because of the logarithmically scaled plot. As we can observe in Table 13, the CVS with \(\varepsilon = 0.08\) and the CVS with \(\varepsilon = 0.07\) yield respectively relative errors on energy \(e_E\) of 0.0004 and 0.0011. Norm #2 presents better results than the Norm #1, but the results on enstrophy are not accurate enough.

6.4.12. Results using the \(H^1\)-normalization (Norm #3)

Figure 61. Results obtained with the Norm #3. Time evolution of the kinetic energy (left) and the enstrophy (right) for the DNS and CVS computations using different tolerance values.

Figure 61 shows that CVS computations based on the \(H^1\)-normalization yield better results for the enstrophy and follow well the dynamics of the DNS computation. Hereby the difference between DNS and CVS decreases for decreasing threshold values and for \(\varepsilon = 0.03\) we find the optimal agreement for the largest compression rate. These observations are confirmed in Table 14.

Figure 62 shows the final streamwise energy spectra for DNS and CVS computations with different tolerance values. Again the agreement between CVS and DNS improves for decreasing tolerances. The CVS computation with \(\varepsilon = 0.03\) seems to be the optimal choice, since both DNS and CVS energy spectra almost coincide everywhere, whatever the wavenumber.

<table>
<thead>
<tr>
<th>Method</th>
<th>(\varepsilon)</th>
<th>CPU time</th>
<th>% CPU</th>
<th>% Mem</th>
<th>% E</th>
<th>% Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td></td>
<td>7d 6h</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>MR 0.06</td>
<td>2d 0h</td>
<td>27.59 %</td>
<td>24.23 %</td>
<td>99.93 %</td>
<td>90.60 %</td>
<td></td>
</tr>
<tr>
<td>MR 0.05</td>
<td>2d 2h</td>
<td>28.74 %</td>
<td>27.46 %</td>
<td>99.69 %</td>
<td>92.81 %</td>
<td></td>
</tr>
<tr>
<td>MR 0.04</td>
<td>2d 10h</td>
<td>29.89 %</td>
<td>30.80 %</td>
<td>99.80 %</td>
<td>96.12 %</td>
<td></td>
</tr>
<tr>
<td>MR 0.03</td>
<td>2d 8h</td>
<td>32.18 %</td>
<td>34.54 %</td>
<td>99.88 %</td>
<td>98.66 %</td>
<td></td>
</tr>
<tr>
<td>MR 0.01</td>
<td>4d 6h</td>
<td>58.62 %</td>
<td>48.20 %</td>
<td>99.95 %</td>
<td>99.91 %</td>
<td></td>
</tr>
</tbody>
</table>

Table 14. Results obtained with the Norm #3. Percentages of CPU time (required on a Pentium IV, 2.5 GHz), memory compression, and errors in comparison with the DNS solution for the kinetic energy \(E\) and enstrophy \(Z\) at \(t = 80\) with different \(\varepsilon\), \(N = 128^3\).

Table 14 shows the efficiency and accuracy of the CVS computations with the Norm #3. We observe that only 34.54% of the memory requirements of the DNS are used to reproduce about 99.88% of the energy and 98.66% of the enstrophy of the DNS computation. For this computation, less than one third of the CPU time of the DNS is required, which means that the CVS is more than three times faster than the DNS.

The vorticity fields in Fig. 56 confirm that we have a very good agreement between DNS and CVS computation with Norm #3 and \(\varepsilon = 0.03\). The last row of the figure presents the adaptive mesh at three different times and
illustrates that the mesh dynamically adapts to the flow evolution and concentrates points in the shear layer, without any heuristical refinement criterion.

Concerning the DNS computation, it is shown that all the scales of the turbulent flow are well resolved. In fact, the energy \( 10^7 \) times lower than the largest energy is reached at the wavenumber \( k \approx 20 \). This value is much lower than the largest admissible wavenumber \( k = 64 \), when using a resolution \( 128^3 \).

6.4.13. Influence of the incoherent part of the CVS computation (Norm #3)

In this part, we compute the energy spectrum of the discarded part of the CVS computation at the final time. We take the difference between the DNS and CVS computations at \( t = 80 \). We compare this result with the DNS data where we applied the coherent vortex extraction (CVE) of the final solution using the same wavelet filtering (Figure 63).

