ESAIM: PROCEEDINGS, February 2007, Vol.16, 181-194
Eric Cancès & Jean-Frédéric Gerbeau, Editors
DOI: 10.1051/proc:2007006

ON SPACE–TIME ADAPTIVE SCHEMES FOR THE NUMERICAL SOLUTION OF PDES *, **, ***

Margarete O. Domingues[1, 2], Olivier Roussel[3] and Kai Schneider[1, 4]

Abstract. A fully adaptive numerical scheme for solving PDEs based on a finite volume discretization with explicit time discretization is presented. The local grid refinement is triggered by a multiresolution strategy which allows to control the approximation error in space. The costly fluxes are evaluated on the adaptive grid only. For automatic time step control a Runge–Kutta–Fehlberg method is used. A dynamic tree data structure allows memory compression and CPU time reduction. For validation different classical test problems are computed. The gain in memory and CPU time with respect to the finite volume scheme on a regular grid is reported and demonstrates the efficiency of the new method.

Résumé. Nous présentons ici une méthode numérique entièrement adaptative pour les EDP, basée sur une discrétisation spatiale en volumes finis et une intégration temporelle explicite de type Runge-Kutta. Une stratégie de type multi-résolution permet d'adapter localement le maillage tout en contrôlant l'erreur d'approximation en espace. Les flux sont évalués sur la grille adaptative uniquement. Une méthode de type Runge-Kutta-Fehlberg est employée afin de choisir automatiquement le pas de temps tout en contrôlant l'erreur d'approximation. Nous proposons en outre une méthode où le pas de temps dépend de l'échelle, afin d'éviter d'utiliser sur tous les niveaux le pas de temps qui garantit la stabilité numérique sur le niveau de grille le plus fin. La structure de données est organisée en arbre graduel, ce qui permet de réduire significativement la place mémoire et le temps de calcul nécessaires. Nous validons ce nouveau schéma numérique à l'aide de différents cas-tests classiques. Nous estimons le gain en place mémoire et en temps de calcul par rapport au même calcul en volumes finis sur la grille la plus fine, afin de montrer l'efficacité de la méthode.

* *The authors thank the CIRM in Marseille for its hospitality and for financial support during the CEMRACS 2005 summer-program where part of the work was carried out.*

** *M.O. Domingues thankfully acknowledges financial support from the European Union project IHP on 'Breaking Complexity' (contract HPRN-CT 2002-00286).*

*** *O. Roussel and K. Schneider acknowledge financial support from the DFG–CNRS Research Program 'LES and CVS of Complex Flows'*

[1] Laboratoire de Modélisation et Simulation Numérique en Mécanique et Génie des Procédés (MSNM-GP), CNRS and Universités d'Aix-Marseille, 38, rue F. Joliot–Curie, 13451 Marseille Cedex 20, France

[2] Laboratório Associado de Computação e Matemática Aplicada (LAC), Instituto Nacional de Pesquisas Espaciais (INPE), Av. dos Astronautas, 1758, 12227-010 São José dos Campos, Brazil

[3] Institut für Technische Chemie und Polymerchemie (TCP), Universität Karlsruhe, Kaiserstr. 12, 76128 Karlsruhe, Germany

[4] Centre de Mathématiques et d'Informatique (CMI), Université de Provence, 39 rue F. Joliot–Curie, 13453 Marseille Cedex 13, France

e-mail: margarete@lac.inpe.br roussel@ict.uni-karlsruhe.de kschneid@cmi.univ-mrs.fr

Contents

1. Introduction

Systems of nonlinear PDEs typically arise from mathematical modeling of different chemical–physical problems encountered in environmental or industrial applications, ranging from meteorology to chemical process engineering.

Frequently, their solutions exhibit a large range of spatial and temporal scales, which however are intermittently distributed in the space–time domain, e.g. in turbulence or in combustion. Advanced numerical discretizations take advantage of this property by introducing some kind of adaptivity in space and time.

This strategy allows to reduce the computational complexity of uniform discretizations while estimating and controlling the error of the solution with respect to the solution computed on the finest regular grid. For the latter the quality of the approximation can be estimated in many cases. Different school of thoughts have been entered upon defining adaptive discretizations, a review is beyond the scope of this indroduction.

Some emerge from *ad hoc* criteria, others are based on more sophisticated *a posteriori* error estimators using control strategies by solving adjoint problems [4,28]. Berger and Oliger [5] introduced adaptive mesh refinement methods, Harten developed first multiresolution based schemes (MR) for conservation laws [19, 20]. Harten's approach guided many other extensions and developments by Cohen el al. [8], Chiavassa and Donat [6], Roussel and coworkers [25, 26]. Starting with a classical discretization either finite volume (FV) or finite differences (FD), the main idea of MR methods is to control the truncation error by estimating the local regularity of the solution. Thresholding the coefficients of a multiresolution representation of the solution allows to retain only few significant coefficients while yielding an error estimate.

For a detailed overview of MR methods we refer to the books of Cohen [7] and Müller [21]. A main bottleneck of most space–adaptive methods using typically explicit or semi–explicit time discretizations is that, for stability reasons, the time step is directly related to the smallest spatial step size. This implies that the larger the number of refinement levels the smaller the size of the time step.

