Second order accurate asynchronous scheme for modeling linear partial differential equations

Asma Toumi, Guillaume Dufour, Ronan Perrussel, Thomas Unfer

Second order accurate asynchronous scheme for modeling linear partial differential equations ☆

Asma Toumi [a,*], Guillaume Dufour [a,1], Ronan Perrussel [b], Thomas Unfer [b]

[a] *ONERA, 2 avenue Edouard Belin, 31400 Toulouse, France*
[b] *LAPLACE-ENSEEIHT 2, rue Charles Camichel BP 7122, 31071 Toulouse Cedex 7, France*

We propose an asynchronous method for the explicit integration of multi-scale partial differential equations. This method is restricted by a local CFL (Courant Friedrichs Lewy) condition rather than the traditional global CFL condition. Moreover, contrary to other local time-stepping (LTS) methods, the asynchronous algorithm permits the selection of independent time steps in each mesh element. We derived an asynchronous Runge–Kutta 2 (ARK2) scheme from a standard explicit Runge–Kutta method and we proved that the ARK2 scheme is second order convergent. Comparing with the classical integration, the asynchronous scheme is effective in terms of computation time.

1. Introduction

Numerical simulation has become a central tool for the modeling of many physical systems (fluid dynamics, plasmas, electromagnetism, etc.). Multi-scale phenomena make the integration of these physical systems difficult in terms of accuracy and computation time. Time-stepping integration techniques used for solving such problems generally fall into two categories: explicit and implicit schemes. In the explicit schemes, all unknown variables are computed at the current time level from quantities already available. Time step is then limited by the most restrictive CFL condition over the whole computation domain. The implicit method allow to overcome this time step constraint but comes with an additional computational cost arising from solving large linear systems. Thus the implicit approach becomes effective in terms of computational cost only if the time steps used are quite large with respect to the CFL condition. However this is not always possible, especially when strongly coupled and multi-scale phenomena such as microwave plasma generation are studied.

To overcome such problems, a number of local time-stepping approaches have been developed. These methods are restricted by a local CFL condition rather than the traditional global CFL condition. In [14], Osher and Sanders proposed a local time-stepping scheme for one-dimensional scalar conservation laws. They gave a thorough analysis of a first order spatial discretization with a local forward Euler time-stepping scheme. This scheme allows each element to take either an entire time step or some fixed number of smaller steps. Tang and Warneck proposed in [17] a class of high resolution local time step schemes for hyperbolic conservations by projecting the solution increments at each local time step. Savcenco et al. [15] constructed a multirate scheme for parabolic problems. This scheme was obtained by adaptation of

☆ This work was funded by the French National Research Agency under contract No. ANR-11-MONU-0019.

* Corresponding author.
E-mail addresses: Asma.Toumi@cmla.ens-cachan.fr (A. Toumi), Guillaume.Dufour@onera.fr (G. Dufour), perrussel@laplace.univ-tlse.fr (R. Perrussel), thomas.unfer@laplace.univ-tlse.fr (T. Unfer).
[1] Fax: +33 (0)5 62 25 25 93.

an implicit Rosenbrock-type scheme. The idea is to compute a prediction for all the cells of the mesh with the trapezoid formula. Then the fine mesh is updated using information from the coarse mesh by quadratic interpolation. Dawson and Kirby [2], developed upwind methods for solving conservation laws which allow local time refinement to be coupled with local spatial refinement. In that scheme a limiter is applied which is adapted to the outcome of previous stages. W. Hundsdorfer et al. [8], noted that several multirate schemes for conservation laws have one of the following defects: there are schemes that are locally inconsistant, e.g. [14,2], and schemes that are not mass-conservative, e.g. [17,15]. They discussed these two defects for one-dimensional conservation laws. In the same context, an error analysis is presented in [7], for explicit partitioned Runge–Kutta methods and multirate methods applied to conservation laws. Hundsdorfer and al. showed that the different multirate methods studied in [8] lead to order reduction of the schemes. To guarantee mass conservation, flux-based decompositions are studied. In this case, the accuracy may deteriorate. Another approach developed by Berger and Oliger [1] involves automatically taking smaller time steps where the mesh is refined. In their approach refined grids are laid over regions of the coarse mesh. Information is then passed between the grids by means of injection and interpolation. Flaherty et al. [3] developed a parallel, adaptive discontinuous Galerkin method with a local forward Euler scheme which relies on interpolating values in time at interfaces between time steps of different sizes. This scheme, however, does not appear to conserve flux along these interfaces. Also, only first order in time methods are discussed. Note that for these different algorithms, local time steps are usually selected to be fractions of the global time step. Recently a new time integration approach has been applied to equations of non-linear elastodynamics [9]. It is based on a discrete spacetime form of Hamilton's variational principle. This algorithm permits the selection of independent time steps in each mesh element. However, this approach is only applicable to Hamiltonian systems. In [11] Omelchenko and Karimabadi presented an asynchronous approach to a diffusion–advection–reaction equation in one dimension. Their method is based on discrete-event simulation (DES). They defined the concept of "flux capacitor", a variable in which they stored the flux between two cells until a later event. Mass conservation is guaranteed when the neighboring cell is updated and the flux capacitors are emptied. In order to improve the accuracy of their asynchronous method, Omelchenko and Karimabadi developed in [12] a second order DES algorithm and showed that, at least numerically, the second order is indeed attained on one-dimensional gas dynamics test problems. More recently, they extended in [13] their DES algorithm to multiple dimensions in the case of "logically uniform meshes". In [16], V.A. Semiletov and S.A. Karabasov proposed an asynchronous time-stepping algorithm for the Compact Accurately Boundary Adjusting high-REsolution Technique (CABARET), which is an Euler method for non-linear aeroacoustic problems. Numerical tests up to 3D show that the asynchronous algorithm maintains the same second-order convergence rate as the original single-time stepping scheme and that it also decreases the absolute numerical error. However, the question of the performance in terms of computational time remains open. In [10] V.A. Semiletov and S.A. Karabasov adapted their asynchronous method for solving fluid dynamics equations. Their approach is based on a transformation of the governing equations in space and time in order to rewrite the initial problem on a uniform Cartesian grid, with adapted conditions at the grid interfaces. As for the numerical implementation of their scheme, the authors combined their new asynchronous method with their CABARET scheme presented in [16] which results in a quasi-second order scheme. The corresponding numerical tests were limited to two-dimensional problems.

In this paper we focus on the asynchronous method proposed by one of the co-author T. Unfer. His method uses local stability criteria and can be used on an arbitrary mesh. In [22], he developed an upwind asynchronous forward Euler scheme for the transport equation in one dimension. We studied the extension of this method to an arbitrary space dimension and we proved through a thorough analysis that the asynchronous scheme is first order convergent. In addition, we noticed that the asynchronous scheme reduces numerical diffusion and CPU time in comparison to the classical first order scheme. To improve the convergence rate of the asynchronous scheme, T. Unfer proposed in [21] an extension of the asynchronous integration procedure to higher order. This method was explored in some details from a practical viewpoint. We then explored it from a more theoretical viewpoint and we proved that the asynchronous scheme as it was presented in [21] is at most of order one. The derivation of higher order explicit LTS methods is a serious problem and only a few methods are available. In [18], A. Taube et al. proposed an ADER-DG scheme with time-accurate LTS for the Maxwell equations. They found via numerical study that their approach maintains the accuracy of the global time-stepping ADER-DG scheme "with a very small increase in computational effort". Grote et al. [4] derived an explicit LTS scheme from standard explicit Runge–Kutta (RK) methods. They have proved that their scheme preserves the accuracy of the original RK scheme. However, their approach is applicable only in the case of two distinct regions in the mesh: a "coarse" region with the larger elements and a "fine" region with the smaller elements. In the fine region, the time step must be a fraction of the coarse time step. Here, we develop an asynchronous RK2 scheme in an arbitrary mesh.

This paper is outlined as follows. In section 2, we summarize basic ideas of the asynchronous methodology. Then, we consider an arbitrary space discretization of a given linear partial differential equation and we apply the ARK2 method. In section 3, we prove that the asynchronous scheme is second order convergent. Next, numerical experiments that illustrate the convergence rate of the asynchronous scheme and validate the gain in computation time are presented in section 4. Finally, some conclusions and final remarks are given in section 5.

2. Numerical asynchronous simulation

For stability reasons, any explicit method has to fulfill a time-step restriction. This may reduce the method efficiency, because automatic grid generation tools for unstructured meshes may produce very small cells in geometrically complicated parts of the computational domain. To overcome this problem, we propose an asynchronous scheme with local time steps.

We consider Ω a polygonal domain in $\mathbb{R}^d$ $(d \geq 1)$. Let $\mathcal{T} = \{K_i,\, i = 1, \ldots, N\}$ be a partition of the domain Ω in N polyhedral volumes K_i. For all $i \in [\![1, N]\!]$, $\mathcal{N}(i) = \{j,\, |K_i \cap K_j| \neq 0\}$ is the set of indices of neighboring volumes of K_i; $|K_i \cap K_j|$ denotes the $(d-1)$ positive measure of $K_i \cap K_j$. Let $\mathcal{N}^+(i)$ (resp. $\mathcal{N}^-(i)$) be the set of the indices of the neighbors of the element K_i who receive (resp. send) the outflow (resp. inflow) fluxes from (resp. to) K_i.

We suppose an arbitrary space discretization of a given linear partial differential equation without a source term, for the sake of simplicity. This leads to a system of coupled ordinary differential equations

$$\frac{dy_i}{dt}(t) = B_i y(t), \quad \forall K_i \in \mathcal{T}, \tag{1}$$

where y_i and y are respectively vectors that contain the degrees of freedom (dofs) in the element K_i and in all the elements, and B_i a matrix coming from the space discretization.

2.1. Asynchronous methodology

The asynchronous methodology is based on two key ideas. First of all, the asynchronous algorithm permits the selection of independent time steps in each mesh element. Then, the sequence of update times of the cells is no longer defined by continuously adding a uniform time step as in the classical case. We need to introduce the sequence of the most urgent refresh time tag $(t^i)_{i\geq 1}$ defined by

$$t^1 = \min_j(\Delta t_j) := \Delta t_i, \quad t^2 = \min\left(\min_{j\neq i}(\Delta t_j), 2\Delta t_i\right), \quad \ldots$$

where for all i, the time step Δt_i of the element K_i is determined from the local stability restriction. Let $(t_i^r)_r$ be the sequence of update times of a given element K_i which is defined by

$$t_i^r = r\Delta t_i, \quad \forall r \geq 0.$$

For each element, the source term depends only on the current cell whereas the fluxes depend also on neighbors. Hence the second basic idea consists in separating the source term and the fluxes. In other words, we need to split the term B_i (1) in three contributions: V_i for a local volume contribution, $F_{i,j}^+$ for a local flux contribution and $F_{i,j}^-$ for a flux contribution from the neighbor. Then (1) becomes

$$\frac{dy_i}{dt}(t) = V_i y_i(t) + \sum_{j\in\mathcal{N}(i)} F_{i,j}^+ y_i(t) + F_{i,j}^- y_j(t), \quad \forall K_i \in \mathcal{T}. \tag{2}$$

The concept of the asynchronous time integration can be summarized in three main phases. First, at the start-up time all densities and fluxes are initialized. Then, a second phase is carried out by continuously applying the following steps until the global simulation clock is advanced past the simulation finish time; see also Fig. 1: find the most urgent cell to be refreshed by using the CFL condition, compute the values of the densities which are needed to compute the fluxes, compute the next refresh time of the current cell and then update the values of the fluxes. Finally, the third phase is to build the solution at the output time.

The number of updated cells in the second step in Fig. 1 depends on the treated system and on the space discretization method. For example, in the case of an upwind scheme, we need to update only the outflow fluxes and then only the current cell K_i and the neighbors K_j, $\forall j \in \mathcal{N}^+(i)$ are involved; see also [20]. Whereas for a centered scheme, we need at least to update all the neighbors. In all cases, only few cells are involved.

Note that the critical point for speeding up an asynchronous method in terms of computation time is the searching algorithm for the most urgent time tag to be treated, some alternative options are presented in [22].

