High order time integration and mesh adaptation with error control for incompressible Navier-Stokes and scalar transport resolution on dual grids

Marc-Arthur N'Guessan, Marc Massot, Laurent Series, Christian Tenaud

High order time integration and mesh adaptation with error control for incompressible Navier-Stokes and scalar transport resolution on dual grids

Marc-Arthur N'Guessan[a,*], Marc Massot[a], Laurent Séries[a], Christian Tenaud[b]

[a] *CMAP, École polytechnique, Route de Saclay, 91128 Palaiseau Cedex, France*
[b] *LIMSI, CNRS, Université Paris-Saclay, Bâtiment 508, Rue John Von Neumann, 91403 ORSAY Cedex, France*

Abstract

Relying on a building block developed by the authors in order to resolve the incompressible Navier-Stokes equation with high order implicit time stepping and dynamic mesh adaptation based on multiresolution analysis with collocated variables, the present contribution investigates the ability to extend such a strategy for scalar transport at relatively large Schmidt numbers using a finer level of refinement compared to the resolution of the hydrodynamic variables, while preserving space adaptation with error control. This building block is a key part of a strategy to construct a low-Mach number code based on a splitting strategy for combustion applications, where several spatial scales are into play. The computational efficiency and accuracy of the proposed strategy is assessed on a well-chosen three-vortex simulation.

*Corresponding author

Email addresses: marc-arthur.nguessan@polytechnique.edu (Marc-Arthur N'Guessan), marc.massot@polytechnique.edu (Marc Massot), laurent.series@polytechnique.edu (Laurent Séries), Christian.Tenaud@limsi.fr (Christian Tenaud)

Keywords: Incompressible Navier-Stokes, high order implicit Runge Kutta, multiresolution analysis, dynamic mesh adaptation, scalar transport, dual grid with error control
2000 MSC: 65M08, 65M50, 76D05, 80A25, 80A32

1. Introduction

Numerical simulations of chemically reacting flows place a considerable strain on computational resources, because of the large spectrum of characteristic spatial and temporal scales involved in these phenomena. Furthermore, the Direct Numerical Simulation (DNS) of low-Mach combustion requires important computational resources, partly due to the highly refined meshes necessary to accurately describe the reactive fronts, but also due to the various numerical schemes, which involve costly linear algebra for Poisson solvers or implicit schemes [1]. One way to reduce the computational effort that has been investigated over the years is the use of Adaptive Mesh Refinement (AMR) [2] to spatially adapt the grid in the reactive fronts, thus reducing the number of unknowns [3, 4, 5, 6].

Whereas such techniques have led to very interesting developments, one of the difficulties of AMR is the heuristics used in order to refine the mesh, which lead to a high compression level but hardly provide any error estimate. Our contribution also focuses on spatial mesh adaptation in order to reduce the memory of such simulation, but rather involves multiresolution (MR) analysis in order to obtain high compression [7, 8], with error control in space and time when coupled to an adaptive splitting technique [9]. In [8], Cohen *et al.* developed the algorithms to efficiently perform adaptive

MR for systems of conservative laws in a finite volume context. There were implemented for examples in [10] for the numerical simulations of premixed and diffusion flames, where the advection-diffusion-reaction problem was spatially discretized on a Cartesian grid with MR adaptation, whereas the flow field was provided analytically at each time step; a similar strategy was investigated in [11] for another application. Using the proposed splitting strategy implies that the missing building block was a solver for the hydrodynamics. We started with the incompressible Navier-Stokes equations and the use of multiresolution and finite volume on adapted grids and tree-data structures made the classical approach on staggered grids [12] or the resolution of the resulting differential algebraic equation (DAE) by a fractional-step method [13, 14, 15] rather impossible or low order. In [16] we developed a high-order time integration based on the Radau IIA Runge-Kutta method, and a finite-volume method coupled to adaptive multiresolution to solve incompressible flows on collocated grids.

One of the particularities of low-Mach combustion is the fact that the flow and the transported species involve different spatial and temporal scales. The ones describing the flow require more computational effort because they involve the numerical resolution of linear systems [12, 17, 18, 14, 15, 19]. But the characteristic spatial scales of the flow are often larger than the scales of the species, and one may exploit this fact by resolving these two sets of equations on different grids. This idea was used in [6], where a full low-Mach combustion solver was designed, with the flow being solved on a uniform coarse grid, while the advection-reaction-diffusion of scalars were solved on a finer grid using AMR. See also [20, 21] for an application of this technique for

phase-field simulations. Our aim is to design a numerical strategy along the same lines, which adapts the mesh at different levels for the hydrodynamics variables and for the species equations with finer discretization, while sticking to error control based on MR on this dual grid.

Low-Mach number combustion is our goal, but in order to introduce the fundamentals of the approach, we rather focus on a simpler problem representative of the difficulties we will encounter and tackle the problem of a scalar transport at various Schmidt numbers by a flow field, which is a solution of the incompressible Navier-Stokes equation, where we introduce a numerical strategy relying on a dual grid for both fields with error control based on MR. To this end, we design a $2D$ configuration inspired by the canonical interaction of vortex pair with the mixing layer of a passive reactant [4]. The strategy is assessed in this academic configuration in terms of accuracy and efficiency and we show that the proposed strategy allows to obtain large gains in terms of computational cost and memory trace, without tempering on the accuracy of the global solution even at relatively large Schmidt numbers.

The outline of the paper is the following. In section 2 the governing equations for an incompressible flow and the advection-diffusion of a passive scalar are presented, as well as the details of the mixing layer and vortex interaction. Then, in section 3 we expose our numerical strategy, namely the adaptive multiresolution algorithm, our spatial discretization strategy for both the flow and the scalar and finally the temporal discretization retained here. We assess the efficiency of this strategy to properly tackle the transported scalar problem at hand in section 4, and finally conclusions are drawn in section 5.

2. Governing equations

We consider the transport of a passive scalar $s(x, y, t)$ in a rectangular domain denoted Ω, of characteristic length L_0, by an incompressible fluid flow which is fully described by two variables, the velocity vector $\boldsymbol{u} = (u_i(x, y, z, t))_{i=1,2}$ and the pressure field $p(x, y, z, t)$. The time variable t varies between 0 and T. The flow momentum and mass balance equations read:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}^t \otimes \boldsymbol{u}) + \nabla p - \nu \, \Delta \boldsymbol{u} = \boldsymbol{f} \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases} \tag{1}$$

and are coupled to a transport equation for the scalar s given by:

$$\frac{\partial s}{\partial t} + \nabla \cdot (s\boldsymbol{u}) - \kappa \, \Delta s = \mathrm{f} \tag{2}$$

where ν is the kinematic viscosity of the fluid, κ is the diffusivity of the scalar, and $\boldsymbol{f}$ and f are source terms. Initial and boundary conditions are added to the system of equations and will be presented in the numerical results section.

In order to mimic the situation in combustion applications where the characteristic length related to the flame inner layer is lower than the scales that have to be resolved in order to properly capture the velocity field [3, 6], we will consider several Péclet number in our study. The Péclet number is the product of the Reynolds number and of the Schmidt number. Given a vortical structure of size L, with a vorticity amplitude of φ, the Péclet number reads $\varphi \, L^2/\kappa$ and represents the ratio between the convection and diffusion processes; it can be interpreted as a ratio of eddy turn over time compared to the diffusion time over the size of the vortex. In order to resolve

cases, where the Péclet number is relatively large, adapting the grid for both hydrodynamic field and scalar field at the same time will lead to much too fine a mesh for the hydrodynamic solver in terms of memory trace and then computational cost for a given accuracy. The idea is then to use a dual grid and couple MR on this dual grid in order to save memory and time, while preserving error control.

