



HAL
open science

Local time-stepping for adaptive multiresolution using natural extension of Runge–Kutta methods

Muller Moreira Lopes, Margarete Oliveira Domingues, Kai Schneider, Odim Mendes

► **To cite this version:**

Muller Moreira Lopes, Margarete Oliveira Domingues, Kai Schneider, Odim Mendes. Local time-stepping for adaptive multiresolution using natural extension of Runge–Kutta methods. *Journal of Computational Physics*, 2019, 382, pp.291-318. 10.1016/j.jcp.2018.10.052 . hal-02616166

HAL Id: hal-02616166

<https://hal.science/hal-02616166>

Submitted on 5 Apr 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Local time-stepping for adaptive multiresolution using natural extension of Runge–Kutta methods

Müller Moreira Lopes^{a,d,*}, Margarete Oliveira Domingues^{b,d}, Kai Schneider^e, Odim Mendes^{c,d}

^a*Graduate program in Applied Computing (CAP)*

^b*Associate Laboratory of Applied Computing and Mathematics (LAC), Coordination of the Associated Laboratories (CTE)*

^c*Space Geophysics Division (DGE), Coordination of Space Sciences (CEA)*

^d*National Institute for Space Research (INPE),*

Av. dos Astronautas 1758, 12227-010 São José dos Campos, São Paulo, Brazil

^e*Institut de Mathématiques de Marseille (I2M), Aix-Marseille Université, CNRS, Centrale Marseille
39 rue F. Joliot–Curie, 13453 Marseille Cedex 13, France*

arXiv:1905.08717v1 [math.NA] 21 May 2019

Abstract

A space-time fully adaptive multiresolution method for evolutionary non-linear partial differential equations is presented introducing an improved local time-stepping method. The space discretisation is based on classical finite volumes, endowed with cell average multiresolution analysis for triggering the dynamical grid adaptation. The explicit time scheme features a natural extension of Runge–Kutta methods which allow local time-stepping while guaranteeing accuracy. The use of a compact Runge–Kutta formulation permits further memory reduction. The precision and computational efficiency of the scheme regarding CPU time and memory compression are assessed for problems in one, two and three space dimensions. As application Burgers equation, reaction-diffusion equations and the compressible Euler equations are considered. The numerical results illustrate the efficiency and superiority of the proposed local time-stepping method with respect to the reference computations.

Keywords: Multiresolution Analysis, Finite Volume, Local time-stepping, Runge–Kutta

1. Introduction

Multiresolution (MR) methods improve the computational performance of numerical solvers of evolutionary partial differential equations when the solution exhibits localised structures, point-wise singularities, boundary layers, shocks, coherent vortices such as encountered in combustion and turbulent flow applications [12, 26, 19].

In these methods, the fast wavelet transform is the key ingredient to speed-up computations. The wavelet coefficients are employed to measure the local smoothness of the solution. Then, a thresholding strategy is used to remove non-significant coefficients, obtaining a grid adapted to the solution. This grid is coarser in smooth regions and finer there where structures and steep gradients are present. The locally refined grid can be rebuilt interactively to a regular grid with an expected error directly related to the chosen threshold. With the use of this adaptive grid, the number of interface flux computations during the time evolution can be significantly reduced, while controlling the error. The adaptive grid is checked

*Corresponding author

Email addresses: muller.lopes@inpe.br (Müller Moreira Lopes),
margarete.domingues@inpe.br, margarete.oliveira.domingues@gmail.com (Margarete Oliveira Domingues),
kai.schneider@univ-amu.fr (Kai Schneider), odim.mendes@inpe.br (Odin Mendes)

before each time step to guarantee that it is sufficiently refined to represent possible new structures in the solution. Therefore, the adaptive grid is dynamically adapted to track the solution in scale and space.

The next step to improve the computational performance associated with these adaptive grids is to use a local time-stepping (LT) approach, especially when explicit time schemes are used. In this work, the combination of MR methods with the local time-stepping approach is denoted as MRLT. LT consists in performing the time evolution of each cell on the adaptive grid independently according to its required time step. Therefore, each cell must have a time step proportional to its refinement and consequently, a larger cell performs a larger time step.

Local time-stepping methods for space adaptive discretisations of partial differential equations have a long tradition, going back to the early work of Osher and Sanders [21]. Related to multiresolution, Müller and Stiriba [20] presented some general MRLT schemes that could be applied either to an explicit time scheme, based on Lagrange projection, or an implicit time scheme, for a reference finite-volume method in space. In [20] they applied LT to one-dimensional scalar conservation laws. Following this work, Coquel *et al.* [3] presented MRLT methods with both explicit and implicit Lagrange-projection schemes, the latter is the novelty concerning [20]. Hejazialhosseini *et al.* combined in [13] first and second order RK schemes to propose an LT approach for MR in blocks with finite volumes for multi-phase compressible flows implemented on multi-core architectures.

More recently, LT methods have been proposed using different approaches for time interpolation required in the algorithms for providing values at intermediate time steps. In [24], the authors use a high-order Taylor type integrator, while a discontinuous Galerkin method with spectral elements is used for spatial discretization. In the context of finite elements methods, [25] proposed an energy conserving LT algorithm based on a second-order leap-frog scheme. Discussions on the proper choice of the time-step in LT schemes are still an important topic of discussion. To this end *a posteriori* error estimators are used in [1, 18], while Gnedin *et al.* (2018) [11] investigate their effect by enforcing the CFL condition locally over every cell. In the context of finite volumes methods, the use of high-order schemes for local time-stepping on adaptive mesh refinement grids were previously discussed in [8]. In that work, the accuracy order in space is obtained through a WENO reconstruction, while the accuracy in the time discretization is achieved by a local space-time discontinuous Galerkin predictor method. This approach yields up to fourth order for compressible Euler equations in 2D. Similar and related works in this context were carried out in [15] and [2].

A detailed discussion on the stability of those MRLT schemes is presented in [14]. In the context of adaptive numerical methods for partial differential equations, the derivation of explicit LT methods based on standard RK schemes typically stays at orders smaller or equal to two [6, 4, 5]. The reason why LT methods are limited to low order in time is because a time synchronization is required; for a discussion we refer to [6]. Recently, higher-order LT schemes have been proposed in the context of discontinuous Galerkin methods. The works of Gassner *et al.* [10, 9] in this context are based on natural continuous extensions for Runge–Kutta methods (NERK). Such schemes have been introduced initially by Zennaro in the late 1980’s for solving general ODEs, with application to delay equations [30]. The idea is to interpolate the intermediate stages of the Runge–Kutta scheme to obtain the values at the requested intermediate time instants required for the time synchronisation in LT schemes. This method is also called Natural Continuous Extension in [30], Continuous Extension Runge–Kutta in [23], and in a general way Continuous Runge–Kutta, as discussed in [28].

Recently, an alternative for higher-order LT methods, again in the context of discontinuous Galerkin spectral element schemes, has been proposed by Winters and Kopriva [29]. The underlying ideas are Adams–Bashforth multi-step schemes.

Moreover, Gassner *et al.* (2011) considered a family of explicit one-step time discretisations for finite volume (FV) and discontinuous Galerkin schemes, which are based on a predictor-corrector formulation

[9].

The aim here is to use the idea of Gassner *et al.* [10] using NERK for the first time in the context of MRLT methods to perform the synchronisation. Thus, an improvement in time accuracy can be obtained. In the current work, only second and third-order schemes are used, but the extension to higher-order is in principle possible. Gassner *et al.* [10] discussed that NERK is 10% slower than the Cauchy—Kovalevskaya scheme. However, to work with the latter is complicated as the analytic solution of some nonlinear PDEs is required. The goal of the proposed approach is to perform simulations using NERK schemes. Therewith, the synchronisation required for the LT approach is possible, and this new class of MRLT methods, named MRLT/NERK, is created. This work presents the application and implementation of the MRLT/NERK method for the second (RK2) and third (RK3) order time evolution, named MRLT/NERK2 and MRLT/NERK3, respectively. The method proposed in this work is applied for solving the two-dimensional Burgers equation, one and three-dimensional reaction-diffusion problems and the two-dimensional compressible Euler equations. The obtained results and CPU times are compared with the MR method using classical RK2 and RK3 time evolution, named MR/RK2 and MR/RK3, respectively. The results are also compared with the MRLT approach given in [6] based on RK2 time evolution, named MRLT/RK2.

In Section 2, we summarise the adaptive multiresolution method proposed in [12]. Then, in Section 3, we discuss the Runge–Kutta methods and the NERK methods used in the current work to perform the proposed MRLT/NERK approach given in Section 4. A convergence analysis is conducted in Section 4.4. Performance comparisons, considering CPU time and errors, among the FV, MR and MRLT approaches given in [6] and the MRLT/NERK approach introduced here, are presented in Section 5. Conclusions are drawn in Section 6.

2. Adaptive multiresolution methods using finite volumes

The following initial value problem for a vector-valued function $\mathbf{Q}(\mathbf{x}, t)$ written in divergence form is considered:

$$\frac{\partial \mathbf{Q}}{\partial t} = -\nabla \cdot \mathbf{F}(\mathbf{Q}, \nabla \mathbf{Q}) + \mathbf{S}(\mathbf{Q}), \quad \text{for } (\mathbf{x}, t) \in \Omega \times [0, +\infty), \Omega \subset \mathbb{R}^d. \quad (1)$$

This problem is given in d space dimensions, completed with initial conditions $\mathbf{Q}(\mathbf{x}, t = 0) = \mathbf{Q}_0(\mathbf{x})$ and appropriate boundary conditions. The terms $\nabla \cdot \mathbf{F}(\mathbf{Q}, \nabla \mathbf{Q})$ and $\mathbf{S}(\mathbf{Q})$ denote the divergence and source term, respectively. The flux \mathbf{F} can be decomposed into advective and diffusive contributions, *i.e.* $\mathbf{F}(\mathbf{Q}, \nabla \mathbf{Q}) = \mathbf{f}(\mathbf{Q}) - \nu \nabla \mathbf{Q}$, where the diffusion coefficient ν is positive and assumed to be constant.

To discretise Equation (1) in space, we use a classical finite volume formulation written in standard form. The domain Ω corresponds to a rectangular parallelepiped in $d = 1, 2$ or 3 dimensions in Cartesian geometry. It is partitioned into cells $(\Omega_i)_{i \in \Lambda}$, $\Lambda = \{0, \dots, i_{\max}\}$ with $\Omega = \bigcup_i \Omega_i$.

Defining the volume of the cell by $|\Omega_i| = \int_{\Omega_i} d\mathbf{x}$, we compute the cell-average $\bar{\mathbf{q}}_i(t)$ of a given quantity \mathbf{Q} on Ω_i at time instant t by,

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

Considering the one-dimensional case, Ω_i is an interval $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ of length $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. Integrating Equation (1) on Ω_i then yields:

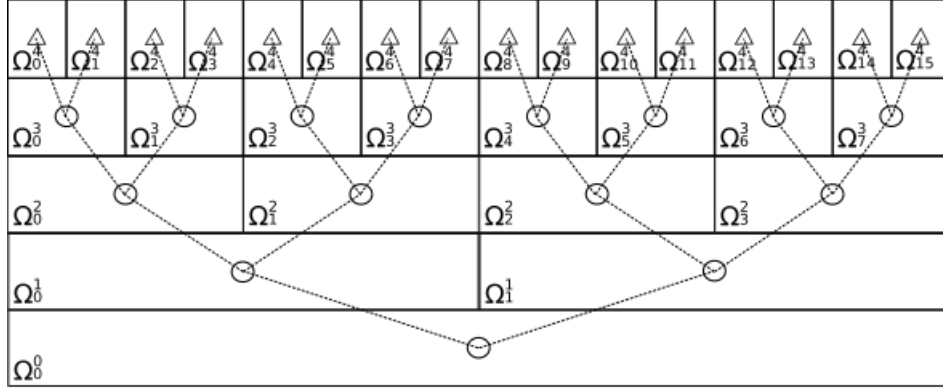
$$\frac{d\bar{\mathbf{q}}_i}{dt}(t) = -\frac{1}{\Delta x_i} \left(\bar{\mathbf{F}}_{i+\frac{1}{2}} - \bar{\mathbf{F}}_{i-\frac{1}{2}} \right) + \bar{\mathbf{S}}_i, \quad (2)$$

where $\bar{\mathbf{F}}$ is the numerical flux and $\bar{\mathbf{S}}_i$ is the source term of the cell Ω_i . This formulation can be extended to two- and three dimensional problems.

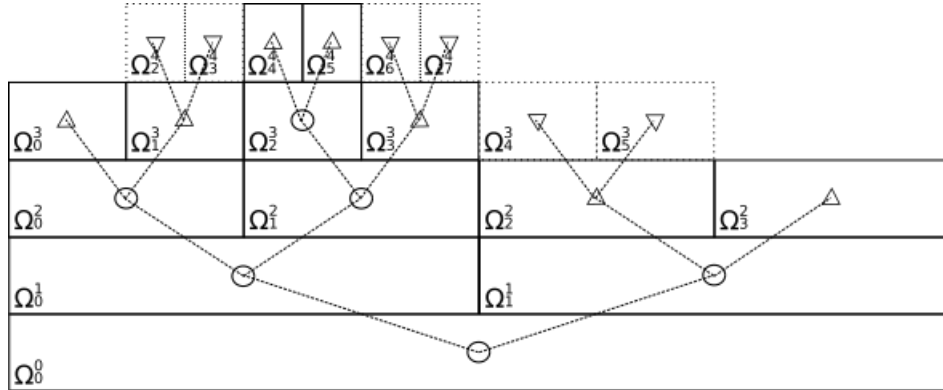
The source term is approximated by $\bar{\mathbf{S}}_i \approx \mathbf{S}(\bar{\mathbf{q}}_i)$, which yields also second-order accuracy in case of a linear source term.

The adaptive multiresolution (MR) analysis in the cell average context, proposed by Harten [12], consists in decomposing the cell-averages of the solution into a multilevel representation. This representation, illustrated in Figure 1a, consists of a hierarchy of nested grids Ω^ℓ , where ℓ is the grid refinement level. Each grid Ω^ℓ consists of a regular grid, as defined for the FV formulation, with $2^{d\ell}$ cells. A cell of refinement level ℓ and position i is denoted by Ω_i^ℓ .

Figure 1a shows the implementation of this structure as a binary tree, where the nodes of level ℓ generate the grid Ω^ℓ . For the two- and three-dimensional cases, this idea is extended by using a quadtree and an octree, respectively.



a) Grid hierarchy for an unidimensional domain.



b) Tree structure for an adapted grid.

Figure 1: Dyadic grid hierarchy of MR methods and its implementation in a tree data structure. a) Example of nested unidimensional grids, the circles represent the internal nodes of the tree structure, while the triangles represent the leaves of the tree. b) Example of an adapted grid using the tree structure from a). The virtual leaves are represented by the triangles ∇ . Adapted from [26].

This sequence of nested grids corresponds to a scheme where a finer scale represents its subsequent coarser scale plus a sum of details between these levels. These details are the wavelet coefficients. The construction of the adapted grid in the MR method consists in representing the grid using only cells with significant wavelet coefficients. For this, the leaves of the most refined level are checked, if their parent cell has a significant wavelet coefficient, larger than a predetermined threshold ϵ , if not, the leaves are deleted. This process is repeated recursively starting from the finest level, going to coarser and coarser levels. The procedure to obtain the projection of the solution on the coarser levels and the wavelet coefficients is given in Section 2.

The grid generation procedure is mathematically supported by the fact that the magnitude of the wavelet coefficients are small in regions where the solution is smooth, while they are significant in regions where the solution exhibits steep gradients. Hence, high compression rates are expected when only a small number of cell-averages is present on the finest scales.

In this work, the data structure for representing the solution is organized as a dynamic graded tree. This tree organization requires that no hole is admitted inside the tree, which means that the connectivity in the tree structure has to be ensured. Moreover, the tree can change in time to track the space-scale evolution of the solution as nodes can be added or removed while guaranteeing its gradedness.

This dynamic graded tree organization implies that neighbours of each cell can have a difference of one refinement level at most. This restriction allows the use of virtual leaves for the flux computations between leaves at different refinement levels. These virtual leaves are auxiliary leaves placed as children nodes of the leaves who have an interface with finer leaves. Their values are predicted using the prediction procedure used to compute the wavelet coefficients. A unidimensional adaptive grid represented in a graded tree structure is shown in Figure 1b.

The flux computations in the MR scheme are performed individually for each leaf. The numerical fluxes are computed between cells at the same refinement level. Using the adaptive grid of Figure 1b, the following interface scenarios for computing the numerical fluxes of a leaf can be identified:

- **Leaf / Leaf or Virtual leaf:** When both leaves belong to the same refinement level, the flux computation is performed in the same way as in the FV method. The following cases can occur: Leaf/Leaf example: Flux between cells Ω_0^3 and Ω_1^3 ; Leaf/Virtual leaf example: Flux between cells Ω_5^4 and Ω_6^4 .
- **Leaf / Internal node:** In this scenario, the current leaf has an interface with a finer leaf. To perform the flux computation in this case, the numerical flux is computed using the virtual children of the leaf and its adjacent leaves. Example: Flux between cells Ω_2^2 and Ω_1^2 .

The construction of the adaptive grid guarantees that the current solution is well represented. However, after the time evolution process, the new solution should also be well represented on this grid, which a priori cannot be ensured. In order to guarantee that the new solution after the time evolution is still well represented on this grid, the leaves with finer neighbours are refined and neighbor cells are added. Further details of the MR scheme and its implementation can be found in [26]. The Algorithms 1 and 2 given in Appendix Appendix A describe the adaptive grid creation and its update, respectively.