2.2. ARK2 algorithm

To present the asynchronous Runge–Kutta 2 algorithm, we start with the semi-discretized system written in the following form as explained in section 2

$$\frac{dy_i}{dt}(t) = V_i y_i(t) + \sum_{j\in\mathcal{N}(i)} F_{i,j}^+ y_i(t) + F_{i,j}^- y_j(t), \quad \forall K_i \in \mathcal{T}. \tag{3}$$

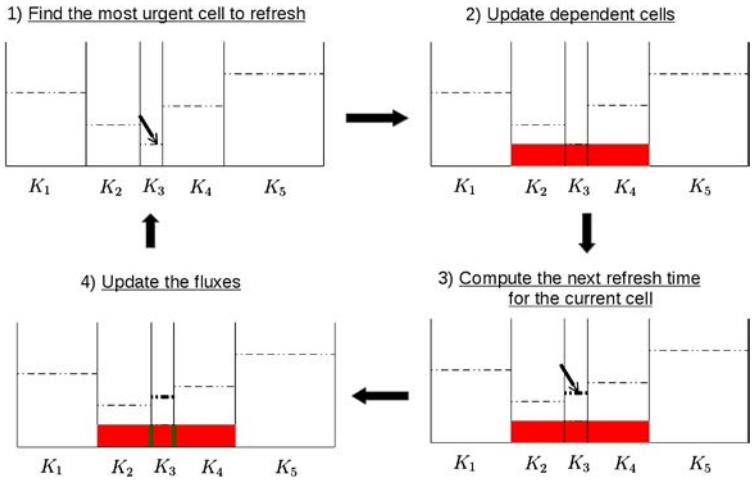

Fig. 1. The four steps of the second phase of the asynchronous algorithm with five elements.

We denote by y_i^m the approximation of $y_i(m\Delta t_i)$ and by $\left(\frac{\Delta y}{\Delta t}\right)_i^m$ the approximation of $\frac{dy_i}{dt}(m\Delta t_i)$. Pre-computed update times correspond to multiples of Δt_i in each K_i. The time advance is performed using the Discrete Time Scheduler (DTS) algorithm as defined in [22], t_{cur} is the current simulation time.

At t_{cur} such that $t_{\text{cur}}/\Delta t_i$ is an integer p, we have to perform the following operations

1. Compute the tentative dofs at t_{cur}

$$\tilde{y}_i(t_{\text{cur}}) = \tilde{y}_i(t_{\text{pre},i}) + (t_{\text{cur}} - t_{\text{pre},i})\left(\frac{\Delta\tilde{y}}{\Delta t}\right)_i(t_{\text{mid},i}),$$

where $t_{\text{mid},i} = t_{\text{cur}} - \left(\frac{t_{\text{cur}}-t_{\text{pre},i}}{2}\right)$ and $t_{\text{pre},i}$ is the last time where the tentative dofs has been updated. The tentative mid-point slope is defined as follows

$$\left(\frac{\Delta\tilde{y}}{\Delta t}\right)_i(t_{\text{mid},i}) = V_i\left(y_i^{p-1} + \frac{\Delta t_i}{2}\left(\frac{\Delta y}{\Delta t}\right)_i^{p-1}\right) + \sum_{j\in\mathcal{N}^+(i)}\left[F_{i,j}^+\left(y_i^{p-1} + \left(t_{\text{mid},i,j} - (p-1)\Delta t_i\right)\left(\frac{\Delta y}{\Delta t}\right)_i^{p-1}\right)\right]$$
$$+ \sum_{j\in\mathcal{N}^-(i)}\left[F_{i,j}^-\left(y_j^{n_j} + \left(t_{\text{mid},i,j} - n_j\Delta t_j\right)\left(\frac{\Delta y}{\Delta t}\right)_j^{n_j}\right)\right],$$

where $n_j = \left\lfloor \frac{t_{\text{pre},i}}{\Delta t_j} \right\rfloor$, $t_{\text{mid},i,j} = t_{\text{next},i,j} - \left(\frac{t_{\text{next},i,j} - t_{\text{pre},i,j}}{2}\right)$ with $t_{\text{pre},i,j} = \max(n_j\Delta t_j, (p-1)\Delta t_i)$ and $t_{\text{next},i,j} = \min((n_j+1)\Delta t_j, p\Delta t_i)$.

2. For all the neighbors of the element i, *i.e.* $j \in \mathcal{N}(i)$, we update the tentative values

$$\tilde{y}_j(t_{\text{cur}}) = \tilde{y}_j(t_{\text{pre},j}) + (t_{\text{cur}} - t_{\text{pre},j})\left(\frac{\Delta\tilde{y}}{\Delta t}\right)_j(t_{\text{mid},j}),$$

$t_{\text{pre},j}$ is the last time where the tentative dofs has been updated and $t_{\text{mid},j}$ is the "mid-point" time for the element j.

3. Affect the dofs and update the last refresh times

$$y_i^p = \tilde{y}_i(t_{\text{cur}}), \quad t_{\text{pre},i} = t_{\text{cur}} \text{ and } t_{\text{pre},j} = t_{\text{cur}}, \quad \forall j \in \mathcal{N}(i).$$

4. After all elements K_i such that $t_{\text{cur}}/\Delta t_i$ is an integer have been updated, update their slopes

$$\left(\frac{\Delta y}{\Delta t}\right)_i^p = V_i y_i^p + \sum_{j\in\mathcal{N}(i)} F_{i,j}^+ y_i^p + F_{i,j}^-\left(y_j^{n_j} + (p\Delta t_i - n_j\Delta t_j)\left(\frac{\Delta y}{\Delta t}\right)_j^{n_j}\right),$$

where $n_j = \left\lfloor \frac{t_{\text{cur}}}{\Delta t_j} \right\rfloor$.

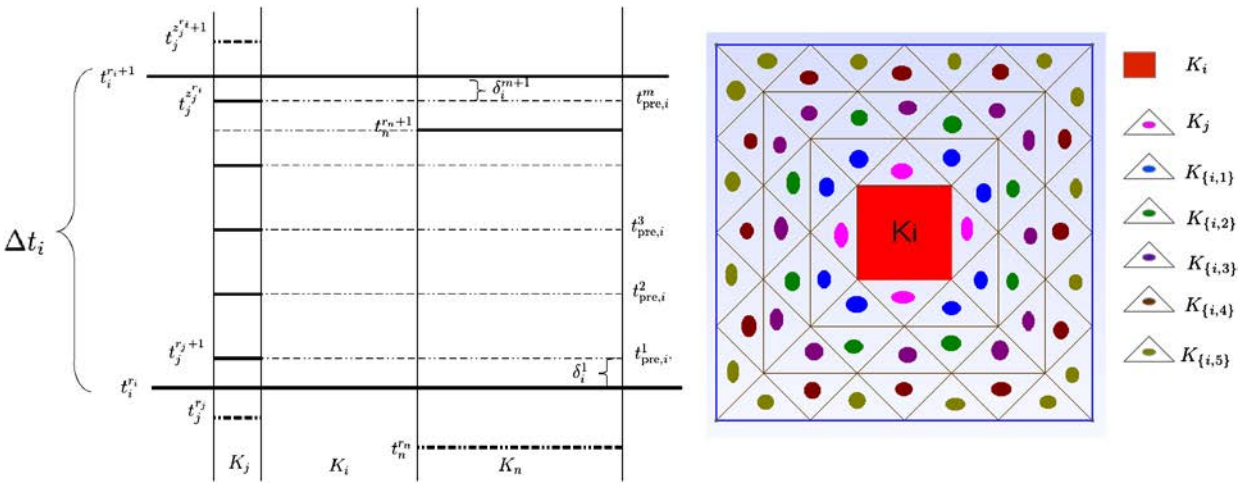

Fig. 2. Notation: the temporal evolution of the system between two instants $t_i^{r_i}$ and $t_i^{r_i+1}$ where the elements j and n are the neighbors of i (left). Dependency of the element K_i on the other elements of the mesh (right). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

The final simulation time should correspond to a common meeting time of all cells *i.e.* a multiple of the lowest common multiple (LCM) of all the local time steps. If it is not the case, we use an interpolation at least of order 2.

3. Properties of the asynchronous scheme

3.1. Convergence of the ARK2 scheme

The asynchronous RK2 scheme is derived from the standard explicit Runge–Kutta 2 method. We shall now prove that the ARK2 scheme is second order convergent.

3.1.1. Notations

1. Let us begin with some notations concerning the temporal evolution of the system between two successive update times of an element K_i which are multiples of Δt_i; see also Fig. 2 (left)
   (a) Let $t_i^{r_i+1} := (r_i+1)\Delta t_i = t_{\text{cur}} \le T$ where T is the final time of the simulation. Suppose that the element K_i has been updated m times between $t_i^{r_i}$ and $t_i^{r_i+1}$ at the instants: $t^1_{\text{pre},i}$, $t^2_{\text{pre},i}$, ..., $t^m_{\text{pre},i}$, $0 \le m \le \left\lfloor \frac{\Delta t_i}{\min_{j\in\mathcal{N}(i)}(\Delta t_j)} \right\rfloor + 1$.
   (b) Let $\delta_i^k = t^k_{\text{pre},i} - t^{k-1}_{\text{pre},i}$, for all $2 \le k \le m$, $\delta_i^1 = t^1_{\text{pre},i} - t_i^{r_i}$, $\delta_i^{m+1} = t_i^{r_i+1} - t^m_{\text{pre},i}$ and $\sum_{k=1}^{m+1} \delta_i^k = \Delta t_i$. Then, $t^k_{\text{mid},i} = t^k_{\text{pre},i} - \delta_i^k/2$ for $1 \le k \le m$ and $t^{m+1}_{\text{mid},i} = t_i^{r_i+1} - \delta_i^{m+1}/2$.
   (c) Suppose that for all $j \in \mathcal{N}(i)$ we have the following assumptions:
      i. between $t_i^{r_i}$ and $t_i^{r_i+1}$, the element j has been updated $z_j^{r_i}$ times at instants which are multiples of Δt_j then $0 \le z_j^{r_i} \le m$. For example, in Fig. 2 (left), $z_n^{r_i} = 1$ and $z_j^{r_i} = 5$ where n and j are two neighbors of i.
      ii. $r_j\Delta t_j \le r_i\Delta t_i$ and at the instant $t^k_{\text{pre},i}$, $r_j^k\Delta t_j$ denote the last update time of the element j which is multiple of Δt_j just before $t^k_{\text{pre},i}$. Note that there may be situations where $r_j^{k+1}\Delta t_j = r_j^k\Delta t_j$. In Fig. 2 for instance, for the neighbor n, $r_j^4\Delta t_j = r_j^3\Delta t_j = r_j^2\Delta t_j = r_j^1\Delta t_j = r_n\Delta t_n$.
      iii. $t^k_{\text{mid},i,j} = t^k_{\text{next},i,j} - \left(\frac{t^k_{\text{next},i,j} - t^k_{\text{pre},i,j}}{2}\right) = \frac{t^k_{\text{next},i,j} + t^k_{\text{pre},i,j}}{2}$, with:
         A. $t^k_{\text{pre},i,j} = \max(r_j^k\Delta t_j, r_i\Delta t_i)$;
         B. $t^k_{\text{next},i,j} = \min((r_j^k+1)\Delta t_j, (r_i+1)\Delta t_i)$.
2. The update of an element K_i involves its neighbors. The neighbors of K_i depend on their neighbors, the neighbors of the neighbors of K_i involves their neighbors and so on. Then, we introduce the following notations; see also Fig. 2 (right).
   (a) The elements K_j, $j \in \mathcal{N}(i)$, are the neighbors of K_i.
   (b) The elements $K_{\{i,1\}}$, $\{i,1\} \in \mathcal{N}(j)$, are the neighbors of K_j and then the neighbors of the neighbors of K_i.
   (c) The elements $K_{\{i,2\}}$, $\{i,2\} \in \mathcal{N}(\{i,1\})$, are the neighbors of the neighbors of the neighbors of K_i.
   (d) The elements $K_{\{i,n\}}$, $\{i,n\} \in \mathcal{N}(\{i,n-1\})$, are the neighbors of the elements $K_{\{i,n-1\}}$.

Let $h^{r_i}_{\{i,t_s\}}$ be the number of dependencies presented by the element K_i between $t_i^{r_i}$ and t_s, where t_s is the time of the last synchronization of the solution which can particularly be the initial time t^0. In other words, the elements $K_{\{i,h^{r_i}_{i,t_s}\}}$ use their values computed at the time of the last synchronization of the solution. Note that the study of the dependency will be necessary for the theoretical study of the ARK2 scheme which needs the control of the temporal evolution of the system. However, for the asynchronous algorithm, the update of a given element involves only the neighbors and then no need of the neighbors of the neighbors. This dependency is implicit and it is hidden behind the intermediate steps of the algorithm.

