An efficient parallel implementation of explicit multirate Runge-Kutta schemes for discontinuous Galerkin computations

B. Seny[a,*], J. Lambrechts[a], T. Toulorge[a], V. Legat[a], J.-F. Remacle[a]

[a] *Institute of Mechanics, Materials and Civil Engineering, Université catholique de Louvain, Louvain-la-Neuve, Belgium*

Abstract

Although explicit time integration schemes require small computational efforts per time step, their efficiency is severely restricted by their stability limits. Indeed, the multi-scale nature of some physical processes combined with highly unstructured meshes can lead some elements to impose a severely small stable time step for a global problem. Multirate methods offer a way to increase the global efficiency by gathering grid cells in appropriate groups under local stability conditions. These methods are well suited to the discontinuous Galerkin framework. The parallelization of the multirate strategy is challenging because grid cells have different workloads. The computational cost is different for each sub-time step depending on the elements involved and a classical partitioning strategy is not adequate any more. In this paper, we propose a solution that makes use of multi-constraint mesh partitioning. It tends to minimize the inter-processor communications, while ensuring that the workload is almost equally shared by every computer core at every stage of the algorithm. Particular attention is given to the simplicity of the parallel multirate algorithm while minimizing computational and communication overheads. Our implementation makes use of the MeTiS library for mesh partitioning and the Message Passing Interface for inter-processor communication. Performance analyses for two and three dimensional practical applications confirm that multirate methods preserve important computational advantages of explicit methods up to a significant number of processors.

*Corresponding author

Email address: bruno.seny@uclouvain.be (B. Seny)

Keywords: high performance computing, parallelization, discontinuous Galerkin, multirate time stepping, explicit Runge-Kutta.

1. Introduction

Improving the efficiency of explicit time integration schemes, which are often severely restricted by the highest allowable stable time step, constitutes a key numerical challenge. Indeed, despite the numerous attractive properties of explicit schemes, the Courant-Friedrichs-Lewy (CFL) condition imposes that the stable time step must be kept under a certain critical value such that the physical information may be captured across the space discretization. For each cell, this value is proportional to the ratio between the grid size and the maximum wave/advective velocity. The smallest ratio determines the stable time step of the global problem. A structured grid with a constant velocity field yields a constant stable time step per cell. In the case of an unstructured grid with a large ratio between cell sizes and/or a highly varying velocity field, there will be a significant difference between the smallest and the largest stable time step. The computational cost of explicit methods may therefore be prohibitively high with respect to the problem size.

Several ways have been investigated to tame the CFL condition. Unconditionally stable implicit time integration schemes are a widespread alternative as they allow for large time steps. But large (non)linear systems of equations need to be solved. Warburton and Hagstrom proposed an algorithm which has a CFL number independent of the spatial order of approximation, for discontinuous Galerkin (DG) methods on structured meshes [1]. Lately, Lörcher et al. developed DG schemes based on a space-time expansion (STE-DG) for unsteady problems [2, 3]. Taylor expansions are used in space and time at the barycenter of the mesh elements and retain high order accuracy in time. Local time stepping methods of arbitrary high order and using arbitrary ratios of time-step sizes have also been introduced in the framework of the ADER schemes (arbitrary high order schemes using derivatives) [4, 5] and, amongst others, for DG discretizations (ADER-DG) [6, 7, 8]. We also mention explicit Runge-Kutta methods with nonuniform time steps (NUTS RK) [9] which have been developed for DG schemes in the field of Computational Aeroacoustics (CAA). To ensure correct communication between meshes with different time step sizes and preserve high-order accuracy, the intermediate solutions are coupled at the critical interfaces with minimum dispersion and dissipation errors.

The multirate approach focusses on reducing the overall computational effort by gathering mesh elements into appropriate groups that satisfy local stability conditions. The time step of a multirate group is an integer multiple of the smallest time step so that all steps are synchronized for every largest time step. The challenge, when developing multirate schemes, consists in providing a coherent transition between the multirate groups so that the information circulates properly. In particular, convergence and conservation properties need to be preserved. Several multirate approaches for conservation laws have been proposed in the literature since the early 1980s [10, 11, 12, 13, 14]. Hundsdorfer et al. [15] analyzed many of these schemes in terms of consistency and mass conservation. Constantinescu et al. [13] and Schlegel et al. [14] proposed strategies that accommodate the transition between bulk groups, where a classic explicit Runge-Kutta (ERK) base method is used, by means of buffer regions, where an adapted ERK method is used. The first method is conservative and preserves the strong stability properties of the base method but is at most second order accurate due to interface treatment between bulk and buffer groups. The second method reaches third order accuracy by using an appropriate base method and may be conservative if the partitioning is based on fluxes rather than on elements [14, 15].

This paper follows in the footsteps of the work accomplished in [16] where we investigated the efficiency of these last two multirate methods in the framework of DG finite elements. We have shown that they were especially well suited for computations on unstructured meshes. Depending on the distribution of the element sizes and the physical phenomena, significant speedups have been observed compared to the equivalent singlerate schemes for realistic geophysical flow problems. Here, we want to extend this multirate approach to the parallel framework. It is challenging since the computational load varies spatially and at the different sub-time steps of the ERK scheme. Load-balancing as well as communication have to be considered carefully. In this paper we focus on the parallelization of the class of multirate methods introduced by Constantinescu with time steps which are fractional powers of two of each other. Several issues are addressed from mesh partitioning strategies to practical implementations aspects. Schlegel et al. already investigated a multi-constraint balancing approach that employs three independent constraints correlated to the two highest temporal refinement levels and the remaining levels for which they obtained acceptable scalings up to a small number of processors, [17]. Here we develop a parallel multirate strategy that shares the workload almost equitably between all processors at every

multirate sub-time step for any number of temporal refinement levels and any number of processors. Illustrations and notations have been chosen to remain very close to the implementation. To the same end, the algorithms described in this paper are fully detailed.

The remainder of this paper is organized as follows: in Section 2 we summarize the DG and ERK methods; the multirate approach of Constantinescu is described in Section 3 and a generic implementation is proposed for the DG framework; Section 4, which constitutes the core of this work, addresses the mesh partitioning issues and proposes a generic parallel multirate algorithm; performance results are presented for three applications in Section 5.

2. The Runge-Kutta discontinuous Galerkin method

The Runge-Kutta discontinuous Galerkin (RKDG) methods were introduced by Cockburn and Shu in [18, 19, 20, 21]. They may be decomposed into two main steps: the DG spatial discretization of a conservation law described by a system of partial differential equations (PDE's) followed by the time integration of the resulting semi-discrete form using a class of ERK schemes.

2.1. The discontinuous Galerkin space discretization

For the sake of simplicity, let us assume a scalar hyperbolic conservation equation defined on a domain Ω:

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0 \quad , \quad u(\mathbf{x}, t^0) = u^0, \tag{1}$$

where u is the conserved unknown quantity while $\mathbf{f}(u)$ is the vector flux associated with u. For the sake of simplicity, the right-hand side is assumed to be zero. An initial condition u^0 is imposed for $t = t^0$ on the entire domain. Consider a spatial discretization of the domain, Ω^h, consisting of $\mathcal{N}$ non-overlapping elements Ω_e:

$$\Omega^h = \bigcup_{e=1}^{\mathcal{N}} \Omega_e. \tag{2}$$

By integrating by parts Eq. (1) multiplied by a sufficiently smooth test function $\hat{u}$ over each mesh element Ω_e we obtain the standard weak formulation:

$$\int_{\Omega_e} \frac{\partial u}{\partial t} \hat{u} d\Omega - \int_{\Omega_e} \mathbf{f}(u) \cdot \nabla \hat{u} d\Omega + \int_{\partial \Omega_e} \mathbf{f}(u) \cdot \mathbf{n} \hat{u} d\Gamma = 0 \qquad \forall e = 1, \cdots \mathcal{N}, \tag{3}$$

where $\mathbf{n}$ is the outward unit normal at the element boundaries. For the DG discretization [22], we seek an approximation $u^h \in \mathcal{U}_p^h$ of the true solution u and choose the test function $\hat{u}$ in the same space such that $\hat{u} = \hat{u}^h \in \mathcal{U}_p^h$. The finite dimensional space $\mathcal{U}_p^h$ denotes the space $P^p(\Omega_e)$ of polynomials of degree at most p on element Ω_e that are L_2 integrable. The approximated solution u^h can be defined for each mesh element Ω_e:

$$u_e^h = \sum_{i=1}^{\mathcal{N}_p} \phi_i U_{e,i}, \tag{4}$$

where ϕ_i are the shape functions and $U_{e,i}$ the nodal values on element e. The number of degrees of freedom (DoF) corresponding to the integration order p is $\mathcal{N}_p$. The DG weak formulation may be written for each mesh element:

$$\int_{\Omega_e} \frac{\partial u^h}{\partial t} \hat{u}^h d\Omega - \int_{\Omega_e} \mathbf{f}(u^h) \cdot \nabla \hat{u}^h d\Omega + \int_{\partial\Omega_e} \mathbf{f}(u^h) \cdot \mathbf{n} \hat{u}^h d\Gamma = 0, \tag{5}$$

where the interface flux term should be handled properly. Unlike for continuous finite element methods, the discrete solutions at element boundaries are multiply defined due to the discontinuity. The discrete normal flux $f_n = \mathbf{f}(u^h) \cdot \mathbf{n}$ depends on values at both sides of the discontinuity (U_e and U_{e+1}) and must be approximated by a numerical flux $\hat{f}_n$. To ensure the robustness and accuracy of the scheme, the numerical flux has to be defined properly. The calculation of these interface fluxes becomes a Riemann problem for which exact [23] or approximate [24, 25] Riemann solvers are used.

Among the numerous vices and virtues of the DG method, we will only emphasize here that a high number of DoF are needed, but they are not shared between elements. Many operations are thus local which allow, among other things, a straightforward parallelization and h- and p-refinement.

2.2. The singlerate explicit Runge-Kutta time discretization

The aim of this section is to introduce some mathematical and algorithmic notations for the ERK methods that will be used all along the paper. Consider that a general multi-dimensional conservation law, described by a system of PDE's, has been discretized spatially by means of the DG method. The whole set of nodal unknowns $\mathbf{U}$ can be viewed as the concatenation of the set of unknowns $\mathbf{U}_e$ associated to each element Ω_e:

$$\mathbf{U} = \{\mathbf{U}_1, \mathbf{U}_2, \cdots, \mathbf{U}_{\mathcal{N}}\}. \tag{6}$$

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$		$\ddots$		
c_s	a_{s1}	a_{s2}	$\cdots$	$a_{s,s-1}$	
	b_1	b_2	$\cdots$	b_{s-1}	b_s

Table 1: Butcher tableau for an s-stage explicit Runge-Kutta method.

Following the method of lines we can write the next semi-discrete form for every mesh element:

$$\mathbf{M}_e \frac{d\mathbf{U}_e}{dt} = \mathbf{F}(\mathbf{U}_e, \mathbf{U}_n, t) = \mathbf{F}^V(\mathbf{U}_e, t) + \mathbf{F}^I(\mathbf{U}_e, \mathbf{U}_n, t), \tag{7}$$

where $\mathbf{M}_e$ is the element-wise mass matrix, $\mathbf{U}_n$ are the unknowns associated with the neighboring elements of e, and $\mathbf{F}$ is the steady-state residual (SSR) resulting from the DG formulation. $\mathbf{F}^V$ (respectively $\mathbf{F}^I$) stands for the volume (respectively interface) part of the SSR. Within this framework an s-stage ERK method computes the next step solution for an element e, $\mathbf{U}_e^{N+1}$, at time t^{N+1} by using the actual solution $\mathbf{U}_e^N$ available at time t^N. Butcher tableaus [26] are a convenient way to represent the coefficients a_{ij}, b_i and c_i of an ERK scheme, see Table 1. The explicitness of the method requires that the matrix a_{ij} is strictly lower triangular.

A possible generic implementation of these methods is given by Algorithm 1, in which we distinguish three main operations: compute the current solution ($\mathcal{C}$); compute the estimate of the time derivatives ($\mathcal{K}$); compute the next step solution ($\mathcal{U}$). Operations $\mathcal{C}$ and $\mathcal{K}$ are performed at each stage i of the the method and for every mesh element e. The element-wise current solutions $\mathbf{V}_e^{(i)}$ are obtained by summing the actual solution $\mathbf{U}_e$ with a linear combination of the time derivative estimates $\mathbf{K}_e^i$. The latter is obtained by the multiplication of the inverse of the element-wise mass matrix $\mathbf{M}_e$ with the flux function $\mathbf{F}_e$ which depends on the current solution on the actual element, $\mathbf{V}_e^{(i)}$, and its neighboring elements, $\mathbf{V}_n^{(i)}$. The update operation $\mathcal{U}$ is realized at the end of the s stages by summing the actual solution with a linear combination of the time derivatives.

Algorithm 1 Implementation of explicit Runge-Kutta methods

Require: $\mathbf{U}^N$, $\mathcal{N}$, s, Δt
  $\mathbf{U} \leftarrow \mathbf{U}^N$

  for $i = 1$ to s **do**

    // Compute the current input ($\mathcal{C}$)
    for $e = 1$ to $\mathcal{N}$ **do**
      $\mathbf{V}_e^{(i)} \leftarrow \mathbf{U}_e + \Delta t \sum_{j=1}^{i-1} a_{ij} \mathbf{K}_e^j$
    end for

    // Compute the SSR ($\mathcal{K}$)
    for $e = 1$ to $\mathcal{N}$ **do**
      $\mathbf{F}_e \leftarrow \mathbf{F}^V(\mathbf{V}_e^{(i)}, t^N + c_i \Delta t) + \mathbf{F}^I(\mathbf{V}_e^{(i)}, \mathbf{V}_n^{(i)}, t^N + c_i \Delta t)$
      $\mathbf{K}_e^i \leftarrow \mathbf{M}_e^{-1} \mathbf{F}_e$
    end for

  end for

  // Update the solution ($\mathcal{U}$)
  for $e = 1$ to $\mathcal{N}$ **do**
    $\mathbf{U}_e \leftarrow \mathbf{U}_e + \Delta t \sum_{j=1}^{s} b_j \mathbf{K}_e^j$
  end for

  $\mathbf{U}^{N+1} \leftarrow \mathbf{U}$

3. Second order explicit multirate Runge-Kutta schemes

Constantinescu and Sandu [13] constructed a family of second order multirate partitioned Runge-Kutta (MPRK) schemes that use different time steps that are integer multiples of the smallest time step. Here, we will only consider the case where the time steps are fractional powers of two of each other, which is a reasonable compromise between simplicity and efficiency. The key idea consists in organizing mesh elements in multirate groups. Buffer groups accommodate the transition between two successive bulk groups which use a base ERK method with time steps $\Delta t/2^n$ and $\Delta t/2^{n+1}$. An adapted method is used for the buffer elements obtained by repeating twice the base ERK method with the largest time step $\Delta t/2^n$. The adapted method performs the same number of stages as its neighboring bulk group with smallest time step and communication is thus possible. The Butcher tableaus of the RK2a method and the adapted version are represented in Table 2a and 2b. Two bulk groups must be separated by at least s connected buffer elements. Indeed, at this distance the adapted method reduces to the base method. This multirate approach is conservative for conservative laws since the b vectors of two adjacent Butcher tableaus are identical. The convergence of the method is limited to second order due to the treatment of the critical interface between a buffer and a bulk group. The pure temporal error of a multirate scheme will be of the same magnitude as the error associated with the largest time step used after a significant number of time steps. The error propagates through elements at the same rate as the information. A detailed mathematical development of the method and its characteristics are available in [16] and [13].