Projection and prediction operators

In order to perform the MR method, some operations for projection and prediction are required. For the MR scheme with finite volumes, where the cell values are local averages, a coarser cell Ω_i^ℓ has its value estimated using the finer values and an unique projection operator $P_{\ell+1 \rightarrow \ell} : \bar{\mathbf{q}}^{\ell+1} \mapsto \bar{\mathbf{q}}^\ell$. In this scheme, the projection operator to obtain the solution of a coarser cell is given by the average value of its children. For the unidimensional case, the projection is performed by:

$$\bar{\mathbf{q}}_i^\ell = P_{\ell+1 \rightarrow \ell} (\bar{\mathbf{q}}_{2i}^{\ell+1}, \bar{\mathbf{q}}_{2i+1}^{\ell+1}) = \frac{1}{2} (\bar{\mathbf{q}}_{2i}^{\ell+1} + \bar{\mathbf{q}}_{2i+1}^{\ell+1}), \quad (3)$$

where $\bar{\mathbf{q}}_i^\ell$ are the average value of the cell Ω_i^ℓ . The same idea is extended for the two and three-dimensional cases.

The prediction operators are used to perform the opposite path of the projection operators, they allow to obtain the values of the finer cells using the values of the coarser ones. For each child cell to be predicted, there is a different prediction operator, represented by $P_{\ell \rightarrow \ell+1}^i : \bar{\mathbf{q}}^\ell \mapsto \bar{\mathbf{q}}^{\ell+1}$ for the one-dimensional case,

$P_{\ell \rightarrow \ell+1}^{i,j} : \bar{\mathbf{q}}^\ell \mapsto \bar{\mathbf{q}}^{\ell+1}$ for the two-dimensional case and $P_{\ell \rightarrow \ell+1}^{i,j,k} : \bar{\mathbf{q}}^\ell \mapsto \bar{\mathbf{q}}^{\ell+1}$ for the three-dimensional case. These operators yield a non-unique approximation of $\bar{\mathbf{q}}_i^{\ell+1}$ by interpolation. We use polynomial interpolation of second degree on the cell-averages, as proposed by Harten [12], which yields third-order accuracy. For the one-dimensional case, it follows that,

$$\tilde{\mathbf{q}}_{2i}^{\ell+1} = P_{\ell \rightarrow \ell+1}^0(\bar{\mathbf{q}}_{i-1}^\ell, \bar{\mathbf{q}}_i^\ell, \bar{\mathbf{q}}_{i+1}^\ell) = \bar{\mathbf{q}}_i^\ell - \frac{1}{8}(\bar{\mathbf{q}}_{i+1}^\ell - \bar{\mathbf{q}}_{i-1}^\ell) \quad (4a)$$

$$\tilde{\mathbf{q}}_{2i+1}^{\ell+1} = P_{\ell \rightarrow \ell+1}^1(\bar{\mathbf{q}}_{i-1}^\ell, \bar{\mathbf{q}}_i^\ell, \bar{\mathbf{q}}_{i+1}^\ell) = \bar{\mathbf{q}}_i^\ell + \frac{1}{8}(\bar{\mathbf{q}}_{i+1}^\ell - \bar{\mathbf{q}}_{i-1}^\ell). \quad (4b)$$

where $\tilde{\mathbf{q}}_i^\ell$ is an approximation of the value $\bar{\mathbf{q}}_i^\ell$. Interpolation operators for higher dimensions can be found in [26]. The operator must satisfy the properties of locality, requiring the interpolation for a child cell to be computed from the cell-averages of its parent and its nearest uncle cells in each direction; and consistency, $P_{\ell+1 \rightarrow \ell} \circ P_{\ell \rightarrow \ell+1} = \text{Identity}$.

The prediction operator is used to obtain the wavelet coefficients \mathbf{d}_i^ℓ of the finer cells. These coefficients are given by the difference between the cell average $\bar{\mathbf{q}}_i^\ell$ and the predicted value $\tilde{\mathbf{q}}_i^\ell$:

$$\mathbf{d}_i^\ell = \bar{\mathbf{q}}_i^\ell - \tilde{\mathbf{q}}_i^\ell. \quad (5)$$

The values \mathbf{d}_i^ℓ are also used for reconstructing the finer levels without interpolation errors. Their norm yields the local approximation error. Moreover, the information of the cell-average value of the two children is equivalent to the knowledge of the cell-average value of the parent and one independent detail. This can be expressed by $(\bar{\mathbf{q}}_{2i}^{\ell+1}, \bar{\mathbf{q}}_{2i+1}^{\ell+1}) \longleftrightarrow (\mathbf{d}_{2i}^{\ell+1}, \bar{\mathbf{q}}_i^\ell)$. This procedure can be applied recursively from level L down to the level 0 creating thus a multiresolution transform of the cell-average values as proposed by Harten [12]. Therefore, we have

$$\bar{\mathbf{q}}^L \mapsto (\bar{D}^L, \bar{D}^{L-1}, \dots, \bar{D}^1, \bar{\mathbf{q}}^0), \quad (6)$$

where \bar{D}^ℓ is the set of wavelet coefficients at level ℓ . Accordingly, the information of the cell-average values of all the leaves is equivalent to the knowledge of the cell-average value of the root and the wavelet coefficients of all the other nodes of the tree structure. For two and three dimensions, respectively, the information of the cell-averages of four and eight children is equivalent to the knowledge of three and seven wavelet coefficients in the different directions and the node value [12, 26].

3. Runge–Kutta methods

After discretising in space the initial value problem given in Eq. (1), the following system of ordinary differential equations in time is obtained:

$$\frac{d\bar{\mathbf{q}}}{dt} = f(t, \bar{\mathbf{q}}), \quad (7)$$

where $\mathbf{Q}(t=0) = \bar{\mathbf{q}}^0$ is the given initial condition. This system yields an equation for each leaf of the grid. By abuse of or to simplify notation the space discretised solution will be denoted again by $\bar{\mathbf{q}}$. The general formulation for an explicit s -stage Runge–Kutta (RK) method can be expressed at time t^{n+1} as:

$$\bar{\mathbf{q}}^{n+1} = \bar{\mathbf{q}}^n + \sum_{i=1}^s b_i k_i, \quad (8)$$

with

$$k_i = \Delta t^n f \left(t^n + c_i \Delta t^n, \bar{\mathbf{q}}^n + \sum_{j=1}^{i-1} a_{ij} k_j \right), \quad (9)$$

where a_{ij} , b_i and c_i are the Runge–Kutta coefficients, and Δt^n is the time-step used to perform the time evolution from the instant t^n to t^{n+1} . The actual convergence order of the RK method depends of the number of stages and a set of well selected RK coefficients.

In this work we consider second order RK methods (RK2) with coefficients c_i and a_{ij} in such a way that they are the same coefficients used in the first and second steps of the RK3 method. Namely, the values of these coefficients are $c_1 = 0$ and $c_2 = a_{21} = 1$. The other coefficients for RK2 are $b_1 = b_2 = \frac{1}{2}$. To perform RK3, further coefficients are $c_3 = \frac{1}{2}$, $a_{31} = a_{32} = \frac{1}{4}$, $b_1 = b_2 = \frac{1}{6}$ and $b_3 = \frac{2}{3}$.

Natural continuous Extension Runge–Kutta (NERK) method

The NERK method, originally introduced by [30], produces an approximation of the solution in the time interval $[t^n; t^n + \Delta t^n]$ using the same coefficients a_{ij} and c_i as in the standard Runge–Kutta methods. The difference between NERK and RK is the use of polynomials β_i instead of the constant coefficients b_i .

The NERK method can be expressed as,

$$\bar{\mathbf{q}}(t^n + \theta \Delta t^n) = \bar{\mathbf{q}}^n + \sum_{i=1}^s \beta_i(\theta) k_i, \quad \theta \in (0, 1], \quad (10)$$

where the polynomials β_i are given as a function of the coefficients b_i of the original RK method.

Using the proposed RK2 coefficients, the following polynomials for the two-stage NERK method are obtained using the methodology given in [23]:

$$\beta_1(\theta) = -\frac{1}{2}\theta^2 + \theta, \quad \beta_2(\theta) = \frac{1}{2}\theta^2. \quad (11)$$

In this work, the application of the NERK method consists in producing approximations of a solution at some intermediate time instants inside the interval $[t^n, t^n + \Delta t^n]$, depending on the choice for RK2 or RK3 time evolution.

Compact formulation

In order to reduce the memory allocation per cell when performing the time evolution, the compact formulation of the RK methods is of particular interest, used for example in [26]. Based on the standard RK methods, we can obtain the following compact formulation for the two and three stage methods,

- RK2: $\bar{\mathbf{q}}^* = \bar{\mathbf{q}}^n + \Delta t^n f(t^n, \bar{\mathbf{q}}^n), \quad \bar{\mathbf{q}}^{n+1} = \frac{1}{2}\bar{\mathbf{q}}^n + \frac{1}{2}\bar{\mathbf{q}}^* + \frac{1}{2}\Delta t^n f(t^n + \Delta t^n, \bar{\mathbf{q}}^*).$
- RK3: $\bar{\mathbf{q}}^* = \bar{\mathbf{q}}^n + \Delta t^n f(t^n, \bar{\mathbf{q}}^n), \quad \bar{\mathbf{q}}^{**} = \frac{3}{4}\bar{\mathbf{q}}^n + \frac{1}{4}\bar{\mathbf{q}}^* + \frac{1}{4}\Delta t^n f(t^n + \Delta t^n, \bar{\mathbf{q}}^*),$
 $\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\bar{\mathbf{q}}^n + \frac{2}{3}\bar{\mathbf{q}}^{**} + \frac{2}{3}\Delta t^n f\left(t^n + \frac{1}{2}\Delta t^n, \bar{\mathbf{q}}^{**}\right).$

When performing the time integration using a local time-stepping approach, *cf.* Section 4, some intermediate values are necessary. In this work, we propose to use the NERK method for that. However, the values k_i are not stored in the RK compact formulation. Hence the NERK solution must be adapted to be compatible with the compact RK method.

The following steps produce a NERK/RK2 approximation at the time instant $t^n + \frac{1}{2}\Delta t^n$. Due to the memory management of the numerical code, where the fluxes are stored at the same memory allocation, each one of the following steps is executed immediately after its corresponding compact RK step:

$$\bar{\mathbf{q}}_{\theta=\frac{1}{2}}^* = \bar{\mathbf{q}}^n + \frac{3}{8}\Delta t^n f(t^n, \bar{\mathbf{q}}^n), \quad \bar{\mathbf{q}}_{\theta=\frac{1}{2}} = \bar{\mathbf{q}}_{\theta=\frac{1}{2}}^* + \frac{1}{8}\Delta t^n f(t^n + \Delta t^n, \bar{\mathbf{q}}^*). \quad (12)$$

where the values $f(t^n, \bar{\mathbf{q}}^n)$ and $f(t^n + \Delta t^n, \bar{\mathbf{q}}^*)$ are the same as those obtained for the compact RK formulation. In this formulation, the values $\bar{\mathbf{q}}_{\theta=\frac{1}{2}}$ and $\bar{\mathbf{q}}_{\theta=\frac{1}{2}}^*$ are stored at the same memory allocation.

For the RK3 time evolution, second order approximations at the time instants $t^n + \frac{1}{4}\Delta t^n$ and $t^n + \frac{3}{4}\Delta t^n$ become necessary. These approximations are required to compute the third step of the RK3. However, the NERK/RK3 method only yields these information after the third step. The proposed solution for this problem is to obtain these approximations via NERK/RK2. Then, it is possible to obtain approximations at these desired instants immediately after the second step of the RK method, under the condition that the coefficients a_{ij} and c_i of both, RK2 and RK3, methods are the same. Therefore we use the following approximations:

- at $t^n + \frac{1}{4}\Delta t^n$: $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}^* = \bar{\mathbf{q}}^n + \frac{7}{32}\Delta t^n f(t^n, \bar{\mathbf{q}}^n), \quad \bar{\mathbf{q}}_{\theta=\frac{1}{4}} = \bar{\mathbf{q}}_{\theta=\frac{1}{4}}^* + \frac{1}{32}\Delta t^n f(t^n + \Delta t^n, \bar{\mathbf{q}}^*).$
- at $t^n + \frac{3}{4}\Delta t^n$: $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}^* = \bar{\mathbf{q}}^n + \frac{15}{32}\Delta t^n f(t^n, \bar{\mathbf{q}}^n), \quad \bar{\mathbf{q}}_{\theta=\frac{3}{4}} = \bar{\mathbf{q}}_{\theta=\frac{3}{4}}^* + \frac{9}{32}\Delta t^n f(t^n + \Delta t^n, \bar{\mathbf{q}}^*).$

These approximations can be performed using similar memory management ideas as for the RK2 evolution.

4. Local time-stepping

To improve further the computational efficiency of the MR method, a local time-stepping approach was proposed in [6]. This approach consists in using an adapted time-step for each leaf individually. This time step is obtained accordingly to the spatial size of the leaf. Thus, small time steps are only used for fine scale leaves, while large time-steps can be used for coarser leaves. This is possible without violating the stability condition of the explicit time discretisation, as shown in [6].

Solutions with point-like singularities are well adapted and yield highly efficient multiresolution representations. For those, the MRLT approach is found to be most efficient. Besides, the adaptive grid created for the MRLT scheme is the same graded tree structure used in the MR scheme [26]. The update procedure for the trees during MRLT schemes is discussed in Section 4.3.

We suppose that the CFL condition implies a time-step Δt for the most refined level. In the LT scheme, each cell of level ℓ performs its time evolution with a proper time step given by:

$$\Delta t_\ell = 2^{L-\ell}\Delta t, \tag{13}$$

where L is the finest level of the grid. From this point, the notation is adjusted in order to facilitate the understanding. Moreover, considering the Courant number σ depending on the ratio between Δt_ℓ and Δx_ℓ , and using the relations between the cell size and the time-step of different levels, the value σ obtained for every cell will thus be the same.

As illustrated in Figure 2, due to the scale dependent time-stepping of the LT scheme, not all leaves of the coarser scales will be evolved during an iteration of the time evolution. We define the coarsest level where the leaves must be evolved in a certain iteration n to be the minimum scale level in which the modulo operator between n and $2^{L-\ell}$ is zero and we denote it as ℓ_{\min} .

The above approach is restricted to second order time accuracy, because the internal steps of standard Runge–Kutta schemes are not compatible with the dyadic grid size. The reason is that the intermediate time steps of higher order RK schemes (order larger than two) do not correspond to the time instants of the solution obtained by the RK method when using the dyadic grid size [6]. A possible solution to overcome this limitation is to use NERK schemes. Their polynomial approximation in time is used to evaluate the solution at the intermediate time instants imposed by the dyadic spatial grid size.

The implementation of high order LT schemes leads to three synchronization challenges during the time evolution of a leaf with a neighbour at a different refinement level. Those challenges are discussed in the following subsections. The whole LT method as proposed in this work is performed as in Algorithm 3.

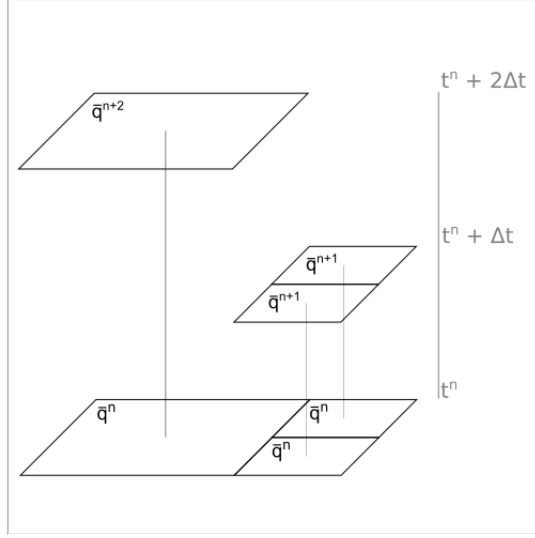


Figure 2: LT evolution of adjacent cells at different scales. The finer cells in this representation need a second time evolution to reach the instant t^{n+2} .

4.1. Synchronization during the Runge–Kutta iteration

Considering the restriction that during an iteration of LT schemes only leaves of refinement level greater or equal ℓ_{\min} are evolved, we know that all of these leaves have their solutions at the same time instant t^n , requiring no synchronization to perform the first RK step. Thus, the first RK step can be done with the MR method using the proper Δt_ℓ value. After performing the first step of the Runge–Kutta method, the leaves at each level are evolved with their own time step, obtaining a first order solution at time instant $t^n + \Delta t_\ell$. Using the fluxes obtained in this step, the values $\bar{\mathbf{q}}_{\theta=\frac{1}{2}}^*$, for RK2, or $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}^*$ and $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}^*$ for RK3, are computed at the leaves of every level evolved in this iteration.

However, for the next RK steps, due to the different time step size, the solution values after the RK step are given in a different time instant for each refinement level. This implies that some synchronization has to be performed.

4.1.1. Second Runge–Kutta step synchronization

In order to perform the second step of the RK method, it is necessary to compute the flux $f(t^n + \Delta t_\ell, \bar{\mathbf{q}}^*)$. This flux can be interpreted as the flux between leaves at the time instant $t^n + \Delta t_\ell$, where the leaf value is a first order approximation $\bar{\mathbf{q}}^* = \bar{\mathbf{q}}^n + a_{21}k_1$. The challenge here lies in the fact that when this approximation is obtained at a finer scale, due to the different time step size used, it is located at an earlier time instant with respect to the approximation at a coarser scale. This situation implies that the flux computation for each scale has to be performed at a different time instant, as shown in the scheme presented in Figure 3. The values required for both cases of the second RK step are not available after the first RK step, requiring thus a synchronization procedure to obtain those values for each scale. Then the second RK step in this scale can be performed. To obtain the missing values to compute the fluxes in the situation given in Figure 3(a), a tree refreshing procedure, described in Section 4.2, must be performed in order to project the solution for scales $\ell > \ell_{\min}$ at time instant $t^n + \Delta t_\ell$ onto the scale $\ell - 1$ at instant $t^n + \Delta t_{\ell-1}$.