3. Let us introduce some elements for the identification:

$$(Bx)_i = V_i x_i + \sum_{j\in\mathcal{N}^+(i)} F^+_{i,j} x_i + \sum_{j\in\mathcal{N}^-(i)} F^-_{i,j} x_j$$
$$(BP_i x)_i = V_i x_i + \sum_{j\in\mathcal{N}^+(i)} F^+_{i,j} x_i$$

where P_i is a diagonal projection matrix on the dof of the element i.

Proposition 1. *The evolution of any given cell i between the two instants $t_i^{r_i} = r_i\Delta t_i$ and $t_i^{r_i+1} = t_i^{r_i} + \Delta t_i$ is given by the following expression*

$$y_i^{r_i+1} = y_i^{r_i} + \Delta t_i\left[V_i\left(y_i^{r_i} + \frac{\Delta t_i}{2}\left(\frac{\Delta y}{\Delta t}\right)^{r_i}_i\right)\right] + \sum_{k=1}^{m+1}\delta_i^k\left[\sum_{j\in\mathcal{N}^+(i)} F^+_{i,j}\left(y_i^{r_i} + \left(t^k_{\mathrm{mid},i,j} - r_i\Delta t_i\right)\left(\frac{\Delta y}{\Delta t}\right)^{r_i}_i\right)\right]$$
$$+ \sum_{k=1}^{m+1}\delta_i^k\left[\sum_{j\in\mathcal{N}^-(i)} F^-_{i,j}\left(y_j^{r_j^k} + \left(t^k_{\mathrm{mid},i,j} - r_j^k\Delta t_j\right)\left(\frac{\Delta y}{\Delta t}\right)^{r_j^k}_j\right)\right].$$

Proof. By definition, we have

$$\begin{aligned} y_i^{r_i+1} &= \tilde{y}_i^{t^m_{\mathrm{pre},i}} + \delta_i^{m+1}\left(\frac{\Delta\tilde{y}}{\Delta t}\right)_i(t^{m+1}_{\mathrm{mid},i}) \\ &= \tilde{y}_i^{t^{m-1}_{\mathrm{pre},i}} + \delta_i^{m}\left(\frac{\Delta\tilde{y}}{\Delta t}\right)_i(t^{m}_{\mathrm{mid},i}) + \delta_i^{m+1}\left(\frac{\Delta\tilde{y}}{\Delta t}\right)_i(t^{m+1}_{\mathrm{mid},i}) \\ &\vdots \\ &= y_i^{r_i} + \sum_{k=1}^{m+1}\delta_i^k\left(\frac{\Delta\tilde{y}}{\Delta t}\right)_i(t^k_{\mathrm{mid},i}) \end{aligned}$$

where for all $1 \le k \le m+1$

$$\left(\frac{\Delta\tilde{y}}{\Delta t}\right)_i(t^k_{\mathrm{mid},i}) = V_i\left(y_i^{r_i} + \frac{\Delta t_i}{2}\left(\frac{\Delta y}{\Delta t}\right)^{r_i}_i\right) + \sum_{j\in\mathcal{N}^+(i)} F^+_{i,j}\left(y_i^{r_i} + \left(t^k_{\mathrm{mid},i,j} - r_i\Delta t_i\right)\left(\frac{\Delta y}{\Delta t}\right)^{r_i}_i\right)$$
$$+ \sum_{j\in\mathcal{N}^-(i)} F^-_{i,j}\left(y_j^{r_j^k} + \left(t^k_{\mathrm{mid},i,j} - r_j^k\Delta t_j\right)\left(\frac{\Delta y}{\Delta t}\right)^{r_j^k}_j\right)$$

To conclude, we just use that $\sum_{k=1}^{m+1}\delta_i^k = \Delta t_i$. □

Proposition 2. *For any given element i, and between two instants $t_i^{r_i}$ and $t_i^{r_i+1}$, we have the following result*

$$\sum_{k=1}^{m+1}\delta_i^k \times t^k_{\mathrm{mid},i,j} = \frac{(2r_i+1)(\Delta t_i)^2}{2}, \forall j \in \mathcal{N}(i). \tag{4}$$

In particular, between the two instants t_s, any time of the synchronization of the solution, and $t_s + \Delta t_i$ we have $\forall j \in \mathcal{N}(i)$

$$\sum_{k=1}^{m+1}\delta_i^k \times t^k_{\mathrm{mid},i,j} = \frac{(\Delta t_i)^2}{2}.$$

Proof. We first prove that, for a given element $j \in \mathcal{N}(i)$, between two successive times of update of the element K_j which are multiples of Δt_j and which are between $t_i^{r_i}$ and $t_i^{r_i+1}$, $t^k_{\text{mid},i,j}$ is constant. In other words, if $r_i\Delta t_i \le (r_j+1)\Delta t_j < (r_j+l)\Delta t_j \le (r_i+l+1)\Delta t_i$, l integer, then for all k such that $(r_j+l)\Delta t_j < t^k_{\text{pre},i} \le (r_j+l+1)\Delta t_j$ we have

$$t^k_{\text{pre},i,j} = \max(r_j^k\Delta t_j, r_i\Delta t_i) = \max((r_j+l)\Delta t_j, r_i\Delta t_i) = (r_j+l)\Delta t_j$$

and

$$\begin{aligned} t^k_{\text{next},i,j} &= \min((r_j^k+1)\Delta t_j, (r_i+1)\Delta t_i) \\ &= \min((r_j+l+1)\Delta t_j, (r_i+1)\Delta t_i) = (r_j+l+1)\Delta t_j. \end{aligned}$$

As a result,

$$t^k_{\text{mid},i,j} = \frac{(2(r_j+l)+1)\Delta t_j}{2}. \tag{5}$$

Now, we prove Equation (4). Suppose that for all $j \in \mathcal{N}(i)$, $z_j^{r_i}$ is the number of times of update of the element K_j at times that are multiples of Δt_j between $r_i\Delta t_i$ and $(r_i+1)\Delta t_i$. There are three cases:

1. If $z_j^{r_i} = 0$ then $r_j\Delta t_j \le r_i\Delta t_i \le (r_i+1)\Delta t_i \le (r_j+1)\Delta t_j$ and for all $1 \le k \le m+1$,
$$t^k_{\text{mid},i,j} = \frac{(r_i+1)\Delta t_i + r_i\Delta t_i}{2} = \frac{(2r_i+1)\Delta t_i}{2},$$
therefore,
$$\sum_{k=1}^{m+1} \delta_i^k \times t^k_{\text{mid},i,j} = \frac{(2r_i+1)(\Delta t_i)^2}{2}.$$

2. If $z_j^{r_i} = 1$ then we have
$$\begin{aligned} \sum_{k=1}^{m+1} \delta_i^k \times t^k_{\text{mid},i,j} &= \frac{((r_j+1)\Delta t_j + r_i\Delta t_i)}{2} \times ((r_j+1)\Delta t_j - r_i\Delta t_i) \\ &+ \frac{((r_i+1)\Delta t_i + (r_j+1)\Delta t_j)}{2} \times \left((r_i+1)\Delta t_i - (r_j+1)\Delta t_j\right) \\ &= \frac{1}{2}\left[(r_i+1)^2\Delta t_i^2 - r_i^2\Delta t_i^2\right] \\ &= \frac{(2r_i+1)(\Delta t_i)^2}{2} \end{aligned}$$

3. If $z_j^{r_i} \ge 2$ then $r_j\Delta t_j \le r_i\Delta t_i < (r_j+1)\Delta t_j < \ldots < (r_j+z_j^{r_i})\Delta t_j \le (r_i+1)\Delta t_i < (r_j+z_j^{r_i}+1)\Delta t_j$. Using (5) we have the following result
$$\begin{aligned} \sum_{k=1}^{m+1} \delta_i^k \times t^k_{\text{mid},i,j} &= \frac{((r_j+1)\Delta t_j + r_i\Delta t_i)}{2} \times ((r_j+1)\Delta t_j - r_i\Delta t_i) + \sum_{s=1}^{z_j^{r_i}-1} \frac{(2(r_j+s)+1)\Delta t_j}{2} \times \Delta t_j \\ &+ \frac{((r_i+1)\Delta t_i + (r_j+z_j^{r_i})\Delta t_j)}{2} \times ((r_i+1)\Delta t_i - (r_j+z_j^{r_i})\Delta t_j) \\ &= \frac{((r_j+1)\Delta t_j)^2}{2} - \frac{(r_i\Delta t_i)^2}{2} + \frac{((r_i+1)\Delta t_i)^2}{2} - \frac{((r_j+z_j^{r_i})\Delta t_j)^2}{2} + \sum_{l=1}^{z_j^{r_i}-1} (2(r_j+l)+1) \times \frac{(\Delta t_j)^2}{2}. \end{aligned}$$

Note that $\sum_{l=1}^{z_j^{r_i}-1}(2(r_j+l)+1)$ is a sum of an arithmetic sequence with a common difference equal to two. Therefore,
$$\begin{aligned} \sum_{l=1}^{z_j^{r_i}-1} (2(r_j+l)+1) &= \left[\sum_{l=0}^{z_j^{r_i}-1} (2(r_j+l)+1)\right] - (2r_j+1) \\ &= z_j^{r_i}\frac{(4r_j+2z_j^{r_i})}{2} - (2r_j+1) \\ &= \left(z_j^{r_i}\right)^2 + 2r_j(z_j^{r_i}-1) - 1 \end{aligned}$$

and then

$$\sum_{k=1}^{m+1} \delta_i^k \times t_{\text{mid},i,j}^k = \frac{(2r_i+1)(\Delta t_i)^2}{2}.$$

The proof is then complete. □

Proposition 3. *For any element i and at each time $t_i^{r_i} = r_i \Delta t_i$, r_i an integer, we have the following expression of the slope i*

$$\begin{aligned}\left(\frac{\Delta y}{\Delta t}\right)_i^{r_i} &= \left(BP_i y^{r_i}\right)_i + \sum_{j\in\mathcal{N}^-(i)} F_{i,j}^- \left[y_j^{r_j} + \delta_{i,j} \left(\left(BP_j y^{r_j}\right)_j + \sum_{\{i,1\}\in\mathcal{N}^-(j)} F_{\{i,1\},j}^- y_{\{i,1\}}^{r_{\{i,1\}}} \right) \right] \\ &+ \sum_{z=1}^{h_{i,t_s}^{r_i}-1} \left(\prod_{n=1}^{z} \delta_{\{i,n-1\},\{i,n\}} \sum_{\{i,n\}\in\mathcal{N}^-(\{i,n-1\})} F_{\{i,n-1\},\{i,n\}}^- \right) \\ &\times \left(\left(BP_{\{i,z\}} y^{r_{\{i,z\}}}\right)_{\{i,z\}} + \sum_{\{i,z+1\}\in\mathcal{N}^-(\{i,z\})} F_{\{i,z\},\{i,z+1\}}^- y_{\{i,z+1\}}^{r_{\{i,z+1\}}} \right),\end{aligned}$$

where for all $j\in\mathcal{N}^-(i)$, $\delta_{i,j} = r_i\Delta t_i - r_j\Delta t_j > 0$ and $r_j\Delta t_j$ is the last time, multiple of Δt_j, of updating the element j just before $t_i^{r_i}$. For all $1 \le z < h_{i,t_s}^{r_i} - 1$, $r_{\{i,z\}}\Delta t_{\{i,z\}} < t_i^{r_i}$ and $\delta_{\{i,z-1\},\{i,z\}} = r_{\{i,z-1\}}\Delta t_{\{i,z-1\}} - r_{\{i,z\}}\Delta t_{\{i,z\}} > 0$ where t_s is the time of the last synchronization of the solution and $h_{\{i,t_s\}}^{r_i}$ is the number of dependencies presented by the element K_i between $t_i^{r_i}$ and t_s. We also supposed that $K_{\{i,0\}} \equiv K_j$, $j\in\mathcal{N}(i)$ are the neighbors of K_i.

