Engineering with Computers
https://doi.org/10.1007/s00366-018-0607-5

ORIGINAL ARTICLE

Three-dimensional CFD simulations with large displacement of the geometries using a connectivity-change moving mesh approach

Nicolas Barral[1] · **Frédéric Alauzet**[2]

Received: 8 January 2018 / Accepted: 26 March 2018

Abstract
This paper deals with three-dimensional (3D) numerical simulations involving 3D moving geometries with large displacements on unstructured meshes. Such simulations are of great value to industry, but remain very time-consuming. A robust moving mesh algorithm coupling an elasticity-like mesh deformation solution and mesh optimizations was proposed in previous works, which removes the need for global remeshing when performing large displacements. The optimizations, and in particular generalized edge/face swapping, preserve the initial quality of the mesh throughout the simulation. We propose to integrate an Arbitrary Lagrangian Eulerian compressible flow solver into this process to demonstrate its capabilities in a full CFD computation context. This solver relies on a local enforcement of the discrete geometric conservation law to preserve the order of accuracy of the time integration. The displacement of the geometries is either imposed, or driven by fluid–structure interaction (FSI). In the latter case, the six degrees of freedom approach for rigid bodies is considered. Finally, several 3D imposed-motion and FSI examples are given to validate the proposed approach, both in academic and industrial configurations.

Keywords Moving mesh · Dynamic mesh · Connectivity change · Compressible flows · ALE: Arbitrary Lagrangian Eulerian · Discrete geometric conservation law · Fluid–structure interaction · 6-DOF

1 Introduction

Fluid–structure interaction (FSI) simulations are required for a wide variety of subjects, from the simulation of jellyfish [23] to the releasing of a missile [47]. The recent development of computing capacities has made it possible to run increasingly complex simulations where moving bodies interact with an ambient fluid in an unsteady way. However, engineers are still far from performing such simulations on a daily basis, largely due to the difficulty of handling the moving meshes induced by the moving geometries.

When the displacement of the geometry is small enough, slightly deforming the original mesh [9, 19] can generally be acceptable. But when large deformation of the boundaries is considered, the mesh quickly becomes distorted and the numerical error due to this distortion quickly becomes too great, until the elements of the mesh finally become invalid, and the simulation has to be stopped. Specific strategies need to be developed to deal with large displacement moving boundary problems. In the case of FSI problems, another difficulty arises from the fact that the displacement of the boundaries is by definition an unknown, and the deformation of the mesh cannot be imposed a priori. Addressing the issue of the mesh movement cannot be separated from addressing the issue of the solver that will compute on these moving meshes. Depending on the strategy employed to deal with the movement, specific numerical methods must be designed to take into account the displacement of the mesh. The question this paper focuses on is: how can we efficiently move the mesh for

✉ Nicolas Barral
n.barral@imperial.ac.uk

Frédéric Alauzet
Frederic.Alauzet@inria.fr

[1] Department of Earth Science and Engineering, Imperial College London, London SW7 2AZ, UK

[2] Gamma3 Team, Inria Saclay Ile-de-France, 91 126 Palaiseau, France

Published online: 18 April 2018

large displacement 3D FSI simulations and what numerical schemes need to be associated to such a strategy?

Three main approaches to address the mesh movement problem can be found in the literature. The first approach consists in having a single body-fitted mesh [11, 29], and moving it along with the moving boundaries. The mesh may thus undergo large deformation. The second approach is the Chimera (or overset) method [12], in which each moving body has its own body-fitted sub-mesh, and the sub-meshes move rigidly together with their body and can overlap one another. Finally the embedded boundary approach [14, 39] uses meshes that are not body-fitted at all: the bodies are embedded in a fixed grid, and techniques such as level-sets are used to recover their moving boundaries. All three approaches have their own strengths and weaknesses. This paper is aligned with our previous works on anisotropic mesh adaptation, with the ultimate goal to use moving geometries in our adaptation framework [8]. For this reason, we focus on body-fitted approaches with one single mesh.

The first strategy to handle body-fitted moving meshes is simply to move the mesh for as long as possible, and remesh (i.e., generate a whole new mesh or a part of it) when the mesh quality becomes critical [11, 29]. For each remeshing, the simulation has to be stopped, a new mesh must be generated, and the solution must be transferred to the new mesh. This approach can be efficient, especially when small displacements are considered and very few remeshings are necessary, because the solver and the meshing aspects are decoupled, and between two remeshings the simulation is fully ALE and free from interpolation errors. However, for larger displacements, the number of remeshings increases to prevent invalid elements from appearing, and this can both be costly and result in poor accuracy due to the solution transfer step. Hence a second strategy has been developed [16, 20], based on the use of local remeshing operations, such as vertex insertion, vertex collapse, connectivity changes and vertex displacements, to preserve a good mesh quality throughout the simulation. The advantage of this method is that it maintains an acceptable mesh quality without needing to stop, remesh and resume the simulation. However, it requires fully dynamic mesh data structures that are permanently updated, which can lead to a loss of CPU efficiency, and the numerous mesh modifications can lead to a loss of accuracy.

Our approach tries to overcome to these drawbacks and is described in detail in [1]. It aims at moving meshes with large displacements of the geometry without ever having to remesh. [By remeshing, we mean stopping the simulation, generating a new mesh (the entire mesh or part of it) and interpolating the solution on the new mesh.] A limited set of mesh modifications are used to preserve the mesh quality throughout the simulation: only connectivity changes (edge swaps) and vertex displacement are performed. This is for several reasons. Notably, performing local mesh modifications within the solver is far simpler than remeshing globally, and connectivity changes can be relatively simply interpreted in terms of evanescent cells for purposes of Arbitrary Lagrangian Eulerian (ALE) numerical schemes. In many body-fitted moving mesh strategies, a lot of CPU time is dedicated to computing the displacement of the mesh, in order to make it follow the moving boundaries. Thanks to frequent mesh optimizations, the cost of this step is reduced by computing the mesh deformation for a large number of solver time steps (i.e., we do it only a few times during the simulation). It is important to note that this approach works best if vertex-centered solvers are considered, because the connectivity changes preserve the number of degrees of freedom.

Some studies try to impose a mesh motion that is directly adapted to the physical phenomena in question, using for instance either so-called Moving Mesh PDEs [32] or a Monge–Ampère equation [15]. However, interesting these approaches may be, they still seem to be time-consuming, especially in 3D, due to the solution of a non-linear equation, and it is unsure whether they can handle complex 3D geometries. Therefore we prefer to prescribe arbitrary movements to the mesh, and use our mesh adaptation framework [42] when necessary.

Regarding numerical solvers, we consider a classic framework for moving meshes: the Arbitrary Lagrangian Eulerian (ALE) framework, which is based on a formulation of the equations that takes into account an arbitrary movement of the vertices. This technique was introduced in the 1970s in [21, 31, 33]. Since then, so many developments have been made in that field that a complete list of them would not fit in this paper. However, one may in particular refer to [11, 24, 25, 29, 30, 46, 49], which mainly focus on improving temporal schemes for ALE simulations.

To our knowledge, very few examples of ALE solvers coupled with connectivity-change moving mesh techniques can be found in literature. In [36] a conservative interpolation is proposed to handle the swaps. In [28, 51] an ALE formulation of the swap operator is built. However, these studies are limited to 2D. Driven by the requirements of industry, we are interested in designing a method that works in 3D. In this paper, a linear interpolation is carried out after each swap instead of using a specific ALE formulation, and we will evaluate the numerical error due to these swaps.

The goal of the present paper is to demonstrate that three-dimensional FSI simulations can be run efficiently by coupling an ALE solver to our connectivity-change moving mesh strategy. The first part of this paper focuses on recalling important aspects of the moving mesh algorithm. The second, third and fourth parts describe in detail the solver used. In the fifth part, some validation test cases are

presented, and finally some examples of complex 3D ALE simulations are given and analyzed. In this paper, we only focus on rigid movements that are involved in rigid-body FSI.

2 Mesh-connectivity-change moving mesh strategy

To handle moving boundaries, we adopt a body-fitted approach, with a single mesh: the inner vertices of the mesh are moved following the moving boundaries to preserve the validity of mesh (i.e., to prevent the mesh from getting tangled). Our strategy involves two main parts:

- *Computing the mesh deformation* Inner vertices are assigned a trajectory depending on the displacement of the boundaries, and thus a position for future time steps.
- *Optimizing the mesh* The trajectories computed in the mesh deformation phase are corrected, and the connectivity of the mesh is modified to preserve the quality of the mesh.

This strategy, detailed below, has proven to be very powerful in 3D [1], since large displacement of complex geometries can be performed while preserving a good mesh quality without any global remeshing (i.e., without ever generating a whole new mesh).

2.1 Linear elasticity mesh deformation method

During the mesh deformation step, a displacement field is computed for the whole computational domain, given the displacement of its boundaries. Trajectories can thus be assigned to inner vertices, or in other words, positions at a future solver time step.

Several techniques can be found to compute this displacement field: implicit or direct interpolation [13, 44], or solving PDEs—the most common of which being Laplacian smoothing [40], a spring analogy [19] and a linear elasticity analogy [6]. It is this last method that we selected, due to its robustness in 3D [58]. The computational domain is assimilated to a soft elastic material, which is deformed by the displacement of its boundaries.

The inner vertices movement is obtained by solving an elasticity-like equation with a $\mathbb{P}_1$ finite element method (FEM):

$$\operatorname{div}(\sigma(\mathcal{E})) = 0, \quad \text{with} \quad \mathcal{E} = \frac{\nabla \mathbf{d} + {}^{T}\nabla \mathbf{d}}{2}, \tag{1}$$

where σ and $\mathcal{E}$ are, respectively, the Cauchy stress and strain tensors, and $\mathbf{d}$ is the Lagrangian displacement of the vertices. The Cauchy stress tensor follows Hooke's law for an isotropic homogeneous medium. Dirichlet boundary conditions are used and the displacement of vertices located on the domain boundary is strongly enforced in the linear system. The linear system is solved by a conjugate gradient algorithm coupled with an LU-SGS pre-conditioner. An advantage of elasticity-like methods is the opportunity they offer to adapt the local material properties of the mesh, especially its stiffness, according to the distortion and efforts borne by each element. In particular, the stiffness of the elements is increased for small elements, in order to limit their distortion. More details can be found in [1].

2.2 Improving mesh deformation algorithm efficiency

The computation of the mesh deformation—here the solution of a linear elasticity problem—is known to be an expensive part of dynamic mesh simulations, and the fact that it is usually performed at every solver time step makes it all the more so.

We propose to combine several techniques to improve the time efficiency of this step. Some regions are rigidified, more specifically a few layers around tiny complex details of the moving bodies, with very small elements. They are moved with exactly the same rigid displacement as the corresponding body, thus avoiding very stiff elements in the elasticity matrix. On the other hand, the elasticity can be solved only on a reduced region, if the domain is big compared to the displacement. A coarse mesh can also be used to solve the elasticity problem, the displacement of the vertices then being interpolated on the computational mesh.

The major improvement we proposed is to reduce the number of mesh deformation computations: the elasticity problem is solved for a large time frame of length Δt instead of doing it at each solver time step δt. While there is a risk of a less effective mesh displacement solution, it is a worthwhile strategy if our methodology is able to handle large displacements while preserving the mesh quality. Solving the previously described mesh deformation problem once for large time frame could be problematic in the case of: (1) curved trajectories of the boundary vertices and (2) accelerating bodies. To enhance the mesh deformation prescription, accelerated-velocity curved, i.e., high-order, vertex trajectories are computed.

The paths of inner vertices can be improved if a constant acceleration $\mathbf{a}$ is provided to each vertex in addition to its speed, which results in an accelerated and curved trajectory.

During time frame $[t, t+\Delta t]$, the position and the velocity of a vertex are updated as follows:

$$\mathbf{x}(t+\delta t) = \mathbf{x}(t) + \delta t\, \mathbf{v}(t) + \frac{\delta t^2}{2}\mathbf{a}$$
$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \delta t\, \mathbf{a}.$$

Prescribing a velocity vector and an acceleration vector to each vertex requires solving two elasticity systems. For both systems, the same matrix, thus the same pre-conditioner, is considered. Only boundary conditions change. If inner vertex displacement is sought for time frame $[t, t+\Delta t]$, boundary conditions are imposed by the location of the body at time $t+\Delta t/2$ and $t+\Delta t$. These locations are computed using body velocity and acceleration. Note that solving the second linear system is cheaper than solving the first one as a good prediction of the expected solution can be obtained from the solution of the first linear system. Now, to define the trajectory of each vertex, the velocity and acceleration are deduced from evaluated middle and final positions:

$$\Delta t\, \mathbf{v}(t) = -3\,\mathbf{x}(t) + 4\mathbf{x}(t+\Delta t/2) - \mathbf{x}(t+\Delta t)$$
$$\frac{\Delta t^2}{2}\mathbf{a} = 2\mathbf{x}(t) - 4\mathbf{x}(t+\Delta t/2) + 2\mathbf{x}(t+\Delta t).$$

In this context, it is mandatory to make sure that the mesh remains valid for the whole time frame $[t, t+\Delta t]$, which is done by computing the sign of the volume of the elements all along their path [1].

2.3 Local mesh optimization

In order to preserve the mesh quality between two mesh deformation computations, it has been proposed [1] to couple mesh deformation with local mesh optimization using smoothing and generalized swapping to efficiently achieve large displacement in moving mesh applications. Connectivity changes are really effective in handling shear and removing highly skewed elements. Here, we briefly recall the mesh optimization procedure.

For 3D meshes, the quality of an element is measured in terms of the element shape by the quality function:

$$Q(K) = \frac{\sqrt{3}}{216}\frac{\left(\sum_{i=1}^{6}\ell^2(\mathbf{e}_i)\right)^{\frac{3}{2}}}{|K|} \quad \in [1, +\infty], \tag{2}$$

where $\ell(\mathbf{e})$ and $|K|$ are edge length and element volume. $Q(K) = 1$ corresponds to a perfectly regular element and $Q(K) < 2$ corresponds to excellent quality elements, while a high value of $Q(K)$ indicates a nearly degenerated element.