The proposed synchronization methodology to perform the second RK step consists in using a first order approximation $\bar{\mathbf{q}}_\ell^n + \frac{1}{2}a_{21}k_1$ as a solution at the time instant $t^n + \frac{1}{2}\Delta t_\ell$ for every $\ell \neq L$. This approximation is in the same time instant as the solution in the next finer level. The use of this approximation consists in predicting the values of the virtual leaves at level $\ell + 1$, at the proper time instant, before performing

the flux computations of the leaves at level $\ell + 1$. This approach is illustrated for the situation presented in Figure 3(a).

In order to simplify the algorithm, the prediction of the virtual leaves at the coarser scales at the proper time instant, necessary to update the next finer level, is performed during the tree refreshing process, explained in detail for every RK step, in Section 4.2. In this Section, we focus on the update of the virtual leaves at level $\ell + 1$ in order to perform the flux computations on leaves of level ℓ , as presented in Figure 3(b).

Initially, the flux computations for the second RK step are performed on the leaves of level L . This choice is due to the fact that at this level, there are no interfaces of the type leaf/internal node, avoiding thus the situation shown in Figure 3(b).

Having updated the level L , the same process is repeated recursively for the levels $L - 1$ down to $L = \ell_{\min}$. However, due to the leaf/internal node scenario, the leaves and virtual leaves of the previously updated level $\ell + 1$ must be synchronized at the instant $t^n + \Delta t_\ell$, which is equivalent to the instant $t^n + 2\Delta t_{\ell+1}$, in order to perform the flux computations of the next coarser level. For that we propose the use of an extrapolated value, resulting in the situation presented in Figure 3(b).

In this work, we propose to obtain an extrapolation at time instant $t^n + 2\Delta t_{\ell+1}$, which allows to compute the fluxes for the second step of the RK method. For this, the value k_2 is used as the value k_1 in a second RK1 time evolution. The first order approximation of the solution at the instant $t^n + 2\Delta t_{\ell+1}$ is computed as:

$$\bar{\mathbf{q}}_{\ell+1}(t^n + 2\Delta t_{\ell+1}) = \bar{\mathbf{q}}_{\ell+1}^* + \Delta t_{\ell+1} f(t + \Delta t_{\ell+1}; \bar{\mathbf{q}}_{\ell+1}^*) = \bar{\mathbf{q}}_{\ell+1}^n + k_1 + k_2 \quad (14)$$

where $\bar{\mathbf{q}}_{\ell+1}^*$ is the first RK step, and $\Delta t_{\ell+1} f(t + \Delta t_{\ell+1}; \bar{\mathbf{q}}_{\ell+1}^*)$ is the flux of the second RK step. Note that this approximation is only possible due to the choice of the coefficients $a_{11} = b_1 = 1$.

Once the leaves of level $\ell + 1$ are extrapolated to the instant $t^n + 2\Delta t_{\ell+1}$, the virtual leaves of this level are also updated. However, in order to predict the value of the virtual leaves at level $\ell + 1$, the values of the virtual leaves and internal nodes of level ℓ must be synchronized at instant $t^n + \Delta t_\ell$ first.

The solution of the virtual leaves and internal nodes of level ℓ in this time instant is obtained during the tree refreshing process. However, in order to improve the solution of the internal nodes, their values at time instant $t^n + \Delta t_\ell$ are updated after the evolution of the leaves of level $\ell + 1$. This update is performed via projection of the extrapolated solution at instant $t^n + 2\Delta t_{\ell+1}$ from the level $\ell + 1$ to ℓ . In contrast to the leaves, which have an approximation at this instant, the internal nodes do not. The values for the internal nodes are obtained via linear extrapolation:

$$\bar{\mathbf{q}}_{\ell+1}(t^n + 2\Delta t_{\ell+1}) = 2\bar{\mathbf{q}}_{\ell+1}^* - \bar{\mathbf{q}}_{\ell+1}^n. \quad (15)$$

This approximation is used to perform the projection of the solution of the internal node to the next finer level ℓ . The projection procedure is given in Algorithm 4. After the projection procedure, the virtual leaves of level $\ell + 1$ have their predicted values. Then, the flux computations and the update of the leaves at level ℓ can be performed. The second step of the RK method is given in Algorithm 5.

During the second RK step, the fluxes are also used for computing the second step of the NERK approximation with second order at the time instants $t^n + \frac{1}{2}\Delta t_\ell$ for RK2 or $t^n + \frac{1}{4}\Delta t_\ell$ and $t^n + \frac{3}{4}\Delta t_\ell$ for RK3 time evolution.

As stated before, the NERK approximation for the RK2 time evolution is used to solve the synchronization according the time evolution problem given in Section 4.3. For the RK3 time evolution, the NERK approximations are used to perform the third RK step.

After the second step of the RK evolution, a solution at the time instant $t^n + \Delta t_\ell$ for the RK2 method, and at $t^n + \frac{1}{2}\Delta t_\ell$ for the RK3 method is obtained. In order to prepare the tree structure for the next RK2

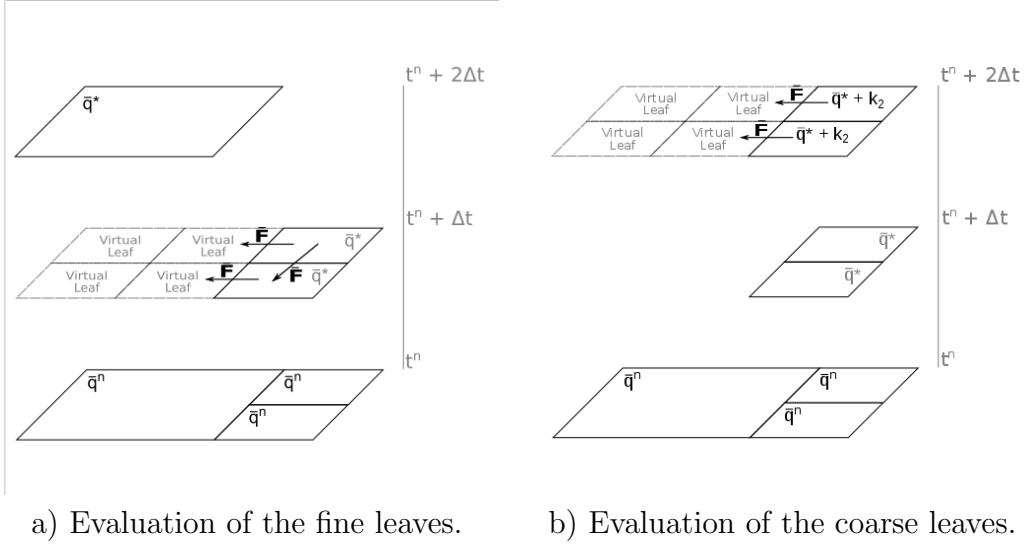


Figure 3: Schematic view of the synchronization issues for the second RK step. a) presents the evaluation of flux computations $\bar{\mathbf{F}}$ at instant $t^n + \Delta t$. For adjacent leaves at the same level we can compute the fluxes directly. However, to compute the fluxes between different levels, we need the virtual leaves of the coarser leaf at the same time instant of the finer leaves. For that, first, we interpolate the coarser leaf $\bar{\mathbf{q}}$ at instance $t^n + \Delta t$ from its respective values at t^n and $t^n + 2\Delta t$. Then, we predict its value at the finer level and obtain its virtual leaves. After that, we evaluate the flux, and then, the RK step for the finer leaves. b) presents the evolution of the flux at $t^n + 2\Delta t$ of the coarser leaf. First, we predict the virtual leaves of the coarse leaf, then we compute $\bar{\mathbf{q}}^* + k_2$, *i.e.* a RK1 step of the finer leaves, which yields a first order approximation. After that, we can compute the flux and evaluate the leaves.

iteration and for the next RK3 step, another tree refreshing procedure must be applied. This procedure is given in detail in Section 4.2. After this process, the information obtained for the cells (internal nodes, leaves and virtual leaves) in every refinement level are illustrated in Figure 4.

4.1.2. Third Runge–Kutta step synchronization

The third step is done only in the case of the RK3 time evolution. As for the second step of the RK evolution, in the third step, the evolution must be done first on the finest scale in order to obtain the necessary values to perform the evolution on the next coarser scale. However there are, indeed, slightly different synchronization issues in this case which we discuss in the following.

As performed in the second RK step, the virtual leaves and the internal nodes have their values updated during the tree refreshing process, the details are discussed in the next section.

To perform the third RK step, the fluxes of level L are computed in the proper time instant $t^n + \frac{1}{2}\Delta t_L$. Then, the third RK step yields a third order solution at the time instant $t^n + \Delta t_L$.

Again as for the second step, this solution is projected onto the next coarser level in order to update the virtual leaves at level L at the instant $t^n + \Delta t_L$, where the fluxes for the third step of the level $L - 1$ shall be computed.

The projection procedure for the third RK step is, as mentioned above, slightly different from the projection procedure in the second step. In this case, the solution at the leaves at level ℓ after the third RK step matches with the time instant where the fluxes of level $\ell - 1$ are computed. Then, the solution is projected from level ℓ to level $\ell - 1$ by simple averaging, obtaining thus a solution at $t^n + \frac{1}{2}\Delta t_{\ell-1}$. This solution is used as an approximation for $\bar{\mathbf{q}}^{**}$. Then, the value of the internal nodes at the instant $t^n + \Delta t_{\ell-1}$ should be updated with a second order solution in order to continue the projections recursively during the third RK step. This update is performed using the following relation obtained from the NERK scheme:

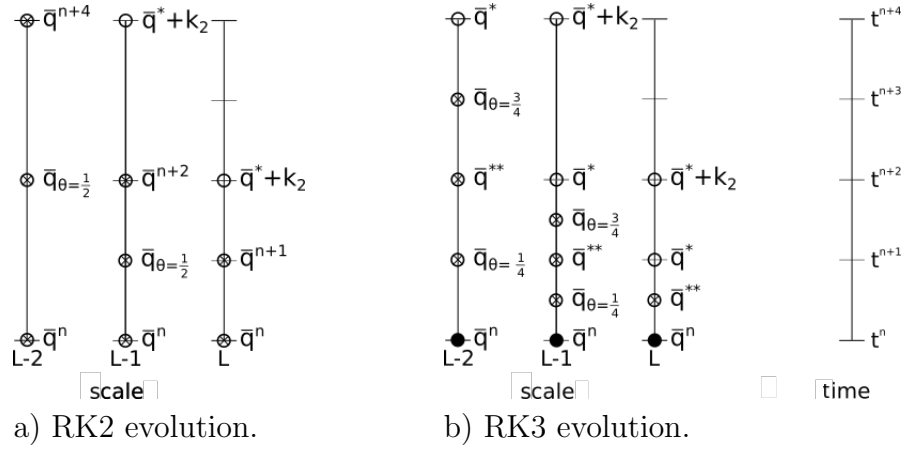


Figure 4: Scheme of the obtained values and its respective instants for cells in each level after performing the second RK step in the LT method. a) for RK2 and b) for the RK3 time evolution. The filled, crossed and clear circles represent an available solution, of first, second and third order, respectively, at the corresponding time instant.

$$\bar{\mathbf{q}}_\ell(t^n + \Delta t_\ell) = \bar{\mathbf{q}}_\ell^n + 2 \left(\bar{\mathbf{q}}_{\ell, \theta = \frac{3}{4}} - \bar{\mathbf{q}}_{\ell, \theta = \frac{1}{4}} \right). \quad (16)$$

The projection procedure inside the third RK step is given in Algorithm 6, and the execution of the third step of the RK evolution in Algorithm 7.

4.2. Tree refreshing with time synchronization

In the MR scheme, due to the same time step for every scale, the leaves of every scale store values corresponding to the same time instant. Therefore, projections of the leaves onto internal nodes receive values at the same time instant. However, in the LT approach, due to fact that leaves of different scales are evolved with different time steps, the tree refreshing may project values at different time instants related to the leaves at the same scale. Accordingly, we need a procedure before each RK step internally to the time cycle, to project the solution to an internal node at its corresponding time instant.

Moreover, these synchronizations are necessary to predict the values of the virtual leaves for the next steps of the RK method, at the required time instant. Only after that, the flux computations can be performed.

Before the first RK step, every scale of level ℓ_{\min} or greater, which must be evolved, has its solution at the same time instant. Therefore, the values of the virtual leaves can be predicted normally, except at level ℓ_{\min} , which has its virtual leaves predicted using the solution at $t^n + \frac{1}{2}\Delta t_{\ell_{\min}-1}$ from the level $\ell_{\min} - 1$.

After the first step of the RK method, each leaf yields a first order solution at the time instant $t^n + \Delta t_\ell$, which corresponds to the time instant $t^n + \frac{1}{2}\Delta t_{\ell-1}$. This implies that the projection of the leaves from level ℓ to $\ell - 1$ produces an approximation at that time instant. The value of an internal node at this instant is given by the mean value of its children:

$$\bar{\mathbf{q}}_{\ell-1} \left(t^n + \frac{1}{2}\Delta t_{\ell-1} \right) = \bar{\mathbf{q}}_{\ell-1}^n + \frac{1}{2}\Delta t_{\ell-1} f \left(t^n, \bar{\mathbf{q}}_{\ell-1}^n \right) = \frac{1}{2^d} \sum_{i=1}^{2^d} \left[\bar{\mathbf{q}}_{\ell, i}^n + \Delta t_\ell f \left(t^n, \bar{\mathbf{q}}_{\ell, i}^n \right) \right] \quad (17)$$

where d is the number of dimensions of the problem and $\bar{\mathbf{q}}_{\ell, i}^n$ is the solution of the children cell i . This value is stored in order to predict the values of the virtual leaves of level ℓ for the second RK step. Furthermore, this value must be extrapolated to the instant $t^n + \Delta t_{\ell-1}$, obtaining the value $\bar{\mathbf{q}}_{\ell-1}^*$. This value, used to

update the virtual leaves of level ℓ during the second RK step, is obtained by linear extrapolation:

$$\bar{\mathbf{q}}_{\ell-1}^* = 2\bar{\mathbf{q}}_{\ell-1} \left(t^n + \frac{1}{2}\Delta t_{\ell-1} \right) - \bar{\mathbf{q}}_{\ell-1}^n, \quad (18)$$

where the value $\bar{\mathbf{q}}_{\ell-1}^n$ is the solution before the time evolution. This value should be stored for every node before the RK procedure. The tree refreshing procedure after the first RK step is given in Algorithm 8.

When performing the RK3 time evolution, before the third RK step, besides the solution, the values $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}$ and $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}$ must be obtained for the internal nodes. The solution in those instants are required to predict the solution of the virtual leaves in the next finer level at the instant required for this RK step.

Those values are obtained through projections from the level L down to the level ℓ_{\min} , initially by projecting the solution $\bar{\mathbf{q}}_{\ell}^{**}$ at the time instant $t^n + \frac{1}{2}\Delta t_{\ell}$, obtaining thus a value at $t^n + \frac{1}{4}\Delta t_{\ell-1}$, which is used as an approximation for $\bar{\mathbf{q}}_{\ell-1, \theta=\frac{1}{4}}$.

Using this solution, the Equation 10 and the value $\bar{\mathbf{q}}_{\ell}^*$, computed during the projection before the second RK step, we obtain the following relation to reconstruct the solution inside the interval $[t^n; t^n + \Delta t^n]$:

$$\bar{\mathbf{q}}_{\ell}(t^n + \theta\Delta t_{\ell}) = [1 - \theta - 12\theta^2] \bar{\mathbf{q}}_{\ell}^n + [\theta - 4\theta^2] \bar{\mathbf{q}}_{\ell}^* + 16\theta^2 \bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}}. \quad (19)$$

This expression is based on the NERK scheme presented in Section 3. Using the value $\theta = \frac{3}{4}$, we have:

$$\bar{\mathbf{q}}_{\ell, \theta=\frac{3}{4}} = -\frac{13}{2}\bar{\mathbf{q}}_{\ell}^n - \frac{3}{2}\bar{\mathbf{q}}_{\ell}^* + 9\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}} \quad (20)$$

Subsequently, using these values, the value $\bar{\mathbf{q}}_{\ell}^{**}$ can be computed as:

$$\bar{\mathbf{q}}_{\ell}^{**} = -\frac{11}{2}\bar{\mathbf{q}}_{\ell}^n - \frac{3}{2}\bar{\mathbf{q}}_{\ell}^* + 8\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}} \quad (21)$$

The solution $\bar{\mathbf{q}}_{\ell}^{**}$ allows to continue the projection procedure recursively by obtaining the value $\bar{\mathbf{q}}_{\ell-1, \theta=\frac{1}{4}}$ at the next coarser level, while the solution $\bar{\mathbf{q}}_{\ell, \theta=\frac{3}{4}}$ is used to compute the value of the virtual leaves at level $\ell + 1$ on the following iterations, as explained in Section 4.1.

The tree refreshing procedure, illustrated in Figure 5, consists in projecting the solution $\bar{\mathbf{q}}_{\ell}^{**}$ from the level L onto the level $L - 1$. This solution is approximated as $\bar{\mathbf{q}}_{L-1, \theta=\frac{1}{4}}$. Then, using the previously obtained value $\bar{\mathbf{q}}_{L-1}^*$, the values $\bar{\mathbf{q}}_{L-1, \theta=\frac{3}{4}}$ and $\bar{\mathbf{q}}_{L-1}^{**}$ can be obtained using Equations (20) and (21). This procedure is recursively repeated down to the coarsest scale ℓ_{\min} .

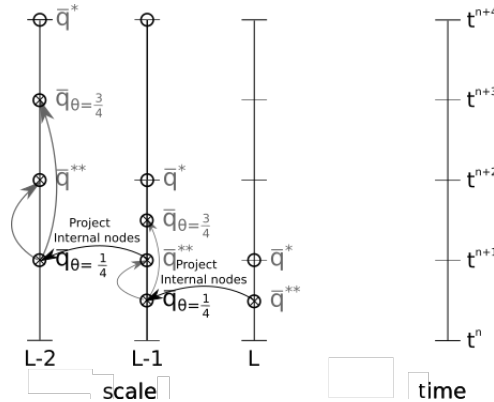


Figure 5: Projection scheme before the third RK step.