Different attempts have been undertaken to couple adaptive time stepping with adaptive space discretization of PDEs. Osher and Sanders developed local time stepping for one-dimensional scalar conservation laws where the space discretization is non–uniform but fixed [23]. Tang and Warnecke [29] presented recently an extension. Collino, Fouquet and Joly [9, 10] proposed another scheme for space-time refinement based on the conservation of a discrete energy through two different discretization grids. This method is a non-interpolatory scheme whose stability condition is not affected by the transition between the two grids. Ferm and Löstedt [18] proposed a local time stepping for adaptive mesh refinement methods coupled with a Runge–Kutta Fehlberg scheme to choose automatically the size of the time step while controlling the error. Results have been presented for hyperbolic conservation laws (i.e., Burgers, Euler and the wave equation) in one space dimension.

For adaptive multiresolution and wavelet methods a scale dependent time step has been introduced by Bacry and coworkers [2]. They applied this method to linear and non-linear parabolic Burgers equation. More recently,

Müller and Stiriba [22] presented a fully adaptive multiresolution finite volume scheme with locally varying time stepping and a predictor corrector method. Applications for one dimensional conservation laws are discussed to show the efficiency and accuracy of the code and first results for Euler equations in two dimensions are exposed. 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 and coworkers [1]. Results for one space dimension look promising, however the extension of this method to higher dimensions seems less advantageous due to the memory requirement.

The aim of the present paper is to develop an adaptive time stepping scheme with automatic error control for the adaptive MRA scheme presented in [25,26]. The adaptive time integration method is based on a Runge–Kutta–Fehlberg method which allows an estimation of the local error in time. The multiresolution transform automatically detects the local regularity of the solution and hence guarantees automatic grid adaption in space. The costly numerical fluxes are evaluated on this locally refined while ensuring strict conservativity. The implementation uses graded tree data structures which allows an efficient representation of the solution on adaptive grids with reduced memory requirements.

The organization of the paper is as follows: In section 2 the space discretization using finite volumes on both regular and adaptive meshes is summarized. The time discretization using standard compact Runge–Kutta and Runge–Kutta–Fehlberg methods for global time adaptivity is also discussed. In section 3 applications of the adaptive methods compared with the results obtained using a finite volume discretization on a regular grid are presented and their accuracy and CPU time reduction are studied. We show results for an advection equation and the viscous Burgers equation, both in one space dimension. Finally, conclusions are drawn and perspectives for future work is given.

2. Space and time discretization

2.1. **Adaptive multiresolution methods using finite volumes**

The initial boundary value problem for parabolic conservation laws we consider in the following can be written as,

$$\frac{\partial u}{\partial t} = \mathcal{D}(u) \quad \text{with} \quad u(x,0) \;=\; u_0(x) \tag{2.1}$$

for $(x,t) \in \Omega \times [0,+\infty)$, $\Omega \subset \mathbb{R}^d$, and completed with appropriate boundary conditions.

The diffusive flux $\mathcal{D}$ depends on u and its gradient ∇u. It can be decomposed into $\mathcal{D}(u) = -\nabla\cdot F(u,\nabla u)+S(u)$ where F is the flux and S is the source term. For space discretization of Eq. (2.1), we use a classical finite volume formulation in the standard conservative form.

The computational domain $\Omega \in I\!R^d$ of Cartesian shape is partitioned into cells $(\Omega_i)_{i\in\Lambda}$, $\Lambda = \{1,\ldots,i_{max}\}$. We then denote by $\bar{q}_i(t)$ the cell-average value of a given quantity q on Ω_i at instant t,

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

where $|\Omega_i| = \int_{\Omega_i} dx$ is the volume of the cell. Integrating Eq. 2.1 on Ω_i yields

$$\frac{d\bar{u}_i}{dt} = \bar{\mathcal{D}}_i(\bar{u}) \tag{2.2}$$

In the following, we will describe the space discretization and the time integration applied to (2.2).

Numerical flux

For the 1D case, Ω_i is a segment $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ with step size $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. The rhs of Eq. (2.2) becomes

$$\bar{\mathcal{D}}_i = -\frac{1}{\Delta x_i}\left(\bar{F}_{i+\frac{1}{2}} - \bar{F}_{i-\frac{1}{2}}\right) + \bar{S}_i \tag{2.3}$$

where we skipped the arguments to simplifiy notation.

Advective and diffusive terms are approximated differently. Roe's scheme [24] with a second-order ENO interpolation, is used for the advective part, whereas, for the diffusive part, we choose a second-order accurate centered scheme. The source term is approximated by $\bar{S}_i \approx S(\bar{u}_i)$. For a general non-linear source term, this choice yields second-order accuracy. A tensor product approach is used to extend this method to higher dimensions for Cartesian geometries.

Conservative 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 as fine as 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. 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. The difference between both approximations is negligible in terms of required nodes [11].

In this work we have a *dynamic tree* i.e., a tree that could change in time when needed, some nodes can be added or removed. Details of the scheme and its implementation are presented in [26].

Multiresolution representation

Starting point is the cell-average multiresolution representation (Harten [19]). The nodes are cell-average values and two operators are defined to navigate through the tree.