3. Numerical strategy

We want to tackle problems where the flow and transported scalars have an inhomogeneous spatial distributions, with localized scalar fronts moving across the domain. We exploit this specificity by dynamically adapting the computational grid thanks to the multiresolution analysis [8, 7], in order to refine the mesh in regions where steep gradients occur and coarsen it elsewhere. By studying the physical characteristics of the problem at hand, we start with a uniform mesh refined enough to capture the smallest length scales. We then apply MR algorithms, at each timestep, to restrain the use of the finest meshes only where the variables present steep gradients, while employing coarser meshes elsewhere. Two attributes of the multiresolution analysis make it perfectly suited for adaptive grid refinement in numerical resolution of PDEs [7]:

1. the adaptation process is based on local regularity indicators of the variable we are approximating. It is thus inherently more accurate than ad-hoc criteria used in AMR
2. we have the ability to monitor the error of the multiresolution process. This adaptation strategy comes with error-tracking capabilities allow-

ing us to control the information loss that data compression necessarily entails

We do not need the same mesh resolution for the flow and the scalar transport; it suggests the use of a *multihierarchy* spatial discretization strategy [6, 20, 21]. We will use one grid for the velocity and pressure fields, another one for the transported scalars, and will perform adaptive MR separately but in a consistent manner on *both* grids. This means that at any timestep we have an hybrid grid for the flow, that is hopefully coarser than the scalars' hybrid grid. Here we only consider the transport of a passive scalar, so we only need to project the velocity field on the scalars' grid when needed: this is done with the inter-level operations of the MR procedure (the prediction and projection operators, which are described in the next section). This will introduce some errors compared to a computation of both the flow and the scalar performed in a single grid, especially regarding the divergence-free constraint on the velocity, nevertheless this error is controlled thanks to the MR algorithm. We will denote by (MMR) this new multihierarchy multiresolution adapative strategy.

We discretize our PDEs in space via a finite volume method. The classical approach for finite volume schemes to approximate incompressible flows, is to use staggered grids for the velocity and pressure, to avoid spurious velocity and pressure modes [12]. However, the inter-level operations between embedded grids necessary for the MR algorithms on the one-hand, and the non-uniform character of the adaptive computational grid on the other hand, hinder the use of the staggered-grid layout in our case. We resort then to a *collocated* arrangement [22], and we have to deal with the spurious modes.

Moreover, our adaptive grid is *non-conforming* ([22, 23, 24], see figure 1). This problem was also taken into account in the finite volume scheme that we designed to couple multiresolution to an incompressible flow solver. This scheme is easily usable to discretize the scalar transport, too.

The spatial discretization yields a Differential Algebraic Equation (DAE) for the velocity and pressure variables, and an Ordinary Differential Equation (ODE) for the scalar, since we only consider here the advection-diffusion of the scalar by the flow. Since the grid can change at each timestep, we prefer to integrate these equations in time by one-step methods. The integration of the DAE needs special care due to its stiffness [25, 26, 27]; in addition the satisfaction of the divergence-free constraint cannot be achieved with an explicit method. We chose the fully implicit two stages Radau IIA method [25]. It is *stiffly accurate* [28], and does not suffer from order reduction when applied to DAEs [25]. It is 3^{rd} order for the velocity, and 2^{nd} order for the pressure. To integrate the scalar we use the classical explicit four stages Runge-Kutta method (RK4). We use the velocity at time n to advance the scalar at time $n+1$, and then advance the velocity.

In what follows, we will first give a short presentation of the adaptive multiresolution algorithms used in a finite volume context; more details can be found in [8, 7, 29]. We then describe briefly our spatial and temporal schemes.

3.1. Adaptive multiresolution strategy

We consider a variable u defined on a computational domain $\Omega =]0, b_x[\times]0, b_y[$, with $(b_x, b_y) \in \mathbb{R}_+^*$. We choose the maximum grid level $l^{max} \in \mathbb{N}^*$ so that the computational mesh of size $2^{-l^{max}} b_x \times 2^{-l^{max}} b_y$ is fine enough to properly

capture all the spatial scales of u in Ω. Let Ω_l be a set of nested dyadic Cartesian grids, indexed by their *refinement level* $l = 0, 1, 2, \cdots, l^{max}$ so that for each l we have:

$$\Omega_l = \{]2^{-l}b_x i, 2^{-l}b_x(i+1)[\times]2^{-l}b_y j, 2^{-l}b_y(j+1)[\quad | \quad (i,j) \in \{0,1,\cdots,2^l-1\} \times \{0,1,\cdots,2^l-1\}\}$$

We define:

$$K^l_{i,j} =]2^{-l}b_x i, 2^{-l}b_x(i+1)[\times]2^{-l}b_y j, 2^{-l}b_y(j+1)[$$
$$S_l = \{0,1,\cdots,2^l-1\} \times \{0,1,\cdots,2^l-1\}$$

The following then holds:

- Ω_l is the disjoint union of cells $K^l_{i,j}$, with $(i,j) \in S_l$ where S_l is the index set of the meshes of Ω_l
- $\overline{\Omega_l} = \overline{\bigsqcup_{(i,j)\in S_l} K^l_{(i,j)}} = \overline{\Omega}$
- if $l < l^{max}$, for any cell $K^l_{i,j} \in \Omega_l$, there exists a unique set of 4 cells K^{l+1}_μ with $\mu \in S_{l+1}$ so that $\overline{K^l_{i,j}}$ is the union of the cells $\overline{K^{l+1}_\mu}$: the cells $K^{l+1}_{2i,2j}$, $K^{l+1}_{2i+1,2j}$, $K^{l+1}_{2i,2j+1}$ and $K^{l+1}_{2i+1,2j+1}$. We denote this set $\mathcal{C}^l_{i,j}$

There is a *natural* tree structure associated with such a set of embedded dyadic grids [8]. The root of the tree is the coarsest cell K^0, and for any cell K^l_γ with $l < l^{max}$, we say that the cells $K^{l+1}_\mu \in \mathcal{C}^l_\gamma$ are the *children* of K^l_γ, and (reciprocally) that K^l_γ is the *parent* of the cells in $\mathcal{C}^l_\gamma$. *Leaves* of the tree are cells with no child. By definition, the initial set of leaves is formed

by the cells at the most refined grid level l^{max}. Here we have *quadtrees* in $2D$. Given $\varepsilon \in \mathbb{R}_+^*$, we build a multiresolution representation of u with the following steps:

1. Initialization

We start by computing a discrete representation $U_{l^{max}} = (u_\gamma)_{\gamma \in S^{l^{max}}}$ of u on $\Omega_{l^{max}}$, where each u_γ is the average of u over the mesh $K_\gamma^{l^{max}}$.

2. Projection

For $l \in \{l^{max}-1, l^{max}-2, \cdots, 1, 0\}$, we derive the approximation U_l on the grid Ω_l by *a projection* [29] of the finer approximation on Ω_{l+1}. For each $\gamma \in S^l$, we have: $u_\gamma = \frac{1}{4}\sum_{\mu \in \mathcal{C}_\gamma^l} u_\mu$, *i.e.* u_γ is the average of the 4 values of the children meshes of K_γ^l.

3. Details computation

For each $K_{i,j}^l$, with $l \in \{0, 1, \cdots, l^{max}-1\}$ and $(i,j) \in S^l$, we derive a local regularity indicator of the variable u. For any vector of values $V = (v_k)$, where the k belongs to a finite set of indexes, let Q^s be a polynomial interpolation defined as:

$$Q^s(k, V) = \sum_{q=1}^{s} \xi_q (v_{k+q} - v_{k-q})$$

with $s \in \mathbb{N}$, and the ξ_q are the coefficients of centered linear polynomial interpolations of order $2s+1$ [30]. For each child $K_{2i+p,2j+q}^{l+1}$ (with $(p,q) \in \{0,1\} \times \{0,1\}$ depending on the child), we compute an approximate value (a.k.a. *a prediction*) $\hat{u}_{2i+p,2j+q}^{l+1}$ of $u_{2i+p,2j+q}^{l+1}$ by polynomial interpolation of the values on the grid Ω_l [29]:

$$\hat{u}^{l+1}_{2i+p,2j+q} = u^l_{i,j} + (-1)^p Q^s(i, u^l_{.,j}) + (-1)^q Q^s(i, u^l_{i,.}) + (-1)^{(p+q)} Q^s_2(i, j; U^l) \quad (3)$$

where Q^s_2 reads:

$$Q^s_2(i, j; U^l) = \sum_{a=1}^{s} \xi_a \sum_{b=1}^{s} \xi_b (u^l_{i+a,j+b} - u^l_{i-a,j+b} - u^l_{i+a,j-b} + u^l_{i-a,j-b})$$

The local regularity indicator (a.k.a. *the detail*) is then defined as: $d^l_{i,j} = \sqrt{\sum_{\mu \in \mathcal{C}^l_{i,j}} (u^{l+1}_\mu - \hat{u}^{l+1}_\mu)^2}$

4. Thresholding

For each l from $l^{max} - 1$ down to 0, we associate a flag `keep-children` to every mesh $K^l_{i,j}$, that is initially set to `false`. Then if $\frac{d^l_{i,j}}{max(d^l_{i,j})} \geq 2^{l-l^{max}} \varepsilon$ we set the flag to `true`, otherwise we keep it to `false`. The maximum is taken over the set of all details of meshes belonging to the tree.