In the third RK step, the fluxes are computed at the intermediary time instant $t^n + \frac{1}{2}\Delta t_\ell$. For an adjacent coarser leaf, this instant is $t^n + \frac{1}{4}\Delta t_{\ell-1}$ or $t^n + \frac{3}{4}\Delta t_{\ell-1}$, depending on the current iteration number. Basically, if the current scale is ℓ_{\min} then the NERK value used to compute the virtual leaves is available at time instant $t^n + \frac{3}{4}\Delta t_{\ell_{\min}-1}$. Otherwise, the value at the time instant $t^n + \frac{1}{4}\Delta t_{\ell-1}$ is used.

The choice which value of the NERK approximation shall be used in the flux computations is shown in Figure 6, where we choose $\ell_{\min} = L - 2$.

Once the projections have been performed from the level L until the level ℓ_{\min} , the virtual leaves have their values $\bar{\mathbf{q}}^{**}_\ell$ predicted using the values $\bar{\mathbf{q}}_{\ell-1, \theta=\frac{3}{4}}$, in case of $\ell = \ell_{\min}$; or using $\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}}$, otherwise.

Then, the values $\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}}$ and $\bar{\mathbf{q}}_{\ell, \theta=\frac{3}{4}}$ for these virtual leaves are obtained using the Equations (20) and (21). Those values are required here to predict the values of the virtual leaves at the next finer scale.

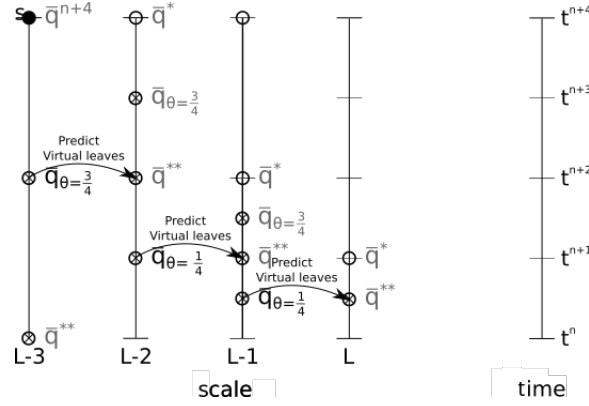


Figure 6: Scheme of the choice of the NERK solution to predict the virtual leaves for the third RK step in a LT iteration with $\ell_{\min} = L - 2$. The solution $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}$ is used to predict the $\bar{\mathbf{q}}^{**}$ values at level ℓ_{\min} . For the other scales, the solution $\bar{\mathbf{q}}^{**}$ is predicted with the values $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}$.

The tree refreshing procedure before the third RK step is detailed in Algorithm 9.

4.3. Synchronization after the time evolution

After finishing the RK steps, the leaves at level ℓ perform a complete time evolution with a time step that is twice the time step of the leaves at level $\ell + 1$. In order to perform the next time evolution at scale ℓ , the finer scale leaves must reach the same time instant. For that, a second time evolution is required for this finer scale leaf.

Moreover, in the next iteration of the time evolution, a new value for ℓ_{\min} is obtained. To predict the virtual leaves at the new scale ℓ_{\min} at the proper time instant, the values $\bar{\mathbf{q}}_{\ell_{\min}-1, \theta=\frac{1}{2}}$ for RK2 or $\bar{\mathbf{q}}_{\ell_{\min}-1}^{**}$ for RK3 are used. For the scales finer than ℓ_{\min} , the virtual leaves are predicted with the MR method.

To solve this synchronization challenge and to maintain the approximation order, a high order approximation at time step $t^n + \frac{1}{2}\Delta t_\ell$, where ℓ is the refinement level of the leaf, is required. In this case, in the tree terminology, it means that for the coarser leaves, an approximation of its virtual children at the instant $t^n + \frac{1}{2}\Delta t_\ell$ is required.

To obtain these values, it is necessary to obtain approximations at the instant $t^n + \frac{1}{2}\Delta t_\ell$ during the Runge–Kutta evolution for every cell of refinement level $\ell \neq L$ and its neighbours, including internal nodes. Then, the values of the virtual children are obtained by the prediction procedure.

For the RK2 time evolution, we use the NERK approach to obtain the approximation with $\theta = \frac{1}{2}$. However, the implementation of the MRLT/NERK approach for the RK3 time evolution requires extra memory. Hence we use the approximation $\bar{\mathbf{q}}^{**}$, obtained after the second step of the compact RK, which

is at the instant $t^n + \frac{1}{2}\Delta t_\ell$, instead of the NERK approximation with $\theta = \frac{1}{2}$ in order to reduce the number of variables to be stored in the problem.

Besides those values, the updated solution, at time instant $t^n + \Delta t_\ell$, is projected onto the coarser level at the time instant $t^n + \frac{1}{2}\Delta t_{\ell-1}$ by simple averaging. This solution is then used as an approximation for $\bar{\mathbf{q}}_{\ell-1, \theta=\frac{1}{2}}$, in the RK2 time evolution, or $\bar{\mathbf{q}}_{\ell-1}^{**}$ for the RK3 time evolution. Using the NERK equations given in Section 3 and the current value of the internal node at $t^n + \Delta t_{\ell-1}$ (RK first step), the solution at instant $t^n + \Delta t_\ell$ is obtained by the following relation:

$$\bar{\mathbf{q}}_\ell(t^n + \Delta t_\ell) = -2\bar{\mathbf{q}}_\ell^n - \bar{\mathbf{q}}_\ell^* + 4\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{2}}. \quad (22)$$

This value is used to compute the value of the virtual leaves at level $\ell_{\min} + 1$ in the following iterations at the second RK step, as explained in Section 4.1.1.

Adapting the grid during local time-stepping

During the LT scheme, in most parts of the numerical simulation, some scales have results at different time instants. Therefore, in order to avoid errors to be caused by converting a cell with a solution value at a determined time instant to another scale where its cells are in a different time instant, there is a need for a criterion when adapting the solution during the LT scheme.

This criterion consists in combining or splitting only leaves from a scale to another scale whose leaves are at the same time instant. These scales can be easily detected due to the fact that the time evolution procedure is applied only in scales at the same time instant. In other words, to split a cell into more refined ones, both scales must be included into the current time evolution iteration. The same is valid to combine cells into a coarser one.

Moreover, beyond this restriction, the remeshing process is identical to the remeshing process of the multiresolution method.

4.4. Some remarks on the convergence and stability of MRLT/NERK schemes

In this section, we perform a stability analysis to check the convergence order of the MRLT/NERK3 method. For that, we compute the interpolation and extrapolation errors in each approximation required for the method, considering the interface between cell of different levels.

To analyze the stability typically a simple linear ODE,

$$\frac{dq}{dt} = \lambda u \quad (23)$$

where a constant complex-valued coefficient $\lambda \in \mathbb{C}$ with negative or zero real part is considered. The above equation is completed with an initial condition. Using Fourier analysis of linear PDEs it can be shown that imaginary values of λ correspond to pure advection, while real (negative) values correspond to pure diffusion equations.

One-step schemes, including Runge–Kutta methods can then be written in the following form,

$$\hat{\mathbf{q}}^{n+1} = g(\lambda\Delta t) \hat{\mathbf{q}}^n \quad (24)$$

where g is a polynomial. In particular, for RK1, RK2 and RK3 we respectively have $g(\lambda\Delta t) = 1 + \lambda\Delta t$, $g(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}\lambda^2\Delta t^2$ and $g(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}\lambda^2\Delta t^2 + \frac{1}{6}\lambda^3\Delta t^3$, which correspond to the truncated Taylor series of $\exp(\lambda\Delta t)$. A method is called of convergence order n if its polynomial $g(\lambda\Delta t)$ reconstructs the Taylor series until the $\lambda^n\Delta t^n$ term.

To compute the polynomial g for the MRLT/NERK methods, the analysis is performed by computing and inserting each error ϵ_i obtained in every approximation required to perform the MRLT method.

The errors obtained in those approximations are inserted into the flux computations of the stability model as follows:

$$f(\bar{\mathbf{q}}) = f(\hat{\mathbf{q}} - \epsilon), \quad (25)$$

where $\bar{\mathbf{q}}$ is an approximation obtained during the MRLT/NERK method and $\hat{\mathbf{q}}$ is the solution that would be obtained by the regular RK method.

In the following, we obtain the errors ϵ for each approximation used for the first RK evolution on both finer and coarser scales. In this evolution, both scales are initially at the same time instant, so there is no approximation error for the first RK step (k_1). In other words, it means that $\bar{\mathbf{q}}^n = \hat{\mathbf{q}}^n$.

In this section, we consider the RK3 scheme as given in Section 3.

Considering that the solution $\bar{\mathbf{q}}^n$ is at the same instant for every scale to be updated in the time evolution, the first RK stage reads,

$$\bar{\mathbf{q}}^* = \bar{\mathbf{q}}^n + \Delta t^n f(t^n, \bar{\mathbf{q}}^n) \quad (26)$$

and yields no interpolation errors due to the LT approach for both scales. That leads to $\bar{\mathbf{q}}^* = \hat{\mathbf{q}}^*$.

Approximations after the first RK step

Here we compute the errors obtained from the predictions and projections after the first RK step, as performed in Section 4.1.1. The solution of the intermediary solution $\bar{\mathbf{q}}^*$ is at a different time instant for each refinement level, requiring thus a series of interpolations and extrapolations, which introduces errors ϵ_1 and ϵ_2 into the scheme.

- ϵ_1 : Prediction error in the finer leaf.

Error obtained in the prediction of $\bar{\mathbf{q}}^*$ in the finer leaf using the values of the solution in the lower level with half of the time-step.

$$\epsilon_1 = \hat{\mathbf{q}}_\ell^* - P_{\ell-1 \rightarrow \ell} \left[\bar{\mathbf{q}}_{\ell-1}^n - \frac{\Delta t_{\ell-1}}{2} f(\bar{\mathbf{q}}_{\ell-1}^n) \right] = 0. \quad (27)$$

Proof: Considering $f(\bar{\mathbf{q}}_{\ell-1}^n) = \lambda \bar{\mathbf{q}}_{\ell-1}^n$

$$\epsilon_1 = \hat{\mathbf{q}}_\ell^n + \Delta t_\ell \lambda \hat{\mathbf{q}}_\ell^n - P_{\ell-1 \rightarrow \ell} \left[\bar{\mathbf{q}}_{\ell-1}^n - \frac{\Delta t_{\ell-1}}{2} \lambda \bar{\mathbf{q}}_{\ell-1}^n \right] \quad (28)$$

$$\epsilon_1 = \hat{\mathbf{q}}_\ell^n + \Delta t_\ell \lambda \hat{\mathbf{q}}_\ell^n - \bar{\mathbf{q}}_\ell - \frac{\Delta t_{\ell-1}}{2} \lambda \bar{\mathbf{q}}_\ell \quad (29)$$

$$\epsilon_1 = 0 \quad (30)$$

- ϵ_2 : Projection error in the coarser leaf. Error obtained in the projection to obtain $\bar{\mathbf{q}}^*$ in the coarser level. This solution is a first order extrapolation to the time instant $t^n + \Delta t_{\ell-1}$ for the leaves in the finer level.

$$\epsilon_2 = \hat{\mathbf{q}}_\ell^* - P_{\ell+1 \rightarrow \ell} \left[\underbrace{\bar{\mathbf{q}}_{\ell+1}^n + \Delta t_{\ell+1} f(\bar{\mathbf{q}}_{\ell+1}^n)}_{\text{First RK step in finer leaf}} + \underbrace{\Delta t_{\ell+1} f(\bar{\mathbf{q}}_{\ell+1}^*)}_{\text{extrapolation to } t^n + \Delta t_{\ell-1}} \right] = -\frac{\Delta t_\ell^2 \lambda^2}{4} \hat{\mathbf{q}}_\ell^n \quad (31)$$

Proof: Here we consider the errors caused in the previous approximations in the flux terms. Using $\mathbf{q}^* = \mathbf{q} + \Delta t f(\mathbf{q}) - \epsilon_1$ inside the flux term:

$$\epsilon_2 = \hat{\mathbf{q}}_\ell^n + \Delta t_\ell f(\hat{\mathbf{q}}_\ell^n) - P_{\ell+1 \rightarrow \ell} \left[\bar{\mathbf{q}}_{\ell+1}^n + \Delta t_{\ell+1} f(\bar{\mathbf{q}}_{\ell+1}^n) + \Delta t_{\ell+1} f(\hat{\mathbf{q}}_{\ell+1}^n + \Delta t_{\ell+1} f(\hat{\mathbf{q}}_{\ell+1}^n) - \epsilon_1) \right] \quad (32)$$

Considering $f(\bar{\mathbf{q}}^n_{\ell+1}) = \lambda \bar{\mathbf{q}}^n_{\ell+1}$ we have

$$\epsilon_2 = \hat{\mathbf{q}}^n_\ell + \Delta t_\ell \lambda \hat{\mathbf{q}}^n_\ell - P_{\ell+1 \rightarrow \ell} \left[\hat{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1} \lambda \hat{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1} \lambda (\hat{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1} \lambda (\hat{\mathbf{q}}^n_{\ell+1})) \right] \quad (33)$$

Applying the projection operator we get

$$\epsilon_2 = \hat{\mathbf{q}}^n_\ell + \Delta t_\ell \lambda \hat{\mathbf{q}}^n_\ell - \hat{\mathbf{q}}^n_\ell - \Delta t_{\ell+1} \lambda \hat{\mathbf{q}}^n_\ell - \Delta t_{\ell+1} \lambda (\hat{\mathbf{q}}^n_\ell + \Delta t_{\ell+1} \lambda \hat{\mathbf{q}}^n_\ell) \quad (34)$$

$$\epsilon_2 = \Delta t_\ell \lambda \hat{\mathbf{q}}^n_\ell - \frac{\Delta t_\ell}{2} \lambda \hat{\mathbf{q}}^n_\ell - \frac{\Delta t_\ell}{2} \lambda \left(\hat{\mathbf{q}}^n_\ell + \frac{\Delta t_\ell}{2} \lambda \hat{\mathbf{q}}^n_\ell \right) \quad (35)$$

$$\epsilon_2 = -\frac{\Delta t_\ell^2 \lambda^2}{4} \hat{\mathbf{q}}^n_\ell \quad (36)$$

Once the interpolation errors are obtained, the second Runge–Kutta step is performed for the finer scale:

$$\bar{\mathbf{q}}^{**} = \frac{3}{4} \bar{\mathbf{q}}^n + \frac{1}{4} \bar{\mathbf{q}}^* + \frac{1}{4} \Delta t^n f(t^n + \Delta t^n, \bar{\mathbf{q}}^*) \quad (37)$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4} \hat{\mathbf{q}}^n + \frac{1}{4} \hat{\mathbf{q}}^* + \frac{1}{4} \Delta t^n f(\hat{\mathbf{q}}^* - \epsilon_1) \quad (38)$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4} \hat{\mathbf{q}}^n + \frac{1}{4} \hat{\mathbf{q}}^* + \frac{1}{4} \Delta t^n f(\hat{\mathbf{q}}^*) \quad (39)$$

$$\bar{\mathbf{q}}^{**} = \hat{\mathbf{q}}^{**} \quad (40)$$

We observe that the $\bar{\mathbf{q}}^{**}$ solution does not have any interpolation errors due to the LT scheme. Then the second order RK step is performed for the coarser scale:

$$\bar{\mathbf{q}}^{**} = \frac{3}{4} \bar{\mathbf{q}}^n + \frac{1}{4} \bar{\mathbf{q}}^* + \frac{1}{4} \Delta t^n f(t^n + \Delta t^n, \bar{\mathbf{q}}^*) \quad (41)$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4} \hat{\mathbf{q}}^n + \frac{1}{4} \hat{\mathbf{q}}^* + \frac{1}{4} \Delta t^n f(\hat{\mathbf{q}}^* - \epsilon_2) \quad (42)$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4} \hat{\mathbf{q}}^n + \frac{1}{4} \hat{\mathbf{q}}^* + \frac{1}{4} \Delta t^n \lambda \hat{\mathbf{q}}^* - \frac{1}{4} \Delta t^n \lambda \epsilon_2 \quad (43)$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4} \hat{\mathbf{q}}^n + \frac{1}{4} \hat{\mathbf{q}}^* + \frac{z}{4} \hat{\mathbf{q}}^* - \frac{\Delta t_\ell \lambda}{4} \epsilon_2 \quad (44)$$

$$\bar{\mathbf{q}}^{**} = \hat{\mathbf{q}}^{**} - \frac{\Delta t_\ell \lambda}{4} \epsilon_2 \quad (45)$$

Here, the approximation has an error of $-\frac{\Delta t_\ell \lambda}{4} \epsilon_2$.