We denote by Λ the ensemble of the indices of the existing nodes, by $\mathcal{L}(\Lambda)$ the restriction of Λ to the leaves, and by Λ_l the restriction of Λ to a level l, $0 \leq l < L$. We denote by $\Omega = \Omega_{0,0}$ the root cell, $\Omega_{l,i}$, $0 \leq l < L$, $i \in \Lambda_l$ the different node cells, $\bar{u}_{l,i}$ the cell-average value of the quantity u on the cell $\Omega_{l,i}$, and $\bar{U}_l = (\bar{u}_{l,i})_{i\in\Lambda_l}$ 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 $P_{l+1\to l}$

$$P_{l+1\to l} : \bar{U}_{l+1} \mapsto \bar{U}_l. \tag{2.4}$$

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

$$\bar{u}_{l,i} = (P_{l+1\to l}\,\bar{U}_{l+1})_i = \frac{1}{2}(\bar{u}_{l+1,2i} + \bar{u}_{l+1,2i+1}) \tag{2.5}$$

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

$$P_{l\to l+1} : \bar{U}_l \mapsto \tilde{U}_{l+1} \tag{2.6}$$

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 neighbours 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 [19]:

$$\begin{aligned} \tilde{u}_{l+1,2i} &= I(\bar{U}_l; l+1, 2i) = \bar{u}_{l,i} + \sum_{m=1}^{s} \gamma_m (\bar{u}_{l,i+m} - \bar{u}_{l,i-m}) \\ \tilde{u}_{l+1,2i+1} &= I(\bar{U}_l; l+1, 2i+1) = \bar{u}_{l,i} - \sum_{m=1}^{s} \gamma_m (\bar{u}_{l,i+m} - \bar{u}_{l,i-m}) \end{aligned} \tag{2.7}$$

The accuracy order of the multiresolution method is denoted by r. A r-th 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

$$\begin{cases} r = 3 \quad \Rightarrow \quad \gamma_1 = -\dfrac{1}{8} \\ \\ r = 5 \quad \Rightarrow \quad \gamma_1 = -\dfrac{22}{128},\ \gamma_2 = \dfrac{3}{128} \end{cases} \tag{2.8}$$

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

$$\bar{d}_{l,i} = \bar{u}_{l,i} - \tilde{u}_{l,i} \tag{2.9}$$

These coefficients are redundant, the sum of the details for all the brothers of a parent cell being equal to zero by definition [19].

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 by

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

For more details on this equivalence, we refer to Harten [19].

For a given level l, it can be summarized by

$$\bar{U}_l \longleftrightarrow (\bar{D}_l,\, \bar{U}_{l-1})$$

Repeating this operation recursively on L levels, one gets the so-called *multiresolution transform* on the cell-average values [19].

$$\bar{\mathbf{M}} : \bar{U}_L \longmapsto (\bar{D}_L,\, \bar{D}_{L-1},\, \ldots, \bar{D}_1,\, \bar{U}_0) \tag{2.10}$$

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.

Following that procedure, a threshold operation is realized, its consists in removing leaves where the details are smaller than a prescribed tolerance ϵ, while preserving the graded tree data structure. One more level is added in that structure as security zone, to account for the evolution of the solution in the next time step.

2.2. **Time integration: Adaptive time stepping**

Adaptive time stepping can be introduced in different ways. Here we are using an extension of the Runge-Kutta-Fehlberg (RKF) method for PDEs [17]. The main idea of this method is to use two Runge-Kutta schemes of different order. A comparison of both results yields an estimation of the truncation error which can then be exploited to control the local error by changing the time step size. For RKF method of second and third orders,

employed here, the same evaluations of the fluxes can be used and hence the additional computational cost is negligible.

In the middle of the 1960's, Fehlberg was the first to propose these methods for solving ODEs, when he discovered a fifth-order method with six function evaluations while another combination of these functions gives a fourth order scheme [15, 16, 27].

Using ODE notation, we consider $\frac{du}{dt} = D(t,u)$ in the following. Therewith Runge-Kutta formulas of order $p-1$ read as follows

$$\check{u}^{\,m+1} = \check{u}^{\,m} + \sum_{i=1}^{p} \check{b}_i\, \kappa_i + \mathcal{O}\left((\Delta t)^p\right) \tag{2.11}$$

$$\begin{aligned} \kappa_1 &= \Delta t\, D(c_1 \Delta t, \check{u}^m) \\ \kappa_2 &= \Delta t\, D(c_2 \Delta t, \check{u}^m + a_{2\,1}\kappa_1) \\ &\ldots \\ \kappa_p &= \Delta t\, D(c_p \Delta t, \check{u}^{\,m} + a_{p\,1}\kappa_1 + \cdots + \check{u}^{\,m} + a_{p\,p-1}\kappa_{p-1}) \end{aligned} \tag{2.12}$$

For the RKF method, we use two Runge–Kutta methods, one of order p,

$$\hat{u}^{\,m+1} = \hat{u}^{\,m} + \sum_{i=1}^{p+1} \hat{b}_i \kappa_i + \mathcal{O}\left((\Delta t)^{p+1}\right) \tag{2.13}$$

and other of order $p-1$, yielding $\check{u}^{\,m+1}$ (eq. 2.11). The organization of the coefficients of both methods is displayed in Table 1(a) and the corresponding values of the coefficients for the RKF method of second and third orders are given in Table 1(b).