For large scale applications on unstructured meshes it is essential to manage multiple levels of refinement. This method can be extended to any number of temporal refinement levels and multirate groups therefore need to be defined properly. A generic way to construct the multirate groups is a crucial requirement. The approach adopted, defined in [16], is briefly summarized in the next section.

3.1. Multirate Groups

A first step when building multirate groups is to determine a reference time step, Δt_*, that will be the largest time step used. It needs to be between Δt_m and Δt_M, that are the maximum stable time step on the most restrictive and the least restrictive element respectively. For simplicity, we

0		
1	1	
	1/2	1/2

(a)

0				
1	1			
0	0	0		
1	0	0	1	
	1/4	1/4	1/4	1/4

(b)

Table 2: Butcher tableaus for the base (a) and adapted (b) method, based on RK2a, with coefficients $\{a_{i,j}^{(0)}, b_j^{(0)}, c_i^{(0)}\}$ and $\{a_{i,j}^{(1)}, b_j^{(1)}, c_i^{(1)}\}$, respectively.

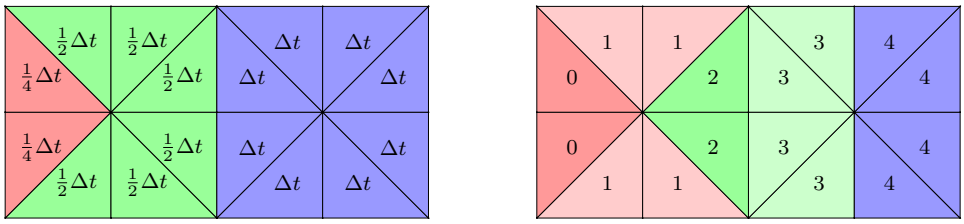
assume multirate setups where Δt_* is a power of two of Δt_m. In this context, we define the maximum multirate exponent:

$$z^* \quad = \quad \log_2 \frac{\Delta t_*}{\Delta t_m}. \tag{8}$$

Accordingly, the number of temporal refinements is $z^* + 1$. It is now possible to allocate each mesh element according to its own stable time step to a subset of time steps:

$$[\Delta t_m,\ 2^{-(z^*-1)}\Delta t_*[\ \cup\ [2^{-(z^*-1)}\Delta t_*,\ 2^{-(z^*-2)}\Delta t_*[\ \cup\ \cdots\ \cup\ [\Delta t_*,\ \Delta t_M]. \tag{9}$$

As an illustrative example, let us consider a simple triangular mesh with element-wise stable time steps. In Fig. 1a, we define the local stable time steps such that $\Delta t_m = \frac{1}{4}\Delta t$, $\Delta t_M = \Delta t$ and $z^* = 2$ for $\Delta t_* = \Delta t$.



(a) Local stable time steps (b) Multirate groups

Figure 1: Illustrative example: the multirate method based on RK2a.

Next, the buffer groups need to be introduced. To distinguish between the multirate groups, we use a similar notation as introduced in [16]:

$$\theta = 2(z^* - z) + \sigma, \tag{10}$$

where $\sigma = 0$ for bulk groups and $\sigma = 1$ for buffer groups. The multirate exponent, z, determines the effective time step of a group, i.e. $\Delta t_*/2^z$. The parameters z and σ may also be obtained directly from the multirate tag θ:

$$\sigma = (\theta \mod 2) \quad , \qquad z = z^* - \lceil \theta/2 \rceil. \tag{11}$$

The mesh elements are thus distributed among $z^* + 1$ bulk groups and z^* buffer groups. Two neighboring mesh elements have either the same multirate tag either two successive multirate tags. The mesh is decomposed as follows:

$$\bigcup_{\theta=0}^{2z^*} \Omega_\theta^h = \Omega^h, \tag{12}$$

where Ω_θ^h is the set of elements that belong to multirate group θ. The multirate load (estimated as the number of evaluations of the SSR for a Δt_* multirate time step) of an element belonging to a multirate group θ is:

$$\lambda_\theta \quad = \quad 2^{\lceil (2z^* - \theta)/2 \rceil} = 2^{z+\sigma}. \tag{13}$$

Fig. 1b represents the multirate groups for the multirate method based on RK2a . Three bulk groups, $\Omega_{0,2,4}^h$, are separated by two buffer groups, $\Omega_{1,3}^h$, of size two. For this example half of the elements are in buffer groups and the theoretical speedup would be 1.6.

The *theoretical speedup*, compared to the equivalent singlerate method, is evaluated as follows:

$$S_{\text{th}}(z^*) \quad = \quad \frac{2^{z^*} |\Omega^h|}{\left(\sum_{\theta=0}^{2z^*} |\Omega_\theta^h| \lambda_\theta \right)}, \tag{14}$$

where $|\Omega_\theta^h|$ stands for the number of elements present in a multirate group θ. In practice, the *effective speedup* is evaluated as the following ratio

$$S_{\text{ef}}(z^*) \quad = \quad \frac{WT(0)}{WT(z^*)}, \tag{15}$$

where $WT(z^*)$ and $WT(0)$ are the wall clock times taken by the multirate and the singlerate methods, respectively, to achieve a fixed physical time.

3.2. Serial implementation of the explicit multirate method

Once the multirate groups are built, they need to be integrated with their own time step and Butcher tableau (buffer or bulk). Moreover, they need to communicate properly with each other. A generic implementation of the multirate method of Constantinescu for any ERK base method and any number of multirate groups must be proposed. Following the element-wise decomposition given by Eq. (6), we define the subset $\mathbf{U}_\theta$ of the discontinuous unknowns corresponding to the variables associated to a multirate group θ:

$$\mathbf{U}_\theta \;=\; \{\mathbf{U}_{t(1)}, \cdots, \mathbf{U}_{t(m)}\}, \tag{16}$$

where m is the number of elements belonging to Ω_θ^h and $t(i)$ represents the mapping of element i of multirate group θ to the general element numbering.

If a multirate group is empty then $\mathbf{U}_\theta = \{\}$ and any operation on this vector would be trivial. We define for each tag θ: $\mathbf{U}_\theta$ (the actual solution), $\mathbf{V}_\theta$ (the current solution) and $\mathbf{K}_\theta^i$ (the SSR $\mathbf{F}_\theta$, multiplied by the inverse of the mass matrix $\mathbf{M}_\theta$, for every stage i of an ERK method). The mass matrix is block-diagonal where each block is the mass matrix associated with an element of the multirate group θ:

$$\mathbf{M}_\theta \;=\; \mathrm{diag}(\mathbf{M}_{t(1)}, ..., \mathbf{M}_{t(m)}). \tag{17}$$

Algorithm 2 gives the pseudo-code to compute $\mathbf{U}^{N+1}$ from $\mathbf{U}^N$. For the sake of simplicity, the algorithm is given for an autonomous system. We assume a multirate setup characterized by a reference time step, Δt_*, and a maximum multirate exponent, z^*. Henceforth, the total number of sub-time steps performed by the multirate group with the smallest time step to reach Δt_* is $s^* = 2^{z^*} s$. Therefore, the multirate algorithm will have s^* stages.

The three main operations $\mathcal{C}$, $\mathcal{K}$ and $\mathcal{U}$ are performed in the multirate algorithm but don't involve all multirate groups at each intermediate stage. To manage these variable sets of multirate groups, two vectors are introduced. The vector Θ, of size s^*, gives the maximum multirate tag up to which operation ($\mathcal{K}$) has to be executed at each stage of the algorithm. This vector is determined by the multirate setup and is fixed for the whole algorithm. When evaluating the interface part of the SSR, operation ($\mathcal{K}$) requires the current solution of the actual multirate group and its two neighboring multirate groups. So, at stage i, the operation of type ($\mathcal{C}$) must be executed up to multirate tag $\Theta[i] + 1$. The vector Π, of size $2z^* + 1$, gives the current stage of the internal ERK method (bulk or buffer) for each multirate group and is

Algorithm 2 Serial implementation of the explicit multirate method

Require: $\mathbf{U}^N$, s, Δt_*, z^*
$s^* \leftarrow 2^{z^*}s$, compute Θ, $\Pi \leftarrow \mathbf{0}$, $\mathbf{U} \leftarrow \mathbf{U}^N$

for $i = 1$ to s^* **do**

// Compute the current input ($\mathcal{C}$)
for $\theta = 0$ to $\Theta[i] + 1$ **do**
$h \leftarrow \Delta t_*/2^z$, $\Pi[\theta] \leftarrow \big(\Pi[\theta] \mod (\sigma+1)s\big) + 1$
$\mathbf{V}_\theta \leftarrow \mathbf{U}_\theta + h \sum_{j=1}^{\Pi[\theta]} a^{(\sigma)}_{\Pi[\theta],j} \mathbf{K}^j_\theta$
end for

// Compute the SSR ($\mathcal{K}$)
for $\theta = 0$ to $\Theta[i]$ **do**
$\mathbf{F}_\theta \leftarrow \mathbf{F}^V(\mathbf{V}_\theta) + \mathbf{F}^I(\mathbf{V}_\theta, \mathbf{V}_\theta) + \mathbf{F}^I(\mathbf{V}_\theta, \mathbf{V}_{\theta+1}) + \mathbf{F}^I(\mathbf{V}_\theta, \mathbf{V}_{\theta-1})$
$\mathbf{K}^{\Pi[\theta]}_\theta \leftarrow \mathbf{M}^{-1}_\theta \mathbf{F}_\theta$
end for

// Update the solution when needed ($\mathcal{U}$)
for $\theta = 0$ to $\Theta[i]$ **do**
$h \leftarrow \Delta t_*/2^z$
if $\Pi[\theta] = (\sigma+1)s$ **then**
$\mathbf{U}_\theta \leftarrow \mathbf{U}_\theta + h \sum_{j=1}^{\Pi[\theta]} b^{(\sigma)}_j \mathbf{K}^j_\theta$
end if
end for

end for
$\mathbf{U}^{N+1} \leftarrow \mathbf{U}$

updated at every stage of the algorithm. It allows us to determine which coefficients of the Butcher tableau are needed for operation ($\mathcal{C}$) and whether the solution should be updated ($\mathcal{U}$). Every multirate group is integrated with its proper time step h. The parameter σ allows us to distinguish between bulk and buffer groups, see Eq. (10), as well as their associated Butcher tableaus, $\{a_{i,j}^{(\sigma)}, b_j^{(\sigma)}, c_i^{(\sigma)}\}$. Table 3 shows some aspects of the multirate algorithm based on the RK2a method for the illustrative example.

For each multirate group, the SSR $\mathbf{F}_\theta$ is evaluated as the sum of four terms depending on the current solution, $\mathbf{V}$: $\mathbf{F}^V(\mathbf{V}_\theta)$ (the volume term of group θ), $\mathbf{F}^I(\mathbf{V}_\theta, \mathbf{V}_\theta)$ (the interface term for interior faces of group θ), $\mathbf{F}^I(\mathbf{V}_\theta, \mathbf{V}_{\theta+1})$ (the interface term for faces between group θ and $\theta+1$) and $\mathbf{F}^I(\mathbf{V}_\theta, \mathbf{V}_{\theta-1})$ (the interface term for faces between group θ and $\theta-1$). The interface terms between two multirate groups are computed when the group with the smallest multirate tag is treated.

i	$\theta=0$				$\theta=1$				$\theta=2$				$\theta=3$				$\theta=4$			
	$\Pi[0]$	$\mathcal{C}$	$\mathcal{K}$	$\mathcal{U}$	$\Pi[1]$	$\mathcal{C}$	$\mathcal{K}$	$\mathcal{U}$	$\Pi[2]$	$\mathcal{C}$	$\mathcal{K}$	$\mathcal{U}$	$\Pi[3]$	$\mathcal{C}$	$\mathcal{K}$	$\mathcal{U}$	$\Pi[4]$	$\mathcal{C}$	$\mathcal{K}$	$\mathcal{U}$
1	1				1				1				1				1			
2	2				2				2											
3	1				3				1											
4	2				4				2				2				2			
5	1				1				1				3				1			
6	2				2				2											
7	1				3				1											
8	2				4				2				4				2			

Table 3: Illustration of the multirate algorithm (RK2a) for the illustrative example for which $s^* = 8$ and $\Theta = [4\ 1\ 1\ 3\ 3\ 1\ 1\ 4]$. The current value of the vector $\Pi[\theta]$ and the operations ($\mathcal{C}$, $\mathcal{K}$, $\mathcal{U}$) that need to be performed are specified at each stage i of the algorithm and for every multirate group θ.

In this multirate algorithm, the memory storage requirements are higher compared to the equivalent singlerate algorithm. The buffer elements require the storage of twice as many time derivatives, $\mathbf{K}$, compared to the bulk elements. The fraction of buffer elements in a multirate setup will thus determine the memory overhead. However, the ratio between buffer and bulk elements is generally not too significant.

4. Parallelization of the explicit multirate Runge-Kutta method

The parallelization of an algorithm consists in dividing the work over multiple processors such that the wall clock time is ideally inversely proportional to the number of processors. In practice however, the unavoidable sequential fraction of a program, according to Amdahl's law [27], the idle times due to an imperfect sharing of the workload between processors and the communication to computation ratio can affect the parallel performance. Moreover, the computer architecture and in particular the latency and bandwidth of the inter-processor communications and memory fetches may influence the global efficiency. Explicit singlerate time integration for space-DG formulations has the advantage of robust scalability properties on parallel computer platforms [28]. The reason is twofold: the inherent properties of an explicit time integration and the spatial locality of the DG discretization.

For explicit schemes, the succession of several simple instructions is easily handled on parallel computers as is the dependence of the solution on the sole previous time steps. The parallelization of implicit schemes is challenging due to the highly coupled nature of the problem. This is why scalable and robust implicit solvers are hard to develop. For transient computations, explicit solvers are often preferred, especially in ocean modeling because those are generally able to provide scalable and efficient parallel solutions. For DG formulations, the weak coupling at inter-element boundaries, realized through a numerical flux formulation, ensures a high parallel efficiency. Although additional DoF are involved due to the discontinuities at element boundaries, their local nature compensates somewhat for this overhead.

A classic strategy, which combines a message passing programming model and mesh partitioning, is generally used to parallelize these explicit schemes. An optimal domain decomposition is obtained if the same number of elements are allocated to each processor whilst the number of faces at the inter-processor boundaries are minimized. Indeed, the computational cost may be assumed constant per element and per time step. The extension of these parallel assets to explicit multirate methods is far from being straightforward because all elements do not require the same number of iterations. Elements belonging to a multirate group θ require λ_θ substantial operations (evaluations of the SSR). Moreover, the number of potential messages exchanged between two elements (if their common interface is on an inter-processor boundary) depends on their respective multirate groups. The computational load and the communication volume are variable spatially and for each sub-

time step of the multirate algorithm. The synchronization of all the multirate groups plays an important role to minimize the idle times. A classical balancing is therefore inefficient because it only allocates the same number of elements per processor.