Approximations after the second RK step

In the following, we compute the errors obtained from the predictions and projections after the second RK step, as performed in Section 4.1.2. The approximations here are the NERK solution at $t^n + \frac{1}{4} \Delta t_{\ell-1}$ to predict the solution $\bar{\mathbf{q}}^{**}_\ell$ on finer scales and the projections of the RK3 evolution of the finer leaves to approximate $\bar{\mathbf{q}}^{**}_{\ell-1}$. The first one is given by:

- ϵ_3 : Prediction error using NERK approximation

$$\epsilon_3 = \hat{\mathbf{q}}^{**}_\ell - P_{\ell-1 \rightarrow \ell} \left[\bar{\mathbf{q}}^n_{\ell-1, \theta=\frac{1}{4}} \right] = \left(\frac{\Delta t_\ell^2 \lambda^2}{8} - \frac{\Delta t_\ell^3 \lambda^3}{16} \right) \hat{\mathbf{q}}^n_\ell \quad (46)$$

Proof: Considering the NERK equation for $\theta = \frac{1}{4}$ given in Section 3 we get,

$$\epsilon_3 = \hat{\mathbf{q}}^{**}_\ell - P_{\ell-1 \rightarrow \ell} \left[\bar{\mathbf{q}}^n_{\ell-1} + \frac{7}{32} \Delta t_{\ell-1} f(\bar{\mathbf{q}}^n_{\ell-1}) + \frac{1}{32} \Delta t_{\ell-1} f(\bar{\mathbf{q}}^*_{\ell-1}) \right] \quad (47)$$

Then, we consider the error ϵ_2 in the flux computations for the coarser scale:

$$\epsilon_3 = \hat{\mathbf{q}}_\ell^n \left(1 + \frac{\Delta t_\ell \lambda}{2} + \frac{\Delta t_\ell^2 \lambda^2}{4} \right) - P_{\ell-1 \rightarrow \ell} \left[\bar{\mathbf{q}}_{\ell-1}^n + \frac{7}{32} \Delta t_{\ell-1} \lambda \bar{\mathbf{q}}_{\ell-1}^n + \frac{1}{32} \Delta t_{\ell-1} \lambda (\hat{\mathbf{q}}_{\ell-1}^* - \epsilon_2) \right] \quad (48)$$

$$\epsilon_3 = \hat{\mathbf{q}}_\ell^n \left(\frac{\Delta t_\ell \lambda}{16} + \frac{\Delta t_\ell^2 \lambda^2}{4} \right) - \frac{\Delta t_\ell \lambda}{16} P_{\ell-1 \rightarrow \ell} [\hat{\mathbf{q}}_{\ell-1}^* - \epsilon_2] \quad (49)$$

$$\epsilon_3 = \hat{\mathbf{q}}_\ell^n \left(\frac{\Delta t_\ell \lambda}{16} + \frac{\Delta t_\ell^2 \lambda^2}{4} \right) - \frac{\Delta t_\ell \lambda}{16} P_{\ell-1 \rightarrow \ell} \left[\hat{\mathbf{q}}_{\ell-1}^n + \Delta t_{\ell-1} f(\hat{\mathbf{q}}_{\ell-1}^n) + \frac{\Delta t_{\ell-1}^2 \lambda^2}{4} \hat{\mathbf{q}}_{\ell-1}^n \right] \quad (50)$$

$$\epsilon_3 = \frac{\Delta t_\ell^2 \lambda^2}{4} \hat{\mathbf{q}}_\ell^n - \frac{\Delta t_\ell^2 \lambda^2}{8} \hat{\mathbf{q}}_\ell^n - \frac{\Delta t_\ell \lambda}{16} P_{\ell-1 \rightarrow \ell} \left[\frac{\Delta t_{\ell-1}^2 \lambda^2}{4} \hat{\mathbf{q}}_{\ell-1}^n \right] \quad (51)$$

$$\epsilon_3 = \left(\frac{\Delta t_\ell^2 \lambda^2}{8} - \frac{\Delta t_\ell^3 \lambda^3}{16} \right) \hat{\mathbf{q}}_\ell^n \quad (52)$$

Afterwards we perform the third RK step for the finer leaf.

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3} \bar{\mathbf{q}}^n + \frac{2}{3} \bar{\mathbf{q}}^{**} + \frac{2}{3} \Delta t^n f \left(t^n + \frac{1}{2} \Delta t^n, \bar{\mathbf{q}}^{**} \right) \quad (53)$$

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3} \hat{\mathbf{q}}^n + \frac{2}{3} \hat{\mathbf{q}}^{**} + \frac{2z}{3} (\hat{\mathbf{q}}^{**} - \epsilon_3) \quad (54)$$

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3} \hat{\mathbf{q}}^n + \frac{2}{3} \left(1 + \frac{\Delta t_\ell \lambda}{2} + \frac{\Delta t_\ell^2 \lambda^2}{4} \right) \hat{\mathbf{q}}^n + \frac{2 \Delta t_\ell \lambda}{3} \left[\left(1 + \frac{\Delta t_\ell \lambda}{2} + \frac{\Delta t_\ell^2 \lambda^2}{4} \right) \hat{\mathbf{q}}^n - \epsilon_3 \right] \quad (55)$$

$$\bar{\mathbf{q}}^{n+1} = \left(1 + \Delta t_\ell \lambda + \frac{\Delta t_\ell^2 \lambda^2}{2} + \frac{\Delta t_\ell^3 \lambda^3}{6} \right) \hat{\mathbf{q}}^n - \frac{2 \Delta t_\ell \lambda}{3} \left(\frac{\Delta t_\ell^2 \lambda^2}{8} - \frac{\Delta t_\ell^3 \lambda^3}{16} \right) \hat{\mathbf{q}}^n \quad (56)$$

$$\bar{\mathbf{q}}^{n+1} = \left(1 + \Delta t_\ell \lambda + \frac{\Delta t_\ell^2 \lambda^2}{2} + \frac{\Delta t_\ell^3 \lambda^3}{12} + \frac{\Delta t_\ell^4 \lambda^4}{24} \right) \hat{\mathbf{q}}^n \quad (57)$$

This shows that the third order accuracy is lost in the evolution of the finer leaf.

- ϵ_4 : Projection error using finer leaf evolution.

$$\epsilon_4 = \hat{\mathbf{q}}_\ell^{**} - P_{\ell+1 \rightarrow \ell} [\bar{\mathbf{q}}_{\ell+1}^{n+1}] = \left(\frac{\Delta t_\ell^2 \lambda^2}{8} - \frac{\Delta t_\ell^3 \lambda^3}{96} - \frac{\Delta t_\ell^4 \lambda^4}{384} \right) \hat{\mathbf{q}}_\ell^n \quad (58)$$

Proof: Using Equation 57 to represent the time evolution for the finer leaf, we have,

$$\epsilon_4 = \hat{\mathbf{q}}_\ell^{**} - P_{\ell+1 \rightarrow \ell} \left[\left(1 + \Delta t_{\ell+1} \lambda + \frac{(\Delta t_{\ell+1} \lambda)^2}{2} + \frac{(\Delta t_{\ell+1} \lambda)^3}{12} + \frac{(\Delta t_{\ell+1} \lambda)^4}{24} \right) \hat{\mathbf{q}}_{\ell+1}^n \right] \quad (59)$$

$$\epsilon_4 = \hat{\mathbf{q}}_\ell^{**} - P_{\ell+1 \rightarrow \ell} \left[\left(1 + \frac{\Delta t_\ell \lambda}{2} + \frac{(\Delta t_\ell \lambda)^2}{8} + \frac{(\Delta t_\ell \lambda)^3}{96} + \frac{(\Delta t_\ell \lambda)^4}{384} \right) \hat{\mathbf{q}}_{\ell+1}^n \right] \quad (60)$$

$$\epsilon_4 = \left(1 + \frac{\Delta t_\ell \lambda}{2} + \frac{\Delta t_\ell^2 \lambda^2}{4} \right) \hat{\mathbf{q}}_\ell^n - \left(1 + \frac{\Delta t_\ell \lambda}{2} + \frac{\Delta t_\ell^2 \lambda^2}{8} + \frac{\Delta t_\ell^3 \lambda^3}{96} + \frac{\Delta t_\ell^4 \lambda^4}{384} \right) \hat{\mathbf{q}}_\ell^n \quad (61)$$

$$\epsilon_4 = \left(\frac{\Delta t_\ell^2 \lambda^2}{8} - \frac{\Delta t_\ell^3 \lambda^3}{96} - \frac{\Delta t_\ell^4 \lambda^4}{384} \right) \hat{\mathbf{q}}_\ell^n \quad (62)$$

And finally, performing the third RK step for the coarser leaf, we get

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\bar{\mathbf{q}}^n + \frac{2}{3}\bar{\mathbf{q}}^{**} + \frac{2}{3}\Delta t_\ell f \left[t^n + \frac{1}{2}\Delta t_\ell, \bar{\mathbf{q}}^{**} \right] \quad (63)$$

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\hat{\mathbf{q}}^n + \frac{2}{3} \left(\hat{\mathbf{q}}^{**} - \frac{\Delta t_\ell \lambda}{4} \epsilon_2 \right) + \frac{2\Delta t_\ell \lambda}{3} [\hat{\mathbf{q}}^{**} - \epsilon_4] \quad (64)$$

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\hat{\mathbf{q}}^n + \frac{2}{3} \left(\hat{\mathbf{q}}^{**} + \frac{\Delta t_\ell^3 \lambda^3}{16} \hat{\mathbf{q}}_\ell^n \right) + \frac{2\Delta t_\ell \lambda}{3} \left[\hat{\mathbf{q}}^{**} - \left(\frac{\Delta t_\ell^2 \lambda^2}{8} - \frac{\Delta t_\ell^3 \lambda^3}{96} - \frac{\Delta t_\ell^4 \lambda^4}{384} \right) \hat{\mathbf{q}}_\ell^n \right] \quad (65)$$

$$\bar{\mathbf{q}}^{n+1} = \hat{\mathbf{q}}^{n+1} + \frac{\Delta t_\ell^3 \lambda^3}{24} \hat{\mathbf{q}}_\ell^n + \left(-\frac{\Delta t_\ell^3 \lambda^3}{12} + \frac{\Delta t_\ell^4 \lambda^4}{144} + \frac{\Delta t_\ell^5 \lambda^5}{576} \right) \hat{\mathbf{q}}_\ell^n \quad (66)$$

$$\bar{\mathbf{q}}^{n+1} = \left(1 + \Delta t_\ell \lambda + \frac{\Delta t_\ell^2 \lambda^2}{2} + \frac{\Delta t_\ell^3 \lambda^3}{8} + \frac{\Delta t_\ell^4 \lambda^4}{144} + \frac{\Delta t_\ell^5 \lambda^5}{576} \right) \hat{\mathbf{q}}^n. \quad (67)$$

This result also shows a loss of the third order accuracy, since the third order term of the Taylor series $\frac{\Delta t_\ell^3 \lambda^3}{6}$ is not found.

4.5. Discussion on the numerical convergence

To study the numerical convergence of local time stepping we consider the advection equation

$$\frac{\partial Q}{\partial t} + \frac{\partial Q}{\partial x} = 0 \quad x \in [0, 1] \quad (68)$$

with periodic boundary conditions and a Gaussian initial condition, given by $Q(x) = \exp(-100(x - 0.25)^2)$. The simulation is performed for one time cycle. In order to avoid errors due to the MR scheme, we performed the adaptive simulations using two fixed grids in the domain. The first one in the interval $[0, 0.5]$ with $\Delta x = 1/512$, and the second in the interval $[0.5, 1.0]$ with $\Delta x = 1/256$, are both fixed during the entire simulation. We also use a centered numerical flux.

The convergence order is computed using a self convergence method, obtaining the rate which the MR and MRLT methods solution converges to a solution as $\Delta t \rightarrow 0$. For that, we perform simulations using subsequently smaller time steps, each one having half of the time step used in the previous simulation. The convergence rate is obtained from the following ratio:

$$\left\| \frac{\bar{\mathbf{q}}_{\Delta t} - \bar{\mathbf{q}}_{\frac{\Delta t}{2}}}{\bar{\mathbf{q}}_{\frac{\Delta t}{2}} - \bar{\mathbf{q}}_{\frac{\Delta t}{4}}} \right\| = \frac{C\Delta t^p - C\left(\frac{\Delta t}{2}\right)^p + O(\Delta t^{p+1})}{C\left(\frac{\Delta t}{2}\right)^p - C\left(\frac{\Delta t}{4}\right)^p + O(\Delta t^{p+1})} = \frac{1 - 2^{-p} + O(\Delta t)}{2^{-p} - 2^{-2p} + O(\Delta t)} = 2^p + O(\Delta t) \quad (69)$$

where $\bar{\mathbf{q}}_{\Delta t}$ is the solution of a simulation using a time step Δt and p is the order of the method, which is approximated using the logarithm:

$$p \approx \log_2 \left\| \frac{\bar{\mathbf{q}}_{\Delta t} - \bar{\mathbf{q}}_{\frac{\Delta t}{2}}}{\bar{\mathbf{q}}_{\frac{\Delta t}{2}} - \bar{\mathbf{q}}_{\frac{\Delta t}{4}}} \right\|. \quad (70)$$

The convergence order obtained for the FV, MR and MRLT methods are given in Table 1. In order to check that $p \rightarrow 0$ as $\Delta t \rightarrow 0$, we perform this test using two different values for the coarsest Δt . We observe that FV/RK2 and FV/RK3 yield second and third order time discretisations, respectively, as expected. For MRLT/NERK2 and MRLT/NERK3 we obtained the expected second order, in particular for the MRLT/NERK3 the second order is justified by the approximation errors ϵ_3 and ϵ_4 which caused the loss of the third order, as shown in the Equations 57 and 67. Another reason for this loss in accuracy was the fact that due to the order barrier for NERK methods [22], a third order solution at instant $t^n + \frac{\Delta t_\ell}{2}$ could not be produced by a 3 stage method. Thus, when a leaf performs its second time evolution inside

the LT cycle, it may use the second order solution from a coarser leaf, causing the loss from third to second order. This issue also justifies the observed loss in accuracy for the MRLT/RK2 method. Here, the coarser leaf produces a first order solution at instant $t^n + \frac{\Delta t_\ell}{2}$ to be used in the second evolution of the finer leaf.

Δt ($\times 10^{-4}$)	Method						
	FV/RK2	MR/RK2	MRLT/RK2	MRLT/NERK2	FV/RK3	MR/RK3	MRLT/NERK3
1.6	1.9991	1.9984	1.0794	2.0154	3.0000	3.0030	1.7895
0.8	2.0003	2.0010	0.9433	2.0002	3.0077	3.0017	1.9152

Table 1: Convergence order for the FV, MR and MRLT methods.

5. Numerical experiments

In this section we present some comparative results of the proposed MRLT/NERK method with the MR, MRLT methods given in [6] and also the traditional FV method on a uniform grid. These methods are applied to solve the two-dimensional Burgers equation, one and three-dimensional reaction-diffusion equations and finally the two-dimensional compressible Euler equations. We use the AUSM+ scheme [17] to compute the numerical flux in Burgers and Euler equations. To compute the advective term for the reaction-diffusion equation, we use the McCormack scheme [27]. The errors are computed in the discrete L_1 norm on the fine grid as:

$$e_{L_1}^{\text{method}} = \frac{1}{2^{Ld}} \sum_{i=0}^{2^{Ld}-1} \|\bar{\mathbf{q}}_i^{\text{ref}} - \bar{\mathbf{q}}_i^{\text{method}}\| \quad (71)$$

where $\bar{\mathbf{q}}^{\text{ref}}$ is the FV/RK3 reference solution of the corresponding problem and $\bar{\mathbf{q}}^{\text{method}}$ is the solution obtained with the analyzed method.

To compare the performance of two adaptive methodologies in terms of CPU time reduction versus accuracy loss, a cost value μ^{method} is defined for each adaptive method as:

$$\mu^{\text{method}} = \frac{e_{L_1}^{\text{method}} \cdot t_{\text{CPU}}^{\text{method}}}{t_{\text{CPU}}^{\text{FV}}}, \quad (72)$$

where $t_{\text{CPU}}^{\text{method}}$ is the CPU time obtained for the analyzed method and $t_{\text{CPU}}^{\text{FV}}$ is the CPU time of the FV method with the same number of scales L and Runge-Kutta of the same order.

The ratio between the cost of different adaptive methods yields the parameter λ , used to measure the advantage of one method compared with the other. In this work, the parameter λ is used to compare the proposed MRLT/NERK methods with the MR and MRLT methods, defined as:

$$\lambda_{\text{MRLT/NERK}}^{\text{method}} = \frac{\mu^{\text{method}}}{\mu_{\text{MRLT/NERK}}} \quad (73)$$

If the parameter λ is larger than 1, the MRLT/NERK approach is considered to be advantageous over the other method. In case of $\lambda < 1$, the MRLT/NERK approach is considered to be disadvantageous over the other method, and in case of $\lambda = 1$, the methods are considered to be equivalent.

All the simulations are performed using a fixed threshold value ϵ in order to simplify the experiments. In [7] computations are presented using a MR methodology with ϵ values which depend of the refinement level.

5.1. Two-dimensional Burgers equation

The Burgers equation is a non-linear PDE which represents a simple model for turbulence and is used in astrophysical applications. The inviscid model, in the two-dimensional case, is given by the following equation:

$$\frac{\partial Q}{\partial t} + \frac{1}{2} \left(\frac{\partial(Q^2)}{\partial x} + \frac{\partial(Q^2)}{\partial y} \right) = 0 \quad (x, y) \in \Omega = [0, 1] \times [0, 1]. \quad (74)$$

The initial condition used in this work is $Q_0(x, y) = \sin(2\pi x) \sin(2\pi y)$, with Dirichlet boundary conditions given by $Q(x, 0) = Q(x, 1) = Q(0, y) = Q(1, y) = 0$.

All simulations are performed with a Courant number $\sigma = 0.5$ and a threshold $\epsilon = 0.01$ until the time instant $t_f = 0.9$. The reference solution for this case is obtained using refinement level $L = 12$.

Figure 7 shows the reference solution and the solution obtained with the MRLT/RK2 and MRLT/NERK2. The respective difference, in modulus, with respect to the reference solution and adaptive grids at the end of the computation are also shown. The method MRLT/RK2 case exhibits larger errors close to the shocks, especially in the peak of the structure and in its background. The other methods present solutions closer to the one obtained with the MRLT/NERK2 method.

The results are compared with the solution in an uniform grid at the same level using a L_1 norm, showing perturbation errors. These errors, CPU time and memory compression are summarized in Table 2. The CPU time and memory of the adaptive methods are given in percentage of the number of leaves used in the FV method with the same number of scales and the same Runge–Kutta scheme. For the two-dimensional Burgers equation, the proposed MRLT/NERK methods present a slight gain in precision and a significant gain in CPU time in relation to the other adaptive methodologies. However, the MRLT schemes with NERK time integration require more memory, which decreases when increasing the resolution, *i.e.* for increasing L .

The parameters λ obtained for the MRLT/NERK methods compared to the MR and MRLT methods are presented in Table 3. For most of the experiments, the proposed methods yield values of λ between 2 and 3. This shows that the NERK-based methods are significantly more efficient than the RK-based MR and MRLT methods.