(a)Organization of the coefficients for RKF methods

c_1				
c_2	a_{21}			
c_3	a_{31}	a_{32}		
$\vdots$	$\vdots$			
c_s	a_{s1}		a_{ss-1}	
	$\hat{b}_1$	$\hat{b}_2$	$\cdots \hat{b}_{s-1}$	$\hat{b}_s$
	$\check{b}_1$	$\check{b}_2$	$\cdots \check{b}_{s-1}$	$\check{b}_s$

(b) Coefficients for RKF 2(3)

0			
1	0	1	
1/2	1/4	1/4	
	1/6	1/6	2/3
	1/2	1/2	0

TABLE 1. Coefficients for Runge-Kutta Fehlberg 2(3) method.

The estimate of the truncation error is defined as the difference between approximations of order p and $p-1$, and hence yields,

$$\delta_{old} := ||\hat{u}^{\,m+1} - \check{u}^{\,m+1}|| \tag{2.14}$$

By construction this truncation error δ_{old} scales as $(\Delta t)^p$. Using a time step Δt_{old}, an error δ_{old} is produced and similarly a new step Δt_{new} produces an error $\delta_{desired}$.

Both are then related by

$$\frac{\Delta t_{new}}{\Delta t_{old}} = \left| \frac{\delta_{desired}}{\delta_{old}} \right|^{1/p}$$

Therefore it follows that,

- if $\delta_{old} < \delta_{desired}$, then the time step can be increased;
- if $\delta_{old} \geq \delta_{desired}$, then the time step must be decreased.

This procedure allows to adjust automatically the step size in order to achieve a prescribed accuracy in time. Nevertheless, when $(\Delta t)_{new}$ is increased too much, the new predicted value may fail to meet the desired accuracy. In the present implementation, this is not allowed as 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 ($\mathcal{S}$). The new time step $(\Delta t)_{new}$ is chosen such that

$$-\frac{\mathcal{S}}{2} \leq \frac{(\Delta t)_{new} - (\Delta t)_{old}}{(\Delta t)_{old}} \leq \frac{\mathcal{S}}{2}$$

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 [18]. 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 $\mathcal{S} = \mathcal{S}(t)$ with a exponential decay during the first time steps, i.e,

$$\mathcal{S}(t) = (\mathcal{S}_0 - \mathcal{S}_{min}) \ \exp\left(-\frac{t}{\Delta t}\right) + \mathcal{S}_{min}.$$

The behaviour of the limiter $\mathcal{S}(t)$ for $t = 0$ is the maximal allowed variation $\mathcal{S}_0$ and, for $t \to \infty$, it is $\mathcal{S}_{min}$, where $\mathcal{S}_{min} < \mathcal{S}_0$. In the present paper we use $\mathcal{S}_0 = 0.1$ and $\mathcal{S}_{min} = 0.01$ for all case studies presented in Section 3. 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%.

The implementation of RKF 2(3) is no more costly than the classical RK 3, thanks to the fact that it is possible to use a compact scheme which uses RK 2 to calculate RK 3 without any additional flux evaluation. In this sense we compute the RK 2

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

and RK 3

$$\begin{aligned} \bar{u}^{**} &= \left(3\bar{u}^n + \bar{u}^* + \Delta t D(\bar{u}^*)\right)/4 \\ \bar{u}^{n+1} &= \left[\bar{u}^n + 2\bar{u}^{**} + 2\Delta t D(\bar{u}^{**})\right]/3 \end{aligned}$$

The storage area in the RKF 2(3) is slightly larger than the usual RK 3 method because it is also necessary to store the second–stage of RK 2.

2.3. **Possible combinations of the different methods**

Summarizing the different strategies, i.e., adaptive space discretization using multiresolution techniques, adaptive time discretization using RKF and local scale dependent time stepping (not presented here), in total six different numerical schemes can be construcuted:

FV : finite volume discretization with fixed time stepping;
MR : adaptive multiresolution method using finite volume discretization with fixed time stepping;
FV/RKF : finite volume discretization with RKF (for adaptive global time stepping);
MR/RKF : adaptive multiresolution using finite volume discretization with RKF ;
MR/LTS : adaptive multiresolution using finite volume discretization with local but fixed time stepping;
MR/LTS/RKF : adaptive multiresolution using finite volume discretization with local and adaptive time stepping using RKF .

A diagram illustrating the possible combination of the three strategies is presented in Fig. 1. The FV scheme always serves as a reference for numerical accuracy, memory and CPU time requirements. The remaining 5 schemes aim at reducing memory and CPU time while maintaining a similar numerical accuracy of the reference scheme FV. The multiresolution strategy allows to reduce the number of grid points and controls the approximation error in space. On the other hand, adaptive time stepping guarantees a desired precision for time evolution. Furthermore, the local scale dependent time stepping permits to use larger time steps for the large scales than on fine scales, which leads to further savings of the CPU time. Finally, using the multiresolution strategy and a combination of the last two time-stepping strategies, we have a fully adaptive space-time scheme MR/LTS/RKF.

For the numerical examples presented in the next section we test, validate and compare the first four schemes. The two last schemes are work in progress and will be presented in a forthcoming paper.

