Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 188 (2003) 493–523
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A conservative fully adaptive multiresolution algorithm for parabolic PDEs

Olivier Roussel [a,b], Kai Schneider [a,c,*], Alexei Tsigulin [d], Henning Bockhorn [b]

[a] *Laboratoire de Modélisation et Simulation Numérique en Mécanique, CNRS et Universités d'Aix-Marseille, 38 rue Frédéric Joliot-Curie, 13451 Marseille cedex 20, France*
[b] *Institut für Chemische Technik, Universität Karlsruhe (TH), Kaiserstr. 12, 76128 Karlsruhe, Germany*
[c] *Centre de Mathématiques et d'Informatique, Université d'Aix-Marseille I, 39 rue Frédéric Joliot-Curie, 13453 Marseille cedex 13, France*
[d] *Institute of Computational Mathematics and Mathematical Geophysics (RAS), Novosibirsk State Technical University, Lavrentiev pr. 6, 630090 Novosibirsk, Russia*

Received 12 August 2002; received in revised form 20 February 2003; accepted 21 February 2003

Abstract

We present a new adaptive numerical scheme for solving parabolic PDEs in Cartesian geometry. Applying a finite volume discretization with explicit time integration, both of second order, we employ a fully adaptive multiresolution scheme to represent the solution on locally refined nested grids. The fluxes are evaluated on the adaptive grid. A dynamical adaption strategy to advance the grid in time and to follow the time evolution of the solution directly exploits the multiresolution representation. Applying this new method to several test problems in one, two and three space dimensions, like convection–diffusion, viscous Burgers and reaction–diffusion equations, we show its second-order accuracy and demonstrate its computational efficiency.

Keywords: Finite volume; Adaptivity; Multiresolution; Parabolic partial differential equation

1. Introduction

The numerical solution of partial differential equations (PDEs) arising from mathematical modeling of physical–chemical problems, like turbulent, reactive or non-reactive flows, typically involves a large number of spatial and temporal scales. In many cases, however, small scales in space are only needed locally, i.e., for

* Corresponding author.
E-mail addresses: roussel@ict.uni-karlsruhe.de (O. Roussel), kschneid@cmi.univ-mrs.fr (K. Schneider), tsigul@ssd2.sscc.ru (A. Tsigulin), bockhorn@ict.uni-karlsruhe.de (H. Bockhorn).

0021-9991/03/$ - see front matter
doi:10.1016/S0021-9991(03)00189-X

solutions being intermittent or exhibiting, e.g., steep gradients or shock-like structures. This property motivates the introduction of some kind of adaptive discretization as the solution may be over-resolved in large subsets of the computational domain when using equidistant fine grids.

A suitable tool to define adaptive discretization schemes are multiresolution techniques which allow an efficient data representation with an accurate estimation of the local approximation error, together with a dynamic grid adaption strategy for evolution problems. In the past, different adaptive methods have been introduced to improve the computational efficiency and to reduce the memory requirement of the algorithms for solving large scale problems. Historically, adaptive grid methods like Multi-Level Adaptive Techniques (MLAT) [9] or Adaptive Mesh Refinement (AMR) methods [2–4,34] were the first to achieve this goal, using a set of locally refined grids where steep gradients or high truncation errors are found. However, the data compression rate is high where the solution is almost constant, but remains low where the solution is smooth.

More recently, adaptive wavelet methods to solve PDEs have been developed. For an overview and a classification of the different methods we refer, e.g., to Dahmen [16], Cohen [12] and Fröhlich and Schneider [23]. The motivation to use wavelet methods to construct numerical schemes is twofold. First, the scale-space representation of functions exhibiting, e.g., locally steep gradients or boundary layers, i.e., functions whose Besov regularity is larger than its Sobolev regularity, is efficient using nonlinear approximation, i.e., by thresholding the wavelet coefficients [16,19]. The result is that only few coefficients are necessary to represent a function for a given accuracy. Second, a large class of differential and integral operators have a sparse representation in a wavelet basis [5] and can be preconditioned by simple rescaling [18,28]. For evolutionary problems, wavelet schemes offer the possibility to adapt the basis automatically in time by simply switching on wavelet coefficients in the neighborhood of the active ones. These properties led to the development of several adaptive wavelet schemes, e.g., for 1D viscous Burgers equations [29–31], for thermo-diffusive flame computations [8,21,23], for 2D Stokes equations [39] and 2D Navier–Stokes equations [10,22,37,38]. The above schemes mainly use wavelets as trial and/or test functions in Petrov–Galerkin schemes.

The current approach is somehow different and can be seen in the spirit of Harten's pioneering work [25,26]. Starting point is a finite volume scheme for hyperbolic conservation laws on a regular grid. Subsequently a discrete multiresolution analysis is used to avoid expensive flux computations in smooth regions, first without reducing memory requirements, e.g., for 1D hyperbolic conservation laws [25], 1D conservation laws with viscosity [6], 2D hyperbolic conservations laws [7], 2D compressible Euler equations [11], 2D hyperbolic conservation laws with curvilinear patches [17] and unstructured meshes [1,14]. A fully adaptive version, still in the context of hyperbolic conservation laws, has been developed to reduce also memory requirements [15,24]. Therewith the solution is represented and computed on a dynamically evolving automatically adapted grid. Different strategies have been proposed to evaluate the flux without requiring a full knowledge of fine grid cell-average values. For an overview on adaptive multiresolution techniques for hyperbolic conservation laws, we refer to Müller [32]. For more details on similarities and differences between adaptive mesh refinement and adaptive wavelet approaches, we refer the reader to Cohen [13].

In the current paper, we present a fully adaptive multiresolution scheme for solving parabolic PDEs in one, two and three space dimensions with different types of boundary conditions. With respect to previous work, we extend the algorithms developed for hyperbolic equations [15] for the case of parabolic ones, together with the corresponding error analysis. As we are concerned with problems arising from physical–chemical context, conservation of physical quantities – e.g., global mass – in flux computations is of special interest. In the case of adaptive flux evaluation, ingoing and outgoing fluxes, both approximated at interfaces from cell-average values of different levels, are not necessarily balanced. Therefore we devise a new formulation for adaptive flux computation at the interfaces between two different levels being strictly conservative.

The paper is organized as follows: in Section 2, a general finite volume method for conservation laws is presented, including the description of space discretization and time integration schemes used here. In Section 3, the conservative version of the fully adaptive multiresolution scheme is described, including a way to ensure conservativity of the flux computations. In Section 4, we describe the algorithm used for the numerical computation. Details concerning the data structure and the implementation are also given. Then, Section 5 contains numerical results to show the accuracy of the algorithm and to demonstrate its efficiency. For the validation of the adaptive method, test-problems like convection–diffusion or viscous Burgers are studied. We report gains in CPU time and memory, as well as convergence rates. The method is also applied to reaction–diffusion problems, in order to compute the time evolution of a flame ball. Finally, we conclude and present some perspectives for future work.

2. Numerical method

2.1. Parabolic conservation laws

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

$$\begin{aligned} &\frac{\partial u}{\partial t} + \nabla \cdot F(u,\nabla u) = S(u), \\ &u(x,0) = u_0(x) \end{aligned} \tag{1}$$

with appropriate boundary conditions.

In the present paper, we restrict ourselves to the case where the diffusive flux is defined by a gradient operator, assuming constant diffusivity $\nu > 0$, i.e.,

$$F(u,\nabla u) = f(u) - \nu \nabla u.$$

We shortly summarize the advective flux f and the source term S for the different test-cases presented in Section 5, which yield simple models for viscous fluid dynamics and combustion problems. For the 1D convection–diffusion equation, we have $(c > 0)$

$$\begin{aligned} f(u) &= cu, \\ S(u) &= 0. \end{aligned}$$

In the case of the 1D viscous Burgers equation, we get

$$\begin{aligned} f(u) &= \frac{u^2}{2}, \\ S(u) &= 0 \end{aligned}$$

and for the reaction–diffusion equation $(\alpha > 0, \beta > 0)$,

$$\begin{aligned} f(u) &= 0, \\ S(u) &= \frac{\beta^2}{2}(1-u)\exp\frac{\beta(1-u)}{\alpha(1-u)-1}. \end{aligned}$$

For ease of notation, we denote by $\mathscr{D}(u,\nabla u) = -\nabla \cdot F(u,\nabla u) + S(u)$ the *divergence and source term*. Thus, (1) can be written in the form

$$\frac{\partial u}{\partial t} = \mathscr{D}(u, \nabla u). \tag{2}$$

2.2. Discretized equations

To discretize (2), we use a classical finite volume formulation in the standard conservative form. In the general case, let us consider the computational domain Ω in dimension d with an arbitrary shape, and let us partition it into cells $(\Omega_i)_{i\in\Lambda}$, $\Lambda = \{1, \ldots, i_{\max}\}$. We then denote $\bar{q}_i(t)$ the cell-average value of a given quantity q on Ω_i at instant t,

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

where $|\Omega_i| = \int_{\Omega_i} \mathrm{d}x$ is the volume of the cell. Integrating (2) on Ω_i yields

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

i.e.,

$$\frac{\partial \bar{u}_i}{\partial t}(t) = \bar{\mathscr{D}}_i(t). \tag{3}$$

Applying the divergence theorem, we get

$$\bar{\mathscr{D}}_i(t) = -\frac{1}{|\Omega_i|}\int_{\partial\Omega_i} F[u(x,t), \nabla u(x,t)] \cdot \sigma_i(x)\,\mathrm{d}x + \bar{S}_i(t), \tag{4}$$

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

Conservativity in the flux computation is ensured if and only if, for two adjacent cells Ω_{i_1} and Ω_{i_2}, the outgoing flux from Ω_{i_1} to Ω_{i_2} balances with the one from Ω_{i_2} to Ω_{i_1}. In the next sections, we will describe the time integration and space discretization schemes applied to (3).

2.3. Time integration

Due to the adaptive space discretization, the grid is changing in time, and therefore we first discretize in time and then in space. Here we use an explicit second-order accurate Runge–Kutta (RK2) scheme. Denoting by Δt 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

$$\begin{aligned} \bar{u}_i^{n+\frac{1}{2}} &= \bar{u}_i^n + \Delta t \bar{\mathscr{D}}_i^n, \\ \bar{u}_i^{n+1} &= \frac{1}{2}\left[\bar{u}_i^n + \bar{u}_i^{n+\frac{1}{2}} + \Delta t \bar{\mathscr{D}}_i^{n+\frac{1}{2}}\right]. \end{aligned} \tag{5}$$

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

$$\bar{u}^{n+1} = \bar{\mathbf{E}}(\Delta t) \cdot \bar{u}^n, \tag{6}$$

where

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

and **I** denotes the identity operator. The discretization of the operator $\bar{\mathscr{D}}$ is described in the following section, the stability conditions being treated in Section 2.5.