4.1. Mesh partitioning for explicit multirate schemes

The selection of a partitioning is critical to achieve a good efficiency. To this end, let us analyze different domain decompositions for the illustrative example of Fig. 1a and 1b with the RK2a multirate method and for two partitions. In Fig. 2a we show the computational cost of each element which can be approximated by the number of SSR evaluations required to reach Δt_*. The total workload of the problem is 80 SSR evaluations.

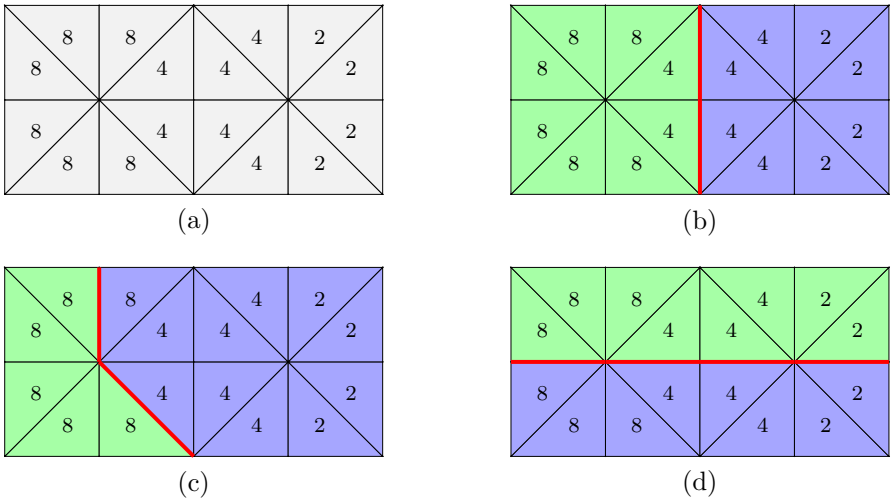


Figure 2: Illustration of the three different partitioning strategies for the illustrative example. Multirate loads, λ_θ, are associated with each mesh element (a). The two resulting optimal partitions P_1 (green) and P_2 (blue) are shown for the classical (b), single-constraint (c) and multi-constraint (d) partitioning. The inter-partition faces are highlighted in red.

In *classical partitionings*, the aim is to ensure that the the total amount of elements is equally distributed on each processor. Among all the possible domain decompositions, the one that yields the fewest inter-processor faces is selected, see Fig. 2b. The multirate workload of P_1 (resp. P_2) is 56 (resp. 24). The processor designated for partition P_2 will wait 32 work units

while partition P_1 will work continuously, see Table 4a. There are two inter-processor faces and every processor has to send and receive four messages of size two, see Table 4b.

	P_1		P_2	
stage	wait	work	work	wait
1	0	8	8	0
2	0	6	0	6
3	0	6	0	6
4	0	8	4	4
5	0	8	4	4
6	0	6	0	6
7	0	6	0	6
8	0	8	8	0
total	0	56	24	32

(a)

	P_1		P_2	
stage	receive	send	send	receive
1	2	2	2	2
2	0	0	0	0
3	0	0	0	0
4	2	2	2	2
5	2	2	2	2
6	0	0	0	0
7	0	0	0	0
8	2	2	2	2
total	8	8	8	8

(b)

Table 4: Classical partitioning: estimation for each processor at each stage of the global algorithm of the (a) work and wait times and (b) sizes of sent and received messages.

In *single-constraint partitionings*, the aim is to ensure that each processor supports the same workload, a single weight is assigned to each mesh element based on its own cost. Afterwards, it is required that the sum of the element weights is equal on each processor. The weight of an inter-element face is evaluated as the sum of the weights of its two neighboring elements. Among all the possible solutions, the one that minimizes the total amount of communications is picked, see Fig. 2c. Partitions P_1 and P_2 have both a total weight of 40 and their inter-processor boundary has a weight of 28. Yet, this partitioning is far from being optimal for multirate because both processors have idle times, see Table 5a. Both processors finish their task after 56 work units (40 working, 16 waiting). The communication consists of 8 different messages that have either a size 1 or 2 for both processors, see Table 5b.

In *multi-constraint partitionings*, the aim is to minimize the waiting times. The key idea is to have a perfect load balance at each sub-time step. In other words, each processor should treat the same number of elements at each stage of the multirate algorithm. Again, the solution that minimizes the communication volume is selected, see Fig. 2d. Both partitions have 3 elements of load 8, 3 of load 4 and 2 of load 2. With this kind of strategy and assuming zero communication costs, none of the two processors need to wait at any stage of the algorithm, see Table 6a. By contrast, communication costs are relatively high: both processors send and receive 8 messages of size

	P_1		P_2	
stage	wait	work	work	wait
1	6	5	11	0
2	0	5	1	4
3	0	5	1	4
4	2	5	7	0
5	2	5	7	0
6	0	5	1	4
7	0	5	1	4
8	6	5	11	0
total	16	40	40	16

(a)

	P_1		P_2	
stage	receive	send	send	receive
1	2	2	2	2
2	2	1	2	1
3	2	1	2	1
4	2	2	2	2
5	2	2	2	2
6	2	1	2	1
7	2	1	2	1
8	2	2	2	2
total	16	12	16	12

(b)

Table 5: Single-constraint partitioning: estimation for each processor at each stage of the global algorithm of the (a) work and wait times and (b) sizes of sent and received messages.

1 or 2. The total inter-processor communication volume is 36.

	P_1		P_2	
stage	wait	work	work	wait
1	0	8	8	0
2	0	3	3	0
3	0	3	3	0
4	0	6	6	0
5	0	6	6	0
6	0	3	3	0
7	0	3	3	0
8	0	8	8	0
total	0	40	40	0

(a)

	P_1		P_2	
stage	receive	send	send	receive
1	4	4	4	4
2	1	1	1	1
3	1	1	1	1
4	3	3	3	3
5	3	3	3	3
6	1	1	1	1
7	1	1	1	1
8	4	4	4	4
total	18	18	18	18

(b)

Table 6: Multi-constraint partitioning: estimation for each processor at each stage of the global algorithm of the (a) work and wait times and (b) sizes of sent and received messages.

A dynamic load-balancing strategy could be more adequate since the workload changes temporally. But this would mean that the partitioning must be reevaluated at each sub-time step of the algorithm. Such a strategy would bring serious implementation complications and could be very costly. The key idea of this work is to compute a mesh partitioning that remains as good as possible at all stages of the algorithm, assuming that the stability conditions are never violated in time. Dynamic multirate strategies could be an extension of this work, for problems with varying CFL conditions in both space and time. From a computational point of view, the multi-constraint partitioning turns out to be the most adequate strategy as it minimizes the

idle times and provides the best static load-balancing. For practical applications with many multirate groups and partitions it is most often impossible to have a perfect multi-constraint load balancing. This type of partitioning is tantamount to consider as many partitioning subproblems as there are temporal refinement levels. With such constraints it is also almost impossible to build contiguous partitions. If the equal sharing of computational cost in both time and space is essential to achieve an acceptable performance, the number and size of communications may quickly become cumbersome. As illustrated by previous examples, the communication volume increases with the quantity and severity of constraints. It will also grow substantially with the number of multirate groups as well as the number of required partitions. In practice, various factors will determine the influence of communication on global performance. In particular, the latency time and bandwidth speed associated with a parallel architecture are crucial. Special care should be taken to minimize the number of messages and their size in the implementation of the parallel multirate algorithm.

4.2. The multi-constraint partitioning strategy

When partitioning finite element meshes for scientific computations, it is a common practice to partition the associated graph. A graph vertex is assigned to each mesh element and an edge links two vertices if two mesh elements share a common interface. Classical mesh partitioning focusses on resolving an optimization problem with the objective of minimizing the edge-cut under the constraint that vertices are equally distributed over all partitions.

The software package MeTiS[1] aims to partition large irregular graphs and large meshes as well as to compute fill-reducing orderings of sparse matrices. Partitioning is done by means of a multilevel graph partitioning algorithm: a smaller graph, obtained by collapsing vertices and edges, is partitioned, and is then uncoarsened to construct a partitioning of the original graph [29, 30, 31]. In the latest version of MeTiS, routines are included to partition a graph with multiple balancing constraints. The multi-constraint partitioning algorithms and their applications are described in detail in [32].

The functionalities of MeTiS meet our requirement for the parallel multirate partitioning problem. Consider a graph $G = (V, E)$ where V is the set

[1] http://glaros.dtc.umn.edu/gkhome/views/metis

of vertices v_i of size $|V|$ and E the set of edges e_j of size $|E|$. A vector of binary weights α_i of size $m = z^* + 1$, corresponding to the number of multirate temporal refinements, is associated to each vertex v_i. Each constraint k represents a set of multirate groups for which the SSR needs to computed together at the same stages of the algorithm. The binary value α_i^k depends on the belonging of a mesh element i (graph vertex v_i) to one of these sets of multirate groups. This vector may be expressed as follows:

$$\alpha_i^k = \begin{cases} 1 & \text{if element } i \in \bigcup_{\theta=0}^{2k-1} \Omega_\theta^h \\ 0 & \text{otherwise} \end{cases} . \tag{18}$$

The mesh elements that require an evaluation of the SSR at each stage of the algorithm will be active for each constraint k. To minimize the total communication volume, a single weight β_j is assigned to each inter-element face (graph edge e_j) and is defined as the sum of the multirate loads of the two elements that it separates. The associated weighted graph provided to the MeTiS partitioner is shown in Fig. 3 for the illustrative example. A vector of three constraints is associated to each graph vertex as well as a single weight to each edge.

For an arbitrary number of partitions p, we define the load imbalance l_k associated to constraint k:

$$l_k = \frac{p \ \max_{q,v_i} \left(\sum_{\mathcal{P}[i]=q} \alpha_i^k \right)}{\sum_{\theta=0}^{2k-1} |\Omega_\theta^h|} \qquad k = 1, \ldots, m. \tag{19}$$

The theoretical global load imbalance is defined by:

$$L = \sum_{k=1}^{m} l_k r_k \quad \text{with} \quad r_k = \frac{\max(1, 2^{z^*-k}) \sum_{\theta=0}^{2k-1} |\Omega_\theta^h|}{\sum_{\theta=0}^{2z^*} |\Omega_\theta^h| \lambda_\theta}, \tag{20}$$

where r_k is the relative computational weight associated to constraint k. The sum of all these computational weights r_k is equal to unity. A perfect load-balancing is performed if $L = 1$. Without any communication overhead, it means that each processor needs exactly the same time to perform the computations. In practice, however, perfect load balancing is rarely achievable. Some or all constraints can be relaxed by associating a tolerance c_k to the constraint k.

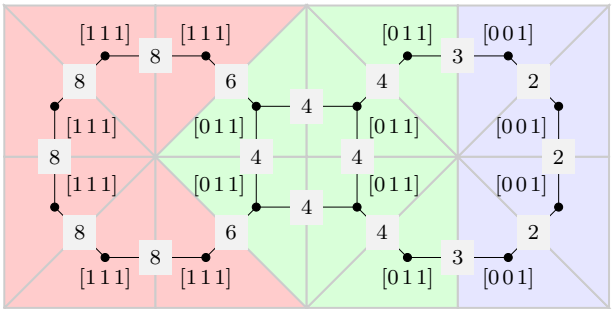


Figure 3: Weighted graph for MeTiS partitioner. A vector of weights α_i is associated to each vertex i and every edge j has a single weight β_j. The relative constraint weights are $r_1 = r_2 = 0.3$ and $r_3 = 0.4$

It is now possible to formulate the constrained minimization problem that will be solved by MeTiS:

Find the p-way partitioning $\mathcal{P}$ of G
that minimizes C the total weighted edge-cut defined by:

$$C = \sum_{e_j \in \mathcal{E}} \beta_j, \tag{21}$$

under the constraints:

$$l_k \leq c_k \qquad k = 1, \ldots, m, \tag{22}$$

$\mathcal{E}$ being the set of graph edges that are cut by a partitioning $\mathcal{P}$.

4.3. Estimation of the mesh partition quality

Defining relevant indicators of the quality of a mesh partitioning is challenging as it depends not only on the computational load balancing but also on the ratio between communication and computation time. The volume and number of communications have to be taken into account, but hardware characteristics such as latency and bandwidth speed also come into play. Moreover, in the multirate context, this ratio will vary both spatially and temporally as the communication volumes are not identical at each stage and for each partition. Despite this, we will try to define some simple indicators.

In order to take into account the number of multirate groups we define a reference multirate setup which corresponds to the maximum possible temporal refinement such that:

$$z^+ = \left\lceil \log_2 \frac{\Delta t_M}{\Delta t_m} \right\rceil \quad , \quad \Delta t_+ = 2^{z^+} \Delta t_m. \tag{23}$$

We estimate the *average communication volume* sent or received per partition during a period Δt_+ for a multirate setup characterized by a maximum multirate exponent z^* as:

$$\varepsilon(p, z^*) \quad = \quad \frac{(2^{z^+ - z^*})}{p} C, \tag{24}$$

where C is the weighted edge-cut, defined by Eq. (21), which depends on the partitioning strategy and z^*. We also introduce an indicator of the *average communication to computation ratio* per partition and per unit time:

$$\zeta(p, z^*) \quad = \quad \frac{C}{p \sum_{\theta=0}^{2z^*} |\Omega_\theta^h| \lambda_\theta}. \tag{25}$$

Both indicators are very rough, as their minimum and maximum values could depend significantly on the partition and on the stage of the algorithm.

4.4. Parallel multirate algorithm

Once the mesh is decomposed, each processor is responsible for one partition. At the inter-processor boundaries, data has to be exchanged to compute the fluxes through the interfaces between elements. Therefore, a distinction must be made between local and remote connections. The former allow communication between two elements processed by the same CPU, while the latter allow communication between two elements processed by different CPU cores. Whilst the local connections are treated as in the sequential case, the remote connections must be handled properly. Ghost elements are introduced at each inter-processor boundary and their purpose is twofold: to act as a receive buffer for the incoming data and to allow the computation of inter-processor interface fluxes. For DG discretizations, there are at most twice as many ghost elements as there are inter-processor faces. The data for the ghost elements must be scattered by the adjacent partitions where the corresponding elements reside. It is important to minimize the number of messages as well as their volume.