We observe a good agreement between the filtered DNS data (Fig. 10, right) and the CVS computation (Fig. 10, left). The incoherent contributions are strongly reduced at all scales and they exhibit a \( k^{-2} \) decay behavior which corresponds to an equipartition of enstrophy, i.e. the incoherent vorticity is decorrelated in physical space. The probability distribution functions of the total, coherent and incoherent velocities, the latter obtained either from the velocity difference between the DNS and CVS computations or the velocity difference between the DNS and the CVE filtered DNS solution, are shown in Fig. 11. The results confirm the strongly reduced variance of the incoherent contributions with respect to both, the total and coherent velocities and show that the high order statistics of the CVS computation are in good agreement with the DNS. Furthermore we find that the incoherent velocity exhibits a Gaussian-like distribution. Thus neglecting incoherent contributions during the flow evolution could justify the modeling of turbulent dissipation.

In this section, we study the influence of the Reynolds number. In addition to the previous case with $Re = 100$, we perform computations for $Re = 50$ and 200. For every CVS computation, we use the Norm #3 and the optimal tolerance $\epsilon_{opt} = 0.03$.

6.4.15. Results with $Re = 50$

The plots in Fig. 65 show the more dissipative behaviour of the flow compared to the case with $Re = 100$ (see Fig. 56). The two first rows of Fig. 65 are two-dimensional isolines of vorticity in the plane $y = 0$ for both DNS and CVS computations. They confirm that the solutions agree as well as in the case $Re = 100$. At the bottom of Fig. 65, the evolution of the adaptive mesh for the CVS computation is plotted, which confirms the efficient self-adaptive behaviour of the multiresolution algorithm.
Figure 65. Time evolution of a weakly compressible mixing layer for $Re = 50$. First row: isolines of vorticity in the plane $y = 0$ for the DNS solution. Second row: same isolines for the CVS computation with $\epsilon = 0.03$ and Norm #3. Third row: Corresponding isosurfaces of vorticity of the CVS $||\omega|| = 0.5$ (black) and $||\omega|| = 0.25$ (gray). Fourth row: Adaptive mesh of the CVS computation. The time instants are $t = 19$ (left), $t = 37$ (center) and $t = 78$ (right).
Table 15 gives the performance of the CVS computation in comparison with the DNS one. The percentages are very similar to the ones obtained for $Re = 100$ using the optimal tolerance (see Table 14), which shows the robustness of the choice of the tolerance. The energy spectra obtained with CVS agrees well for every scale with the reference DNS (Figure 66, left side).

The time evolution of the kinetic energy and the enstrophy (Figure 66, center and right side) shows likewise a good agreement between CVS and DNS.

6.4.16. $Re = 200$

Compared with the case $Re = 100$ (Fig. 56), we observe that the structures are less dissipated for $Re = 200$ and that three-dimensional effects are, as expected, more intense (Fig. 67). The isolines of the two-dimensional vorticity in the plane $y = 0$ (Fig. 67) look identical for CVS and DNS computations, except a slight difference for the left vortex at $t = 78$.

Table 16. Percentages of CPU time, memory compression, and errors for $E$ and $Z$, in comparison with the DNS solution, $N^3 = 128^3$, $Re = 200$.

The results given in Table 16 for the percentages of CPU time, memory, energy and enstrophy are also very close to the ones computed with $Re = 100$ (see Table 14) using the same tolerance $\epsilon = 0.03$.

The final energy spectra in the streamwise direction obtained by CVS and DNS methods are shown in Fig. 68. They are in good accordance for almost every wavenumber. However, in the smallest scales ($k > 10$), the energy
Figure 67. Time evolution of a weakly compressible mixing layer for $Re = 200$. First row: Isolines of vorticity in the plane $y = 0$ for the reference DNS solution. Second row: same isolines for the CVS computation with $\epsilon = 0.03$ and norm #3. Third row: Corresponding isosurfaces of vorticity $||\omega|| = 0.5$ (black) and $||\omega|| = 0.25$ (gray) for the CVS computation. Fourth row: Adaptive mesh for the CVS computation. The time instants are $t = 19$ (left), $t = 37$ (center) and $t = 78$ (right).
spectra loses accuracy. Despite this, the CVS computation conserves 99.82% of the energy and 98.24% of the enstrophy of the DNS, which satisfies our expectations.

