Computers and Fluids 229 (2021) 105102

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

A conservative multirate explicit time integration method for computation of compressible flows

Ramzi Messahel *, Gilles Grondin, Jérémie Gressier, Julien Bodart

ISAE-SUPAERO, Université de Toulouse, France

ARTICLE INFO

Keywords:
Multirate explicit time integration
Finite volume
Spectral-differences
High performance computing
Direct numerical simulation

ABSTRACT

In the context of high fidelity simulation of compressible flows (LES and DNS) at extreme scale (small time steps) on massively parallel supercomputers, explicit time integration methods are widely used since they both have a good computational cost trade-off at small time scales and require a small number of parallel communications, thus having little impact on the parallel strategy compared to their implicit counterpart. However, the synchronous nature of the time integration of the governing equations can be a severe constraint when the stability condition is applied globally since the time scales are related to the flow properties and the mesh resolution, which may show strong variations throughout the computational domain. We propose to further improve the efficiency (CPU wall-time reduction) of explicit Runge–Kutta methods by developing a multirate explicit time integration method, by means of flux interpolation at the boundary between cells evolving with different time-steps, which enforces the conservation properties. In terms of computational efficiency, the presented multirate time integration method is easy to implement in pre-existing Eulerian compressible Navier–Stokes codes, requires less additional memory storage, and provides a considerable speed-up while being robust and preserving the order of accuracy of the legacy explicit Runge–Kutta time integration method. The multirate time integration method is implemented in the massively parallel finite volume and high-order (spectral difference) IC^3 code (Bodart et al., 2016) (fork of the solver CharLES^X), but it can also be applied to any flux-based spatial method such as discontinuous Galerkin or others. For a targeted $y^+ = 0.2$ on the developed turbulent channel flow test case at $Re_\tau = 392$; a 2.48 effective speedup is obtained versus an expected theoretical speedup of 2.53.

1. Introduction

Explicit time integration methods are widely used for high-fidelity compressible flow simulations at small time scales in modern parallel codes due to their efficiency (limited amount of computations per iteration) and the small number of communications required in comparison with implicit time integration when small time steps are involved [1]. However, the main drawback of explicit time schemes is their conditional stability determined by the Courant-Friedrichs-Lewy (CFL) stability condition [2]. For each cell of the computational domain, this CFL stability criterion defines a maximum local time step proportional to the ratio between a characteristic cell length (e.g., grid size) and the maximum characteristic speed (wave/advection) by a fixed $\mathrm{CFL}_{\max}$ factor specific to the time scheme, and states that the local-step cannot exceed this maximum local time step. Under these constraints, the stable global time step is determined by the smallest local time step amongst all cells. For multi-scale phenomena such as highly turbulent compressible flows where extensive fine meshing is needed locally for the numerical method to be able to converge and capture the physical phenomena, this approach may lead to unduly expensive computations if the global time step imposed for the sake of stability is much smaller than the computed local time step for a large number of cells. Explicit multirate time integration methods aim at overcoming such limitations of explicit time integration methods imposed by the CFL condition, to considerably reduce the overall CPU time, increase the code productivity of people using the code, and finally solve more complex and realistic problems. To do so, the computational domain is decomposed into several time stepping classes of cells sharing the same class time step which satisfies the CFL stability condition within the whole class of cells, while minimizing the ratio of the local time steps to the different class time steps.

Two main difficulties arise from explicit multirate time integration methods: the global synchronization of the time stepping classes and the processing of inter-class boundaries between cells advancing in time with different class time steps. The global synchronization aims

* Corresponding author.
E-mail address: ramzi.messahel@isae-supaero.fr (R. Messahel).

https://doi.org/10.1016/j.compfluid.2021.105102
Received 10 June 2020; Received in revised form 5 July 2021; Accepted 19 July 2021
Available online 5 August 2021

at preserving temporal coherence to avoid any physical time-lag in space (due to the spatial decomposition of the computational domain into local time stepping classes), and also, to impose the conservation property by the means of *a-posteriori* local flux corrections. The processing of inter-class boundaries must then enforce the conservation property and be implemented efficiently to be deployed in a multidimensional and unstructured mesh framework. In the work of Constantinescu and Sandu [3], Puppo and Semplice [4], Seny et al. [5] and others, the transition between classes with different time steps is managed by inserting an intermediate buffer time stepping class. In this work, we propose a more compact approach where management of the transition between time stepping classes will be done directly at the inter-class boundary without passing through a transition buffer zone making the approach more computationally efficient in an unstructured mesh context as it requires fewer data and computations at the inter-class interface. In a high-performance computing (HPC) framework, a straightforward application of explicit multirate time integration may unbalance the computational load across the processes/threads since the number of time integrations varies between time stepping classes. It follows that an efficient parallel implementation of explicit multirate time integration methods becomes a third difficulty.

Early explicit multirate time integration methods [6–8] were found to be either locally inconsistent or non-mass-conservative (see Hundsdorfer et al. [9]). Later, two explicit multirate time integration methods based on explicit Runge–Kutta (ERK) methods were proposed by Constantinescu and Sandu [3] and Schlegel et al. [10]. In these methods, the difficulty of processing the time stepping inter-class boundaries is handled by creating buffer time stepping regions. This buffer region is created inside the time stepping classes with the biggest time step so that at least two cells separate the time stepping classes, while its time step is set as the minimum of the two neighboring class time steps. The main drawback of this approach is the loss of computational efficiency due to the underestimation of the local time steps of the buffer region. However, the first method is conservative and preserves the strong-stability-preserving (SSP) properties of the synchronous ERK-based method but was found to achieve at most second-order accuracy in time. The second one achieved third-order accuracy in time but is non-conservative (see [10,11]). Seny et al. [5] proposed an efficient parallel strategy for a Discontinuous-Galerkin type implementation of the explicit multirate time integration method of Constantinescu and Sandu [3]. More recently, Jeanmasson et al. [12] proposed other conservative in mass second-order and third-order schemes. The conservation property is recovered with an additional correctional phase to enforce the mass conservation. These latest schemes proposed by Jeanmasson et al. [12] were shown to be equivalent or more accurate than schemes proposed by Constantinescu and Sandu [3] on the presented numerical tests.

In this paper, we present another robust explicit multirate time integration that has the benefits of being conservative by construction, of preserving the order of accuracy of the legacy single rate method, and of treating locally the transition between the different local time stepping classes at the inter-class boundary through conservative flux interpolation techniques. The local treatment of the inter-class boundaries avoids using overlapping buffer local time stepping classes and consequently improves the efficiency of the parallel implementation for solving three-dimensional problems. In Section 2, we give a brief description of the considered governing Navier–Stokes equations and the legacy spatial and time discretization methods; the basis and principles of the proposed conservative multirate explicit time integration and its implementation including the grid partitioning strategy are described in Sections 3 and 4, respectively; finally, validation and performance results are presented in Section 5.

2. Numerical method

2.1. Governing equations

In this work, we consider the compressible three-dimensional Navier–Stokes equations in their local conservative form as follows:

$$\partial_t \boldsymbol{Q} + \nabla_{\boldsymbol{x}} \cdot \boldsymbol{F} = \boldsymbol{S}, \tag{1}$$

where $\boldsymbol{Q}$, $\boldsymbol{F}$ and $\boldsymbol{S}$ denote the conserved variables, the flux tensor in the x, y and z directions and the source terms, respectively. The conserved variables and flux vectors are defined as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho E \end{pmatrix}, \tag{2}$$

$$\boldsymbol{F} = \boldsymbol{F}^{\text{inviscid}} - \boldsymbol{F}^{\text{viscous}} = \begin{pmatrix} \rho \boldsymbol{u} \\ \rho \boldsymbol{u} \otimes \boldsymbol{u} + p\overline{\overline{\mathbf{I}}}_{\mathrm{d}} \\ (\rho E + p)\boldsymbol{u} \end{pmatrix} - \begin{pmatrix} 0 \\ \overline{\overline{\tau}} \\ \overline{\overline{\tau}} \cdot \boldsymbol{u} + \boldsymbol{q} \end{pmatrix}, \tag{3}$$

where ρ, $\boldsymbol{u}$, p, E, $\overline{\overline{\tau}}$ and $\boldsymbol{q}$ are the density, the velocity vector, the pressure, the total energy, the stress tensor and the heat flux vector, respectively.

The pressure and total energy are related by the ideal gas law with:

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho \boldsymbol{u} \cdot \boldsymbol{u}\right), \tag{4}$$

where γ is the constant ratio of specific heats.

2.2. Finite-volume method

The present study was conducted using the *Finite-volume* (FV) method implemented in the massively parallel compressible flow solver CharLES$^{\mathrm{X}}$ (see [13]). The FV method is a cell-centered control volume (cv) based discretization approach on unstructured polyhedral meshes where the flux is computed at each control volume face using a convex combination of a non-dissipative central flux and a dissipative upwind flux (see [14–16]), such as

$$\boldsymbol{F}^{\mathrm{i}} = \alpha \boldsymbol{F}^{\mathrm{i}}{}_{\text{central}} + (1 - \alpha)\boldsymbol{F}^{\mathrm{i}}{}_{\text{upwind-HLLC}} \quad . \tag{5}$$

The α parameter is a local face parameter. It is used to choose the flux calculation according to the geometric quality regions. For good quality regions, the α parameter tends to unity such that a low-dissipative or non-dissipative blended flux is used. In the opposite scenario, the α parameter is decreased to avoid instabilities due to the centered scheme through the addition of numerical dissipation in the blended flux. The code CharLES$^{\mathrm{X}}$ also allows solving shocks through the use of shock sensors (dilatation, vorticity, density gradient, and pressure gradient-based sensors) and the activation of second-order ENO schemes.

The considered left and right face data for flux computation are high order polynomial reconstructed data on an extensive stencil. For more details on the CharLES$^{\mathrm{X}}$ code numerical methods and its implementation, the reader can refer to [14–16]. Arbitrary-Lagrangian–Eulerian and sliding mesh capabilities were recently added by Saez-Mischlich et al. [17].

2.3. Spectral-difference method

In this paper, we consider the high-order compact *Spectral-difference* (SD) method on unstructured hexahedral grids (see Kopriva [18], Liu et al. [19] and Sun et al. [20]) that has been implemented in the department code IC^3 [1], a fork of an already well established LES FV solver CharLES$^{\mathrm{X}}$ developed at the Center for Turbulence Research [13].

Coordinate transformation. For computation efficiency purposes, each hexahedral cell is transformed from the physical domain (x, y, z) to a reference hexahedral element $(\xi, \eta, \zeta) \in [0, 1]^3$ (see Fig. 1) by a linear bijective mapping function $\Phi(x, y, z)$ defined as follows

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \sum_{i=1}^{8} M_i(\xi, \eta, \zeta) \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}, \tag{6}$$

where (x_i, y_i, z_i) and $M_i(\xi, \eta, \zeta)$ denotes the mesh points coordinates and the shape functions, respectively. Considering the transformation function in Eq. (6), the governing equations are expressed in the reference domain coordinate system by

$$\partial_t \hat{\boldsymbol{Q}} + \nabla_{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{F}} = 0, \tag{7a}$$

$$\begin{cases} \hat{\boldsymbol{Q}} = |\boldsymbol{J}| \, \boldsymbol{Q} \\ \hat{\boldsymbol{F}} = |\boldsymbol{J}| \, \underline{\underline{\boldsymbol{J}}}^{-1} \boldsymbol{F} \end{cases} \quad , \quad \underline{\underline{\boldsymbol{J}}} = \nabla_{\boldsymbol{\xi}} \boldsymbol{x} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}} = \begin{pmatrix} \partial_\xi x & \partial_\eta x & \partial_\zeta x \\ \partial_\xi y & \partial_\eta y & \partial_\zeta y \\ \partial_\xi z & \partial_\eta z & \partial_\zeta z \end{pmatrix}, \tag{7b}$$

where $|\boldsymbol{J}|$ is the determinant of the Jacobian matrix $\underline{\underline{\boldsymbol{J}}}$.

Spatial discretization. In the SD method used in this study, two sets of mesh points are defined for each element (see Fig. 2):

- the solution points which are chosen to be the Gauss points,
- and the flux points which are chosen to be the Gauss–Legendre points that have proven to be more stable (see [21]).

The discrete solution of the conservative variables is defined at the solution points and then interpolated to flux points where the fluxes are computed using a Lagrange polynomial interpolation. At this stage, it is important to notice that the solution used for interpolation at the flux points is continuous inside each element and it is but discontinuous at elements interfaces. The common conservative flux at the discontinuous Riemann interface is computed using a Riemann solver (both Harten–Lax–van Leer-Contact and Rusanov Riemann solvers are implemented in our department code IC^3). The computation of the inviscid fluxes and the reconstruction of the viscous fluxes are detailed in [20].

2.4. Explicit Runge–Kutta methods

General form. To evolve the solution in time, we consider the class of ERK time integration methods applied to the semi-discrete problem

$$\partial_t \boldsymbol{Q} = \mathrm{RHS}(\boldsymbol{Q}) = -\nabla_{\boldsymbol{x}} \cdot \boldsymbol{F}, \tag{8}$$

where the spatial derivatives regrouped in the right-hand side $\mathrm{RHS}(\boldsymbol{Q})$ are, firstly, discretized using either a FV or a SD method.

Let us consider the discrete times $t_0 < t_1 < \cdots < t_N$ and a computed solution $\boldsymbol{Q}(t_n)$ at time t_n with $0 \leq n < N$. The general form of explicit s-stages RK methods is given by

$$\forall i = 1, \ldots, s \begin{cases} t_{n,i} &= t_n + c_i \, \Delta t_n \, , \\ \boldsymbol{Q}_{n,i} &= \boldsymbol{Q}(t_n) + \Delta t_n \sum_{k=1}^{i-1} a_{i,k} \, \boldsymbol{p}_{n,k} \, , \\ \boldsymbol{p}_{n,i} &= \mathrm{RHS}(\boldsymbol{Q}_{n,i}) \, , \end{cases}$$
$$\boldsymbol{Q}(t_{n+1}) = \boldsymbol{Q}(t_n) + \Delta t_n \sum_{k=1}^{s} b_k \, \boldsymbol{p}_{n,k} \quad , \tag{9}$$

where $\Delta t_n = t_{n+1} - t_n$ is the time step and matrices $\boldsymbol{A} = (a_{i,j})_{s \times s}$, $\boldsymbol{b} = (b_i)_s$ and $\boldsymbol{c} = (c_i)_s$ are composed of the coefficients defining the RK methods, which must satisfy $\forall i \in [1, s], \sum_{k=1}^{i-1} a_{i,k} = c_i$ and $\sum_{k=1}^{s} b_k = 1$. In Fig. 3, the RK methods coefficients are represented in the Butcher tableau form. In the specific case of explicit time integration presented here, the matrix $\boldsymbol{A} = (a_{i,j})_{s \times s}$ is lower triangular with a null diagonal.

Courant-friedrichs-lewy (CFL) stability condition. Although ERK methods provide the advantage of having a good computational cost trade-off at small time scales and require a small number of parallel communications compared to their implicit counterpart; the computational time step is restricted by the CFL condition to preserve the stability condition and, consequently, the convergence of the numerical solution. For a given control volume (CV) Ω_k, the CFL condition is locally defined by

$$\mathrm{CFL}_k = \max(a_k^{\mathrm{conv}}, a_k^{\mathrm{acou}}) \frac{\Delta t_k}{l_k} \leqslant \mathrm{CFL}_{\max}, \tag{10}$$

where CFL_k, a_k^{conv}, a_k^{acou}, Δt_k, l_k and $\mathrm{CFL}_{\max}$ denote the local CFL number, convective wave speed, acoustic wave speed, time step, characteristic length and the maximum authorized CFL number, respectively. To ensure the CFL condition on the whole domain, the stable global time step is determined by the smallest local time step defined by the CFL definition in Eq. (10) among all control volumes.

This approach may lead to expensive computation if the stability-driven global time step is much smaller than the local time step for a great number of cells. The next section presents a conservative multirate explicit time integration method that overcomes the restriction of the local time step by the global time step enabling the solution to evolve with locally different time steps.

3. Conservative multirate explicit integration methods

3.1. Definition and management of time-step classes

The explicit multirate time integration method consists in integrating temporally the different cells of the mesh with different time-steps while ensuring the temporal coherence of the time integration, i.e., the first-order accuracy, and potentially its accuracy at higher orders. For practical reasons of implementation and efficiency, the N_{cell} cells of the computational domain are grouped into $1 + N_{\mathrm{c}}$ local time-step classes

$$C_i, \; 0 \leq i \leq N_{\mathrm{c}}$$

, where the cells of class C_i share the same global time-step ΔT_i which is specifically defined for a class. The ratio between the different class time-steps can be an integer [3,6,8,10,23–25] or a real [26,27].