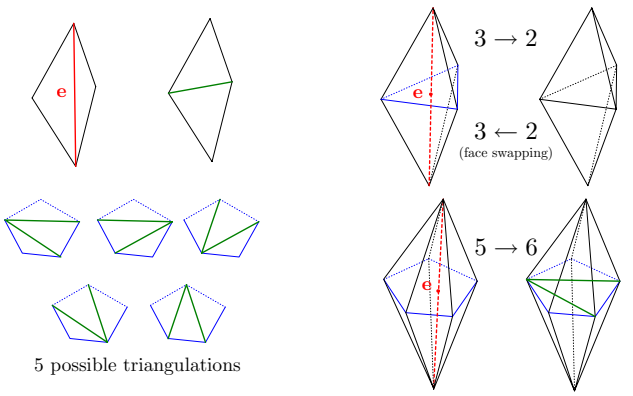

Fig. 1 Top left, the swap operation in two dimensions. Top right, edge swap of type $3 \to 2$ and face swap $2 \to 3$. Bottom left, the five possible triangulations of the pseudo-polygon for a shell having five elements. Bottom right, an example of $5 \to 6$ edge swap. For all these figures, shells are in black, old edges are in red, new edges in green and the pseudo-polygon is in blue. (Color figure online)

The first mesh optimization tool is vertex smoothing which consists in relocating each vertex inside its ball of elements, i.e., the set of elements having P_i as their vertex. For each tetrahedron K_j of the ball of P_i, a new optimal position P_j^{opt} for P_i can be proposed to form a regular tetrahedron:

$$P_j^{opt} = G_j + \sqrt{\frac{2}{3}}\frac{\mathbf{n}_j}{\ell(\mathbf{n}_j)},$$

where F_j is the face of K_j opposite vertex P_i, G_j is the center of gravity of F_j, $\mathbf{n}_j$ is the inward normal to F_j and $\ell(\mathbf{n}_j)$ the length of $\mathbf{n}_j$. The final optimal position P_i^{opt} is computed as a weighted average of all these optimal positions $\{P_j^{opt}\}_{K_j \supset P_i}$, the weight coefficients being the quality of K_j. This way, an element of the ball is all the more dominant if its quality in the original mesh is bad. Finally, the new position is analyzed: if it improves the worst quality of the ball, the vertex is directly moved to its new position.

The second mesh optimization tool to improve mesh quality is generalized swapping/local-reconnection (Fig. 1). Let α and β be the two tetrahedra vertices opposite the common face $P_1P_2P_3$. Face swapping consists of suppressing this face and creating the edge $\mathbf{e} = \alpha\beta$. In this case, the two original tetrahedra are deleted and three new tetrahedra are created. This swap is called $2 \to 3$. The reverse operator can also be defined by deleting three tetrahedra sharing such a common edge $\alpha\beta$ and creating two new tetrahedra sharing face $P_1P_2P_3$. This swap is called $3 \to 2$.

A generalization of this operation exists and acts on shells of tetrahedra [1, 26]. For an internal edge $\mathbf{e} = \alpha\beta$, the shell of $\mathbf{e}$ is the set of tetrahedra having $\mathbf{e}$ as common edge. The different edge swaps are generally denoted $n \to m$, where n

is the size of the shell and m is the number of new tetrahedra. In this work, edge swaps $3 \rightarrow 2$, $4 \rightarrow 4$, $5 \rightarrow 6$, $6 \rightarrow 8$ and $7 \rightarrow 10$ have been implemented. In our algorithm, swaps are only performed if they improve the quality of the mesh.

These operations are well-known in the field of mesh generation [26, 56], but are not necessarily efficient in the context of this work. Notably, performing too many of them results in slow codes, whereas the use of bad quality functions results in poor quality meshes. The interest of the method used in this paper lies how and when optimizations are performed. The mesh optimizations are performed element by element, and only when they are needed. Smoothing is performed for every vertex, provided it increases the quality of the corresponding ball. Swaps are only performed, i.e., when the quality of an element decreases and passes a certain threshold. Tetrahedra are treated in quality order from the worst to the best one. The operation is performed only if it verifies quality criteria on the current position of the mesh and on the final position given by the mesh deformation. A key to performing efficient swaps in the moving mesh context is to allow a slight quality degradation in the future. Details on this optimization step can be found in [1].

2.4 Handling of boundaries

The mesh of the boundaries is moved rigidly, and the vertices are not usually moved on the surface (no displacement in the tangential directions). However, in some cases, such as when a body is moving very close to the bounding box of the domain, it can be useful to move the vertices of the bounding box as well. In this case, we can allow tangential displacement on the boundary. The risk of deforming a curved surface being too great, we only do this for planes aligned with the Cartesian frame. To do so, the displacements along the tangential axes are simply considered as new degrees of freedom. For instance, for a plane (x, y), the displacements along the x-axis and the y-axis are considered as degrees of freedom and are added to the elasticity system. The displacement along the z-axis is still set to 0, and thus is not added to the system.

2.5 Moving mesh algorithm

The overall connectivity-change moving mesh algorithm is described in Algorithm 1, where the different phases described above are put together. When coupled with a flow solver (see Sect. 3), the flow solver is called after the optimization phase. In this algorithm, $\mathcal{H}$ stands for meshes, $\mathcal{S}$ for solutions, Q for quality (see Relation (2)), $\mathbf{d}_{|\partial\Omega_h}$ for the displacement on the boundary, and $\mathbf{v}$ and $\mathbf{a}$ for speed and acceleration. Δt and δt are time steps whose meaning is detailed below.

Algorithm 1 Connectivity-Change Moving Mesh Algorithm with Curved Trajectories

Input: $\mathcal{H}^0$, $\mathcal{S}^0$, Δt^0

While $(t < T^{end})$

1. $\Delta t = \Delta t^0$
2. Solve mesh deformation: compute vertices trajectories
   (a) $\left\{\mathbf{d}^{body}_{|\partial\Omega_h}(t+\Delta t/2)\right\}$ = Compute body vertex displacement from current translation speed $\mathbf{v}^{body}$, rotation speed $\boldsymbol{\theta}^{body}$ and acceleration $\mathbf{a}^{body}$ for $[t, t+\Delta t/2]$
   $\mathbf{d}(t+\Delta t/2)$ = Solve elasticity system $\left(\mathbf{d}^{body}_{|\partial\Omega_h}(t+\Delta t/2), \Delta t/2\right)$
   (b) $\left\{\mathbf{d}^{body}_{|\partial\Omega_h}(t+\Delta t)\right\}$ = Compute body vertex displacement from current translation speed $\mathbf{v}^{body}$, rotation speed $\boldsymbol{\theta}^{body}$ and acceleration $\mathbf{a}^{body}$ for $[t, t+\Delta t]$
   $\mathbf{d}(t+\Delta t)$ = Solve elasticity system $\left(\mathbf{d}^{body}_{|\partial\Omega_h}(t+\Delta t), \Delta t\right)$
   (c) $\{\mathbf{v}, \mathbf{a}\}$ = Deduce inner vertex speed and acceleration from both displacements $\{\mathbf{d}(t+\Delta t/2), \mathbf{d}(t+\Delta t)\}$
   (d) If predicted mesh motion is invalid then $\Delta t = \Delta t/2$ and goto 2.
   Else $T^{els} = t + \Delta t$
3. Moving mesh stage, with mesh optimizations and solver solution
   While $(t < T^{els})$
   (a) δt^{opt} = Get optimization time step $\left(\mathcal{H}^k, \mathbf{v}, \text{CFL}^{geom}\right)$
   (b) δt^{solver} = Get solver time step $\left(\mathcal{H}^k, \mathcal{S}^k, \mathbf{v}, \text{CFL}\right)$
   (c) $\delta t = \min(\delta t^{opt}, \delta t^{solver})$
   (d) If $t > t^{opt}$
      i. $\mathcal{H}^k$ = Swaps optimization $\left(\mathcal{H}^k, Q^{swap}_{target}\right)$
      ii. $\mathbf{v}^{opt}$ = Vertex smoothing $\left(\mathcal{H}^k, Q^{smoothing}_{target}, Q_{max}\right)$
      iii. $t^{opt} = t + \delta t^{opt}$
   (e) $\mathcal{S}^{k+1}$ = Solve Equation of State $\left(\mathcal{H}^k, \mathcal{S}^k, \delta t, \mathbf{v}, \mathbf{v}^{opt}\right)$
   (f) $\mathcal{H}^{k+1}$ = Move the mesh and update vertex speed $\left(\mathcal{H}^k, \delta t, \mathbf{v}, \mathbf{v}^{opt}, \mathbf{a}\right)$
   (g) Check mesh quality. If too distorted: solve new deformation problem or stop.
   (h) $t = t + \delta t$
   EndWhile

EndWhile

In Algorithm 1, three time steps appear: a large one Δt for the mesh deformation computation, a smaller one δt^{opt} corresponding to the steps where the mesh is optimized, and the solver time step δt^{solver}. Δt, is currently set manually at the beginning of the computation. After each mesh deformation solution, the quality of the mesh in the future is analyzed: if the quality is too low, the mesh deformation is problem is solved again with a smaller Δt (Algorithm 1 step 2(d)). Moreover, if the mesh quality degrades, a new mesh deformation solution is computed (Algorithm 1 step 3(g)). δt^{opt} is computed automatically, using the CFL^{geom} parameter as described below. Determining δt^{solver} will be discussed in Sect. 3. If the solver time step δt^{solver} is greater than the optimization time step, then the solver time step is truncated to follow the optimizations. If δt^{solver} is smaller than the optimization time step—which is almost always the case—several iterations

of the flow solver are performed between two optimization steps.

2.6 Moving mesh time steps

A good restriction to be imposed on the mesh movement to limit the apparition of flat or inverted elements is that vertices cannot cross too many elements on a single move between two mesh optimizations. Therefore, a geometric parameter CFL^{geom} is introduced to control the number of stages used to perform the mesh displacement between t and $t + \Delta t$. If CFL^{geom} is greater than one, the mesh is authorized to cross more than one element in a single move. In practice, CFL^{geom} is usually set between 1 and 8. The moving geometric time step is given by:

$$\delta t^{opt} = \mathrm{CFL}^{geom} \max_{P_i} \frac{h(\mathbf{x}_i)}{\mathbf{v}(\mathbf{x}_i)}, \tag{3}$$

where $h(\mathbf{x}_i)$ is the smallest height of all the elements in the ball of vertex P_i. In practice, when coupled with a flow solver, the actual time step is the minimum between the flow solver time step and the geometric one.

3 Arbitrary Lagrangian Eulerian flow solver

An Arbitrary Lagrangian Eulerian (ALE) flow solver has been coupled to the moving mesh process described in Algorithm 1. In this section, we discuss in detail the implemented solver, and all the choices that were made from the numerous possibilities available in the literature.

3.1 Euler equations in the ALE framework

We consider the compressible Euler equations for a Newtonian fluid in their ALE formulation. The ALE formulation allows the equations to take arbitrary motion of the mesh into account. Assuming that the gas is perfect, inviscid and that there is no thermal diffusion, the ALE formulation of the equations is written, for any arbitrary closed volume $C(t)$ of boundary $\partial C(t)$ moved with mesh velocity $\boldsymbol{w}$:

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{C(t)} \mathbf{W}\mathrm{d}\mathbf{x}\right) + \int_{\partial C(t)} (\mathcal{F}(\mathbf{W}) - \mathbf{W} \otimes \boldsymbol{w}) \cdot \mathbf{n}\,\mathrm{d}\mathbf{s} \\
&\quad = \int_{C(t)} \mathbf{F}_{\text{ext}}\,\mathrm{d}\mathbf{x} \\
&\Longleftrightarrow \frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{C(t)} \mathbf{W}\mathrm{d}\mathbf{x}\right) + \int_{\partial C(t)} (\mathbf{F}(\mathbf{W}) - \mathbf{W}(\boldsymbol{w} \cdot \mathbf{n}))\,\mathrm{d}\mathbf{s} \\
&\quad = \int_{C(t)} \mathbf{F}_{\text{ext}}\,\mathrm{d}\mathbf{x},
\end{aligned} \tag{4}$$

where

$$\begin{cases}
\mathbf{W} = (\rho, \rho\mathbf{u}, \rho e)^T \text{ is the conservative variables vector} \\
\mathcal{F}(\mathbf{W}) = \left(\rho\mathbf{u},\ \rho u_x\mathbf{u} + p\mathbf{e}_x,\ \rho u_y\mathbf{u} + p\mathbf{e}_y,\ \rho u_z\mathbf{u} + p\mathbf{e}_z,\ \rho\mathbf{u}h\right) \text{ is the flux tensor} \\
\mathbf{F}(\mathbf{W}) = \mathcal{F}(\mathbf{W}) \cdot \mathbf{n} \\
= \left(\rho\eta,\ \rho u_x\eta + pn_x,\ \rho u_y\eta + pn_y,\ \rho u_z\eta + pn_z,\ \rho e\eta + p\eta\right)^T \\
\mathbf{F}_{\text{ext}} = \left(0,\ \rho\,\mathbf{f}_{\text{ext}},\ \rho\mathbf{u} \cdot \mathbf{f}_{\text{ext}}\right)^T \text{ is the contribution of the external forces,}
\end{cases}$$

and we have noted ρ the density of the fluid, p the pressure, $\mathbf{u} = (u_x, u_y, u_z)$ its Eulerian velocity, $\eta = \mathbf{u} \cdot \mathbf{n}$, $q = \|\mathbf{u}\|$, ε the internal energy per unit mass, $e = 1/2\,q^2 + \varepsilon$ the total energy per unit mass, $h = e + p/\rho$ the enthalpy per unit mass of the flow, $\mathbf{f}_{\text{ext}}$ the resultant of the volumic external forces applied on the particle and $\mathbf{n}$ the outward normal to interface $\partial C(t)$ of $C(t)$.