The time evolution of energy and enstrophy (Fig. 68, center and right side) confirm the good agreement between the CVS and DNS computations, keeping in mind that CVS requires around one third of the CPU time and the memory required by the DNS (Table 16).

The time evolution of energy and enstrophy (Fig. 68, center and right side) confirm the good agreement between the CVS and DNS computations, keeping in mind that CVS requires around one third of the CPU time and the memory required by the DNS (Table 16).

**Figure 68.** Energy spectra in the streamwise direction at $t = 80$ (left). Time evolution of the kinetic energy (center) and enstrophy (right) for the CVS and DNS computations, $N^3 = 128^3$, $Re = 200$.

**6.4.17. Comparison of energy spectra for different Reynolds numbers**

In order to assess the influence of the Reynolds number on the streamwise energy spectra, we compare computations for three Reynolds numbers (Fig. 69). In every case we find a good agreement between the CVS and DNS computations for the whole range of wavenumbers. For increasing Reynolds, we also see that the range of active wavenumbers increases and that the CVS is able to reproduce well the small scale behaviour.

**Figure 69.** DNS and CVS energy spectra at $t = 80$ for $Re = 50$ (left), $Re = 100$ (center) and $Re = 200$ (right).

**6.4.18. Dependence of the relative error on final energy and enstrophy with the Reynolds number**

Additional computations for $Re = 75$, $Re = 125$, $Re = 150$ and $Re = 175$ have been performed, to check if the relative error of energy and enstrophy for CVS compared to DNS shows a Reynolds number dependence or remains constant. The energy and enstrophy values at $t = 80$ vs the Reynolds number are plotted in Figure 70, for both DNS and CVS computations. First, we observe that both energy and enstrophy increase with $Re$ and that the gap between CVS and DNS computations slightly increases.
However, we find that, for the kinetic energy, the relative error yields $e_E = 1.797 \cdot 10^{-3}$ for $Re = 50$, whereas, for $Re = 200$, we have $e_E = 1.722 \cdot 10^{-3}$. This means that the relative error remains almost constant. For the enstrophy, similar observations can be made, i.e. for $Re = 50$ the relative error of enstrophy is $e_Z = 1.397 \cdot 10^{-2}$ and for $Re = 200$ we have $e_Z = 1.764 \cdot 10^{-2}$. This shows that the relative error on enstrophy slightly increases, but is still of the same order of magnitude. We conclude that the relative errors on energy and enstrophy remain stable for the investigated range of Reynolds numbers.

6.4.19. Time evolution of a mixing layer computed at resolution $256^3$

Finally we present a CVS computation with norm #3 and $\epsilon = 0.03$ at resolution $N = 256^3$. The domain is now set to $[-60, 60]^3$, so that the smallest space step is the same as in the case $128^3$. As before, the Reynolds number is set to $Re = 200$, and the final time corresponds to $t = 80$.

Since the method has previously been validated for the same smallest space step, and since the DNS computation would require more than two months, we only perform the CVS computation here. The goal of this test-case is not to validate the method, but to show that larger CPU time and memory compressions can be reached with this method when using a larger maximal resolution. To estimate the CPU time that such a DNS computation would require, we performed the DNS computation on a few iterations only. We found that the DNS computation would approximately require 71 days of CPU time on a single PC.

The time evolution of the mixing layer is shown in Fig. 71. As before, we observe the efficient self-adaptive behavior of the algorithm, which exhibits more points in the shear zone, and less and less points far from this zone. The performance of the CVS computation is shown in Table 17. As expected, the performance increases with the number of scales, and the CVS computation is here more than four times faster than the DNS one.

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU time</th>
<th>% CPU</th>
<th>% Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVS</td>
<td>16d 17h</td>
<td>23.40</td>
<td>17.06</td>
</tr>
</tbody>
</table>

Table 17. Percentages of CPU time, and CPU and memory compression for the CVS computation. Results for $Re = 200$ and $N^3 = 256^3$.

7. Conclusions

In these lecture notes we reviewed different adaptive multiresolution schemes for nonlinear evolutionary PDEs in Cartesian geometries, in one, two and three space dimensions.