The introduction of ghost elements results in the addition of new DoF. For this, we assume that $\tilde{\mathbf{X}}$ represents a vector with the ghost element variables associated to the non-ghost vector $\mathbf{X}$. In the same way, $\tilde{\mathbf{X}}_\theta$ gives the ghost DoF corresponding to multirate group θ. By considering the set of interfaces I in a mesh, a distinction is made between inter-processor interfaces, I^*, and intra-processor interfaces, $I \backslash I^*$. To handle inter-pocessor communications, two functions are defined, that take as argument a list of multirate tags:

`START_COMM`$(\theta_0, \cdots, \theta_n)$
`END_COMM`$(\theta_0, \cdots, \theta_n)$

The first one initializes communication between the boundary elements of a partition and the corresponding ghost elements of the adjacent partitions from tag θ_0 to θ_n. The second one waits for those communications to finish. By doing so, it is possible that at each stage of the algorithm at most one message is sent and received by each processor. In practice, the non-blocking communications will be handled with MPI[2] routines.

Algorithm 3 gives a parallel implementation for the multirate methods of Constantinescu, which is quite similar to the sequential algorithm but with communications between the different processors now included. An illustration of the parallel multirate algorithm for the illustrative example is given in Fig. 4. At each stage of the algorithm, the current solutions, $\mathbf{V}_\theta$, are evaluated for all relevant multirate groups ($\mathcal{C}$). Next, the current solutions at inter-processor boundaries are transmitted to the neighboring partitions ($\mathcal{S}$). While the data is sent, the volume and intra-processor interface residuals are computed ($\mathcal{K}[V, I \backslash I^*]$). The processors then receive the current solutions of the relevant partitions and multirate groups ($\mathcal{R}$) to compute the interface SSR at inter-processor boundaries ($\mathcal{K}[I^*]$). Finally, the solution is updated when required ($\mathcal{U}$). For this implementation, some extra computations have to be performed since the SSR is computed twice at all inter-processor faces.

In such an implementation we perform all the computations that do not require information from the ghost elements while data is in transit. If the communication to computation ratio is not significant, the communication should be hidden by computation. However, the multi-constraint partitioning objective is to minimize the total weighted edge-cut, and it is not possible,

[2] http://www.mpi-forum.org/

Algorithm 3 Parallel implementation of the explicit multirate method

Require: $\mathbf{U}^N$, s, Δt_*, z^*

$s^* \leftarrow 2^{z^*}s$, compute Θ, compute $\mathcal{P}$, $\Pi \leftarrow \mathbf{0}$, $\mathbf{U} \leftarrow \mathbf{U}^N$

for $i = 1$ to s^* **do in parallel**

// Compute the current input ($\mathcal{C}$)
for $\theta = 0$ to $\Theta[i] + 1$ **do**
$h \leftarrow \Delta t_*/2^z$, $\Pi[\theta] \leftarrow \big(\Pi[\theta] \mod (\sigma+1)s\big) + 1$
$\mathbf{V}_\theta \leftarrow \mathbf{U}_\theta + h\sum_{j=1}^{\Pi[\theta]} a^{(\sigma)}_{\Pi[\theta],j}\mathbf{K}^j_\theta$
end for

`START_COMM`$(0, \cdots, \Theta[i]+1)$

// Compute the volume and intra-processor interface SSR ($\mathcal{K}[V, I\backslash I^*]$)
for $\theta = 0$ to $\Theta[i]$ **do**
$\mathbf{F}_\theta \leftarrow \mathbf{F}^V(\mathbf{V}_\theta) + \mathbf{F}^{I\backslash I^*}(\mathbf{V}_\theta, \mathbf{V}_\theta) + \mathbf{F}^{I\backslash I^*}(\mathbf{V}_\theta, \mathbf{V}_{\theta+1}) + \mathbf{F}^{I\backslash I^*}(\mathbf{V}_\theta, \mathbf{V}_{\theta-1})$
end for

`END_COMM`$(0, \cdots, \Theta[i]+1)$

// Compute the inter-processor interface SSR ($\mathcal{K}[I^*]$)
for $\theta = 0$ to $\Theta[i]$ **do**
$\mathbf{F}_\theta \leftarrow \mathbf{F}_\theta + \mathbf{F}^{I^*}(\mathbf{V}_\theta, \tilde{\mathbf{V}}_\theta) + \mathbf{F}^{I^*}(\mathbf{V}_\theta, \tilde{\mathbf{V}}_{\theta+1}) + \mathbf{F}^{I^*}(\mathbf{V}_\theta, \tilde{\mathbf{V}}_{\theta-1})$
$\mathbf{K}^{\Pi[\theta]}_\theta \leftarrow \mathbf{M}^{-1}_\theta \mathbf{F}_\theta$
end for

// Update the solution when needed ($\mathcal{U}$)
for $\theta = 0$ to $\Theta[i]$ **do**
$h \leftarrow \Delta t_*/2^z$
if $\Pi[\theta] = (\sigma+1)s$ **then**
$\mathbf{U}_\theta \leftarrow \mathbf{U}_\theta + h\sum_{j=1}^{\Pi[\theta]} b^{(\sigma)}_j \mathbf{K}^j_\theta$
end if
end for

end for
$\mathbf{U}^{N+1} \leftarrow \mathbf{U}$

Figure 4: Illustration of the parallel multirate algorithm for the illustrative example. At every stage of the algorithm the main operations are highlighted and divided into four columns: (1) the nodes for which the current input have to be computed are marked in blue; (2) the volumes and inter-processor interfaces for which the steady-state residuals have to be computed are colored in red, and cyan arrows indicate that the data is in transfer; (3) the intra-processor steady-state residuals that have to be computed are colored in red; (4) the nodes for which the solutions have to be updated are marked in green. Stages 6 and 7 are omitted since they are equivalent to stages 2 and 3.

to our knowledge, to equitably share the edge-cut weight over all partitions. Accordingly, even if the workload is perfectly shared by all the partitions at

each stage of the algorithm, the communication to computation ratio and the number of extra interface SSR can vary strongly from one partition to the other and from one sub-time step to the other. These factors can affect the parallel efficiency, especially when the problem is subject to a combination of a large number of multirate groups with small workloads per partition. This is because communication may become cumbersome for computer clusters with an important latency and small bandwidth speed. Furthermore, while for large problems the amount of extra interface residuals may be negligible compared to the rest of the computational work, it may not be the case for a small number of elements per processor, a high number of multirate groups and a significant weighted edge-cut.

4.5. Evaluation of parallel efficiency

The parallel efficiency of our algorithm will be evaluated by measuring the strong scalability of the applications. The problem size stays fixed whilst the number of processors is increased. The *theoretical speedup of the parallel multirate method*, is estimated as:

$$S_{\text{th}}(z^*, p) \;=\; S_{\text{th}}(z^*)p, \tag{26}$$

where $S_{\text{th}}(z^*)$ is the sequential theoretical multirate speedup. The effective efficiency of the parallel multirate algorithm will be evaluated by measuring wall clock time (sum of the CPU time, I/O time and the communication overhead) for different numbers of processors. For each experiment we consider the reference wall clock time as the maximum of the measured times amongst all the processors involved. The *effective speedup of the overall parallel multirate algorithm* can be estimated with the following ratio:

$$S_{\text{ef}}(z^*, p) \;=\; \frac{WT(0,1)}{WT(z^*, p)}, \tag{27}$$

where $WT(0,1)$ represents the wall clock time of the singlerate method on a single processor and $WT(z^*, p)$ the wall clock time of the multirate method on p processors. The *intrinsic parallel speedup of a multirate algorithm* is computed as follows:

$$R_{\text{ef}}(z^*, p) \;=\; \frac{WT(z^*, 1)}{WT(z^*, p)}, \tag{28}$$

The theoretical value of this speedup is $R_{\text{th}}(z^*, p) = p$.

5. Numerical Results

Three applications have been selected to validate the parallel multirate strategy and evaluate its efficiency compared to the equivalent singlerate strategy. The first two deal with two-dimensional ocean modeling. The numerical simulations are performed with the **S**econd-generation **L**ouvain-la-Neuve **I**ce-ocean **M**odel (SLIM[3]) [33, 34, 35, 36] developed by our team. This model makes use of DG finite elements on unstructured meshes. Firstly, the three different partitioning strategies will be compared in terms of parallel efficiency and partitioning quality for an academic benchmark, the Stommel Gyre. Likewise, the influence of the number of multirate groups will be examined. Secondly, the parallel performance will be evaluated for a realistic application, a tsunami wave. Finally, the parallel multirate algorithm will be applied to a three-dimensional case of acoustic propagation in an idealized turbofan engine intake. For all applications, DG elements are used for the spatial discretization whilst the two stage, second order multirate scheme (RK2aC) is used for the temporal integration. The unstructured meshes are built by means of the open source software GMSH[4] [37]. For the mesh partitioning, a tolerance of 1.03 (minimum advised by MeTIS) is imposed for each constraint for all experiments

The parallel code will be tested for parallel efficiency by analyzing the strong scaling results on both the "ace50"[5] and the "Lemaître 2"[6] clusters. Whilst all the nodes are exploited on the "ace50" cluster, only a fraction of the nodes are used on the "Lemaître 2" cluster. When a node is exploited, it is used in an exclusive mode, meaning that no other job is running on it.

In our code, the major part of the operations use highly optimized BLAS[7] (Basic Linear Algebra Subprograms) level 3 (BLAS3) routines to perform matrix-matrix multiplications which represent a substantial fraction of the total wall clock time (see for example [38]). The BLAS implementation is

[3] http://www.climate.be/SLIM

[4] http://www.geuz.org/gmsh/

[5] 16 computing nodes with 2 processor of 4 cores each, Intel Xeon(R)L5420 at 2.50GHz (total of 128 cores). They are interconnected by Gigabit ethernet. For each node the memory is 16 GB and the cache sizes are L1=64K, L2=6144K, L3=0K.

[6] 112 computing nodes with 2 processors of 6 cores each, Intel Xeon(R) E5649 at 2.53 GHz (total of 1344 cores). They are interconnected by Infiniband QDR. For each node the memory is 48 GB and the cache sizes are L1=64K, L2=256K, L3=12288K.

[7] http://www.netlib.org/blas/

the one of the Intel(R) Math Kernel Library[8] (MKL). The multiple levels of caches in the memory hierarchy of modern processors are managed in an efficient way for these operations. For most of the experiments in this section, we will observe strong scalings that are close to the expected parallel speedups. The scalability results show even slightly super-linear speedups for a reasonable number of processors per computer node. This effect is basically due to increased total cache capacity and less contention for memory bandwidth.

5.1. Wind driven circulation in a square basin

The mean horizontal velocity vector $\mathbf{u}$ and the free-surface elevation η for shallow waters are computed by means of a depth-averaged barotropic 2D model. We consider the following non-conservative shallow water equations:

$$\frac{\partial \eta}{\partial t} + \boldsymbol{\nabla} \cdot \big((h+\eta)\mathbf{u}\big) \;=\; 0, \tag{29}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot (\boldsymbol{\nabla} \boldsymbol{u}) + f\mathbf{k} \times \mathbf{u} + g\boldsymbol{\nabla}\eta \;=\; \frac{1}{H}\boldsymbol{\nabla} \cdot \big(H\nu(\boldsymbol{\nabla}\boldsymbol{u})\big) + \frac{\boldsymbol{\tau}^s - \boldsymbol{\tau}^b}{\rho H}, \tag{30}$$

where f, g, ν and ρ are respectively the Coriolis parameter, the gravitational acceleration, the horizontal eddy viscosity and the mean water density. The actual water depth is $H = h + \eta$, where h is the reference water depth below the mean sea level. The bottom and wind stresses are parametrized as $\boldsymbol{\tau^b}$ and $\boldsymbol{\tau^s}$ respectively. The equations are discretized in space with DG finite elements for both elevation and velocity fields.

The Stommel Gyre test case [39] simulates a wind driven circulation in a closed square basin defined on the domain $[0, L] \times [0, L]$ where $L = 10^6\ m$ with a constant seabed defined by $h(x, y) = 1000\ m$. The wind forcing is defined by the following expression:

$$\boldsymbol{\tau^s} \;=\; 0.1 \sin\big(\pi \frac{y}{L}\big)\boldsymbol{e}_x, \tag{31}$$

while the bottom stress is a linear dissipation term defined as $\boldsymbol{\tau^b} = -\rho h \gamma \mathbf{u}$ which balances the forcing, with $\gamma = 10^{-6}\mathrm{s}^{-1}$. For this test case, the viscosity parameter, ν, is set to zero. A detailed description of this test case with the

[8] http://software.intel.com/en-us/intel-mkl

steady solution can be found in [33]. Since the source term is independent of time the system is autonomous and Algorithm 3 may be used as it stands.

An unstructured mesh, composed of 647,100 triangles, was generated with prescribed element sizes at the four corners of the square basin. The element sizes vary linearly from the upper-left corner to the three others corners. Discontinuous elements of order 1 are used for the spatial discretization. Since the bathymetry, h, is constant everywhere, the stable time steps associated to the elements depend only on their characteristic size. The radius of the inscribed circle of the triangle is used as element size measurement. The smallest and largest stable time steps of the problem are $\Delta t_m \approx 0.0179$s and $\Delta t_M = 3.2159$s, respectively. The resulting ratio is roughly 180.

Considering the RK2aC multirate method, this problem allows a maximum of 8 temporal refinement levels, meaning that $z^+ = 7$ and $\Delta t_+ \approx 2.2868$s . The theoretical speedup corresponding to this multirate setup is $S_{th}(z^+) \approx 6.8542$ and there are 15 different multirate groups (8 bulk and 7 buffer). Table 7 gives some relevant information about the multirate groups. The buffer elements cover about 9.37 % of the entire mesh. Multirate groups are illustrated in Fig. 5a where buffer elements are colored in black and bulk elements are colored depending on their effective multirate time step.

Table 7: Stommel Gyre: distribution of mesh elements in multirate groups with corresponding multirate exponents and loads, for $z^* = z^+$.