Proof. The expression of the slope i at $t_i^{r_i}$ is given by the following formula

$$\begin{aligned}\left(\frac{\Delta y}{\Delta t}\right)_i^{r_i} &= V_i y_i^{r_i} + \sum_{j\in\mathcal{N}^+(i)} F_{i,j}^+ y_i^{r_i} + \sum_{j\in\mathcal{N}^-(i)} F_{i,j}^- \left(y_j^{r_j} + \delta_{i,j} \left(\frac{\Delta y}{\Delta t}\right)_j^{r_j} \right) \\ &= \left(BP_i y^{r_i}\right)_i + \sum_{j\in\mathcal{N}^-(i)} F_{i,j}^- \left(y_j^{r_j} + \delta_{i,j} \left(\frac{\Delta y}{\Delta t}\right)_j^{r_j} \right),\end{aligned}$$

the elements j are the neighbors of the element i and for all $j\in\mathcal{N}^-(i)$, $\delta_{i,j} = r_i\Delta t_i - r_j\Delta t_j > 0$ where $r_j\Delta t_j = t_j^{r_j} < t_i^{r_i}$ is the last time, multiple of Δt_j, where the element j has been updated. Using the notations illustrated in Fig. 2 we have the following result

$$\left(\frac{\Delta y}{\Delta t}\right)_j^{r_j} = \left(BP_j y^{r_j}\right)_j + \sum_{\{i,1\}\in\mathcal{N}^-(j)} F_{j,\{i,1\}}^- \left(y_{\{i,1\}}^{r_{\{i,1\}}} + \delta_{j,\{i,1\}} \left(\frac{\Delta y}{\Delta t}\right)_{\{i,1\}}^{r_{\{i,1\}}} \right),$$

where for all $j\in\mathcal{N}(i)$ and for all $\{i,1\}\in\mathcal{N}(j)$, $r_{\{i,1\}}\Delta t_{\{i,1\}} = t_{\{i,1\}}^{r_{\{i,1\}}} < t_j^{r_j}$ is the last time, multiple of $\Delta t_{\{i,1\}}$, where the element $K_{\{i,1\}}$ has been updated. Then $\delta_{j,\{i,1\}} = r_j\Delta t_j - r_{\{i,1\}}\Delta t_{\{i,1\}} > 0$ and

$$\left(\frac{\Delta y}{\Delta t}\right)_{\{i,1\}}^{r_{\{i,1\}}} = \left(BP_{\{i,1\}} y^{r_{\{i,1\}}}\right)_{\{i,1\}} + \sum_{\{i,2\}\in\mathcal{N}^-(\{i,1\})} F_{\{i,1\},\{i,2\}}^- \left(y_{\{i,2\}}^{r_{\{i,2\}}} + \delta_{\{i,1\},\{i,2\}} \left(\frac{\Delta y}{\Delta t}\right)_{\{i,2\}}^{r_{\{i,2\}}} \right).$$

Likewise,

$$\left(\frac{\Delta y}{\Delta t}\right)_{\{i,2\}}^{r_{\{i,2\}}} = \left(BP_{\{i,2\}} y^{r_{\{i,2\}}}\right)_{\{i,2\}} + \sum_{\{i,3\}\in\mathcal{N}^-(\{i,2\})} F_{\{i,2\},\{i,3\}}^- \left(y_{\{i,3\}}^{r_{\{i,3\}}} + \delta_{\{i,2\},\{i,3\}} \left(\frac{\Delta y}{\Delta t}\right)_{\{i,3\}}^{r_{\{i,3\}}} \right).$$

In the same way, we define for all $3 \le z < h_{i,t_s}^{r_i} - 1$, the slopes $\left(\frac{\Delta y}{\Delta t}\right)_{\{i,z\}}^{r_{\{i,z\}}}$ and $\delta_{\{i,z-1\},\{i,z\}} = r_{\{i,z-1\}}\Delta t_{\{i,z-1\}} - r_{\{i,z\}}\Delta t_{\{i,z\}} > 0$. In particular, for $z = h_{i,t_s}^{r_i} - 1$ the slope is defined by

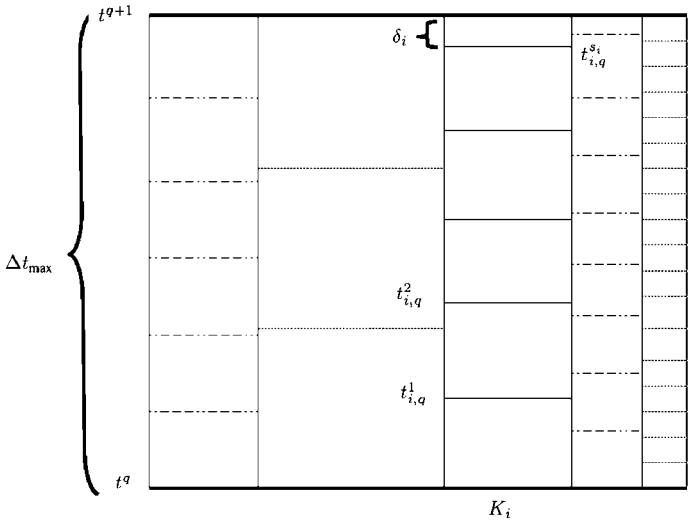

Fig. 3. The temporal evolution of the system between two instants of synchronization t^q and $t^{q+1} = t^q + \Delta t_{\max}$; example of five elements.

$$\left(\frac{\Delta y}{\Delta t}\right)^{r_{\{i,h_{i,t_s}^{r_i}-1\}}}_{\{i,h_{i,t_s}^{r_i}-1\}} = \left(BP_{\{i,h_{i,t_s}^{r_i}-1\}} y^{r_{\{i,h_{i,t_s}^{r_i}-1\}}}\right)_{\{i,h_{i,t_s}^{r_i}-1\}} + \sum_{\{i,h_{i,t_s}^{r_i}\}\in\mathcal{N}^-(\{i,h_{i,t_s}^{r_i}-1\})} F^-_{\{i,h_{i,t_s}^{r_i}-1\},\{i,h_{i,t_s}^{r_i}\}} \Bigg[y^{t_s}_{\{i,h_{i,s}^{r_i}\}} + \delta_{\{i,h_{i,t_s}^{r_i}-1\},\{i,h_{i,t_s}^{r_i}\}} \left(\frac{\Delta y}{\Delta t}\right)^{t_s}_{\{i,h_{i,t_s}^{r_i}\}} \Bigg],$$

where, $t^{r_{\{i,h_{i,t_s}^{r_i}\}}}_{\{i,h_{i,t_s}^{r_i}\}} = t_s$. Consequently,

$$\begin{aligned}\left(\frac{\Delta y}{\Delta t}\right)^{r_i}_i &= \left(BP_i y^{r_i}\right)_i + \sum_{j\in\mathcal{N}^-(i)} F^-_{i,j}\Bigg[y_j^{r_j} + \delta_{i,j}\Bigg[\left(BP_j y^{r_j}\right)_j + \sum_{\{i,1\}\in\mathcal{N}^-(j)} F^-_{j,\{i,1\}}\Bigg[y^{r_{\{i,1\}}}_{\{i,1\}} + \delta_{j,\{i,1\}}\Bigg[\left(BP_{\{i,1\}} y^{r_{\{i,1\}}}\right)_{\{i,1\}} \\ &+ \sum_{\{i,2\}\in\mathcal{N}^-(\{i,1\})} F^-_{\{i,1\},\{i,2\}}\Bigg[y^{r_{\{i,2\}}}_{\{i,2\}} + \delta_{\{i,1\},\{i,2\}}\Bigg[\ldots + \ldots + \delta_{\{i,h_i^{r_i}-2\},\{i,h_i^{r_i}-1\}}\Bigg[\left(BP_{\{i,h_i^{r_i}-1\}} y^{r_{\{i,h_i^{r_i}-1\}}}\right)_{\{i,h_i^{r_i}-1\}} \\ &+ \sum_{\{i,h_i^{r_i}\}\in\mathcal{N}^-(\{i,h_i^{r_i}-1\})} F^-_{\{i,h_i^{r_i}-1\},\{i,h_i^{r_i}\}}\Bigg[y^0_{\{i,h_i^{r_i}\}} + \delta_{\{i,h_i^{r_i}-1\},\{i,h_i^{r_i}\}} (By^0)_{\{i,h_i^{r_i}\}}\Bigg]\Bigg]\Bigg]\Bigg]\Bigg]\Bigg]\Bigg]\Bigg]\end{aligned}$$

and then

$$\begin{aligned}\left(\frac{\Delta y}{\Delta t}\right)^{r_i}_i &= \left(BP_i y^{r_i}\right)_i + \sum_{j\in\mathcal{N}^-(i)} F^-_{i,j}\left(y_j^{r_j} + \delta_{i,j}\left(\left(BP_j y^{r_j}\right)_j + \sum_{\{i,1\}\in\mathcal{N}^-(j)} F^-_{\{i,1\},j} y^{r_{\{i,1\}}}_{\{i,1\}}\right)\right) \\ &+ \sum_{z=1}^{h_i^{r_i}-1}\left(\prod_{n=1}^{z}\delta_{\{i,n-1\},\{i,n\}} \sum_{\{i,n\}\in\mathcal{N}^-(\{i,n-1\})} F^-_{\{i,n-1\},\{i,n\}}\right) \\ &\times \left(\left(BP_{\{i,z\}} y^{r_{\{i,z\}}}\right)_{\{i,z\}} + \sum_{\{i,z+1\}\in\mathcal{N}^-(\{i,z\})} F^-_{\{i,z\},\{i,z+1\}} y^{r_{\{i,z+1\}}}_{\{i,z+1\}}\right),\end{aligned}$$

we supposed that $K_{\{i,0\}} \equiv K_j$, $j \in \mathcal{N}(i)$ are the neighbors of K_i. □

To study the convergence rate of the asynchronous scheme, we first need to suppose that the solution is synchronized at the instants t_s which are multiples of a fixed time step $\Delta t_{\max}$. In the case of meshes where all the elements have a common meeting time, the solution is automatically synchronized at the instants which are multiples of the lowest common multiple (LCM) of all the local time steps and then $\Delta t_{\max}$ is equal to the LCM. If it is not the case, so that the local time steps are completely independent, we have to suppose that the solution is regularly synchronized at the instants multiples of $\Delta t_{\max}$ where, in this case, $\Delta t_{\max}$ is computed in function of the time steps Δt_i. For instance, we can choose $\Delta t_{\max} = \max_i(\Delta t_i)$. Note that the hypothesis of regular synchronization of the solution has no influence on the asynchronous aspect; see Fig. 3. In fact, the cells advance independently between t^q and t^{q+1}. It is important to note that

the regular synchronization of the solution was imposed only for demonstration purposes. Numerically, the synchronization is only made at the final time of the simulation independently of the mesh; see the ARK2 algorithm. Let $(t^q)_{q\geq 0}$ be the sequence defined by $t^{q+1} = t^q + \Delta t_{\max}$. The study of the convergence of the ARK2 scheme will be done between two instants of synchronization: t^q and t^{q+1}. Let $(t_q^n)_{1\leq n\leq N_q}$ be the sequence of the most urgent refresh time tags between t^q and t^{q+1}, where $N_q = \sum_i s_i$ and for all i, $s_i = \left\lfloor \frac{\Delta t_{\max}}{\Delta t_i} \right\rfloor$. $(t_{i,q}^k)_k$ is the sequence of the temporal evolution of the element i between t^q and t^{q+1} defined by $t_{i,q}^k = t^q + k\Delta t_i$ (Fig. 3).