These schemes are shown to be computationally efficient and numerically accurate for a large variety of test-cases, like convection-diffusion or reaction-diffusion equations, the compressible Euler equations and the
Figure 71. Time evolution of a weakly compressible mixing layer at resolution \( N = 256^3 \). CVS computation with \( \epsilon = 0.03 \) and norm #3. First row: Two-dimensional isolines of vorticity in the plane \( y = 0 \), 10 isolines between 0.1 and 1. Second row: Corresponding isosurfaces of vorticity \( ||\omega|| = 0.5 \) (black) and \( ||\omega|| = 0.25 \) (gray). Third Row: Adaptive mesh of the CVS computation. The corresponding time instants are \( t = 19 \) (left), \( t = 37 \) (center) and \( t = 78 \) (right).

Navier-Stokes equations. Starting point of these adaptive methods are either finite volume or finite difference discretizations on regular equidistant grids, combined with explicit time integration schemes. Using discrete multiresolution analysis techniques the computational grid is reduced by deleting non significant grid points while maintaining the second order accuracy of the scheme. A dynamical adaptation strategy which exploits the multiscale representation of the solution by adding neighbored coefficients in scale and space to account for translation and the creation of finer scales of the solution allows to advance the grid in time. For the evaluation of the numerical fluxes in the finite volume schemes on the locally refined grid we devise a conservative scheme without increasing significantly the number of costly flux evaluations. The presented error analysis yields a theoretical relation for the choice of a level dependent threshold for convection-diffusion equations in order to
To guarantee the second order accuracy, which is verified numerically. The adaptive multiresolution algorithms are implemented using graded tree data structures to represent the adaptive grid in the computer memory.

A recursive procedure is used to address each element of the tree. Although this concept is slightly more complex, i.e. an $O(N \log N)$ complexity instead of $O(N)$ (where $N$ denotes the number of active grid points), this choice enables us to avoid hash-tables, which require large arrays and therefore much memory which may be prohibitive for large scale 3D computations.

We also reviewed an efficient space-adaptive multiresolution method with local time stepping to solve evolutionary PDEs in Cartesian geometry. It is again based on a finite volume discretization with explicit time integration, both of second order.

In comparison to the finite volume scheme on a regular grid this new scheme allows further speed-up due to an improved time advancement using larger time steps on large scales without violating the stability condition of the explicit scheme.

However, as different scales evolve with different time steps, a synchronization of the tree data structure becomes necessary. The synchronization limits currently the time scheme to second order Runge–Kutta methods as for higher order schemes this task becomes much more difficult.

The efficiency of the different algorithms is demonstrated by comparing the performance in terms of CPU time and memory requirements with a finite volume method using the same numerical schemes on the finest regular grid with a static data structure. In general we observe that the relative performance increases with the number of required levels and tends towards a minimal value which depends on the test case. Furthermore the gain increases significantly with the spatial dimension of the problem. We have also shown that a suitable thresholding of the wavelet coefficients maintains the second order accuracy of the finite volume scheme on the regular grid. The local time stepping hence represents a moderate, but significant, speed-up with insignificant loss of accuracy.

We presented an aspect of turbulence modeling using adaptive multiresolution techniques.

Here we reviewed an extension of the Coherent Vortex Simulation approach previously developed for incompressible turbulence to weakly compressible flows. The adaptive multiresolution method was applied to solve the three-dimensional compressible Navier–Stokes equations in a Cartesian geometry. It was shown that the time evolution of the coherent flow contributions can be computed efficiently using the adaptive multiresolution method. As generic test case for free shear flows, we considered weakly compressible turbulent mixing layers at low Reynolds numbers ($50 < Re < 200$). Different thresholding rules based on different norms, i.e. $L^1$, $L^2$ and $H^1$ norms, have been investigated. We found that the $H^1$ based threshold yield the best results in terms of accuracy and efficiency. Concerning the computational efficiency, we showed that the CVS computation is approximately three time faster than the DNS for a maximal resolution of $128^3$, and more than four times faster for a maximal resolution $256^3$. The CVS computation also requires less than one third of the memory that the DNS requires for a maximal resolution of $128^3$. In the case $256^3$, this factor is reduced to almost one sixth. Additionally, all the dynamically active scales of the flow are well resolved using a reduced number of
degrees of freedom. The presented CVS results are of course problem dependent, i.e., on the considered flow and its dimensionless parameters. This work can be seen as a first feasibility study of CVS for weakly compressible shear flows and further investigations for higher Reynolds and Mach numbers and other flow types are necessary and will be reported in future work. Applications of the adaptive multiresolution method to supersonic flows considering the compressible Euler equations can be found in [?], which illustrates the potential of the adaptive multiresolution method for higher Mach numbers.