3.2 Spatial discretization

As regards spatial discretization of the solver, we use an edge-based finite-volume approach, with an HLLC Riemann approximate solver and second-order MUSCL gradient reconstruction. The main difference when translating these schemes from the standard formulation to the ALE formulation is the addition of the mesh velocities in the wave speeds of the Riemann problem.

3.2.1 Edge-based finite volume solver

The domain Ω is discretized by a tetrahedral unstructured mesh $\mathcal{H}$. The vertex-centered finite volume formulation consists in associating a control volume denoted $C_i(t)$ with each vertex P_i of the mesh and at each time t. The dual finite volume cell mesh is built by the rule of medians. The common boundary $\partial C_{ij}(t) = \partial C_i(t) \cap \partial C_j(t)$ between two neighboring cells $C_i(t)$ and $C_j(t)$ is decomposed into several triangular interface facets. The normal flux $\mathbf{F}_{ij}(t)$ along each cell interface is taken to be constant (not in time but in space), just like the solution $\mathbf{W}_{ij}$ on the interface.

Rewriting System (4) for $C(t) = C_i(t)$, we get the following semi-discretization at P_i:

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\left(|C_i(t)|\mathbf{W}_i(t)\right) + \\
&\sum_{P_j \in V_i} |\partial C_{ij}(t)|\,\boldsymbol{\Phi}_{ij}\left(\mathbf{W}_i(t),\ \mathbf{W}_j(t),\ \mathbf{n}_{ij}(t),\ \sigma_{ij}(t)\right) = 0,
\end{aligned} \tag{5}$$

- $\mathbf{W}_i(t)$ is the mean value of state $\mathbf{W}$ in cell C_i at time t
- V_i is the set of all neighboring vertices of P_i, i.e., the mesh vertices connected connected to P_i by an edge

- $\mathbf{n}_{ij}$ is the outward normalized normal (with respect to cell C_i) of cell interface ∂C_{ij}
- $\mathbf{F}_{ij}(t) = \mathcal{F}(\mathbf{W}_{ij}(t)) \cdot \mathbf{n}_{ij}(t)$ is an approximation of the physical flux through $\partial C_{ij}(t)$.
- $\sigma_{ij}(t) = \dfrac{1}{|\partial C_{ij}(t)|} \displaystyle\int_{\partial C_{ij}(t)} \boldsymbol{w}_{ij}(t) \cdot \mathbf{n}_{ij}(t) \mathrm{d}\boldsymbol{s}$ is the normal velocity of cell interface $\partial C_{ij}(t)$
- $\boldsymbol{\Phi}_{ij}\big(\mathbf{W}_i(t), \mathbf{W}_j(t), \mathbf{n}_{ij}(t), \sigma_{ij}(t)\big) \approx \mathbf{F}_{ij}(t) - \mathbf{W}_{ij}(t)\sigma_{ij}(t)$ is the numerical flux function used to approximate the flux at cell interface $\partial C_{ij}(t)$.

The computation of the convective fluxes is performed monodimensionally in the direction normal to each finite volume cell interface. Consequently, the numerical evaluation of the flux function $\boldsymbol{\Phi}_{ij}$ at interface ∂C_{ij} can be achieved by solving, at each time step, a one-dimensional Riemann problem in direction $\mathbf{n}_{ij} = \mathbf{n}$ with initial values $\mathbf{W}_\mathrm{L} = \mathbf{W}_i$ on the left of the interface and $\mathbf{W}_\mathrm{R} = \mathbf{W}_j$ on the right. The normal speed to the interface is temporarily noted σ for clarity.

3.2.2 HLLC numerical flux

The methodology provided by Batten [10] can be extended to the Euler equations in their ALE formulation. The HLLC flux is then described by three waves phase velocities:

$$S_\mathrm{L} = \min(\eta_\mathrm{L} - c_\mathrm{L}, \tilde{\eta} - \tilde{c}) \quad \text{and} \quad S_\mathrm{R} = \max(\eta_\mathrm{R} + c_\mathrm{R}, \tilde{\eta} + \tilde{c})$$
$$S_\mathrm{M} = \frac{\rho_\mathrm{R}\eta_\mathrm{R}(S_\mathrm{R} - \eta_\mathrm{R}) - \rho_\mathrm{L}\eta_\mathrm{L}(S_\mathrm{L} - \eta_\mathrm{L}) + p_\mathrm{L} - p_\mathrm{R}}{\rho_\mathrm{R}(S_\mathrm{R} - \eta_\mathrm{R}) - \rho_\mathrm{L}(S_\mathrm{L} - \eta_\mathrm{L})}$$

and two approximate states:

$$\mathbf{W}_\mathrm{L}^* = \begin{cases} \rho_\mathrm{L}^* = \rho_\mathrm{L}\dfrac{S_\mathrm{L} - \eta_\mathrm{L}}{S_\mathrm{L} - S_\mathrm{M}} \\ p_\mathrm{L}^* = p^* = \rho_\mathrm{L}\big(\eta_\mathrm{L} - S_\mathrm{L}\big)\big(\eta_\mathrm{L} - S_\mathrm{M}\big) + p_\mathrm{L} \\ (\rho\mathbf{u})_\mathrm{L}^* = \dfrac{\big(S_\mathrm{L} - \eta_\mathrm{L}\big)\rho\mathbf{u}_\mathrm{L} + \big(p^* - p_\mathrm{L}\big)\mathbf{n}}{S_\mathrm{L} - S_\mathrm{M}} \\ (\rho e)_\mathrm{L}^* = \dfrac{\big(S_\mathrm{L} - \eta_\mathrm{L}\big)\rho e_\mathrm{L} - p_\mathrm{L}\eta_\mathrm{L} + p^* S_\mathrm{M}}{S_\mathrm{L} - S_\mathrm{M}} \end{cases},$$

$$\mathbf{W}_\mathrm{R}^* = \begin{cases} \rho_\mathrm{R}^* = \rho_\mathrm{R}\dfrac{S_\mathrm{R} - \eta_\mathrm{R}}{S_\mathrm{R} - S_\mathrm{M}} \\ p_\mathrm{R}^* = p^* = \rho_\mathrm{R}\big(\eta_\mathrm{R} - S_\mathrm{R}\big)\big(\eta_\mathrm{R} - S_\mathrm{M}\big) + p_\mathrm{R} \\ (\rho\mathbf{u})_\mathrm{R}^* = \dfrac{\big(S_\mathrm{R} - \eta_\mathrm{R}\big)\rho\mathbf{u}_\mathrm{R} + \big(p^* - p_\mathrm{R}\big)\mathbf{n}}{S_\mathrm{R} - S_\mathrm{M}} \\ (\rho e)_\mathrm{R}^* = \dfrac{\big(S_\mathrm{R} - \eta_\mathrm{R}\big)\rho e_\mathrm{R} - p_\mathrm{R}\eta_\mathrm{R} + p^* S_\mathrm{M}}{S_\mathrm{R} - S_\mathrm{M}} \end{cases},$$

where $\eta = \mathbf{u} \cdot \mathbf{n}$ is the interface normal velocity and $\tilde{.}$ are Roe average variables [52]. The HLLC flux through the interface is finally given by:

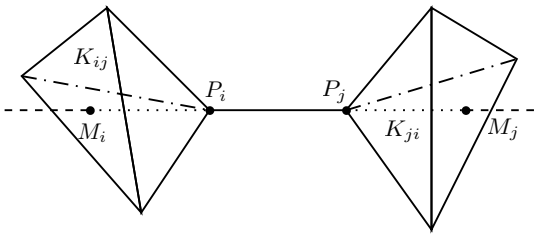

Fig. 2 Downstream K_{ij} and upstream K_{ji} tetrahedra associated with edge $\mathbf{P}_i\mathbf{P}_j$

$$\boldsymbol{\Phi}_\mathrm{HLLC}(\mathbf{W}_\mathrm{L}, \mathbf{W}_\mathrm{R}, \mathbf{n}, \sigma) = \begin{cases} \mathbf{F}_\mathrm{L} - \sigma\mathbf{W}_\mathrm{L} & \text{if } S_\mathrm{L} - \sigma > 0 \\ \mathbf{F}_\mathrm{L}^* - \sigma\mathbf{W}_\mathrm{L}^* & \text{if } S_\mathrm{L} - \sigma \le 0 < S_\mathrm{M} - \sigma \\ \mathbf{F}_\mathrm{R}^* - \sigma\mathbf{W}_\mathrm{R}^* & \text{if } S_\mathrm{M} - \sigma \le 0 \le S_\mathrm{R} - \sigma \\ \mathbf{F}_\mathrm{R} - \sigma\mathbf{W}_\mathrm{R} & \text{if } S_\mathrm{R} - \sigma < 0 \end{cases}.$$

The HLLC approximate Riemann solver has the following properties. It automatically: (1) satisfies the entropy inequality; (2) resolves isolated contacts exactly; (3) resolves isolated shocks exactly and (4) preserves positivity.

3.2.3 High-order scheme

The previous formulation reaches at best a first-order spatial accuracy. A MUSCL type reconstruction method has been designed to increase the order of accuracy of the scheme. The idea is to use extrapolated values $\mathbf{U}_{ij}$ and $\mathbf{U}_{ji}$ of $\mathbf{U}$ at interface ∂C_{ij} to evaluate the flux, where $\mathbf{U} = (\rho, \mathbf{u}, p)$ is the vector of physical variables. The following approximation is performed: $\boldsymbol{\Phi}_{ij} = \boldsymbol{\Phi}(\mathbf{U}_{ij}, \mathbf{U}_{ji}, \mathbf{n}_{ij}, \sigma_{ij})$ with $\mathbf{U}_{ij}$ and $\mathbf{U}_{ji}$ linearly interpolated state values on each side of the interface:

$$\mathbf{U}_{ij} = \mathbf{U}_i + \frac{1}{2}(\nabla\mathbf{U})_{ij}\mathbf{P}_i\mathbf{P}_j, \quad \text{and} \quad \mathbf{U}_{ji} = \mathbf{U}_j + \frac{1}{2}(\nabla\mathbf{U})_{ji}\mathbf{P}_i\mathbf{P}_j. \tag{6}$$

In contrast to the original MUSCL approach, the approximate "slopes" $(\nabla\mathbf{U})_{ij}$ and $(\nabla\mathbf{U})_{ji}$ are defined for each edge using a combination of centered, upwind and nodal gradients.

The centered gradient related to edge $\mathbf{P}_i\mathbf{P}_j$, is defined implicitly along edge $\mathbf{P}_i\mathbf{P}_j$ via relation:

$$(\nabla\mathbf{U})_{ij}^C \mathbf{P}_i\mathbf{P}_j = \mathbf{U}_j - \mathbf{U}_i. \tag{7}$$

Upwind and downwind gradients, which are also related to edge $\mathbf{P}_i\mathbf{P}_j$, are computed using the upstream and downstream tetrahedra associated with this edge. These tetrahedra are, respectively, denoted K_{ij} and K_{ji}. K_{ij} (resp. K_{ji}) is the unique tetrahedron of the ball of P_i (resp. P_j) whose opposite face is crossed by the straight line prolongating edge $\mathbf{P}_i\mathbf{P}_j$; see Fig. 2. Upwind and downwind gradients of edge $\mathbf{P}_i\mathbf{P}_j$ are then defined as:

$$(\nabla \mathbf{U})_{ij}^{U} = (\nabla \mathbf{U})_{|_{K_{ij}}} \quad \text{and} \quad (\nabla \mathbf{U})_{ij}^{D} = (\nabla \mathbf{U})_{|_{K_{ji}}},$$

where

$$\nabla \mathbf{U}_{|_K} = \sum_{P \in K} \left(\nabla \phi_P \otimes \mathbf{U}_P \right)$$

is the $\mathbb{P}^1$-Galerkin gradient on element K and ϕ_P is the basis function associated with P. Parametrized nodal gradients are built by introducing the β-scheme:

$$\begin{aligned} \nabla \mathbf{U}_{ij} \mathbf{P}_i \mathbf{P}_j &= (1-\beta)(\nabla \mathbf{U})_{ij}^{C} \mathbf{P}_i \mathbf{P}_j + \beta (\nabla \mathbf{U})_{ij}^{U} \mathbf{P}_i \mathbf{P}_j, \\ \nabla \mathbf{U}_{ji} \mathbf{P}_i \mathbf{P}_j &= (1-\beta)(\nabla \mathbf{U})_{ij}^{C} \mathbf{P}_i \mathbf{P}_j + \beta (\nabla \mathbf{U})_{ij}^{D} \mathbf{P}_i \mathbf{P}_j, \end{aligned}$$

where $\beta \in [0,1]$ is a parameter controlling the amount of upwinding. For instance, the scheme is centered for $\beta = 0$ and fully upwind for $\beta = 1$.

The most accurate β-scheme is obtained for $\beta = 1/3$, also called the V4-scheme. This scheme is third-order for the two-dimensional linear advection problem on structured triangular meshes. In our case, for the non-linear Euler equations on unstructured meshes, a second-order scheme with a fourth-order numerical dissipation is obtained [18]. The high-order gradients are given by:

$$\begin{aligned} (\nabla \mathbf{U})_{ij}^{V4} \mathbf{P}_i \mathbf{P}_j &= \tfrac{2}{3} (\nabla \mathbf{U})_{ij}^{C} \mathbf{P}_i \mathbf{P}_j + \tfrac{1}{3} (\nabla \mathbf{U})_{ij}^{U} \mathbf{P}_i \mathbf{P}_j, \\ (\nabla \mathbf{U})_{ji}^{V4} \mathbf{P}_i \mathbf{P}_j &= \tfrac{2}{3} (\nabla \mathbf{U})_{ij}^{C} \mathbf{P}_i \mathbf{P}_j + \tfrac{1}{3} (\nabla \mathbf{U})_{ij}^{D} \mathbf{P}_i \mathbf{P}_j. \end{aligned}$$

3.2.4 Limiter

The previous MUSCL schemes are neither monotone nor positive. Therefore, limiting functions must be coupled to the previous high-order gradient evaluations to preserve the monotonicity and positivity of the scheme. To this end, the gradient of Relation (6) is replaced by a limited gradient denoted $(\nabla \mathbf{U}^{\lim})_{ij}$. Here, the three-entry limiter introduced by Dervieux [17], which is a generalization of the SuperBee limiter, will be used :

$$\left(\nabla \mathbf{U}^{\lim}\right)_{ij} \mathbf{P}_i \mathbf{P}_j = L\Big(\left(\nabla \mathbf{U}^{D}\right)_{ij} \mathbf{P}_i \mathbf{P}_j, \left(\nabla \mathbf{U}^{C}\right)_{ij} \mathbf{P}_i \mathbf{P}_j, \left(\nabla \mathbf{U}^{V4}\right)_{ij} \mathbf{P}_i \mathbf{P}_j \Big) \tag{8}$$

with

$$L(a,b,c) = \begin{cases} 0 & \text{if } ab \le 0 \\ \operatorname{sign}(a) \min(2\,|a|, 2\,|b|, |c|) & \text{otherwise} \end{cases}.$$

The operator L defined above is applied component by component.