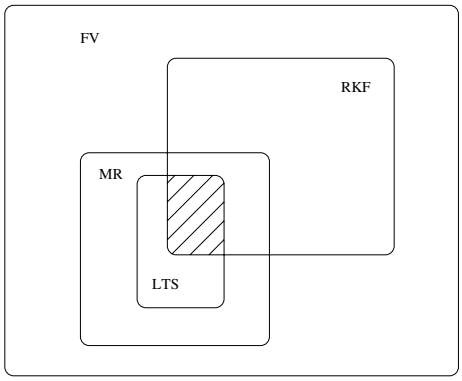

FIGURE 1. Set presentation of the different discretization methods. Each set corresponds to a different numerical method and their intersections to possible combinations. FV denotes the finite volume space discretization, MR stands for the adaptive multiresolution method. The time discretization is done either by RKF, or by scale dependent time stepping LTS, or by the RKF method and a possible combination of the two latter (hatched region).

3. Numerical results

In this section, we present different numerical examples in one space dimension using finite volume second-order accurate schemes with third order Runge–Kutta time integration. In the case of an adaptive space discretization a multiresolution analysis of order $r = 3$ is used. The global time adaption is done by the Runge–Kutta–Fehlberg 2(3) method. In the following the different methods are applied to a linear advection and a viscous Burgers equation.

3.1. **Advection equation**

First we consider a linear advection equation

$$\frac{\partial u}{\partial t} + c\,\frac{\partial u}{\partial x} = 0,$$

where $u = u(x,t)$, $t \geq 0$, $x \in [-1,1]$, and $c > 0$ is the constant velocity. The boundary conditions are periodic and the initial condition is

$$u_0(x) = \exp\left(-50\,x^2\right).$$

In the numerical computations, we chose c=1 and a resolution of 256 points.

In Figure 2 we show the time evolution of the time step for different initial CFL, i.e., for different initial time step sizes, using the finite volume scheme on a regular grid with global time adaptivity and the RKF 2(3) method. We observe that the time steps tend to converge in all cases to a time step of $\approx 5.2 \times 10^{-3}$, , i.e. an initial CFL ≈ 0.65. To avoid a "bad choice" of the initial CFL, the code allows a bigger limiter in the beginning time steps. These "bad choices" of initial CFL could however increase the global error, as presented in Table 2. The automatic step size control of the solution reduces the number of time steps and hence the computational cost (cf. Table 2). We can also observe that the error with respect to the analytical solution computed at the final time $t = 1$ is reduced for the time adaptive schemes, compared to the finite volume scheme with fixed time stepping, except for the initial CFL$= 1$ using the L^∞ norm. In Figure 4 we compare L^2, L^1, L^∞ norms for the RK 3 method with CFL$= 0.5$ and the RKF 2(3) method. The results show that the choice of the initial time step does not influence the error at the final time instant as all computations yield a similar result. Figure 3 shows the CPU time spent for different choices of the initial CFL. The CPU time decreases as the initial CFL increases. This is directly related to the number of time steps needed to compute the solution up to $t = 1$, as could be observed in Fig. 5. For this test case, we can thus conclude that RKF 2(3) is more efficient than the conventional RK 3 method.

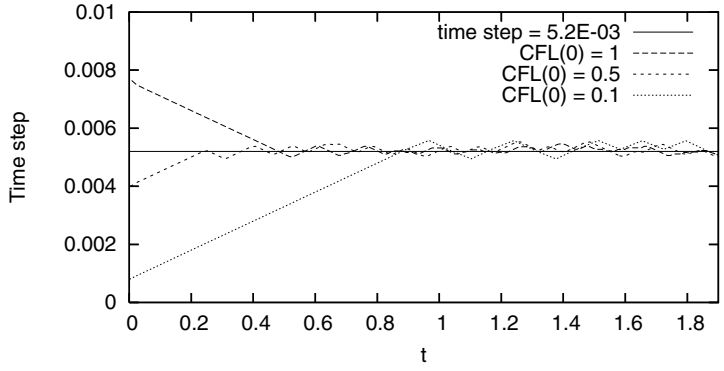

FIGURE 2. Evolution of the time step depending on the initial CFL value for the finite volume scheme with global time adaptivity using the RKF 2(3) method for the advection equation. Grid containing 256 points, $\delta_{desired} = 10^{-3}$, $\mathcal{S}_{min} = 0.01, \mathcal{S}_0 = 0.1$.

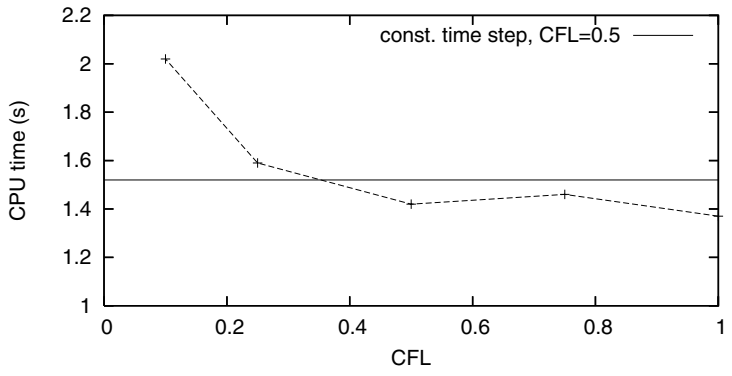

FIGURE 3. CPU time vs. initial CFL value for the advection equation. The grid contains 256 points. The parameters are $\delta_{desired} = 10^{-3}$, $\mathcal{S}_{min} = 0.01$, and $\mathcal{S}_0 = 0.10$.