θ	0	1	2	3	4	5	6	7
$\lvert\Omega_\theta^h\rvert$	15543	4187	51415	7702	72188	8646	84289	10465
z	7	6	6	5	5	4	4	3
λ_θ	128	128	64	64	32	32	16	16

θ	8	9	10	11	12	13	14
$\lvert\Omega_\theta^h\rvert$	95064	12260	108459	13728	158624	3631	899
z	3	2	2	1	1	0	0
λ_θ	8	8	4	4	2	2	1

The classical, single-constraint and multi-constraint partitioning strategies are compared in terms of partitioning quality and resultant parallel performance, for $z^* = z^+$. The average communication volume per partition

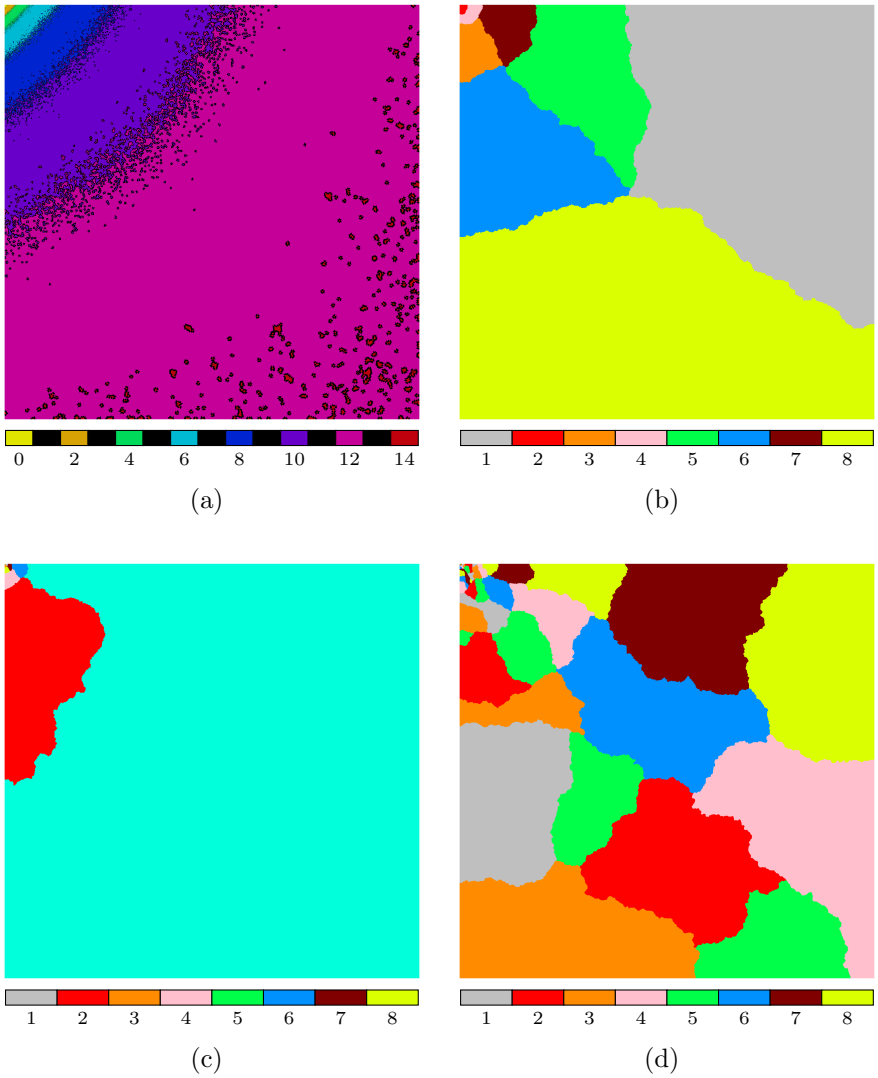


Figure 5: Stommel Gyre: mesh made up of 647,100 triangles, multirate setup $z^* = z^+$. Multirate tags with buffer elements in black (a). Distribution of mesh elements among 8 partitions resulting from (b) the classical, (c) the single-constraint and (d) the multi-constraint partitioning strategies.

for a Δt_+ period is shown in Fig. 6a. It turns out that the multi-constraint partitioning strategy yields an important weighted edge-cut, C, due to the multiple constraints involved. The single-constraint partitioning strategy

yields more inter-processor communications than the classical one because many weighty interfaces are involved. However, the average communication volume tends to decrease much faster with the number of partitions for the multi-constraint partitioning strategy. As an example, consider the distribution of the mesh elements among 8 partitions for the classical, Fig. 5b, the single-constraint, Fig. 5c, and the multi-constraint, Fig. 5d, partitioning strategy. Observe that only the first two partitioning strategies yield contiguous partitions. The average communication to computation ratio per partition and per time unit, see Fig. 6b, becomes rapidly more significant with the number of partitions for the multi-constraint partitioning strategy than for the two others. Hiding communication by computations will become increasingly difficult.

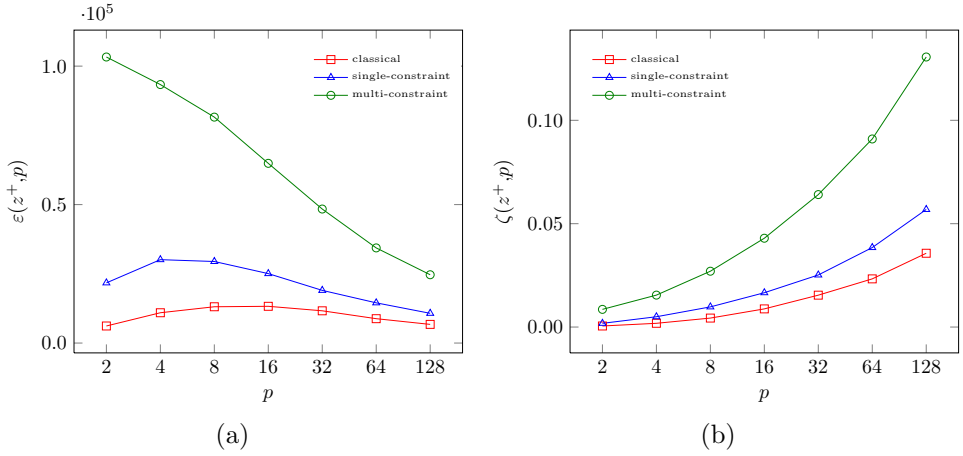


Figure 6: Stommel Gyre: comparison of (a) the average communication volume per partition for a Δt_+ period, see Eq. (24), and (b) the average communication to computation ratio per partition and per time unit, see Eq. (25) for the three partitioning strategies.

The theoretical global load imbalance L, that depends on the number of partitions p and the partitioning strategies, is illustrated in Fig. 7a. The multi-constraint partitioning strategy yields global imbalances that stays below the prescribed tolerance, 1.03 (the highest load imbalance is roughly 1.02 for 128 partitions). For the other two strategies, the global load imbalance grows critically with p and could therefore be crippling for parallel performance. The effective global load imbalances, measured for a $5\Delta t_+$ period, confirm the general theoretical tendency for the classical and multi-constraint partitioning strategies, see Fig. 7b. But the single-constraint partitioning

strategy appears to behave significantly better than expected, especially for a high number of processors. This phenomenon is due to the non-blocking communications that allow processors to continue computations while some other partitions are not yet at the same stage in the algorithm. We added barriers (meaning that all processors have to wait until the slowest one finishes its work) at each stage of the global algorithm, when messages need to be received. A better matching with the theoretical expectations is observed for the single-constraint strategy, see dashed lines in Fig. 7b. The effective imbalance for the multi-constraint partitioning strategy on 128 processors is probably due to the high requirements in simultaneous memory access as well as the high communication to computation ratio. For this configuration, the full 128 cores of the "ace50" cluster are exploited.

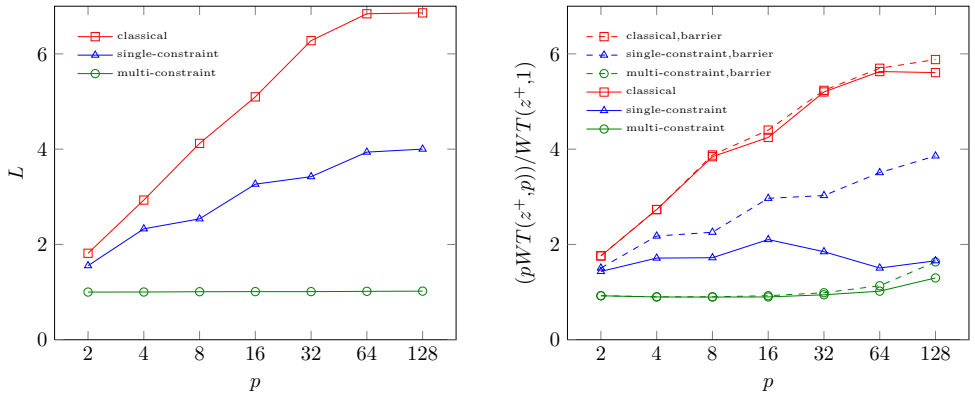


Figure 7: Stommel Gyre: comparison of (a) the theoretical and (b) the measured load imbalances for the three partitioning strategies. The wall clock times are evaluated for a $5\Delta t_+$ period with and without a barrier when receiving the messages.

The strong scaling results computed with respect to the wall clock times measured for the multirate setup on one processor are shown in Fig. 8a. The multi-constraint partitioning strategy attains almost perfect scaling. Fig. 8b shows the overall speedup of the parallel multirate algorithm compared to the wall clock time measured for the equivalent singlerate method on one processor. The multi-constraint partitioning strategy preserves the theoretical multirate speedup with the increasing number of processors.

The acceleration of multirate *versus* singlerate depends on the number of temporal refinement levels. Table 8 gives the theoretical multirate speedups

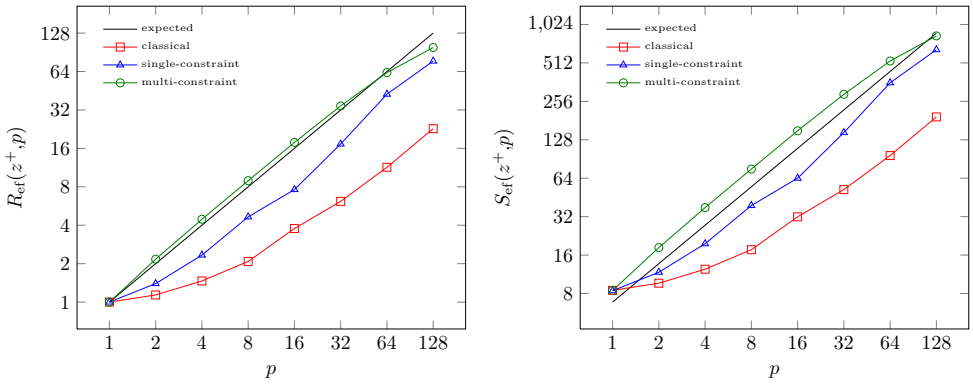


Figure 8: Stommel Gyre: comparison of the parallel performance, on the "ace50" cluster, for the three parallel strategies in terms of (a) the intrinsic parallel multirate speedup, see Eq. (28), and (b) the overall speedup of the parallel multirate algorithm compared to the singlerate case, see Eq. (27). These experiments are based on the multirate setup for which $z = z^+$ and for a $5\Delta t_+$ period.

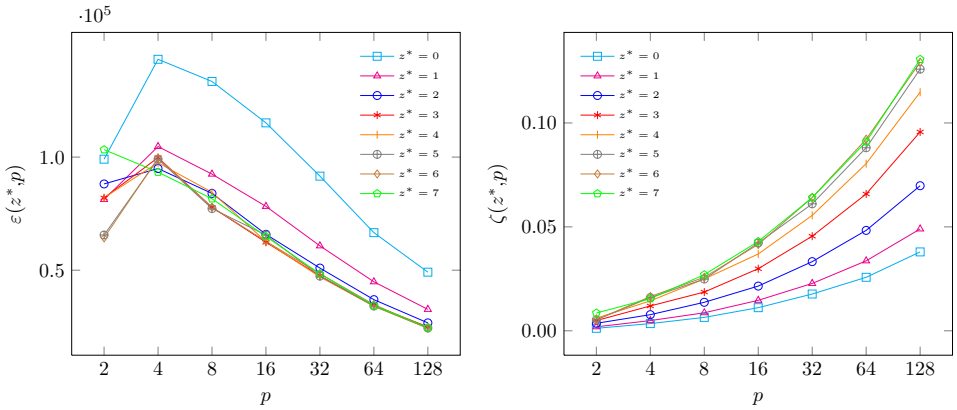
in function of the maximum multirate exponent, z^*. As expected, this speedup reaches a threshold for a certain multirate exponent. Indeed, at a certain point, the fraction of elements that could use a larger time step becomes negligible.

Table 8: Stommel Gyre: theoretical multirate speedups depending on the maximum multirate exponent z^*.

z^*	0	1	2	3	4	5	6	7
S_{th}	1.0000	1.9408	3.3817	4.9615	6.1115	6.6735	6.8537	6.8542

In Fig. 9a we compare the average communication volume per partition for different z^*. Fewer and fewer computations need to be performed with an increasing number of temporal refinements, but also the amount of communications for a fixed period decreases. Apart from the case where $p = 2$, the general tendency is that the total communication volume, for a fixed period, decreases significantly when the multirate speedup increases. By contrast, Fig. 9b shows that the average communication to computation per partition

and per time unit increases significantly when z^* is increased. The total communication volume decreases with the number of multirate groups and the number of partitions but much more slowly than the computational load.



(a) (b)

Figure 9: Stommel Gyre: comparison of (a) the average communication volume per partition for a Δt_+ period, see Eq. (24), and (b) the average communication to computation ratio per partition and per time unit, see Eq. (25) for the eight different z^*.

The global imbalances resulting from the multi-constraint partitioning for the different z^* are depicted in Fig. 10. At least up to 128 partitions, the global imbalance remains under the prescribed tolerance for each multirate setup. If L tends to grow with p and z^*, the evolutions are far from being smooth. The multi-constraint algorithm of MeTiS remains a heuristic that tries to respect the prescribed tolerance while minimizing the weighted edge-cut. It is thus difficult to predict the load imbalances.

The following parallel performance studies have been performed on the "ace50" cluster. The multi-core computer nodes are filled progressively with the same amount of processes per node. Fig. 11a gives the intrinsic multirate strong scalings as a function of z^* and p. Even if the scalings tend to decrease for large z^* and p they remain satisfactory. The overall multirate speedups are globally preserved for increasing p, as illustrated by Fig. 11b. As an example, for $z^* = 7$ and $p = 64$ the global speedup is roughly 541.5, which is much higher than the 438.6 expected theoretically.

From these experiments, it turns out that using a high number of multirate groups still yields generally the best overall speedup even though it implies more constraints on the partitioning. The adverse effects of adding

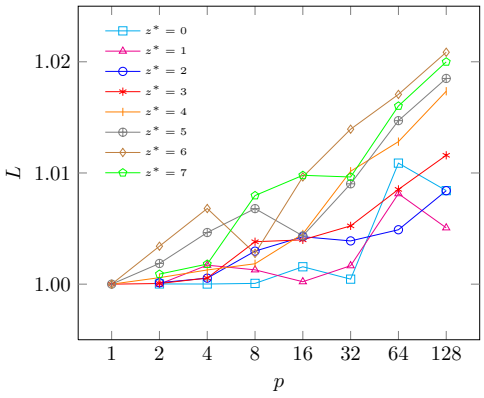


Figure 10: Stommel Gyre: comparison of the global load imbalance resulting of the multi-constraint partitioning strategy for the eight different z^*.