To improve further the compression of the CVS computations the threshold value $\epsilon$ has to be increased and thus more wavelet coefficients are discarded. However in this case the corresponding incoherent flow contributions cannot be simply neglected and some subgrid scale modeling is required to account for turbulent dissipation, similar to LES models. This approach called SCALES has been introduced in [32] and could be one possible direction for future work to obtain higher compression rates.

The restriction to Cartesian geometries can be surmounted using a volume penalization approach, introduced originally by Angot et al. [?] for incompressible flows and successfully applied with adaptive wavelet methods in [?]. An extension of the volume penalization to compressible flows has been published recently in [?] where, in addition to the permeability parameter, an impedance parameter is introduced to get rid of the sound waves inside the obstacle or solid wall regions. Current work is going in this direction to develop efficient CVS methods for compressible flows in complex geometries.

A hot topic and question of ongoing research for the adaptive multiresolution techniques is their parallel implementation on massively parallel machines to perform large scale 3D computations. To reach this goal, the data structure is organized as a “forest”, i.e. an ensemble of trees, each one working on a different processor. Future work will focus on fully adaptive simulations of complex multi-physics phenomena with a multitude of spatial and temporal scales.
APPENDIX A. MULTIRESOLUTION IMPLEMENTATION FOR FV METHODS

A.1. FV-MR Algorithms

In the following, the principle of the algorithm is presented. First, depending on the initial condition, an initial graded tree is created. Then, given the graded tree structure, a time evolution is made on the leaves. Then details are computed by multiresolution transform, in order to remesh the tree. To be able to navigate inside the tree structure, we propose to use a recursive algorithm. The chosen data structure can handle 1D, 2D and 3D Cartesian geometries.

A flowchart of the MR algorithm for FV method is given in Fig. 73. A more detailed description is given below.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{algorithm.png}
\caption{Main algorithm}
\end{figure}

**Step 1: INITIALIZE**
- **Initialize parameters**
  e.g. number of Runge-Kutta (RK) steps, number of time steps, maximum level, domain size
- **Create the initial graded tree structure**
  - Create first cell (root) and compute its cell-average value with the initial condition
  - Split cell and compute the cell-average values in the children cells with the initial condition
  - Compute details in the children cells by multiresolution transform
  - IF the detail in a child cell is higher than the prescribed tolerance, THEN split this child
  - The former child becomes a parent. Repeat the same procedure until all the children cell have low details or the maximum level is reached.
DO n=1, number_of_time_steps

Step 2: TIME EVOLUTION

- Compute Runge-Kutta steps
  
  DO m=0, number_of_RK_steps-1
  
  - Compute the divergence operator for all the leaves
    
    Compute \((\mathcal{D}^{n+1,m}_{l,i})\) for all \(l < L, i \in L(l)\)
  
  - Compute a Runge-Kutta step for all the leaves,
    
    e.g. for number_of_RK_steps = 2,
    
    \[
    \begin{align*}
    m &= 0 : \bar{u}^{n+1,0}_{l,i} = \bar{u}^{n,0}_{l,i} + \Delta t \bar{D}^{n,0}_{l,i}, \quad 0 \leq l < L, i \in L(l) \\
    m &= 1 : \bar{u}^{n+1,1}_{l,i} = \frac{1}{2} \left[ \bar{u}^{n,0}_{l,i} + \bar{u}^{n,1}_{l,i} + \Delta t \bar{D}^{n,1}_{l,i} \right], \quad 0 \leq l < L, i \in L(l)
    \end{align*}
    \]
  
  END DO m

- Check Stability
  
  IF one value is overflow THEN the computation is considered as numerically unstable