In this work, we choose a ratio of two between the time-steps of two successive classes: $\Delta T_{N_{\mathrm{c}}} = \Delta t_{\min}$, $\Delta T_i = 2 \, \Delta T_{i+1}$. The partitioning of the computational domain into separate local time-step classes is based on the local time-step of each cell, which is a function of: the CFL number, the local physical field (convection and acoustic speeds) and the grid-cell characteristic length (Eq. (10)). The local time-step class partition is built as follows:

1. Computation of the time-step for each cell Ω_k, $\forall k \in I_{N_{\mathrm{cell}}} = [1, N_{\mathrm{cell}}]$:

$$\Delta t_k = \Delta t_k(\mathrm{CFL}_k, l_k(\Omega_k), a_k^{\mathrm{conv}}, a_k^{\mathrm{acou}}) \quad ; \tag{11a}$$

2. Computation of the minimum and maximum time-steps:

$$\Delta t_{\min} = \min_{k \in I_{N_{\mathrm{cell}}}} (\Delta t_k) \quad , \quad \Delta t_{\max} = \max_{k \in I_{N_{\mathrm{cell}}}} (\Delta t_k) \quad ; \tag{11b}$$

3. Computation of the number $N_{\mathrm{c}} + 1$ of classes needed to cover the time-step range $[\Delta t_{\min}; \Delta t_{\max}]$:

$$N_{\mathrm{c}} = \left\lfloor \log_2 \left(\Delta t_{\max} \Big/ \Delta t_{\min} \right) \right\rfloor \; ; \tag{11c}$$

4. Computation of each class time-step:

$$\forall i \in [0, N_{\mathrm{c}}], \; \Delta T_i = 2^{N_{\mathrm{c}} - i} \Delta t_{\min} \quad ; \tag{11d}$$

5. Computation of local time-step intervals:

$$J_i = \begin{cases} [\Delta T_i, \Delta T_{i-1}), \; \forall i \in [1, N_{\mathrm{c}}] \, , \\ [\Delta T_i, \Delta t_{\max}], \; i = 0 \quad ; \end{cases} \tag{11e}$$

6. Definition of the time-step classes

$$C_i, \; 0 \leq i \leq N_{\mathrm{c}}$$

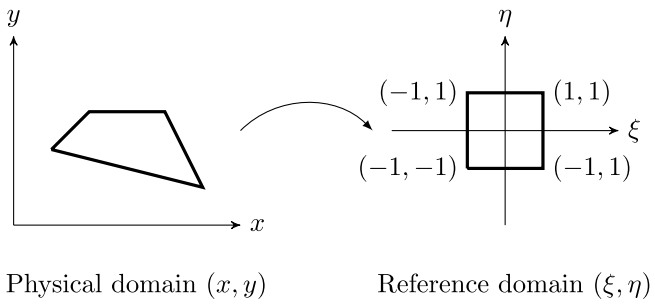


Fig. 1. Linear transformation from the physical domain (x, y) to reference domain (ξ, η).

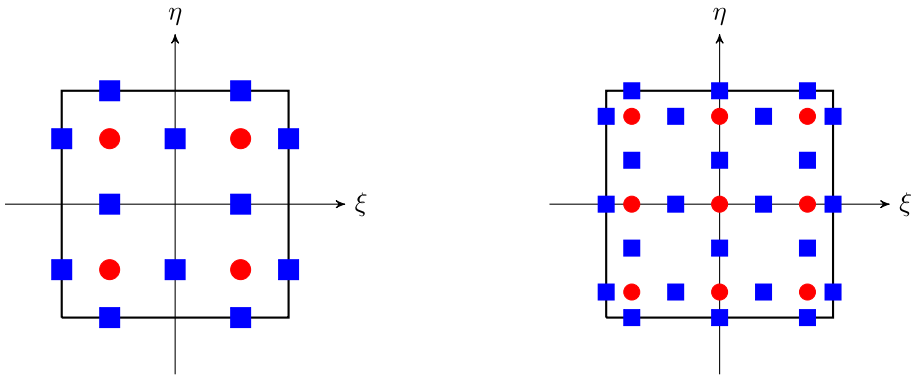


Fig. 2. Representation of the solution points (circles) and flux points (solid squares) for two-dimensional SD methods: second order (left) and third order (right).

General form of ERK methods

$$\begin{array}{c|c} \boldsymbol{c} & \boldsymbol{A} \\ \hline & \boldsymbol{b}^{\mathrm{T}} \end{array}$$

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} \\ \hline & b_1 & b_2 & \cdots & b_{s-1} \quad b_s \end{array}$$

RK1

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

RK2a

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

RK3 SSP, Shu and Osher

$$\begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ 1/2 & 1/4 & 1/4 & \\ \hline & 1/6 & 1/6 & 2/3 \end{array}$$

RK4

$$\begin{array}{c|ccccc} 0 & & & & & \\ 1/2 & 1/2 & & & & \\ 1/2 & 0 & 1/2 & & & \\ 1 & 0 & 0 & 1 & & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 & \end{array}$$

Fig. 3. Butcher's tableaux for ERK methods, see [22].

which contain all cells Ω_k with a local time-step Δt_k in time-step interval J_i:

$$C_i = \left\{ \Omega_k, k \in I_{N_{\text{cell}}} \mid \Delta t_k \in J_i \right\} \quad . \tag{11f}$$

Each cell of a class C_i ($0 \leq i \leq N_c$) is integrated in time with time-step $\Delta T_i = 2^{N_c - i} \Delta t_{\min}$, the lower bound of interval J_i, in order to locally satisfy the CFL condition (Eq. (10)). The distribution of classes and local time-steps are shown in Fig. 4. For one global iteration of time-step

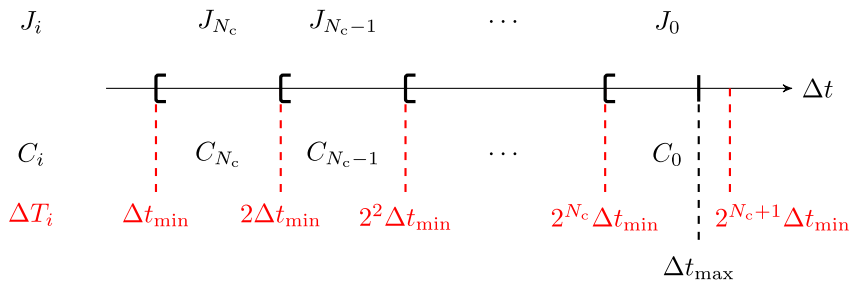


Fig. 4. Distribution of classes and time-steps in logarithmic scale.

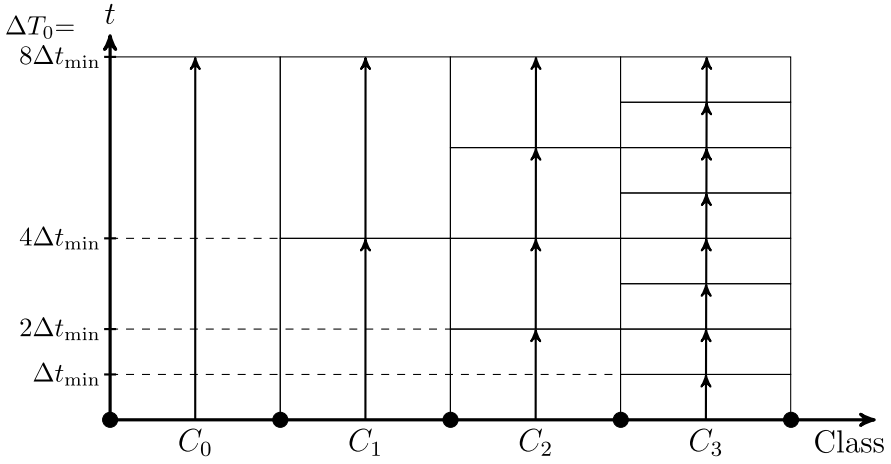


Fig. 5. A four-class ($N_c = 3$) example of multirate time integration over a global synchronized time-step $\Delta T_0 = 2^{N_c}\Delta t_{\min} = 8\Delta t_{\min}$.

$\Delta T_0 = 2^{N_c}\Delta t_{\min}$, cells of a class C_i are integrated in time 2^i times with respect to their class time-step ΔT_i as shown in Fig. 4.

One major difficulty in evolving the solution asynchronously is the treatment of the interface between cells advancing with different class time-steps (inter-class faces). For practical and efficiency reasons, a difference of at most one class between two adjacent cells is imposed as a constraint: Otherwise, one would require either to add layers of cells with intermediate time-steps, which would be complex to implement, or a large number of time integrations (four, eight, or higher power of two) on the smaller time-step side while the larger time-step side would not evolve, which may cause stability problems or deteriorate the accuracy of the method.

To enforce this constraint, an additional step is performed: when two adjacent cells belong to classes with an absolute index difference greater than one, then the cell belonging to the class with the smaller index (i.e. with the greater time-step) is transferred into the class with higher index complying with the index difference constraint, to satisfy the local CFL stability condition. This step is repeated while required. An example of multirate time integration over a global synchronized time-step ΔT_0 is shown in Fig. 5.

It is important to underline that the previous reasoning allows decomposing the domain into integration classes based on the definition of the cells' local time-step defined as a function of the local CFL, the physical characteristics of the flow, and the mesh metrics. Considering that the local CFL is proportional to the time-step (see Eq. (10)), it is also possible to have the same reasoning for the definition of the classes based on the local CFL if fixed time-steps is considered. In the particular case of incompressible flows with a low Mach number, the convective speed is negligible compared to the acoustic speed and the domain decomposition into classes can be based on the grid size which is proportional to the time step and inversely proportional to the CFL.

Expected gain. Let $n_i = \mathrm{card}(C_i)$ and N_{cell} be the number of cells in class C_i and the total number of cells, respectively. The numbers Z_{sync} and Z_{async} of synchronous and multirate time integrations needed to perform a global time integration over a global synchronized time-step $\Delta T_0 = 2^{N_c}\Delta t_{\min}$ and the theoretical expected gain G_{th} are given by

$$Z_{\mathrm{sync}} = \sum_{i=0}^{N_c} n_i\, 2^{N_c}\ ,\ Z_{\mathrm{async}} = \sum_{i=0}^{N_c} n_i\, 2^{i}\ ,\ G_{\mathrm{th}} = \frac{Z_{\mathrm{sync}}}{Z_{\mathrm{async}}}\quad . \tag{12}$$

3.2. Flux processing at multirate classes interfaces

For the sake of clarity, we only consider two-class multirate time integration cases ($N_c = 1$). The generalization of the method to N_c classes and the global synchronization algorithm are presented later in Section 3.3.

In the following subsections, we denote by $\boldsymbol{F}^{n,j}_{k+1/2}$, $\boldsymbol{Q}^{n,j}_{k}$ and $\boldsymbol{Q}^{n,j}_{k+1}$ the flux at the $k+1/2$ interface between cells Ω_k and Ω_{k+1} and the states at these cells at a given ERK sub-step j of time-step n, respectively. The considered interface between classes C_0 and C_1 is located at interface $i+1/2$, with cells Ω_k belonging to class C_0 for $k \le i-1$ and to class C_1 for $k \ge i$.

3.2.1. Non-conservative multirate RK1 method

A first straightforward approach for evolving asynchronously the cells of the two classes C_0 and C_1 with respective time-steps $\Delta T_0 = 2\Delta t_{\min}$ and $\Delta T_1 = \Delta t_{\min}$ is to advance in time each cell C_k with respect to its local class time-steps ΔT_k $(k = 0, 1)$ starting by the smallest time-step class C_1 following by the largest time-step class C_0. The successive multirate steps are enumerated below and shown in Fig. 6

1. Computation of fluxes at the faces of classes C_0 and C_1 faces based on the known states at time t_0 $\left(\boldsymbol{F}^0_{k+1/2}\left(\boldsymbol{Q}^0_k, \boldsymbol{Q}^0_{k+1}\right)_{k=i-2,i-1,i}\right.$, $\boxed{\boldsymbol{Q}^0_{i-1}}$ and $\boxed{\boldsymbol{Q}^0_{i}}$ in Fig. 6, respectively$\left.\right)$.

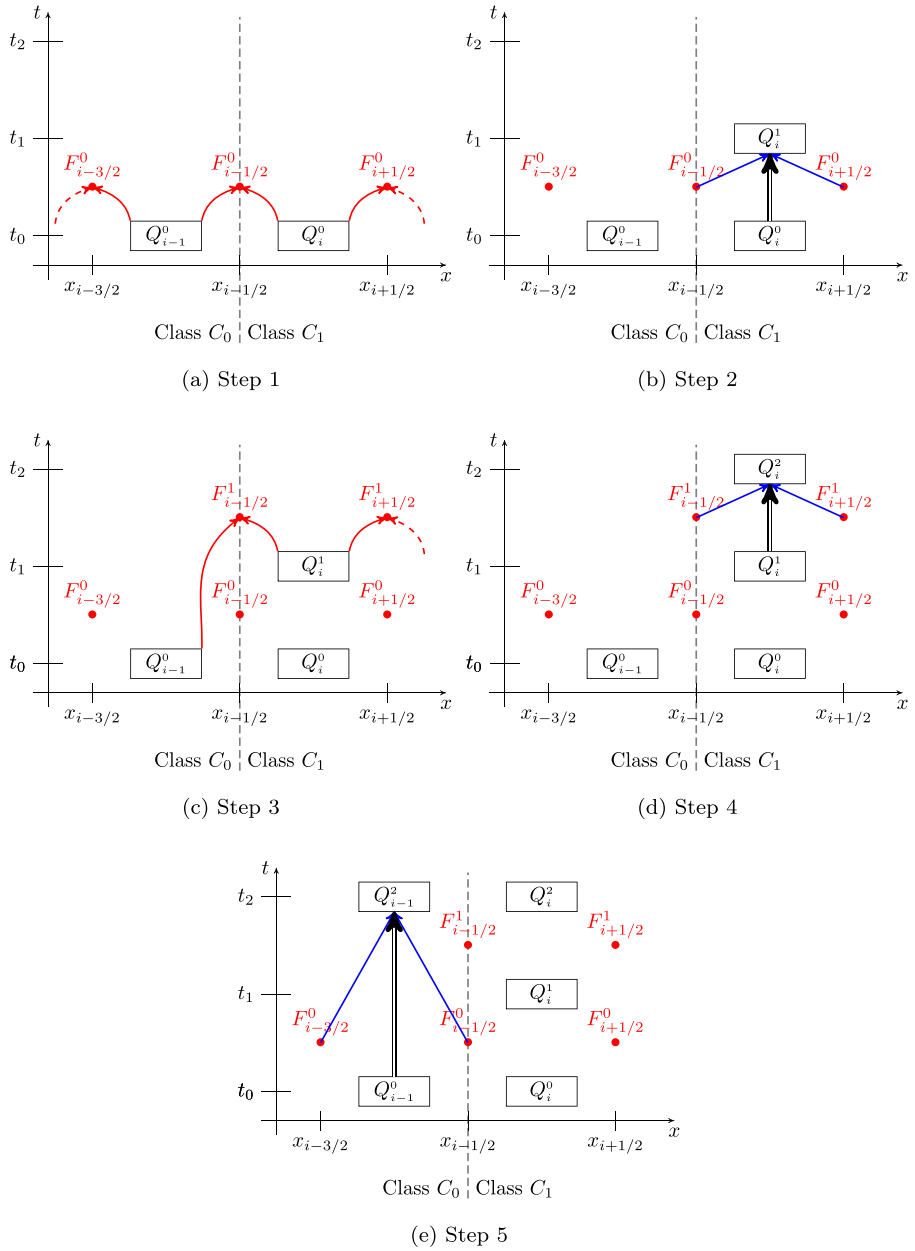


Fig. 6. Non-conservative local time-stepping ERK1 algorithm between times t_0 and $t_2 = t_0 + 2\Delta t_{\text{min}}$ with intermediate time $t_1 = t_0 + \Delta t_{\text{min}}$.