5. Grading

For each $K^l_{i,j}$, with $l \in \{0, 1, \cdots, l^{max} - 1\}$ and $(i, j) \in S^l$, we denote by $R^l_{i,j}$ the indexes of the nodes needed for the computation 3. Then for l from $l^{max} - 1$ down to 0, if the flag `keep-children` of $K^l_{i,j}$ is set to `true`, for each cell K^l_γ with $\gamma \in R^l_{i,j}$, we set the `keep-children` of its parent to `true`.

6. Pruning

For each $K^l_{i,j}$, with $l \in \{0, 1, \cdots, l^{max} - 1\}$ and $(i, j) \in S^l$, if its flag `keep-children` is set to `false`, then we *discard* its children from the tree structure. Let Λ be the set of indexes (l, γ), so that the cell K^l_γ belongs to the

tree structure, let $\mathcal{M}$ be its set of leaves and $L(\Lambda)$ the indexes corresponding to these leaves . Λ and $\mathcal{M}$ evolve after the preceding pruning procedure, in such a way that the cells belonging to $\mathcal{M}$ are still a disjoint partition of Ω. If we denote: $\mathcal{M}_\Lambda U = (u_\gamma^l)_{(l,\gamma)\in L(\Lambda)}$, then $\mathcal{M}_\Lambda U$ is a *multiresolution approximation* of u, a new discrete and hybrid (because the leaves in $\mathcal{M}$ may not have the same size anymore) representation of this variable on the computational domain Ω.

The multiresolution analysis [7] ensures that there exist a constant C independent of ε so that: $\|U_{l^{max}} - \mathcal{M}_\Lambda U\| \leq C\varepsilon$, and so we have a control of the precision of our hybrid approximation with regard to the most refined uniform representation (see for example [31]). We proceed with some remarks about this adaptation strategy:

1. If the interpolation stencil s in step 3 is so that $s \geq 1$, then the Grading step ensures that the level of two adjacent cells in $\mathcal{M}$ can differ by at most one unit (if K_γ^l and $K_\mu^{l'}$ are adjacent cells, then $l' \in \{l-1, l, l+1\}$). In this study, except stated otherwise, s will be set to 1 (figure 1 gives an example of a graded mesh discretization in this case)
2. We can combine steps 2 to 6 with a PDE numerical solver S to perform dynamic grid adaptation in the following way. Suppose that we start with an initial condition on u discretized over the most refined uniform grid. We apply the preceding adaptation strategy and obtain new sets Λ^0 and $\mathcal{M}^0$, and a multiresolution approximation $\mathcal{M}_\Lambda U^0$, that we will simply denote U^0. We then apply S to U^0 to obtain $\tilde{U}^1$ on $\mathcal{M}^0$, that we use to compute new projection values (step 2) and new details (step

3) for the nodes in Λ^0 that are not leaves. We modify step 4, and add another procedure: suppose that mesh K_γ^l is a leaf in Λ^0, that $l < l^{max}$, and that d_μ^{l-1} is the detail of its parent cell (computed from $\tilde{U}^1$). If $\frac{d_\mu^{l-1}}{max(d_\mu^{l-1})} \geq 2^{4s+4} 2^{l-1-l^{max}} \varepsilon$, then we reconstruct the children of K_γ^l, we compute new values in these cells from K_γ^l and its neighbors, using the interpolation of step 3, and we set the `keep-children` flag of K_γ^l to `true`. From then we apply steps 5 and 6 to obtain new sets Λ^1 and $\mathcal{M}^1$, and a new vector U^1. We re-apply S and steps 2 to 6 to obtain U^2, and so on

3. We can adapt multiple variables $u_1, u_2, \cdots u_m$ on the same grid: we perform steps 1 to 4 for each variable separately, and we set the flag of a cell to `true` if it is set to `true` for at least one of the variables. We then perform steps 5 and 6, and the grid obtained will be accurate enough for all the variables. We can also discretize two variables u_1 and u_2 on two completely separate grids, but on the same domain Ω. If we need values from u_2 to perform a computation on the grid of u_1 for example, we can always use the projection and prediction operators (steps 2 and 3) to perform this values transfer between grids

3.2. Spatial discretization

$\mathcal{M}$ is the adaptive mesh that partitions the computational domain. For every rectangular mesh $K \in \mathcal{M}$, we denote by x_K the center of K, $m(K)$ its (Lebesgue) measure, $\mathcal{N}_K$ its set of neighbours, $\mathcal{E}_K = \{\sigma_{K|L} \mid \sigma_{K|L}$ the edge separating meshes K and L for $L \in \mathcal{N}_K\}$ its set of edge boundaries, $d_{K,\sigma}$ the Euclidian distance between x_K and σ, and $m(\sigma)$ the measure of

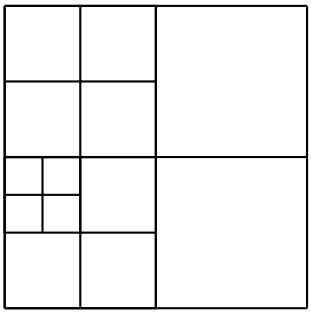


Figure 1: Example of a graded quadtree discretization

σ, for $\sigma \in \mathcal{E}_K$. A discretized variable V on $\mathcal{M}$ is represented by its vector components $V = (v_K)_{K\in\mathcal{M}}$. We also denote l_K the level to which the mesh K belongs to in the grid hierarchy (cf. section 3.1).

The quantities that we are trying to approximate are the velocity $\boldsymbol{u} = (u_i(\boldsymbol{x}, t))_{i=1,2}$, the pressure $p(\boldsymbol{x}, t)$ and the transported scalar $v(\boldsymbol{x}, t)$. We start with the finite volume scheme to solve the PDE (1) that results in a Differential Algebraic Equation (D) for the quantities:

$$\boldsymbol{U} = (u_{i,K}(t))_{i=1,2,K\in\mathcal{M}}$$

$$P = (p_K(t))_{K\in\mathcal{M}}$$

The DAE (D) is found by approximating the differential spatial operators that appear in (1).

We follow [23] in designing our finite volume scheme, and discretize (1) for every mesh K and component $i = 1, 2$ in the following way:

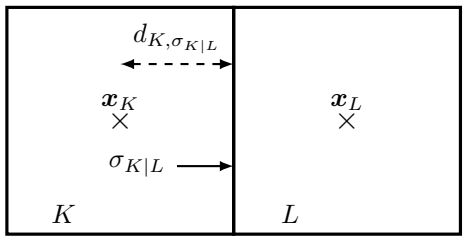
$$\begin{cases} m(K)\frac{du_{i,K}}{dt} + \underbrace{\nu \sum_{\sigma \in \mathcal{E}_K} F_{K,\sigma}(u_i)}_{\text{Diffusion}} + \underbrace{m(K)\partial_K^{(i)} P}_{\text{Pressure gradient}} + \underbrace{C_K^{(i)}(\boldsymbol{U})}_{\text{Convection}} = \int_K f_i(\boldsymbol{x}) d\boldsymbol{x} = m(K) F_{i,K} \\ m(K) div_K \boldsymbol{U} = \sum_{L \in \mathcal{N}_K} \Phi_{K|L}(\boldsymbol{U}) = 0 \end{cases} \tag{4}$$

$\nu \sum_{\sigma \in \mathcal{E}_K} F_{K,\sigma}(u_i)$ are approximations of the diffusive fluxes of the quantity u_i through the set of boundaries $\mathcal{E}_K$ of K. $\sum_{L \in \mathcal{N}_K} \Phi_{K|L}(\boldsymbol{U})$ are an approximation of the mass fluxes through the boundaries of K, with $\mathcal{N}_K$ its set of neighbours. Given the type of mesh we have to deal with (cf. section 3.1), we need to distinguish between 3 cases for the fluxes computation: the case where the meshes K and L are at the same level ($l_K = l_L$), the case where L has level $l_L = l_K + 1$, and finally the case where L has level $l_L = l_K - 1$. The three cases are explicited in (figure 2) below.