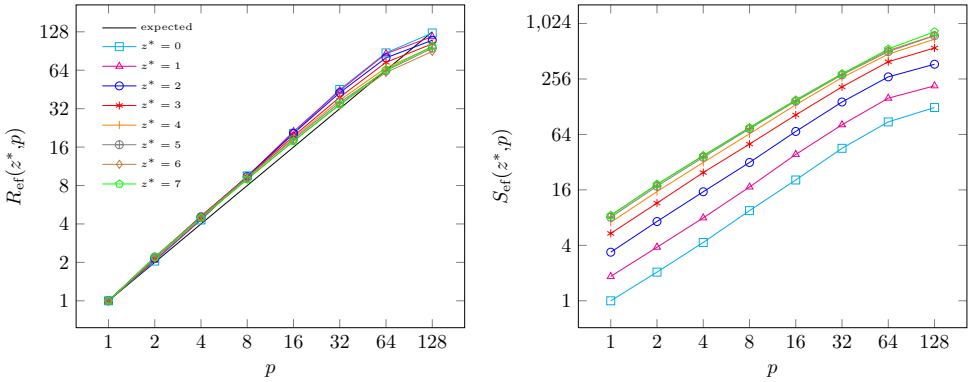


Figure 11: Stommel Gyre: comparison of the parallel performance, on the "ace50" cluster, for the eight different z^* in terms of (a) the intrinsic parallel multirate speedup, see Eq. (28) and (b) the overall parallel multirate speedup compared to the singlerate case, see Eq. (27). Wall clock times have been measured for a $5\Delta t_+$ period.

multirate groups are mostly not sufficient to jeopardize the intrinsic multirate speedup of the problem. However, for significantly high numbers of multirate groups and processors it is probably advisable to find a trade-off between small variations in multirate speedup and the number of partitioning constraints.

5.2. Propagation of a tsunami wave

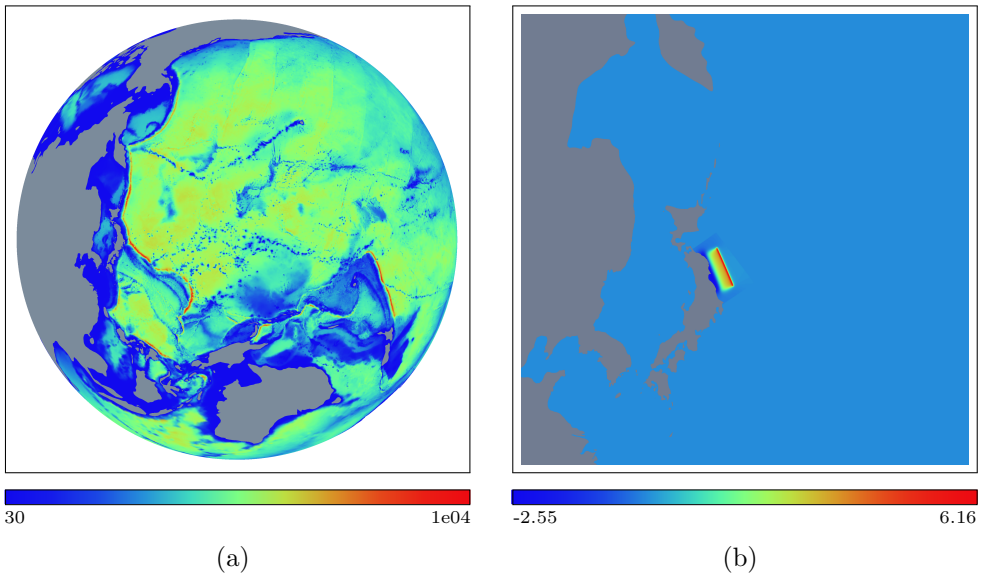
Using the shallow water model defined by Equations (29) and (30), we now consider the simulation of the tsunami that struck Japan on 11 March 2011. The water was put in motion due to a sudden vertical shift of the water column. It was generated by an earthquake near the coast of Honshu. A mesh of the world ocean made up of 1,757,467 triangles was generated using GMSH [37]. Element sizes are prescribed in order to fit the topography of the coastlines, take into account the bathymetry [40] and increase the resolution in a region of interest (around Japan). The spatial discretization is carried out with DG elements of polynomial order 2.

We solve the shallow water equations, Eq. (29), on the sphere using stereographic coordinates [41]. The viscosity parameter ν is zero and there is no wind forcing, i.e. $\boldsymbol{\tau}^s = \mathbf{0}$. The bottom stress is parametrized with the Chézy-Manning-Strickle formulation [42]:

$$\frac{\boldsymbol{\tau}^b}{\rho} = n^2 g \frac{||\boldsymbol{u}||\boldsymbol{u}}{H^{1/3}}, \tag{32}$$

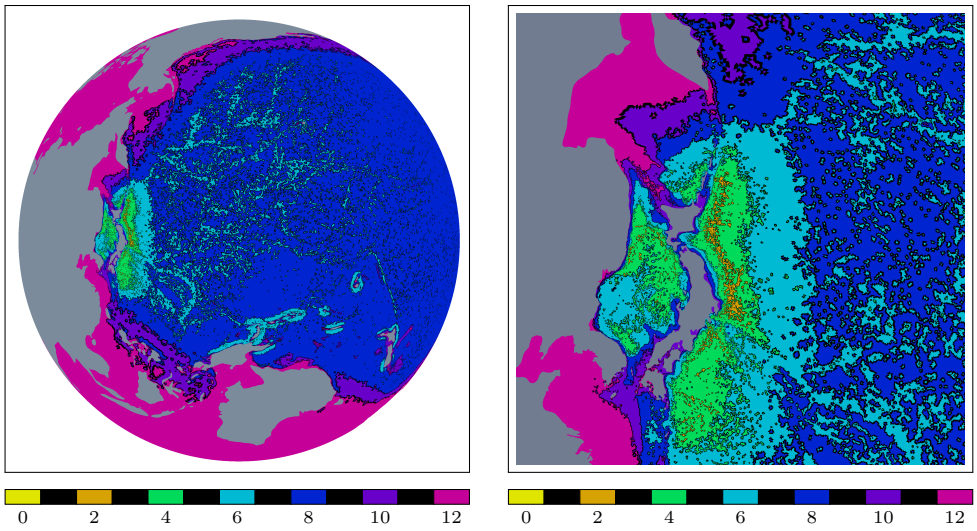
where the Manning coefficient n is equal to 0.03 $s/m^{1/3}$. Impermeable boundary conditions are imposed on all coastlines. The initial condition is illustrated in Fig. 12b while the bathymetry is represented for the world ocean in Fig. 12a. The bathymetry has been smoothed and cropped at 30 m water depth.

The stable time steps are computed as a function of the mesh size and actual water depth at the initial time (sum of bathymetry and initial condition). The minimum and maximum element-wise stable time steps are 0.116 s and 229.979 s, respectively. We selected a maximum multirate exponent $z^* = 6$ (corresponding to a reference time step $\Delta t_* = 7.449$ s), despite the maximum number of possible temporal refinements being $z^+ = 10$. The corresponding theoretical multirate speedup is roughly 9.32 and is only 0.17 % lower than for the $z^* = 10$ case, yet we omit four temporal refinement levels. Consequently, fewer constraints are needed for the mesh partitioning. The multirate groups for this multirate setup are depicted for the world ocean, Fig. 13a, and a zoom around Japan, Fig. 13b. The distribution of the elements in the multirate groups with the corresponding multirate exponents and loads is given in Table 9. The buffer elements cover roughly 14.69 % of the entire mesh.



(a) (b)

Figure 12: Japanese tsunami: bathymetry (a) and initial condition (b).



(a) (b)

Figure 13: Japanese tsunami ($z^* = 6$): multirate groups.

The multi-constraint partitioning strategy is used to partition the mesh for the considered multirate setup. Fig. 14a compares the average commu-

Table 9: Japanese tsunami ($z^* = 6$): distribution of mesh elements in multirate groups with corresponding multirate exponents and loads.

θ	0	1	2	3	4	5	6
$\lvert\Omega_\theta^h\rvert$	882	1945	31885	25216	140929	39077	240338
z	6	5	5	4	4	3	3
λ_θ	64	64	32	32	16	16	8

θ	7	8	9	10	11	12
$\lvert\Omega_\theta^h\rvert$	150981	913257	28762	83076	12189	88930
z	2	2	1	1	0	0
λ_θ	8	4	4	2	2	1

nication volume per partition for a fixed period Δt_* for the singlerate and multirate setups. The average communication to computation ratio per partition and time unit are represented in Fig. 14b, for both multirate and singlerate. Note that these indicators are normalized with respect to to the effective multirate loads λ_θ despite the fact that we are dealing with a second order spatial discretization. As for the Stommel Gyre test case, it turns out that even if the total communication volume is lower in the multirate case for a fixed period, the communication to computation ratio increases critically faster for the multirate strategy. The global load imbalances, L, for the multirate case all remain below the prescribed tolerance (1.03) up to 128 partitions. For $p = 256$ we have $L \approx 1.04$.

An illustration of the distribution of the mesh elements among 16 partitions, resulting from the multi-constraint strategy, is given in Fig. 15. The mesh partitions are not contiguous, yet locally we can observe that the elements are well organized in compact patches.

The following parallel efficiency analyses were performed on the "Lemaître 2" cluster. Remember that we are showing strong scaling results. Only 32 nodes of the cluster were exploited to perform the efficiency measurements. For each experiment the processes are equally distributed among all nodes. We performed tests up to 256 processors, which represents 8 processes per node. The intrinsic strong scalings of multirate and singlerate are compared in Fig. 16a, while the overall parallel speedups of both methods normalized with respect to the wall clock time taken on one processor for the singlerate case are illustrated in Fig. 16b. In Table 10, we show the speedups and

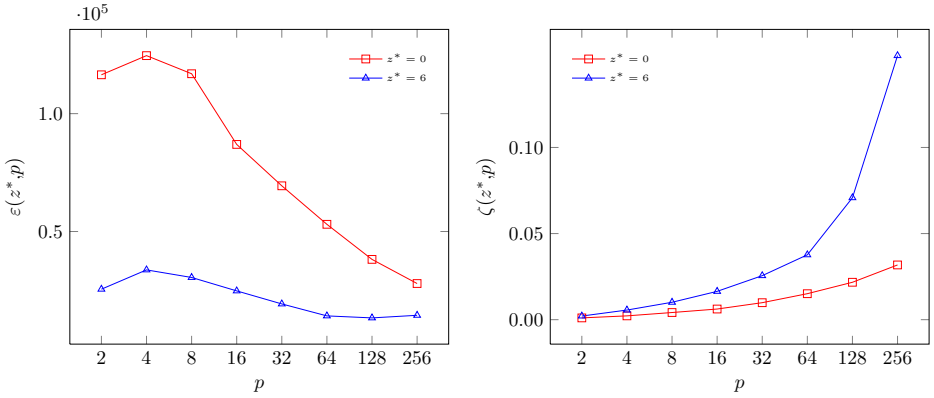


Figure 14: Japanese tsunami ($z^* = 6$): (a) average communication volume per partition for a period Δt_*, see Eq. (24); (b) average communication to computation ratio per partition and time unit, see Eq. (25).

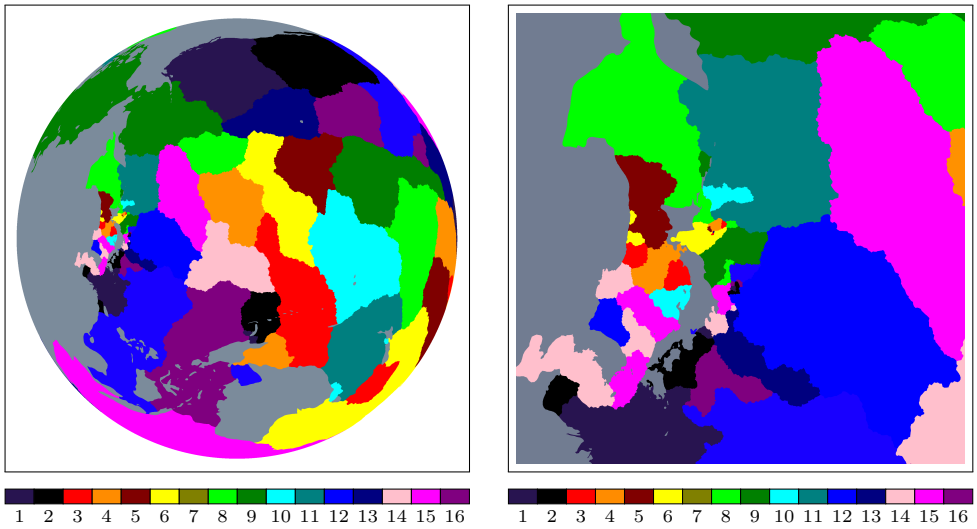


Figure 15: Japanese tsunami ($z^* = 6$): distribution of the mesh elements among 16 partitions resulting of the multi-constraint partitioning strategy.

the associated parallel efficiencies expressed as percentages. A slightly better scaling is observed for the multirate version. The theoretical multirate speedup is preserved up to 128 processors. The scaling, for both approaches,

starts to deflect slightly at around 128 processors. This is most likely due to the high number of processors per node, which requires many simultaneous memory accesses.

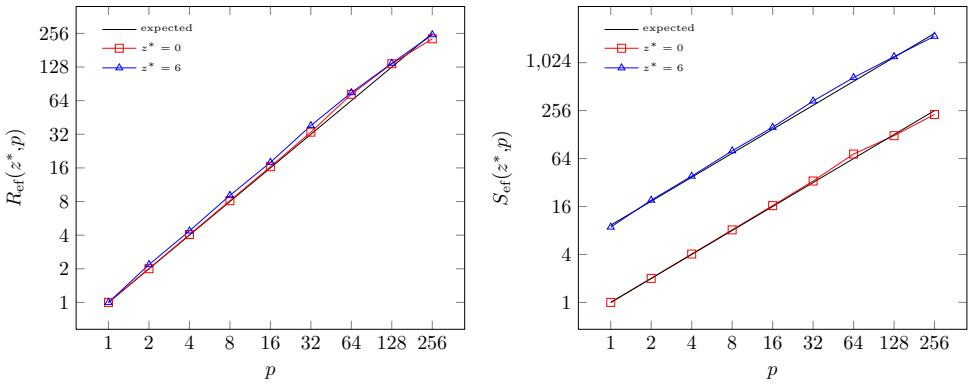


Figure 16: Japanese tsunami ($z^* = 6$): Comparison of the parallel performance, on the "Lemaître 2" cluster, for singlerate and multirate in terms of (a) the intrinsic parallel multirate speedup, see Eq. (28), and (b) the overall parallel multirate speedup, see Eq. (27). The wall clock times have been measured for a Δt_* period.

Table 10: Parallel strong scaling for the Japanese tsunami application.

p	1	2	4	8	16	32	64	128	256
$R_{\text{ef}}(1,p)$	1.00	2.00	4.05	8.14	16.40	33.44	72.91	124.03	228.96
%	100.00	100.08	101.18	101.72	102.50	104.49	113.92	96.90	89.44
$R_{\text{ef}}(6,p)$	1.00	2.18	4.36	9.08	17.92	38.27	75.44	138.19	249.05
%	100.00	108.91	109.11	113.44	112.02	119.61	117.88	107.96	97.29
$S_{\text{ef}}(6,p)$	8.82	19.20	38.48	80.01	158.02	337.43	665.11	1218.33	2195.71
%	94.59	103.02	103.21	107.31	105.97	113.14	111.51	102.13	92.03

Snapshots of the simulated sea surface elevation are shown in Fig. 17 for both singlerate and multirate after two and four hours of physical time. These simulations were performed on 64 processors on the "Lemaître 2" cluster. As the spatial error is largely dominant compared to the temporal errors, the solutions are quasi identical with singlerate and multirate time integrators.