3.2.5 Boundary conditions

Boundary conditions are computed vertexwise. Several conditions are used, but only one, the slipping condition, is applied to moving bodies. In the context of ALE simulation, this condition has to take into account the displacement of the body. Consequently, we impose weakly[1]

$$\mathbf{u}_i \cdot \mathbf{n}_i = \sigma_i, \tag{9}$$

where $\mathbf{n}_i$ is the DGCL compatible unitary boundary face normal and σ_i is the boundary face velocity.

The standard ALE slipping boundary flux of vertex P_i reduces to:

$$\mathbf{\Phi}_{\text{slip}}\left(\mathbf{W}_i, \mathbf{n}_i, \sigma_i\right) = \begin{pmatrix} 0 \\ -p_i \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|} \\ -p_i \sigma_i \end{pmatrix} |\partial C_i|_\Gamma, \tag{10}$$

where $|\partial C_i|_\Gamma = \sum_{K_j \supset P_i} \frac{1}{3} |K_j|$ is the interface area, see Sect. 3.3.6, p_i is the vertex pressure, $\mathbf{n}_i = \frac{\sum_{K_j \supset P_i} |K_j| \mathbf{n}_{K_j}}{\sum_{K_j \supset P_i} |K_j|}$ is the mean outward normal of the boundary interface and . However, when high-order numerical schemes are considered, such a boundary condition creates oscillations in the density and the pressure when shock waves impact normally the boundary. We thus prefer considering a mirror state and apply an approximate Riemann state to diminish these oscillations.

We thus have to evaluate the flux between the state on the boundary $\mathbf{W}$ and the ALE mirror state $\overline{\mathbf{W}}$:

$$\mathbf{W}_i = \begin{pmatrix} \rho_i \\ \rho_i \mathbf{u}_i \\ \rho_i E_i \end{pmatrix} \quad \text{and} \quad \overline{\mathbf{W}}_i = \begin{pmatrix} \rho_i \\ \rho \mathbf{u}_i - 2\,\rho_i\,(\mathbf{u}_i \cdot \mathbf{n}_i - \sigma_i)\,\mathbf{n}_i \\ \rho E_i - 2\,\rho_i\,\sigma_i\,(\mathbf{u}_i \cdot \mathbf{n}_i - \sigma_i) \end{pmatrix}$$

as the mirror state verifies

$$\overline{p}_i = p_i, \quad \overline{\varepsilon}_i = \varepsilon_i, \quad \overline{c}_i = c_i, \quad \overline{\mathbf{u}}_i \cdot \overline{\mathbf{n}}_i = 2\,\sigma_i - \mathbf{u}_i \cdot \mathbf{n}_i \quad \text{and} \quad \overline{H}_i = H_i - 2\,\sigma_i\,(\mathbf{u}_i \cdot \mathbf{n}_i - \sigma_i).$$

To evaluate the boundary flux, we consider the HLLC numerical flux between the state and the mirror state:

$$\mathbf{\Phi}_{\text{slip}}\left(\mathbf{W}_i, \mathbf{n}_i, \sigma_i\right) = \mathbf{\Phi}_{\text{HLLC}}\Big(\mathbf{W}_i, \overline{\mathbf{W}}_i, \mathbf{n}_i, \sigma_i\Big). \tag{11}$$

Note that by definition we have

$$\mathbf{\Phi}_{\text{HLLC}}(\mathbf{W}_i, \mathbf{W}_i, \mathbf{n}_i, \sigma_i) = \mathbf{F}(\mathbf{W}_i) - \sigma_i \mathbf{W}_i.$$

Thus, if Condition (9) is satisfied, then $\mathbf{W}_i = \overline{\mathbf{W}}_i$ and the flux $\mathbf{\Phi}_{\text{slip}}\left(\mathbf{W}_i, \mathbf{n}_i, \sigma_i\right)$ simplifies to the form in Relation (10). In general, this condition is not satisfied, so we use Relation (11).

[1] We do not enforce the numerical solution to verify $\mathbf{u}_i.\mathbf{n}_i = \sigma_i$

3.3 Time discretization

Temporal discretization is a more complex matter. In this work, we chose an explicit time discretization, which is simpler than implicit discretizations. Our time discretization is compliant with the discrete geometric conservation law, which can be used to rigorously determine when the geometric parameters that appear in the fluxes should be computed.

3.3.1 The geometric conservation law

We need to make sure that the movement of the mesh is not responsible for any artificial alteration of the physical phenomena involved, or at least, to make our best from a numerical point of view for the mesh movement to introduce an error of the same order as the one introduced by the numerical scheme. If System (4) is written for a constant state, assuming $\mathbf{F}_{\text{ext}} = 0$, we get, for any arbitrary closed volume $C = C(t)$:

$$\frac{\mathrm{d}(|C(t)|)}{\mathrm{d}t} - \int_{\partial C(t)} (\boldsymbol{w} \cdot \mathbf{n}) \, \mathrm{d}s = 0. \tag{12}$$

As the constant state is a solution of the Euler equations, if boundaries transmit the flux towards the outside as it comes, we find a purely geometrical relation inherent to the continuous problem. For any arbitrary closed volume $C = C(t)$ of boundary $\partial C(t)$, Relation (12) is integrated into:

$$|C(t+\delta t)| - |C(t)| = \int_t^{t+\delta t} \int_{\partial C(t)} (\boldsymbol{w} \cdot \mathbf{n}) \, \mathrm{d}s \mathrm{d}t, \quad \text{with } t \text{ and } t+\delta t \in [0, T], \tag{13}$$

which is usually known as the geometric conservation law (GCL). From a geometric point of view, this relation states that the algebraic variation of the volume of C between two instants equals the algebraic volume swept by its boundary.

The role of the GCL in ALE simulations has been analyzed in [22]. It has been shown that the GCL is neither a necessary nor a sufficient condition to preserve time accuracy; however, violating it can lead to numerical oscillation [46]. In [24] the authors show that compliance with the GCL guarantees an accuracy of at least the first order in some conditions. Therefore, most would agree that the GCL should be enforced at the discrete level for a large majority of cases.

3.3.2 Discrete GCL enforcement

A new approach to enforcing the discrete GCL was proposed in [46, 57, 58], in which the authors proposed a framework to build ALE high-order temporal schemes that reach approximately the design order of accuracy. The originality of this approach consists in precisely defining which ALE parameters are true degrees of freedom and which are not. In contrast to other approaches [35, 38, 48], they consider that the times and configurations at which the fluxes are evaluated do not constitute a new degree of freedom to be set thanks to the ALE scheme. To maintain the design accuracy of the fixed-mesh temporal integration, the moment at which the geometric parameters, such as the cells' interfaces' normals or the upwind/downwind tetrahedra must be computed, is entirely determined by the intermediate configurations involved in the chosen temporal scheme. The only degree of freedom to be set by enforcing the GCL at the discrete level is σ. Incidentally, it is implicitly stated that $\boldsymbol{w}$ is never involved alone but only hidden in the term $\sigma\|\mathbf{n}\|$ which represents the instantaneous algebraic volume swept.

In practical terms, the interfaces normal speeds are found by simply rewriting the scheme for a constant discrete solution, which leads to a small linear system that is easily invertible by hand. This procedure is detailed in the next section for one Runge–Kutta scheme. Any fixed-mesh explicit RK scheme can be extended to the case of moving meshes thanks to this methodology, and the resulting RK scheme is naturally DGCL. Even if this has not been proven theoretically, the expected temporal order of convergence has also been observed numerically for several schemes designed using this method [57].

3.3.3 RK schemes

Runge–Kutta (RK) methods are famous multi-stage methods to integrate ODEs. In the numerical solution of hyperbolic PDEs, notably the Euler equations, the favorite schemes among the huge family of Runge–Kutta schemes are those satisfying the strong stability preserving (SSP) property [53, 55]. In what follows, we denote by SSPRK(S,P) the S-stage RK scheme of order P. We adopt the following notations:

$$\mathbf{f}_i^s = \sum_{j=1}^{n_i} \boldsymbol{\Phi}(\mathbf{W}_i^s, \mathbf{W}_j^s, \boldsymbol{\eta}_{ij}^s, \sigma_{ij}^s), \tag{14}$$

with

$$\begin{cases} n_i & \text{the number of edges in the ball of vertex } P_i \\ \boldsymbol{\eta}_{ij}^s & \text{the outward } \textbf{non - normalized} \text{ normal to the portion of the interface of cell } C_i^s \text{ around edge } e_{ij} \\ \sigma_{ij}^s & \text{normal speed of the interface around edge } e_{ij} \text{ of cell } C_i^s. \end{cases}$$

Superscript notation X^s indicates that the quantity considered is the X obtained at stage s of the Runge–Kutta process.

Table 1 Butcher and Shu–Osher representations of the third-order 4-step Runge–Kutta scheme (SSPRK(4,3))

Butcher representation	Shu–Osher representation	$t^s = t^n + c_s\,\tau$
$\mathbf{Y}_i^{\,0} = \mathbf{Y}_i^{\,n}$	$\mathbf{Y}_i^{\,0} = \mathbf{Y}_i^{\,n}$	$c_0 = 0, \quad t^0 = t^n$
$\mathbf{Y}_i^{\,1} = \mathbf{Y}_i^{\,0} + \frac{\tau}{2}\mathbf{f}_i^0$	$\mathbf{Y}_i^{\,1} = \mathbf{Y}_i^{\,0} + \frac{\tau}{2}\mathbf{f}_i^0$	$c_1 = \frac{1}{2}, \quad t^1 = t^n + \frac{1}{2}\tau$
$\mathbf{Y}_i^{\,2} = \mathbf{Y}_i^{\,0} + \frac{\tau}{2}\left(\mathbf{f}_i^0 + \mathbf{f}_i^1\right)$	$\mathbf{Y}_i^{\,2} = \mathbf{Y}_i^{\,1} + \frac{\tau}{2}\mathbf{f}_i^1$	$c_2 = 1, \quad t^2 = t^{n+1}$
$\mathbf{Y}_i^{\,3} = \mathbf{Y}_i^{\,0} + \frac{\tau}{6}\left(\mathbf{f}_i^0 + \mathbf{f}_i^1 + \mathbf{f}_i^2\right)$	$\mathbf{Y}_i^{\,3} = \frac{2}{3}\mathbf{Y}_i^{\,0} + \frac{1}{3}\mathbf{Y}_i^{\,2} + \frac{\tau}{6}\mathbf{f}_i^2$	$c_3 = \frac{1}{2}, \quad t^3 = t^n + \frac{1}{2}\tau$
$\mathbf{Y}_i^{\,4} = \mathbf{Y}_i^{\,0} + \frac{\tau}{2}\left(\frac{1}{3}\mathbf{f}_i^0 + \frac{1}{3}\mathbf{f}_i^1 + \frac{1}{3}\mathbf{f}_i^2 + \mathbf{f}_i^3\right)$	$\mathbf{Y}_i^{\,4} = \mathbf{Y}_i^{\,3} + \frac{\tau}{2}\mathbf{f}_i^3$	$c_4 = 1, \quad t^4 = t^{n+1}$

For instance, C_i^s is the cell associated with vertex P_i when the mesh has been moved to its sth Runge–Kutta configuration. Coefficients $\left(c_s\right)_{0\le s\le S}$ indicate the relative position in time of the current Runge–Kutta configuration: $t^s = t^n + c_s\,\tau \quad$ with $\tau = t^{n+1} - t^n$. Finally, we denote by A_{ij}^s the volume swept by the interface around edge e_{ij} of cell C_i between the initial Runge–Kutta configuration and the sth configuration.

3.3.4 Application to the SSPRK(4,3) scheme

This approach was used, for example, to build the third-order 4-step Runge–Kutta scheme [51], whose Butcher and Shu–Osher representations are given in Table 1.