The different equations (4) amount to a nonlinear system that can be written in matrix form (for example $\int_K \nabla \cdot \boldsymbol{u} \approx D \cdot \boldsymbol{U}$ where D is a *divergence* matrix). We formally define the *gradient* matrix as $-D^t$, that is, the discrete gradient is the *dual* operator of the discrete divergence [22, 23, 32]. This way, we make sure that our discretization mimics this property of the continuous PDE. In addition, we do not then have to specify boundary conditions for the pressure, which can be a tricky operation [33]. Finally, we define the convective term: $C_K^{(i)}(\boldsymbol{U}) = m(K) div_K(\boldsymbol{U} \otimes \boldsymbol{U})^{(i)}$.

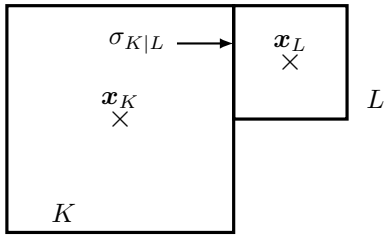
The scheme written here is complete for periodic boundary conditions. For Neumann or Dirichlet boundary conditions however, we need a special treament for the discretization of the diffusion and mass fluxes of the velocity

Case $l_L = l_K$



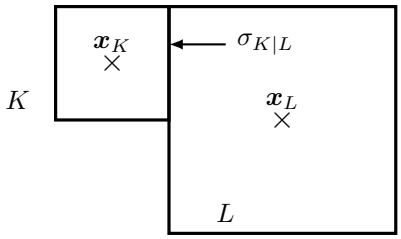
$$F_{k,\sigma_{K|L}}(u_i) = m(\sigma_{K|L})\frac{u_{i,K} - u_{i,L}}{d_{K,\sigma_{K|L}} + d_{L,\sigma_{K|L}}};$$
$$\Phi_{K|L}(\boldsymbol{U}) = m(\sigma_{K|L})\frac{u_{1,K} + u_{1,L}}{2}$$

Case $l_L = l_K + 1$



$$F_{k,\sigma_{K|L}}(u_i) = m(\sigma_{K|L})\frac{u_{i,K} - u_{i,L}}{\frac{d_{K,\sigma_{K|L}}}{2} + d_{L,\sigma_{K|L}}};$$
$$\Phi_{K|L}(\boldsymbol{U}) = m(\sigma_{K|L})\frac{u_{1,K} + u_{1,L}}{2}$$

Case $l_L = l_K - 1$



$$F_{k,\sigma_{K|L}}(u_i) = m(\sigma_{K|L})\frac{u_{i,K} - u_{i,L}}{d_{K,\sigma_{K|L}} + \frac{d_{L,\sigma_{K|L}}}{2}};$$
$$\Phi_{K|L}(\boldsymbol{U}) = m(\sigma_{K|L})\frac{u_{1,K} + u_{1,L}}{2}$$

Figure 2: Computation of the fluxes depending on the interface case

near the boundary of the domain. We will not get into the details of this implementation here, because the main goal of this article is not to describe the collocated spatial discretization. It is quite classical for collocated meshes in the litterature, and will be presented in a subsequent paper in preparation. We discretize (2) for every mesh K using the approximation fluxes of (4):

$$m(K)\frac{dS_K}{dt} + \underbrace{\kappa \sum_{\sigma \in \mathcal{E}_K} F_{K,\sigma}(s)}_{\text{Diffusion}} + \underbrace{m(K) div_K(\boldsymbol{U}S)}_{\text{advection}} = \int_K \mathrm{f}(\boldsymbol{x}) d\boldsymbol{x} = m(K)\mathrm{F}_K \tag{5}$$

3.3. Temporal discretization

We now have to solve the following *Hessenberg index 2* DAE in the time parameter for the velocity and the pressure variables:

$$\begin{cases} \Gamma \dfrac{dU_i}{dt} = \nu L_i U_i + D_i^t P - \ (\sum_{j=1,2} D_j U_i U_j) + \Gamma S_i(t) \\ \sum_{i=1,2} D_i U_i = S_{div}(t) \end{cases} \tag{6}$$

and the following ODE in the time parameter for the transported scalar:

$$\Gamma^{sc} \frac{dS}{dt} = \kappa L^{sc} S - \ (\sum_{j=1,2} D_j S U_j) + \Gamma^{sc} S_{sc}(t) \tag{7}$$

Here Γ, L_i and D_i are square matrices of size $card(\mathcal{M}) \times card(\mathcal{M})$. Γ is the mass matrix, the diagonal matrix so that $\Gamma_{i(K),i(K)} = m(K)$ for the mesh $K \in \mathcal{M}$. The L_i operators are the Laplacian matrices for the diffusive terms, and the D_i are the divergence matrices described above. The vectors $S_i(t)$ include the discretized source terms F_i and the eventual boundary conditions, and the vector $S_{div}(t)$ include the boundary conditions for the divergence

constraint.

Since the flow and the scalar may not be solved on the same grid, we denote $\mathcal{M}^{sc}$ the grid for the scalar. Γ^{sc}, L^{sc} and D_i^{sc} are square matrices of size $card(\mathcal{M}^{sc}) \times card(\mathcal{M}^{sc})$. Γ^{sc} is the mass matrix, the diagonal matrix so that $\Gamma^{sc}_{i(K),i(K)} = m(K)$ for the mesh $K \in \mathcal{M}^{sc}$. The L^{sc} operator is the Laplacian matrix for the diffusive terms, and the D_i^{sc} are the divergence matrices described above. The vectors $S_{sc}(t)$ include the discretized source term F and the eventual boundary conditions.

We start with the resolution of (6) and we solve it with the Runge-Kutta **two-stage Radau IIA** method, which is 3^{rd} order for the velocity and 2^{nd} order for the pressure [25, 26]. We derive it here with the ε-embedding method [25] in the following way. Given velocities and pressure fields $(\boldsymbol{U^0}, P^0)$ at time t_0, we want to obtain an approximate solution of (6) $(\boldsymbol{U^1}, P^1)$, at time $t_0 + h = t_1$ with an implicit 2-stage Runge-Kutta method. The ε-embedding method recasts equation (6) in the following form:

$$
\begin{aligned}
\Gamma g_i^1 = \quad & \Gamma U_i^0 + h a_{11}\big(\nu L_i g_i^1 + D_i^t\ k^1 - (\sum_{j=1,2} D_j g_i^1 g_j^1) + \Gamma S_i(t_0 + c_1\ h)\big) \\
& + h a_{12}\big(\nu L_i g_i^2 + D_i^t\ k^2 - (\sum_{j=1,2} D_j g_i^2 g_j^2) + \Gamma S_i(t_0 + c_2\ h)\big) \\
\Gamma g_i^2 = \quad & \Gamma U_i^0 + h a_{21}\big(\nu L_i g_i^1 + D_i^t\ k^1 - (\sum_{j=1,2} D_j g_i^1 g_j^1) + \Gamma S_i(t_0 + c_1\ h)\big) \\
& + h a_{22}\big(\nu L_i g_i^2 + D_i^t\ k^2 - (\sum_{j=1,2} D_j g_i^2 g_j^2) + \Gamma S_i(t_0 + c_2\ h)\big)
\end{aligned}
$$

$$
\begin{aligned}
\sum_{i=1,2} D_i g_i^1 = \quad & S_{div}(t_0 + c_1 h) \\
\sum_{i=1,2} D_i g_i^2 = \quad & S_{div}(t_0 + c_2 h)
\end{aligned}
$$

$$
\begin{aligned}
\Gamma U_i^1 = \quad & \Gamma U_i^0 + hb_1\big(\nu L_i g_i^1 + D_i^t\ k^1 - (\sum_{j=1,2} D_j g_i^1 g_j^1) + \Gamma S_i(t_0 + c_1\ h)\big) \\
& + hb_2\big(\nu L_i g_i^2 + D_i^t\ k^2 - (\sum_{j=1,2} D_j g_i^2 g_j^2) + \Gamma S_i(t_0 + c_2\ h)\big) \\
P^1 = \quad & (1 - h\sum_{i,j=1}^{2} b_i\omega_{ij})P^0 + \sum_{i,j\ =1}^{2} b_i\omega_{ij}k^j
\end{aligned} \tag{8}
$$

h is the timestep, g_i^j, k^j are intermediate variables, and $W = (\omega_{ij})_{1\leq i,j\leq 2}$ is the inverse of the matrix $A = (a_{ij})_{1\leq i,j\leq 2}$ corresponding to the Radau IIA method (see table 1). Since this method is stiffly accurate, we actually have $U_i^1 = g_i^2$; and since it is L-stable, we actually have $(1 - h\sum_{i,j=1}^{2} b_i\omega_{ij}) = 0$ [25], which simplify the computations. The presence of the convective terms in (1) implies that (8) is a nonlinear system in the variables (g_i^j, k^j), that has been solved here with simple fixed-point Picard iterations [34].