Table 2: Two-dimensional Burgers equation: L_1 errors, CPU time and memory compression.

Finest scale level	Method	Error ($\times 10^{-2}$) Q	CPU Time (% FV)	Memory
$L = 8$	MR/RK2	1.5032	68.4	19.5
	MRLT/RK2	1.7701	67.8	19.5
	MRLT/NERK2	1.2958	40.3	24.2
	MR/RK3	1.5045	184.9	19.4
	MRLT/NERK3	1.2932	94.7	24.2
$L = 9$	MR/RK2	1.3615	34.7	8.9
	MRLT/RK2	1.6645	32.5	8.9
	MRLT/NERK2	1.1457	16.7	10.3
	MR/RK3	1.3614	95.4	8.9
	MRLT/NERK3	1.1423	42.7	10.3
$L = 10$	MR/RK2	1.2890	14.7	4.1
	MRLT/RK2	1.6016	13.5	4.1
	MRLT/NERK2	1.0740	6.7	4.4
	MR/RK3	1.2893	15.3	4.1
	MRLT/NERK3	1.0699	6.4	4.4

Note: All adaptive computations use $\epsilon = 10^{-2}$; the final time is $t_f = 0.9$. The computations have been carried out on an Intel CoreTM*i7* CPU 2.67GHz. FV/RK2 CPU time: 2.0 min ($L = 8$); 15.7 min ($L = 9$); 2.3 h ($L = 10$). FV/RK3 CPU Time: 1.2 min ($L = 8$); 8.0 min ($L = 9$); 3.3 h ($L = 10$).

Table 3: Two-dimensional Burgers equation:
Computational gain λ of the MRLT/NERK methods with respect to the MR and MRLT methods.

Finest scale level		MR/RK2	MRLT/RK2	MR/RK3
$L = 8$	MRLT/NERK2	1.96	2.29	-
	MRLT/NERK3	-	-	2.27
$L = 9$	MRLT/NERK2	2.46	2.82	-
	MRLT/NERK3	-	-	2.66
$L = 10$	MRLT/NERK2	2.63	3.00	-
	MRLT/NERK3	-	-	2.88

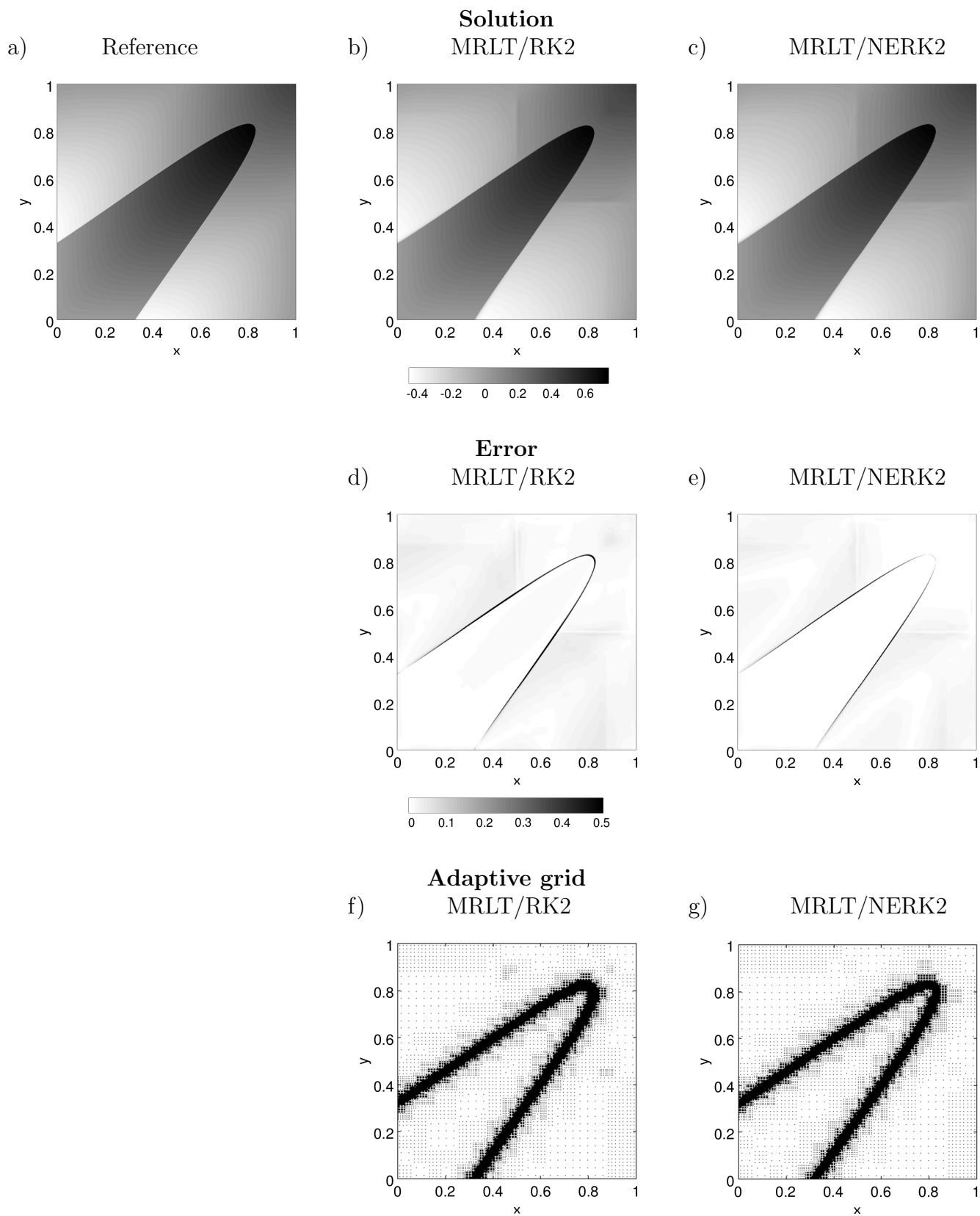


Figure 7: Reference solution for the two-dimensional Burgers equation a), the solutions obtained by the MRLT/RK2 b) and MRLT/NERK2 c) methods with its respective errors and the corresponding adaptive grids. For all cases we use $L = 10$ and the time instant is $t_f = 0.9$.

5.2. Reaction-diffusion equations

We consider reaction-diffusion equations in one and three space dimensions and study the formation of a flame front ignited with a spark in an environment with flammable premixed gas. This kind of problem is a prototype of non-linear parabolic equations with a non-linearity in the source term, see e.g. [26, 6].

5.2.1. One-dimensional case

In the one-dimensional case, considering equal mass and heat diffusion, this problem can be modeled by the following equation:

$$\frac{\partial T}{\partial t} + v_f \frac{\partial T}{\partial x} = \frac{\partial^2 T}{\partial x^2} + \omega(T) \quad \text{for } x \in (-15, 15) \quad (75)$$

where the function $T(x, t)$ is the dimensionless temperature normalized between 0 (unburned gas) and 1 (burned gas), $v_f = \int \omega dx$ is the flame velocity and $\omega(T)$ is the chemical reaction rate, given by:

$$\omega(T) = \frac{Ze^2}{2}(1 - T) \exp\left(\frac{Ze(1 - T)}{\tau(1 - T) - 1}\right) \quad (76)$$

where Ze is a dimensionless activation energy, known as Zeldovich number, and τ is the burnt-unburnt temperature ratio. In this one-dimensional case, the unburned gas concentration Y is defined as $Y = 1 - T$.

In the numerical experiments, the following initial condition is used:

$$T(x, 0) = \begin{cases} 1, & \text{if } x \leq 1 \\ \exp(1 - x), & \text{otherwise} \end{cases} \quad (77)$$

with boundary conditions given by:

$$\frac{\partial T}{\partial x}(-15, t) = 0, \quad T(15, t) = 0 \quad (78)$$

The subsequent simulations are performed with the parameters $Ze = 10$, $\tau = 0.8$ and a Courant number $\sigma = 0.5$ until the final time instant $t_f = 5.0$. The adaptive simulations are performed with a threshold $\epsilon = 0.01$. The reference solution for this case is obtained using the same Courant number and a refinement level $L = 13$. Figure 8 shows the reference solution and the solution obtained by the MRLT/RK2 and MRLT/NERK2 methods with its respective errors and the corresponding final grids. In this one-dimensional case, the adaptive grid is represented by the position of each cell (x -axis) and its refinement level (y -axis).

The adaptive methods present a larger error in the flame front region, especially for the variable ω . Among the adaptive methods, MRLT/RK2 has the smallest errors, while the other methods present very similar errors. However, the MRLT/NERK methods still requires the lowest CPU time. These times and L_1 errors are assembled in Table 4.

The computational gain of the MRLT/NERK methods compared to the MR and MRLT methods are given in Table 5. In this case, the MRLT/NERK methods yields the most expressive results compared to the MR methods in terms of computational gain. The gain of the MRLT/NERK2 method compared with the MRLT/RK2 method is small for the cases $L = 12$ and 13 . The corresponding results are presented in Table 5.

Table 4: One-dimensional reaction-diffusion equations: L_1 errors, CPU time and memory compression.

Finest scale level	Method	Error ($\times 10^{-4}$)			CPU Time (%FV)	Memory
		T	Y	ω		
$L = 11$	MR/RK2	5.045	5.045	24.564	31.9	3.1
	MRLT/RK2	5.666	5.666	30.192	18.0	3.1
	MRLT/NERK2	4.380	4.380	21.308	7.2	3.1
	MR/RK3	5.045	5.045	24.566	31.3	3.1
	MRLT/NERK3	4.543	4.543	22.173	5.3	3.1
$L = 12$	MR/RK2	5.043	5.043	24.559	15.9	1.5
	MRLT/RK2	1.780	1.780	14.256	8.0	1.5
	MRLT/NERK2	4.728	4.728	23.107	2.8	1.5
	MR/RK3	5.043	5.043	24.560	16.0	1.5
	MRLT/NERK3	4.755	4.755	23.261	2.3	1.5
$L = 13$	MR/RK2	5.054	5.054	24.609	7.9	0.7
	MRLT/RK2	2.904	2.904	15.062	3.9	0.7
	MRLT/NERK2	4.900	4.900	23.903	1.4	0.7
	MR/RK3	5.054	5.054	24.609	7.9	0.7
	MRLT/NERK3	4.904	4.904	23.923	1.1	0.7

Note: All adaptive computations use $\epsilon = 10^{-2}$; final time: $t_f = 5.0$. Computed on an Intel CoreTM_{i7} CPU 2.67GHz. FV/RK2 CPU time: 51.4 min ($L = 11$); 6.9 h ($L = 12$); 54.6 h ($L = 13$). FV/RK3 CPU Time: 51.4 min ($L = 11$); 6.9 h ($L = 12$); 54.6 h ($L = 13$).

Table 5: One-dimensional reaction-diffusion equations: Computational gain, for the variable T , of the proposed MRLT/NERK methods compared to the MR and MRLT methods.

Finest scale level		MR/RK2	MRLT/RK2	MR/RK3
$L = 11$	MRLT/NERK2	5.10	3.23	-
	MRLT/NERK3	-	-	6.55
$L = 12$	MRLT/NERK2	6.05	1.07	-
	MRLT/NERK3	-	-	7.37
$L = 13$	MRLT/NERK2	5.82	1.65	-
	MRLT/NERK3	-	-	7.40

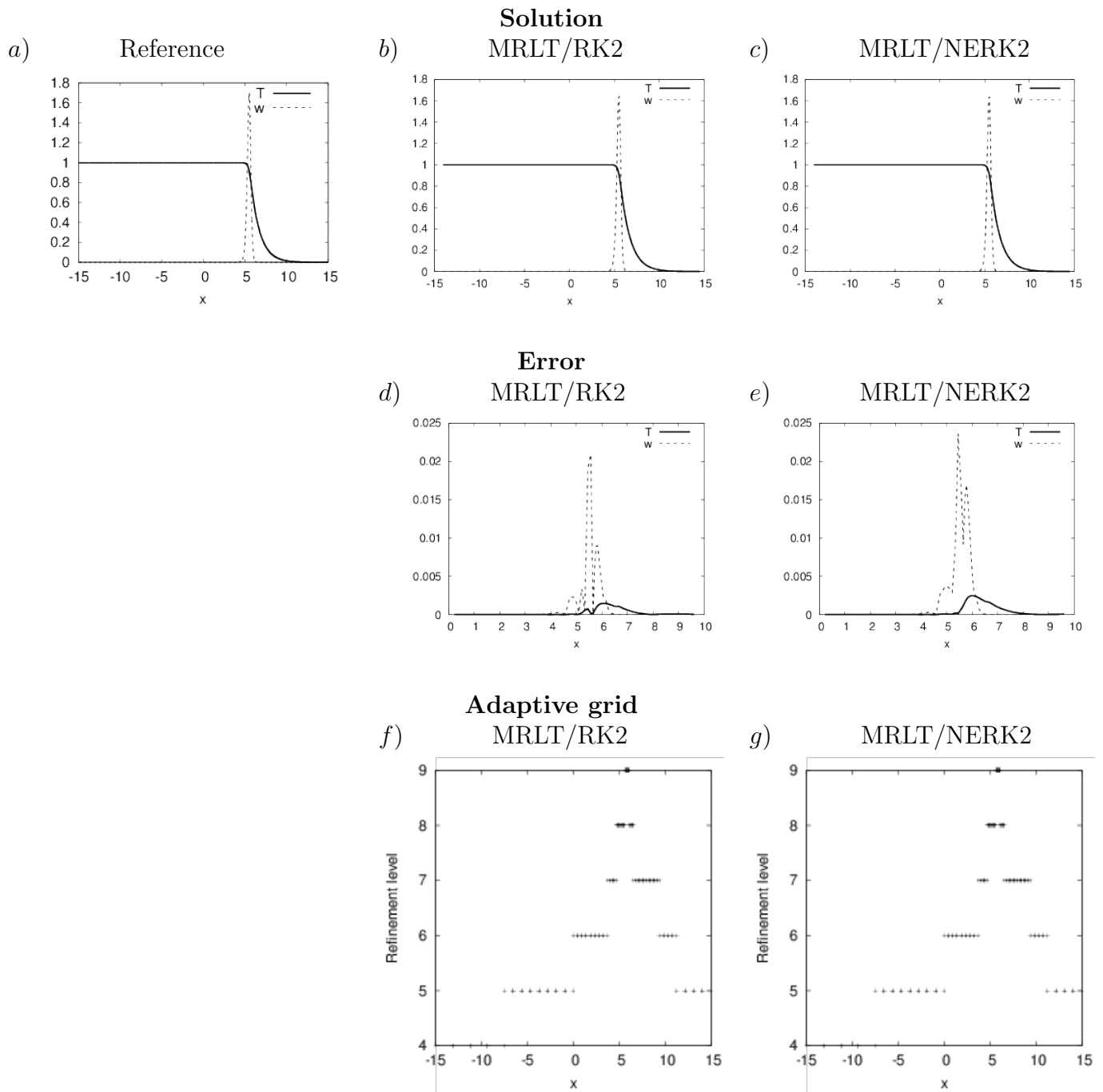


Figure 8: Reference solution of the one-dimensional reaction-diffusion equations a) and the solutions obtained by the MRLT/RK2 (b) and MRLT/NERK2 (c) methods with its respective errors (d,e) and final adaptive grids (f,g). For all cases we use $L = 13$.

5.2.2. Three-dimensional case

In the three-dimensional case, the reaction-diffusion equations read:

$$\frac{\partial T}{\partial t} = \nabla^2 T + \omega - s \quad (79a)$$

$$\frac{\partial Y}{\partial t} = \frac{1}{Le} \nabla^2 Y - \omega \quad (79b)$$

where Le denotes the Lewis number, which defines the ratio of mass and heat diffusion and with the chemical reaction rate ω :

$$\omega(T, Y) = \frac{Ze^2}{2Le} Y \exp\left(\frac{Ze(T-1)}{1+\tau(T-1)}\right) \quad (80)$$

According to the Stefan–Boltzmann law, the heat loss due to radiation s is modeled by:

$$s(T) = \kappa [(T + \tau^{-1} - 1)^4 - (\tau^{-1} - 1)^4] \quad (81)$$

where κ is a dimensionless radiation coefficient. In this work, we use $\kappa = 0.1$.

The initial condition, described by spherical coordinates, is:

$$T(r, 0) = \begin{cases} 1, & \text{if } r \leq r_0 \\ \exp\left(1 - \frac{r}{r_0}\right), & \text{otherwise} \end{cases} \quad (82)$$

$$Y(r, 0) = \begin{cases} 0, & \text{if } r \leq r_0 \\ 1 - e\left(Le\left(1 - \frac{r}{r_0}\right)\right), & \text{otherwise} \end{cases} \quad (83)$$

where $r_0 = 1$ is the initial radius of the ellipsoidal flame ball and $r = \sqrt{\frac{X^2}{a^2} + \frac{Y^2}{b^2} + \frac{Z^2}{c^2}}$ with:

$$X = x \cos(\theta) - y \sin(\theta) \quad (84)$$

$$Y = [x \sin(\theta) + y \cos(\theta)] \cos(\phi) - z \sin(\phi) \quad (85)$$

$$Z = [x \sin(\theta) + y \cos(\theta)] \sin(\phi) + z \cos(\phi) \quad (86)$$

The boundary conditions are of homogeneous Neumann type. In these simulations we use the parameters $Ze = 10$, $\tau = 0.64$, $Le = 0.3$, $\theta = \frac{\pi}{3}$, $\phi = \frac{\pi}{4}$, $a = \frac{3}{2}$, $b = \frac{3}{2}$, $c = 3$, a threshold factor $\epsilon = 0.01$ and a Courant number $\sigma = 0.1$ until the final time instant $t_f = 5.0$. The reference solution for this case is obtained using the same Courant number and the refinement level $L = 7$. Figure 9 shows the isosurface of T for the reference and the MRLT/NERK2 and MRLT/NERK3 methods. It also shows the solutions, difference from the reference solution in modulus, and projections of every cell center at the plane xz . The adaptive methodologies present higher errors close to the flame ball fronts, which are more rounded in comparison to the FV solutions. The adaptive grids are similar for all adaptive methodologies, with a higher concentration of refined cells in the region of the front and inside the flame ball.