2. Computation of class C_1 new states ($\boxed{\boldsymbol{Q}^1_i}$ on Fig. 6, respectively) at time $t_1 = t_0 + \Delta t_{\text{min}}$ through summation of states and aggregation of fluxes as defined in Eq. (9) ($\boxed{\boldsymbol{Q}^0_i}$, $\boldsymbol{F}^0_{i-1/2}$ and $\boldsymbol{F}^0_{i+1/2}$ in Fig. 6).
3. Computation of fluxes at the faces of class C_1 faces from the known states at times t_0 and $t_1 = t_0 + \Delta t_{\text{min}}$ for class C_0 and C_1 cells, respectively. ($\boldsymbol{F}^1_{i-1/2}(\boldsymbol{Q}^0_{i-1}, \boldsymbol{Q}^1_i)$, $\boldsymbol{F}^1_{i+1/2}(\boldsymbol{Q}^1_i, \boldsymbol{Q}^1_{i+1})$, $\boxed{\boldsymbol{Q}^0_i}$ and $\boxed{\boldsymbol{Q}^1_{i+1}}$ in Fig. 6, respectively).
4. Computation of class C_1 new states ($\boxed{\boldsymbol{Q}^2_i}$ in Fig. 6, respectively) at time $t_2 = t_0 + 2\Delta t_{\text{min}}$ through summation of states and aggregation of fluxes as defined in Eq. (9) ($\boxed{\boldsymbol{Q}^1_i}$, $\boldsymbol{F}^1_{i-1/2}$ and $\boldsymbol{F}^1_{i+1/2}$ in Fig. 6).
5. Computation of class C_0 new states ($\boxed{\boldsymbol{Q}^2_{i-1}}$ in Fig. 6, respectively) at time $t_0 + \Delta T_0$ through summation of states and aggregation of fluxes as defined in Eq. (9) ($\boxed{\boldsymbol{Q}^0_{i-1}}$, $\boldsymbol{F}^0_{i-3/2}$ and $\boldsymbol{F}^0_{i-1/2}$ in Fig. 6).

This multirate algorithm presents two major drawbacks :

- At the third step, the C_0/C_1 interface flux is computed at time $t_1 = t_0 + \Delta t_{\text{min}}$ ($\boldsymbol{F}^1_{i-1/2}$ on Fig. 6) from the non-updated C_0 states at time t_0 ($\boxed{\boldsymbol{Q}^0_{i-1}}$ on Fig. 6). This time inconsistency introduces second-order time inaccuracy since we are locally staggered in time.
- The conservation property is lost at the C_0/C_1 interface since the present algorithm uses different fluxes on each side of the interface ($\boldsymbol{F}^0_{i-1/2}$ on the C_0 side and the two fluxes $\boldsymbol{F}^0_{i-1/2}$ and

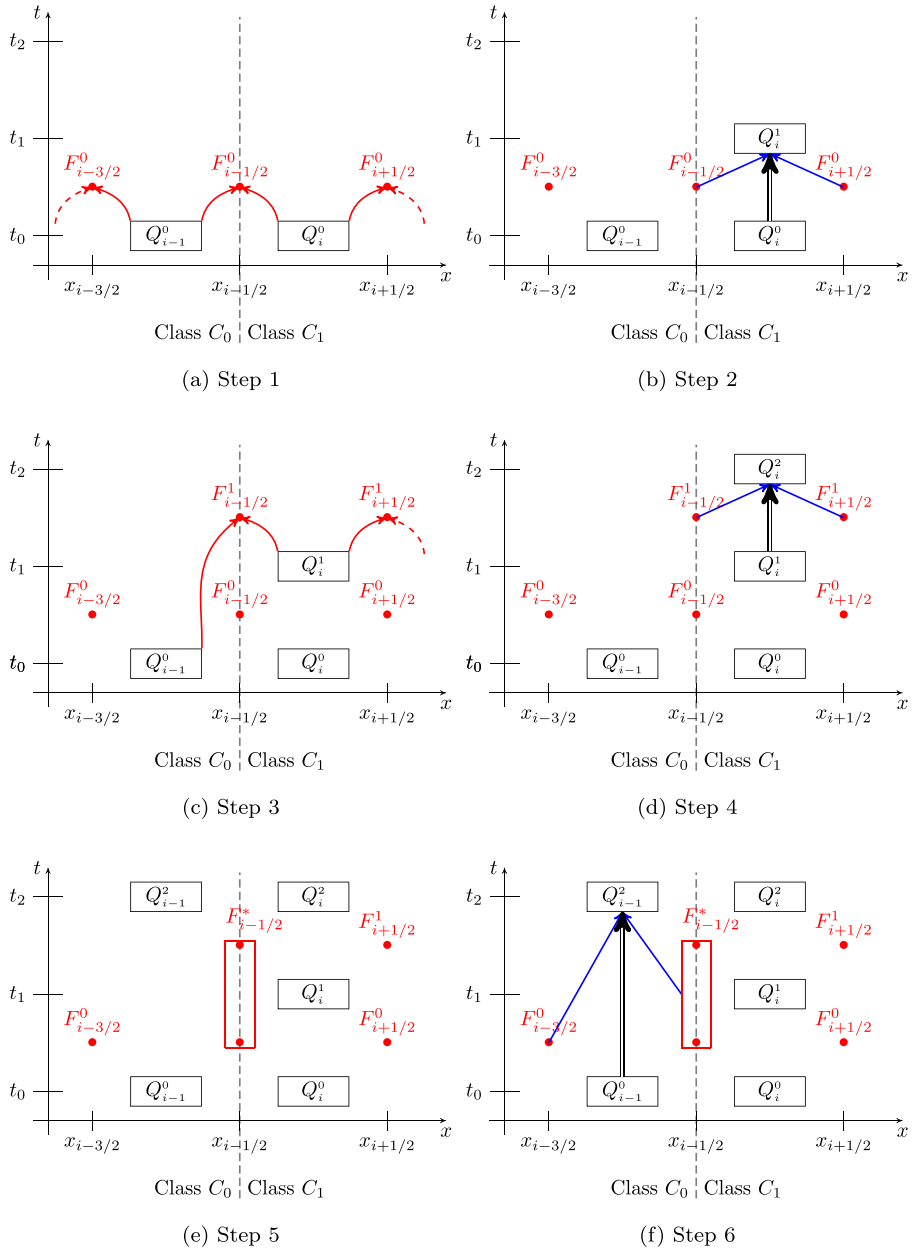


Fig. 7. Conservative local time-stepping ERK1 algorithm between times t_0 and $t_2 = t_0 + 2\Delta t_{\min}$ with intermediate time $t_1 = t_0 + \Delta t_{\min}$, where the corrected flux is given by $\boldsymbol{F}^*_{i-1/2} = \frac{1}{2}\left[\boldsymbol{F}^0_{i-1/2}(\boldsymbol{Q}^0_{i-1}, \boldsymbol{Q}^0_i) + \boldsymbol{F}^1_{i-1/2}(\boldsymbol{Q}^0_{i-1}, \boldsymbol{Q}^1_i)\right]$.

$\boldsymbol{F}^1_{i-1/2}$ on the C_1 side in Fig. 6, respectively) in order to obtain the final synchronized states ($\boxed{\boldsymbol{Q}^2_{i-1}}$ and $\boxed{\boldsymbol{Q}^2_i}$ in Fig. 6).

3.2.2. Conservative multirate RK1 method

To ensure the conservation (strict exchange of fluxes at the C_0/C_1 interface), it is necessary to modify the previous Section 3.2.1 algorithm. To do so, it is proposed to correct the C_0/C_1 interface fluxes ($\boldsymbol{F}^0_{i-1/2}$ in Fig. 6) used to compute class C_0 states at time $t_0 + \Delta T_0$ ($\boxed{\boldsymbol{Q}^2_{i-1}}$ in Fig. 6) so that conservation is enforced.

Let us examine the flux balance over a time integration from time t_0 to $t_2 = t_0 + \Delta T_0$ with intermediate time $t_1 = t_0 + \Delta T_1$ for class C_1 cells. According to Eq. (9), class C_0 final states $\boldsymbol{Q}^{C_0}(t_2)$ ($\boxed{\boldsymbol{Q}^2_{i-1}}$ in Fig. 6) are given by

$$\boldsymbol{Q}^{C_0}(t_2) = \boldsymbol{Q}^{C_0}(t_0) + 2\Delta t_{\min}\left[\mathrm{RHS}(\boldsymbol{Q}^{C_0}(t_0))\right], \tag{13}$$

while class C_1 final states $\boldsymbol{Q}^{C_1}(t_2)$ ($\boxed{\boldsymbol{Q}^2_i}$ on Fig. 6) are given by

$$\boldsymbol{Q}^{C_1}(t_2) = \boldsymbol{Q}^{C_1}(t_0) + \Delta t_{\min}\left[\mathrm{RHS}(\boldsymbol{Q}^{C_1}(t_0)) + \mathrm{RHS}(\boldsymbol{Q}^{C_1}(t_1))\right]. \tag{14}$$

For the sake of clarity, let us assume the classes interface C_0/C_1 to be located between the two cells $\boldsymbol{Q}_k$ and $\boldsymbol{Q}_{k+1}$ of classes C_0 and C_1, respectively. We denote by $\boldsymbol{F}^{C_0}_{k+1/2}(\boldsymbol{Q}_k, \boldsymbol{Q}_{k+1})$ and $\boldsymbol{F}^{C_1}_{k+1/2}(\boldsymbol{Q}_k, \boldsymbol{Q}_{k+1})$ the accumulated fluxes contribution at the C_0 and C_1 sides of the C_0/C_1 interface that are extracted from the sum of fluxes in the RHS computation (see Eqs. (13) and (14)), respectively. Considering time integration with respect to the same local time-step $\Delta t_{\min}$, we can extract the accumulated fluxes contribution at both C_0 and C_1 sides

from Eqs. (13) and (14) as

$$\boldsymbol{F}^{C_0}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)=2\boldsymbol{F}^{0}_{k+1/2}\left(\boldsymbol{Q}^0_k,\boldsymbol{Q}^0_{k+1}\right) \tag{15}$$

and

$$\boldsymbol{F}^{C_1}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)=\boldsymbol{F}^{0}_{k+1/2}\left(\boldsymbol{Q}^0_k,\boldsymbol{Q}^0_{k+1}\right)+\boldsymbol{F}^{1}_{k+1/2}\left(\boldsymbol{Q}^0_k,\boldsymbol{Q}^1_{k+1}\right)\ , \tag{16}$$

where $\boldsymbol{F}^{n}_{k+1/2}=\boldsymbol{F}_{k+1/2}\left(t_n\right)$.

Now, in order to enforce the conservation (the strict exchange of fluxes $\boldsymbol{F}^{C_0}_{k+1/2}=\boldsymbol{F}^{C_1}_{k+1/2}$), we substitute a corrected flux $\boldsymbol{F}^{*}_{k+1/2}$ for the flux $\boldsymbol{F}^{0}_{k+1/2}$ in Eq. (15) so that

$$\boldsymbol{F}^{C_0}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)=\boldsymbol{F}^{C_1}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)\ , \tag{17}$$

which can be written as

$$\underbrace{2\boldsymbol{F}^{*}_{k+1/2}}_{\boldsymbol{F}^{C_0}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)}=\underbrace{\boldsymbol{F}^{0}_{k+1/2}\left(\boldsymbol{Q}^0_k,\boldsymbol{Q}^0_{k+1}\right)+\boldsymbol{F}^{1}_{k+1/2}\left(\boldsymbol{Q}^0_k,\boldsymbol{Q}^1_{k+1}\right)}_{\boldsymbol{F}^{C_1}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)}\ , \tag{18}$$

which yields the expression

$$\boldsymbol{F}^{*}_{k+1/2}=\frac{1}{2}\left[\boldsymbol{F}^{0}_{k+1/2}\left(\boldsymbol{Q}^0_k,\boldsymbol{Q}^0_{k+1}\right)+\boldsymbol{F}^{1}_{k+1/2}\left(\boldsymbol{Q}^0_k,\boldsymbol{Q}^1_{k+1}\right)\right]\ . \tag{19}$$

Using the corrected flux in Eq. (19), the conservative version of the previous multirate algorithm is obtained through the substitution of the last step with the two following steps

- Substitution of the C_0/C_1 interface fluxes ($\boldsymbol{F}^{0}_{i-1/2}$ in Fig. 7) at time t_0 by the corrected fluxes ($\boldsymbol{F}^{*}_{i-1/2}$ in Fig. 7) according to Eq. (19), so that conservation is enforced.
- Computation of class C_0 new states $\left(\boxed{\boldsymbol{Q}^2_{i-1}}\right.$ in Fig. 7, respectively$\left.\right)$ at time $t_0+\Delta T_0$ through the summation of the states and the aggregation of fluxes as defined in Eq. (9) $\left(\boxed{\boldsymbol{Q}^0_{i-1}}\right.$, $\boldsymbol{F}^{0}_{i-3/2}$ and $\boldsymbol{F}^{*}_{i-1/2}$ in Fig. 7$\left.\right)$.

The successive steps of the conservative multirate algorithm are shown in Fig. 7.

3.2.3. Generalization to higher-order ERK methods

In Section 3.2.2, we have derived a conservative multirate time integration method for the legacy synchronous ERK1 method. Considering that the general form of ERK methods in Eq. (9), with no loss of generality, we can extend the previous reasoning to all legacy synchronous ERK methods: computation of the aggregated fluxes to enforce conservation through *a posteriori* corrections (Eq. (13) to Eq. (19)). The general procedure is the following:

1. Computation of fluxes at the faces of class C_1 faces based on the known states at time t_0 $\left(\boldsymbol{F}^{0}_{k+1/2}\left(\boldsymbol{Q}^0_k,\boldsymbol{Q}^0_{k+1}\right)_{k=i-1,i-1,i}\right.$, $\boxed{\boldsymbol{Q}^0_{i-1}}$ and $\left.\boxed{\boldsymbol{Q}^0_{i}}\right.$ in Fig. 8, respectively$\left.\right)$.
2. Computation of class C_1 new states $\left(\boxed{\boldsymbol{Q}^1_{i}}\right.$ on Fig. 8, respectively$\left.\right)$ at time $t_1=t_0+\Delta T_{min}$ through the summation of the states and the aggregation of fluxes as defined in Eq. (9) advancing on RK sub-steps. At each RK sub-step $j=1,s$, the class C_0 cells are frozen to their initial state (at time t_0), and the C_0/C_1 interface fluxes are computed based on both the C_0 frozen states and the local C_1.
3. We repeat the first step considering the known states at time t_1 for class C_1 cells and the frozen states at time t_0 for class C_0 cells $\left(\boxed{\boldsymbol{Q}^0_{i-1}}\right.$ and $\left.\boxed{\boldsymbol{Q}^1_{i}}\right.$ in Fig. 8, respectively$\left.\right)$.
4. We repeat the second step to compute class C_1 new states $\left(\boxed{\boldsymbol{Q}^1_{i}}\right.$ on Fig. 8, respectively$\left.\right)$ at time $t_2=t_0+2\Delta T_{min}$ through the summation of the states and the aggregation of fluxes.
5. Construction of the accumulated fluxes $\boldsymbol{F}^{C_1}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)$ at the C_1 side of the C_0/C_1 interface during the two local time-steps marching (step 5 in Fig. 8). According to Section 3.2.2, the class C_1 final states $\left(\boxed{\boldsymbol{Q}^2_{i}}\right.$ on Fig. 8$\left.\right)$ are given by

$$\boldsymbol{Q}^{C_1}(t_2)=\boldsymbol{Q}^{C_1}(t_0)+\Delta t_{\min}\sum_{j=1}^{s}b_j\left[\mathrm{RHS}\left(\boldsymbol{Q}^{C_1}(t_{0,j})\right)+\mathrm{RHS}\left(\boldsymbol{Q}^{C_1}(t_{1,j})\right)\right], \tag{20}$$

   where $(b_j)_{1\le j\le s}$ are the Butcher coefficients. It follows that

$$\boldsymbol{F}^{C_1}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)=\sum_{j=1}^{s}b_j\left[\boldsymbol{F}^{0,j}_{k+1/2}+\boldsymbol{F}^{1,j}_{k+1/2}\right]\ , \tag{21}$$

   where $\boldsymbol{F}^{n,j}_{k+1/2}=\boldsymbol{F}_{k+1/2}\left(t_{n,j}\right)$ and $t_{n,j}=t_n+c_j\Delta t_{\min}$ (c_j, $j=1,s$ are the Butcher coefficients, see Fig. 3). According to Section 3.2.2, the corrective flux $\boldsymbol{F}^{*}_{k+1/2}$ at the C_0 side of the C_0/C_1 interface is computed from $\boldsymbol{F}^{C_1}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)$ so that conservation is enforced (by construction). It is defined by

$$\boldsymbol{F}^{*}_{k+1/2}\leftarrow\frac{1}{2}\sum_{j=1}^{s}b_j\left[\boldsymbol{F}^{0,j}_{k+1/2}+\boldsymbol{F}^{1,j}_{k+1/2}\right]\ . \tag{22}$$

6. Computation of class C_0 new states $\left(\boxed{\boldsymbol{Q}^1_{i}}\right.$ on Fig. 8, respectively$\left.\right)$ at time $t_2=t_0+2\Delta T_{min}$ through summation of states and aggregation of fluxes as defined in Eq. (9) advancing on RK sub-steps, where the fluxes $\boldsymbol{F}^{C_0}_{k+1/2}\left(\boldsymbol{Q}_k,\boldsymbol{Q}_{k+1}\right)$ at the C_0/C_1 interface are replaced, for each RK sub-step, by the corrective flux weighted by the Butcher coefficients $(b_j)_{1\le j\le s}$

$$\boldsymbol{F}^{n,j}_{k+1/2}\leftarrow b_j\boldsymbol{F}^{*}_{k+1/2}, \tag{23}$$

   to enforce the conservation property through a strict exchange of fluxes.