We now treat the spurious pressure and velocity modes. The spurious modes are due to the linear part of equation (1) [17], so that we only have to consider the Stokes equation for their treatment. If we do not take into account the convective terms in (8), the velocity and pressure at time t^{n+1}

$\frac{1}{3}$	$\frac{5}{12}$	$\frac{-1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{4}$	$\frac{1}{4}$

Table 1: Butcher array of the Radau IIA scheme

are obtained by inverting the matrix $E = \begin{pmatrix} B & M \\ N & 0 \end{pmatrix}$, where:

- B is block-diagonal matrix consisting of d (the space dimension) diagonal blocks B_i, with $B_i = \begin{pmatrix} \Gamma - a_{11}h\nu L_i & -a_{12}h\nu L_i \\ -a_{21}h\nu L_i & \Gamma - a_{22}h\nu L_i \end{pmatrix}$
- $M = \begin{pmatrix} M_1 \\ M_2 \end{pmatrix}$, with $M_i = \begin{pmatrix} -a_{11}hD_i^t & -a_{12}hD_i^t \\ -a_{21}hD_i^t & -a_{22}hD_i^t \end{pmatrix}$
- $N = \begin{pmatrix} N_1 & N_2 \end{pmatrix}$, with $N_i = \begin{pmatrix} D_i & 0 \\ 0 & D_i \end{pmatrix}$

The spurious modes that will affect the solution of our problem are the vectors of the form $\begin{pmatrix} \boldsymbol{v} \\ q \end{pmatrix}$ which are in the null-space of E: $E \begin{pmatrix} \boldsymbol{v} \\ q \end{pmatrix} = \begin{pmatrix} B & M \\ N & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{v} \\ q \end{pmatrix} = 0$. If $\boldsymbol{v} \neq 0$, it will pollute the quantities g_i^1, g_i^2 in (8), causing the apparition of spurious modes in the computed solution; and of course the same is true with q regarding k^1, k^2. But as we wrote earlier, we only need the velocity for the scalar, not the pressure, so we are just going to ensure that there are no spurious modes for the velocity. There might be spurious mode in the pressure, but if we make sure that they do not affect the velocity, we are ensured of the precision of our velocity computation.
We remark here that by construction, $-\nu L_i$ is a symmetric and diagonally dominant matrix, so that it is diagonalizable (in $\mathbb{R}$) and all its eigenvalues are positive. The matrix B depends on h the timestep, and we use the following result:

Theorem 1. *There exists $h_0 > 0$ so that for each $0 < h \leq h_0$, if $\begin{pmatrix} \boldsymbol{v} \\ q \end{pmatrix}$ is a null vector of the matrix E, then $\boldsymbol{v} = 0$.*

We will use the following lemma:

Lemma 2. *If a matrix B is as defined above, then there exists $h_0 > 0$ so that for each $0 < h \leq h_0$ and for any real vector $\boldsymbol{z}$ the two following propositions are equivalent:*

(a) $\boldsymbol{z}^t B \boldsymbol{z} = 0$

(b) $\boldsymbol{z} = 0$

Proof. $(b) \implies (a)$ is obvious for any $h \in \mathbb{R}_+^*$. We then turn to $(a) \implies (b)$. For $h \to 0$, B converges to a diagonal matrix where each block is the mass matrix. Hence, in the limit, all its eigenvalues are positive. By the continuous dependency of the eigenvalues on the matrix coefficients, there is some $h_0 > 0$ such that all eigenvalues of B, for $h \leq h_0$, are positive too. In that case $\boldsymbol{z}^t B \boldsymbol{z} = 0$ implies that $\boldsymbol{z} = 0$, which concludes the proof of the lemma.

Proof. We have that $B\boldsymbol{v} + Mq = 0$. If we multiply this vector by $\boldsymbol{v}^t$, we have $\boldsymbol{v}^t B \boldsymbol{v} + \boldsymbol{v}^t M q = 0$. Or $\boldsymbol{v}^t M q = q^t (M^t \boldsymbol{v})$, and if we write $\boldsymbol{v} = \begin{pmatrix} \boldsymbol{v}^1 \\ \boldsymbol{v}^2 \end{pmatrix}$, we have that $M^t \boldsymbol{v} = \begin{pmatrix} -a_{11} h (\sum_{i=1}^2 D_i \boldsymbol{v}^i) - a_{21} h (\sum_{i=1}^2 D_i \boldsymbol{v}^i) \\ -a_{12} h (\sum_{i=1}^2 D_i \boldsymbol{v}^i) - a_{22} h (\sum_{i=1}^2 D_i \boldsymbol{v}^i) \end{pmatrix} = 0$, because $N\boldsymbol{v} = 0$. Hence $\boldsymbol{v}^t B \boldsymbol{v} = 0$.

We see that if we choose a real h_0 that satisfies the conditions of the lemma, then necessarily, $\boldsymbol{v} = 0$, which concludes the proof of the theorem.

We see that we can always choose h small enough to preclude the apparition of spurious modes for the velocity field, the one that is detrimental for the transport of scalars.
The resolution of the ODE (7) is done with the RK4 method, which is 4^{th} order for the scalar. We discretize the time interval $[0, T]$ with fixed timesteps of size h, the timestep used to solve the flow. RK4 being an explicit method, we have to take into account stability restrictions when solving the scalar equation, that do not apply to the Radau IIA method. In practice we thus integrate (7) with a timestep h^{sc} smaller than h. We use the velocity at time nh to integrate the scalar from nh to $(n+1)h$ with $\lfloor \frac{h}{h^{sc}} \rfloor$ iterations of RK4, and then use one iteration of (8) to advance the flow variables from nh to $(n+1)h$.

4. Numerical experiments

4.1. Initial configuration

The initial configuration of the flow consists of three vortices $(\omega_1, \omega_2, \omega_3)$ distributed on the horizontal axis with their centers located respectively at the center of the domain $(0, 0)$, and at $(-x_0, 0)$ and $(x_0, 0)$. They have the following shapes:

$$\omega_1 = -\frac{1}{2}\varphi\left(1 + \tanh\left(\frac{1}{\delta_0}(r_0 - \sqrt{x^2+y^2})\right)\right)$$
$$\omega_2 = +\frac{1}{2}\frac{\varphi}{2}\left(1 + \tanh\left(\frac{1}{\delta_0}(r_0 - \sqrt{(x-x_0)^2+y^2})\right)\right)$$
$$\omega_3 = +\frac{1}{2}\frac{\varphi}{2}\left(1 + \tanh\left(\frac{1}{\delta_0}(r_0 - \sqrt{(x+x_0)^2+y^2})\right)\right)$$

The initial condition for the scalar is given by

$$s = \tanh\left(\frac{1}{\delta_1}y\right)$$

It aims at mimicking an initial configuration where the scalar is a constant value in the upper half of the domain, and another one in the lower half of the domain (we can think for example of a gas mixture with fresh fuel in the upper half, and hot air in the lower half). The source terms of (1) and (2) are set to 0. Figure (3) shows the initial vorticity and scalar field.