2.4. Numerical flux

We now consider a fixed time t^n and, in the following, the superscript n is omitted everywhere. For the 1D case, Ω_i is a segment $\left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$ with step size $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. Eq. (4) becomes

$$\bar{\mathscr{D}}_i = -\frac{1}{\Delta x_i}\left(\bar{F}_{i+\frac{1}{2}} - \bar{F}_{i-\frac{1}{2}}\right) + \bar{S}_i. \tag{7}$$

Advective and diffusive terms are approximated differently. For the advective part, we use Roe's scheme [35] with a second-order ENO interpolation, whereas, for the diffusive part, we choose a second-order accurate centered scheme. Bihari [6] showed that the resulting global scheme, which is non-linear, is second-order accurate in space:

$$\bar{F}_{i+\frac{1}{2}} = f^{\mathrm{R}}\left(\bar{u}^-_{i+\frac{1}{2}}, \bar{u}^+_{i+\frac{1}{2}}\right) - \nu\frac{\bar{u}_{i+1} - \bar{u}_i}{\Delta x_{i+\frac{1}{2}}}, \tag{8}$$

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

$$f^{\mathrm{R}}(u^-, u^+) = \frac{1}{2}[f(u^-) + f(u^+) - |a(u^-, u^+)|(u^+ - u^-)], \tag{9}$$

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

$$\begin{aligned} \bar{u}^-_{i+\frac{1}{2}} &= \bar{u}_i + \frac{1}{2}M\left(\bar{u}_{i+1} - \bar{u}_i, \bar{u}_i - \bar{u}_{i-1}\right), \\ \bar{u}^+_{i+\frac{1}{2}} &= \bar{u}_{i+1} + \frac{1}{2}M\left(\bar{u}_{i+2} - \bar{u}_{i+1}, \bar{u}_{i+1} - \bar{u}_i\right), \end{aligned} \tag{10}$$

where M is the *Min–Mod* limiter, which chooses the minimal slope between the left and right sides, i.e.,

$$M(a, b) = \begin{cases} a & \text{if } |a| \leqslant |b|, \\ b & \text{if } |a| > |b|. \end{cases}$$

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. (3) can be written as

$$\frac{\partial \bar{u}_{i,j}}{\partial t}(t) = \bar{\mathscr{D}}_{i,j}(t), \tag{11}$$

where

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

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

$$\begin{aligned}
\bar{F}_{i+\frac{1}{2},j} &= f^{\mathrm{R}}\left(\bar{u}^{-}_{i+\frac{1}{2},j}, \bar{u}^{+}_{i+\frac{1}{2},j}\right) - \nu\frac{\bar{u}_{i+1,j} - \bar{u}_{i,j}}{\Delta x_{i+\frac{1}{2}}},\\
\bar{F}_{i,j+\frac{1}{2}} &= f^{\mathrm{R}}\left(\bar{u}^{-}_{i,j+\frac{1}{2}}, \bar{u}^{+}_{i,j+\frac{1}{2}}\right) - \nu\frac{\bar{u}_{i,j+1} - \bar{u}_{i,j}}{\Delta y_{j+\frac{1}{2}}},
\end{aligned} \tag{12}$$

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

$$\frac{\partial \bar{u}_{i,j,k}}{\partial t}(t) = \bar{\mathscr{D}}_{i,j,k}(t), \tag{13}$$

where

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

The fluxes are in this case

$$\begin{aligned}
\bar{F}_{i+\frac{1}{2},j,k} &= f^{\mathrm{R}}\left(\bar{u}^{-}_{i+\frac{1}{2},j,k}, \bar{u}^{+}_{i+\frac{1}{2},j,k}\right) - \nu\frac{\bar{u}_{i+1,j,k} - \bar{u}_{i,j,k}}{\Delta x_{i+\frac{1}{2}}},\\
\bar{F}_{i,j+\frac{1}{2},k} &= f^{\mathrm{R}}\left(\bar{u}^{-}_{i,j+\frac{1}{2},k}, \bar{u}^{+}_{i,j+\frac{1}{2},k}\right) - \nu\frac{\bar{u}_{i,j+1,k} - \bar{u}_{i,j,k}}{\Delta y_{j+\frac{1}{2}}},\\
\bar{F}_{i,j,k+\frac{1}{2}} &= f^{\mathrm{R}}\left(\bar{u}^{-}_{i,j,k+\frac{1}{2}}, \bar{u}^{+}_{i,j,k+\frac{1}{2}}\right) - \nu\frac{\bar{u}_{i,j,k+1} - \bar{u}_{i,j,k}}{\Delta z_{k+\frac{1}{2}}},
\end{aligned} \tag{14}$$

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

2.5. Numerical stability

As the time step is the same for all scales, the stability condition is the one of the same finite volume scheme on the finest grid. For the linear convection–diffusion equation, denoting by c the velocity and by Δx the smallest step size, the CFL number σ is given by

$$\sigma = \frac{c\Delta t}{\Delta x}$$

and the mesh Reynolds number Re by

$$Re = \frac{c\Delta x}{\nu}.$$

Bihari [6] showed that a sufficient stability condition for the above finite volume scheme is

$$\sigma \leqslant \min\left(\frac{Re}{2}, \frac{6}{Re}\right). \tag{15}$$

Moreover, a sufficient condition, so that this scheme is Total Variation Diminishing (TVD), is

$$\sigma \leqslant \frac{Re}{Re+4}. \tag{16}$$

The main advantage of an explicit treatment for the diffusive term is that no linear system needs to be solved. However, it usually implies that $\Delta t = \mathrm{O}(\Delta x^2)$. Only for $Re \gg 1$, we have $\Delta t = \mathrm{O}(\Delta x)$.

3. 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.

3.1. 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 [19] showed that the difference between both approximations is negligible in terms of required nodes.

Following [15], we first introduce a terminology to define the tree structure.

- The *root* is the basis of the tree;
- A *node* is a 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. 1 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;

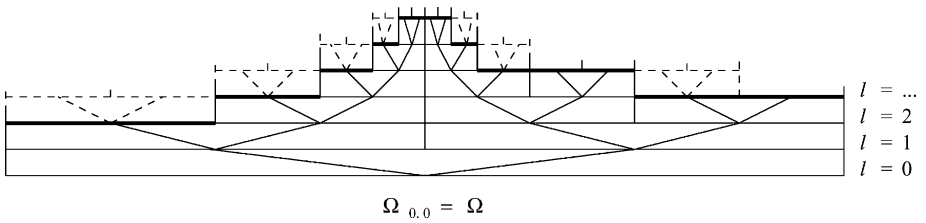

Fig. 1. Example of graded tree data structure in 1D for $s = 1$, $s' = 2$.

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 and will be explained in the next section.

3.2. Multiresolution representation

Starting point is the cell-average multiresolution representation [25]. The nodes are cell-average values and two operators are defined to navigate through the tree. A complete description of the 1D multiresolution representation is given in this section and a brief description explains how to extend it to higher dimensions in Cartesian geometry using a tensor product approach.

We denote by Λ the ensemble of the indices of the existing nodes, by $\mathscr{L}(\Lambda)$ the restriction of Λ to the leaves, and by Λ_l the restriction of Λ to a level l, $0 \leqslant l < L$.

For the 1D case, we denote by $\Omega = \Omega_{0,0}$ the root cell, $\Omega_{l,i}$, $0 \leqslant l < L$, $i \in \Lambda_l$ the different node cells, $\bar{q}_{l,i}$ the cell-average value of the quantity q on the cell $\Omega_{l,i}$, and $\bar{Q}_l = (\bar{q}_{l,i})_{i \in \Lambda_l}$ the ensemble of the existing cell-average values at the level l.

To estimate the cell-averages of a level l from the ones of the level $l+1$, we use the *projection* (or restriction) operator $P_{l+1 \to l}$ (Fig. 2).

$$P_{l+1 \to l} : \bar{U}_{l+1} \mapsto \bar{U}_l. \tag{17}$$

This operator is *exact* and *unique*, given that the parent cell-average is nothing but the weighted average of the children cell-averages. For a regular grid structure in 1D, it is simply defined by the mean value

$$\bar{u}_{l,i} = (P_{l+1 \to l} \bar{U}_{l+1})_i = \frac{1}{2}(\bar{u}_{l+1,2i} + \bar{u}_{l+1,2i+1}). \tag{18}$$

To estimate the cell-averages of a level $l+1$ from the ones of the level l, we use the *prediction* (or prolongation) operator $P_{l \to l+1}$ (Fig. 3).

$$P_{l \to l+1} : \bar{U}_l \mapsto \hat{U}_{l+1}. \tag{19}$$

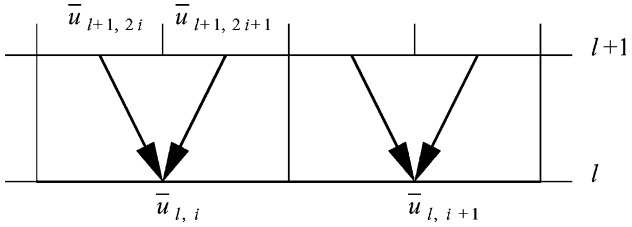

Fig. 2. Projection operator $P_{l+1 \to l}$ in 1D.

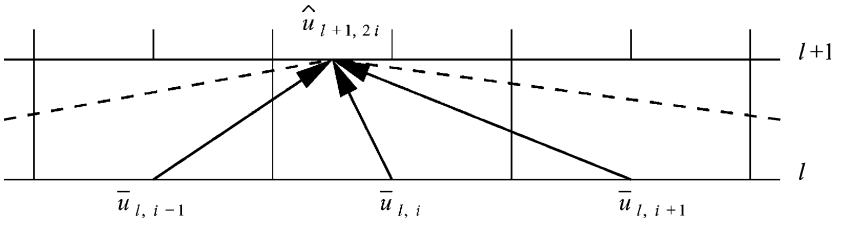

Fig. 3. Prediction operator $P_{l\to l+1}$ in 1D.

This operator gives an *approximation* of $\bar{U}_l$ at the level $l+1$ by interpolation. It is *not unique*, nevertheless, in order to be applicable in the dynamic graded tree structure as defined above, this operator must satisfy two properties:

- It has to be *local*, i.e., the interpolation for a child is made from the cell-averages of its parent and its s nearest uncles in each direction;
- It has to be *consistent with the projection*, i.e., $P_{l+1\to l}\circ P_{l\to l+1} = \mathrm{Id}$.

For a regular grid structure in 1D, we use as prediction operator a polynomial interpolation on the cell-average values, like the one proposed by Harten [25]:

$$\begin{aligned}\hat{u}_{l+1,2i} &= I(\bar{U}_l; l+1, 2i) = \bar{u}_{l,i} + \sum_{m=1}^{s} \gamma_m(\bar{u}_{l,i+m} - \bar{u}_{l,i-m}),\\ \hat{u}_{l+1,2i+1} &= I(\bar{U}_l; l+1, 2i+1) = \bar{u}_{l,i} - \sum_{m=1}^{s} \gamma_m(\bar{u}_{l,i+m} - \bar{u}_{l,i-m}).\end{aligned} \tag{20}$$

The accuracy order of the multiresolution method is denoted by r. A rth order accuracy corresponds to a polynomial interpolation of degree $(r-1)$. The degree r is therefore related to the number of required nearest uncles s by the relation $r = 2s+1$. The corresponding coefficients used in the computations are

$$\begin{aligned} r = 3 \;&\Rightarrow\; \gamma_1 = -\frac{1}{8},\\ r = 5 \;&\Rightarrow\; \gamma_1 = -\frac{22}{128},\ \gamma_2 = \frac{3}{128}.\end{aligned} \tag{21}$$

The *detail* is the difference between the exact and the predicted value. In the 1D case, it is defined as

$$\bar{d}_{l,i} = \bar{u}_{l,i} - \hat{u}_{l,i}. \tag{22}$$

These coefficients are redundant, the sum of the details for all the brothers of a parent cell being equal to zero by definition [25]. Given that a parent has 2^d children, only $2^d - 1$ details are independent. Thus, the knowledge of the cell-average value on the 2^d children is equivalent to the knowledge of the cell-average value of the parent and these $2^d - 1$ independent details. This can be expressed in 1D by

$$\left(\bar{u}_{l+1,2i}, \bar{u}_{l+1,2i+1}\right) \;\leftrightarrow\; \left(\bar{d}_{l+1,2i}, \bar{u}_{l,i}\right).$$

For more details on this equivalence, we refer to Harten [25]. For a given level l, it can be summarized by

$$\bar{U}_l \;\leftrightarrow\; (\bar{D}_l, \bar{U}_{l-1}).$$

Repeating this operation recursively on L levels, one gets the so-called *multiresolution transform* on the cell-average values [25].

$$\bar{\mathbf{M}} : \bar{U}_L \mapsto (\bar{D}_L, \bar{D}_{L-1}, \ldots, \bar{D}_1, \bar{U}_0). \tag{23}$$

For the 2D case, we denote by $\Omega = \Omega_{0,0,0}$ the root cell, $\Omega_{l,i,j}$, $0 \leqslant l < L$, $(i,j) \in \Lambda_l$ the different node cells, by $\bar{q}_{l,i,j}$ the cell-average value of the quantity q on the cell $\Omega_{l,i,j}$. For ease of notation, we denote by $\bar{Q}_l = (\bar{q}_{l,i,j})_{(i,j)\in\Lambda_l}$ the ensemble of the existing cell-average values at the level l.