- Compute the integral values
  
  e.g. \(\int_{\Omega} |u| \, dx\)

Step 3: REMESH

- Refresh Tree
  
  Recalculate the values in nodes and virtual leaves by projection from the leaves

- Adapt the graded tree structure
  
  - For the whole tree from the leaves to the root \((0 \leq l < L, i \in \Lambda_l)\)
    
    * Compute detail in the node \(d_{l,i}\) by a multiresolution transform
    * IF the details in this node and in its brothers are smaller than the prescribed tolerance
      THEN the cell and its brothers are deletable
  
  - For the whole tree from the leaves to the root \((0 \leq l < L, i \in \Lambda_l)\)
    
    * IF this node and all its children nodes are deletable
      AND the children nodes are simple leaves (i.e. they are leaves without virtual children)
      THEN delete children
      (This means that we keep one more level)
    * IF this node has no children
      AND it is not deletable
      AND it is not at the maximum level
      THEN create the children for this node
      (This means that we add one more level for each undeletable leaf)

Step 4: OUTPUT

Write the mesh and the cell-average values into a file.

: END DO n

Step 5: FINISH

Deallocate tables

Thus, the algorithm can schematically be summarized by

\[
\bar{u}^{n+1} = \bar{M}^{-1} \cdot \mathbf{T}(\epsilon) \cdot \bar{M} \cdot \bar{E}(\Delta t) \cdot \bar{u}^n
\]  

where \(\bar{M}\) is the multiresolution transform operator, \(\bar{M}^{-1}\) its inverse operator, \(\mathbf{T}(\epsilon)\) the thresholding operator with the prescribed tolerance \(\epsilon\), and \(\bar{E}\) the discrete time evolution operator, as defined in (16).
A.2. Data structure

In the implementation of the algorithm, we use a dynamic graded tree structure to represent data in the computer memory. The adaptive grid corresponds to a set of nested dyadic grids generated by refining recursively a given cell depending on the local regularity of the solution.

In the program, the tree is represented by a set of nodes with links between them. The main element of the tree structure is the node, which consists of a set of geometric and physical quantities, and pointers to the children, which are in the next finer subgrid. The root is the first cell of the tree structure.

Each node can be addressed by its level number $l$ and its coordinate in this level $i$ ($i, j$ in 2D, and $i, j, k$ in 3D). The procedure GetNode achieves this goal: starting from the root element, it finds the path to the cell, level by level, by opting for the child which is an ancestor of the target cell.

Thus, any node can be found by a maximum of $L$ steps, $L$ being the maximal level number. Denoting by $N_{\text{max}}$ the maximal number of cells, we have $N_{\text{max}} = 2^{dL}$, i.e. $L = \frac{1}{d} \log_2 N_{\text{max}}$, $d$ being the dimension. Therefore, in the worse case, the global algorithm is of $O(N \log N)$ complexity. Nevertheless, for a highly compressed solution, the $L$ steps are required only in small regions and the number of cells is usually much smaller than $N_{\text{max}}$.

Another possibility is to avoid a GetNode procedure and to use a hash-table. In this case, the algorithm is at least of $O(N)$ complexity. However, this table can be large and requires much memory, especially for large scale 3D computations. That is why we decided here to opt for a solution which optimizes the memory requirements, with a slightly more complex algorithm.

A.3. Local scale-dependent time stepping

We propose a MR/LTS algorithm that uses the multiresolution framework described in the previous section. Assuming that $\Delta t$ is the time step for cells in the finest scale level $L$, the main principle is that the cells at lower levels $L - l$ are evolved with time step $2^l \Delta t$. Consequently, if $U^n_{\text{LTS}}$ represents the numerical solution at $t^n = n \Delta t$ on the adaptive grid $\Gamma^n = \Gamma^* L \subset \Omega_L$, then one complete time cycle of the local time stepping evolution operator evolves the solution from $t^n$ to $t^{n+2^L}$. This adaptive time strategy combined with the multiresolution scheme gives the fully adaptive time cycle.
\[ \bar{U}_{\text{LTS}}^{n+2^l} = (T(\epsilon) \, E_{\text{LTS}} \, R) \bar{U}_{\text{LTS}}^n \]

where \( E_{\text{LTS}} \) denotes the evolution operator using the adaptive multiresolution FV discretization combined with a scale-dependent time stepping. Recall that \( T(\epsilon) \) corresponds to the coarsening (thresholding), and \( R \) to the refinement operations.