The successive steps of the conservative multirate algorithm are shown in Fig. 8.

3.3. Recursive synchronization

For the sake of clarity, the previous algorithms were presented for the particular case of a two-class problem ($N_c=1$) where synchronization between classes C_0 and C_1 occurs at the end of the sequence: $C_1\rightarrow C_1\rightarrow C_0$.

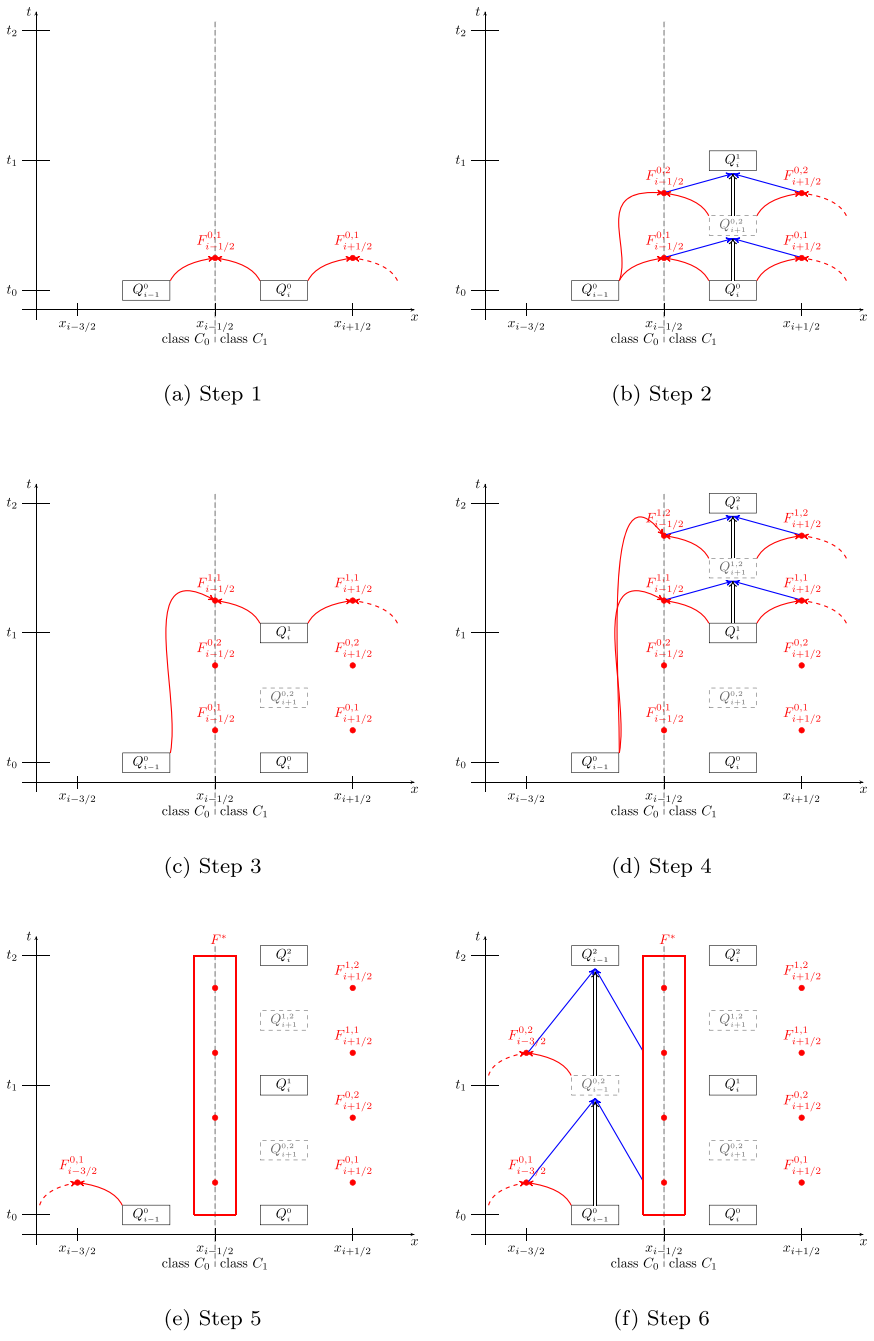
It is possible to generalize the method by noting that the global synchronization of a $(k+1)$-class case ($N_c=k$) is a nested succession of two-class (C_j-C_{j+1})-synchronizations for $j\in[0,N_c-1]$ given by the integration sequence $C_{j+1}\rightarrow C_{j+1}\rightarrow C_j$, which are synchronized every $2^{N_c+1-j}\Delta t_{\min}$, resetting the corrective fluxes to zero at the C_j/C_{j+1}.

For a recursive synchronized time integration over the global synchronization time-step $\Delta t_0=2^{N_c-1}\Delta t_{\min}$, we count a total number of $N_{seq}=2^{N_c+1}-1$ local class time integrations.

In Fig. 9, the global class synchronization sequence is shown for a four-class problem ($N_c=3$), where the local classes time integration sequence and the detailed sequence are shown on the right side figure and on the left side table, respectively.

3.4. Numerical implementation

In this sub-section, we remind the key elements of the conservative multirate explicit time integration method and we summarize its implementation in explicit compressible Navier–Stokes codes. The key elements are the following:



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

(e) Step 5 (f) Step 6

Fig. 8. Conservative multirate ERK2 algorithm between times t_0 and $t_2 = t_0 + 2\Delta t_{\min}$ with intermediate time $t_1 = t_0 + \Delta t_{\min}$ where the corrected flux is given by $\boldsymbol{F}^*_{i-1/2} = \frac{1}{2}\sum_{j=1}^{2} b_j \left[\boldsymbol{F}^{0,j}_{i-1/2} + \boldsymbol{F}^{1,j}_{i-1/2}\right]$.

1. Definition and management of local time-step classes: For practical reasons of implementation and efficiency, the N_{cell} cells of the computational domain are grouped into $1 + N_c$ time-step classes

   $$C_i,\ 0 \le i \le N_c,$$

   where the cells of the class C_i share the same time-step ΔT_i. In this work, we choose a ratio of two between the time-steps of two successive classes ($\Delta T_{N_c} = \Delta t_{\min}$, $\Delta T_i = 2\Delta T_{i+1}$) (see Section 3.1).
2. Successive nested synchronous time integrations of time-step classes' cells. The recursive sequence of synchronization is detailed in Section 3.3.
3. Enforcement of the conservation (strict exchange of fluxes) at class interfaces by use of a corrective aggregated flux (see Section 3.2.3).

The implementation of the proposed multirate time integration method is highlighted in blue in the main explicit Runge–Kutta time marching solver and the computation of the right-hand-side algorithms 1 and 2, respectively.

The first key element is achieved by adding to the explicit Runge–Kutta solver function (**computeAsyncClasses** function in algorithm 1) after reading/computing the mesh and initializing the physical fields (**readMesh** and **initialize** functions in algorithm 1) because the definition and the management the CFL number, the local physical fields (convection and acoustic speeds) and the grid-cell characteristic length.

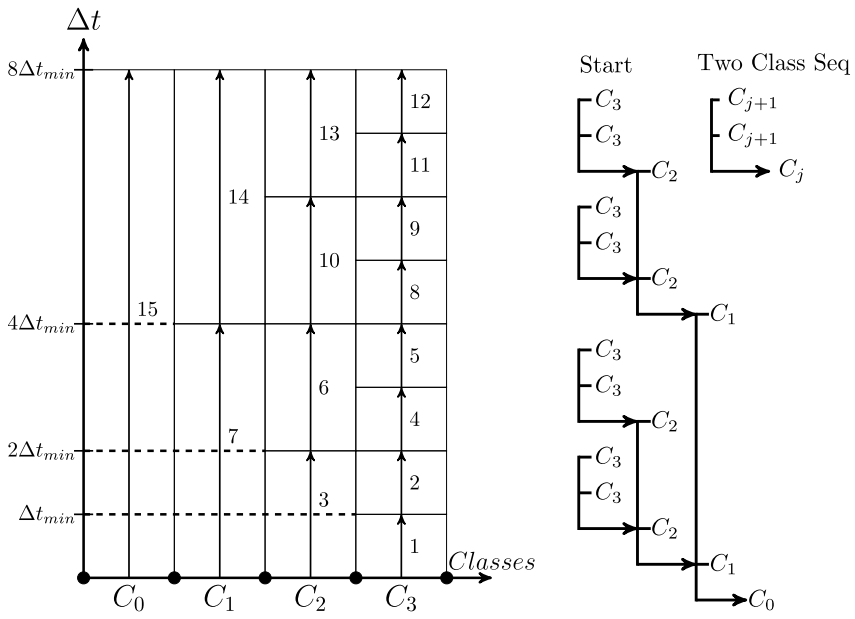


Fig. 9. Global synchronization sequence for a four-class case ($N_c = 3$).

The second key element is achieved at each global time-step through the addition of an external loop that iterates on the successive synchronous local time-step classes integration (see Fig. 9). The loop index *iclass* is passed to the right-hand-side and the flux computation functions (**computeRHS** and **CalcFluxNum** procedures, respectively) in order to limit the computations to the concerned cells of class C_{iclass} and to manage the inter-classes $C_{\text{iclass}-1}|C_{\text{iclass}}$ and $C_{\text{iclass}}|C_{\text{iclass}+1}$ boundary faces.

The last key element is achieved by applying a specific treatment to inter-classes boundary faces: if the face is on the inter-classes $C_{\text{iclass}-1}|C_{\text{iclass}}$ boundary then compute the flux and aggregate it to the corrective flux, else ($C_{\text{iclass}}|C_{\text{iclass}+1}$ boundary) consider the aggregated corrective flux.

Algorithm 1: ERK solver.

```
1: procedure ERKsolver
2:     …
3:     readMesh()
4:     initialize(Q)
5:     computeAsyncClasses(Q,mesh)
6:     …
7:     t ← 0
8:     while t < t_final do
9:         Q(t_1) ← Q(t)
10:        for iter ← 1, 2^nclasses − 1 do
11:            for irk ← 1, s^rk do
12:                iclass ← getClassId(iter)
13:                p_irk−1,iclass ← computeRhs(Q(t_irk−1,iclass), irk)
14:                Q(t_irk,iclass) ← Q(t_irk−1,iclass) + Δt_n,iclass Σ_{k=1}^{irk−1} ã_irk−1,k p_k,iclass
15:                …
16:            end for
17:            t ← t + Δt
18:        end for
19:    end while
20: end procedure
```

Algorithm 2: Computation of the RHS.

```
1: procedure computeRhs(Q(t,iclass), irk)
2:     CalcVarAtFace(iclass)
3:     CalcGradAtFace(iclass)
4:     CalcFluxNum(iclass,irk)
5:     CalcRhsFromFlux(iclass)
6: end procedure
7: procedure CalcFluxNum(iclass,irk)
8:     …
9:     for each face iface of a cell Ω_k ∈ C_iclass do
10:        if iface is an interior face then
11:            computeFlux(iface)
12:        else
13:            if iface is a C_iclass−1|C_iclass boundary face then
14:                computeFlux(iface)
15:                F*_iface ← F*_iface + b_j F^irk_iface
16:            else
17:                Use of F*_iface at the C_iclass|C_iclass+1 boundary
18:    end for each
19: end procedure
```

3.5. Advantages and drawbacks

The conservative multirate time integration method and algorithms proposed in this paper exhibit the following advantages

- Providing a potential CPU-time speedup by considerably reducing the computations and the MPI communications and exchanged data for cells integrated with larger time-steps. Indeed, considering the four classes example presented in Fig. 9, one can see that the corrective flux (but also intermediate solution from C_1 classes) is only exchanged once for class C_0 instead of eight times for its legacy single rate counterpart.
- Offering a simple implementation/integration in existing compressible Navier–Stokes codes. Indeed, the code structure of the legacy synchronous solver is maintained and, in comparison to other methods proposed in the literature (see [5]), the processing of the inter-class boundary is local and algebraically simple. In the parallel implementation proposed by Seny et al. [5], the authors introduce a bulk group of cells that overlap the inter-class boundaries and cells from sides of the latest boundary. Such an approach is efficient for two-dimensional problems but tends to be less efficient and more complex to implement for unstructured three-dimensional codes.

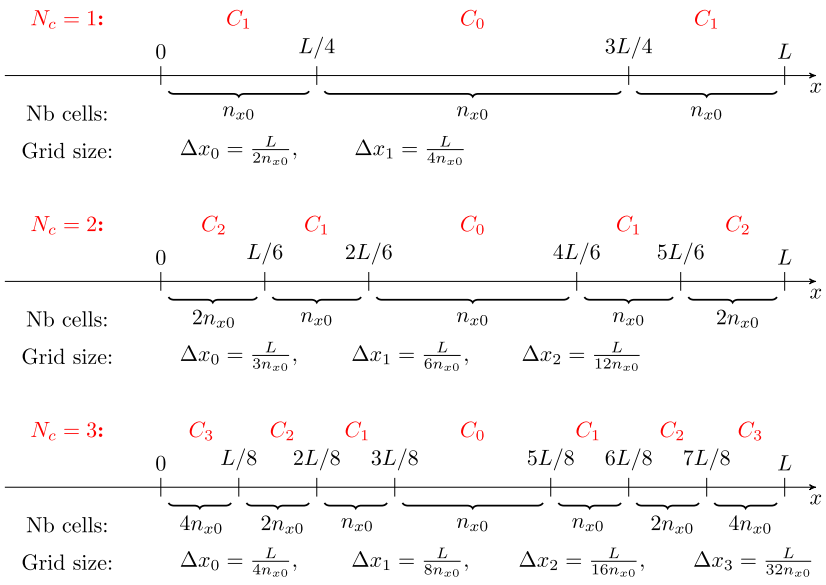


Fig. 10. Grid overview where n_{x0} is the base class C_0 number of cells.

However, the proposed method exhibits also drawbacks such as:

- The requirement of extra storage for data related to the definition and the management of local time-step classes and the aggregated corrective fluxes.
- The MPI distribution and static management of the time-step classes: since the class distribution is performed after the mesh partitioning using the ParMETIS library [28] and the different memory allocations in each MPI process, it is possible to efficiently handle load-balancing by constraining the graph-partitioning with adequate face/cv based weights before entering the time-marching solver loop. However, the dynamic variation of local time-step classes may unbalance the workload across MPI processes and may require a new mesh-partitioning to re-equilibrate the workload which is complex in the middle of the run but not impossible [29,30].

From an implementation point of view, the proposed algorithms exhibit the following drawback in an parallel computing framework

- The sequential approach which consists in, synchronously, locally integrating the cell classes with a local time-step in the framework of parallel computations on unstructured meshes: indeed, the cells of lower local time-step classes (higher local time-step) need to wait for the computation of higher local time-step classes to gather the aggregated corrective fluxes; thus, it may force MPI processes to wait while others are busy and, consequently, impact the global parallel efficiency. In the worst case scenario, if a processor contains only cells of class C_0, it will have to wait for all others to complete their time integration steps to use the C_0/C_1 correction fluxes to enforce the conservation properties.
- The dynamic load balancing is not considered and is performed only once at the beginning.

We remark that these two latest drawbacks are just "software issues" and non-blocking in a general context. Although it is not discussed and presented in this paper, the first one can be resolved with a smart domain-decomposition considering the distribution of at least one cell of each class per processor and managing the load using a weight-constrained cell using the ParMETIS library [28] domain-partitioning and, the second one can be added as described in [29,30].