Theorem 1. *For all the elements i and at any instant $t_{i,q}^{r_i} = t^q + r_i\Delta t_i$ such that $t^q < t_{i,q}^{r_i} \leq t^{q+1}$, the following expression holds true*

$$y_i^{t_{i,q}^{r_i}} = y_i^{t^q} + (r_i\Delta t_i)(By^{t^q})_i + \frac{(r_i\Delta t_i)^2}{2}(B^2y^{t^q})_i \tag{6}$$

$$+ \sum_{n=3}^{2r_i} (\Delta t_i)^n \alpha_{n,r_i}(BP_i)^{n-2}(B^2y^{t^q})_i + R_{i,asyn}^{r_i}, \tag{7}$$

where: $\alpha_{n,0} = 0\ \forall n \geq 3$, $\alpha_{3,r_i} = \alpha_{3,r_i-1} + \frac{(r_i-1)^2+(r_i-1)}{2}$, $\alpha_{4,r_i} = \alpha_{3,r_i-1} + \alpha_{4,r_i-1} + \frac{(r_i-1)^2}{2}$, $\alpha_{n,r_i} = \alpha_{n,r_i-1} + \alpha_{n-1,r_i-1} + \frac{\alpha_{n-2,r_i-1}}{2}$ $\forall 5 \leq i \leq 2(r_i-1)$, $\alpha_{2r_i-1,r_i} = \alpha_{2(r_i-1),r_i-1} + \frac{\alpha_{2(r_i-1)-1,r_i-1}}{2}$, $\alpha_{2r_i,r_i} = \frac{\alpha_{2(r_i-1),r_i-1}}{2}$, *and*

$$\|R_{i,asyn}^{r_i}\|_\infty \leq \sum_{n=3}^{M_i^{r_i}} C_{n,r_i}(\Delta t_i)^n \|B^n\|_\infty \|y^{t^q}\|_\infty \tag{8}$$

$M_i^{r_i} = \max(2r_i, h_i^{r_i})$ *and for all* $3 \leq n \leq M_i^{r_i}$, C_{n,r_i} *is a bounded constant.*

Proof. The proof of Theorem 1 is very technical and many formulas are long, it is then placed in the appendix. □

Theorem 2. *The asynchronous Runge–Kutta 2 scheme is a second order accurate scheme.*

Proof. Let $y_i(t^{q+1})$ be the semi-discrete exact solution of the ODE (3) at t^{q+1}, $y_{i,syn}^{t^{q+1}}$ is the solution of the classical global time step Runge–Kutta 2 method at t^{q+1} and $y_i^{t^{q+1}}$ the asynchronous solution. Then,

$$\|y_i(t^{q+1}) - y_i^{t^{q+1}}\|_\infty \leq \|y_i(t^{q+1}) - y_{i,syn}^{t^{q+1}}\|_\infty + \|y_{i,syn}^{t^{q+1}} - y_i^{t^{q+1}}\|_\infty \tag{9}$$

The classical global time step RK2 scheme is second order convergent then,

$$\|y_i(t^{q+1}) - y_{i,syn}^{t^{q+1}}\|_\infty \leq C_0(\Delta t_{\max})^3,$$

where, for $\Delta t_{\max}$ sufficiently small, C_0 is a constant independent of $\Delta t_{\max}$. To complete the proof we thus need to treat the second term on the right of (9). First, using Theorem 1, we get for all i

$$y_i^{t_{q,i}^{s_i}} = y_i^{t^q} + (s_i\Delta t_i)(By^{t^q})_i + \frac{(s_i\Delta t_i)^2}{2}(B^2y^{t^q})_i + \sum_{n=3}^{2s_i}(\Delta t_i)^n\alpha_{n,s_i}(BP_i)^{n-2}(B^2y^{t^q})_i + R_{i,\text{asyn}}^{t_{q,i}^{s_i}}$$

where for all i, $t_{q,i}^{s_i} = t^q + s_i\Delta t_i$, $s_i = E\left(\frac{\Delta t_{\max}}{\Delta t_i}\right)$. As a first step, suppose that $\frac{\Delta t_{\max}}{\Delta t_i}$ is an integer. As a result,

$$y_i^{t^{q+1}} \equiv y_i^{t_{q,i}^{s_i}} = y_i^{t^q} + (\Delta t_{\max})(By^{t^q})_i + \frac{(\Delta t_{\max})^2}{2}(B^2y^{t^q})_i + \sum_{n=3}^{2s_i}(\Delta t_i)^n\alpha_{n,s_i}(BP_i)^{n-2}(B^2y^{t^q})_i + R_{i,\text{asyn}}^{t^{q+1}}$$

Consequently,

$$\|y_{i,syn}^{t^{q+1}} - y_i^{t^{q+1}}\|_\infty = \|E_i^{t^{q+1}}\|_\infty,$$

where

$$E_i^{t^{q+1}} = \sum_{n=3}^{2s_i}(\Delta t_i)^n\alpha_{n,s_i}B^{n-2}(B^2y^{t^q})_i - \sum_{n=3}^{2s_i}(\Delta t_i)^n\alpha_{n,s_i}(BP_i)^{n-2}(B^2y^{t^q})_i - R_{i,\text{asyn}}^{t^{q+1}}$$

$$= \sum_{n=3}^{2s_i}(\Delta t_i)^n\alpha_{n,s_i}\left[\sum_{j\in\mathcal{N}^-(i)}F_{i,j}\right]^{n-2}(B^2y^{t^q})_j - R_{i,\text{asyn}}^{t^{q+1}}, \quad B \text{ is linear.}$$

Therefore,

$$\|E_i^{t^{q+1}}\|_\infty \leq C_1(\Delta t_{\max})^3, \tag{10}$$

where for $\Delta t_{\max}$ sufficiently small, C_1 is a constant independent of $\Delta t_{\max}$. Finally, note that for the case where $\frac{\Delta t_{\max}}{\Delta t_i}$ is not an integer we just need to synchronize the solution at t^{q+1} using a second order interpolation as it was explained when the ARK2 algorithm has been presented. □

3.2. Comparison with the LTS-RK2 scheme presented in [4]

In [4], Grote et al. derived an explicit LTS-RK scheme from standard explicit Runge–Kutta (RK) methods in the case of two distinct regions in the mesh: a "coarse" region with the larger elements and a "fine" region with the smaller elements. In the fine region, the time step must be a fraction of the coarse time step. They use the classical RK method of a given order inside the fine region and quadrature formula inside the coarse region. The intermediate values needed at the coarse and fine mesh interface are obtained through a judicious combination of interpolation and Taylor expansion. They have proved that their scheme preserves the accuracy of the original RK scheme. We started from their LTS-RK scheme and we adapted the algorithm presented in [4] to the asynchronous aspect in order to apply the three main phases of the asynchronous integration. In particular, each cell is treated independently. In this paper, we only focus on order two. In the case of the mesh imposed in [4], we proved that the ARK2 degenerates into the LTS-RK2 scheme. More precisely, using the asynchronous algorithm, we get the same result at each global time step Δt, where Δt denote the time step dictated by the CFL-condition in the coarser part of the mesh. Whereas, at each time step of size $n\Delta t/p$ where $\Delta t/p$ is the local time step inside the refined region of the mesh with p and n are integers and $0 < n < p$, the result is different. In fact, we have the following result

Proposition 4. *In the case of two distinct regions in the mesh where the time step of the fine region is a fraction of the time step of the coarse region, the ARK2 scheme degenerates into the LTS-RK2 scheme presented in [4].*

Proof. Resuming the notation used in [4]: $\Omega = \Omega_c \cup \Omega_f$ where Ω_c is the coarse region with the large time step Δt_c and Ω_f is the fine region with the small time step $\Delta t_f = \frac{\Delta t_c}{p}$, p is an integer. Introduce some elements for the identification

$$\begin{aligned}
(Bx)_i &= V_i x_i + \sum_{j\in\mathcal{N}^+(i)} F^+_{i,j}x_i + \sum_{j\in\mathcal{N}^-(i)} F^-_{i,j}x_j \\
(BPx)_i &= V_i x_i + \sum_{j\in\mathcal{N}^+(i)} F^+_{i,j}x_i + \sum_{j\in\mathcal{N}^-_f(i)} F^-_{i,j}x_j \\
(B(I-P)x)_i &= \sum_{j\in\mathcal{N}^-_c(i)} F^-_{i,j}x_j
\end{aligned}$$

where $\mathcal{N}_c(i)$ and $\mathcal{N}_f(i)$ are the neighbors of the element i which are respectively in the coarse mesh and the fine mesh. In other words, $j \in \mathcal{N}_c(i)$ then $j \in \mathcal{N}(i) \cap \Omega_c$ and $j \in \mathcal{N}_f(i)$ then $j \in \mathcal{N}(i) \cap \Omega_f$. P is a diagonal matrix, its diagonal entries are equal to zero or one. It identify the unknowns associated with the locally refined region. $(I - P)$ identify the unknowns associated with the coarse region. The previous results are proved for any given of mesh, in particular for a mesh with two regions. In fact, using the previous notations we have $\Delta t_{\max} = \Delta t_c$. This is a natural choice because the solution is regularly synchronized at the instants multiples of Δt_c. Then the sequence $(t^q)_q$ is defined by $t^{q+1} = t^q + \Delta t_c$, for all $i \in \Omega_f$, $(t^k_{i,q})_{0\leq k\leq p}$ is defined by $t^k_{i,q} = k\Delta t_f$ and for all $j \in \Omega_c$, $(t^k_{j,q})_{0\leq k\leq 1}$ is defined by $t^0_{j,q} = t^q$ and $t^1_{j,q} = t^{q+1}$. Let us use the previous results of Proposition 1 and Proposition 2 for a fine element, the proof for the coarse elements can be done in the same way. For all $i \in \Omega_f$ we have

$$\begin{aligned}
y_i^{t^{r_i+1}_{i,q}} &= y_i^{t^{r_i}_{i,q}} + \Delta t_f \left(BPy^{t^{r_i}_{i,q}}\right)_i + \frac{(\Delta t_f)^2}{2}\left(BP\left(\frac{\Delta y}{\Delta t}\right)^{t^{r_i}_{i,q}}\right)_i \\
&\quad + \Delta t_f \left(B(I-P)y^{t^q}\right)_i + \frac{(2r_i+1)(\Delta t_f)^2}{2}\left(B(I-P)\left(\frac{\Delta y}{\Delta t}\right)^{t^q}\right)_i,
\end{aligned}$$

with

$$\left(\frac{\Delta y}{\Delta t}\right)_i^{t^{r_i}_{i,q}} = \left(BPy^{t^{r_i}_{i,q}}\right)_i + \left(B(I-P)\left(y^{t^q} + r_i\Delta t_f\left(\frac{\Delta y}{\Delta t}\right)^{t^q}\right)\right)_i$$

Comparing with the expression of the slope given in Proposition 3, the dependencies with the neighbors are simplified. In fact, in this particular case, there are only two types of elements the fine cells which are grouped in the matrix P and the coarse cells which are grouped in the matrix $(I-P)$. All the fine elements has been updated in the same times and it is also the case for the coarse elements. Consequently, the expression of y_i (6), i a fine element, in Theorem 1 is written in the following form

$$y_i^{t_{i,q}^{r_i}} = y_i^{t^q} + (r_i\Delta t_f)(By^{t^q})_i + \frac{(r_i\Delta t_f)^2}{2}(B^2y^{t^q})_i + \sum_{m=3}^{2r_i}(\Delta t_f)^m\alpha_{m,r_i}(BP_i)^{m-2}(B^2y^{t^q})_i,$$

and then we find the same expression given in [4]. □

3.3. Originality of the asynchronous formulation in ARK2

The originality of our approach with respect to the LTS scheme developed in [4], and more generally with respect to any other standard local time-stepping approach, lies within its ability to adapt to any arbitrary mesh and its resilience for being second order convergent scheme independently of the chosen mesh. Indeed, for the standard LTS schemes, local time steps are usually selected to be fractions of the global time step whereas the asynchronous scheme permits the selection of independent time steps in each mesh element.

However, as mentioned in the introduction, the idea of using a different time-step in every element already exists in the literature. In [11], Omelchenko and Karimabadi introduced such an asynchronous scheme for drift-diffusion transport. The major difference between their asynchronous approach and the one presented here lies within the way the fluxes are updated. In [11], the concept of "flux capacitor", a variable in which the flux between two cells is stored, is used. In our approach, desynchronized fluxes can be used which allows fluxes recomputations in order to ensure the second-order convergence.

In [18], A. Taube et al. proposed an ADER-DG scheme with independent time steps for the Maxwell equations with arbitrary high order of accuracy in space and time, although no proof of the convergence rate is given. Their method consists in replacing the time derivatives with space derivatives. The computation of the flux between the grid cells is then based on the solution of generalized Riemann problems. Our asynchronous method is different, since it is derived from the standard explicit Runge–Kutta 2 method and thus the numerical solution needs to go through intermediate stages in order to be advanced in time.

4. Numerical tests

We consider the two-dimensional Maxwell equations in transverse electric mode:

$$\begin{cases} \frac{\partial}{\partial t}E_x &= \frac{1}{\epsilon}\frac{\partial}{\partial y}H_z \quad \text{in} \quad \Omega\times(0,T) \\ \frac{\partial}{\partial t}E_y &= -\frac{1}{\epsilon}\frac{\partial}{\partial x}H_z \quad \text{in} \quad \Omega\times(0,T) \\ \frac{\partial}{\partial t}H_z &= \frac{1}{\mu}\frac{\partial}{\partial y}E_x - \frac{1}{\mu}\frac{\partial}{\partial x}E_y \quad \text{in} \quad \Omega\times(0,T) \end{cases} \tag{11}$$

where $E=\left(E_x,E_y,0\right)^T$ and $H=(0,0,H_z)^T$ are respectively the electric and magnetic fields. ϵ and μ are, respectively, the electric permittivity and magnetic permeability. Ω is the computational domain and T is the final time of the simulation. For the spatial discretization, we apply the second order P^1-discontinuous Galerkin scheme with upwind fluxes.