The projection operator for a regular Cartesian grid is then defined by

$$\bar{u}_{l,i,j} = (P_{l+1\to l}\bar{U}_{l+1})_{i,j} = \frac{1}{4}\sum_{n=0}^{1}\sum_{p=0}^{1}\bar{u}_{l+1,2i+n,2j+p}$$

and the prediction operator based on a linear polynomial interpolation is defined by, for $n,p \in \{0,1\}$,

$$\hat{u}_{l+1,2i+n,2j+p} = I(\bar{U}_l; l+1, 2i+n, 2j+p).$$

The four values $I(\bar{U}_l; l+1, 2i, 2j)$, $I(\bar{U}_l; l+1, 2i+1, 2j)$, $I(\bar{U}_l; l+1, 2i, 2j+1)$ and $I(\bar{U}_l; l+1, 2i+1, 2j+1)$ are given in Appendix A.

As in the 1D case, the details are defined by $\bar{d}_{l,i,j} = \bar{u}_{l,i,j} - \hat{u}_{l,i,j}$ and the knowledge of the cell-average values on the 4 children is equivalent to the knowledge of the cell-average value of the parent and 3 independent details.

Analogously to the 2D case, for the 3D case we denote by $\Omega = \Omega_{0,0,0,0}$ the root cell, $\Omega_{l,i,j,k}$, $0 \leqslant l < L$, $(i,j,k) \in \Lambda_l$ the different node cells, $\bar{q}_{l,i,j,k}$ the cell-average value of the quantity q on the cell $\Omega_{l,i,j,k}$, and $\bar{Q}_l = (\bar{q}_{l,i,j,k})_{(i,j,k)\in\Lambda_l}$ the ensemble of the existing cell-average values at the level l.

The projection operator for a regular Cartesian grid becomes

$$\bar{u}_{l,i,j,k} = (P_{l+1\to l}\bar{U}_{l+1})_{i,j,k} = \frac{1}{8}\sum_{n=0}^{1}\sum_{p=0}^{1}\sum_{q=0}^{1}\bar{u}_{l+1,2i+n,2j+p,2k+q}$$

and the prediction operator based on a linear polynomial interpolation is defined by, for $n,p,q \in \{0,1\}$,

$$\hat{u}_{l+1,2i+n,2j+p,2k+q} = I(\bar{U}_l; l+1, 2i+n, 2j+p, 2k+q).$$

For the eight values $I(\bar{U}_l; l+1, 2i+n, 2j+p, 2k+q)$, $n,p,q \in \{0,1\}$, we refer to Appendix A.

As in the 1D and 2D cases, the details are $\bar{d}_{l,i,j,k} = \bar{u}_{l,i,j,k} - \hat{u}_{l,i,j,k}$ and the knowledge of the cell-average values on the 8 children is equivalent to the knowledge of the cell-average value of the parent and 7 independent details.

In conclusion, the knowledge of the cell-average values of all the leaves is equivalent to the knowledge of the cell-average value of the root and the details of all the other nodes of the tree structure.

3.3. Error analysis

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

$$\|\bar{u}^L_{\mathrm{ex}} - \bar{u}^L_{\mathrm{MR}}\| \leqslant \|\bar{u}^L_{\mathrm{ex}} - \bar{u}^L_{\mathrm{FV}}\| + \|\bar{u}^L_{\mathrm{FV}} - \bar{u}^L_{\mathrm{MR}}\|, \tag{24}$$

where $\|\cdot\|$ denotes, e.g., the $\mathscr{L}^1$, $\mathscr{L}^2$, or $\mathscr{L}^\infty$ 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

$$\|\bar{u}^L_{\mathrm{ex}} - \bar{u}^L_{\mathrm{FV}}\| \leqslant C2^{-\alpha L}, \quad C > 0, \tag{25}$$

where α 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. [15] showed that, if the details on a level l are deleted when smaller than a prescribed tolerance ϵ_l, if the discrete time evolution operator $\bar{\mathbf{E}}$ is contractive in the chosen norm, and if the tolerance ϵ_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

$$\|\bar{u}^L_{\mathrm{FV}} - \bar{u}^L_{\mathrm{MR}}\| \leqslant Cn\epsilon, \quad C > 0, \tag{26}$$

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

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

For the linear convection–diffusion equation with the numerical scheme defined above, the time step Δt, following (16), must verify

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

Denoting X the size of the domain and Δx the smallest space step, we have $\Delta x = X2^{-L}$, from which we deduce that

$$\Delta t = C\frac{\Delta x^2}{4\nu + c\Delta x} = C\frac{X2^{-2L}}{4\nu + cX2^{-L}}, \quad 0 < C < 1.$$

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.,

$$\epsilon 2^{2L}\left(4\nu + cX2^{-L}\right) \propto 2^{-\alpha L}.$$

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

$$\epsilon \propto \frac{2^{-(\alpha+1)L}}{Pe + 2^{L+2}}. \tag{27}$$

For the inviscid case (i.e., $\nu = 0$ or $Pe \to +\infty$), (27) is equivalent to the result obtained by Cohen et al. [15], i.e., $\epsilon \propto 2^{-(\alpha+1)L}$. In the numerical computations in Section 5, the so-called *reference tolerance* will be set to

$$\epsilon_{\mathrm{R}} = C\frac{2^{-(\alpha+1)L}}{Pe + 2^{L+2}}. \tag{28}$$

To choose an acceptable value for the factor C, a series of computations with different tolerances will be necessary, as shown in Section 5.1.

3.4. 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 (see Figs. 2 and 3).

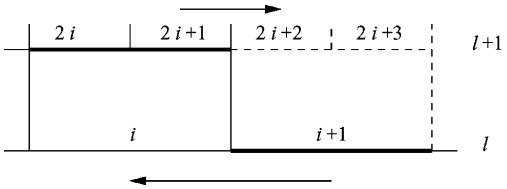

Fig. 4. Ingoing and outgoing flux computation in 1D for two different levels.

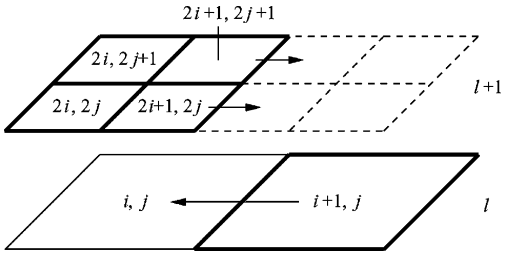

Fig. 5. Ingoing and outgoing flux computation in 2D for two different levels.

As shown in Fig. 4, the outgoing flux from $\Omega_{l+1,2i+1}$ in the right direction $F_{l+1,2i+1\to l+1,2i+2}$ is not balanced with the outgoing flux from $\Omega_{l,i+1}$ in the left direction $F_{l,i+1\to l,i}$. 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. 5.

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.,

$$F_{l,i,j\to l,i+1,j} = F_{l+1,2i+1,2j\to l+1,2i+2,2j} + F_{l+1,2i+1,2j+1\to l+1,2i+2,2j+1}.$$

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. Algorithm implementation

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 (see Figs. 6 and 7).

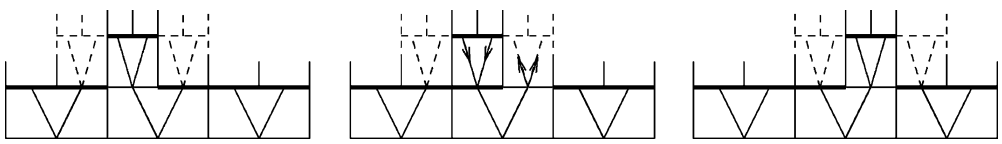

Fig. 6. Time evolution of the tree structure at time step t^n (*left*), at time step t^{n+1} before remeshing (*middle*), and at time step t^{n+1} after remeshing (*right*). Deletable cells are represented by a *bold* line, undeletable cells by a *thin* line, and virtual leaves by a *dotted* line.

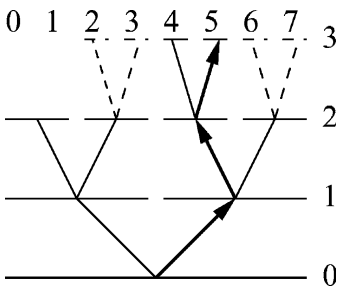

Fig. 7. Path used by *GetNode* to reach the node $l = 3$, $i = 5$.

4.1. Description

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 $(\mathscr{D}_{l,i}^{n+1,m})_{0 \leqslant l<L, i \in \mathscr{L}(\Lambda_l)}$
  - Compute a Runge–Kutta step for all the leaves, e.g., for number_of_RK_steps = 2,

$$m = 0 : \bar{u}_{l,i}^{n,1} = \bar{u}_{l,i}^{n,0} + \Delta t \bar{\mathscr{D}}_{l,i}^{n,0}, \quad 0 \leqslant l < L, \ i \in \mathscr{L}(\Lambda_l)$$

$$m = 1 : \bar{u}_{l,i}^{n+1,0} = \frac{1}{2}\left[\bar{u}_{l,i}^{n,0} + \bar{u}_{l,i}^{n,1} + \Delta t \bar{\mathscr{D}}_{l,i}^{n,1}\right], \quad 0 \leqslant l < L, \ i \in \mathscr{L}(\Lambda_l)$$

  END DO m
- *Check Stability*
  IF one value is overflow THEN the computation is considered as numerically unstable
- Compute the integral values
  e.g., $\int_\Omega |u| \mathrm{d}x$

Step 3. *REMESH*

- *Refresh Tree*
  Recalculate the values in nodes and virtual leaves by projection from the leaves
- *Adapt the graded tree structure* (Fig. 6)
  - For the whole tree from the leaves to the root ($0 \leqslant l < L, \ i \in \Lambda_l$)
    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 ($0 \leqslant l < L, i \in \Lambda_l$)
  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

$$\bar{u}^{n+1} = \bar{\mathbf{M}}^{-1} \cdot \mathbf{T}(\epsilon) \cdot \bar{\mathbf{M}} \cdot \bar{\mathbf{E}}(\Delta t) \cdot \bar{u}^n, \tag{29}$$

where $\bar{\mathbf{M}}$ is the *multiresolution transform* operator, $\bar{\mathbf{M}}^{-1}$ its inverse operator, $\mathbf{T}(\epsilon)$ the *thresholding* operator with the prescribed tolerance ϵ, and $\bar{\mathbf{E}}$ the discrete *time evolution* operator, as defined in (6).

4.2. 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 (Fig. 7): 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 $N_{\max}$ the maximal number of cells, we have $N_{\max} = 2^{dL}$, i.e., $L = \frac{1}{d} \log_2 N_{\max}$, d being the dimension. Therefore, in the worst case, the global algorithm is of $\mathrm{O}(N \log N)$ 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 $N_{\max}$.