With its advantages and drawbacks, the proposed conservative multirate time integration method shall potentially reach the theoretical expected gain G_{th} (see Eq. (12)). As discussed, the management of the load-balancing is more complex with this method. In particular, if the local time-step classes change during the simulation, this could be expensive in CPU time, but not "strictly impossible". Indeed, solutions to similar problems exist in the context of dynamic load balancing management for "Adaptive Mesh Refinement" applications [29,30]. In the context of dynamic evolution of cell classes over time, it should be noted that one may end up with only one time-integration class throughout the whole domain in the worst-case scenario ($G_{\text{th}} = 1$). In this case, we should have the same number of computations per cell as in the single rate case, which would be a lower bound in performance. In a massively parallel context, this could even lead to imbalance in the distribution: processes with cells undergoing time-integration class shift due to time-step reduction would see an increase in their computational load, which would even further reduce the theoretical gain. In this case, a new domain decomposition would be required in order to re-balance loads and avoid loss in efficiency. In the next section, we will numerically investigate the accuracy of the method (conservation of the legacy synchronous global space/time convergence order) using the linear advection academic benchmark test; and we will demonstrate the validation of the scheme on the Sod shock tube test with discontinuities where the conservation property is required. Finally, we will demonstrate its performances in terms of effective CPU-time speedup through well-documented DNS simulations of developed turbulent channel flows at $Re_\tau = 392$ [31].

4. Numerical results

4.1. Linear advection: Error analysis

Problem description. To investigate the accuracy and the characterization of the conservative multirate time integration method, let us consider the periodic one-dimensional linear advection of a density pulse by solving the Euler's equations on the computational domain $\Omega = [0, 1]$. The initial conditions at time $t = 0$ are the following

$$\left(U_0; V_0; W_0; P_0\right) = (10; 0; 0; 100), \tag{24}$$

$$T_0 = 100 + 10\cos(2\pi x), \tag{25}$$

$$\rho_0 = P_0/(r_{\text{gaz}} T_0), \tag{26}$$

where $r_{\text{gaz}} = 1/\gamma$ is chosen to defined the unperturbed flow at $M_0 = 0.01$.

We consider two numerical setups:

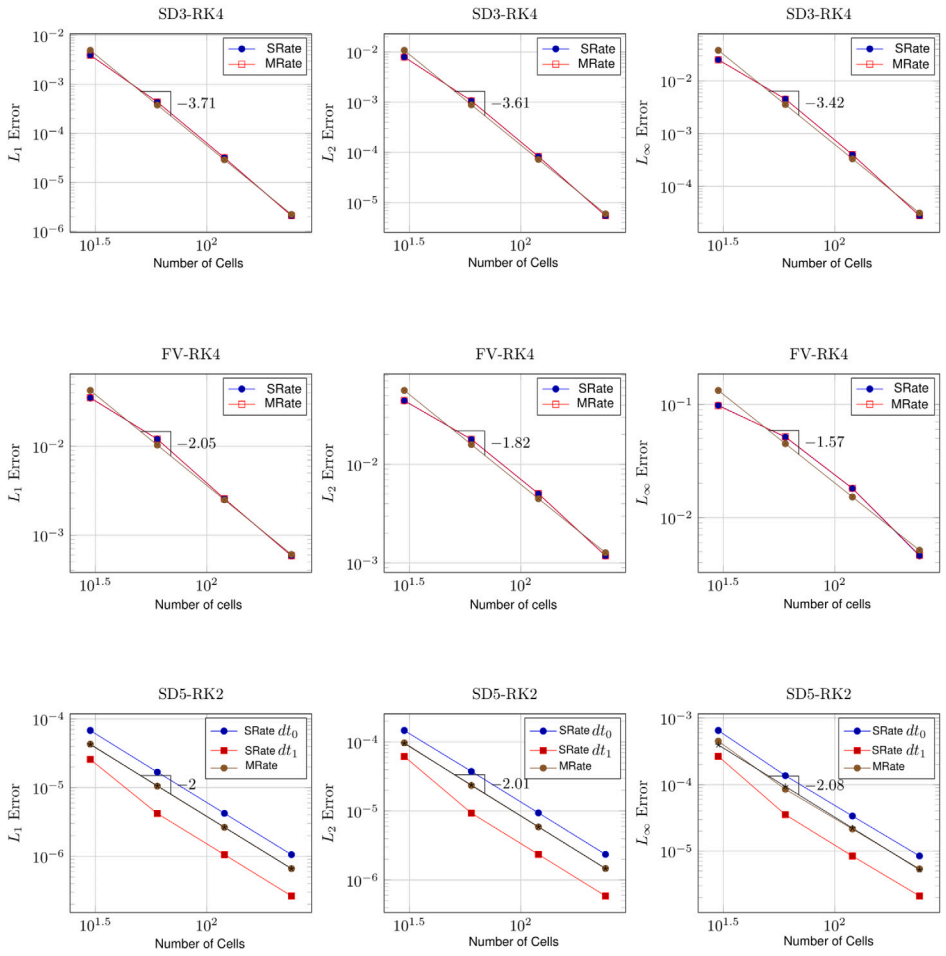


Fig. 11. Error and effective global order plots of the one dimensional linear advection of a density pulse for a two-class problem ($N_c = 1$).

1. A first numerical setup to investigate the accuracy of the method where we consider a two-class multirate time integration problems ($N_c = 1$) through the decomposition of the computational domain Ω into two local class time step domains

   - An inner domain $\Omega_0^{N_c=1} = [1/4; 3/4]$ that contains n_{x0} uniform class C_0 cells with class time step $\Delta T_0 = 2\Delta T_1$.
   - An outer domain $\Omega_1^{N_c=1} = [0/4; 1/4] \cup [3/4; 4/4]$ that contains $n_{x1} = 2n_{x0}$ uniform class C_1 cells with class time step ΔT_1.

   The domains $\Omega_0^{N_c=1}$ and $\Omega_1^{N_c=1}$ are thus, respectively, discretized into n_{x0} and $n_{x1} = 2n_{x0}$ uniform cells for $n_{x0} = 10, 20, 40, 80$. Periodic conditions are used at the boundaries. Cross tests are conducted considering on one side both finite volume (FV) and high order spectral difference (SD) methods (with the same number of cells and cell sizes) and on the other side single rate and conservative multirate time integration schemes. A sketch of the grid is shown in Fig. 10. Simulations are run for a physical time of $t_f = 0.4$ which is equivalent to four passages of the pulse through the domain.
2. A second numerical setup to investigate the behavior of non-conservative and conservative multirate time integration methods for different number of classes ($N_c = 2$ and $N_c = 3$). We remind that, without loss of generality, any problem involving more than two classes time integration can be processed as a sequence of local two-classes time integration as discussed in Section 3.4.
   We consider corresponding single rate fourth order RK4 time integration scheme (Fig. 3), the HLLC approximate solver [32] as well as *Monotonic Upwind Scheme for Conservation Laws* (MUSCL, see van Leer [33]) spatial scheme. The computational domains are the following

   - an inner domain $\Omega_0^{N_c=2} = [1/3; 2/3]$ that contains n_{x0} uniform class C_0 cells with class time step $\Delta T_0 = 4\Delta T_2$,
   - an intermediate domain $\Omega_1^{N_c=2} = [1/6; 1/3] \cup [2/3; 5/6]$ that contains $n_{x1} = 2n_{x0}$ uniform class C_1 cells with class time step $\Delta T_1 = 2\Delta T_2$,
   - an outer domain $\Omega_2^{N_c=2} = [0; 1/6] \cup [5/6; 1]$ that contains $n_{x2} = 4n_{x0}$ uniform class C_2 cells with class time step ΔT_2,

   And

   - an inner domain $\Omega_0^{N_c=3} = [3/8; 5/8]$ that contains n_{x0} uniform class C_0 cells with class time step $\Delta T_0 = 8\Delta T_3$,
   - an intermediate domain $\Omega_1^{N_c=3} = [2/8; 3/8] \cup [5/8; 6/8]$ that contains $n_{x1} = 2n_{x0}$ uniform class C_1 cells with class time step $\Delta T_1 = 4\Delta T_3$,
   - an intermediate domain $\Omega_2^{N_c=3} = [1/8; 2/8] \cup [6/8; 7/8]$ that contains $n_{x2} = 4n_{x0}$ uniform class C_2 cells with class time step $\Delta T_2 = 2\Delta T_3$,
   - an outer domain $\Omega_3^{N_c=3} = [0; 1/8] \cup [7/8; 1]$ that contains $n_{x3} = 8n_{x0}$ uniform class C_3 cells with class time step ΔT_3,

   where $n_{x0} = 10$. A sketch of the grids are shown in Fig. 10. Simulations are run for a physical time of $t_f = 1.6$ which is equivalent to twelves passages of the pulse through the domain. Physical time is increased compared to the first setup to amplify the numerical errors and compare results.

Simulations are run at constant global CFL which leads to the local definition of the cell time step

$$\Delta t_i = \text{CFL}\frac{\Delta x_i}{U_0}. \tag{27}$$

In the first simulation setup, we compare each multirate solution to the two legacies single rate solutions with global time-steps ΔT_0 and $\Delta T_1 = \Delta T_0/2$. To satisfy globally the CFL condition for both single rate simulations with time steps ΔT_0 and $\Delta T_1 = \Delta T_0/2$, we set the constant global CFL to 0.5 instead of 1.

Results and discussions. Considering the exact reference solution y^{ref}, L_q norms used to compute the error of the numerical solution y are defined by

$$\begin{aligned} L_q(y) &= \left(\sum \Delta x_i \left|y(x_i) - y^{\text{ref}}(x_i)\right|^q\right)^{\frac{1}{q}}, \\ L_\infty(y) &= \max\left|y(x_i) - y^{\text{ref}}(x_i)\right|. \end{aligned} \tag{28}$$

The effective global space/time scheme order O_q with respect to the norm q is estimated based on two numerical solutions y^1 and y^2 for constant CFL through a spatial refinement of ratio 2 and doubling the time step between y^1 and y^2 as follows

$$O_q(y) = \log_2\left(\frac{L_q(y^1)}{L_q(y^2)}\right). \tag{29}$$

Firstly, the accuracy of the conservative multirate time integration method is investigated in terms of numerical error and global scheme order through its comparison with the respective base single rate time integration method. The errors and effective global order (slope of the curves) are shown in Fig. 11 for three space–time configurations: centered finite volume/4th order Runge–Kutta (FV-RK4), 3rd order spectral difference/4th order Runge–Kutta (SD3-RK4) and 5th order spectral difference/2nd order Runge–Kutta (SD5-RK2). For all configurations, it is deduced from the slope of the lines that the global order of both algorithms is maintained using the multirate approach in comparison to the legacy single rate approach. Since the temporal errors are negligible compared to the spatial ones, we consider in the last configuration (SD5-RK2) a high 5th order spatial scheme (to decrease the spatial errors) and a low second-order temporal scheme (to increase the temporal errors) to highlight the behavior of the numerical temporal errors for the multirate approach in comparison with two legacies single rate approach results using constant time steps ΔT_0 and $\Delta T_1 = \Delta T_1/2$. As it is observed in Fig. 11, the multirate solution which integrates locally the solution with two different time steps ΔT_0 and ΔT_1 is between the two synchronous solutions with respective global time steps ΔT_0 and ΔT_1; and also, the global order is still maintained for this last configuration. These last results are coherent in the sense that the multirate solution is less accurate than the synchronous solution with the smaller time step and more accurate than the synchronous solution with the higher time step.

Secondly, the variation of the number of classes and the conservation properties effects on the numerical simulation are investigated for a fixed case combining the legacy RK4 single rate time integration method and the FV with a combination of HLLC-MUSCL schemes. We present the comparison of the legacy single rate time integration method and both conservative and non-conservation multirate variants in Fig. 12 for $N_c = 1$, $N_c = 2$ and $N_c = 3$ cases, respectively. For each case $N_c = i$ $(i = 1, 2, 3)$, the single rate simulation are run with the smallest local time step ΔT_i.

The comparison of the results for $N_c = 1$, $N_c = 2$, and $N_c = 3$ cases show that the multirate time integration method is robust and that in both cases the solution is not qualitatively affected by the wave passing through the interfaces.

Comparison between single rate and multirate shows that the multirate time integration solution is more accurate than its legacy single rate method. This can be explained by the fact the multirate solution is integrated in time locally with a greater or equal local CFL (compared to the single rate solution) that, consequently, results in reducing the diffusion errors by decreasing locally the total number of local time integrations. In the context of problems involving shocks, the application of the multirate approach can lead to a better shock capture reducing the diffusion errors locally at the shock location as it will be shown in the next subsection for the Sod Euler shock tube test.

Finally, the comparison between the conservative and non-conservative approaches shows that neglecting the conservation property of the method introduces locally at the inter-classes boundaries numerical errors that accumulate and result in a convection error that manifests itself as a spatial shift of the density pulse as shown in Fig. 12.

4.2. Euler equation: Sod shock-tube

Numerical setup. To investigate the ability of the conservative multirate time integration method to solve nonlinear gas dynamics conservation laws while preserving the accuracy and stability of the single rate time integration methods, we consider the Sod shock-tube test-case on the domain $\Omega = [0, 5]$ with the following initial conditions:

$$\boldsymbol{Q}(x,0) = \begin{cases} \boldsymbol{Q}_L & \text{if} \quad x < 2.5 \\ \boldsymbol{Q}_R & \text{if} \quad x \geq 2.5 \end{cases}, \tag{30}$$

where $\boldsymbol{Q}_L = (\rho_L, u_L, p_L) = (1, 0, 1)$ and $\boldsymbol{Q}_R = (\rho_R, u_R, p_R) = (0.125, 0, 0.1)$. The following Dirichlet boundary conditions are considered at both left and right ends as

$$\boldsymbol{Q}(x,t) = \begin{cases} \boldsymbol{Q}_L & \text{if} \quad x = 0 \\ \boldsymbol{Q}_R & \text{if} \quad x = 5 \end{cases} \tag{31}$$

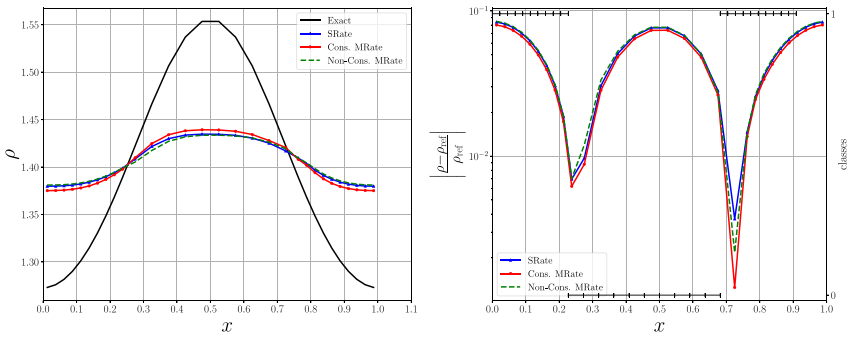
As discussed in Section 3.1, for nonlinear cases such as Euler equations the classification depends both on the cell size and the wave speeds (convective and acoustic), and its evolution in time is dynamic, meaning that it can potentially change at every global time iteration. To consider the temporal evolution of the classification of the cells, we will consider the discretization of the domain Ω into $n_x = 100$ uniform cells such that the time step computation and, by consequence, the time integration classes are fully defined by the physical properties. That is to say that in this case, the cell size is constant while the convection and acoustic speed waves are not, hence the latter triggers the cell classification. Initially, we have the following two-class problem ($n_c = 1$) defined by the initial conditions (Eq. (30)):

- $\Omega_1 = [0; 2.5]$ with class time step $\Delta T_1 = \Delta t_{\min}$,
- $\Omega_0 = [2.5; 5]$ with class time step $\Delta T_0 = 2\Delta T_1$.

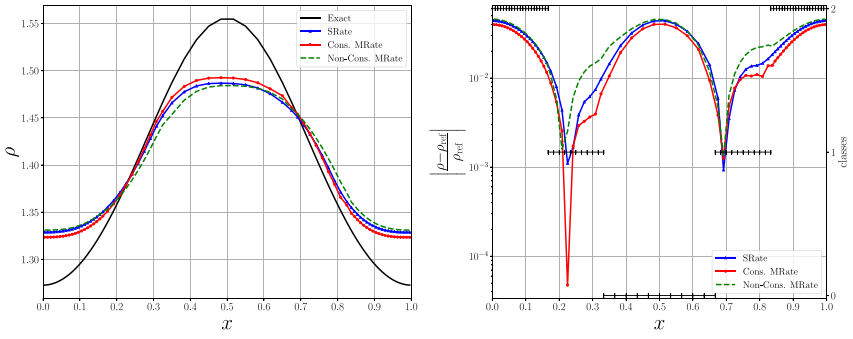
The simulations are run for a physical time $t_f = 0.8$ before the shock-wave reaches the domain's right boundary. At the final time, the numerical solution is compared to the exact solution that we consider as the reference solution in Eq. (28) for the accuracy and global convergence order analysis.