In Figure 76, the basic ideas of the local scale-dependent RK2 time stepping are presented for one time cycle and two scale levels. First, we compute one step of RK2 with the local time step. In case the cell level is not on the finest level, we store its values and we use it to interpolate the cell-average value for the previous half local time step, that was not computed (see Figure 76, top-left). Then, with these interpolation values, we perform the RK2 update on the values for the neighbors on the finest level if they exist (see Figure 76, top-right). Following it, we return the cell-average value that we have already stored (see Figure 76, bottom-left). Finally, we advance in time the finest levels and we perform the RK2 usual update (see Figure 76, bottom-right).

We recall that:

- We are working with a spatially graded tree data structure, so we only have one level difference between two neighbour cells.
- We only permit prediction operations on the tree data structure when one complete time cycle is finished. This means that coarsening of the mesh is forbidden during the LTS time cycle, but refinement is possible.

For the whole adaptive tree, one can summarize this cycle of time evolution as follows:

I. Input leaves and time step (Algorithm A.1).
II. Compute LTS time evolution cycle (Algorithm A.2).
III. Adapt tree, update the tree values, combining or splitting them (Algorithm A.5).

More details are presented in Algorithms A.1 to A.5, which are given in the following. Here \( \mathcal{L}(\Lambda) \) are the leaves of the tree \( \Lambda \) and \( \mathcal{V}(\Lambda) \) are the virtual leaves of the tree. The symbol \( \tilde{\ } \) denotes the temporary values.

Algorithm A.1 LTS: INPUT

Require: \( \bar{U}_{\mathcal{L}(\Lambda)}^0 = \{ \bar{u}_{l,i}^0 | (l,i) \in \mathcal{L}(\Lambda) \} \) \{all the leaves at initial time\}
Ensure: \( \bar{U}_{\mathcal{L}(\Lambda)}^n = \{ \bar{u}_{l,i}^n | (l,i) \in \mathcal{L}(\Lambda) \} \) \{all the leaves at the time step \( n \}\}
Require: \( \Delta t = \Delta t_L \) \{the time step on the finest level is set as reference time step\}
Ensure: \( \Delta t_l = 2^{L-l} \Delta t, \quad 0 < l \leq L \) \{time step at the level \( l \}\}

LTS: CYCLE

References

**Figure 76.** Scheme of local scale-dependent time stepping.


Algorithm A.2 LTS: CYCLE

for all iterations $n$ do

LTS: COMPUTE VIRTUAL LEAVES at iteration $n$ (Algorithm A.3)
{STAGE 1}

for all levels from $l = 0$ until $L$ do

for all cells $i$ such that $0 \leq i < 2^l$ do

if $(l, i) \in \mathcal{L}(\Lambda)$ then

$p = 2^{L-l}$

{ $p$ denotes the number of steps being advanced at the level $l$ }

if $(n-1) \mod p = 0$ then

$\vec{D}^n_{l,i} = D(\vec{u}^n_{l,i-2}, \vec{u}^n_{l,i-1}, \vec{u}^n_{l,i+1}, \vec{u}^n_{l,i+2})$ \{Compute divergence\}

$\vec{u}^{n+p}_{l,i} = \vec{u}^n_{l,i} + \Delta t \vec{D}^n_{l,i}$ \{RK 1st stage\}

if $(l \neq L)$ then

$\vec{u}^{n+p+2}_{l,i} = \frac{1}{2} \left( \vec{u}^{n+p}_{l,i} + \bar{u}^n_{l,i} \right)$ \{Store values at intermediary state\}

LTS: UPDATE VIRTUAL LEAVES at iteration $n$ (Algorithm A.4)
{STAGE 2}

for all levels from $l = 0$ until $L$ do

for all cells $i$ such that $0 \leq i < 2^l$ do

if $(l, i) \in \mathcal{V}(\Lambda)$ then

$q = i \div 2$

{the cell $(l-1, q)$ is the parent of the cell $(l, i)$, $i$ being odd or even}

$\bar{u}^n_{l,i} = P_{l-1 \rightarrow l} \left( \vec{u}^n_{l-1,q-1}, \vec{u}^n_{l-1,q} \right)$