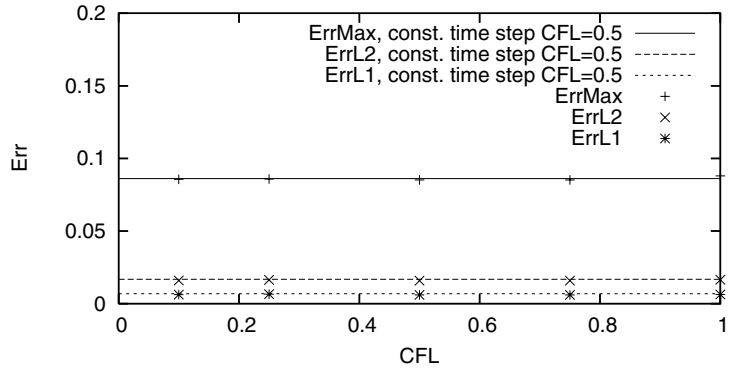

FIGURE 4. Errors vs. initial CFL value for the advection equation. The grid contains 256 points. The parameters are $\delta_{desired} = 10^{-3}$, $\mathcal{S}_{min} = 0.01$, and $\mathcal{S}_0 = 0.10$.

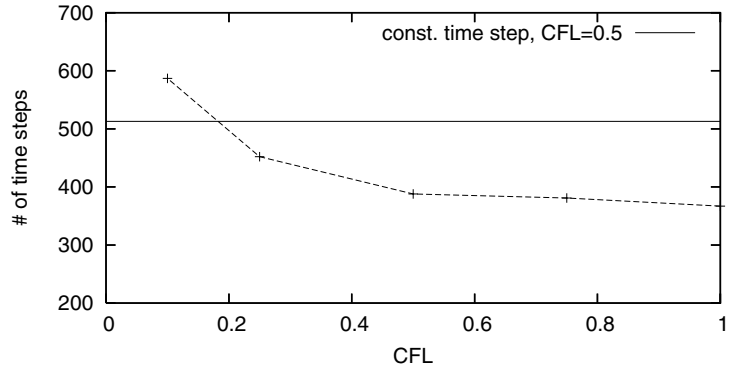

FIGURE 5. Number of time steps vs. initial CFL value for the advection equation. The grid contains 256 points. The parameters are $\delta_{desired} = 10^{-3}$, $\mathcal{S}_{min} = 0.01$, and $\mathcal{S}_0 = 0.10$.

Init. CFL	# steps	% CPU time	Initial time step	Final time step	L^∞-error	L^2-error	L^1-error
0.50	513	100	3.91e-03	3.91e-03	8.61e-02	1.68e-02	6.84e-03
1.00	367	90	7.64e-03	5.27e-03	8.80e-02	1.65e-02	6.27e-03
0.75	381	96	5.73e-03	5.18e-03	8.52e-02	1.58e-02	5.97e-03
0.50	388	93	3.90e-03	5.15e-03	8.52e-02	1.58e-02	5.97e-03
0.25	452	105	2.00e-03	4.88e-03	8.58e-02	1.64e-02	6.57e-03
0.10	587	133	7.98e-04	4.94e-03	8.56e-02	1.60e-02	6.23e-03

TABLE 2. Initial CFL, number of time steps, CPU time, initial and final time steps, and errors for the advection equation. Mesh containing 256 points, $\delta_0 = 10^{-3}$, $\mathcal{S}_{min} = 0.01$, $\mathcal{S}_0 = 0.10$. The first line corresponds to the constant time step with CFL=0.5.

3.2. **Viscous Burgers equation**

Next we consider the viscous Burgers equation as an example to model shock formation,

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} - \nu\,\frac{\partial^2 u}{\partial x^2} = 0,$$

where $u = u(x,t)$, $t \geq 0$, $x \in [-1,1]$ and $\nu = 10^{-2}/\pi$. The above equation is completed with periodic boundary conditions and the initial condition is given by,

$$u(x,0) = -\sin(\pi x).$$

Figure 6 shows the initial condition and the numerical solution at $\pi t = 1.6037$ computed with the MR/RKF method. Table 3 summarizes the memory and CPU time compressions for the 5 different methods using 3 different max-

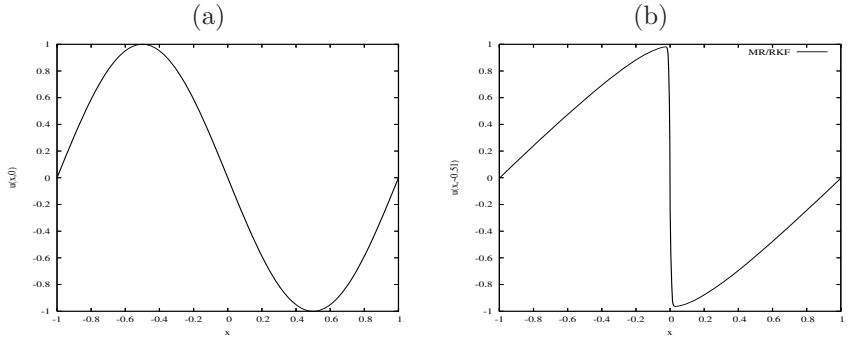

FIGURE 6. Initial condition and the numerical solution of the Burgers equation at $t = 1.6037/\pi \approx 0.51$ computed using the MR/RKF.