For this scheme to be DGCL, it must preserve a constant solution $\mathbf{W}_i = \mathbf{W}_0$, as stated in Sect. 3.3.1. In this specific case, our conservative variable is $\mathbf{Y}_i = |C_i|\mathbf{W}_0$ and the purely physical fluxes vanish, leading to $\mathbf{f}_i^s = -\mathbf{W}_0 \sum_{j=1}^{n_i} \boldsymbol{\eta}_{ij}^s \sigma_{ij}^s$. Therefore, the scheme reads:

$$
\begin{aligned}
|C_i^0| &= |C_i^n| \\
|C_i^1| - |C_i^0| &= \sum_{j=1}^{n_i} A_{ij}^1 = \frac{\tau}{2}\sum_{j=1}^{n_i} \boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 \\
|C_i^2| - |C_i^0| &= \sum_{j=1}^{n_i} A_{ij}^2 = \frac{\tau}{2}\sum_{j=1}^{n_i} \left(\boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 + \boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1\right) \\
|C_i^3| - |C_i^0| &= \sum_{j=1}^{n_i} A_{ij}^3 = \frac{\tau}{6}\sum_{j=1}^{n_i} \left(\boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 + \boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1 + \boldsymbol{\eta}_{ij}^2 \sigma_{ij}^2\right) \\
|C_i^4| - |C_i^0| &= \sum_{j=1}^{n_i} A_{ij}^4 = \frac{\tau}{2}\sum_{j=1}^{n_i} \left(\frac{1}{3}\boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 + \frac{1}{3}\boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1 + \frac{1}{3}\boldsymbol{\eta}_{ij}^2 \sigma_{ij}^2 \right. \\
&\qquad \left. + \boldsymbol{\eta}_{ij}^3 \sigma_{ij}^3\right),
\end{aligned}
$$

where A_{ij}^s is the volume swept by the interface around edge e_{ij} of cell C_i between the initial and the s^{th} Runge–Kutta configuration. A natural necessary condition for the above relations to be satisfied is to have, for each interface around edge e_{ij} of each finite volume cell C_i:

$$
\begin{aligned}
&\begin{pmatrix} A_{ij}^1 \\ A_{ij}^2 \\ A_{ij}^3 \\ A_{ij}^4 \end{pmatrix} = \tau \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 \\ \boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1 \\ \boldsymbol{\eta}_{ij}^2 \sigma_{ij}^2 \\ \boldsymbol{\eta}_{ij}^3 \sigma_{ij}^3 \end{pmatrix} \\
&\Leftrightarrow \begin{pmatrix} \boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 \\ \boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1 \\ \boldsymbol{\eta}_{ij}^2 \sigma_{ij}^2 \\ \boldsymbol{\eta}_{ij}^3 \sigma_{ij}^3 \end{pmatrix} = \frac{1}{\tau} \begin{pmatrix} 2 & 0 & 0 & 0 \\ -2 & 2 & 0 & 0 \\ 0 & -2 & 6 & 0 \\ 0 & 0 & -2 & 2 \end{pmatrix} \begin{pmatrix} A_{ij}^1 \\ A_{ij}^2 \\ A_{ij}^3 \\ A_{ij}^4 \end{pmatrix}.
\end{aligned}
\tag{15}
$$

Therefore, the normal speed of the interface around edge e_{ij} of cell C_i must be updated in the Runge–Kutta process as follows :

$$
\sigma_{ij}^0 = \frac{2A_{ij}^1}{\tau\,\boldsymbol{\eta}_{ij}^0}, \quad \sigma_{ij}^1 = \frac{-2A_{ij}^1 + 2A_{ij}^2}{\tau\,\boldsymbol{\eta}_{ij}^1}, \quad \sigma_{ij}^2 = \frac{-2A_{ij}^2 + 6A_{ij}^3}{\tau\,\boldsymbol{\eta}_{ij}^2}, \text{and}
$$
$$
\sigma_{ij}^3 = \frac{-2A_{ij}^3 + 2A_{ij}^4}{\tau\,\boldsymbol{\eta}_{ij}^3},
\tag{16}
$$

and the $\boldsymbol{\eta}_{ij}^s$ and A_{ij}^s are computed on the mesh once it has been moved to the sth Runge–Kutta configuration.

3.3.5 Practical computation of the volumes swept

The interface of a finite volume cell is made up of several triangles, connecting the middle of an edge to the center of gravity of a face and the center of gravity of that tetrahedron; see Fig. 3. The two triangles of the interface sharing one edge within a tetrahedron are coplanar (i.e., the middle of an edge, the center of gravity of the two faces neighboring this edge and the center of gravity of tetrahedron are coplanar). The union of these two triangles is called the facet associated to the edge and the tetrahedron.

At configuration $t^s = t^0 + c_s\,\tau$, (with $t^0 = t^n$), the outward non-normalized normal $\boldsymbol{\eta}_{ij}^s$ and the volume swept A_{ij}^s are com-

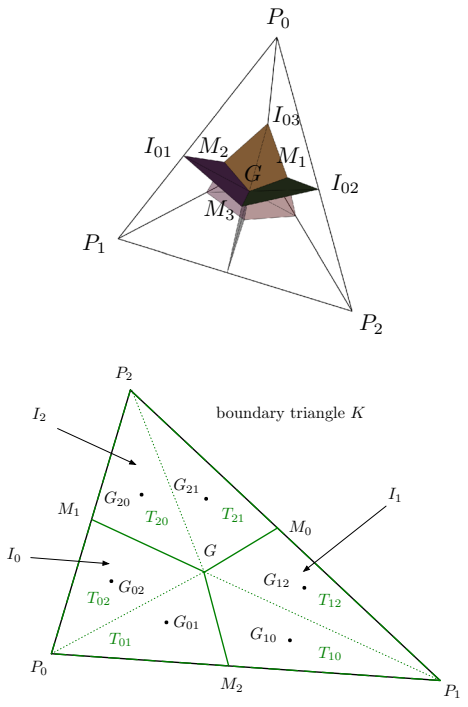

Fig. 3 Interface of a finite volume cell made up of facets (top) and boundary interface (bottom)

puted as described in [49]. As the cell interface is made up of several facets, the total swept volume is the sum of the volumes swept by each facet.

Let us assume that the facet considered is associated with edge $\mathbf{e}_{ij} = P_i P_j$ and belongs to tetrahedron $K = (P_0, P_1, P_2, P_3)$. In what follows, $i \neq j \neq l \neq m \in [\![0, 3]\!]$, G denotes the center of gravity of tetrahedron K, M_m denotes the gravity center of face $F_m = (P_i, P_j, P_l)$ of tetrahedron K and M_l the center of gravity of face $F_l = (P_i, P_j, P_m)$. The outward non-normalized normal of the facet is given by:

$$\begin{aligned}\tilde{\boldsymbol{\eta}}^s_{ij,K} =& \frac{1}{4}\mathbf{G^0M^0_m} \wedge \mathbf{G^sM^s_l} + \frac{1}{4}\mathbf{G^sM^s_m} \wedge \mathbf{G^0M^0_l} \\ &+ \frac{1}{2}\mathbf{G^0M^0_m} \wedge \mathbf{G^0M^0_l} + \frac{1}{2}\mathbf{G^sM^s_m} \wedge \mathbf{G^sM^s_l},\end{aligned} \tag{17}$$

where

$$\begin{aligned}\mathbf{G^0M^0_m} &= \frac{1}{12}\left(P^0_i + P^0_j - 3P^0_m + P^0_l\right),\\ \mathbf{G^sM^s_m} &= \frac{1}{12}\left(P^s_i + P^s_j - 3P^s_m + P^s_l\right),\\ \mathbf{G^0M^0_l} &= \frac{1}{12}\left(P^0_i + P^0_j + P^0_m - 3P^0_l\right) \text{ and}\\ \mathbf{G^sM^s_l} &= \frac{1}{12}\left(P^s_i + P^s_j + P^s_m - 3P^s_l\right).\end{aligned}$$

The volume swept by the facet is:

$$A^s_{ij,K} = c_s\,\tau\,(\boldsymbol{w}_G)^s_{ij,K} \cdot \tilde{\boldsymbol{\eta}}^s_{ij,K}, \tag{18}$$

with the mean velocity written as:

$$(\boldsymbol{w}_G)^s_{ij,K} = \frac{1}{36\,c_s\,\tau}\left(13\boldsymbol{w}^s_i + 13\boldsymbol{w}^s_j + 5\boldsymbol{w}^s_m + 5\boldsymbol{w}^s_l\right)$$

with

$$\boldsymbol{w}^s_\xi = P^s_\xi - P^0_\xi.$$

Finally, the total volume swept by the interface around edge e_{ij} of cell C_i is obtained by summing over the shell of the edge, i.e., all the tetrahedra sharing the edge:

$$A^s_{ij} = \sum_{K \in \text{Shell}(i,j)} A^s_{ij,K}. \tag{19}$$

It is important to understand that normals $\tilde{\boldsymbol{\eta}}^s_{ij,K}$ are pseudo-normals which are used only to compute the volumes swept by the facets. They must not be mistaken for normals to facets taken at t^s, $\boldsymbol{\eta}^s_{ij,K}$, which are used for the computation of ALE fluxes.

3.3.6 Volumes swept by boundary interfaces

The pseudo-normals and swept volumes of boundary faces are computed in a similar way. Let $K = (P_0, P_1, P_2)$ be a boundary triangle, as in Fig. 3. Let M_0, M_1 and M_2 be the middles of the edges and G the center of gravity of K. The triangle is made up of three quadrangular finite volume interfaces: (P_0, M_2, G, M_1) associated with cell C_0, (P_1, M_0, G, M_2) with C_1 and (P_2, M_1, G, M_0) with C_2. Each boundary interface I_i is made of two sub-triangles, noted T_{ij} and T_{ik} with $j, k \neq i$.

The volume swept by interface I_i between the initial and current Runge–Kutta configurations is the sum of the volumes swept by its two sub-triangles:

$$A^s_{i,K} = A^s_{T_{ij}} + A^s_{T_{ik}} = c_s\tau\,\boldsymbol{w}^s_{G_{ij}} \cdot \tilde{\boldsymbol{\eta}}^s_{ij} + c_s\tau\,\boldsymbol{w}^s_{G_{ik}} \cdot \tilde{\boldsymbol{\eta}}^s_{ik}, \tag{20}$$

where $\boldsymbol{w}^s_{G_{ij}}$ is the velocity of the center of gravity G_{ij} of triangle T_{ij} and $\tilde{\boldsymbol{\eta}}^s_{ij}$ is the pseudo-normal associated with T_{ij}, computed between the initial and current Runge–Kutta configurations.

The six triangles T_{ij} are coplanar, so their pseudo-normals have the same direction. Moreover, as median cells are used, their pseudo-normals also have the same norm, which is equal to one sixth of the norm of the pseudo-normal to triangle K. This common pseudo-normal is therefore equal to:

$$\tilde{\boldsymbol{\eta}}^s = \frac{1}{6}\tilde{\boldsymbol{\eta}}^s_K = \frac{1}{6}\cdot\frac{1}{3}\left[\frac{1}{4}\mathbf{P}^0_0\mathbf{P}^0_1 \wedge \mathbf{P}^s_0\mathbf{P}^s_2 + \frac{1}{4}\mathbf{P}^s_0\mathbf{P}^s_1 \wedge \mathbf{P}^0_0\mathbf{P}^0_2 + \frac{1}{2}\mathbf{P}^0_0\mathbf{P}^0_1 \wedge \mathbf{P}^0_0\mathbf{P}^0_2 + \frac{1}{2}\mathbf{P}^s_0\mathbf{P}^s_1 \wedge \mathbf{P}^s_0\mathbf{P}^s_2\right], \tag{21}$$

thus

$$A^s_{i,K} = c_s\tau\left[\boldsymbol{w}^s_{G_{ij}} + \boldsymbol{w}^s_{G_{ik}}\right]\cdot\tilde{\boldsymbol{\eta}}^s, \tag{22}$$

where

$$\boldsymbol{w}^s_{G_{ij}} = \frac{1}{18\,c_s\,\tau}\left(11\boldsymbol{w}^s_i + 5\boldsymbol{w}^s_j + 2\boldsymbol{w}^s_k\right) \quad \text{with} \quad \boldsymbol{w}^s_\xi = P^s_\xi - P^0_\xi.$$

Finally, the total volume swept by the boundary interface is:

$$A^s_i = \sum_{K\in\mathrm{Ball}(i)} A^s_{i,K}. \tag{23}$$

3.3.7 MUSCL approach and RK schemes

Regarding spatial accuracy, we have seen that the order of accuracy can be enhanced using the MUSCL-type reconstruction with upwinding. However, in the ALE context, one must determine how and when upwind/downwind elements should be evaluated to compute upwind/downwind gradients which are necessary for the β-schemes. This question is neither answered in the literature and generally approximations are carried out. For instance, some papers propose to use upwind and downwind elements at t^n for the whole Runge–Kutta process. However, this choice should be consistent with the considered time integration scheme. Following the framework of [57], it is clear that preserving the expected order of accuracy in time imposes that the upwind/downwind elements and the gradients are computed on the current Runge–Kutta configuration, i.e., on the mesh at t^s. Therefore, similarly to geometric parameters, the upwind/downwind elements and the gradients should be re-evaluated at each step of the Runge–Kutta stage.

3.3.8 Computation of the time step

The maximal allowable time step for the numerical scheme is:

$$\tau(P_i) = \frac{h(P_i)}{c_i + \|\mathbf{u}_i - \boldsymbol{w}_i\|}, \tag{24}$$

where $h(P_i)$ is the smallest height in the ball of vertex P_i, c_i is the speed of sound, $\mathbf{u}_i$ is the Eulerian speed (the speed of the fluid, computed by the solver) and $\boldsymbol{w}_i$ is the speed of the mesh vertex. The global time step is then given by $\tau = \mathrm{CFL}\ \min_{P_i}(\tau(P_i))$.

3.3.9 Handling the swaps

An ALE formulation of the swap operator, satisfying the DGCL, was proposed in 2D in [51], but its extension to 3D is very delicate because it requires to handle 4D geometry, and it has not been carried out yet. Instead of considering an ALE scheme for the swap operator, our choice in this work is to perform the swaps between two solver iterations, i.e., at a fixed time t^n. This consequently means that during the swap phase, the mesh vertices do not move, and thus the swaps do not impact the ALE parameters $\boldsymbol{\eta}$ and $\boldsymbol{w}$, unlike [51] where the swaps are performed during the solver iteration, i.e., between t^n and t^{n+1}. After each swap, the solution should be updated on the new configuration. Two interpolation methods are considered here.

The first, and simplest, one is to perform a linear interpolation to recover the solution. As only the connectivity changes and not the vertices positions, the solution at the vertices does not change, i.e., nothing has to be done. This interpolation is DGCL compliant, since the constant state is preserved (in fact, any linear state is preserved), but it does not conserve the mass (i.e., it does not conserve the integral of the conservative variable) which is problematic for conservative equations when discontinuities are involved in the flow.