The L_1 errors, CPU time and memory compression are summarized in Table 6. In this case, the proposed MRLT/NERK methods present a loss in precision and memory usage with a gain in CPU time in relation to the other adaptive methodologies. However, the parameters λ obtained for the MRLT/NERK methods compared to the MR and MRLT methods, presented in Table 7, still yield favorable values, especially for the third order methods. Thus we find that the MRLT/NERK methods are slightly more efficient than the MR and MRLT methods for the three-dimensional flame ball case, besides a small loss in precision.

Table 6: Three-dimensional reaction-diffusion equations: L_1 errors, CPU time and memory compression.

Finest scale level	Method	Error ($\times 10^{-4}$)			CPU Time (%FV)	Memory
		T	Y	ω		
$L = 7$	MR/RK2	4.516	7.338	21.831	15.6	2.5
	MRLT/RK2	4.511	7.267	21.798	14.9	2.5
	MRLT/NERK2	4.513	7.636	22.038	10.9	7.7
	MR/RK3	4.516	7.338	21.831	14.4	2.5
	MRLT/NERK3	4.526	7.369	21.856	5.5	2.5

Note: All adaptive computations use $\epsilon = 10^{-2}$; final time: $t_f = 5.0$. Computed on a Quad-Core AMD OpteronTM CPU 2.4GHz. FV/RK2 CPU time: 25.8h ($L = 7$). FV/RK3 CPU Time: 38.5h ($L = 7$).

Table 7: Three-dimensional reaction-diffusion equations: Computational gain, for the variable T , of the MRLT/NERK methods compared to the MR and MRLT methods.

Finest scale level		MR/RK2	MRLT/RK2	MR/RK3
$L = 7$	MRLT/NERK2	1.43	1.36	-
	MRLT/NERK3	-	-	2.61

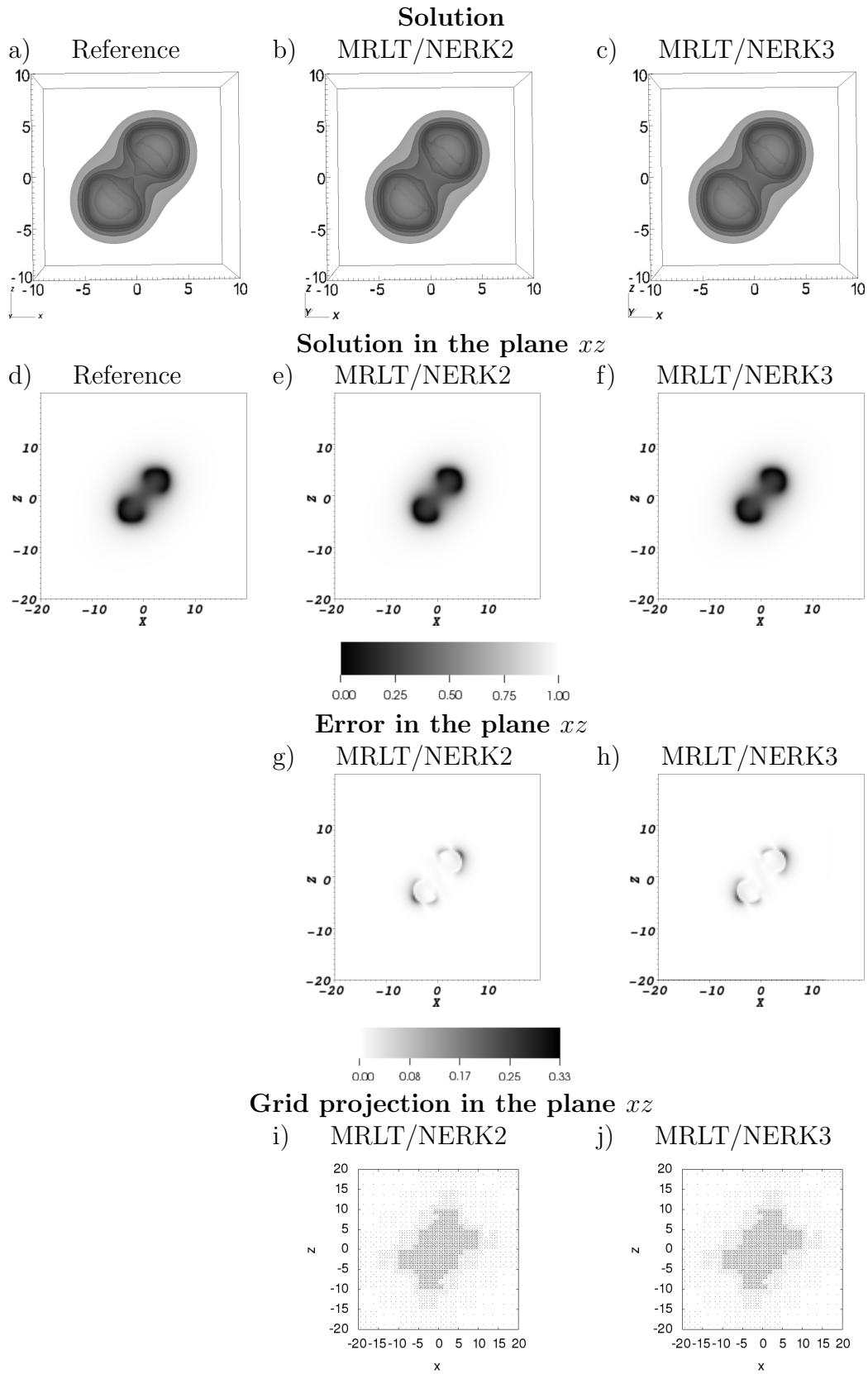


Figure 9: Isosurface and xz plane reference solution for the variable T in three-dimensions at $t_f = 5.0$. The solutions are obtained with the MRLT/NERK2 and MRLT/NERK3 methods with its respective errors in the xz plane and projections of the center of the adaptive grid cell onto the xz plane. The isosurface plot were build using 5 linearly scaled values.

5.3. Compressible two-dimensional Euler equations

The Euler equations, which describe the dynamics of a non-ionized gas, are given by the following relations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \quad (87a)$$

$$\frac{\partial \rho \vec{v}}{\partial t} + \nabla \cdot (\rho \vec{v}) \vec{v} + \nabla p = 0 \quad (87b)$$

$$\frac{\partial E}{\partial t} + \nabla \cdot \left(\vec{v} (E + p) \right) = 0 \quad (87c)$$

where $\vec{v} = (v_x, v_y)$ is the velocity, ρ is the fluid density, p is the pressure, and the energy per mass unity E is given by $E = \rho e + \frac{\rho \|\vec{v}\|^2}{2}$. This system is completed by the equation of state of an ideal gas $p = \frac{\rho T}{\gamma Ma^2}$, where γ is the specific heat ratio, T is the temperature and Ma is the Mach number.

The initial condition used in this numerical test is the classical Lax–Liu configuration #6 [16, 7]. In this initial condition configuration, the domain is divided into four quadrants, ordered from 1st to 4th, and defined by the subdomains $[0.5; 1] \times [0.5; 1]$, $[0; 0.5] \times [0.5; 1]$, $[0; 0.5] \times [0; 0.5]$ and $[0.5; 1] \times [0; 0.5]$ respectively. The initial values for each quadrant are given in Table 8. This problem is simulated until the final instant $t_f = 0.25$, using homogeneous Neumann boundary conditions. The numerical parameters are the Courant number $\sigma = 0.5$, $Ma = 1$, and $\gamma = 1.4$ inside the domain $[0; 1] \times [0; 1]$.

Table 8: Initial condition for two-dimensional Euler equations.

Variables	Quadrant			
	1 st	2 nd	3 rd	4 th
Density, ρ	1.00	2.00	1.00	3.00
Pressure, p	1.00	1.00	1.00	1.00
x -velocity, v_x	0.75	0.75	-0.75	-0.75
y -velocity, v_y	-0.50	0.50	0.50	-0.50

Figure 10 shows the reference solution and the solution obtained with the MRLT/RK2, MRLT/NERK2 methods and its respective difference, in modulus, from the reference. Furthermore, the corresponding adaptive grid using $L = 12$ with CFL $\sigma = 0.5$ is shown.

The L_1 errors, CPU time and memory compression are presented in Table 9. In this case, the proposed MRLT/NERK methods yield a slightly gain in precision with a significant gain in CPU time in relation to the other adaptive methodologies. The memory usage of all adaptive methodologies is quite similar. Due to the gain in both precision and CPU time, the parameters λ for the MRLT/NERK methods compared with the MR and MRLT methods, presented in Table 10 have expressive values, around 3 for most of the experiments. Thus, using this metric, the proposed methods for the Euler two-dimensional case are significantly more efficient than the MR and MRLT methods.

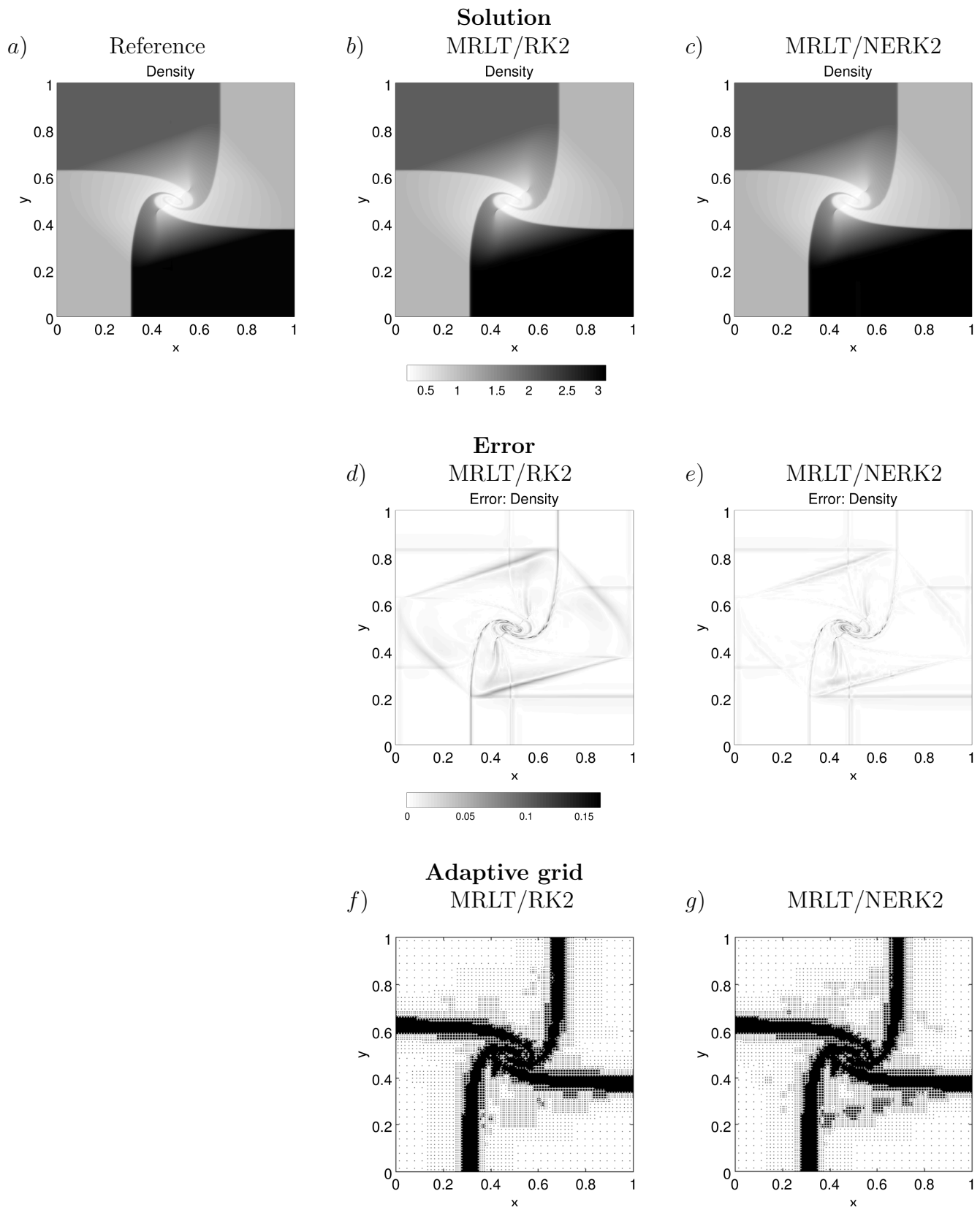


Figure 10: Reference solution for the two-dimensional Euler equations(a), the solutions obtained by the MRLT/RK2 (b) and MRLT/NERK2 methods, using $L = 10$ scales, with its respective errors(d, e), and corresponding adaptive grids (f, g) at final time $t_f = 0.25$.

Table 9: Two-dimensional Euler equations: L_1 errors, CPU time and memory compression.

Finest scale level	Method	Error ($\times 10^{-1}$)						CPU Time (%FV)	Memory
		ρ	p	T	E	v_x	v_y		
$L = 8$	MR/RK2	2.2095	0.6075	0.9581	2.0499	1.7521	0.7180	40.7	26.4
	MRLT/RK2	2.2085	0.6049	0.9572	2.0463	1.7510	0.7167	33.7	26.0
	MRLT/NERK2	2.2096	0.6077	0.9582	2.0503	1.7521	0.7183	12.9	26.6
	MR/RK3	2.2093	0.6075	0.9580	2.0498	1.7520	0.7180	46.3	26.4
	MRLT/NERK3	2.2095	0.6078	0.9581	2.0504	1.7522	0.7182	14.6	26.6
$L = 9$	MR/RK2	1.1365	0.3133	0.5217	1.0470	0.9012	0.3934	24.4	14.0
	MRLT/RK2	1.1369	0.3133	0.5219	1.0483	0.9012	0.3931	22.6	13.9
	MRLT/NERK2	1.1355	0.3128	0.5216	1.0458	0.9012	0.3931	7.4	14.2
	MR/RK3	1.1364	0.3133	0.5217	1.0469	0.9012	0.3934	27.3	14.0
	MRLT/NERK3	1.1355	0.3130	0.5216	1.0461	0.9013	0.3931	8.0	14.2
$L = 10$	MR/RK2	0.5835	0.1604	0.2765	0.5353	0.4611	0.2115	14.7	7.2
	MRLT/RK2	0.5864	0.1643	0.2782	0.5441	0.4622	0.2131	12.8	7.2
	MRLT/NERK2	0.5821	0.1596	0.2762	0.5335	0.4609	0.2110	4.1	7.3
	MR/RK3	0.5834	0.1604	0.2765	0.5352	0.4611	0.2115	11.2	7.2
	MRLT/NERK3	0.5821	0.1598	0.2762	0.5338	0.4610	0.2109	3.6	7.3

Note: All adaptive computations use $\epsilon = 10^{-2}$; final time: $t_f = 0.25$. Computed on an Intel CoreTM*i7* CPU 2.67GHz. FV/RK2 CPU time: 10.1 min ($L = 8$); 73.3 min ($L = 9$); 8.8 h ($L = 10$). FV/RK3 CPU Time: 11.7min ($L = 8$); 91.7 min ($L = 9$); 13.8 h ($L = 10$).

6. Conclusions

In this work, we introduced a new local time-stepping for adaptive multiresolution methods using NERK time integration schemes [30]. Interpolating values of the intermediate Runge–Kutta stages yield the required values at intermediate time steps, which are necessary for the time evolution. Hence the current limitation of local time stepping to second order schemes can be overcome, and the required synchronised solution can be obtained. The proposed new methodology has been implemented and validated for two and three stage NERK schemes. In principle the extension of MRLT/NERK schemes to even higher order is possible, but the computational cost would increase due to the order barrier discussed in [22]. For NERK methods of order larger or equal to three, Owren and Zennaro [23] have shown that the order of approximation is reduced concerning the underlying RK method. This order reduction implies that increasing the order of NERK schemes beyond three would become less efficient and thus further research is indeed necessary to obtain well performing local time-stepping schemes with order larger than three. With the presented numerical experiments we assessed the precision and efficiency of the proposed two and three stage MRLT/NERK approach for different classical nonlinear evolution equations, i.e., for Burgers, reaction-diffusion and the compressible Euler equations considering Cartesian geometries in one, two and three space dimensions. In all adaptive computations, we observed a significant gain in CPU time in comparison with uniform grid computations where the efficiency is increasing with the grid resolution. Nevertheless, the precision of the uniform grid computations is controlled in the MRLT/NERK schemes, and the order of convergence is maintained. Regarding memory consumption, we observed for the MRLT/NERK schemes no necessary increase, compared to MR and classical MRLT methods.

Table 10: Two-dimensional Euler equations:
 Computational gain, for the variable ρ , of the proposed MRLT/NERK methods compared with the MR and MRLT methods.

Finest scale level		MR/RK2	MRLT/RK2	MR/RK3
$L = 8$	MRLT/NERK2	3.15	2.61	-
	MRLT/NERK3	-	-	3.17
$L = 9$	MRLT/NERK2	3.30	3.05	-
	MRLT/NERK3	-	-	3.41
$L = 10$	MRLT/NERK2	3.59	3.14	-
	MRLT/NERK3	-	-	3.11

The precision of the MRLT/NERK computations is very reasonable, and in all cases, we found errors about the same order of magnitude as for the MRLT computations using classical RK schemes.

In conclusion, we showed that the MRLT/NERK methods are advantageous compared to the MR and MRLT approaches in all studied cases, obtaining even significant performance gains in some examples. For the two-dimensional Euler equations, for instance, the MRLT/NERK simulations only required one-third of the CPU time necessary for the MR and MRLT computations. Most of these gains are due to the significant reduction in CPU time obtained in the MRLT/NERK methods.