The computational domain Ω is the square $[-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$, the boundary conditions for the velocity are free-flow boundary conditions, and we impose homogeneous Neumann boundary conditions for the transported scalar. We set $\nu = 5.10^{-4}$, $\varphi = 100$, $\delta_0 = 0.01$, $r_0 = 0.03$, $x_0 = 0.09$, and $\delta_1 = 0.01$.
We turn to an analysis of the relevant spatial scales of this configuration. The characteristic scales of the flow are the core size of each vortex $2r_0$ and the distance between the vortices center x_0. The characteristic scale for the scalar is the thickness of the mixing layer δ_1. It can happen that δ_1 is an order of magnitude smaller than x_0 and $2r_0$, which means that the scalar needs a finer mesh than the flow to be accurately solved numerically. The dimensionless parameter of interest here is the Peclet number, $Pe = \frac{4\varphi r_0^2}{\kappa}$,

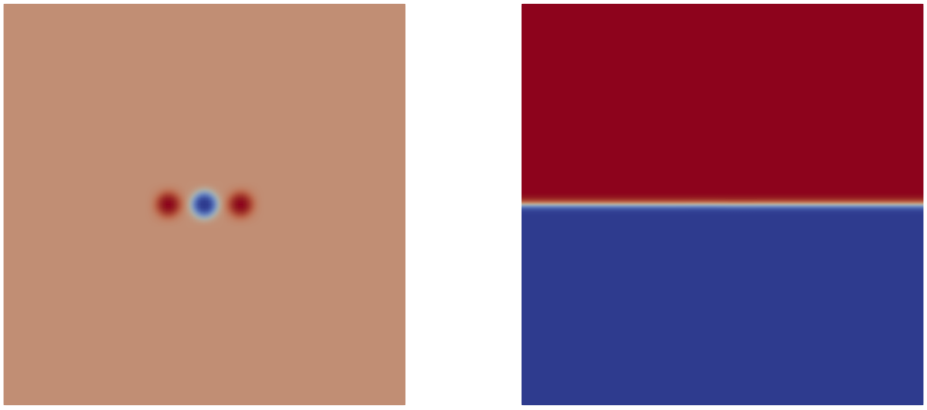


Figure 3: Initial values of the numerical experiment. Left: vorticity field; right: passive scalar

which represents the ratio between advective and diffusive processes in the scalar transport [17]. If Pe is small then the diffusive forces are prevalent, and they tend to quickly thicken δ_1. In this case a grid resolving the velocity may be good enough to solve the scalar transport. But if Pe is large, then the advective forces are prevalent, and the initial small mixing layer is advected in a spiral way by the flow, almost without thickening due to diffusion. In that case the grid must capture the small initial structures of the scalar.
We rewrite the Peclet number $Pe = ReSc$, where $Re = \frac{4\varphi r_0^2}{\nu}$ is the Reynolds number of the flow, and $Sc = \frac{\nu}{\kappa}$ is the Schmidt number. We fix $Sc = 0.1$ to investigate the error control capability of our method, and we will later consider larger Schmidt numbers. With such parameters, a uniform grid with $l^{max} = 7$ is fine enough to properly describe the flow, but in some situation this might be too coarse for the scalar. In each of the following computations,

we fix $h = 2.5 \times 10^{-3}$ for the timestep for the flow, and $h^{sc} = 1.25 \times 10^{-4}$ for the scalar. We verified that, for each of the following computations, these timesteps were small enough to ensure convergence in time; also, the use of high order temporal integrators ensures the fact that the temporal errors are negligible compared to the spatial errors. We will denote by (MR) a classical multiresolution adaptive strategy where the flow and the scalar are computed on the same grid. Thus the initial uniform grid has to be refined enough to capture the spatial scales of both the scalar and the flow, and the adaptive process produces a grid adapted for both of them. We recall that we denote by (MMR) the new multihierarchy multiresolution adapative strategy that we developed here, with separate grids for the flow and the scalar. For the (MMR) strategy, we denote l_v^{max} the maximum level for the flow grid, and l_{sc}^{max} the maximum level for the scalar.

4.2. Numerical results

First we compare the result of our numerical strategy with a computation run on a uniform grid. For the (MMR) computation, we set $l_v^{max} = 7$, $l_{sc}^{max} = 8$ for the scalar, we choose $\varepsilon = 10^{-3}$ for thresholding parameter, and run the computation from 0 to 0.5. The uniform grid is set at $l^{max} = 8$ for both the flow and the scalar. The results for three times, $t = 0.0075$, $t = 0.025$ and $t = 0.5$, are shown on figure (4), where for each time we have the adapted grid for the scalar on the left, the scalar value for the adapted grid on the center, and the scalar value for the uniform grid computation on the right. We see here a good agreement suggesting that our adaptive strategy is able to produce the same results as on a uniform fine grid. We

also show the evolution of the vorticity field on figure (5).

Next we check the ability to control the multiresolution error with our new adaptive strategy. We recall that we adapt both the flow and the scalar, but on different grid. Here, we set $l_v^{max} = 7$, and $l_{sc}^{max} = 7$ or $l_{sc}^{max} = 8$. In each case, we run the simulation until $t = 0.5$, and at this time we compute the L^2-norm of the error between the adapted scalar value, and the value obtained with the same two-grids algorithm, but with both the flow and the scalar computed on their respective uniform most refined grid. We do this for different values of ε. Figure (6) shows that we still preserve the ability to control the adaptation error with ε, the error decreasing linearly with the threshold parameter.

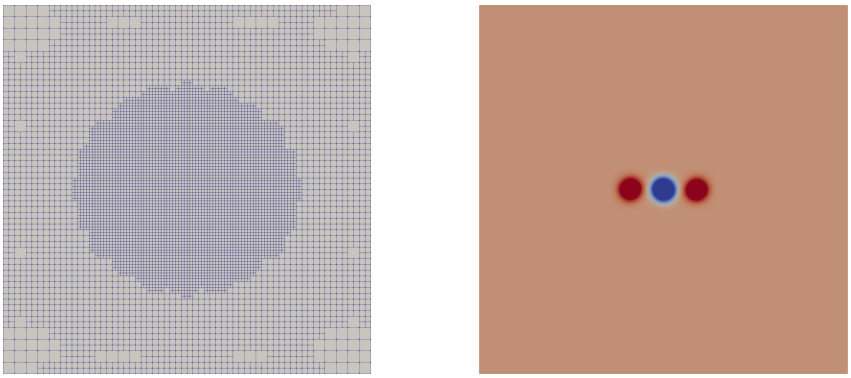
Our last numerical test deals with variable Péclet numbers. As we stated earlier, the flow and the scalar might need different resolutions to be adequately solved, and in our case this is mainly related to the relative importance of the advection versus the scalar diffusion. We set $\varepsilon = 10^{-3}$, and we run the computation until $t = 0.5$, with different Schmidt numbers. We study the values for $Sc = 0.1$, $Sc = 1$ and $Sc = 10$. For each Schmidt number, we run three computations: one with a uniform grid at $l^{max} = 7$ for the flow and $l^{max} = 10$ for the scalar (our reference for comparisons), one with (MR) where $l^{max} = 7$, and one (MMR) with $l_v^{max} = 7$ and $l_{sc}^{max} = 9$. The results are shown in figure (7), where we zoom in closer to the mixing layer. We see that at $Sc = 0.1$ and $Sc = 1$, the (MR) and (MMR) are in good agreement with the reference simulation. But for $Sc = 10$, the (MR) solution is not accurate, and we actually have to resort to a grid with $l_{sc}^{max} = 9$ for the scalar to properly describe the phenomenon. The computation would