imal scales $L = 11$, 12 and 13, which correspond to grids of size 2^L. First we observe that all space and/or time adaptive methods require less CPU time than the finite volume method using a fixed time step (FV). For example for $L = 13$ we find that the adaptive multiresolution method with a fixed time step (MR) only requires 28% of the CPU time and 19.4% of the memory compared to the FV method. Using the adaptive space discretization with RK 3, we reduce by 45% and 63% the CPU time and using MR/RKF with RKF 2(3) we reduce in 86% and 92% the total CPU time, on the scale levels 11 and 12 compared to FV. Using in addition adaptive time stepping the CPU and memory requirement can be further decreased. For the MR/RKF method using $L = 13$ scales we only need 5% of the CPU time with respect to the FV method and 17.6% of the memory. The FV/RKF is competitive with $L = 11$ scales leading to a reduction of 80% of the CPU time. Increasing the number of scales to $L = 12$ requires however more CPU time than the FV and hence we decided not to perform further computations using this method.

To verify the precision of the different numerical methods we compare the slope $|\partial u/\partial x|_{max}$ at $\pi t = 1.6037$ with the slope of the exact solution given in [3], which yields 152. Table 3 shows that the slope of the numerical solutions is close to the one of the exact solution, and that the precision is improved for an increasing number of scales. The introduction of adaptive time stepping does not effect the memory compression and the precision of the computation. It can also be noticed that when the solution exhibits small scales, the desired accuracy in time has to be reduced in order to avoid numerical instabilities, like spurious oscillations.

Figure 7 (left) shows the evolution of the time step for both FV/RKF and MR/RKF methods. For the former, we observe after some initial adjustment an oscillatory behaviour, while for the latter, we find that the time step follows the dynamics of the shock formation. In Figure 7 (right) we plot the evolution of the time step for the MR/RKF method using different numbers of maximum scales, $L = 11, 12$ and 13. For late times, .i.e., $t > 0.4$, we see that the time step is decreasing for increasing spatial resolution. At earlier time the evolutions strongly depend on the the maximum number of scales and we find different behaviors during the evolution.

4. Conclusions and perspectives

We presented and validated a new space-time adaptive numerical scheme to solve PDEs. Starting point is a finite volume discretization with explicit time interation. Then we coupled the previously developed adaptive

Method	Scales	# steps	% CPU Time	% Memory	$\lvert\partial u/\partial x\rvert_{max}$	$\delta_{desired}$
FV	11	18346	100	100	149.6	
FV/RKF	11	4153	20	100	149.5	10^{-5}
MR	11	18346	55	42.7	149.5	
MR/RKF	11	3697	14	43.3	147.5	10^{-5}
FV	12	70767	100	100	151.7	
MR	12	70767	37	26.2	149.8	
MR/RKF	12	13370	8	27.2	149.4	10^{-6}
FV	13	277839	100	100	153.2	
MR	13	277839	28	19.4	151.8	
MR/RKF	13	56741	5	17.6	150.7	10^{-6}

TABLE 3. CPU and memory compression for the discretization methods, using an initial CFL = 0.4, $\epsilon = 10^{-2}$, the limiters are $\mathcal{S}_{min} = 0.01$, and $\mathcal{S}_0 = 0.10$. At $\pi t = 1.6037$ we compare the slope of the numerical solutions with the exact one $|\partial u/\partial x|_{max} \approx 152$.

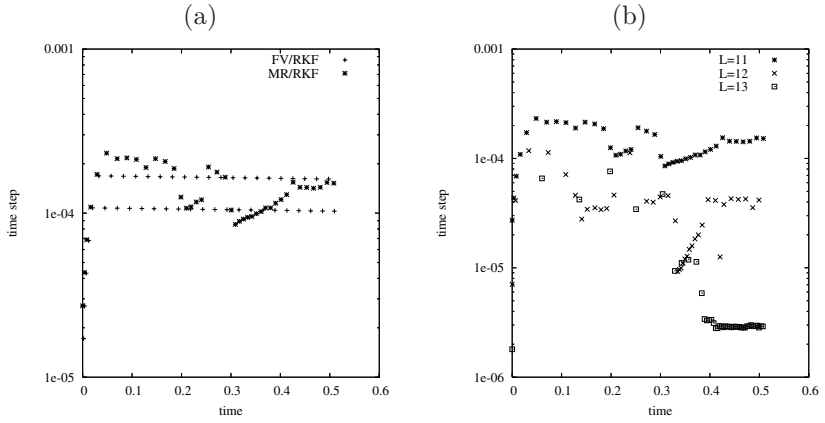

FIGURE 7. Evolution of the time step for Burgers equation from $t = 0$ to $t \approx 0.51$. The final time corresponds to the maximum slope. Methods FV/RKF and MR/RKF with $L = 11$ (left) and method MR/RKF for $L = 11, 12$ and 13 (right).

multiresolution scheme [25, 26] with a Runge–Kutta–Fehlberg method to adjust automatically the size of the time step. We demonstrated the efficiency of the new method for different test problems and studied its performance by comparing the CPU time and the memory requirements with the finite volume method using uniform discretization and a third order Runge–Kutta scheme for time integration. As the implemented Runge–Kutta–Fehlberg 2(3) method uses a low storage in memory, it is necessary to introduce a limiter of the time step to avoid unappropriated choices of the time step, e.g., large time steps which may violate the precision of the computation. This limiter is a function of time and it is usually larger during the first time steps. It also avoids that unappropriated initial time steps increase the CPU time. The computational extra cost of Runge–Kutta–Fehlberg 2(3) is very low because it is possible to use compact formulas to perform the time evolution, i.e., we use the previous values from RK 2 to evaluate the RK 3. The results showed that the Runge–Kutta–Fehlberg global time adaptivity coupled with the multiresolution technique reduced significantly the CPU time without loosing accuracy.