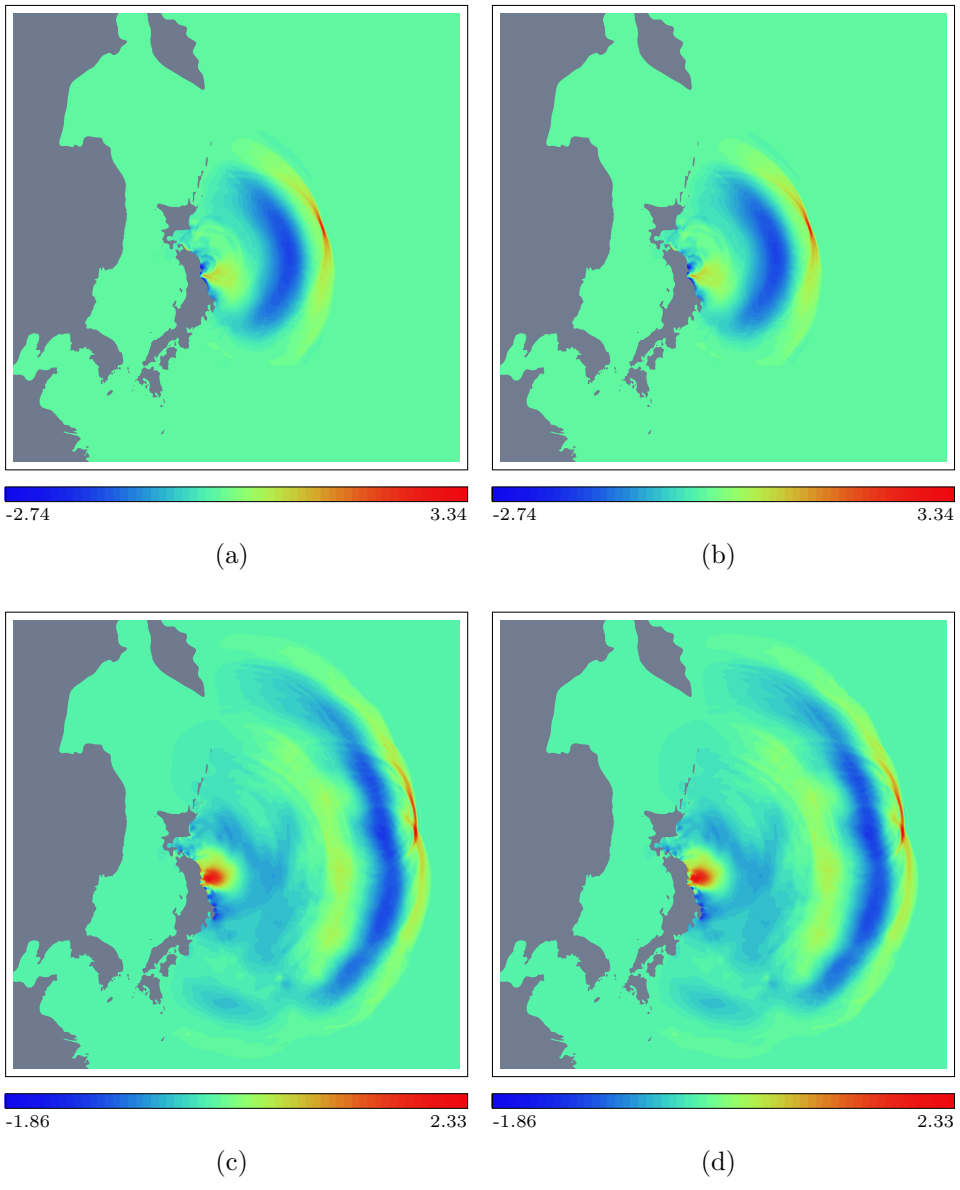


Figure 17: Japanese tsunami: sea surface elevation in meters at T=2h with singlerate (a) and multirate (b); T=4h with singlerate (c) and multirate (d).

5.3. Acoustic propagation in a turbofan engine intake

The last application deals with the aeroacoustics of an idealized turbofan engine intake [43]. The three-dimensional geometry, shown in Fig. 18a, is a

cylindrical duct of slowly-varying cross-section. The duct is annular on the fan side, due to the conical part that represents the spinner, and becomes hollow on the inlet side. We consider a compressible, isentropic and irrotational mean flow with Mach number of 0.5 in the negative-x direction. The aim of the simulation is to compute the propagation of acoustic perturbations in this non-uniform mean flow, with an acoustic excitation in the fan plane corresponding to the first radial mode, the azimuthal mode $m = 10$ and a dimensionless angular frequency $\omega = 16$. The case is extensively described in [43].

The equations that govern the evolution of inviscid perturbations about a non-uniform mean flow are the linearized Euler equations:

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}_0 + \rho_0 \boldsymbol{u}) = 0, \tag{33}$$

$$\frac{\partial (\rho_0 \boldsymbol{u})}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{u}_0 \otimes \rho_0 \boldsymbol{u} + \boldsymbol{I} p) = \boldsymbol{0}, \tag{34}$$

$$\frac{\partial p}{\partial t} + \boldsymbol{\nabla} \cdot (c_0^2 \rho_0 \boldsymbol{u} + p \boldsymbol{u}_0) = 0, \tag{35}$$

where the unknowns $(\rho, \boldsymbol{u}, p)$ are respectively the perturbations in density, three-dimensional velocity vector and pressure, and $(\rho_0, \boldsymbol{u}_0, p_0)$ are the corresponding quantities for the mean flow; c_0 is the local speed of sound. The coupling between entropy, vorticity and acoustic modes due to the non-uniform mean flow is neglected, because we are only interested in acoustic perturbations of an irrotational mean flow [44].

Acoustically rigid wall boundary conditions are prescribed on the duct and the spinner. Non-reflecting boundary conditions are imposed in the inlet and fan planes, with a superposed acoustic excitation in the fan plane. The analytical solutions derived in [43] are used to impose the mean flow and the acoustic excitation.

Simulations are run with a DG spatial discretization of polynomial degree of 3. The mesh, shown in Fig. 18, is composed of 49,281 tetrahedral elements. The uniform element size enforced in most of the domain is determined by the requirements on numerical dissipation (attenuation lower than 0.01 dB per wavelength). However, the mesh has to be refined near the tip of the spinner, in order to correctly represent the curvature of the geometry. This

refinement yields an important gap between the smallest and largest stable time steps associated with the tetrahedra ($\Delta t_m = 2.8 \times 10^{-5}$ s and $\Delta t_M = 1.459 \times 10^{-3}$ s).

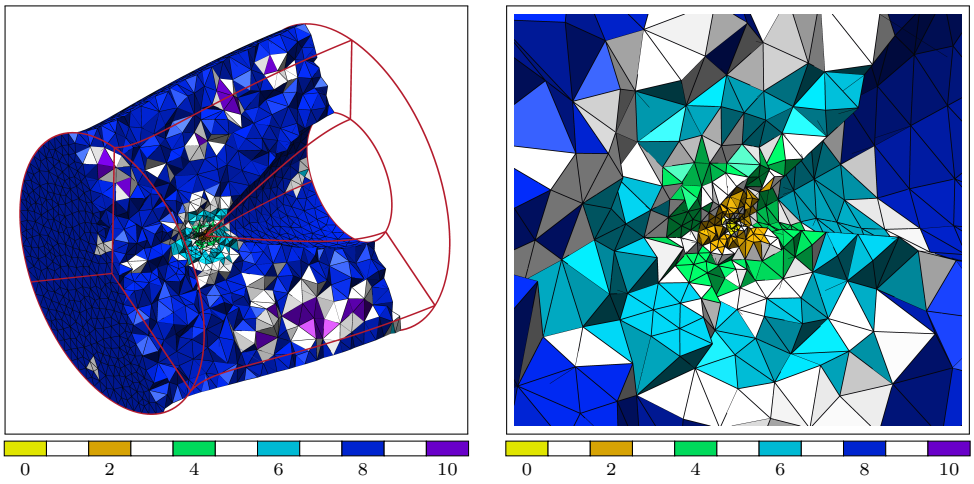
The multirate setup associated with the turbofan problem is illustrated by Table 11 and Fig. 18. We chose to use the maximum number of temporal refinements possible, meaning that $z^* = z^+ = 5$ and $\Delta t_* = 8.9 \times 10^{-4}$. Buffer elements represent 13.43 % of the total number of mesh elements. About 76 % of the elements belong to multirate group 8 which means that a substantial theoretical multirate speedup is achieved, i.e. $S_{\text{th}}(5) \approx 10.932$. From this, it also follows that only few elements remain in the other multirate groups which will complicate the partitioning. As an example, 286 mesh elements belong to the bulk and buffer groups that have the maximum multirate load ($\theta = 0$ and $\theta = 1$) meaning that, for 256 processing elements, roughly 1.12 elements of these groups are required on each processor.

Table 11: Turbofan ($z^* = 5$): distribution of elements among 13 multirate groups with corresponding multirate exponents and loads.

θ	0	1	2	3	4	5	6	7	8	9	10
$\lvert\Omega_\theta^h\rvert$	116	170	684	519	1272	685	2500	2015	37595	3229	496
z	5	4	4	3	3	2	2	1	1	0	0
λ_θ	32	32	16	16	8	8	4	4	2	2	1

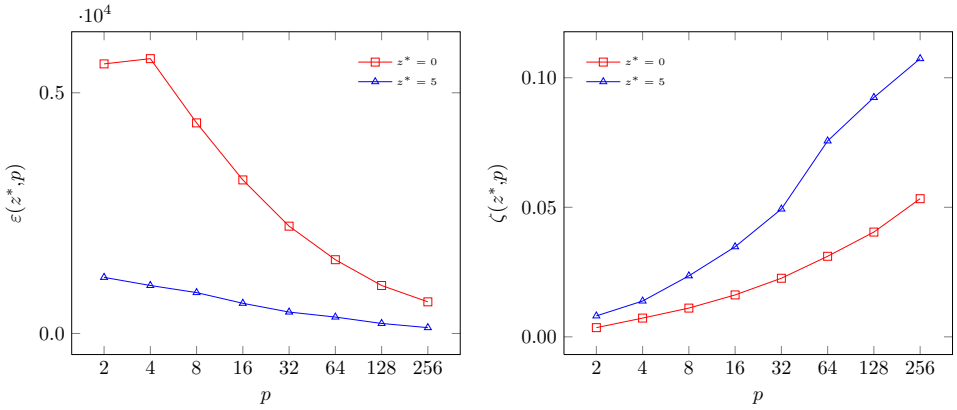
We analyze the quality of the mesh decompositions resulting from the multi-constraint heuristic of MeTiS. The estimated communication volume per partition for a Δt_* period are given for multirate and singlerate in Fig. 19a. The average communication to computation ratio per partition and per time unit are depicted for both methods in Fig. 19b.

The "Lemaître 2" cluster was used in the same manner as for the tsunami application. The intrinsic parallel scaling is represented for multirate and singlerate in Fig. 20a. The overall speedups compared to the wall clock time for the singlerate method on a single processor are illustrated in Fig. 20b. Table 12 contains the speedups and the associated parallel efficiencies expressed as percentages. Despite the small number of elements per partition for some multirate groups for a high number of processors, the scalings show that parallel multirate still yields important benefits compared to singlerate.



(a) (b)

Figure 18: Turbofan ($z^* = 5$): distribution of the elements among 13 multirate groups for the whole engine (a) and a zoom near the tip of the spinner.



(a) (b)

Figure 19: Turbofan ($z^* = 5$): (a) average communication volume per partition for a period Δt_*, see Eq. (24); (b) average communication to computation ratio per partition and time unit, see Eq. (25).

To illustrate the numerical results, the instantaneous pressure field is shown in Fig. 21. Those simulations were performed on 32 processors and the total wall clock times for singlerate and multirate were 1.133×10^5 s and

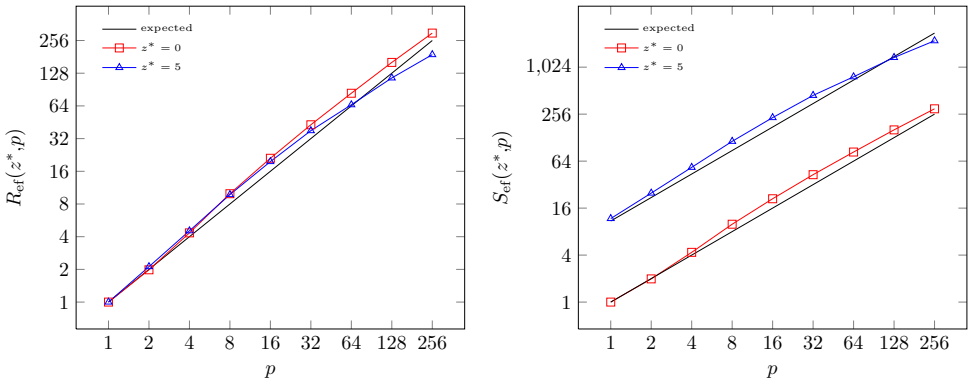


Figure 20: Turbofan ($z^* = 5$): Comparison of the parallel performance on the "Lemaître 2" cluster for singlerate and multirate in terms of (a) the intrinsic parallel multirate speedup, see Eq. (28), and (b) the overall parallel multirate speedup, see Eq. (27). The wall clock times have been measured for a Δt_* period

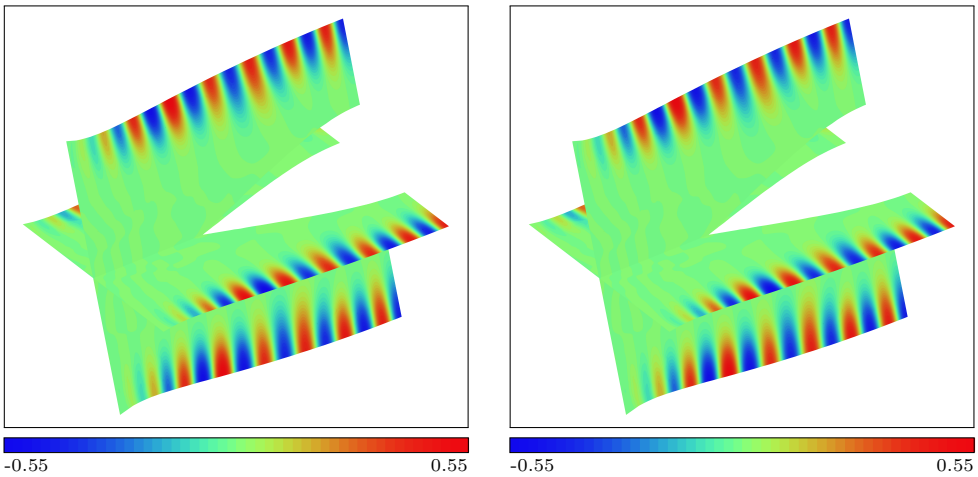
Table 12: Parallel strong scaling for the turbofan application.

p	1	2	4	8	16	32	64	128	256
$R_{\text{ef}}(1,p)$	1.00	1.98	4.34	9.95	21.09	42.90	83.71	160.94	299.68
%	100.00	99.14	108.45	124.35	131.83	134.07	130.79	125.73	117.06
$R_{\text{ef}}(5,p)$	1.00	2.12	4.53	9.76	19.66	37.86	65.58	115.76	189.76
%	100.00	105.94	113.27	122.01	122.86	118.32	102.47	90.44	74.13
$S_{\text{ef}}(5,p)$	11.76	24.91	53.26	114.73	231.08	445.05	770.88	1360.73	2230.64
%	107.53	113.91	121.79	131.19	132.11	127.22	110.18	97.24	79.71

1.084×10^4 s, respectively. This represents a speedup of roughly 10.457 , which is close to the theoretical one.