(a) $t = 0.0075$

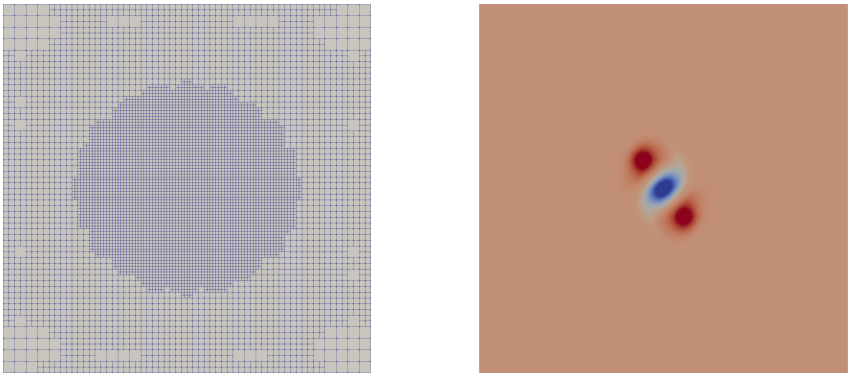
(b) $t = 0.25$

(c) $t = 0.5$

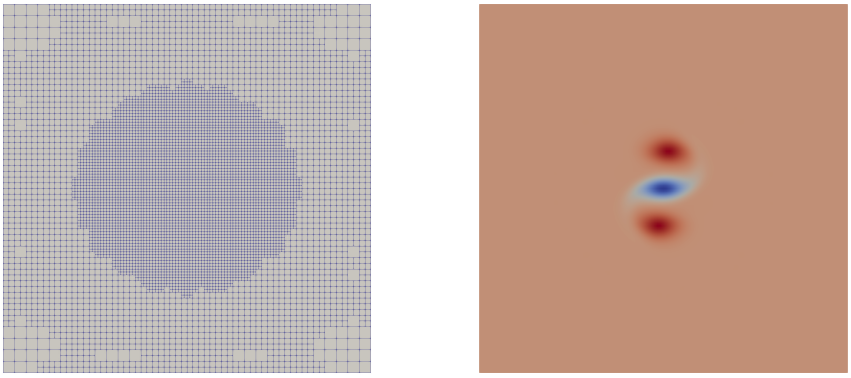
Figure 4: Interaction of a passive scalar and a vortex field. Comparison between a computation on a uniform grid at $l^{max} = 8$, and a computation done with (MMR), where $l_v^{max} = 7$ and $l_{sc}^{max} = 8$. Left: adapted grid for the scalar; center: scalar value on the adapted grid; right: scalar value on the uniform grid



(a) $t = 0.0075$



(b) $t = 0.25$



(c) $t = 0.5$

Figure 5: Evolution of the vorticity field with grid adaptation by multiresolution, where $l_v^{max} = 7$. Left: adapted grid; right: vorticity field

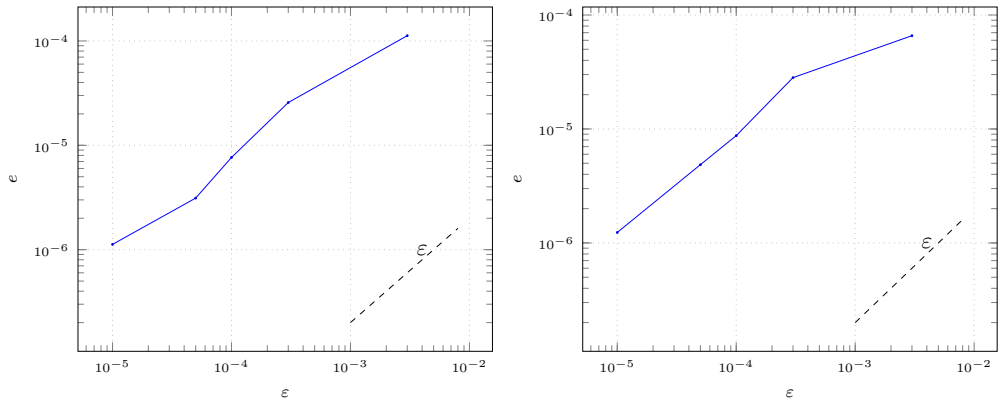


Figure 6: L^2 norm of the multiresolution error versus the threshold parameter ε for the scalar. Left: $l_{sc}^{max} = 7$; right: $l_{sc}^{max} = 8$

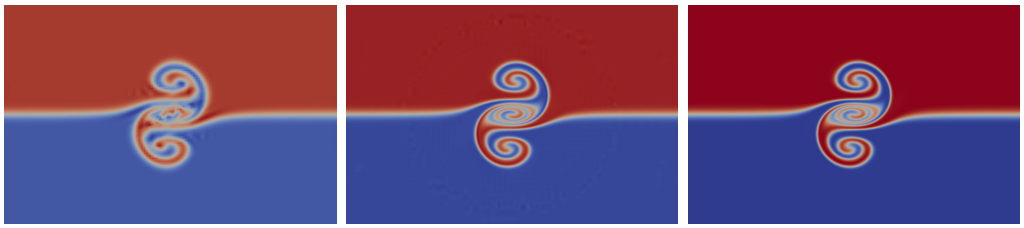
take much longer if we had to resort to a unique grid, even if we use the adaptive multiresolution for the flow and the scalar: in our case the velocity grid has around 7000 cells, while the scalar grid as around 63000 cells, so this would mean solving a linear system for the flow almost ten times higher, even though the physics of the flow does not require such accuracy. This case clearly advocates the usefulness of our method.

ε	3.10^{-2}	3.10^{-3}	3.10^{-4}
Speedup	$\times 8.3$	$\times 8.04$	$\times 8.13$

Table 2: Interaction of a passive scalar and a vortex field. Comparison between the (MR) and (MMR) computing times for different thresholding parameters. (MR) with $l^{max} = 8$; (MMR) with $l_v^{max} = 7$ and $l_{sc}^{max} = 8$. For a given value ε of the thresholding parameter, the adapted grid computation with (MR) is *speedup* times longer than the computation with (MMR)

(a) $Sc = 0.1$

(b) $Sc = 1$



(c) $Sc = 10$

Figure 7: Interaction of a passive scalar and a vortex field. Comparison between uniform, (MR) and (MMR) solutions at $t = 0.5$ for different Schmidt numbers. From left to right: (MR) with $l^{max} = 7$; (MMR) with $l_v^{max} = 7$ and $l_{sc}^{max} = 9$; uniform with $l_v^{max} = 7$ and $l_{sc}^{max} = 10$, reference

ε	3.10^{-2}	3.10^{-3}	3.10^{-4}
Speedup	$\times 71.32$	$\times 38.40$	$\times 16.5$

Table 3: Interaction of a passive scalar and a vortex field. Comparison between the uniform grid and (MMR) computing times for different thresholding parameters. Uniform grid with $l^{max} = 8$; (MMR) with $l_v^{max} = 7$ and $l_{sc}^{max} = 8$. For a given value ε of the thresholding parameter, the uniform grid computation is *speedup* times longer than the computation with (MMR)

4.3. Cost comparison

Finally, we want to discuss the computational advantages of the (MMR) method. Before doing so, we mention that the computing costs are determined in huge part by the software implementation. The code used here is a research/developement software, which primary purpose was not to obtain high-performance computing, but rather to showcase the ability of the (MMR) method to obtain correct and precise simulation results. The three computational settings, the uniform grid, the (MR) and the (MMR) all share the same routines for the linear space operations (matrix-vector multiplication, vector-vector addition, vector-vector inner product, linear solver): we use the optimized PETSc libraries [35, 36, 37] which are written in C. But the data structures and routines needed for the adaptive multiresolution implementation in the (MR) and (MMR) simulations are written in Python, for sequential computations only.

We will proceed to two different comparisons. The first one will help us to determine the computational gain between the (MR) and the (MMR) methods. While both methods use grid adaptation by multiresolution, the single

grid algorithm has to be set at the refinement requirement of the scalar transport, which is not the case for the two grids algorithm. Clearly, the (MMR) method will generate much less meshes than the (MR) one for the flow, and we want to quantify the resulting speedup.

We set $Sc = 0.1$, for both the (MR) and (MMR) methods we set $l_{sc}^{max} = 8$, and for the (MMR) method we also fix $l_v^{max} = 7$. We run the computation from 0 to 0.5, for various values of the thresholding parameters. We save the computing time for the (MR) and (MMR) computations, and we report the ratio between the (MR) computing time and the (MMR) computing time in table (2). The (MMR) method is 8 times faster than the (MR) method here, which is a very good speedup given the precision that we are still able to obtain.

The second comparison concerns the (MMR) method and the uniform grid computation. These kind of comparisons are tricky, because we are dealing with two different softwares. A common mistake is to use the code for the adapted grid to do a computation on a uniform grid; that is not the case here, because we built a specific code for the uniform grid computation, that do not suffer at all from the overhead of the multiresolution. What is more, this code runs entirely on PETSc, so that is fairly optimized compared to the (MMR) code that relies heavily on Python. Nevertheless, we can take full advantage of working on a coarser grid for the flow, and we quantify the resulting speedup.