Another possibility is to avoid a *GetNode* procedure and to use a hash-table. In this case, the algorithm is at least of $\mathrm{O}(N)$ 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.

5. Numerical results

In this section, we present 1D, 2D, 3D numerical results using a second-order accurate scheme with a multiresolution accuracy of $r = 3$. This means that only one nearest uncle in each direction is

required. This choice enables us to maintain a narrow tree structure around regions where high levels are needed.

5.1. 1D convection–diffusion equation

We consider a linear-convection–diffusion equation for $(x,t) \in [-1,1] \times [0,+\infty)$, $c > 0$, $v > 0$,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = v\frac{\partial^2 u}{\partial x^2}. \tag{30}$$

Considering as characteristic length scale the size of the domain X and as characteristic time scale $T = c/X$, this equation can be written in the following dimensionless form

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \frac{1}{Pe}\frac{\partial^2 u}{\partial x^2}, \tag{31}$$

where Pe denotes the Peclet number $Pe = \frac{cX}{v}$. We choose as initial condition

$$u_0(x) = \begin{cases} 1 & \text{if } x \leqslant 0, \\ 0 & \text{if } x > 0 \end{cases}$$

and Dirichlet conditions at the left and right boundaries, i.e.,

$$u(0,t) = 1,$$
$$u(1,t) = 0$$

The analytic solution is given in Hirsch [27]

$$u_{\text{ex}}(x,t) = \frac{1}{2}\text{erfc}\left(\frac{x-t}{2}\sqrt{\frac{Pe}{t}}\right). \tag{32}$$

5.1.1. Numerical results for a given scale and a given tolerance at $Pe = 1000$

The numerical solution of (31) at $t = 0.5$ is given in Fig. 8 for $Pe = 1000$, $\epsilon = 10^{-3}$ and $L = 11$ scales, which corresponds to a maximum of $2^{11} = 2048$ cells. In the right part of Fig. 8, only the leaves of the mesh are represented. The phenomenon observed here is a linear propagation of a contact discontinuity in the right direction, the diffusivity changing the initial discontinuity into a sharp, but continuous, slope. The Peclet number is here the control parameter.

First, we can notice that, for the chosen tolerance $\epsilon = 10^{-3}$, the last level $L = 11$ is never required. Then, the highest level is reached around the steep gradient region, which proves that the multiresolution method automatically detects the region where small scales are necessary and tracks the propagation phenomenon. In Fig. 9, the time evolution of $\mathscr{L}^\infty$- and $\mathscr{L}^1$-errors are plotted for both finite volume and multiresolution computations. We clearly see that the $\mathscr{L}^\infty$-error decreases with time for the finite volume computation, whereas it slightly increases for the multiresolution computation. The difference between both curves also increases for the $\mathscr{L}^1$-error, although it is not so visible. As a consequence, the *perturbation error* accumulates in time, which confirms the theoretical result in (26) for the $\mathscr{L}^1$-error. This result is also satisfied numerically for the $\mathscr{L}^\infty$-error in this case.

5.1.2. Influence of maximal level and tolerance on CPU time, memory and errors at $Pe = 1000$

In this part, CPU time, memory requirements, $\mathscr{L}^\infty$- and $\mathscr{L}^1$-errors are given for different maximal levels and tolerances. CPU time and memory requirements are compared to the ones obtained by the finite

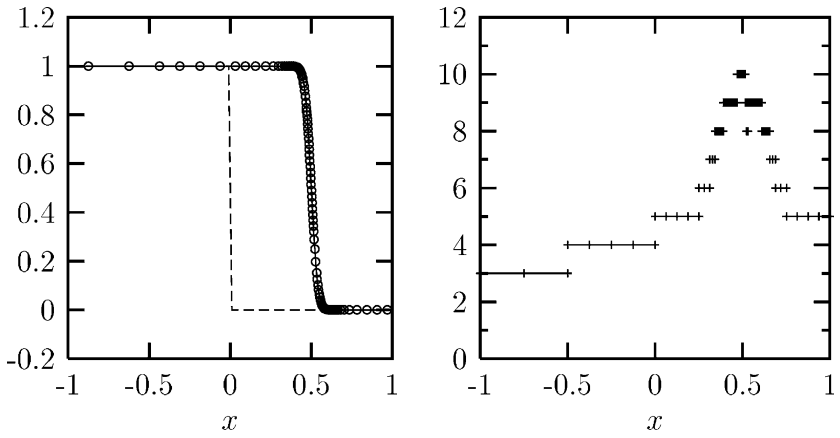

Fig. 8. Left: Initial solution (*dashed*), analytic solution (*plain*), and computed points by multiresolution (*circles*) at $t = 0.5$ for the convection–diffusion equation $Pe = 1000$, $L = 11$, $\epsilon = 10^{-3}$. Right: corresponding tree structure at $t = 0.5$.

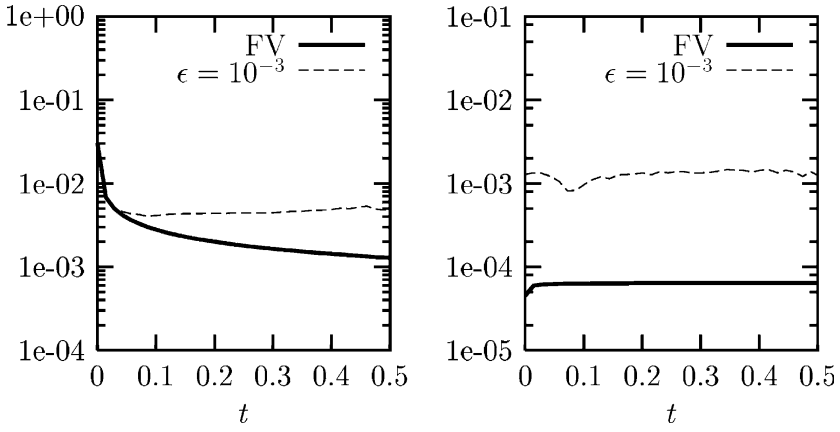

Fig. 9. Errors $\|\bar{u} - \bar{u}_{ex}\|_\infty$ (*left*) and $\|\bar{u} - \bar{u}_{ex}\|_1$ (*right*) from $t = 0$ to $t = 0.5$ for the convection–diffusion equation $Pe = 1000$, $L = 11$, $\epsilon = 10^{-3}$.

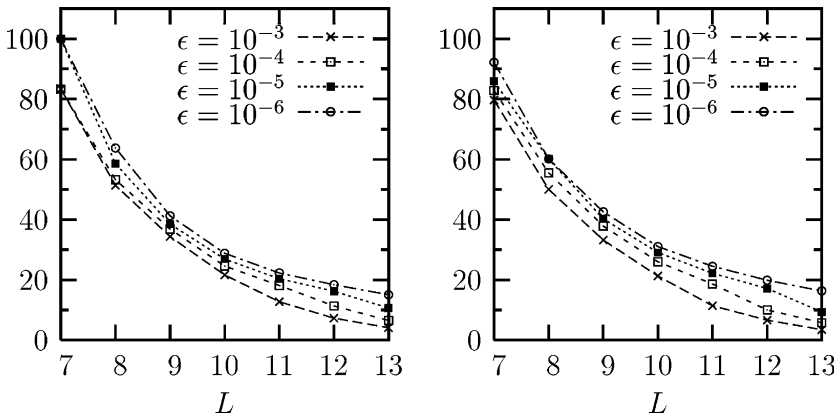

Fig. 10. Percentage of CPU compression (*left*) and percentage of memory compression (*right*) for different scales L and for different tolerances ϵ at $t = 0.2$ for the convection–diffusion equation, $Pe = 1000$.

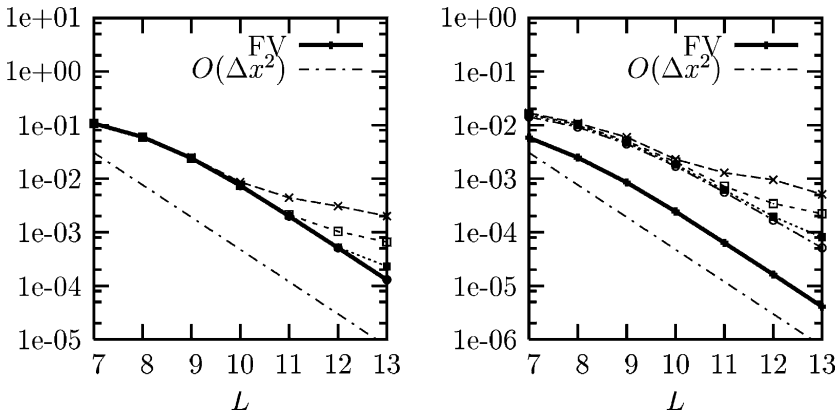

Fig. 11. Errors $\|\bar{u} - \bar{u}_{ex}\|_\infty$ (*left*) and $\|\bar{u} - \bar{u}_{ex}\|_1$ (*right*) for different scales L and different tolerances ϵ at $t = 0.2$ for the convection–diffusion equation, $Pe = 1000$. For the missing legend, see Fig. 10.

volume method with the same numerical schemes and on the finest regular grid. It is expressed in percent (Fig. 10). The corresponding errors are given in Fig. 11.

We first notice that, for such a configuration and for $L > 7$, the multiresolution computation is always cheaper in CPU time and memory requirements than the finite volume method on the finest grid. However, the advantage is only significant for a large L, whatever the tolerance. Fig. 11 shows that, as expected, a computation cannot be second-order accurate for small scales and big tolerances. As shown in Section 3.3, the accumulated *perturbation error* is responsible for this loss of accuracy.

5.1.3. Dependency of CPU time, memory and error on maximal level with the reference tolerance at Pe = 1000

In this part, we verify that multiresolution computations with the reference tolerance defined in (28)

$$\epsilon_R = C\frac{2^{-(\alpha+1)L}}{Pe + 2^{L+2}} \tag{33}$$

enables us to maintain the same second-order accuracy as for the finite volume method on the finest regular grid, while reducing CPU time and memory requirements. To determine the constant C, we refer to the results obtained in the previous part for different tolerances (Fig. 11). Reading both curves, we consider as sufficiently accurate the computation performed with $Pe = 1000$, $L = 10$, and $\epsilon = 10^{-4}$. Thus we find a factor $C = 5 \times 10^8$. We observe the second-order accuracy of the multiresolution computation (Fig. 13), while CPU time and memory requirements decrease with L and reach a minimum around 20% for high levels (Fig. 12).

We also observe an insignificant difference between finite volume and multiresolution computations in the $\mathscr{L}^\infty$-error curve of Fig. 13. This is due to the fact that the maximum of the error is in the region of the steep gradient, where the discretization error is dominating and the perturbation error is negligible, given that all available scales are used there.

5.1.4. Dependency of CPU time, memory and error on maximal level with the reference tolerance at Pe = 10,000

To evaluate the influence of the Peclet number in the estimation of ϵ_R, we now perform computations with $Pe = 10{,}000$. The diffusion phenomenon is less important than in the previous case, and therefore the gradient will be steeper. As before, we observe the second-order accuracy of multiresolution computations (Fig. 15), while CPU time and memory compressions decrease with L and seem to reach a minimum around 10% for high levels (Fig. 14). Due to the fact that there is less diffusion, both CPU time and memory performances are better than for $Pe = 1000$.