4.1. Numerical convergence of the ARK2 scheme

We shall now present numerical experiments which validate the order of convergence of the ARK2 method derived in section 3. The one-dimensional tests assessing the second order convergence of the ARK2 scheme have already been published in [20]. In this paper, we consider the two-dimensional Maxwell equations (11). The initial condition: $(E_x)_0 = (E_y)_0 = (H_z)_0 = cos(2\pi x)$, Ω is a rectangular domain of size $[0,1]\times[0,1]$ and the boundary conditions are supposed periodic.

First, we divide $[0,1]$ in X-direction, respectively in Y-direction, into tree equal parts. The left and right intervals are discretized with an equidistant mesh of size Δx, respectively Δy. The middle interval is discretized with an equidistant mesh of size $p\Delta x/q$, respectively $p\Delta y/q$ where p and q are integers; see also Fig. 4 left. We start with a simple case, mesh 1, where the local time steps are fractions of the global time steps with $p=4$ and $q=1$. Then for a more general case, mesh 2, we suppose that $p=2$, $q=3$ and then $p/q<1$. Finally, mesh 3 is to test the ARK2 algorithm in the case where the time steps are independent, we suppose that the positions x_i and y_i follow a polynomial low and thus

$$x_i = 4(1-\alpha)\left(\frac{i-1}{N_x-1}-\frac{1}{2}\right)^3 + \alpha\left(\frac{i-1}{N_x-1}-\frac{1}{2}\right)+\frac{1}{2}, \tag{12}$$

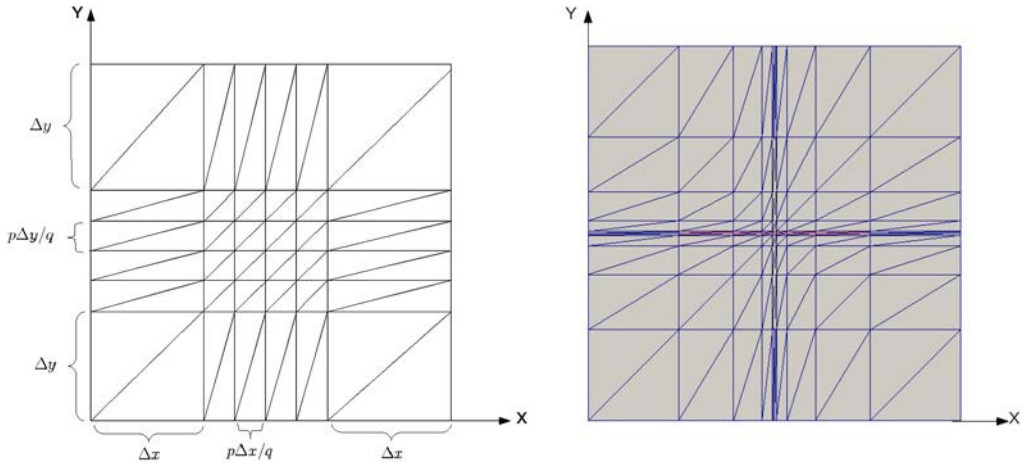

Fig. 4. The computational domain $\Omega = [0, 1] \times [0, 1]$ with the refined region in the middle (left) and with polynomial mesh in X and Y directions; example $N_x = N_y = 10$ and $\alpha = 0.01$ (right). Each rectangle is then splitted into two triangles.

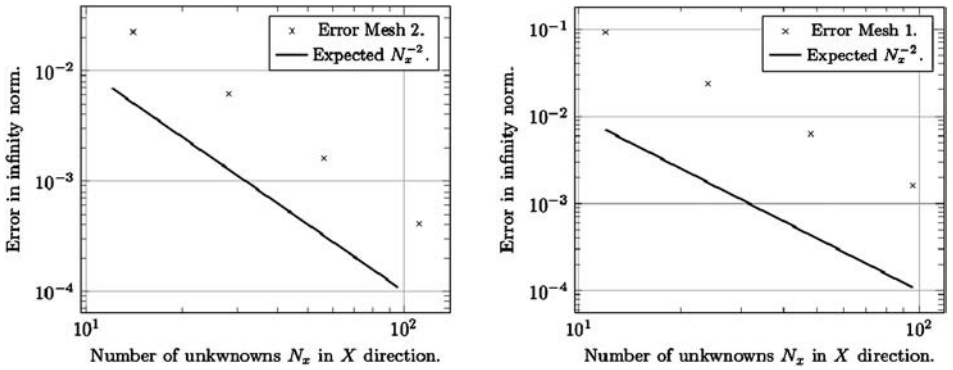

Fig. 5. Second order ARK2 scheme for mesh 1 and mesh 2.

and

$$y_i = 4(1-\alpha)\left(\frac{i-1}{N_y-1} - \frac{1}{2}\right)^3 + \alpha\left(\frac{i-1}{N_y-1} - \frac{1}{2}\right) + \frac{1}{2}, \tag{13}$$

where N_x, respectively N_y, is the number of cells in X-direction, respectively in Y-direction; see also Fig. 4 right. The parameter α define the slope at $x = 1/2$ and $y = 1/2$. For each mesh, each rectangle is splitted into two triangles.

For each mesh, we plot the infinity norm of the error versus various number of cells in X direction in logarithmic scale. Note that for the three treated meshes the number of cells in X direction and the number of cells in Y direction were chosen to be equal. We present a numerical study of the convergence rate of the asynchronous scheme in the case of mesh 1 in Fig. 5 left, in the case of mesh 2 in Fig. 5 right and in the case of mesh 3 in Fig. 6.

4.2. Effective gain in computational cost

The asynchronous method has been developed in order to reduce the computation cost in the case of multi-scale problems. To study the performance of our asynchronous method we consider the two-dimensional Maxwell equations (11) with null initial conditions and for the boundary conditions we inject by one of the sides of the computational domain Ω, a plane wave with the equation:

$$u(x, t) = p_1 \sin(2\pi \times p_2 \times (t - x/c)), \tag{14}$$

where $p_1 = 1$ and $p_2 = 10^9$ are respectively the amplitude and the frequency of the wave and $c = 3 \times 10^8$ m/s is the speed of light. $\lambda = c \times T$ is the wavelength and $T = 10^{-9}$ s is the final simulation time. Ω is a rectangular domain of size $[0, \lambda] \times [0, \lambda]$. To built a multi-scale mesh, we add a circle inside Ω; see Fig. 7 left. (the circle represents a two-dimensional projection of a cylindrical electric wire). The circle is of radius $r = 0.001 \times \lambda << \lambda$. The multi-scale mesh produced by GMSH contains 8416 triangles with $A_{max} = 6.5 \times 10^{-5}$ and $A_{min} = 2.5 \times 10^{-9}$ where A_{max} (resp. A_{min}) is the area of the

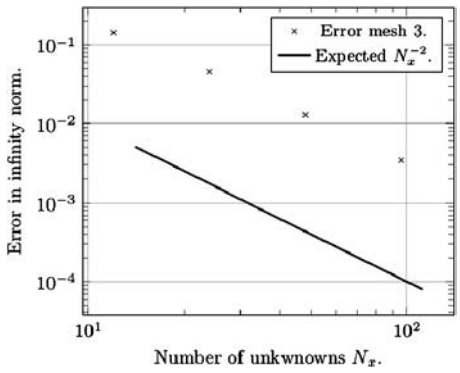

Fig. 6. Second order ARK2 scheme for the polynomial mesh where the parameter $\alpha = 0.01$.

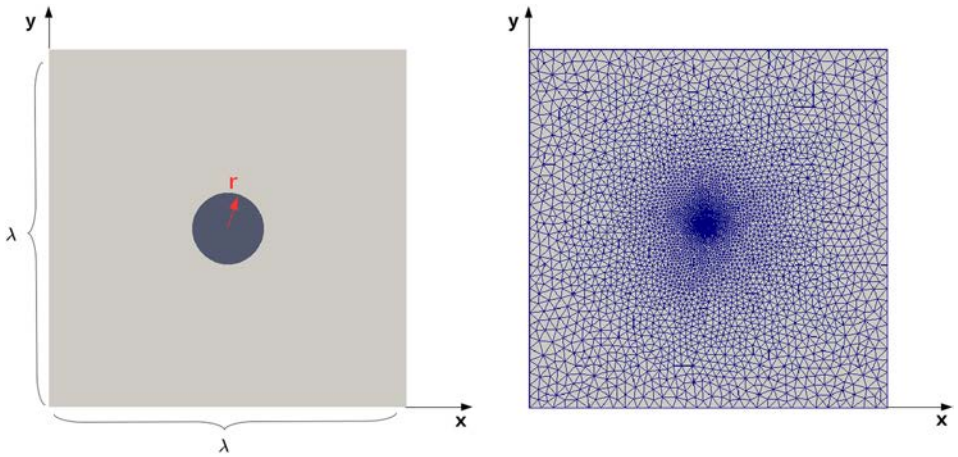

Fig. 7. The computational domain $\Omega = [0, \lambda] \times [0, \lambda]$ with circle in the middle of radius r (left) and the multi-scale mesh of Ω (right).

largest (resp. smallest) triangle; see Fig. 7 right. The expected factor of acceleration is 18. Numerically, the speed-up is equal to 17.8 with 1 hour and 40 minutes for the global time step RK2 scheme and only 5 minutes for our asynchronous integration.

The theoretical expected speed-up is an a priori estimate on the time reduction factor which depends only on the mesh. As expected, the real asynchronous speed-up is lower than the theoretical one mainly because of two reason. The first is that the asynchronous scheme access the data in memory in a more random way, yielding a higher number of "cache-miss" within the processor. The second is that the algorithm may require extra computations in order to ensure the second-order accuracy. This computational overhead depends on the implementation and on the test-case, nonetheless experiments conducted in both 2D and 3D on various test-case showed that the ratio between the real and the optimal speed-ups lies within a range from 1.5 to 2 [19]. This means that for application with a theoretical speed-up exceeding 3, the asynchronous integration practically provides an effective CPU speed-up.

4.3. Strategy for an asynchronous ARK2 parallel computation

The numerical example presented in the previous section shows the efficiency (in terms of computational time) of the asynchronous approach with respect to the global time stepping method. However, it has to be noted that it has been performed in a sequential approach. Conversely, given a sequential algorithm, many strategies for parallel computing have been developed in order to gain computational time. While the focus of this paper is on the theoretical order of accuracy for the ARK2 algorithm, we will present here a first insight on the possible strategies for a parallelization of the ARK2 algorithm, allowing for further computational time saving.

Deriving a parallel approach for the asynchronous integration method has first been studied in the context of a first-order Lax–Wendroff asynchronous time integration algorithm [5]. In this section, we will focus on its application to the ARK2 method, following the openMP memory paradigm. Still, the ARK2 method is also compatible with distributed memory strategy introduced in [5]. For shared memory, the key idea is to have as many DTS as openMP threads. The master thread is the master of time and is the only allowed to change the simulation time. All threads (master and slaves) have an action

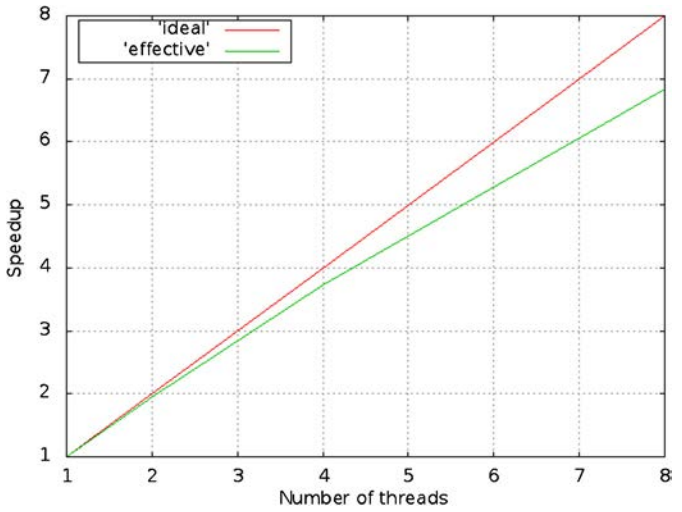

Fig. 8. The speed-up versus the number of threads. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

list to execute. In this context, an action corresponds to the refreshment of one cell of the mesh (Steps 1 to 4 in the ARK2 algorithm, see section 2.2).