We consider corresponding single rate fourth-order RK4 time integration scheme (Fig. 3), the HLLC approximate solver [32] as well as *Monotonic Upwind Scheme for Conservation Laws* (MUSCL, see van Leer [33]) spatial scheme. The CFL number is set to $\text{CFL}_0 = 0.5$ that is close to the stability limit of the single rate time integration method to assess the behavior of both the multirate time integration method, that is to say,

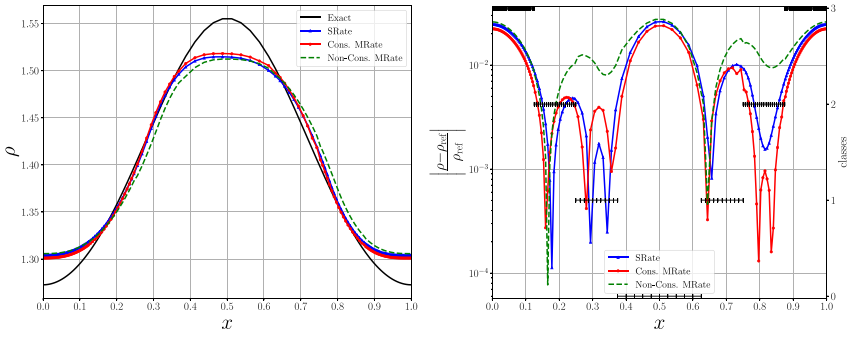
- The stability of the multirate time integration method.
- The accuracy of the numerical solution by switching from the legacy single rate to the multirate time integration method and the effect of the numerical diffusion error reduction by local increase of the CFL number.



(a) $N_c = 1$



(b) $N_c = 2$



(c) $N_c = 3$

Fig. 12. Results and error plots in black solid lines, blue solid lines with triangles, red solid lines with circles and dashed green lines the exact, RK4 single rate, conservative and non-conservative RK4 multirate simulation, respectively. Respective meshes are displayed in the error plots in black and positioned with respect to its local time integration class right respect to the right y-axis..

Results and discussions. The numerical solutions and errors plots are shown in Figs. 13–16 for velocity, density, energy and pressure respectively. In the aforementioned figures, numerical solutions are shown for the legacy single rate time integration simulation with the maximum allowed CFL and the multirate time integration approach with a local class CFL varying between the smallest and highest local cell CFL. From a qualitative point of view, very good agreement between the legacy single rate and multirate approaches is observed.

From a quantitative point of view, we see from the errors plots that the multirate approaches remain at least as accurate than the legacy single rate approach and even more accurate locally in the shock-discontinuity region as the solution is integrated locally in time with a greater CFL that results in reducing the diffusion errors by decreasing locally the total number of local time integrations.

In Fig. 17 we exhibit the dynamic evolution of the domain local time classes partition where solid blue line plot represents the density profile along the shock tube and red plots represent the computational mesh by intervals positioned at the mesh cells class id with respect to the right y-axis. Considering that the mesh is uniform, the local cell CFL varies with respect to the physical properties. We can clearly observe the evolution of the cells' class id and, *in fine*, the evolution of the inter-class interface that moves to the right with respect to shock front movement.

4.3. DNS of developed turbulent channel flows at $Re_\tau = 392$ [31]

To investigate wall turbulence properties, turbulent models, and numerical methods, the Direct Numerical Simulation (DNS) of Turbulent Channel Flows (TCF) have been widely studied from the first DNS performed by Kim et al. [34] ($Re_\tau = 180$) to the later DNS (see Alfonsi et al. [35] for a review and an outline of the works on DNS of TFC). In this paper, we consider the validation and the performance analysis of the presented multirate time integration method against the DNS of TCF conducted by Moser et al. [31] at ($Re_\tau = 392$). The simulation database provided by Moser et al. [31] is being considered as a reference solution.

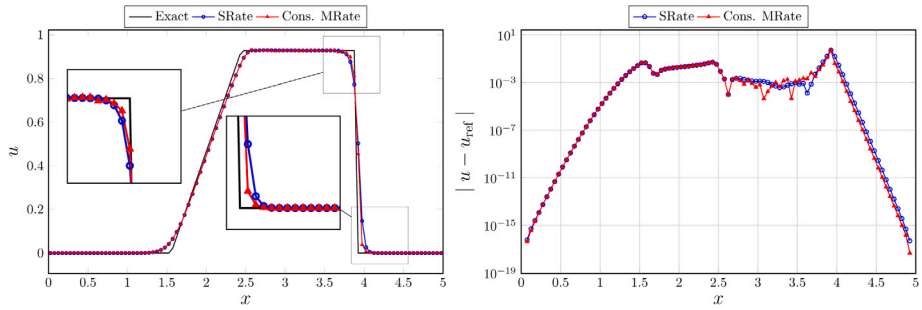


Fig. 13. Numerical solution and error plots for velocity using 4th order RK4 time integration and MUSCL space discretization schemes at CFL = 0.5.

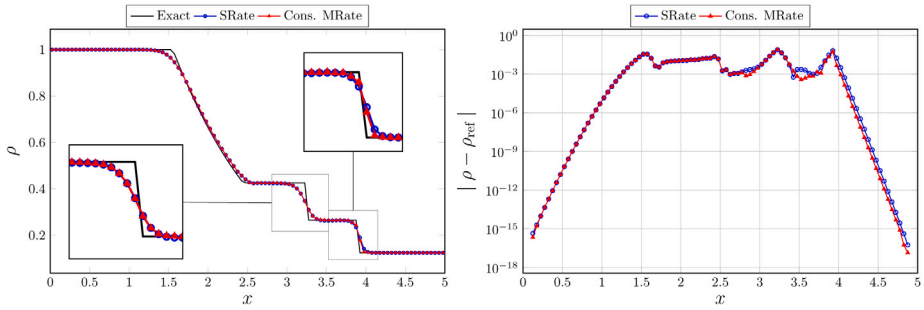


Fig. 14. Numerical solution and error plots for density using 4th order RK4 time integration and MUSCL space discretization schemes at CFL = 0.5.

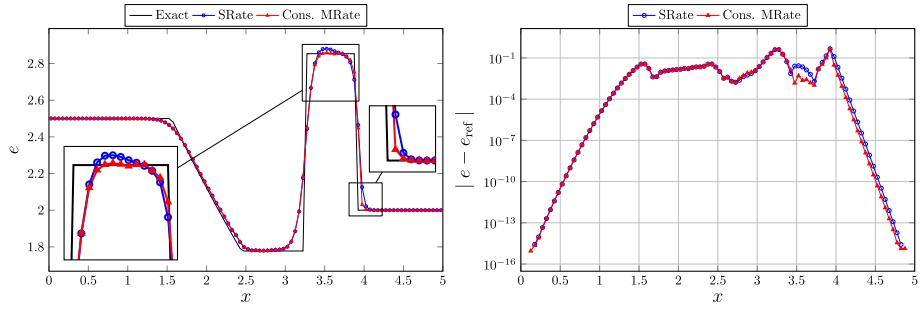


Fig. 15. Numerical solution and error plots for energy using 4th order RK4 time integration and MUSCL space discretization schemes at CFL = 0.5.

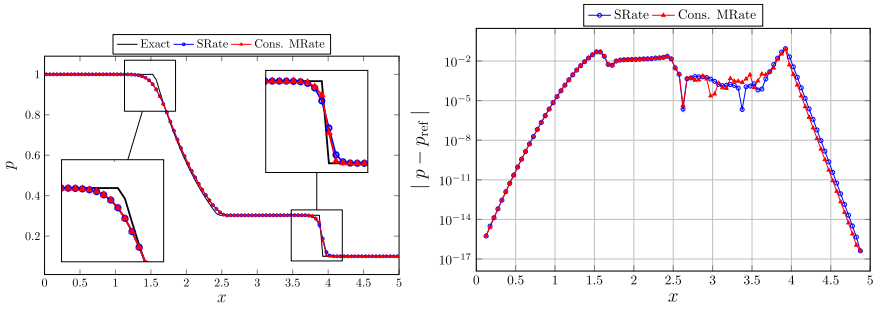


Fig. 16. Numerical solution and error plots for pressure using 4th order RK4 time integration and MUSCL space discretization schemes at CFL = 0.5.

Problem description. The DNS of TCF consists of conducting a high fidelity simulation of a three-dimensional fluid flow moving between two parallel infinite flat plates in one dominant direction.

The parallel infinite plates are modeled by the means of a finite bounded box $\Omega = [0; 6] \times [-1; 1] \times [-1.5; 1.5]$ ($L_x = 6,\ L_y = 2,\ L_z = 3$) with bottom and top no-slip iso-thermal wall boundary condition (in y-direction) and periodic boundary conditions in both stream-wise and span-wise directions (x-direction and z-direction respectively).

Physical parameters. The density, pressure and temperature follow the ideal gas law given in Eq. (4) with a specific heat-ratio set to $\gamma = 1.4$. The fluid viscosity is given by the following viscosity power law

$$\mu = \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^{0.76}, \tag{32}$$

where μ, T, μ_{ref}, T_{ref} are respectively the viscosity, the temperature, the reference viscosity and temperature. These quantities are set to $\mu_{\text{ref}} = 2.898551 \cdot 10^{-5}$, $T_{\text{ref}} = 1.0$.

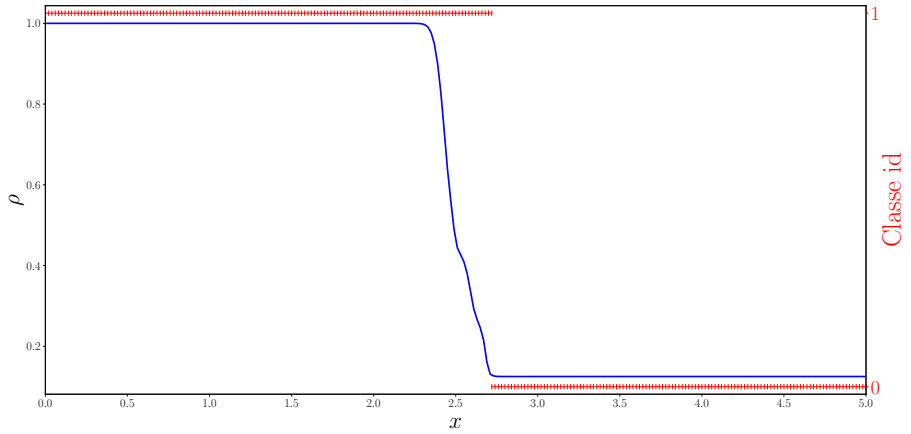


Fig. 17. Animation of the evolution in time of the inter-class interface with respect to the shock interface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To characterize the turbulent flow properties, we consider the following space and time averaging of a given field function f

$$\overline{f}(y,t) = \frac{1}{2L_xL_z}\iint \big(f(x,y,z,t)+f(x,-y,z,t)\big)\mathrm{d}x\mathrm{d}z\ ,\ y\in[0,L_y/2]\ , \quad (33)$$

$$\langle f\rangle_t(x,y,z) = \frac{1}{N_\mathrm{s}}\sum_{i=0}^{N_\mathrm{s}-1} f(x,y,z,t_\mathrm{s}+i\Delta t_\mathrm{s})\ , \quad (34)$$

where N_s, t_s and Δt_s denote the number of record-samples, the starting time of sample recording and the physical time interval between two records, respectively. Through the use of Eqs. (33) and (34), the root-mean-square (rms) fluctuation of a field function f is given by

$$f' = \sqrt{\langle \overline{f'^2}\rangle_t}\ , \quad (35)$$

where $f' = f - \overline{f}$ denotes the field function f fluctuation. Considering Eq. (33), we can define the wall quantities: $\rho_\mathrm{w} = \overline{\rho}(y_\mathrm{w})$, $\mu_\mathrm{w} = \overline{\mu}(y_\mathrm{w})$, $T_\mathrm{w} = \overline{T}(y_\mathrm{w})$ and $\tau_\mathrm{w} = \overline{\mu\frac{\partial u}{\partial y}}|_{(y=y_\mathrm{w})}$; and the bulk quantities:

$$\begin{aligned} \rho_b &= \frac{1}{L_xL_yL_z}\int_V \rho\mathrm{d}V, & (\rho u)_b &= \frac{1}{L_xL_yL_z}\int_V \rho u\mathrm{d}V, \\ Re_b &= \frac{\rho_b u_b L_y}{\mu_\mathrm{w}}, & M_b &= \frac{u_b}{c_\mathrm{w}} = \frac{u_b}{\sqrt{\gamma r_\mathrm{gaz}T_\mathrm{w}}}, \end{aligned} \quad (36)$$

where ρ_b, $(\rho u)_b$, Re_b, M_b denote the bulk: density, momentum, Reynolds number and Mach number, respectively.

It follows that the definition of the wall friction velocity u_τ, the friction Reynolds number Re_τ, and skin friction coefficient C_f are given by:

$$u_\tau = \sqrt{\frac{\tau_\mathrm{w}}{\rho_\mathrm{w}}}, \quad Re_\tau = \frac{\rho_\mathrm{w}u_\tau L_y}{2\mu_\mathrm{w}} \quad \text{and} \quad C_f = \frac{2\tau_\mathrm{w}}{\left(\rho_\mathrm{w}u_\tau\right)^2}. \quad (37)$$

The normalized mean stream-wise velocity profile U^+ is defined as

$$U^+\left(y^+\right) = \frac{\langle\overline{u}\rangle_t}{u_\tau}, \quad y^+ = yRe_\tau, \quad y\in[0,h]\ , \quad (38)$$

where the friction velocity u_τ is computed based on the time-averaged wall density $\langle\rho_\mathrm{w}\rangle_t$ and wall shear-stress $\langle\tau_\mathrm{w}\rangle_t$. The parameter h denotes the channel half-width and is equal to unity.

Near the wall (viscous sub-layer, $y^+ < 5$), the normalized mean stream-wise velocity profile U^+ follows a linear law, whereas near the center of the channel it follows a log law profile defined as

$$\begin{aligned} &U^+\left(y^+\right) = y^+\ ,\ \text{if } y^+ < 5\ , \\ &U^+\left(y^+\right) = \frac{1}{\kappa}\ln\left(y^+\right) + C^+\ ,\ \text{otherwise} \end{aligned} \quad (39)$$

where κ is the Von Kàrmàn constant set to $\kappa = 0.41$ and C^+ is a constant set to $C^+ = 5.0$.

Finally, the normalized *rms* velocity in each direction are defined as follows

$$\begin{aligned} u'\left(y^+\right) &= \frac{\sqrt{\langle\overline{u^2}\rangle_t - \langle\overline{u}\rangle_t^2}}{u_\tau}, \quad v'\left(y^+\right) = \frac{\sqrt{\langle\overline{v^2}\rangle_t - \langle\overline{v}\rangle_t^2}}{u_\tau} \\ &\text{and } w'\left(y^+\right) = \frac{\sqrt{\langle\overline{w^2}\rangle_t - \langle\overline{w}\rangle_t^2}}{u_\tau}. \end{aligned} \quad (40)$$

Simulation parameters. We solve the Navier–Stokes equations using both FV (fully-centered, $\alpha = 1$ in Eq. (5)) and third-order SD spatial methods and both single rate and multirate third-order Runge–Kutta ($RK3SSP$) time integration methods. To maintain a constant mass flow rate, the following time-varying bulk source terms are added to the governing equations (right-hand-side in Eq. (1))

$$\begin{aligned} S^\mathrm{T} &= \left[\begin{array}{ccc} 0 & \overrightarrow{b_f} & \overrightarrow{b_f}.\vec{u} \end{array}\right],\ \overrightarrow{b_f} = \left[\begin{array}{ccc} \frac{(\rho u)_b - (\rho u)_\mathrm{target}}{t_\mathrm{diff}} & 0 & 0 \end{array}\right] \\ &\text{and } t_\mathrm{diff} = \frac{\rho_b L_x}{(\rho u)_b} \end{aligned} \quad (41)$$

The term t_diff denotes the diffusion time and $(\rho u)_\mathrm{target}$ is an imposed constant targeted bulk momentum which is set to $(\rho u)_\mathrm{target} = 0.2$. The simulation is run for a total physical time $t_{sim} \approx 20t_\mathrm{diff}$ such that the first five diffusion times are needed to reach a stationary state and the last $15t_\mathrm{diff}$ to collect the data and compute the different statistics.