We set $Sc = 0.1$, and for the (MMR) method we set $l_{sc}^{max} = 8$, and $l_v^{max} = 7$. For the uniform grid computation we set $l^{max} = 8$, and we run the computation from 0 to 0.5, for various values of the thresholding parameters. We save

the computing time for the uniform grid algorithm once, the different computing times for the (MMR) computations, and we report the ratio between the uniform grid computing time and the (MMR) computing time in table (3). The speedup obtained with the (MMR) method, even with a threshold parameter as small as 3.10^{-4}, is already impressive, and we expect it to be even greater with an optimized implementation of the adaptive multiresolution.

We finish this section by recalling that these results were obtained with an in-house code, on relatively small test cases. But we expect them to be even better for the (MMR) on a more optimized code for two reasons: (i) the multiresolution algorithms are linear in time, while the matrix inversion necessary in incompressible flow computations are polynomial in time, with a degree greater than 2 generally, thus the overhead of the multiresolution algorithms is negligible when compared to the grid size reduction, and (ii) with the (MMR) we can be at an even higher level of refinement for the scalar.

5. Conclusion

We introduce a new spatial adaptive strategy to efficiently solve the transport of a passive scalar by an incompressible flow. The flow and the scalar are discretized in a finite-volume context on different grids, and multiresolution adaptive refinement techniques are applied to both variables. Regarding the incompressible Navier-Stokes equations, the velocity and pressure unknwons are discretized on a collocated setting, and the DAE resulting from the spatial discretization is solved with an implicit high-order Runge-Kutta method,

allowing third-order accuracy in time for the velocity. The ODE resulting from the spatial discretization of the scalar is solved with an explicit fourth order Runge-Kutta method. This new strategy is particularly suited for configurations where the characteristic scales of the scalar are much smaller than the characteristic scales of the flow, as is often the case in chemically reacting flows. We can achieve an accurate description for the scalar, without paying the price of solving the flow on a grid with a much larger number of unknowns that would be necessary for the description of the latter. We review this on a manufactured case of 3 vortices interacting with a passive scalar mixing layer.

The next step will consist in combining these techniques with high-order operator splitting techniques for an efficient and accurate resolution of chemically reacting flows.

Acknowledgments

This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH.

[1] B. A. V. Bennett, M. D. Smooke, Local rectangular refinement with application to nonreacting and reacting fluid flow problems, Journal of Computational Physics 151 (2) (1999) 684–727.

[2] M. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, Journal of Computational Physics 53 (3) (1984) 484–512.

[3] M. Day, J. Bell, Numerical simulation of laminar reacting flows with

complex chemistry, Combustion Theory and Modelling 4 (4) (2000) 535–556.

[4] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, M. L. Welcome, A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations, Journal of Computational Physics 142 (1) (1998) 1–46.

[5] R. B. Pember, L. H. Howell, J. B. Bell, P. Colella, W. Y. Crutchfield, W. A. Fiveland, J. P. Jessee, An adaptive projection method for unsteady, low-Mach number combustion, Combustion Science and Technology 140 (1-6) (1998) 123–168.

[6] C. Safta, J. Ray, H. N. Najm, A high-order low-Mach number AMR construction for chemically reacting flows, Journal of Computational Physics 229 (24) (2010) 9299–9322.

[7] M. Postel, Approximations multiéchelles, in: École de printemps de mécanique des fluides numérique, Aussois, 2001.

[8] A. Cohen, S. M. Kaber, S. Muller, M. Postel, Fully adaptive multiresolution finite volume schemes for conservation laws, Mathematics of computation 72 (2001) 183–225.

[9] S. Descombes, M. Duarte, T. Dumont, V. Louvet, M. Massot, Adaptive time splitting method for multi-scale evolutionary partial differential equations., Confluentes Mathematici 3 (3) (2011) 413–443.

[10] M. Duarte, S. Descombes, C. Tenaud, S. Candel, M. Massot, Time–space

adaptive numerical methods for the simulation of combustion fronts, Combustion and Flame 160 (6) (2013) 1083–1101.

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

[12] F. Harlow, J. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Physics of Fluids 8 (1965) 2182–2189.

[13] A. Chorin, Numerical solution of the Navier-Stokes equations, Mathematics of Computation 22 (1968) 745–762.

[14] R. Temam, Une méthode d'approximation de la solution des équations de Navier-Stokes, Bulletin de la Société Mathématique de France 96 (1968) 115–152.

[15] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier-Stokes equations, Journal of Computational Physics 59 (2) (1985) 308–323.

[16] M.-A. N'Guessan, L. Séries, C. Tenaud, M. Massot, A high-order Runge-Kutta method coupled to a multiresolution strategy to solve the incompressible Navier-Stokes equations, in: NASA Technical Memorandum, Proceedings of the 2018 Summer Program, NASA Ames Research Center, 2018.

[17] P. Gresho, R. Sani, Incompressible flow and the finite element method,

advection-diffusion and isothermal laminar flow, John Wiley & Sons, 2000.

[18] M. Rhie, W. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, AIAA Journal 21 (1983) 1525–1532.

[19] R. Ranjan, C. Pantano, A collocated method for the incompressible Navier-Stokes equations inspired by the box scheme, J. Comput. Phys. 232 (1) (2013) 346–382.

[20] A. Schmidt, A multi-mesh finite element method for phase-field simulations, in: Interface and Transport Dynamics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 208–217.

[21] A. Schmidt, A multi-mesh finite element method for 3D phase field simulations, in: Free Boundary Problems, Birkhäuser Basel, Basel, 2004, pp. 293–301.

[22] E. Chénier, R. Eymard, O. Touazi, Numerical results using a colocated finite-volume scheme on unstructured grids for incompressible fluid flows, Numerical Heat Transfer, Part B: Fundamentals 49 (3) (2006) 259–276.

[23] E. Chénier, R. Eymard, R. Herbin, A collocated finite volume scheme to solve free convection for general non-conforming grids, J. Comput. Physics 228 (2009) 2296–2311.

[24] R. Eymard, T. Gallouet, R. Herbin, Discretization of heterogeneous and anisotropic diffusion problems on general nonconforming meshes

SUSHI: a scheme using stabilization and hybrid interfaces, IMA Journal of Numerical Analysis 30 (4) (2010) 1009–1043.

[25] E. Hairer, G. Wanner, Solving ordinary differential equations II, Springer series in computational mathematics, Springer-Verlag Berlin Heidelberg, 1996.

[26] B. Sanderse, Energy-conserving Runge-Kutta methods for the incompressible Navier-Stokes equations, J. Comput. Phys. 233 (2013) 100–131.

[27] U. M. Ascher, L. R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, Society for Industrial and Applied Mathematics, 1998.

[28] A. Prothero, A. Robinson, On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations, Mathematics of Computation - Math. Comput. 28 (1974) 145–162.

[29] Tenaud, C., Duarte, M., Tutorials on adaptive multiresolution for mesh refinement applied to fluid dynamics and reactive media problems, ESAIM: Proc. 34 (2011) 184–239.

[30] B. L. Bihari, A. Harten, Multiresolution schemes for the numerical solution of 2D conservation laws I, SIAM Journal on Scientific Computing 18 (2) (1996) 315–354.

[31] M. Duarte, Adaptive numerical methods in time and space for the simulation of multi-scale reaction fronts, Ph.D. thesis, École Centrale Paris, France (2011).

[32] A. Guittet, M. Theillard, F. Gibou, A stable projection method for the incompressible Navier-Stokes equations on arbitrary geometries and adaptive quad/octrees, Journal of Computational Physics 292 (2015) 215–238.

[33] M. Gresho, R. Sani, On pressure boundary conditions for the incompressible navier-stokes equations, International Journal for Numerical Methods in Fluids 7 (1987) 621–659.

[34] S. Turek, Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach, 1998.

[35] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Petsc home page, http://www.mcs.anl.gov/petsc (2008).

[36] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Petsc users manual, Tech. Rep. ANL-95/11 - Revision 3.2, Argonne National Laboratory (2011).

[37] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Efficienct management of parallelism in object oriented numerical software libraries, in: E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhauser Press, 1997, p. 163–202.