Thread safety is ensured since each action belongs to a single thread and is therefore rescheduled within its own DTS. Data safety is also ensured because the algorithm has been split in two parts. The first part consists in computing intermediate variables such as interface fluxes or source terms and generating an "integration list", which is the lists of all cells for which the time derivatives have to be recomputed at the current time. The second part consists in integrating the equations in the dedicated cells and recomputing the time derivatives using the conservation equation (flux balance and source term) according to the integration list. This list is then split within the threads. Hence, date safety is guaranteed provided that a single cell appears at most once in the integration list. Indeed, a thread may access data from a cell and from its neighbors but the writing occurs only within the considered cell.

The difference with [5] lies within the fact that for first order time integration, the action list is treated then rescheduled within the DTS. For ARK2, the action list is treated once to compute the new mid-point slope but is not rescheduled and is treated a second time instead. An action can be treated the second time provided that for each cell, the mid-point slopes are already available from the neighbor cells (from previous simulation times or from another thread or from the thread itself).

A multi-threaded scaling speed-up test has been performed on a single node of the Eos supercomputer at Calmip [6]. The test case, which is strongly asynchronous, is the same as the one described on Fig. 4 (left), with a 101×101 mesh, generated using the polynomial law with $\alpha = 0.01$. The speed-up obtained on one node, with thread numbers ranging from 1 to 8, is represented on Fig. 8 and shows that the openMP parallel version of the asynchronous algorithm has a good performance with respect to the ideal scaling. However, the used parallel implementation is only a first step and there is still room for improvements. A much more detailed study of the parallel paradigm for the ARK2 algorithm as well as in-depth performance analysis for both distributed and shared memory parallelization still remains to be made, this will be the subject of future works.

5. Conclusion

In the present paper, we derived an asynchronous RK2 scheme from the standard Runge–Kutta scheme which permits the selection of independent time steps in each mesh element. We presented the ARK2 algorithm to discretize a given linear partial differential equation with an arbitrary space discretization. At each time step, only one cell is involved and only the neighbors need to be updated, which makes the method easily applicable to any arbitrary mesh. We proved that the ARK2 scheme is a second order convergent scheme and thus it preserves the accuracy of the original RK2 scheme. Numerical experiments assess the convergence rate of the ARK2 scheme in one dimensional case, have already been published in [20]. In this paper we focused in two dimensional tests. The numerical study shows that the ARK2 scheme is second order convergent independently of the chosen mesh, which validates the theoretical results. Moreover, the numerical tests indicate that the use of the optimal CFL condition imposed on each cell independently of other cells, does not impact the accuracy and efficiency of the classical RK2 scheme. Furthermore, the results demonstrate that, comparing with the classical integration, the asynchronous scheme is effective in terms of computation time. In this paper, a simple academic two-dimensional test has been treated. More representative physical tests that validate the gain in computation time may be found in [19]. To increase the speedup efficiency of the asynchronous integration, a parallelization strategy using shared memory paradigm has been explained.

For the sake of simplicity, we imposed the linearity of the treated partial differential equation. The asynchronous method is likely to prove useful for non-linear systems. In the present work, we only focused on order two, mainly for reasons of readability and clarity. Extension of the algorithm in the case of ARKp ($p > 2$) will be the subject of future works.

Appendix A. Proof of Theorem 1

Proof. The proof is done by using the mathematical induction. We suppose that the formula (6) is true for $t_q^n < t^{q+1}$ and that $t_q^{n+1} = t_{i,q}^{r_i+1}$. Then according to Proposition 1 we have

$$\begin{aligned}
y_i^{t_{i,q}^{r_i+1}} &= y_i^{t_{i,q}^{r_i}} + \sum_{k=1}^{m+1} \delta_i^k \left[V_i \left(y_i^{t_{i,q}^{r_i}} + \frac{\Delta t_i}{2} \left(\frac{\Delta y}{\Delta t} \right)_i^{t_{i,q}^{r_i}} \right) \right] \\
&+ \sum_{k=1}^{m+1} \delta_i^k \left[\sum_{j \in \mathcal{N}^+(i)} F_{i,j}^+ \left(y_i^{t_{i,q}^{r_i}} + \left(t_{\mathrm{mid},i,j}^k - r_i \Delta t_i \right) \left(\frac{\Delta y}{\Delta t} \right)_i^{t_{i,q}^{r_i}} \right) \right] \\
&+ \sum_{k=1}^{m+1} \delta_i^k \left[\sum_{j \in \mathcal{N}^-(i)} F_{i,j}^- \left(y_j^{t_{j,q}^{r_j}} + \left(t_{\mathrm{mid},i,j}^k - r_j^k \Delta t_j \right) \left(\frac{\Delta y}{\Delta t} \right)_j^{t_{j,q}^{r_j}} \right) \right]
\end{aligned}$$

$y_i^{t_{i,q}^{r_i+1}}$ is then given by:

$$\begin{aligned}
y_i^{t_{i,q}^{r_i+1}} &= y_i^{t_{i,q}^{r_i}} + \Delta t_i \left(V_i y_i^{t_{i,q}^{r_i}} + \sum_{j \in \mathcal{N}^+(i)} F_{i,j}^+ y_i^{t_{i,q}^{r_i}} \right) + \frac{(\Delta t_i)^2}{2} V_i \left(\frac{\Delta y}{\Delta t} \right)_i^{t_{i,q}^{r_i}} \\
&+ \left[\sum_{j \in \mathcal{N}^+(i)} F_{i,j}^+ \left(\sum_{k=1}^{m+1} \delta_i^k \times t_{\mathrm{mid},i,j}^k - r_i (\Delta t_i)^2 \right) \left(\frac{\Delta y}{\Delta t} \right)_i^{t_{i,q}^{r_i}} \right] \\
&+ \sum_{k=1}^{m+1} \delta_i^k \left[\sum_{j \in \mathcal{N}^-(i)} F_{i,j}^- \left(y_j^{t_{j,q}^{r_j}} + \left(t_{\mathrm{mid},i,j}^k - r_j^k \Delta t_j \right) \left(\frac{\Delta y}{\Delta t} \right)_j^{t_{j,q}^{r_j}} \right) \right].
\end{aligned}$$

According to Proposition 2, we have $\sum_{k=1}^{m+1} \delta_i^k \times t_{\mathrm{mid},i,j}^k = \frac{(2r_i+1)(\Delta t_i)^2}{2}, \forall j \in \mathcal{N}(i)$. As a result, $\sum_{k=1}^{m+1} \delta_i^k \times t_{\mathrm{mid},i,j}^k - r_i(\Delta t_i)^2 = \frac{(\Delta t_i)^2}{2}, \forall j \in \mathcal{N}(i)$. Then,

$$\begin{aligned}
y_i^{t_{i,q}^{r_i+1}} &= y_i^{t_{i,q}^{r_i}} + \Delta t_i \left(BP_i y^{t_{i,q}^{r_i}} \right)_i + \frac{(\Delta t_i)^2}{2} BP_i \left(\frac{\Delta y}{\Delta t} \right)_i^{t_{i,q}^{r_i}} && (15) \\
&+ \sum_{k=1}^{m+1} \delta_i^k \left[\sum_{j \in \mathcal{N}^-(i)} F_{i,j}^- \left(y_j^{t_{j,q}^{r_j}} + \left(t_{\mathrm{mid},i,j}^k - r_j^k \Delta t_j \right) \left(\frac{\Delta y}{\Delta t} \right)_j^{t_{j,q}^{r_j}} \right) \right]. && (16)
\end{aligned}$$

Now we use the Proposition 3 and the induction hypothesis to get the following expression of the slope i

$$\left(\frac{\Delta y}{\Delta t} \right)_i^{t_{i,q}^{r_i}} = (By^{t^q})_i + (r_i \Delta t_i)(B^2 y^{t^q})_i + \frac{(r_i \Delta t_i)^2}{2} BP_i (B^2 y^{t^q})_i + \sum_{n=3}^{2r_i} (\Delta t_i)^n \alpha_{n,r_i} (BP_i)^{n-1} (B^2 y^{t^q})_i + \tilde{R}_{i,\mathrm{asyn}}^{t_{i,q}^{r_i}},$$

where

$$\begin{aligned}
\tilde{R}_{i,\mathrm{asyn}}^{t_{i,q}^{r_i}} &= (SR)_i^{t_{i,q}^{r_i}} + (RJ)_i^{t_{i,q}^{r_i}} + (RZ)_i^{t_{i,q}^{r_i}} \\
(SR)_i^{t_{i,q}^{r_i}} &= BP_i R_{i,\mathrm{asyn}}^{t_{i,q}^{r_i}} + \sum_{j \in \mathcal{N}^-(i)} F_{i,j}^- \left[R_{j,\mathrm{asyn}}^{t_{j,q}^{r_j}} + \delta_{i,j} \left[BP_j R_{j,\mathrm{asyn}}^{t_{j,q}^{r_j}} + \sum_{\{i,1\} \in \mathcal{N}^-(j)} F_{\{i,1\},j}^- R_{\{i,1\},\mathrm{asyn}}^{t_{\{i,1\},q}^{r_{\{i,1\}}}} \right] \right] \\
&+ \sum_{z=1}^{h_{i,t^q}^{r_i}-1} \left(\prod_{n=1}^{z} \delta_{\{i,n-1\},\{i,n\}} \sum_{\{i,n\} \in \mathcal{N}^-(\{i,n-1\})} F_{\{i,n-1\},\{i,n\}}^- \right) \\
&\times \left[BP_{\{i,z\}} R_{\{i,z\},\mathrm{asyn}}^{t_{\{i,z\},q}^{r_{\{i,z\}}}} + \sum_{\{i,z+1\} \in \mathcal{N}^-(\{i,z\})} F_{\{i,z\},\{i,z+1\}}^- R_{\{i,z+1\},\mathrm{asyn}}^{t_{\{i,z+1\},q}^{r_{\{i,z+1\}}}} \right].
\end{aligned}$$

$$(RJ)_i^{t_{i,q}^{r_i}} = \sum_{j\in\mathcal{N}^-(i)} F_{i,j}^- \Bigg[\Bigg[\frac{(r_j\Delta t_j)^2}{2}(B^2y^{t^q})_j \sum_{n=3}^{2r_j}(\Delta t_j)^n\alpha_{n,r_j}(BP_j)^{n-2}(B^2y^{t^q})_j\Bigg]$$

$$+\delta_{i,j}\Bigg[BP_j\Bigg[(r_j\Delta t_j)(By^{t^q})_j + \frac{(r_j\Delta t_j)^2}{2}(B^2y^{t^q})_j$$

$$+\sum_{n=3}^{2r_j}(\Delta t_j)^n\alpha_{n,r_j}(BP_j)^{n-2}(B^2y^{t^q})_j\Bigg] + \sum_{\{i,1\}\in\mathcal{N}^-(j)} F_{\{i,1\},j}^-\Bigg[r_{\{i,1\}}\Delta t_{r_{\{i,1\}}}(By^{t^q})_{\{i,1\}}$$

$$+\frac{(r_{\{i,1\}}\Delta t_i)^2}{2}(B^2y^{t^q})_{\{i,1\}} + \sum_{n=3}^{2r_{\{i,1\}}}(\Delta t_{r_{\{i,1\}}})^n\alpha_{n,r_{\{i,1\}}}(BP_{\{i,1\}})^{n-2}(B^2y^{t^q})_{\{i,1\}}\Bigg]\Bigg]\Bigg]$$

and

$$(RZ)_i^{t_{i,q}^{r_i}} = \sum_{z=1}^{h^{t_{i,q}^{r_i}}-1}\left(\prod_{n=1}^{z}\delta_{\{i,n-1\},\{i,n\}}\sum_{\{i,n\}\in\mathcal{N}^-(\{i,n-1\})}F_{\{i,n-1\},\{i,n\}}^-\right)\times$$

$$\Bigg[BP_{\{i,z\}}\Bigg[y_{\{i,z\}}^{t^q} + (r_{\{i,z\}}\Delta t_{r_{\{i,z\}}})(By^{t^q})_{\{i,z\}} + \frac{(r_{\{i,z\}}\Delta t_i)^2}{2}(B^2y^{t^q})_{\{i,z\}} + \sum_{n=3}^{2r_{\{i,z\}}}(\Delta t_{r_{\{i,z\}}})^n\alpha_{n,r_{\{i,z\}}}(BP_{\{i,z\}})^{n-2}(B^2y^{t^q})_{\{i,z\}}\Bigg]$$

$$+\sum_{\{i,z+1\}\in\mathcal{N}^-(\{i,z\})}F_{\{i,z\},\{i,z+1\}}^-\Bigg[y_{\{i,z+1\}}^{t^q} + (r_{\{i,z+1\}}\Delta t_{r_{\{i,z+1\}}})(By^{t^q})_{\{i,z+1\}} + \frac{(r_{\{i,z+1\}}\Delta t_i)^2}{2}(B^2y^{t^q})_{\{i,z+1\}}$$

$$+\sum_{n=3}^{2r_{\{i,z+1\}}}(\Delta t_{r_{\{i,z+1\}}})^n\alpha_{n,r_{\{i,z+1\}}}(BP_{\{i,z+1\}})^{n-2}(B^2y^{t^q})_{\{i,z+1\}}\Bigg]\Bigg]$$

$(SR)_i^{t_{i,q}^{r_i}}$ is a function of the asynchronous rests, $(RJ)_i^{t_{i,q}^{r_i}}$ is the rest coming from the neighbors j where $j\in\mathcal{N}^-(i)$ and $(RZ)_i^{t_{i,q}^{r_i}}$ is the rest coming from the dependency of the element i on the other elements of the mesh.