The MR/RKF method performs a global adaption of the time step, i.e. for all spatial scales the time step evaluation is based on the precision required by the finest scale. However, it is also possible to combine this technique with local scale dependent time stepping, which allows different time step sizes for different scales.

In this case, different scales evolve with different time steps, hence a synchronization of the tree data structure becomes necessary. This is work in progress and will be presented in two forthcoming papers [12, 13].

On a longer term perspective we also plan to extend this space-time adaptive scheme to the 3D Navier-Stokes equations together with the Coherent Vortex Simulation method [14] to compute efficiently turbulent flows.

We thank S. Gomes for fruitful discussions. We are also thankful to M–G. Dejean for her helpful assistance.

References

[1] J. Alam, N.-K.-R. Kevlahan, and O.V. Vasilyev. Simultaneous space-time adaptive wavelet solution of nonlinear partial differential equations. *J. Comput. Phys.*, (214):829–857, 2006.

[2] E. Bacry, S. Mallat, and G. Papanicolaou. A wavelet based space-time adaptive numerical-method for partial-differential equations. M^2AN, 26:793–834, 1992.

[3] C. Basdevant, M. Deville, P. Haldenwang, J. M. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A. T. Patera. Spectral and finite difference solutions of Burgers equation. *Comput. & Fluids*, 14(1):23–41, 1986.

[4] R. Becker and R. Rannacher. An optimal control approach to error control and mesh adaptation. *Acta Numer.*, 10:1–102, 2001.

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

[6] G. Chiavassa and R. Donat. Point value multi-scale algorithms for 2d compressible flow. *SIAM J. Sci. Comput.*, 23(3):805–823, 2001.

[7] A. Cohen. Wavelet Methods in Numerical Analysis. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of Numerical Analysis*, volume VII. Elsevier, Amsterdam, 2000.

[8] A. Cohen, S. M. Kaber, S. Müller, and M. Postel. Fully adaptive multirresolution finite volume schemes for conservation laws. *J. Comput. Phys.*, 72:183–225, 2000.

[9] F. Collino, T. Fouquet, and P. Joly. A conservative space-time mesh refinement method for the 1-d wave equation. I. Construction. *Numer. Math.*, 95(2):197–221, 2003.

[10] F. Collino, T. Fouquet, and P. Joly. A conservative space-time mesh refinement method for the 1-d wave equation. II. Analysis. *Numer. Math.*, 95(2):223–251, 2003.

[11] R. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.

[12] M. O. Domingues, S. M. Gomes, O. Roussel, and K. Schneider. An adaptive multiresolution methods with local-time stepping for evolutionary PDEs. 2006. (in preparation).

[13] M. O. Domingues, O. Roussel, and K. Schneider. An adaptive multiresolution method for parabolic PDEs with time-step control. *Appl. Numer. Math.*, 2006. (submitted).

[14] M. Farge and K. Schneider. Coherent vortex simulation (CVS), a semi-deterministic turbulence model using wavelets. *Flow, Turbulence and Combustion*, 66(4):393–426, 2001.

[15] E. Fehlberg. New high-order Runge–Kutta formulas with step size control for systems of first order and second order diferential equation. *Z. Angew. Math. Mech.*, 44(T17-T29), 1964.

[16] E. Fehlberg. Klassische Runge–Kutta Formeln fünfter und siebenter Ordnung mit Schrittweiten-Kontrolle. *Computing*, 4:93–106, 1969.

[17] L. Ferm and P. Löstedt. Adaptive error control for steady state solutions of inviscid flow. *J. Sci. Comput.*, 23:1777–1798, 2002.

[18] L. Ferm and P. Löstedt. Space-time adaptive solution of first order PDES. *J. Sci. Comput.*, 26(1):83 – 110, 2006.

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

[20] A. Harten. Multiresolution representation of data: a general framework. *SIAM J. Numer. Anal.*, 33(3):385–394, 1996.

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

[22] S. Müller and Y. Stiriba. Fully adaptive multiscale schemes for conservation laws employing locally varying time stepping. Technical Report 238, IGPM, RWTH Aachen, April 2004. (to appear in J. Sci. Comput. in 2006).

[23] S. Oscher and R. Sanders. Numerical approximations to nonlinear conservation laws with locally varying time space grid. *Math. Comp.*, 43:321–336, 1983.

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

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

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

[27] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*, volume 12 of *Text in Applied Mathematics*. Springer-Verlag, Berlin, 2nd edition, 1991.

[28] E. Suli and D. F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.
[29] H. Z. Tang and G. Warnecke. A class of high resolution schemes for hyperbolic conservation laws and convection-diffusion equations with varying time and space grids. *SIAM J. Sci. Comput.*, 26(4):1415–1431, 2005.