Acknowledgements

The authors are indebted to Prof. Claus-Dieter Munz who motivated the use of NERK methods for local time-stepping. We thank Dr. Olivier Roussel for developing the original Carmen Code and fruitful scientific discussions during the years. ML thankfully acknowledges financial support from CAPES and CNPq grant 140626/2014 – 0, Brazil. MD thankfully acknowledges financial support from Ecole Centrale Marseille, CNPq grant 306038/2015 – 3 and FAPESP grant 2015/25624 – 2. OM thankfully acknowledges financial support from CNPq grant 307083/2017 – 9 and FINEP grant 01.120527.00, Brazil. KS acknowledges financial support from the ANR Grant 15–CE40 – 0019 (AIFIT), France.

7. References

- [1] W. Auzinger and O. Koch. An improved local error estimator for symmetric time-stepping schemes. *Applied Mathematics Letters*, 82:106–110, 2018.
- [2] W. Boscheri, M. Dumbser, and O. Zanotti. High order cell-centered lagrangian-type finite volume schemes with time-accurate local time stepping on unstructured triangular meshes. *Journal of Computational Physics*, 291:120–150, 2015.
- [3] F. Coquel, L. Q. Nguyen, M. Postel, and Q. H. Tran. Local time stepping applied to implicit-explicit methods for hyperbolic systems. *SIAM Multiscale Modeling and Simulation*, 8(2):540–570, 2010.
- [4] J. Díaz and M. J. Grote. Conserving explicit local time-stepping for second-order wave equations. *SIAM Journal on Scientific Computing*, 31(3):1985–2014, 2009.
- [5] J. Díaz and M. J. Grote. Multi-level explicit local time-stepping methods for second-order wave equations. *Computer Methods in Applied Mechanics and Engineering*, 291:240–265, 2015.
- [6] M. O. Domingues, S. M. Gomes, O. Roussel, and K. Schneider. An adaptive multiresolution scheme with local time stepping for evolutionary PDEs. *Journal of Computational Physics*, 227(8):3758–3780, 2008.
- [7] M. O. Domingues, S. M. Gomes, O. Roussel, and K. Schneider. Space-time adaptive multiresolution techniques for compressible Euler Equations. In C. A. de Moura and C. S. Kubrusly, editors, *The Courant-Friedrichs-Lewy (CFL) Condition: 80 Years After Its Discovery.*, pages 101–117. Birkhäuser, 2012.
- [8] M. Dumbser, O. Zanotti, A. Hidalgo, and D. S. Balsara. ADER-WENO finite volume schemes with space-time adaptive mesh refinement. *Journal of Computational Physics*, 248:257–286, 2013.
- [9] G. Gassner, M. Dumbser, F. Hindenlang, and C.-D. Munz. Explicit one-step time discretizations for discontinuous Galerkin and finite volume schemes based on local predictors. *Journal of Computational Physics*, 230(11):4232–4247, 2011. Special issue High Order Methods for CFD Problems.
- [10] G. J. Gassner, F. Hindenlang, and C.-D. Munz. A Runge–Kutta based discontinuous Galerkin method with time accurate local time stepping. *Advances in Computational Fluid Dynamics*, 2:95–118, 2011.
- [11] N. Y. Gnedin, V. A. Semenov, and A. V. Kravtsov. Enforcing the Courant-Friedrichs-Lewy condition in explicitly conservative local time stepping schemes. *Journal of Computational Physics*, 359:93–105, 2018.
- [12] A. Harten. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. *Communications on Pure and Applied Mathematics*, 48:1305–1342, 1995.
- [13] B. Hejzialhosseini, D. Rossinelli, M. Bergdorf, and P. Koumoutsakos. High order finite volume methods on wavelet-adapted grids with local time-stepping on multicore architectures for the simulation of shock-bubble interactions. *Journal of Computational Physics*, 229(22):8364–8383, 2010.
- [14] N. Hovhannisyan and S. Müller. On the stability of fully adaptive multiscale schemes for conservation laws using approximate flux and source reconstruction strategies. *IMA Journal of Numerical Analysis*, 30(4):1256–1295, 2010.

- [15] L. Krivodonova. An efficient local time-stepping scheme for solution of nonlinear conservation laws. *Journal of Computational Physics*, 229:8537–8551, 2010.
- [16] P. D. Lax and X-D. Liu. Solution of two-dimensional Riemann problems of gas dynamics by positive schemes. *SIAM Journal on Scientific Computing*, 19(2):319–340, 1998.
- [17] M.-S. Liou. A sequel to ausm: Ausm+. *Journal of Computational Physics*, 129:364–382, 1996.
- [18] M. Mayr, W. A. Wall, and M. W. Gee. Adaptive time stepping for fluid-structure interaction solvers. *Finite Elements in Analysis and Design*, 141:55–69, 2018.
- [19] S. Müller. *Adaptive multiscale schemes for conservation laws*, volume 27 of *Lectures Notes in Computational Science and Engineering*. Springer, Heidelberg, 2003.
- [20] S. Müller and Y. Stiriba. Fully adaptive multiscale schemes for conservation laws employing locally varying time stepping. *Journal of Scientific Computing*, 30(3):493–531, 2007.
- [21] S. Osher and R. Sanders. Numerical approximations to nonlinear conservation laws with locally varying time and space grids. *Mathematics of Computation*, 41:321–336, 1983.
- [22] B. Owren and M. Zennaro. Order barriers for continuous explicit Runge-Kutta methods. *Mathematics of Computation*, 56:645–661, 1991.
- [23] B. Owren and M. Zennaro. Derivation of efficient, continuous, explicit Runge-Kutta methods. *SIAM Journal on Scientific and Statistical Computing*, 13(6):1488–1501, 1992.
- [24] H. Qi, X. Wang, J. Zhang, and J. Wang. An ADER discontinuous Galerkin method with local time-stepping for transient electromagnetics. *Computer Physics Communications*, 229:106–115, 2018.
- [25] M. Rietmann, M. Grote, D. Peter, and O. Schenk. Newmark local time stepping on high-performance computing architectures. *Journal of Computational Physics*, 334:308–326, 2017.
- [26] O. Roussel, K. Schneider, A. Tsigulin, and H. Bockhorn. A conservative fully adaptive multiresolution algorithm for parabolic PDEs. *Journal of Computational Physics*, 188:493–523, 2003.
- [27] B. Van Leer. Towards the ultimate conservative difference scheme iii. upstream-centered finite-difference schemes for ideal compressible flow. *Journal of Computational Physics*, 23(3):263–275, 1977.
- [28] R. Vermiglio and M. Zennaro. Multistep natural continuous extensions of Runge-Kutta methods: the potential for stable interpolation. *Applied Numerical Mathematics*, 12(6):521–546, 1993.
- [29] A. R. Winters and D. A. Kopriva. High-order local time stepping on moving DG spectral element meshes. *Journal of Scientific Computing*, 58(1):176–202, 2014.
- [30] M. Zennaro. Natural continuous extensions of Runge-Kutta methods. *Mathematics of Computation*, 46(173):119–133, 1986.

Algorithm 1 Construction of the adaptive grid.

- Do Algorithm 2 {Compute the leaves that will belong to the adaptive grid}
- Refine each leaf with a finer neighbour leaf.
 - Set these new leaves as virtual leaves;
-

Algorithm 2 Grid adaptation.

Require: Finest scale level L of the solution.

Require: Solution at level L .

Require: Threshold value ϵ .

{Selection of the nodes in the adaptive grid}

for $\ell = L-1 \rightarrow 0$, $\ell = \ell-1$ **do**

- Project the solution from the grid $\Omega^{\ell+1}$ to Ω^ℓ ;
- Predict the solution of the grid $\Omega^{\ell+1}$ based on the project solution in Ω^ℓ ;
- Compare the original solution of the grid $\Omega^{\ell+1}$ with the predicted one, and then obtain the wavelet coefficients $\bar{D}^{\ell+1}$, as the difference with these solutions;

{Elimination of the unnecessary nodes and the imposition of a graded tree}

for every leaf $\in \Omega^{\ell+1}$ **do**

if $|\mathbf{d}^{\ell+1}| \leq \epsilon$ and adjacent leaves are inside $\Omega^{\ell+1}$ or Ω^ℓ **then**

- Remove the leaf from the grid $\Omega^{\ell+1}$;

end if

end for

end for

Algorithm 3 Single iteration of the LT scheme.

Require: Coarsest scale ℓ_{\min} to be evolved in this time evolution;

Require: Current iteration number n ;

Compute $\ell_{\min} = \min_{\ell} [\text{mod} (n, 2^{L-\ell}) = 0]$;

for $\ell = L \rightarrow \ell_{\min}, \ell = \ell - 1$ **do**

for Every internal node $\in \Omega^{\ell}$ **do**

 Obtain the solution $\bar{\mathbf{q}}_{\ell}^n$ by projecting the solution from its child nodes via simple averaging;

end for

end for

if ℓ_{\min} is not the coarsest scale of the grid **then**

for Every internal node $\in \Omega^{\ell_{\min}-1}$ **do**

 Obtain the NERK solution with $\theta = \frac{1}{2}$ by projecting $\bar{\mathbf{q}}_{\ell_{\min}}^n$.

 Extrapolate this solution to the instant $t^{n+2^{L-\ell_{\min}}}$ (Section 4.3).

end for

for Every virtual leaf $\in \Omega^{\ell_{\min}}$ **do**

 Predict $\bar{\mathbf{q}}_{\ell}^n$ using the NERK (RK2) or $\bar{\mathbf{q}}_{\ell_{\min}-1}^{**}$ (RK3) solutions of the cells $\in \Omega^{\ell_{\min}-1}$ at instant t^n .

end for

end if

for $\ell = \ell_{\min} + 1 \rightarrow L, \ell = \ell + 1$ **do**

for Every virtual leaf $\in \Omega^{\ell}$ **do**

 Predict $\bar{\mathbf{q}}_{\ell}^n$ using the values of the cells $\in \Omega^{\ell-1}$ at time instant t^n .

end for

end for

Remeshing process of cells with refinement level greater or equal than ℓ_{\min} ;

for $\ell = L \rightarrow \ell_{\min}, \ell = \ell - 1$ **do**

for Every leaf $\in \Omega^{\ell}$ **do**

 Perform flux computations at instant t^n ;

 First RK step;

 First order interpolation of the RK evolution at instant $t^n + \frac{1}{2}\Delta t_{\ell}$;

if we are computing RK2 time evolution **then**

 Compute $\bar{\mathbf{q}}_{\theta=\frac{1}{2}}^*$;

else if we are computing RK3 time evolution **then**

 Compute $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}^*$;

 Compute $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}^*$;

end if

end for

end for

Tree refreshing before the second RK step (Algorithm 8):

Second RK step of the time evolution (Algorithm 5);

if Performing RK3 time evolution **then**

 Tree refreshing before the third RK step (Section 4.2):

 Third RK step of the time evolution (Algorithm 7);

end if

Algorithm 4 Projection procedure inside the second RK step.

Require: Scale ℓ to receive the projection.

Require: Number of dimensions d of the problem.

for Every internal node $\in \Omega^\ell$ **do**

$\bar{\mathbf{q}}_\ell^* = 0$; {result after RK1, set here to zero to store the projection}

for Every child cell $\in \Omega^{\ell+1}$ **do**

if The cell is a leaf **then**

$\bar{\mathbf{q}}_\ell^* \leftarrow \bar{\mathbf{q}}_\ell^* + \bar{\mathbf{q}}_{\ell+1}(t^n + 2\Delta t_{\ell+1})$; {Add the already extrapolated value $\bar{\mathbf{q}}_{\ell+1}$ at $t^n + 2\Delta t_{\ell+1}$ }

else if The cell is an internal node **then**

$\bar{\mathbf{q}}_\ell^* \leftarrow \bar{\mathbf{q}}_\ell^* + 2\bar{\mathbf{q}}_{\ell+1}^* - \bar{\mathbf{q}}_{\ell+1}^n$; {Add a linear extrapolation of the values $\bar{\mathbf{q}}_{\ell+1}$ at $t^n + 2\Delta t_{\ell+1}$ }

end if

end for

$\bar{\mathbf{q}}_\ell^* \leftarrow \frac{1}{2^d} \bar{\mathbf{q}}_\ell^*$;

end for

Algorithm 5 Performing the second RK step in the LT approach.

for $\ell = L \rightarrow \ell_{\min}$, $\ell = \ell - 1$ **do**

if $\ell \neq L$ **then**

Project the leaves and internal nodes from level $\ell + 1$ onto level ℓ (Algorithm 4);

Predict the values of the virtual leaves of level $\ell + 1$ at instant $t^n + 2\Delta t_{\ell+1}$;

{These two steps compute the update in time of the virtual leaves}

end if

for every leaf $\in \Omega^\ell$ **do**

Flux computation using the values at instant $t^n + \Delta t_\ell$;

Second step of compact RK;

Perform the approximation at instant $t^n + 2\Delta t_\ell$ given in Equation (14);

if we are computing RK2 time evolution **then**

Compute $\bar{\mathbf{q}}_{\theta=\frac{1}{2}}$;

else if we are computing RK3 time evolution **then**

Compute $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}$;

Compute $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}$;

end if

end for

end for

Algorithm 6 Projection procedure inside the third RK step.

Require: Scale ℓ to receive the projection.

Require: Number of dimensions d of the problem.

for every internal node $\in \Omega^\ell$ **do**

$\bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell) = 0$; {result after RK2, set here to zero to store the projection}

for Every child cell $\in \Omega^{\ell+1}$ **do**

$\bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell) \leftarrow \bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell) + \bar{\mathbf{q}}_{\ell+1}(t^n + \Delta t_{\ell+1})$; {Add value $\bar{\mathbf{q}}_{\ell+1}$ from RK3 3rd step }

end for

$\bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell) \leftarrow \frac{1}{2^d} \bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell)$

$\bar{\mathbf{q}}_\ell^{**} \approx \bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell)$;

To obtain a 2nd order approximation for $\bar{\mathbf{q}}_\ell(t^n + \Delta t_\ell)$, use Equation (16).

end for

Algorithm 7 Performing the third RK step in the LT approach.

for $\ell = L \rightarrow \ell_{\min}$, $\ell = \ell - 1$ **do**

if $\ell \neq L$ **then**

– Project the leaves and internal nodes from level $\ell + 1$ onto level ℓ (Algorithm 6);

– Predict the values of the virtual leaves of level $\ell + 1$ at instant $t^n + \Delta t_{\ell+1}$;

{These two steps compute the update in time of the virtual leaves}

end if

for Every leaf $\in \Omega^\ell$ **do**

– Flux computation using the values at instant $t^n + \frac{1}{2}\Delta t_\ell$;

– Third step of compact RK;

end for

end for

Algorithm 8 Tree refreshing before the second RK step.

Require: Scale ℓ to receive the projection.

Require: Number of dimensions d of the problem.

```

for  $\ell = L - 1 \rightarrow \ell_{\min}, \ell = \ell - 1$  do
  for Every internal node  $\in \Omega^\ell$  do
     $\bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell) = 0$ ; {Set the solution equal zero to perform the averaging of its children cells.}
    for Every child cell  $i \in \Omega^{\ell+1}$  do
       $\bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell) \leftarrow \bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell) + \bar{\mathbf{q}}_{\ell+1, i}^*$ ;
    end for
     $\bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell) \leftarrow \frac{1}{2^d} \bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell)$ 
    Obtain a 1st order approximation for  $\bar{\mathbf{q}}_\ell^*$  using Equation (18).
  end for
end for
for  $\ell = \ell_{\min} \rightarrow L, \ell = \ell + 1$  do
  for Every virtual leaf  $\in \Omega^\ell$  do
    Use the approximated solution at level  $\ell - 1$  at time instant  $t^n + \frac{1}{2}\Delta t_{\ell-1}$  to predict the solution  $\bar{\mathbf{q}}_\ell^*$ .
    Obtain the solution  $\bar{\mathbf{q}}_\ell(t^n + \frac{1}{2}\Delta t_\ell)$  by linear interpolation. {These two steps predict the solution of
    the virtual leaves in the proper time instant to update the level  $\ell$ .}
  end for
end for

```

Algorithm 9 Tree refreshing before the third RK step.

Require: Scale ℓ to receive the projection.

Require: Number of dimensions d of the problem.

```

for  $\ell = L - 1 \rightarrow \ell_{\min}, \ell = \ell - 1$  do
  for Every internal node  $\in \Omega^\ell$  do
     $\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}} = 0$ ;
    for Every child cell  $i \in \Omega^{\ell+1}$  do
       $\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}} \leftarrow \bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}} + \bar{\mathbf{q}}_{\ell+1, i}^{**}$ ;
    end for
     $\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}} \leftarrow \frac{1}{2^d} \bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}}$ 
    Compute  $\bar{\mathbf{q}}_{\ell, \theta=\frac{3}{4}}$  using Equation (20).
    Compute  $\bar{\mathbf{q}}_\ell^{**}$  using Equation (21).
  end for
end for
for  $\ell = \ell_{\min} \rightarrow L, \ell = \ell + 1$  do
  for Every virtual leaf  $\in \Omega^\ell$  do
    if  $\ell = \ell_{\min}$  then
      Use the solution  $\bar{\mathbf{q}}_{\ell-1, \theta=\frac{3}{4}}$ , at time instant  $t^n + \frac{1}{2}\Delta t_\ell$ , to predict the solution  $\bar{\mathbf{q}}_\ell^{**}$ .
    else
      Use the solution  $\bar{\mathbf{q}}_{\ell-1, \theta=\frac{1}{4}}$ , at time instant  $t^n + \frac{1}{2}\Delta t_\ell$ , to predict the solution  $\bar{\mathbf{q}}_\ell^{**}$ .
    end if
    Compute  $\bar{\mathbf{q}}_{\ell, \theta=\frac{1}{4}}$  using Equation (21).
    Compute  $\bar{\mathbf{q}}_{\ell, \theta=\frac{3}{4}}$  using Equation (20).
  end for
end for

```