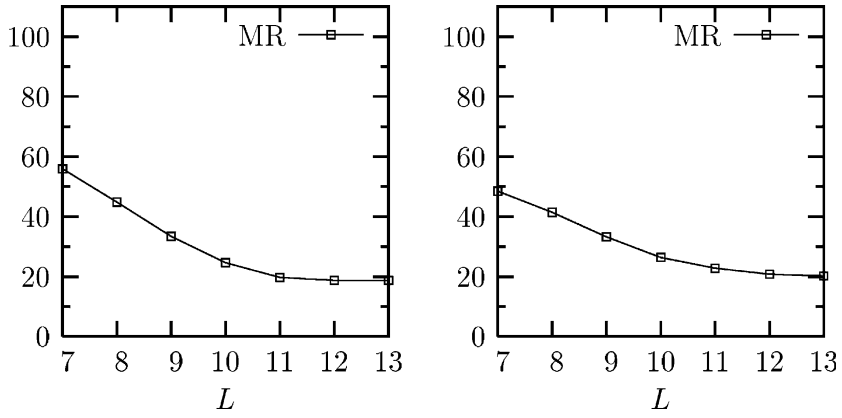

Fig. 12. Percentage of CPU compression (*left*) and percentage of memory compression (*right*) for different scales L and for the reference tolerance ϵ_R at $t = 0.2$ for the convection–diffusion equation, $Pe = 1000$.

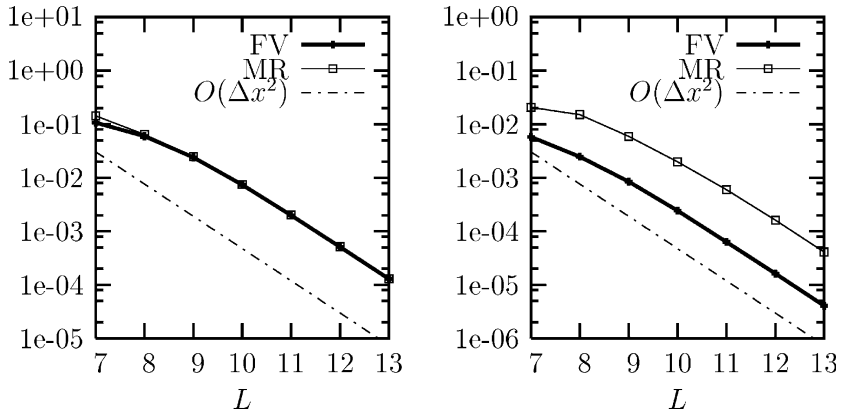

Fig. 13. Errors $\|\bar{u} - \bar{u}_{ex}\|_\infty$ (*left*) and $\|\bar{u} - \bar{u}_{ex}\|_1$ (*right*) for different scales L and the reference tolerance ϵ_R at $t = 0.2$ for the convection–diffusion equation, $Pe = 1000$.

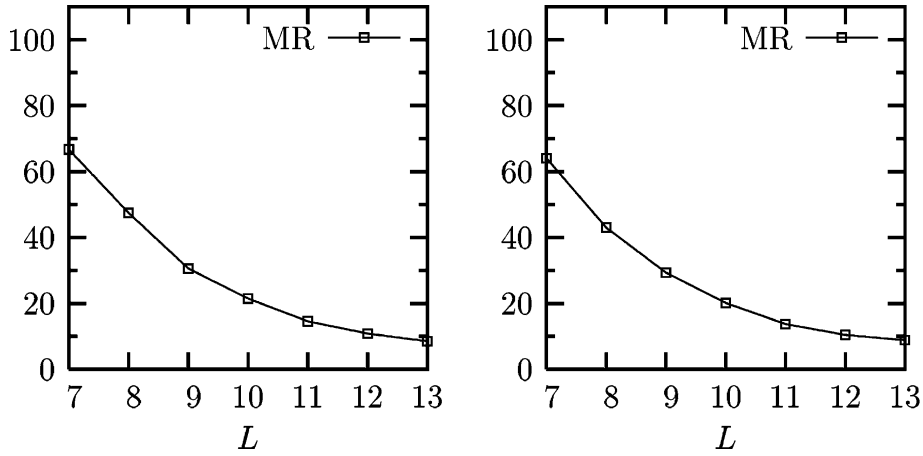

Fig. 14. Percentage of CPU compression (*left*) and percentage of memory compression (*right*) for different scales L and the reference tolerance ϵ_R at $t = 0.2$ for the convection–diffusion equation, $Pe = 10000$.

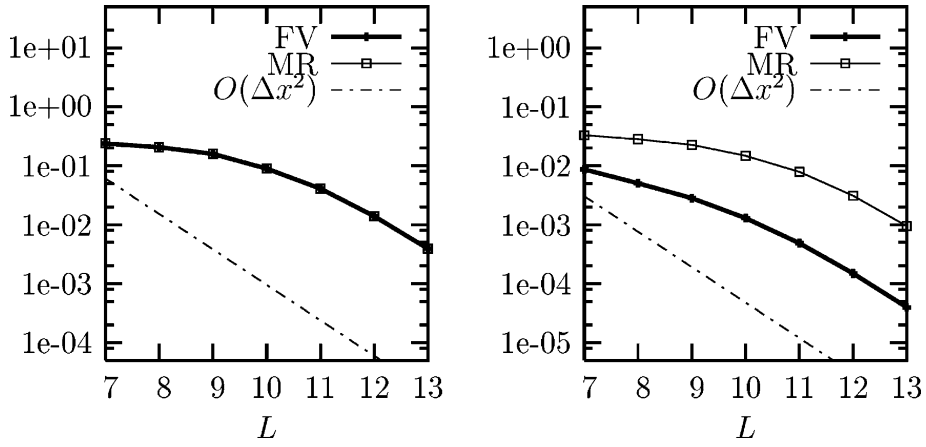

Fig. 15. Errors $\|\bar{u} - \bar{u}_{ex}\|_\infty$ (*left*) and $\|\bar{u} - \bar{u}_{ex}\|_1$ (*right*) for different scales L and the reference tolerance ϵ_R at $t = 0.2$ for the convection–diffusion equation, $Pe = 10000$.

As in Section 5.1.3 and for the same reason, there is no visible difference between finite volume and multiresolution computations in the $\mathscr{L}^\infty$-error curve of Fig. 15.

5.2. 1D viscous Burgers equation

We now perform multiresolution computations for the viscous Burgers equation, which contains a non-linear convective term, and for which analytic solutions are known. For $(x,t) \in [-1,1] \times [0,+\infty)$, it can be written in the dimensionless form

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}, \tag{34}$$

where Re is the Reynolds number, with the initial condition

$$u_0(x) = \begin{cases} 1 & \text{if } x \leqslant 0, \\ 0 & \text{if } x > 0 \end{cases}$$

and Dirichlet conditions at the right and left boundaries, i.e.,

$$u(0,t) = 1,$$
$$u(1,t) = 0.$$

The analytic solution is given in [27]

$$u_{ex}(x,t) = \frac{1}{2}\left[1 - \tanh\left(\left(x - \frac{t}{2}\right)\frac{Re}{4}\right)\right]. \tag{35}$$

5.2.1. Numerical results for a given scale and a given tolerance at $Re = 1000$

The numerical solution of (34) at $t = 0.5$ is given in Fig. 16 for $Re = 1000$, $\epsilon = 10^{-3}$ and $L = 11$ scales, which corresponds to a maximum of $2^{11} = 2048$ cells. As for the convection–diffusion computation, only

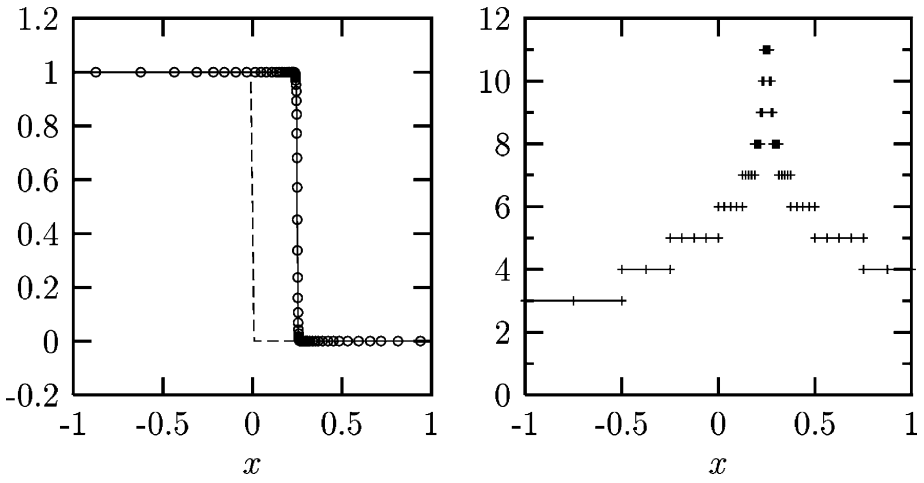

Fig. 16. Left: Initial solution (*dotted*), analytic solution (*plain*), and computed points by multiresolution (*circles*) at $t = 0.5$ for the viscous Burgers equation $Re = 1000$, $L = 11$, $\epsilon = 10^{-3}$. Right: corresponding tree structure at $t = 0.5$.

the leaves of the mesh are represented in Fig. 16 (*right*). We observe here a non-linear propagation of a "shock" in the right direction, the diffusivity having the same effect on the discontinuity as in the linear case.

We notice that, this time, all available scales are used, given that the gradient is steeper than in the linear case. The time evolution of the errors between computed and analytic solutions are depicted in Fig. 17. This time, one gets the same $\mathscr{L}^\infty$-error as with the finite volume method on the finest grid. Therefore the choice for ϵ is well adapted.

5.2.2. Dependency of CPU time, memory and error on maximal level with the reference tolerance at Re = 1000

We repeat the computations performed for the convection–diffusion equation with the same reference tolerance ϵ_R, which is this time

$$\epsilon_R = C\frac{2^{-(\alpha+1)L}}{Re + 2^{L+2}}. \tag{36}$$

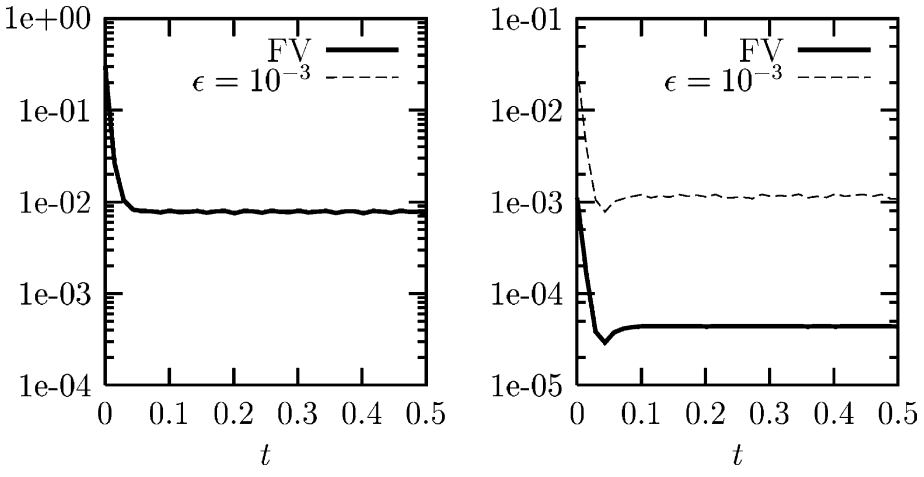

Fig. 17. Errors $\|\bar{u} - \bar{u}_{ex}\|_\infty$ (*left*) and $\|\bar{u} - \bar{u}_{ex}\|_1$ (*right*) for the viscous Burgers equation $Re = 1000$, $t = 0.5$, $L = 11$, $\epsilon = 10^{-3}$.