The second method is the P^1-exact conservative interpolation following [2, 5]. It is a simplified version of the latter because the cavity of the swap configuration is fixed. The mass conservation property of the interpolation operator is achieved by element–element intersections. The idea is to find, for each element of the new configuration, its geometric intersection with all the elements of the previous configuration it overlaps and to mesh this geometric intersection with simplices. We are then able to use a Gauss quadrature formula to exactly compute the mass which has been locally transferred. High-order accuracy is obtained through the reconstruction of the gradient of the solution from the discrete data and the use of some Taylor formulae. Unfortunately, this high-order interpolation can lead to a loss of monotonicity. The maximum principle is recovered by correcting the interpolated solution in a conservative manner, using a limiter strategy very similar to the one used for finite volume solvers. Finally, the solution values at vertices are reconstructed from this piecewise linear by element discontinuous representation of the solution. The algorithm is summarized in Algorithm 2, where

m_K stands for the integral of any conservative quantities (density, momentum and energy) on the considered element. This method is also compliant with the DGCL.

Algorithm 2 Conservative Interpolation Process

1. For all elements K_{back} of the original cavity, compute solution mass $m_{K_{back}}$ and gradient $\nabla_{K_{back}}$
2. For all elements K_{new} of the new cavity, recover solution mass $m_{K_{new}}$ and gradient $\nabla_{K_{new}}$:
   (a) compute the intersection of K_{new} with all K^i_{back} it overlaps
   (b) mesh the intersection polygon/polyhedra of each couple of elements (K_{new}, K^i_{back})
   (c) compute $m_{K_{new}}$ and $\nabla_{K_{new}}$ using Gauss quadrature formulae
   $\Longrightarrow$ a piecewise linear discontinuous representation of the mass on the new cavity is obtained
3. Correct the gradient to enforce the maximum principle
4. Set the solution values to vertices by an averaging procedure

Moreover, after each swap, the data of the finite volume cells (volume and interface normals) are updated, together with the topology of the mesh (edges and tetrahedra). This requires to have a flow solver with dynamic data. In Sects. 6.1 and 6.2, we will quantify the error due to the use of swaps in numerical simulations.

4 FSI coupling

The moving boundaries can have an imposed motion, or be driven by fluid–structure interaction. A simple solid mechanics solver is coupled to the flow solver described previously. The chosen approach is the 6-DOF (6 Degrees of Freedom) approach for rigid bodies.

4.1 Movement of the geometries

In this work, the bodies are assumed to be rigid, of constant mass and homogeneous, i.e., their mass is uniformly distributed in their volume. The bodies we consider will never break into different parts. Each rigid body B is fully described by:

Physical quantities Its boundary ∂B and its associated inward normal $\mathbf{n}$, its mass m assumed to be constant, its $d \times d$ matrix of inertia $\mathcal{J}_G$ computed at G which is symmetric and depends only on the shape and physical nature of the solid object.[2]

Kinematic quantities The position of its center of gravity $\mathbf{x_G} = (x(t), y(t), z(t))$, its angular displacement vector $\boldsymbol{\theta} = \boldsymbol{\theta}(t)$ and its angular speed vector $\mathrm{d}\boldsymbol{\theta}/\mathrm{d}t$.

[2] The moment of inertia relative to an axis of direction $\mathbf{a}$ passing through G where $\mathbf{a}$ is an arbitrary unit vector is given by: $J_{(G\mathbf{a})} = \mathbf{a}^T \mathcal{J}_G \mathbf{a}$.

$\mathbf{F}_{\text{ext}}$ denotes the resultant vector of the external forces applied on B, $\mathbf{M}_G(\mathbf{F}_{\text{ext}})$ the kinetic moment of the external forces applied on B computed at G and $\mathbf{g}$ the gravity vector. We assume that the bodies are only submitted to forces of gravity and fluid pressure. The equations for solid dynamics in an inertial frame then read:

$$\begin{cases} m\dfrac{\mathrm{d}^2\mathbf{x_G}}{\mathrm{d}t^2} &= \mathbf{F}_{\text{ext}} &= \displaystyle\int_{\partial B} p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\mathrm{d}\boldsymbol{s} \;+\; m\mathbf{g} \\ \mathcal{J}_G\dfrac{\mathrm{d}^2\boldsymbol{\theta}}{\mathrm{d}t^2} &= \mathbf{M}_G(\mathbf{F}_{\text{ext}}) &= \displaystyle\int_{\partial B} \left[\left(\boldsymbol{s} - \mathbf{x_G}\right) \times p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\right]\mathrm{d}\boldsymbol{s}. \end{cases} \tag{25}$$

4.2 Discretization

The equation governing the position of the center of gravity of the body is easy to solve since it is linear. Its discretization is straightforward. However, the discretization of the second equation, which controls the orientation of the body, is more delicate. Since the matrix of inertia $\mathcal{J}_G$ depends on $\boldsymbol{\theta}$, it is a non-linear second-order ODE. The chosen discretization is extensively detailed in [50] and is based on rewriting of the equations in the frame of the moving body.

4.3 Explicit coupling

As the geometry must be moved in accordance with the fluid computation, the same time integration scheme has been taken to integrate the fluid and the solid equations. Therefore, time-advancing of the rigid bodies ODE System is performed using the same RKSSP scheme as the one used to advance the fluid numerical solution. The coupling is loose and explicit as the external forces and moments acting on rigid objects are computed on the current configuration.

5 Implementation and parallelization

Most parts of the code are parallelized with p-threads using an automatic p-thread parallelization library [45] coupled with a space filling curve renumbering strategy [4]. The Hilbert space filling curve based renumbering strategy is used to map mesh geometric entities, such as vertices, edges, triangles and tetrahedra, into a one-dimensional interval. In numerical applications, it can be viewed as a mapping from the computational domain onto the memory of a computer. The local property of the Hilbert space filling curve implies that entities which are neighbors on the memory 1D interval are also neighbors in the computational domain. Therefore, such a renumbering strategy has a significant impact on the efficiency of a code. We can state the following benefits: it reduces the number of cache misses during indirect addressing loops, and it reduces

cache-line overwrites during indirect addressing loops, which is fundamental for shared memory parallelism.

The automatic parallelization library cuts the data into small chunks that are compact in terms of memory (because of the renumbering), then uses a dynamic scheduler to allocate the chunks to the threads to limit concurrent memory accesses. In the case of a loop performing the same operation on each entry of a table, the loop is split into many sub-loops. Each sub-loop will perform the same operation (symmetric parallelism) on equally-sized portions of the main table and will be concurrently executed. It is the scheduler's job to make sure that two threads do not write on the same memory location simultaneously. When indirect memory accesses occur in loops, memory write conflicts can still arise. To deal with this issue, an asynchronous parallelization is considered rather than of a classic gather/scatter technique, i.e., each thread writes in its own working array and then the data are merged.

Our moving mesh strategy performs frequent mesh optimizations that impact the ALE speeds of the vertices, so it is much more efficient to have the whole moving mesh part included in the solver code. To handle the moving mesh, semi-dynamic data structures are necessary. Since we do not do vertex insertion or deletion, only some of the data structures have to be dynamic (edge and tetrahedra lists), whereas some remain static (the list of vertices, as well as all the data associated to vertices). In the case of dynamic data, the appropriate organization of the memory which reduces concurrent accesses can be lost, and thus imposes re-sorting the data according to the Hilbert numbering from time to time in order to maintain a good speedup.

Some parts of the code have not yet been parallelized, in particular the elasticity solution and the mesh optimization step. The optimizations are difficult to parallelize because they are greatly dependent on the order in which they are performed. Although changing this order leads to the same overall result, it is not exactly the same, and thus, the dynamic handling of the library makes each run non reproducible. To have a parallel and reproducible mesh optimization step, the operators should be rewritten with different algorithms to make them independent of the process order. Such algorithms are generally less efficient in terms of CPU time.

6 Verification of the solver

In the following section, we present academic test cases in order to assess the behavior of the different parts of the solver: the fluid ALE solver, the rigid body dynamic solver and the FSI coupling.

6.1 Static vortex in a rotating mesh

The first test case is used to validate the ALE solver, and study the impact of the moving mesh on the accuracy of the solution. We study the conservation of a static vortex (similar to [34]) on a rotating mesh.

This case is an extension of a 2D case: we consider an extruded cylinder, and the initial solution is constant along the z axis. The domain is a cylinder of radius 5 and of height 1. The initial solution is defined as follows. Let $r_0 = 1$ be a characteristic radius and the reference state be:

$$\rho_\infty = 1,\ u_{x\infty} = 0,\ u_{y\infty} = 0,\ u_{z\infty} = 0,\ p_\infty = 1 \text{ and } \gamma = 1.4. \tag{26}$$

Let us define the following quantities:

$$D_\infty = \frac{1}{2}\frac{1}{(2\pi)^2}\frac{\rho_\infty}{p_\infty}, = \frac{1}{8\pi^2} \quad B_\infty = 2r_0^2 - \frac{\gamma-1}{\gamma}D_\infty,$$
$$E_\infty = (2r_0^2)^2 - B_\infty^2.$$

For a point of cylindrical coordinates (r, θ, z), the enthalpy and speed of the vortex are:

$$H_{\text{vor}} = \frac{\gamma}{\gamma-1}\frac{p_\infty}{\rho_\infty} + \frac{1}{2}(u_{x\infty}^2 + u_{y\infty}^2 + u_{z\infty}^2), \qquad v_{\text{vor}} = \frac{1}{2\pi}\frac{r}{r^2+1}.$$

And finally the initial solution is:

$$\begin{cases} u_{x0} = u_{x\infty} - v_{\text{vor}}\sin\theta = -v_{\text{vor}}\sin\theta \\ u_{y0} = u_{y\infty} + v_{\text{vor}}\cos\theta = v_{\text{vor}}\cos\theta \\ u_{z0} = 0 \\ p_0 = p_\infty \exp\left(\frac{2D_\infty}{\sqrt{E_\infty}}\operatorname{atan}\left(\frac{2r^2+B_\infty}{\sqrt{E_\infty}}\right) - \frac{\pi}{2}\right) \\ \rho_0 = \frac{\frac{\gamma}{\gamma-1}p_0}{H_{\text{vor}} - \frac{1}{2}v_{\text{vor}}}. \end{cases} \tag{27}$$

This initial configuration results in a cylindrical vortex rotating around the z axis, that remains in a constant state over time (density, pressure and speed are preserved). To avoid errors due to boundary conditions, the exact solution is enforced for every vertex farther than 90% of the cylinder radius (typically just a few layers of elements). The mesh is rotated rigidly around the z axis. Several rotation speeds were considered, all of which led to the same conclusions. Here, we only present the results for an angular speed of 0.34° per time unit, which corresponds approximately to the speed of the vortex at a radius $r = 5$. We address both space and time convergence.

6.1.1 Space convergence

For the space convergence study, we consider a set of uniform meshes of sizes varying from 10,000 vertices to 1.8 million vertices; see Table 2. The simulation is run until

Table 2 Test case of the static vortex without swaps. Sizes of the meshes used and number of solver time steps performed

	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5	Mesh 6	Mesh 7	Mesh 8	Mesh 9
# vertices ($\times 10^3$)	12	44	138	185	256	371	585	965	1825
# tetrahedra ($\times 10^3$)	48	211	691	942	1330	1965	3122	5287	10,218
# time steps	4673	7280	15,608	14,321	14,321	17,275	29,996	32,245	44,680

time $t = 540$, which corresponds to half a revolution of the cylinder. Since an exact solution is known, it is easy to compute an error with respect to this exact solution on each mesh. Two cases are compared: in one case the mesh is fixed, and in the other case the mesh is rotated as described previously. Since no mesh optimizations are performed, it allows us to make sure that the extra ALE terms are well computed. The results are shown in Fig. 4. On the left graph, we plot the error varying with time for the different meshes, fixed (red) and rotating (blue). We can see that, as expected, the error diminishes as the size of the mesh grows, but more significantly the curves of the error for the moving and static meshes are almost superimposed. This confirms the accuracy of the ALE scheme. To study the spatial order of convergence, the error is plotted with respect to the mesh size at different times on the right-hand graph. We can see that the order of convergence reached is slightly greater than 2, which is coherent with the space scheme used.

6.1.2 Time convergence

This study was carried out in 2D for reasons of efficiency. Our goal was to study the impact of the Runge–Kutta schemes used and the CFL parameter. We consider a disc of radius 5, with the same initial solution as previously (Eq. 27) and rotating with the same speed. Three time discretizations are used: a standard first-order explicit scheme (SSPRK(1,1)), a five-step second-order Runge–Kutta scheme (SSPRK(5,2)) and a four-step third-order Runge–Kutta scheme (SSPRK(4,3)) described in Sect. 3.3.4. Four CFL numbers are set for each case: $\mathrm{CFL}_{\max}$, $3/4 \times \mathrm{CFL}_{\max}$, $1/2 \times \mathrm{CFL}_{\max}$ and $1/10 \times \mathrm{CFL}_{\max}$. $\mathrm{CFL}_{\max}$ is, respectively, 1, 4 and 2 for the schemes SSPRK(1,1), SSPRK(5,2) and SSPRK(4,3).

The results are gathered in Fig. 5. On the first graph, the error over time is plotted for the different temporal schemes and CFL numbers, while on the second, we plot the error varying with the CFL number at different time steps. On both graphs, we can see that the error decreases with the order of the Runge–Kutta scheme and with the CFL number. Whereas the standard explicit scheme is very sensitive to the time step, the error does not vary much for the SSPRK(4,3) and SSPRK(5,2). What is interesting to notice is that we have to use a CFL number of 0.1 $\mathrm{CFL}_{\max}$ for a standard explicit scheme to reach the same level of accuracy as the SSPRK(4,3) scheme with a CFL number at its maximum or the the SSPRK(5,2) scheme with a CFL number of 0.75 times its maximum. To reach a non-dimensional time equal to 3 with the standard explicit scheme with a CFL of 0.1, 30 time steps are required, whereas only 5 are required for the SSPRK(5,2) scheme with a CFL of 3, and 6 for the SSPRK(4,3) with a CFL of 2. Thus we can go 6 times faster with the high-order Runge–Kutta schemes and still obtain the same solution accuracy, which is obviously interesting.