Now, we replace $\left(\frac{\Delta y}{\Delta t}\right)_i^{r_i}$ in the formula of $y_i^{r_i+1}$ given by the equation (15) and we use the induction hypothesis to replace the values of $y_i^{r_i}$ and $y_j^{r_j}$ for all $j\in\mathcal{N}(i)$. Which gives the following result

$$y_i^{t_{i,q}^{r_i+1}} = \Bigg[y_i^{t^q} + (r_i\Delta t_i)(By^{t^q})_i + \frac{(r_i\Delta t_i)^2}{2}(B^2y^{t^q})_i$$

$$+\sum_{n=3}^{2r_i}(\Delta t_i)^n\alpha_{n,r_i}(BP_i)^{n-2}(B^2y^{t^q})_i + R_{i,\text{asyn}}^{t_{i,q}^{r_i}}\Bigg] + \Delta t_i BP_i\Bigg[y_i^{t^q} + (r_i\Delta t_i)(By^{t^q})_i$$

$$+\frac{(r_i\Delta t_i)^2}{2}(B^2y^{t^q})_i + \sum_{n=3}^{2r_i}(\Delta t_i)^n\alpha_{n,r_i}(BP_i)^{n-2}(B^2y^{t^q})_i + R_{i,\text{asyn}}^{t_{i,q}^{r_i}}\Bigg]$$

$$+\frac{(\Delta t_i)^2}{2}BP_i\Bigg[(By^{t^q})_i + (r_i\Delta t_i)(B^2y^{t^q})_i + \frac{(r_i\Delta t_i)^2}{2}BP_i(B^2y^{t^q})_i$$

$$+\sum_{n=3}^{2r_i}(\Delta t_i)^n\alpha_{n,r_i}(BP_i)^{n-1}(B^2y^{t^q})_i + \tilde{R}_{i,\text{asyn}}^{t_{i,q}^{r_i}}\Bigg] + \sum_{k=1}^{m+1}\delta_i^k\Bigg[\sum_{j\in\mathcal{N}^-(i)}F_{i,j}^-\Bigg[\Bigg[y_j^{t^q}$$

$$+(r_j^k\Delta t_j)(By^{t^q})_j + \frac{(r_j^k\Delta t_j)^2}{2}(B^2y^{t^q})_j + \sum_{n=3}^{2r_j^k}(\Delta t_j)^n\alpha_{n,r_j^k}(BP_j)^{n-2}(B^2y^{t^q})_j$$

$$+ R^{r_j^k}_{j,\text{asyn}}\Big] + \Big(t^k_{\text{mid},i,j} - r^k_j \Delta t_j\Big)\Big[(By^{t^q})_j + (r^k_j \Delta t_j)(B^2 y^{t^q})_j$$

$$+ \frac{(r^k_j \Delta t_j)^2}{2} BP_j (B^2 y^{t^q})_j + \sum_{n=3}^{2r^k_j} (\Delta t_j)^n \alpha_{n,r^k_j} (BP_j)^{n-1} (B^2 y^{t^q})_j + \tilde{R}^{r_j^k}_{j,\text{asyn}}\Big]\Big]\Big].$$

We use Proposition 2 to compute the following term

$$\frac{(r_i \Delta t_i)^2}{2}(B^2 y^{t^q})_i + (r_i (\Delta t_i)^2) BP_i \Big((By^{t^q})_i\Big) + \frac{(\Delta t_i)^2}{2} BP_i \Big((By^{t^q})_i\Big)$$

$$+ \sum_{k=1}^{m+1} \delta^k_i \Big[\sum_{j\in\mathcal{N}^-(i)} F^-_{i,j}\Big[(r^k_j \Delta t_j)(By^{t^q})_j + \Big(t^k_{\text{mid},i,j} - r^k_j \Delta t_j\Big)(By^{t^q})_j\Big]\Big] = \frac{((r_i+1)\Delta t_i)^2}{2}(B^2 y^{t^q})_i.$$

Therefore,

$$y_i^{t_{i,q}^{r_i+1}} = y_i^{t^q} + ((r_i+1)\Delta t_i)(By^{t^q})_i + \frac{((r_i+1)\Delta t_i)^2}{2}(B^2 y^{t^q})_i + \sum_{n=3}^{2(r_i+1)} (\Delta t_i)^n \alpha_{n,r_{i+1}} (BP_i)^{n-2} (B^2 y^{t^q})_i + R^{t_{i,q}^{r_i+1}}_{i,\text{asyn}},$$

where,

$$R^{t_{i,q}^{r_i+1}}_{i,\text{asyn}} = R^{t_{i,q}^{r_i}}_{i,\text{asyn}} + \Delta t_i BP_i R^{t_{i,q}^{r_i}}_{i,\text{asyn}} + \frac{(\Delta t_i)^2}{2} BP_i \tilde{R}^{t_{i,q}^{r_i}}_{i,\text{asyn}} + \sum_{k}^{m+1} \delta^k_i \left(R^{t_{j,q}^{r_j^k}}_{j,\text{asyn}} + \Big(t^k_{mid,i,j} - r^k_j \Delta t_j\Big) \tilde{R}^{t_{j,q}^{r_j^k}}_{j,\text{asyn}} \right)$$

$$+ \sum_{k}^{m+1} \delta^k_i \sum_{j\in\mathcal{N}^-(i)} F^-_{i,j}\Big[\frac{(r^k_j \Delta t_j)^2}{2}(B^2 y^{t^q})_j + \sum_{n=3}^{2r^k_j} (\Delta t_j)^n \alpha_{n,r^k_j} (BP_j)^{n-2} (B^2 y^{t^q})_j$$

$$+ \Big(t^k_{mid,i,j} - r^k_j \Delta t_j\Big)\Big[(r^k_j \Delta t_j)(B^2 y^{t^q})_j + \frac{(r^k_j \Delta t_j)^2}{2}) BP_j (B^2 y^{t^q})_j + \sum_{n=3}^{2r^k_j} (\Delta t_j)^n \alpha_{n,r^k_j} (BP_j)^{n-1} (B^2 y^{t^q})_j\Big]\Big].$$

To conclude, it remains to be noted that $R^{r_i}_{i,\text{asyn}}$ and $\tilde{R}^{r_i}_{i,\text{asyn}}$ are bounded using the induction hypothesis, in particular, the formula (8). Then the formula (6) is verified at $t^{r_i+1}_{i,q}$ with

$$\|R^{t_{i,q}^{r_i+1}}_{i,\text{asyn}}\|_\infty \le \sum_{n=3}^{M_i^{r_i+1}} C_{n,r_i+1} (\Delta t_i)^n \|B^n\|_\infty \|y^0\|_\infty,$$

$M_i^{r_i+1} = \max(2(r_i+1), h_{i,q}^{r_i+1})$. □

References

[1] M. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[2] C. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM J. Sci. Comput. 22 (2000) 2256–2281.
[3] J.E. Flaherty, R.M. Loy, M.S. Shephard, B.K. Szymanski, J.D. Teresco, L.H. Ziantz, Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws, J. Parallel Distrib. Comput. 47 (1997) 139–152.
[4] M.J. Grote, M. Mehlin, T. Mitkova, Runge–Kutta based explicit local time-stepping methods for wave propagation, February 2014, preprint No. 2014-05.
[5] R. Guivarch, G. Joslin, R. Perrussel, D. Ruiz, J. Tshimanga, T. Unfer, Parallelisation of MACOPA, a multi-physics asynchronous solver, in: 12th International Meeting on High Performance Computing for Computational Science, VECPAR, 2016, http://vecpar.fe.up.pt/2016/docs/VECPAR_2016_paper_14.pdf.
[6] https://www.calmip.univ-toulouse.fr/.
[7] W. Hundsdorfer, D.I. Ketcheson, I. Savostianov, Error analysis of explicit partitioned Runge–Kutta schemes for conservation laws, J. Sci. Comput. 63 (3) (2015) 633–653, 2000 Mathematics Subject Classification: 65L06, 65M06, 65M20.
[8] W. Hundsdorfer, A. Mozartova, V. Savcenco, Monotonicity conditions for multirate and partitioned explicit Runge–Kutta schemes, in: R. Ansorge, et al. (Eds.), Recent Developments in the Numerics of Nonlinear Hyperbolic Conservation Laws, in: Notes Numer. Fluid Mech., vol. 120, Springer, 2013, pp. 177–195.
[9] A. Lew, J.E. Mardsen, M. Ortiz, M. West, Asynchronous variational integrators, Arch. Ration. Mech. Anal. 167 (2003) 85.
[10] A.P. Markesteijn, S.A. Karabasov, Time asynchronous relative dimension in space method for multi-scale problems in fluid dynamics, J. Comput. Phys. 258 (2014) 137–164.
[11] Y.A. Omelchenko, H. Karimabadi, Self-adaptive time integration of flux-conservative equations with sources, J. Comput. Phys. 216 (1) (20 July 2006) 179–194.
[12] Y.A. Omelchenko, H. Karimabadi, A time-accurate explicit multi-scale technique for gas dynamics, J. Comput. Phys. 226 (2007) 282–300.
[13] Y.A. Omelchenko, H. Karimabadi, HYPERS: a unidimensional asynchronous framework for multiscale hybrid simulations, J. Comput. Phys. 231 (2012) 1766–1780.

[14] S. Osher, R. Sanders, Numerical approximations to nonlinear conservation laws with locally varying time and space grids, Math. Comput. 41 (1983) 321–336.
[15] V. Savcenco, W. Hundsdorfer, J.G. Verwer, A multirate time stepping strategy for stiff ordinary differential equations, BIT Numer. Math. 47 (2007) 137–155.
[16] V.A. Semiletov, S.A. Karabasov, CABARET scheme with conservation-flux asynchronous time-stepping for nonlinear aeroacoustics problems, J. Comput. Phys. 253 (2013) 157–165.
[17] H.Z. Tang, Warnecke, A class of high resolution schemes for hyperbolic conservation laws and convection–diffusion equations with varying time and space grids, preprint, 2003.
[18] A. Taube, M. Dumbser, C.D. Munz, R. Schneider, A high-order discontinuous Galerkin method with time–accurate local time stepping for the Maxwell equations, Int. J. Numer. Model. 22 (2009) 77–103.
[19] A. Toumi, Methode numerique asynchrone pour la modelisation de phenomenes multi-echelles, PhD. Thesis, University of Toulouse III, France, 2016.
[20] A. Toumi, G. Dufour, R. Perrussel, T. Unfer, Asynchronous numerical scheme for modeling hyperbolic systems, C. R. Acad. Sci. Paris, Ser. I 353 (2015) 843–847.
[21] T. Unfer, An asynchronous framework for the simulation of the plasma/flow interaction, J. Comput. Phys. 236 (2013) 229–246.
[22] T. Unfer, J.-P. Boeuf, F. Rogier, An asynchronous scheme with local time-stepping for multi-scale transport problems: application to gas discharges, J. Comput. Phys. 227 (1) (2007) 898–918.