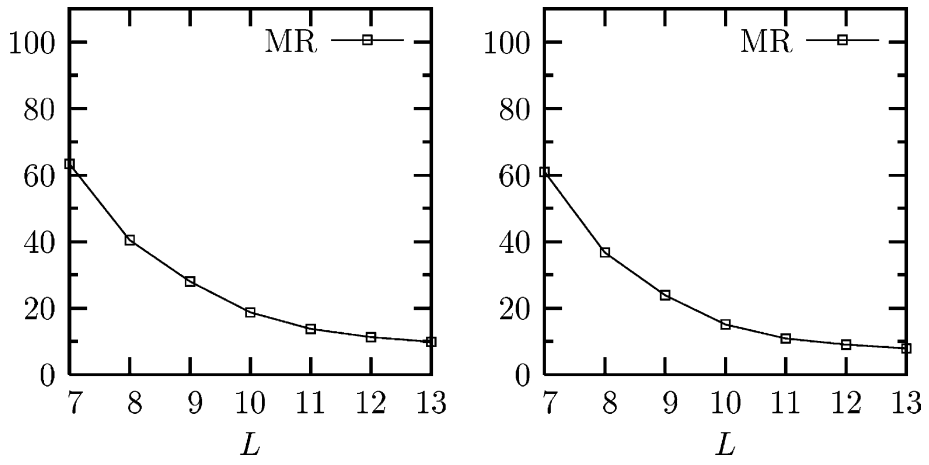

Fig. 18. Percentage of CPU compression (*left*) and percentage of memory compression (*right*) for different scales L and for the reference tolerance ϵ_R for the viscous Burgers equation $Re = 1000$, $t = 0.2$.

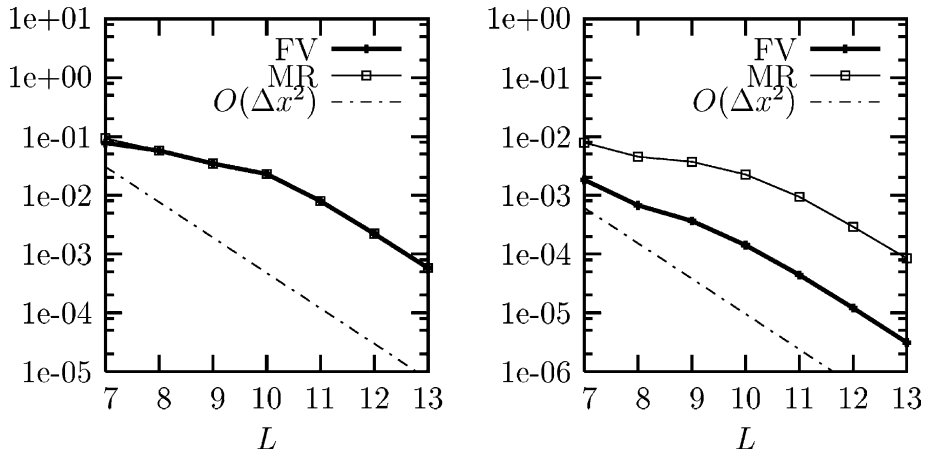

Fig. 19. Errors $\|\bar{u} - \bar{u}_{ex}\|_\infty$ (*left*) and $\|\bar{u} - \bar{u}_{ex}\|_1$ (*right*) for different scales L and for the reference tolerance ϵ_R for the viscous Burgers equation $Re = 1000$, $t = 0.2$.

The same value for the factor C is used as in the linear case. We notice the second-order accuracy of the computation ($\mathscr{L}^\infty$ and $\mathscr{L}^1$-errors in Fig. 19). The CPU time and memory compression are decreasing with L and reaching a minimum around 10% for the highest levels (Fig. 18).

We finally remark that the chosen finite volume scheme conserves both momentum and energy. Due to the choice made for flux computations, this is also the case for the adaptive multiresolution scheme. Nevertheless, in the adaptive case, the difference between exact and computed momentum and energy shows small oscillations of amplitude lower than 10^{-5}. This is due to the remeshing of the grid at each time step.

5.3. 1D reaction–diffusion equation

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. It can be written in its dimensionless form, for $(x, t) \in [0, 20] \times [0, +\infty)$,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + S(u), \tag{37}$$

$$S(u) = \frac{\beta^2}{2}(1-u)\exp\frac{\beta(1-u)}{\alpha(1-u)-1}, \tag{38}$$

where α is the temperature ratio and β is the dimensionless activation energy (Zeldovich number). We choose as initial condition

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

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 Neuman condition at the left boundary and a Dirichlet condition at the right boundary.

$$\begin{aligned} &\frac{\partial u}{\partial x}(0,t) = 0, \\ &u(20,t) = 0. \end{aligned} \tag{40}$$

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

In Fig. 20, we observe the *flame propagation* in the x-direction. The highest level is reached in the region of the reaction zone, ie. 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

$$v_f = \int_\Omega S \, dx \tag{41}$$

with the asymptotic value given in Peters and Warnatz [33] (Table 1). We observe that the value of v_f is approximately the same for finite volume and multiresolution computations, for the three different values of

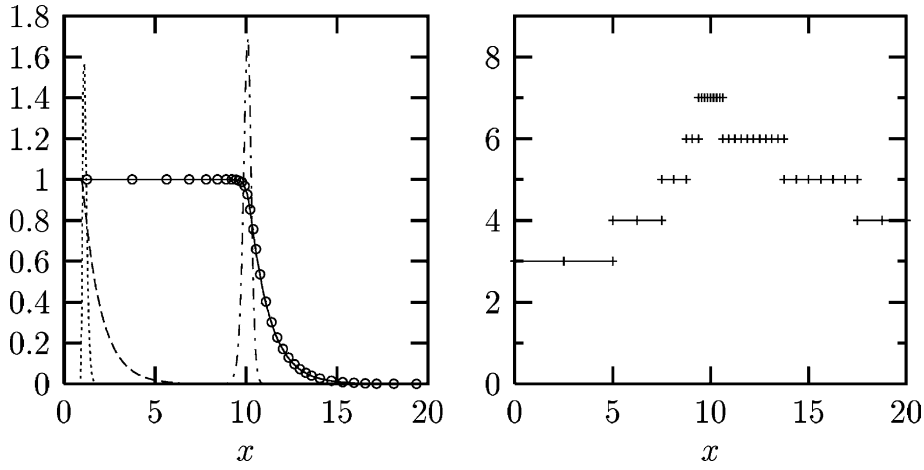

Fig. 20. Left: Initial condition for u (*dashed*) and $S(u)$ (*dotted*), solution by finite volume method for u(*plain*), solution by multiresolution method for u (*circles*) and $S(u)$ (*dash-dotted*) at $t = 10$ for the reaction-diffusion equation, $\alpha = 0.8$, $\beta = 10$, $L = 8$, $\epsilon = 5.10^{-2}$. Right: corresponding tree structure at $t = 10$.

Table 1
Flame velocity, CPU and memory compression for finite volume and multiresolution methods

Method	v_f	% CPU	% Mem
FV	0.916	100.0	100.0
MR $\epsilon = 5 \times 10^{-2}$	0.917	36.0	32.6
MR $\epsilon = 10^{-2}$	0.916	54.2	47.1
MR $\epsilon = 10^{-3}$	0.916	79.0	67.2
Asymptotic	0.908		

the tolerance. All these values are comparable with the asymptotic one. Hence we can conclude that the value $\epsilon = 5 \times 10^{-2}$ is well adapted.

5.4. 2D convection–diffusion equation

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 $(x, y, t) \in [-5, 5]^2 \times [0, +\infty)$

$$\frac{\partial u}{\partial t} + \mathbf{V} \cdot \nabla u = \frac{1}{Pe} \nabla^2 u \tag{42}$$

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

$$u(x, y, t) = \frac{Pe}{4\pi t} \, e^{-Pe(((x-t)^2+y^2)/4t)}. \tag{43}$$

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

$$u_0(x, y) = \frac{Pe}{4\pi\tau} \, e^{-Pe((x^2+y^2)/4\tau)}$$

we get the analytic solution

$$u(x, y, t) = \frac{Pe}{4\pi(t+\tau)} \, e^{-Pe(((x-t)^2+y^2)/4(t+\tau))}. \tag{44}$$

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

5.4.1. Numerical results for a given scale and a given tolerance at $Pe = 1000$

The numerical solution of (42) for an initial Gaussian bump is represented in Fig. 21 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.

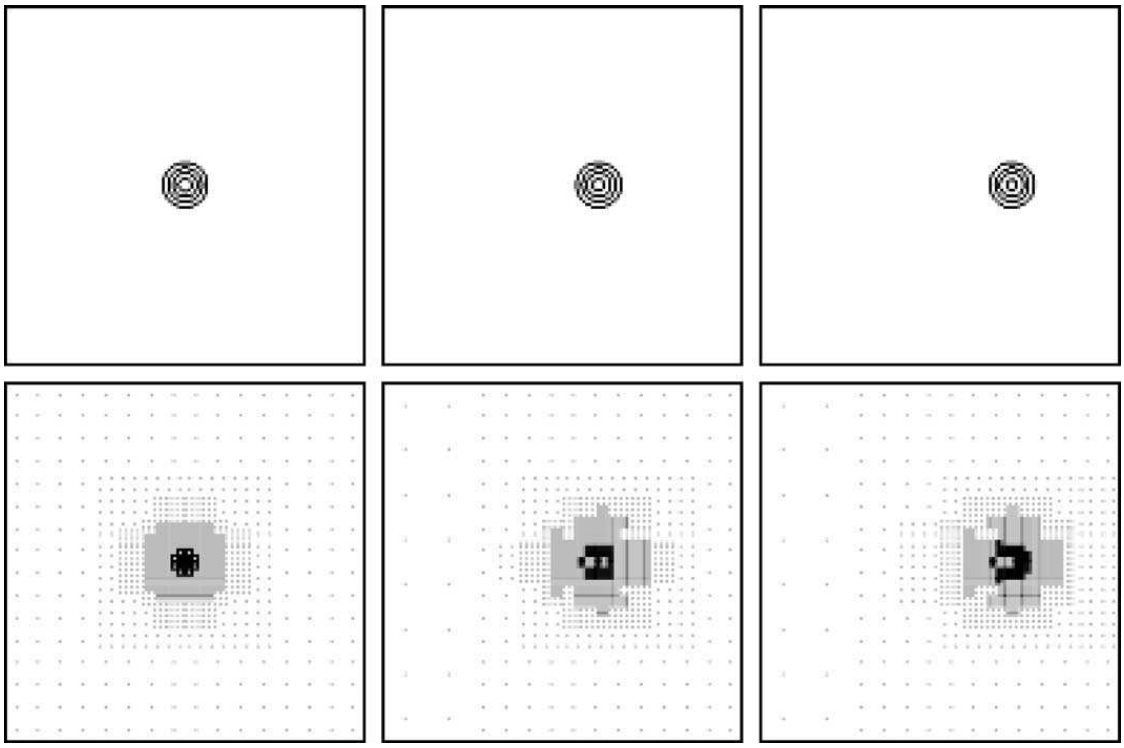

Fig. 21. Isolines $u = 0.3$, 0.5, 0.7 and 0.9 (*top*) and corresponding mesh (*bottom*) for the 2D convection–diffusion equation at $t = 0$ (*left*), $t = 1$ (*middle*), and $t = 2$ (*right*).