Influence of swaps

It is important to study the impact of edge/face swaps on the solution accuracy in 3D. However, it is difficult to set up a meaningful test case with swaps. The number of swaps should be constant (in proportion to the size of the mesh),

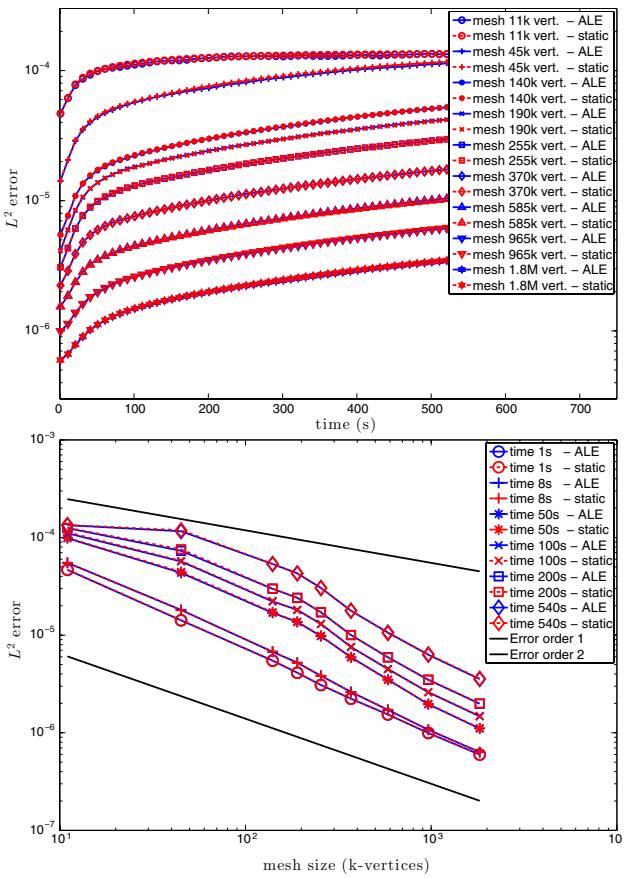

Fig. 4 Test case of the static vortex (3D). Error curves comparing the simulation on a fixed mesh (red) and on a moving mesh (blue) for several mesh sizes. Top, the evolution of the error over time, bottom convergence curves with respect to the size of the mesh. (Color figure online)

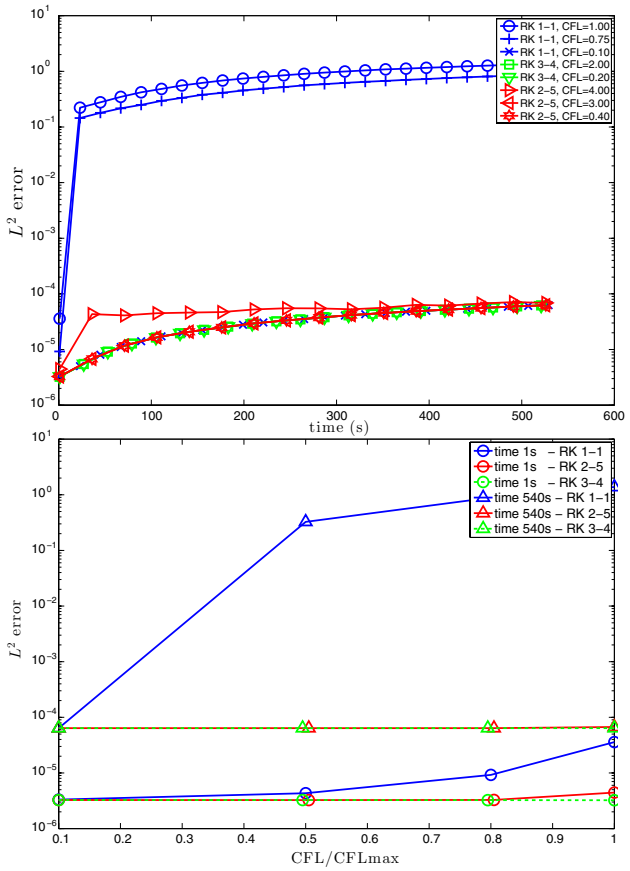

Fig. 5 Test case of the static vortex (2D). Error curves comparing several temporal schemes: standard explicit Euler scheme (blue), SSPRK(4,3) (green), SSPRK(5,2) (red). Top, evolution of the error over time. Bottom, error at a fixed time varying with the CFL number. (Color figure online)

and they should also be uniformly distributed in both time and space.

On all the simulations run with the connectivity-change moving mesh strategy, we noticed that, on average, less than one swap per 10,000 tetrahedra and per time step is performed. To fit this observation, in this example, we swap all the bad elements to preserve the mesh quality every 15 solver time steps. Then, if not enough swaps were performed, we randomly swap elements to reach a number of 1 swap per 666 tetrahedra. Not all random swaps are actually performed, because some of them affect the quality of the mesh too drastically, so the number of swaps is not perfectly controlled but this method allows us to have an average number of swaps close to one per 10,000 tetrahedra and per time step for all the meshes, evenly distributed in the domain, as stated in Table 3.

We again consider 3D meshes, rotating with the same speed of 0.34° per time unit, and the simulations are run up to 270s. In the absence of discontinuity in the solution, a linear interpolation is performed after the swaps. The results are gathered in Fig. 6. The error is slightly higher with swaps (in green) than without (in blue and red), however the error curves remain very close to those without swaps. The discrepancy can mainly be explained by the non conservative characteristic of the linear interpolation. Even without a conservative formulation, the error introduced in this example is not so large, and the same second-order convergence rate is observed. In the next example, we analyze the impact of a conservative formulation in the case of discontinuities in the solution.

To sum up, this static vortex test case allowed us to validate our ALE solver. We asymptotically reach an order of convergence of 2 for the spatial error, and we made sure there was no additional error introduced by the ALE terms. As regards temporal convergence, the importance of a high-order Runge–Kutta scheme was established. Finally, we showed that edge/face swapping—artificially created in this case, but mandatory to preserve the quality of the mesh (and thus the accuracy of the solution) when complex geometric displacements are involved—only creates a small error, even if used without a specific conservative treatment.

6.2 Sod's shock tube with a rotating circle

To show the impact of the choice of the interpolation when swaps are performed in the presence of discontinuities, we consider the well-known Sod's shock tube problem [54]. The tube domain size is $[0, 1] \times [0, 0.2]$, and initially, the tube is filled with two fluids at rest (left part is $x \leq 0.5$ and right part is $x > 0.5$) verifying:

$$\rho_{\text{left}} = 1, \quad \mathbf{u}_{\text{left}} = 0, \quad p_{\text{left}} = 1$$

and

$$\rho_{\text{right}} = 0.125, \quad \mathbf{u}_{\text{right}} = 0, \quad p_{\text{right}} = 0.1.$$

Table 3 Test case of the static vortex with swaps. Number of swaps for each mesh. On the last line, the number of swaps is close to one swap per time step and per 10,000 tetrahedra for all the meshes

	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5	Mesh 6	Mesh 7	Mesh 8	Mesh 9
# tetrahedra ($\times 10^3$)	48	211	691	942	1330	1965	3122	5287	10,218
# time steps	2338	3647	7788	7493	7367	8948	12,040	16,085	21,449
# swaps ($\times 10^3$)	21	114	763	996	1337	2319	5204	10,838	26,719
# swaps/time step/tet ($\times 10^{-4}$)	1.80	1.49	1.42	1.41	1.36	1.32	1.38	1.27	1.22

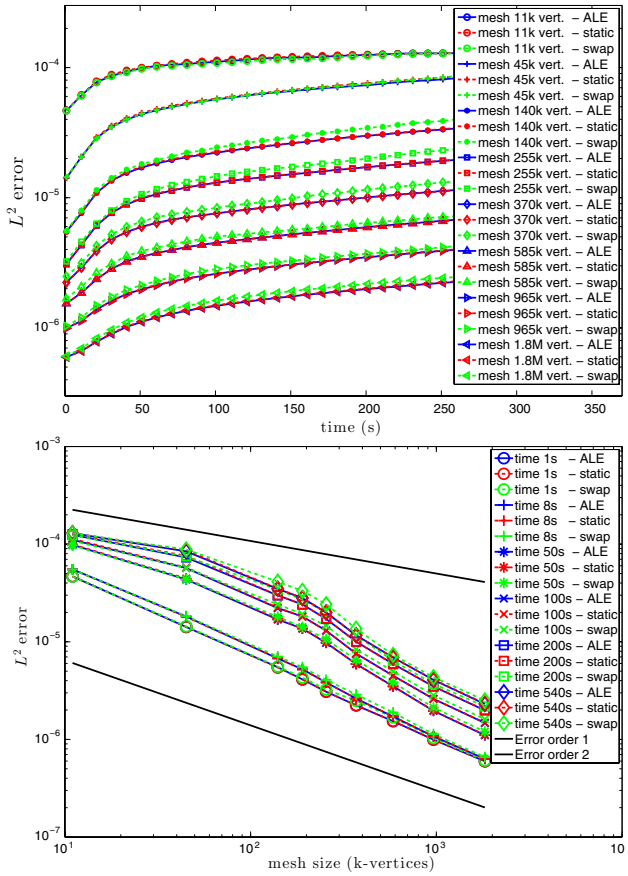

Fig. 6 Test case of the static vortex (3D). Error curves comparing the simulation on a fixed mesh (red), on a moving mesh without swaps (blue) and on a moving mesh with swaps (green) for several mesh sizes. Top, the evolution of the error over time, bottom, convergence curves with respect to the size of the mesh. (Color figure online)

The solution is computed until non-dimensioned time $t = 0.25$, and a shock wave and a contact discontinuity propagate on the right-hand side of the tube while a rarefaction wave propagates on the left-hand side. To analyze the ALE scheme and the impact of the swaps, we define a circular region of center (0.75, 0.1) and radius 0.05 which performs two full rotations within the simulation time frame; see Fig. 7. As the circular region is rotating, the mesh is sheared and swaps are performed around the circular region to maintain a good mesh quality. As the shock wave and the contact discontinuity move through the rotating region during the simulation time frame, we can analyze the impact of the interpolation stage when swaps are performed. For comparison, we also run the simulation on the same fixed mesh. Figure 8 shows the results on the fixed mesh (top), and on the moving mesh using the linear interpolation when swaps are performed (middle) and using the P^1-exact conservative interpolation (bottom). Extraction of the density profile along the line $y = 0.0483256$ is given in Fig. 9.

We can observe that some defects in the solution appear in the contact discontinuity when the linear interpolation is considered due to the rotation of a part of the domain. These defects disappear when the conservative interpolation is used. The defects are pointed out on the density profile when we zoom on the contact discontinuity; see Fig. 9 (right). But, they remain minor when we observe the overall profile.

6.3 Piston

The third validation test case is an FSI piston case from [37]. It is originally a 1D problem, which is extended to 3D. As shown in Fig. 10, we consider a gas contained in a 3D cylindrical chamber, closed at one end by a wall, and at the other end by a moving piston of mass m_p and of cross-section A_p. Besides the forces of pressure, the piston is submitted to a restoring force modeled by a spring of rigidity k_p. A natural frequency of the piston can be defined by $f_p = \frac{1}{2\pi}\sqrt{\frac{k_p}{m_p}}$. Let L_0 be the length of the chamber at rest, and $u(t)$ be the displacement of the piston with respect to the position at rest. The gas is initially at rest, and the piston is held in position u_0. We consider that all the variables are uniform on each cross-section of the cylinder (1D assumption).

It is shown in [37] that the piston is submitted to a resultant force applied to its center of gravity:

$$F_p = -k_p\, u(t) + A_p(p(t) - p(0)), \tag{28}$$

with $p(t)$ being the pressure on the piston. No effects of gravity are taken into account.

In practice, the domain at $t = 0$ is a rectangular box of dimensions $[-1, 0.2] \times [-0.5, 0.5] \times [-0.5, 0.5]$. The

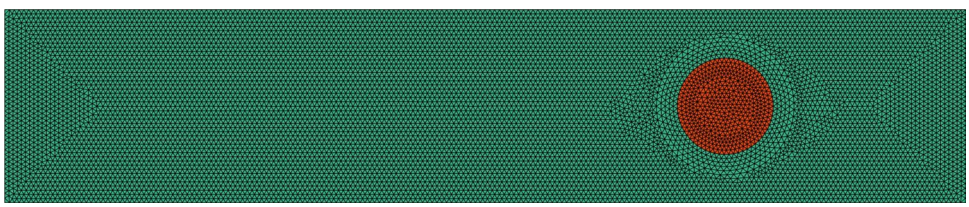

Fig. 7 Sod's shock tube problem with a rotating region. Domain geometry where the green region is fixed and the circular red region is rotating (performing two full rotations during the simulation). (Color figure online)