For both FV and SD simulations, the mesh counts $[N_x, N_y, N_z] = [127, 159, 127]$ grid cells in the stream-wise, the normal to walls and the span-wise direction, respectively. The mesh is built with a uniform grid spacing $\Delta x^+ \approx 18.5$ and $\Delta z^+ \approx 9.25$ expressed in wall unit in the stream-wise and the span-wise directions, respectively. In the normal to walls direction, the grid spacing varies in the range $\Delta y^+ \approx [0.4 - 24.4]$ and the value of the first grid point expressed in wall unit is $y_w^+ = 0.2$ (computed based on the distance between the first grid centroid and the two parallel plates for both FV and SD methods).

For the sake of data collection and statistics computation, we conducted the simulations with a constant minimum local time step Δt_min. In this configuration, we consider the flow to be incompressible (Ma 0.2) and, thus, we can assume that the stability condition is dominated by the grid size. Under the condition of incompressibility of the flow and considering that the time step is quasi-constant per cell, we have conducted a preliminary stability study conditioned by the variability of the time step for the single rate approach. By ramping up the time step we were able to deduce the stability limit Δt_{limit} and we fixed in purpose $\Delta t_\mathrm{min} = \Delta t_{limit} = 5e-4$. To reach $t_{sim} \approx 20t_\mathrm{diff}$, the simulations were run for a total of $niter_{srate} = 4.000.000$ single rate iterations which is equivalent to $niter_{mrate} = 500.000$ global synchronized multi-rate iterations and corresponding to a total physical time $t_{phy} = 2000$.

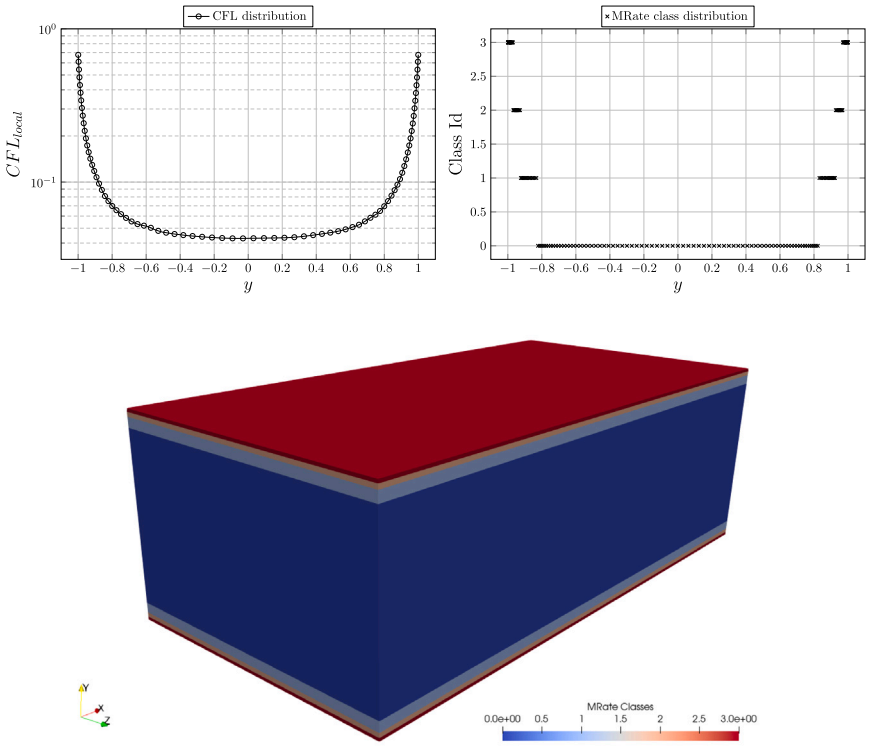


Fig. 18. Plot of the local CFL and the class distributions along the Y axis.

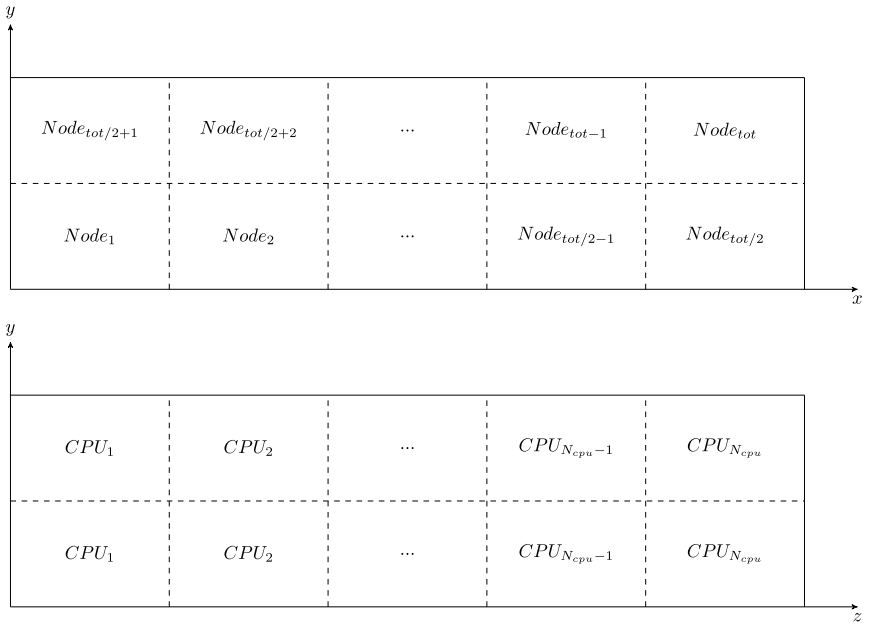


Fig. 19. Domain decomposition.

Considering the flow low Mach number and the fixed $\Delta t_{\min}$, we obtain a domain decomposition based on the CFL definition provided in Eq. (10) which is inversely proportional to the grid size as discussed in Section 3.1. This leads to a maximum local $\text{CFL}_{max} \approx 0.68$ and a minimum local $\text{CFL}_{min} \approx 0.043$ close to the wall and channel center where the grid sizes in Y-direction are minimum and maximum, respectively. Finally, the definition and management of time step classes, presented in Section 3.1, yields to a four-class multirate time integration problem ($N_c = 3$) as follows multirate time integration classes:

- $\Omega_3 = [-1; -0.97] \cup [0.97; 1]$ with class time step $\Delta T_3 = \Delta t_{\min}$ composed of 580644 cells with local CFL $\in \left[\text{CFL}_{max}/2, \text{CFL}_{max}\right]$,
- $\Omega_2 = [-0.97; -0.925] \cup [0.925; 0.97]$ with class time step $\Delta T_2 = 2\Delta t_{\min}$ composed of 354838 cells with local CFL $\in \left[\text{CFL}_{max}/4, \text{CFL}_{max}/2\right]$,
- $\Omega_1 = [-0.925; -0.825] \cup [0.825; 0.925]$ with class time step $\Delta T_1 = 4\Delta t_{\min}$ composed of 419354 cells with local CFL $\in \left[\text{CFL}_{max}/4, \text{CFL}_{max}/8\right]$,

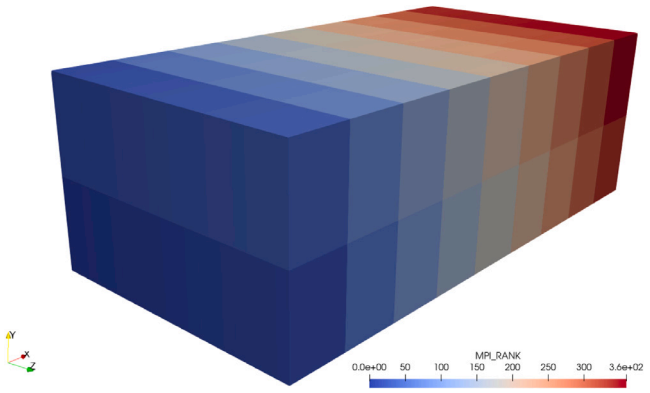
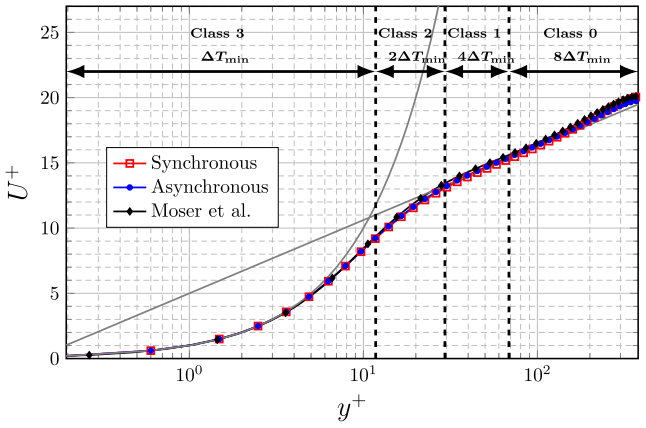
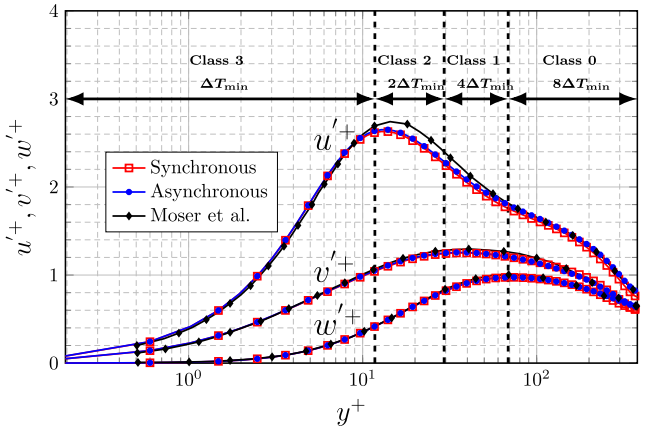


Fig. 20. Domain decomposition for $Node_{tot} = 20$ nodes and $N_{cpu} = 18$ cores per node.



(a) Mean stream-wise velocity profiles normalized in wall units. Grey lines represent the law of the wall described in eq. (39).



(b) Root-mean-square profiles normalized in wall units.

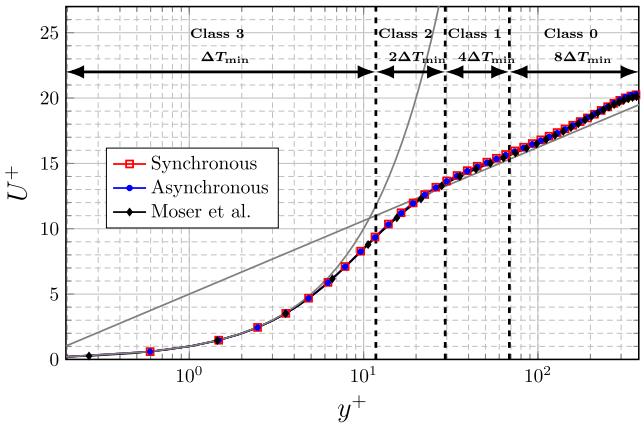
Fig. 21. Comparison of Synchronous and Asynchronous RK3SSP results with Moser et al. [31] DNS results using the centered-based finite-volume space discretization.

- $\Omega_0 = [-0.825; 0.825]$ with class time step $\Delta T_0 = 8\Delta t_{\min}$ composed of 1209675 cells with local CFL $\in \left[\text{CFL}_{max}/8, \text{CFL}_{max}/16\right]$ (we remark that $\text{CFL}_{min} \in \left[\text{CFL}_{max}/8, \text{CFL}_{max}/16\right]$).

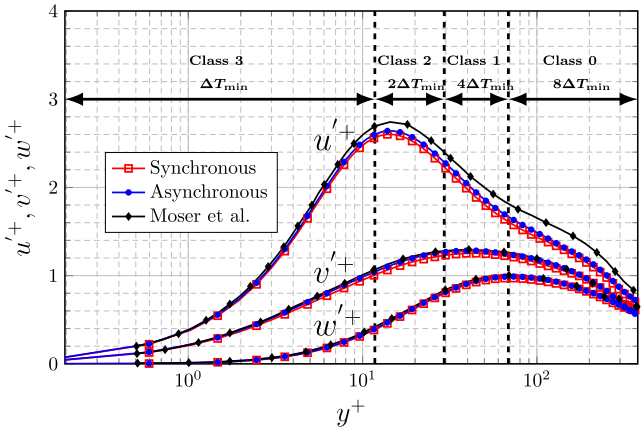
The local time-step for each time integration class is set considering the lowest local CFL and Eq. (10). In Fig. 18, we show the local CFL and the multirate class distributions along the Y-axis.

For efficiency reasons, we would like to

- overcome the drawback related to the sequential approach of integrating (cell classes/locally) by imposing that all cores contain cells of each class to avoid forcing some cores with higher local time-steps to wait for other cores with lower classes time-steps to catch-up as discussed in Section 3.5.
- Limit the number of inter-nodes communications preferring to increase communications of cores shared by the same node.
- Consider the symmetry of the multirate problem: same class distribution in X and Z directions and symmetric decomposition of the classes in the Y direction.



(a) Mean stream-wise velocity profiles normalized in wall units. Grey lines represent the law of the wall described in eq. (39).



(b) Root-mean-square profiles normalized in wall units.

Fig. 22. Comparison of Synchronous and Asynchronous RK3SSP results with Moser et al. [31] DNS results using the third-order spectral-differences space discretization.

For this purpose, we considered a symmetric orthogonal two-dimensional grid decomposition into a total $Node_{tot}$ nodes in the X–Y plane (two nodes per dimension Y $Node_{tot}$ and $Node_{tot}/2$ nodes in X direction) and a one dimensional domain decomposition in the Z direction into the constant total number of cores per nodes N_{cpu} as shown in Fig. 19.

With such a domain decomposition and considering the symmetric multirate class distribution shown in Fig. 18, we can see that each core contain cells from all classes and that pair of cells at each side of the inter-core boundaries share the same multirate class Id.

In this study, we consider a total number of $Node_{tot} = 20$ nodes and a number of $N_{cpu} = 18$ cores per node for a total of 360 cores. It follows, that some cores will have slightly more cells than others but will not globally penalize the overall efficiency as demonstrated below in the results paragraph. A sketch of the domain decomposition is shown in Fig. 20

Results and discussions. We first focus on the validation of the multirate method by comparing the obtained results with the legacy synchronous counterpart. We rely on the analysis of the mean-flow characterization (as defined in Eq. (39)). That is to say, the mean stream-wise velocity profiles and the root-mean-squared velocity profiles as shown in Figs. 21 and 22, respectively. In Figs. 21 and 22, we show that the mean flow and the root-mean-squared characteristics for both synchronous and multirate approaches are well recovered through the comparison of the obtained results with the reference DNS data provided by [31]. Regarding the validation of the proposed method, we observe a good quantitative agreement between the multirate time integration approach and the legacy synchronous using both the finite volume and the spectral differences solvers. This demonstrates the robustness of the method, as the obtained results preserve the order of accuracy of its legacy synchronous method, independently from the choice of the spatial discretization method.