5.4.2. 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., $C = 5 \times 10^8$. 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. 22). This time, as no

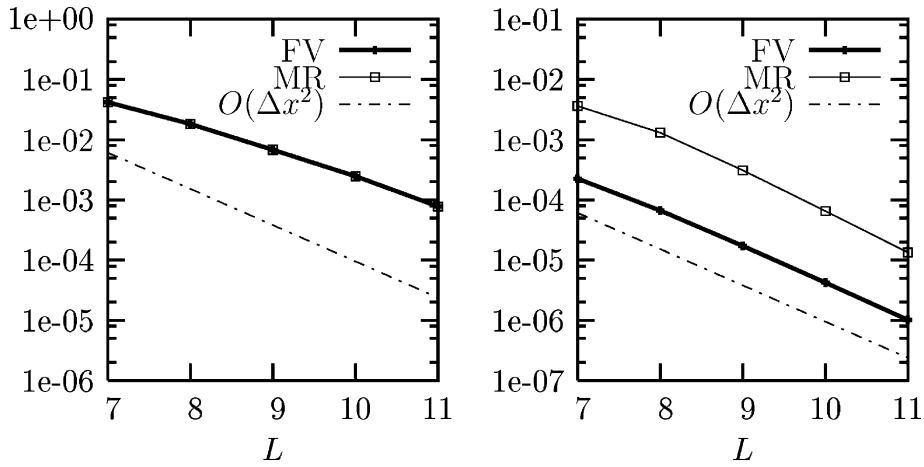

Fig. 22. Errors $\|\bar{u} - \bar{u}_{ex}\|_\infty$ (*left*) and $\|\bar{u} - \bar{u}_{ex}\|_1$ (*right*) for different scales L and the reference tolerance ϵ_R for the 2D convection–diffusion equation $Pe = 1000$, $t = 0.5$.

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.

5.5. 2D reaction–diffusion equation

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 [23]. The resulting equation in the dimensionless form is

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + S(u), \tag{45}$$

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 perturbate 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

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

We consider that the reaction takes place in a closed box with adiabatic walls, and hence we choose Neuman 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 $\epsilon = 5 \times 10^{-2}$, like in the 1D case.

As in Fröhlich and Schneider [23], we observe a relaxation of the elliptic flame towards a circularly symmetric structure which is then growing in space (Fig. 23). The finest resolution would correspond to $(2^8)^2 = 256^2$ cells. On average we only use 6763 out of $256^2 = 65{,}536$ 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%.

5.6. 3D reaction–diffusion equation

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 $(x, y, z, t) \in [-20, 20]^3 \times [0, +\infty)$

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + S(u), \tag{46}$$

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

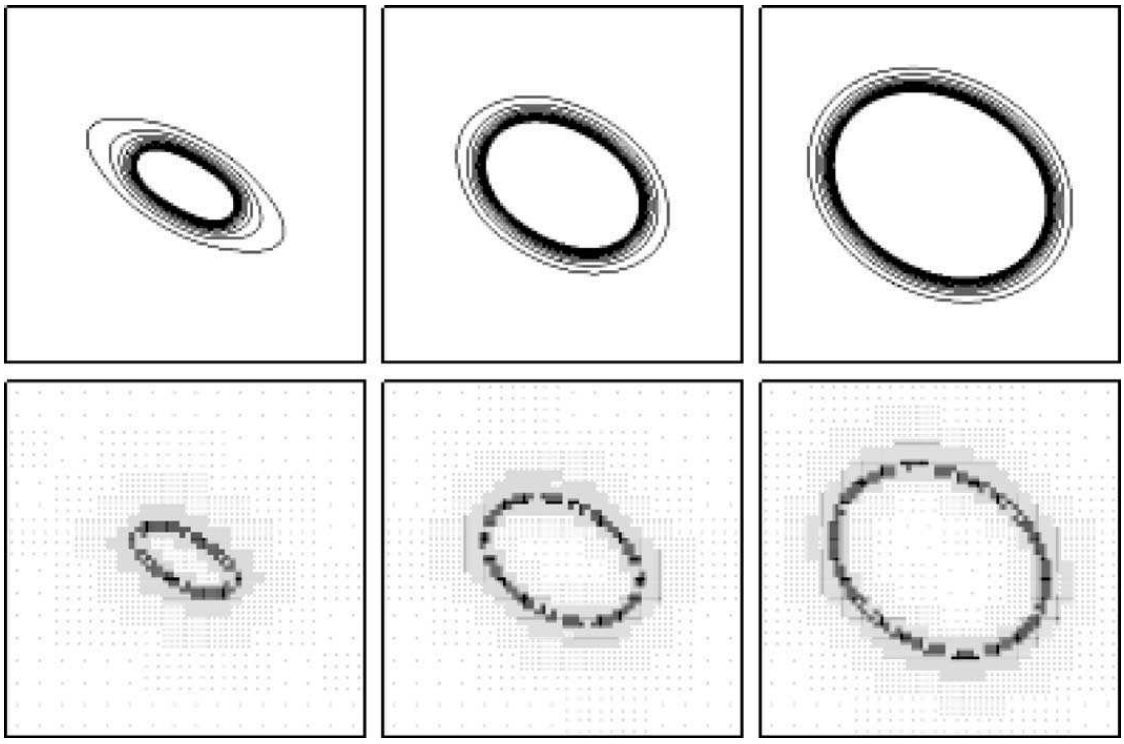

Fig. 23. Isolines $u = 0.1$ to 1 and corresponding mesh at $t = 2$ (*left*), $t = 6$ (*middle*) and $t = 10$ (*right*) for the 2D reaction–diffusion equation.

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

where

$$\begin{aligned} X &= x\cos\theta + y\sin\theta, \\ Y &= -x\sin\theta + y\cos\theta, \\ Z &= z. \end{aligned}$$

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

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

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 (Fig. 24). The finest resolution would correspond to $(2^7)^3 = 128^3$ cells. On average we only use 39,636 out of $128^3 = 2{,}097{,}152$ control volumes, which yields a memory compression of 1.89%. Comparing the elapsed

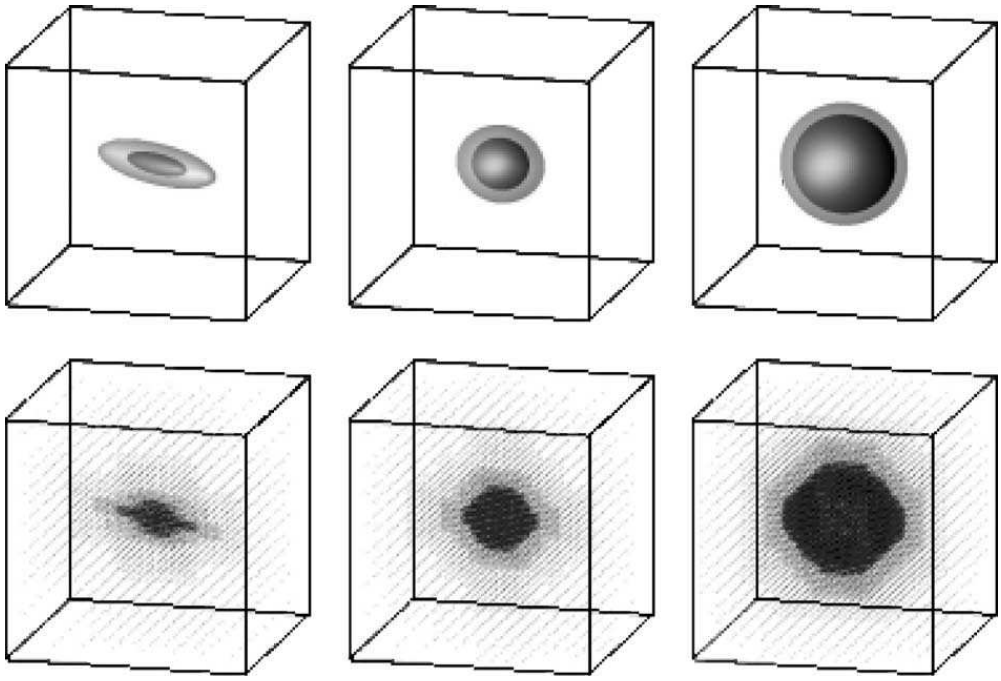

Fig. 24. Isosurfaces $u = 0.5$ (*black*), 0.1 (*gray*) and corresponding mesh at $t = 0$ (*left*), $t = 6$ (*middle*) and $t = 12$ (*right*) for the 3D reaction–diffusion equation.

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 [36].

6. Conclusion

In the present paper we developed a new fully adaptive numerical scheme to speed up finite volume computations of nonlinear parabolic PDEs in Cartesian geometry, in one, two and three space dimensions. We demonstrated its computational efficiency and the numerical accuracy by computing several test-cases of linear and non-linear parabolic PDEs.

Starting point of the method is a finite volume discretization on a regular equidistant grid, together with an explicit time integration, both of second order. 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 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 guarantee the second-order accuracy, which is verified numerically. The adaptive algorithm is implemented using a graded tree data structure 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.

The accuracy of the algorithm has been validated by solving convection–diffusion and viscous Burgers equations. We compared the computed solutions with the exact ones and studied the error as a function of the maximal level and the prescribed tolerance. The presented theoretical relation for the level dependent tolerance to maintain the second order accuracy has been confirmed by our computations.

To demonstrate the efficiency of the algorithm, we have compared the performance in terms of CPU time and memory requirements to a finite volume method using the same numerical schemes on the finest regular grid with a static data structure. We have shown 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.

Finally, we presented several applications to combustion problems, i.e., thermo-diffusive flame fronts and 2D and 3D flame balls. Solving reaction-diffusion equations in one, two and three dimensions we have shown that the adaptive algorithm can be efficiently used to solve stiff nonlinear problems with reduced CPU and memory requirements (see Table 1).

Current work is dealing with the parallel implementation of the algorithm on a PC cluster 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 the adaptive simulation of pulsating flames for large activation energies and slowly diffusing reactants and to study the instability behavior of flame balls in the fully nonlinear regime. We also plan to extend the developed scheme to systems of reactive Navier–Stokes equations, in order to take into account hydrodynamic effects in combustion problems and to use the CVS (Coherent Vortex Simulation) approach [20] to simulate and to model turbulent reactive flows on adaptive grids. A complementary direction is the use of implicit time discretization for the diffusive terms (see Figs. 23 and 24).

Acknowledgements

The authors would like to thank Albert Cohen, Marie Postel and Sidi-Mahmoud Kaber for their valuable advice related to fully adaptive multiresolution schemes, Wolfgang Dahmen and Siegfried Müller for constructive discussions and fruitful comments on the paper, and finally Bastien Pellicioli for his help in numerical computations. This work was partially supported by the Commission of the European Communities Contract FMRX-CT98-0184 TMR Project "Wavelets in Numerical Simulation", and by the French–German–Russian Trilateral Project (Network No. 007).