{Predict value from parent and uncles}

Algorithm A.3 LTS: COMPUTE VIRTUAL LEAVES at iteration $n$

for all levels from $l = 0$ until $L$ do

for all cells $i$ such that $0 \leq i < 2^l$ do

if $(l, i) \in \mathcal{V}(\Lambda)$ then

$q = i \div 2$

{the cell $(l-1, q)$ is the parent of the cell $(l, i)$, $i$ being odd or even}

$\bar{u}^n_{l,i} = P_{l-1 \rightarrow l} \left( \vec{u}^n_{l-1,q-1}, \vec{u}^n_{l-1,q} \right)$

{Predict value from parent and uncles}


Algorithm A.4 LTS: UPDATE VIRTUAL LEAVES at iteration \( n \)

\[
\text{for all levels from } l = 0 \text{ until } L \text{ do}
\]
\[
\text{for all cells } i \text{ such that } 0 \leq i < 2^l \text{ do}
\]
\[
\text{if } (l, i) \in \mathcal{V}(\Lambda) \text{ then}
\]
\[
q = i \div 2
\]
\[
\{ \text{the cell } (l - 1, q) \text{ is the parent of the cell } (l, i), i \text{ being odd or even} \}
\]
\[
\tilde{u}_{l,i}^n \leftarrow P_{l-1-l} \left( \tilde{u}_{l-1,q-1}^n, \tilde{u}_{l-1,q}^n, \tilde{u}_{l-1,q+1}^n \right)
\]
\[
\{ \text{Predict value from parent and uncles} \}
\]

Algorithm A.5 LTS: ADAPT TREE at iteration \( n \)

Require: \( n \)

\[
\text{for all levels from } l = 0 \text{ until } L \text{ do}
\]
\[
\text{for all cells } i \text{ such that } 0 \leq i < 2^l \text{ do}
\]
\[
\text{if } (l, i) \in \mathcal{L}(\Lambda) \text{ then}
\]
\[
p = 2^{L-l}
\]
\[
\{ p \text{ denotes the number of steps being advanced at the level } l \}
\]
\[
\text{if } (n \mod p) = 0 \text{ then}
\]
\[
\text{Delete children if possible}
\]
\[
\text{if } (|d^n_{l+1,2i}| < \epsilon_{l+1}) \text{ and } (|d^n_{l+1,2i+1}| < \epsilon_{l+1}) \text{ and } (|d^n_{l,i}| < \epsilon_l) \text{ then}
\]
\[
\text{Deallocate children:}
\]
\[
\text{Deallocate } \left( \tilde{u}_{l+1,2i}^n \right)
\]
\[
\text{Deallocate } \left( \tilde{u}_{l+1,2i+1}^n \right)
\]
\[
\text{else}
\]
\[
\text{if } (n \mod p \div 2 = 0) \text{ then}
\]
\[
\text{Add children when necessary}
\]
\[
\text{if } (|d^n_{l,i}| > \epsilon_l) \text{ then}
\]
\[
\text{Allocate children:}
\]
\[
\text{Allocate } \left( \tilde{u}_{l+1,2i}^n \right)
\]
\[
\text{Allocate } \left( \tilde{u}_{l+1,2i+1}^n \right)
\]
\[
\{ \text{Predict value from parent and uncles} \}
\]
\[
\tilde{u}_{l+1,2i}^n = P_{l-1-l} \left( 2i; \tilde{u}_{l+1,i-1}^n, \tilde{u}_{l+1,i}^n, \tilde{u}_{l+1,i+1}^n \right)
\]
\[
\tilde{u}_{l+1,2i+1}^n = P_{l-1-l} \left( 2i + 1; \tilde{u}_{l+1,i-1}^n, \tilde{u}_{l+1,i}^n, \tilde{u}_{l+1,i+1}^n \right)
\]


**Acknowledgments**

M.O. Domingues thankfully acknowledges financial support from the ANR project “M2TFP”, Ecole 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. We are grateful to Dominique Fouger and Varlei E. Menconni for their helpful computational assistance.