Secondly, we are interested in the performance analysis of the proposed conservative multirate time integration method in terms of computational time speedup thanks to the reduction of the total number of time integrations per cell (see Eq. (12)) and the MPI-communications. For this specific test case and both domain and class decompositions we can estimate MPI communications and exchanged data ratio between single rate and multirate time integration approaches to be similar to the theoretical gain $G_{\text{th}} = 2.53$. For the sake of the performance analysis, let us define the following reference CPU-time per iteration

$$t^*_{\text{cpu}} = \frac{t_{\text{cpu}} n_{\text{procs}}}{n_{\text{dof}} n_{\text{cells}} n_{\text{iter}}}, \tag{42}$$

where t_{cpu}, n_{procs}, n_{dof}, n_{cells}, n_{dof} and n_{iter} denote the cpu walltime, the total number of processors, the total number of cells, the number of degrees of freedoms per cell (one and twenty-seven for the FV and SD3 methods, resp.) and the total number of synchronous iterations, respectively. In Fig. 23, we show the comparison of normalized CPU-time per iteration defined between multirate and synchronous approaches for the four main subroutines of the RHS computation:

- **calcVarAtFace_{FV,SD3}:** field data extrapolation at the cell faces,

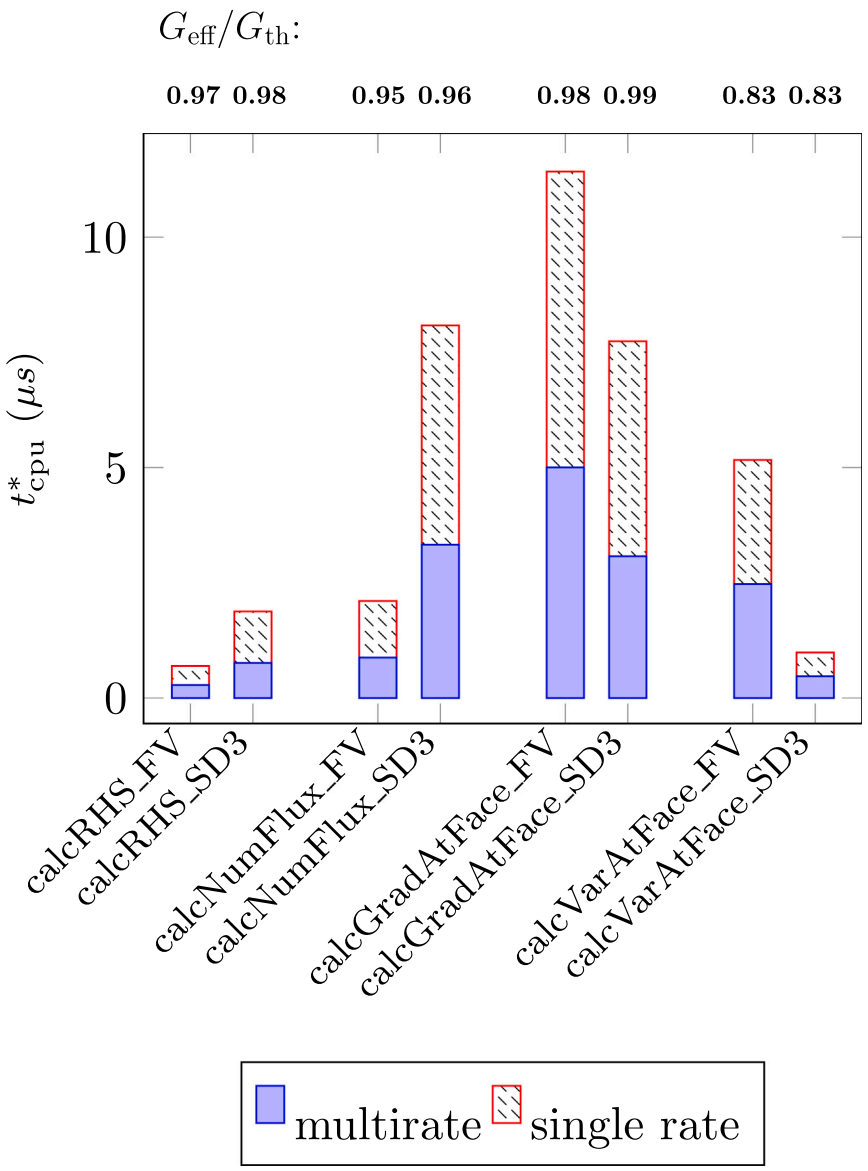


Fig. 23. Comparison of the reference CPU-time (see Eq. (42)) for the four main subroutines of the RHS computation. The effective gain G_{eff} normalized by the theoretical gain $G_{\text{th}} = 2.53$ (see Eq. (12)) is displayed on the top.

- **calcGradAtFace_{FV,SD3}:** gradient of field data extrapolation at cell faces,
- **calcNumFlux_{FV,SD3}:** computation of numerical fluxes,
- **calcRHS_{FV,SD3}:** computation of the RHS from the numerical fluxes.

The previous subroutines represent $\approx 90\%$ of the overall computational time. Only **calcVarAtFace_{FV,SD3}:** and **calcGradAtFace_{FV,SD3}:** contains MPI-communication that exchanges the missing neighbors cells data at the inter-processors boundaries necessary to compute the states and the gradients at the cell face.

The effective gain G_{eff} normalized by the theoretical gain $G_{\text{th}} = 2.53$ (see Eq. (12)) is displayed on the top of Fig. 23. It can be observed that the proposed multirate time integration method is very efficient as the ratio $G_{\text{eff}}/G_{\text{th}}$ is close to unity for the SD3 scheme when the time step classes are well managed (with an adequate domain decomposition such as explained in Section 3.5) and the workload per processor well equilibrated in a massive-parallel framework. CPU times for a complete iteration (single rate) enclosing MPI communications are 828 µs and 0.024 s for the FV (one DoF per cell) and SD3 (27 DoF per cell) methods, respectively.

5. Conclusion and perspectives

In this paper, we proposed a *conservative multirate explicit time integration* method that can significantly improve the efficiency of explicit Runge–Kutta time integration of any hyperbolic system and especially addressed to compressible flows. The efficiency (in terms of effective CPU time speedup) and robustness have been assessed through the study of linear advection and one-dimensional Sod shock tube benchmark-tests and, the direct numerical simulation of a developed turbulent channel flow at $Re_\tau = 392$.

The investigation of the method's accuracy for the linear advection test case has shown that the global convergence order of single rate time integration is maintained compared to its conservative multirate version. We have also checked the conservation property through the one-dimensional Sod shock tube and demonstrated that the multirate method can be successfully applied in a nonlinear context involving shocks and with varying local cell time step and dynamic evolution of the time integration class decomposition. Through the direct numerical simulation of the turbulent channel flow, we have finally proved the versatility of the integration method with the application to two different spatial discretization methods (FV and SD).

Regarding the improvement of explicit Runge–Kutta methods efficiency in terms of CPU time speedup, we have demonstrated in the case of the DNS of turbulent channel flow that a considerable 2.48 speedup is reached for the specific configuration (physical and numerical parameters) described in the previous subsection.

A direct perspective would consist in considering dynamic management of time step classes that consider a redefinition of the time step classes partition in a massively parallel framework. This task is highly challenging in an HPC context as the dynamic management of classes may unbalance the processors' workload and potentially require a new domain decomposition to optimize the computational efficiency. Indeed, solutions to similar problems exist in the context of dynamic load balancing management for "Adaptive Mesh Refinement" applications [29,30]. Combining "Adaptive Mesh Refinement" and multirate methods in addition to the "dynamic management of time step classes" would enable solving very efficiently strongly nonlinear problems using explicit time integration methods where the time step is highly dependent on the physical fields and thus highly variable in space and time such as combustion or fast transient dynamics problems.

CRediT authorship contribution statement

Ramzi Messahel: Formal analysis, Conceptualization, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Gilles Grondin:** Conceptualization, Methodology, Writing – review & editing. **Jérémie Gressier:** Supervision, Methodology, Writing – review & editing, Funding acquisition. **Julien Bodart:** Supervision, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank Athanasios Boutsikakis and Radouan Boukharfane for many helpful discussions and for carefully reviewing a first draft of the paper. This research was supported by the French Ministry of Defense through a financial support of the DGA. This work was performed using HPC resources from GENCI-IDRIS and GENCI-CINES on Jean Zay and Occigen, respectivly (Grant A0082A07178); and CALMIP on Olympe (Grant 2020-p1425).

References

[1] Bodart J, Gressier J, Lamouroux R, Grondin G, Grabner F. A fair performance comparison between high order and classical finite volume schemes for unstructured grids and complex turbulent flows. In: ECCOMAS 2016 - European congress on computational methods in applied sciences and engineering - Crete Island (Greece), June; 2016.

[2] Courant R, Friedrichs K, Lewy H. Über die partiellen Differenzengleichungen der mathematischen Physik. Math Ann 1928;100(1):32–74. http://dx.doi.org/10.1007/BF01448839.

[3] Constantinescu E, Sandu A. Multirate timestepping methods for hyperbolic conservation laws. J Comput Sci 2007;33:239–78. http://dx.doi.org/10.1007/s10915-007-9151-y.

[4] Puppo G, Semplice M. Numerical entropy and adaptivity for finite volume schemes. Commun Comput Phys 2011;10(5):1132–60. http://dx.doi.org/10.4208/cicp.250909.210111a.

[5] Seny B, Lambrechts J, Toulorge T, Legat V, Remacle J. An efficient parallel implementation of explicit multirate Runge–Kutta schemes for discontinuous Galerkin computations. J Comput Phys 2014;256:135–60. http://dx.doi.org/10.1016/j.jcp.2013.07.041.

[6] Osher S, Sanders R. Numerical approximations to nonlinear conservation laws with locally varying time space grid. Math Comp 1983;43:321–36. http://dx.doi.org/10.1090/S0025-5718-1983-0717689-8.

[7] Maurits N, van der Ven H, Veldman A. Explicit multi-time stepping methods for convection-dominated flow problems. Comput Methods Appl Mech Engrg 1998;157(1):133–50. http://dx.doi.org/10.1016/S0045-7825(98)80002-9, http://www.sciencedirect.com/science/article/pii/S0045782598800029.

[8] Dawson C, Kirby R. High resolution schemes for conservation laws with locally varying time steps. SIAM J Sci Comput 2000;22(6):2256–81. http://dx.doi.org/10.1137/S1064827500367737.

[9] Hundsdorfer W, Mozartova M, Savcenco V. Analysis of explicit multirate and partitioned Runge–Kutta schemes for conservation laws. CWI report. MAS-E, vol. 0715, Centrum voor Wiskunde en Informatica; 2007.

[10] Schlegel M, Knoth O, Arnold M, Wolke R. Multirate Runge–Kutta schemes for advection equations. J Comput Appl Math 2009;226(2):345–57. http://dx.doi.org/10.1016/j.cam.2008.08.009.

[11] Hundsdorfer W, Ketcheson DI, Savostianov I. Error analysis of explicit partitioned Runge-Kutta schemes for conservation laws. J Sci Comput 2015;63(3):633–53. http://dx.doi.org/10.1007/s10915-014-9906-1.

[12] Jeanmasson G, Mary I, Mieussens L. Explicit local time stepping scheme for the unsteady simulation of turbulent flows. In: ICCFD10 - Tenth international conference on computational fluid dynamics - Barcelona (Spain), July; 2018.

[13] Bermejo-Moreno I, Bodart J, Larsson J, Barney B, Nichols J, Jones S. Solving the compressible Navier-Stokes equations on up to 1.97 million cores and 4.1 trillion grid points. In: Proceedings of the international conference on high performance computing, networking, storage and analysis. SC '13, New York, NY, USA: ACM; 2013, p. 62:1–62:10. http://dx.doi.org/10.1145/2503210.2503265, http://doi.acm.org/10.1145/2503210.2503265.

[14] Khalighi Y, Nichols JW, Lele SK, Ham F, Moin P. Unstructured large Eddy simulation for prediction of noise issued from turbulent jets in various configurations. In: 17th AIAA/CEAS aeroacoustics conference (32nd AIAA aeroacoustics conference), 05-08 June. 2011, http://dx.doi.org/10.2514/6.2011-28, https://arc.aiaa.org/doi/pdfplus/10.2514/6.2011-2886.

[15] Brès GA, Nichols JW, Lele SK, Ham FE. Towards best practices for jet noise predictions with unstructured large Eddy simulations. In: 42nd AIAA fluid dynamics conference and exhibit 25-28 June. 2012, http://dx.doi.org/10.2514/6.2012-2965, https://arc.aiaa.org/doi/10.2514/6.2012-2965.

[16] George KJ, Lele SK. Large Eddy simulation of airfoil self-noise at high Reynolds number. In: 22nd AIAA/CEAS aeroacoustics conference, 30 May - 01 June. 2016, http://dx.doi.org/10.2514/6.2012-2965, https://arc.aiaa.org/doi/10.2514/6.2012-2965.

[17] Sáez-Mischlich G, Grondin G, Bodart J, Jacob MC. Assessment of LES using sliding interfaces. In: García-Villalba M, Kuerten H, Salvetti MV, editors. Direct and large Eddy simulation XII. Cham: Springer International Publishing; 2020, p. 405–10.

[18] Kopriva DA. A staggered-grid multidomain spectral method for the compressible Navier–Stokes equations. J Comput Phys 1998;143(1):125–58. http://dx.doi.org/10.1006/jcph.1998.5956, https://www.sciencedirect.com/science/article/pii/S0021999198959563.

[19] Liu Y, Vinokur M, Wang Z. Spectral difference method for unstructured grids I: Basic formulation. J Comput Phys 2006;216(2):780–801. http://dx.doi.org/10.1016/j.jcp.2006.01.024, https://www.sciencedirect.com/science/article/pii/S0021999106000106.

[20] Sun Y, Wang Z, Liu Y. High-order multidomain spectral difference method for the Navier–Stokes equations on unstructured hexahedral grids. Commun Comput Phys 2006;2(2):310–33.

[21] Van den Abeele K, Lacor C, Wang Z. On the stability and accuracy of the spectral difference method. J Sci Comput 2008;37(2):162–88. http://dx.doi.org/10.1007/s10915-008-9201-0.

[22] Shu C, Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. J Comput Phys 1988;77(2):439–71. http://dx.doi.org/10.1016/0021-9991(88)90177-5, http://www.sciencedirect.com/science/article/pii/0021999188901775.

[23] Tang H, Warnecke G. High resolution schemes for conservation laws and convection-diffusion equations with varying time and space grids. J Comput Math 2006;24(2):121–40. http://dx.doi.org/10.1002/fld.3646, http://www.jstor.org/stable/43694072.

[24] Seny B, Lambrechts J, Comblen R, Legat V, Remacle J-F. Multirate time stepping for accelerating explicit discontinuous Galerkin computations with application to geophysical flows. Internat J Numer Methods Fluids 2013;71(1):41–64. http://dx.doi.org/10.1002/fld.3646, http://arxiv.org/abs/https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.3646, https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.3646.

[25] Savcenco V, Hundsdorfer W, Verwer JG. A multirate time stepping strategy for stiff ordinary differential equations. BIT Numer Math 2007;47(1):137–55. http://dx.doi.org/10.1007/s10543-006-0095-7.

[26] Liu L, Li X, Hu F. Nonuniform time-step Runge–Kutta discontinuous Galerkin method for computational aeroacoustics. J Comput Phys 2010;229(19):6874–97. http://dx.doi.org/10.1016/j.jcp.2010.05.028, http://www.sciencedirect.com/science/article/pii/S0021999110002895.

[27] Liu L, Li X, Hu F. Nonuniform-time-step explicit Runge–Kutta scheme for high-order finite difference method. Comput & Fluids 2014;105:166–78. http://dx.doi.org/10.1016/j.compfluid.2014.09.008, http://www.sciencedirect.com/science/article/pii/S0045793014003454.

[28] Karypis G, Kumar V. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. 2009, http://www.cs.umn.edu/~metis.

[29] Rettenmaier D, Deising D, Ouedraogo Y, Gjonaj E, De Gersem H, Bothe D, et al. Load balanced 2D and 3D adaptive mesh refinement in OpenFOAM. SoftwareX 2019;10:100317. http://dx.doi.org/10.1016/j.softx.2019.100317, https://www.sciencedirect.com/science/article/pii/S2352711018301699.

[30] Zhiling Lan, Taylor VE, Bryan G. Dynamic load balancing for structured adaptive mesh refinement applications. In: International conference on parallel processing, 2001. 2001, p. 571–9. http://dx.doi.org/10.1109/ICPP.2001.952105.

[31] Moser RD, Kim J, Mansour NN. Direct numerical simulation of turbulent channel flow up to $Re\tau$=590. Phys Fluids 1999;11(4):943–5. http://dx.doi.org/10.1063/1.869966, http://arxiv.org/abs/10.1063/1.869966.

[32] Toro EF, Spruce M, Speares W. Restoration of the contact surface in the HLL-Riemann solver. Shock Waves 1994;4(1):25–34. http://dx.doi.org/10.1007/BF01414629.

[33] van Leer B. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. J Comput Phys 1979;32(1):101–36. http://dx.doi.org/10.1016/0021-9991(79)90145-1, http://www.sciencedirect.com/science/article/pii/0021999179901451.

[34] Kim J, Moin P, Moser R. Turbulence statistics in fully developed channel flow at low Reynolds number. J Fluid Mech 1987.

[35] Alfonsi G, Ciliberti SA, Mancini M, Primavera L. Direct numerical simulation of turbulent channel flow on high-performance GPU computing system. Computation 2016;4(1). http://dx.doi.org/10.3390/computation4010013, http://www.mdpi.com/2079-3197/4/1/13.