Appendix A

In the appendix we devise the explicit formulae of the prediction operator for the linear polynomial interpolation in the 2D and 3D cases. For the 2D case, Bihari and Harten [7] obtained the following values using a tensor product approach. For $n,p \in \{0,1\}$, we have

$$\begin{aligned}\hat{\boldsymbol{u}}_{l+1,2i+n,2j+p} &= I(\bar{U}_l; l+1, 2i+n, 2j+p) \\ &= \bar{\boldsymbol{u}}_{l,i,j} - (-1)^n Q_x^s(\bar{U}_l; i, j) - (-1)^p Q_y^s(\bar{U}_l; i, j) + (-1)^{np} Q_{xy}^s(\bar{U}_l; i, j), \end{aligned} \tag{A.1}$$

where

$$Q_x^s(\bar{U}_l; i, j) = \sum_{n=1}^{s} \gamma_n \left(\bar{\boldsymbol{u}}_{l,i+n,j} - \bar{\boldsymbol{u}}_{l,i-n,j} \right),$$

$$Q_y^s(\bar{U}_l;i,j)=\sum_{p=1}^{s}\gamma_p\Big(\bar{\boldsymbol{u}}_{l,i,j+p}-\bar{\boldsymbol{u}}_{l,i,j-p}\Big),$$

$$Q_{xy}^s(\bar{U}_l;i,j)=\sum_{n=1}^{s}\gamma_n\sum_{p=1}^{s}\gamma_p\Big(\bar{\boldsymbol{u}}_{l,i+n,j+p}-\bar{\boldsymbol{u}}_{l,i+n,j-p}-\bar{\boldsymbol{u}}_{l,i-n,j+p}+\bar{\boldsymbol{u}}_{l,i-n,j-p}\Big).$$

We apply the same method to get the prediction in the 3D case. Thus, for $n,p,q\in\{0,1\}$, we have

$$\begin{aligned}\hat{\boldsymbol{u}}_{l+1,2i+n,2j+p,2k+q}&=I(\bar{U}_l;l+1,2i+n,2j+p,2k+q)\\&=\bar{\boldsymbol{u}}_{l,i,j,k}-(-1)^nQ_x^s(\bar{U}_l;i,j,k)-(-1)^pQ_y^s(\bar{U}_l;i,j,k)-(-1)^qQ_z^s(\bar{U}_l;i,j,k)\\&\quad+(-1)^{np}Q_{xy}^s(\bar{U}_l;i,j,k)+(-1)^{pq}Q_{yz}^s(\bar{U}_l;i,j,k)+(-1)^{nq}Q_{xz}^s(\bar{U}_l;i,j,k)\\&\quad-(-1)^{npq}Q_{xyz}^s(\bar{U}_l;i,j,k),\end{aligned}\tag{A.2}$$

where

$$Q_x^s(\bar{U}_l;i,j,k)=\sum_{n=1}^{s}\gamma_n\Big(\bar{\boldsymbol{u}}_{l,i+n,j,k}-\bar{\boldsymbol{u}}_{l,i-n,j,k}\Big),$$

$$Q_y^s(\bar{U}_l;i,j,k)=\sum_{p=1}^{s}\gamma_p\Big(\bar{\boldsymbol{u}}_{l,i,j+p,k}-\bar{\boldsymbol{u}}_{l,i,j-p,k}\Big),$$

$$Q_z^s(\bar{U}_l;i,j,k)=\sum_{q=1}^{s}\gamma_q\Big(\bar{\boldsymbol{u}}_{l,i,j,k+q}-\bar{\boldsymbol{u}}_{l,i,j,k-q}\Big),\tag{A.3}$$

$$Q_{xy}^s(\bar{U}_l;i,j,k)=\sum_{n=1}^{s}\gamma_n\sum_{p=1}^{s}\gamma_p\Big(\bar{\boldsymbol{u}}_{l,i+n,j+p,k}-\bar{\boldsymbol{u}}_{l,i+n,j-p,k}-\bar{\boldsymbol{u}}_{l,i-n,j+p,k}+\bar{\boldsymbol{u}}_{l,i-n,j-p,k}\Big),$$

$$Q_{yz}^s(\bar{U}_l;i,j,k)=\sum_{p=1}^{s}\gamma_p\sum_{q=1}^{s}\gamma_q\Big(\bar{\boldsymbol{u}}_{l,i,j+p,k+q}-\bar{\boldsymbol{u}}_{l,i,j+p,k-q}-\bar{\boldsymbol{u}}_{l,i,j-p,k+q}+\bar{\boldsymbol{u}}_{l,i,j-p,k-q}\Big),$$

$$Q_{xz}^s(\bar{U}_l;i,j,k)=\sum_{n=1}^{s}\gamma_n\sum_{q=1}^{s}\gamma_q\Big(\bar{\boldsymbol{u}}_{l,i+n,j,k+q}-\bar{\boldsymbol{u}}_{l,i+n,j,k-q}-\bar{\boldsymbol{u}}_{l,i-n,j,k+q}+\bar{\boldsymbol{u}}_{l,i-n,j,k-q}\Big),$$

$$\begin{aligned}Q_{xyz}^s(\bar{U}_l;i,j,k)=\sum_{n=1}^{s}\gamma_n\sum_{p=1}^{s}\gamma_p\sum_{q=1}^{s}\gamma_q\Big(&\bar{\boldsymbol{u}}_{l,i+n,j+p,k+q}-\bar{\boldsymbol{u}}_{l,i+n,j+p,k-q}-\bar{\boldsymbol{u}}_{l,i+n,j-p,k+q}-\bar{\boldsymbol{u}}_{l,i-n,j+p,k+q}+\bar{\boldsymbol{u}}_{l,i+n,j-p,k-q}\\&+\bar{\boldsymbol{u}}_{l,i-n,j+p,k-q}+\bar{\boldsymbol{u}}_{l,i-n,j-p,k+q}-\bar{\boldsymbol{u}}_{l,i-n,j-p,k-q}\Big).\end{aligned}$$

As in the 1D case, the multiresolution accuracy r is related to the number of required nearest uncles by $r=2s+1$. The corresponding coefficients γ_n for $r=3$ and $r=5$ are

$$r = 3 \;\Rightarrow\; \gamma_1 = -\frac{1}{8},$$
$$r = 5 \;\Rightarrow\; \gamma_1 = -\frac{22}{128},\ \gamma_2 = \frac{3}{128}.$$

References

[1] R. Abgrall, A. Harten, Multiresolution representation in unstructured meshes, SIAM J. Numer. Anal. 35 (6) (1998) 2128–2146.
[2] J. Bell, M.J. Berger, J. Saltzman, M. Welcome, Three-dimensional adaptive mesh refinement for hyperbolic conservation laws, SIAM J. Sci. Comput. 15 (1994) 127–138.
[3] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 67–84.
[4] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[5] G. Beylkin, R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms I, Comm. Pure Appl. Math. 44 (1991) 141–183.
[6] B.L. Bihari, Multiresolution schemes for conservation laws with viscosity, J. Comput. Phys. 123 (1996) 207–225.
[7] B.L. Bihari, A. Harten, Multiresolution schemes for the numerical solution of 2-D conservation laws I, SIAM J. Sci. Comput. 18 (2) (1997) 315–354.
[8] H. Bockhorn, J. Fröhlich, K. Schneider, An adaptive two-dimensional wavelet–vaguelette algorithm for the computation of flame balls, Combust. Theory Model. 3 (1999) 1–22.
[9] A. Brandt, Multi-level adaptive solutions to boundary value problems, Math. Comp. 31 (1977) 333–390.
[10] P. Charton, V. Perrier, A pseudo-wavelet scheme for the two-dimensional Navier–Stokes equations, Mat. Apl. Comput. 15 (1996) 139–160.
[11] G. Chiavassa, R. Donat, Point value multi-scale algorithms for 2D compressible flows, SIAM J. Sci. Comput. 23 (3) (2001) 805–823.
[12] A. Cohen, Wavelet methods in numerical analysis, in: P.G. Ciarlet, J.L. Lions (Eds.), Handbook of Numerical Analysis, Elsevier, Amsterdam, 2000.
[13] A. Cohen, Adaptive methods for PDE's – Wavelets or mesh refinement? in: International Conference of Mathematics, Beijing, 2002.
[14] A. Cohen, N. Dyn, S.M. Kaber, M. Postel, Multiresolution finite volume schemes on triangles, J. Comput. Phys. 161 (2000) 264–286.
[15] A. Cohen, S.M. Kaber, S. Müller, M. Postel, Fully adaptive multiresolution finite volume schemes for conservation laws, Math. Comp. 72 (2002) 183–225.
[16] W. Dahmen, Wavelet and multiscale methods for operator equations, Acta Numer. 6 (1997) 55–228.
[17] W. Dahmen, B. Gottschlich-Müller, S. Müller, Multiresolution schemes for conservation laws, Numer. Math. 88 (3) (2001) 399–443.
[18] W. Dahmen, A. Kunoth, Multilevel preconditioning, Numer. Math. 63 (1992) 315–344.
[19] R. DeVore, Nonlinear approximation, Acta Numer. 7 (1998) 51–150.
[20] M. Farge, K. Schneider, Coherent Vortex Simulation (CVS), a semi-deterministic turbulence model using wavelets, Flow Turbul. Combust. 66 (4) (2001) 393–426.
[21] J. Fröhlich, K. Schneider, An adaptive wavelet Galerkin algorithm for one-and two-dimensional flame computations, Eur. J. Mech. B/Fluids 13 (4) (1994) 439–471.
[22] J. Fröhlich, K. Schneider, Numerical simulation of decaying turbulence in an adaptive wavelet basis, Appl. Comput. Harm. Anal. 3 (1996) 393–397.
[23] J. Fröhlich, K. Schneider, An adaptive wavelet–vaguelette algorithm for the solution of PDEs, J. Comput. Phys. 130 (1997) 174–190.
[24] B. Gottschlich-Müller, S. Müller, Adaptive finite volume schemes for conservation laws based on local multiresolution techniques, in: M. Fey, R. Jeltsch (Eds.), Hyperbolic Problems: Theory, Numerics, Applications, Birkhäuser, 1999, pp. 385–394.
[25] A. Harten, Multiresolution algorithms for the numerical solution of hyperbolic conservation laws, Comm. Pure Appl. Math. 48 (1995) 1305–1342.
[26] A. Harten, Multiresolution representation of data: a general framework, SIAM J. Numer. Anal. 33 (3) (1996) 1205–1256.
[27] C. Hirsch, in: Numerical Computation of Internal and External Flows, vol. 2, Wiley, 1990.
[28] S. Jaffard, Wavelet methods for fast resolution of elliptic problems, SIAM J. Numer. Anal. 29 (1992) 965–986.
[29] J. Liandrat, P. Tchamitchian, Resolution of the 1D regularized Burgers equation using a spatial wavelet approximation: algorithms and numerical results, Technical Report No. 90-83, ICASE, 1990.

[30] Y. Maday, V. Perrier, J.C. Ravel, Adaptivité dynamique sur base d'ondelettes pour l'approximation d'équations aux dérivées partielles, C. R. Acad. Sci. Paris, Série I 312 (1991) 405–410.
[31] Y. Maday, J.C. Ravel, Adaptivité par ondelettes: conditions aux limites et dimensions supérieures, C. R. Acad. Sci. Paris, Série I 315 (1992) 85–90.
[32] S. Müller, in: Adaptive multiscale schemes for conservation laws, Lecture Notes in Computational Science and Engineering, vol. 27, Springer, Heidelberg, 2003.
[33] N. Peters, J. Warnatz (Eds.), Numerical methods in laminar flame propagation, Notes on Numerical Fluid Mechanics, vol. 6, Vieweg, 1982.
[34] J.J. Quirk, An adaptive grid algorithm for shock hydrodynamics, PhD thesis, Cranfield Institute of Technology, College of Aeronautics, 1991.
[35] P.L. Roe, Approximate Riemann solvers parameter vectors and difference schemes, J. Comput. Phys. 43 (1981) 357–372.
[36] O. Roussel, K. Schneider, A fully adaptive multiresolution scheme for 3D reaction–diffusion equations, in: R. Herbin, D. Kröner (Eds.), Finite Volumes for Complex Applications, vol. 3, Hermes Penton Science, 2002, pp. 833–840.
[37] K. Schneider, M. Farge, Adaptive wavelet simulation of a flow around an impulsively started cylinder using penalisation, Appl. Comput. Harm. Anal. 12 (2002) 374–380.
[38] K. Schneider, N.K.R. Kevlahan, M. Farge, Comparison of an adaptive wavelet method and nonlinearly filtered pseudospectral methods for two-dimensional turbulence, Theoret. Comput. Fluid Dynamics 9 (1997) 191–206.
[39] K. Urban, A wavelet-Galerkin algorithm for the driven-cavity-Stokes-problem in two space dimensions, in: M. Feistauer, R. Rannacher, K. Kozel (Eds.), Numerical Modelling in Continuum Mechanics, Charles-University Prague, 1995, pp. 278–289.