6. Conclusion and discussion

In this paper, we have proposed a strategy to parallelize explicit multirate schemes in the framework of DG methods. In particular, we have developed a generic parallel algorithm for the multirate ERK methods introduced by Constantinescu and Sandu. The multi-constraint partitioning library of MeTiS is used to distribute the mesh elements amongst the desired number of processors in an adequate manner for the considered multirate setup. The key idea is to ensure that at each stage of the multirate algorithm, the same number of cells are active on each processor, while minimizing the number



(a) (b)

Figure 21: Turbofan: pressure field on two orthogonal planes after roughly 3.64 s for (a) the singlerate, $z^* = 0$, and (b) the multirate, $z^* = 5$, case.

of inter-processor communications. In this way, we expect that the computational load is equally shared in space and time amongst the processors and that the idle times due to synchronization are minimized.

We have evaluated the quality of the domain decompositions and the effective total parallel multirate speedup on several numerical applications. It was shown that the multi-constraint partitioning strategy outperforms the classical and single-constraint partitioning, both theoretically and experimentally. The experiments show that the effect of adding a temporal refinement level has a positive influence on the global performance as long as the theoretical multirate speedup is significantly higher even if it increases the number of constraints for the partitioning.

The parallel performance studies reveal, however, that strong scalability is usually achieved more easily with singlerate than with multirate, especially for a high number of processors compared to the number of mesh elements. Several factors may influence the global multirate parallel performance: critical communication to computation ratios, load imbalances due to imperfect partitioning and small numbers of operations that have to be performed at certain stages of the algorithm. Moreover, some features of modern computer architectures, like the latency and bandwidth of memory access and inter-processor communications, may also affect the efficiency.

This approach is clearly limited when the number of elements per multirate group and per processor is very low. Indeed, singlerate explicit methods have proven high scalability down to a very small number of elements per processor which is not achievable in the parallel multirate context. For high order DG methods in particular, where less elements are used for a same total number of DoF, it may be extremely difficult to obtain as good strong parallel scaling as in the singlerate case, even if the communication to computation ratio is lower than for low order DG. However, we believe that the parallel multirate strategy can yield important speedups compared to the classical explicit methods for many realistic applications combined with a reasonable number of processors. Even in the worst-case scenarios, the parallel multirate strategy should be at least as efficient as in the singlerate case. The limitations caused by a small number of elements per processor are probably far less drastic than for the parallel implementation of implicit and implicit-explicit schemes.

The parallel multirate algorithm can still be improved and optimized by considering several challenging aspects of multi-core programming, which shall be addressed in future work. As an example, we should consider the issue of finding the best distribution of the processes among the different computer nodes. Indeed, the intra-node contiguity could be improved by gathering appropriate partitions onto the same nodes in order to minimize inter-node communications, which are more expensive. Another perspective consists in developing a dynamic version of this parallel multirate strategy to address temporal changes of the element-wise stable time steps. The challenge will then be to find relevant indicators to determine whether or not it is worth modifying the multirate groups and/or repartitioning the mesh to achieve greater computational efficiency. Along the same line, the efficiency and the validity of the parallel multirate strategy should also be evaluated for strongly nonlinear problems where the local time steps are hard to estimate and highly variable in space and time. In this context, dynamic load balancing is imperative because the optimal multirate setup varies dramatically during the simulation, which implies that the computational load shall be evaluated and redistributed many times. The computational cost of these operations could severely impact the global efficiency of the method.

Acknowledgments

Bruno Seny is a Research fellow with the Belgian Fund for Research in Industry and Agriculture (FRIA). Jonathan Lambrechts is supported by grants from the Belgian National Fund for Scientific Research (FNRS). The present study was carried out within the scope of the project "Taking up the challenges of multi-scale marine modeling", funded by the Fédération Wallonie-Bruxelles, as Actions de Recherche Concertées, under contract ARC 10-15/028. Computational resources for the "Lemaître 2" cluster were provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Equipements de Calcul Intensif en Fédération Wallonie-Bruxelles (CECI) funded by the Fond de la Recherche Scientifique de Belgique (FRS-FNRS). The "ace50" supercomputer is a cluster from the Applied and Computational Electromagnetics group at the University of Liège. A significant part of this work has been performed during the stay of Bruno Seny, Jonathan Lambrechts and Vincent Legat at the University of Cambridge (UK), as participants in the research programme Multiscale Numerics for the Atmosphere and Ocean of the Isaac Newton Institute for Mathematical Sciences. We wish to thank our hosts Hilary Weller, David Ham, Matthew Piggott, Todd Ringler, Nigel Wood and all the staff of the Isaac Newton Institute for their kind hospitality and a very stimulating environment. The authors gratefully thanks Christopher Thomas for his careful re-reading of this document.

References

[1] T. Warburton, T. Hagstrom, Taming the CFL number for discontinuous Galerkin methods on structured meshes, SIAM Journal on Numerical Analysis 46 (6) (2008) 3151–3180. doi:10.1137/060672601.

[2] F. Lörcher, G. Gassner, C.-D. Munz, A discontinuous Galerkin scheme based on a space-time expansion I. Inviscid compressible flow in one space dimension, Journal of Scientific Computing 32 (2) (2007) 175–199. doi:10.1007/s10915-007-9128-x.

[3] G. Gassner, F. Lörcher, C.-D. Munz, A discontinuous Galerkin scheme based on a space-time expansion II. Viscous flow equations in multi dimensions, Journal of Scientific Computing 34 (3) (2008) 260–286. doi:10.1007/s10915-007-9169-1.

[4] M. Dumbser, C.-D. Munz, Arbitrary high order discontinuous Galerkin schemes. Numerical methods for hyperbolic and kinetic problems, in: S. Cordier, T. Goudon, M. Gutnic, E. Sonnendrucker (Eds). IRMA Series in Mathematics and Theoretical Physics, EMS Publishing House, 2005, pp. 295–333.

[5] D. S. Balsara, C. Meyer, M. Dumbser, H. Du, Z. Xu, Efficient implementation of ADER schemes for Euler and magnetohydrodynamical flows on structured meshes - Speed comparisons with Runge-Kutta methods, Journal of Computational Physics 235 (2013) 934 – 969. doi:10.1016/j.jcp.2012.04.051.

[6] M. Dumbser, M. Käser, E. F. Toro, An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes V: Local time stepping and p-adaptivity, Geophysical Journal International 171 (2) (2007) 695–717. doi:10.1111/j.1365-246X.2007.03427.x.

[7] M. Dumbser, C.-D. Munz, Building blocks for arbitrary high order discontinuous Galerkin schemes, Journal of Scientific Computing 27 (1-3) (2006) 215–230. doi:10.1007/s10915-005-9025-0.

[8] J. Qiu, M. Dumbser, C.-W. Shu, The discontinuous Galerkin method with Lax-Wendroff type time discretizations, Computer Methods in Applied Mechanics and Engineering 194 (42-44) (2005) 4528 – 4543. doi:10.1016/j.cma.2004.11.007.

[9] L. Liu, X. Li, F. Q. Hu, Nonuniform time-step Runge-Kutta discontinuous Galerkin method for computational aeroacoustics, Journal of Computational Physics 229 (19) (2010) 6874 – 6897. doi:10.1016/j.jcp.2010.05.028.

[10] S. Osher, R. Sanders, Numerical approximations to nonlinear conservation laws with locally varying time and space grids, Mathematics of Computation 41 (164) (1983) 321–336.

[11] C. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM Journal on Scientific Computing 22 (6) (2000) 2256–2281. doi:10.1137/S1064827500367737.

[12] H. Tang, G. Warnecke, High resolution schemes for conservation laws and convection-diffusion equations with varying time and space grids, Journal of Computational Mathematics 24 (2) (2006) 121–140.

[13] E. M. Constantinescu, A. Sandu, Multirate timestepping methods for hyperbolic conservation laws, Journal of Scientific Computing 33 (3) (2007) 239–278. doi:10.1007/s10915-007-9151-y.

[14] M. Schlegel, O. Knoth, M. Arnold, R. Wolke, Multirate Runge-Kutta schemes for advection equations, Journal of Computational and Applied Mathematics 226 (2) (2009) 345–357. doi:10.1016/j.cam.2008.08.009.

[15] W. Hundsdorfer, A. Mozartova, V. Savcenco, Analysis of explicit multirate and partitioned Runge-Kutta schemes for conservation laws, Tech. Rep. MAS-E0715 (2007).

[16] B. Seny, J. Lambrechts, R. Comblen, V. Legat, J.-F. Remacle, Multirate time stepping for accelerating explicit discontinuous Galerkin computations with application to geophysical flows, International Journal for Numerical Methods in Fluids 71 (1) (2013) 41–64. doi:10.1002/fld.3646.

[17] M. Schlegel, O. Knoth, M. Arnold, R. Wolke, Implementation of splitting methods for air pollution modeling, Geoscientific Model Development Discussions 4 (4) (2011) 2937–2972. doi:10.5194/gmdd-4-2937-2011.

[18] B. Cockburn, C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework, Mathematics of Computation 52(186) (1989) 411–435.

[19] B. Cockburn, C.-W. Shu, The Runge-Kutta local projection P1-discontinuous Galerkin method for scalar conservation laws, Mathematical Modelling and Numerical Analysis 25 (3) (1991) 337–361.

[20] B. Cockburn, C.-W. Shu, The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems, Journal of Computational Physics 141 (2) (1998) 199–224. doi:10.1006/jcph.1998.5892.

[21] B. Cockburn, C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (3) (2001) 173–261. doi:10.1023/A:1012873910884.

[22] B. Cockburn, G. E. Karniadakis, C. W. Shu, Discontinuous Galerkin Methods. Theory, computation and applications, Lecture Notes in Computational Science and Engineering, 11, Springer-Verlag, Berlin, 2000.

[23] S. Godunov, Finite difference methods for the computation of discontinuous solutions of the equations of fluid dynamics, Math. Sbornik 47 (1959) 271–306.

[24] P. L. Roe, Approximate Riemann solvers, parameters vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357–372. doi:10.1016/0021-9991(81)90128-5.

[25] E. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics: a Practical Introduction, Springer-Verlag, Berlin, 1997.

[26] J. Butcher, Numerical methods for ordinary differential equations in the 20th century, Journal of Computational and Applied Mathematics 125 (1-2) (2000) 1–29. doi:10.1016/S0377-0427(00)00455-6.

[27] G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: Proceedings of the April 18-20, 1967, spring joint computer conference, AFIPS '67 (Spring), ACM, New York, NY, USA, 1967, pp. 483–485. doi:10.1145/1465482.1465560.

[28] J. F. Kelly, F. X. Giraldo, Continuous and discontinuous Galerkin methods for a scalable three-dimensional nonhydrostatic atmospheric model: Limited-area mode, Journal of Computational Physics 231 (24) (2012) 7988 – 8008. doi:10.1016/j.jcp.2012.04.042.

[29] B. Hendrickson, R. Leland, A multilevel algorithm for partitioning graphs, in: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM), Supercomputing '95, ACM, New York, NY, USA, 1995. doi:10.1145/224170.224228.

[30] G. Karypis, V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, Journal of Parallel and Distributed Computing 48 (1) (1998) 96–129. doi:10.1006/jpdc.1997.1404.

[31] G. Karypis, V. Kumar, A fast and highly quality multilevel scheme for partitioning irregular graphs, SIAM Journal of Scientific Computing 20 (1) (1998) 359–392. doi:10.1137/S1064827595287997.

[32] G. Karypis, V. Kumar, Multilevel algorithms for multi-constraint graph partitioning, in: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM), Supercomputing '98, IEEE Computer Society, Washington, DC, USA, 1998, pp. 1–13.

[33] R. Comblen, J. Lambrechts, J.-F. Remacle, V. Legat, Practical evaluation of five partly discontinuous finite element pairs for the non-conservative shallow water equations, International Journal for Numerical Methods in Fluids 63 (6) (2010) 701–724. doi:10.1002/fld.2094.

[34] J. Lambrechts, E. Hanert, E. Deleersnijder, P.-E. Bernard, V. Legat, J.-F. Remacle, E. Wolanski, A multi-scale model of the hydrodynamics of the whole Great Barrier Reef, Estuarine, Coastal and Shelf Science 79 (1) (2008) 143–151. doi:10.1016/j.ecss.2008.03.016.

[35] B. de Brye, A. de Brauwere, O. Gourgue, T. Kärnä, J. Lambrechts, R. Comblen, E. Deleersnijder, A finite-element, multi-scale model of the Scheldt tributaries, river, estuary and ROFI, Coastal Engineering 57 (9) (2010) 850 – 863. doi:10.1016/j.coastaleng.2010.04.001.

[36] O. Gourgue, R. Comblen, J. Lambrechts, T. Kärnä, V. Legat, E. Deleersnijder, A flux-limiting wetting-drying method for finite-element shallow-water models, with application to the Scheldt Estuary, Advances in Water Resources 32 (12) (2009) 1726–1739. doi:10.1016/j.advwatres.2009.09.005.

[37] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-D finite element mesh generator with buit-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331. doi:10.1002/nme.2579.

[38] E. Marchandise, J. Remacle, N. Chevaugeon, A quadrature-free discontinuous Galerkin method for the level set equation, Journal of Computational Physics 212 (1) (2006) 338–357. doi:10.1016/j.jcp.2005.07.006.

[39] J. Pedlosky, Geophysical Fluid Dynamics, second edition, Springer-Verlag, Heidelberg, 1987.

[40] J. Lambrechts, R. Comblen, V. Legat, C. Geuzaine, J.-F. Remacle, Multiscale mesh generation on the sphere, Ocean Dynamics 58 (5-6) (2008) 461–473. doi:10.1007/s10236-008-0148-3.

[41] K. Slaoui, J. Lambrechts, S. Blaise, V. Legat, J.-F. Remacle, Stereographic coordinates for efficient high-order finite element method on the sphere, to appear.

[42] W. Graf, M. Altinakar, Fluvial hydraulics, Wiley Chichester, UK, 1998.

[43] S. W. Rienstra, W. Eversman, A numerical comparison between the multiple-scales and finite-element solution for sound propagation in lined flow ducts, Journal of Fluid Mechanics 437 (2001) 367–384. doi:10.1017/S0022112001004438.

[44] C. Bogey, C. Bailly, D. Juvé, Computation of flow noise using source terms in linearized Euler's equations, AIAA Journal 40 (2) (2002) 235–243. doi:10.2514/2.1665.