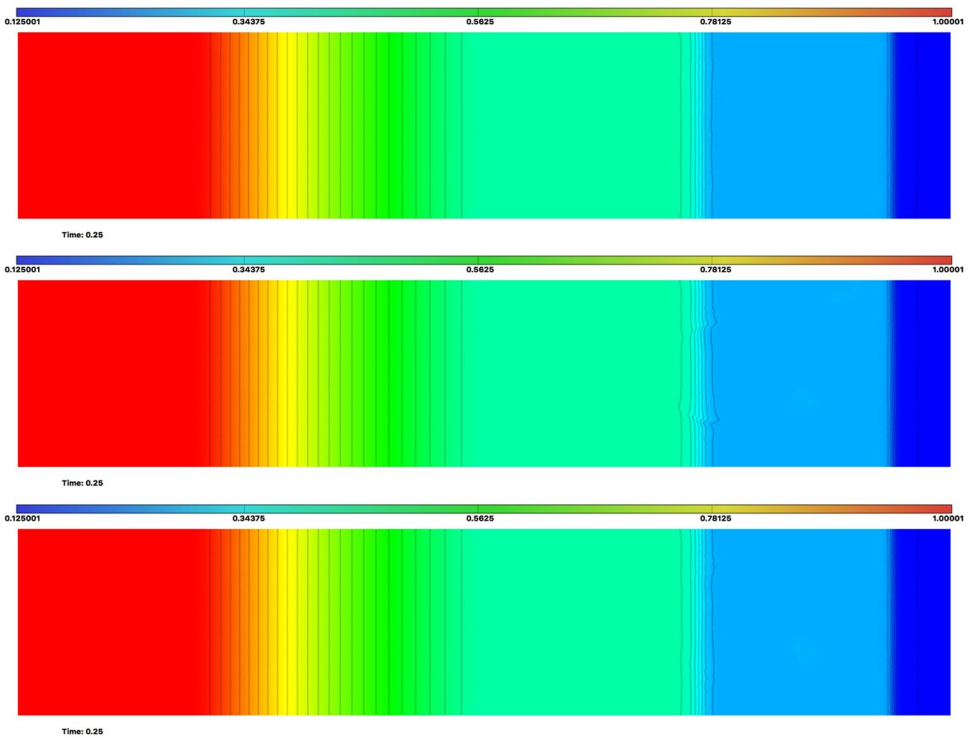

Fig. 8 Sod's shock tube problem with a rotating region. From top to bottom, density iso-values and iso-lines at non-dimensioned time $t = 0.25$ for the fixed mesh, the moving mesh with linear interpolation, and the moving mesh with the P^1-exact conservative interpolation

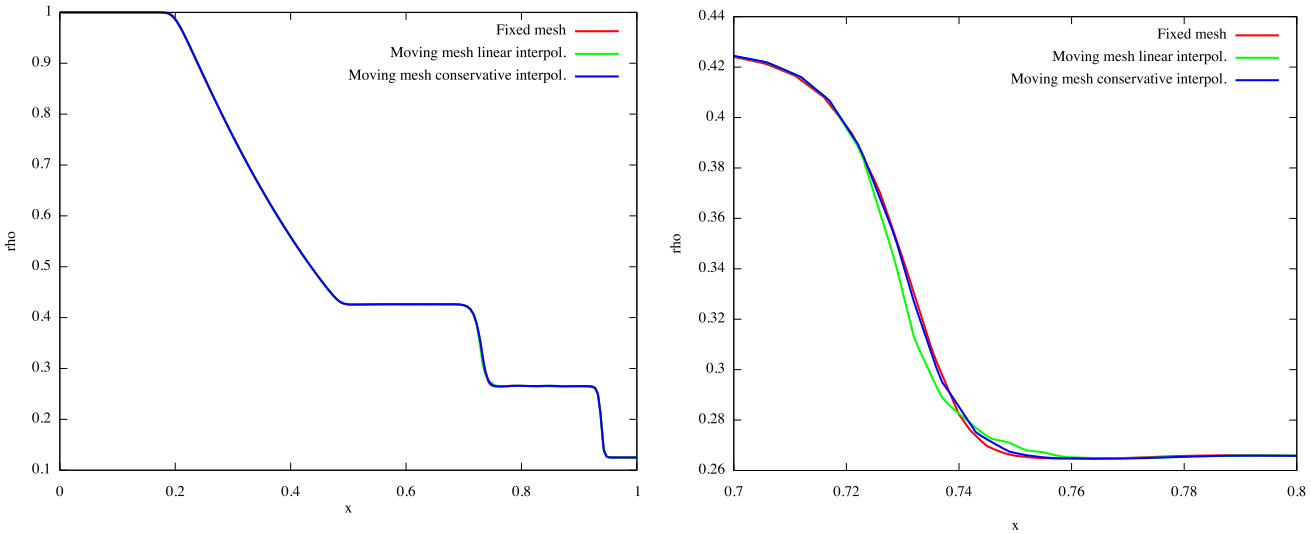

Fig. 9 Sod's shock tube problem with a rotating region. Extraction of the density profile along the line $y = 0.0483256$ at non-dimensioned time $t = 0.25$. Bottom, zoom on the contact discontinuity region

vertical face at $x = 0.2$ is mobile, and all the other faces are fixed, which corresponds to $l_0 = 1$ and $u_0 = 0.2$. The piston rigidity is set to $k_p = 10^7\ \mathrm{N\,m^{-1}}$, and a variety of piston masses (and thus frequencies) are considered: $m_p = 4$ kg, 10 kg, 100 kg and 1000 kg, corresponding, respectively, to frequencies $f_p = 252$ Hz, 159 Hz, 50 Hz and 16 Hz. The other parameters are: $\rho_0 = 1.1615\ \mathrm{kg\,m^{-3}}$, $p_0 = 10^5$ Pa, $\mathbf{u}_0 = 0\ \mathrm{m\,s^{-1}}$.[3] Slipping conditions are imposed on all the boundary faces (fixed and mobile). From a moving mesh

[3] For more details on the description of the test in [37], see the source code referred in the paper.

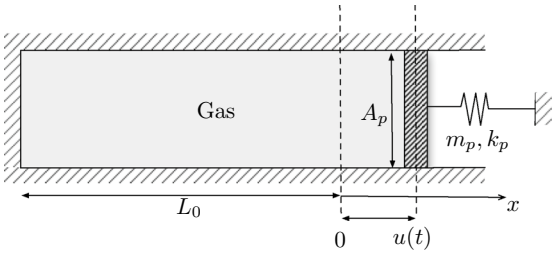

Fig. 10 Piston test case. 2D representation of the problem

Table 4 Piston test case. Natural frequencies and periods for the different masses used

m_p (kg)	4	10	100	1000
f_p (Hz)	252	159	50	16
T_p (s)	0.004	0.00628	0.0199	0.0628

point of view, it is essential to allow tangential movement of the vertices on the moving faces of the box.

The quantities of interest are the displacement and the pressure of the piston. For the pressure, we consider the pressure at its center of gravity. These two quantities are plotted in Fig. 11. Regarding the piston movement, it is periodic, as expected, with a period $T_p = 1/f_p$ (see Table 4), the amplitude of the oscillations being damped due to FSI effects. These curves are similar to the results of [37] for the cases $m_p = 10$ kg, 100 kg and 1000 kg. To highlight the FSI effect, we added a case with a smaller mass $m_p = 4$ kg, where the reduction of the amplitude of the oscillations is even clearer. As concerns the piston pressure, the results also correspond to [37]. We ran the case with several mesh sizes (from 200,000 to one million vertices) and several temporal schemes, and all results are consistent. The pressure fields at a time around 2/5 of the final time are shown in Fig. 12 for the four masses considered. We can see that the 1D structure of the solution is well preserved.

7 Some application examples

Finally, we analyze the behavior of our strategy on two more complex, industrial-like, examples of simulations in three dimensions. These examples are challenging due both to the size of the meshes considered, and to the large displacement of the geometries. Note that a linear interpolation was used after edge/face swaps as no shocks are present in the flow-field, and it does not alter the conclusions of these examples.

7.1 Two F117 aircraft crossing flight paths

The first example models two notional F117 aircraft with crossing flight paths. This problem illustrates well the efficiency of the connectivity-change moving mesh algorithm in handling large displacements of complex geometries without global remeshing. When both aircraft cross each other, the mesh deformation encounters a large shearing due to the opposite flight directions. The connectivity-change mesh deformation algorithm easily handles this complex displacement thanks to the mesh local reconnection. Therefore, the mesh quality remains very good during the whole displacement, without any remeshing step.

As concerns the fluid simulation, the aircraft are moved with an imposed motion of translation and rotation at a speed of Mach 0.4, in an initially inert uniform fluid: at $t = 0$ the speed of the air is null everywhere. Transmitting boundary conditions are used on the sides of the surrounding box, while slipping conditions are imposed on the two F117 bodies. After a short phase of initialization, the flow is established when the two F117s pass each other, and the density fields around the aircraft and in their wake interact. Acoustic waves are created in front of the F117s due to the instantaneous setting in motion of the aircraft. This is not realistic, but our aim was to validate our moving mesh approach rather than run a physically realistic simulation. In Fig. 13, a zoom on the geometry of the two aircraft is shown. In Fig. 14, we show both the moving mesh aspect of the simulation and the flow solution at different time steps.

The mesh is composed of 585,000 vertices and (initially) 3.5 million tetrahedra. The whole simulation requires 22 elasticity solutions and 1600 optimization steps for a total of 2,500,000 swaps. The final mesh average quality of 1.4 is excellent and we notice that 99.8% of the elements have a quality smaller than 2 and only 52 elements have a quality higher than 5.

This simulation was run in a reasonable time: 18 h were necessary to do 39,000 time steps on a machine with 20 hyperthreaded i7 cores at 2.5 GHz. Very few elasticity solutions are requires compared to the number of solver time steps, and the total time required for the solution of the elasticity problems is only 25 min, which represents only 2.3% of the total time. The good quality of the mesh ensures an acceptable solution accuracy throughout the simulation. The optimization steps (swapping and smoothing) only take 25 min. As for the impact of the swaps, it is difficult to evaluate, since the simulation cannot be run without them. One can notice that on average, only 66 swaps per solver time step were performed,

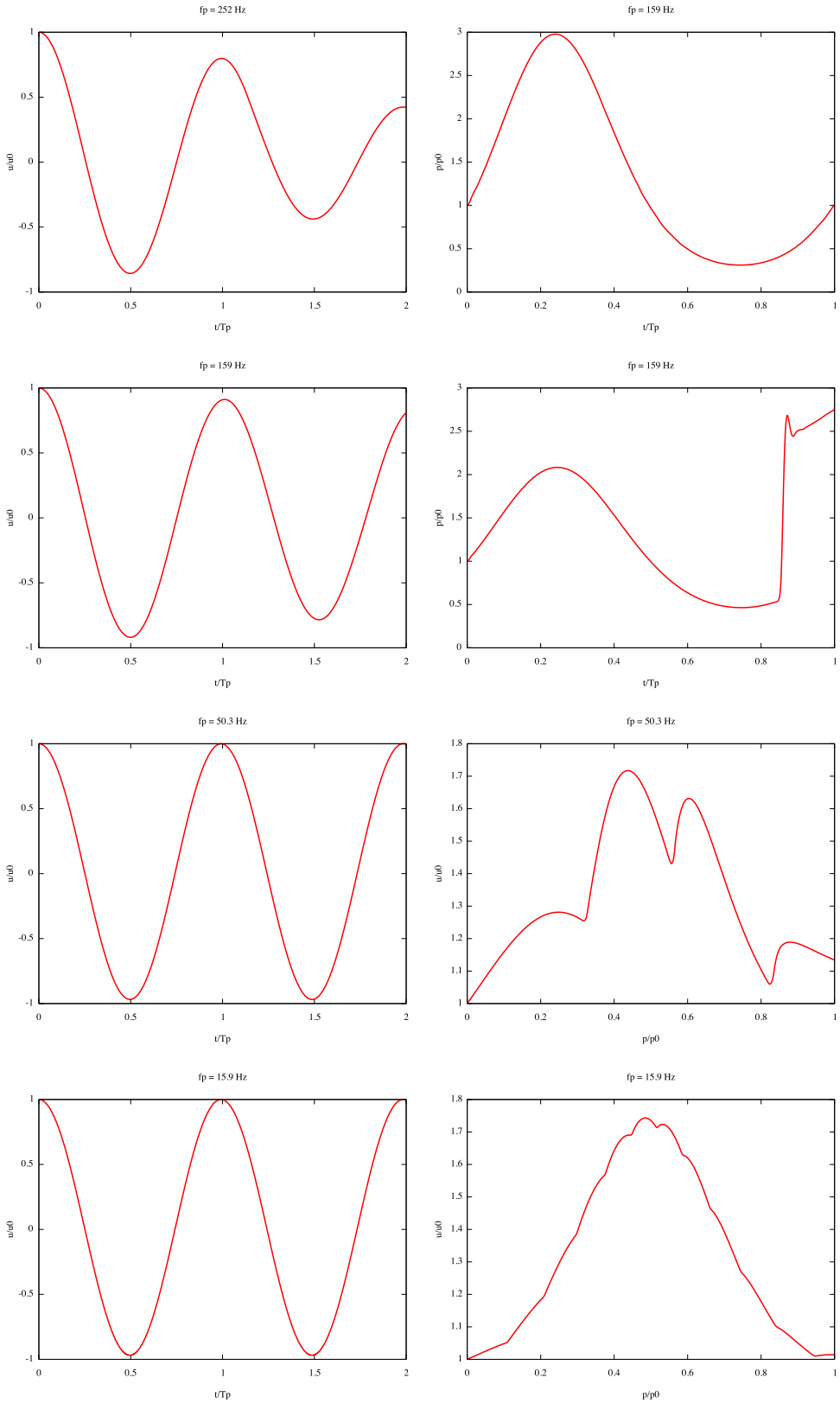

Fig. 11 Piston test case. Displacement (right) and piston pressure (left)

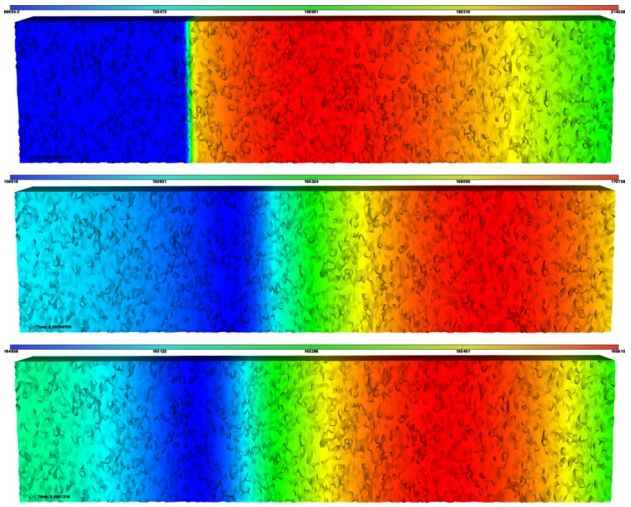

Fig. 12 Piston test case. Pressure fields at 2/5 of the final time. $m = 10$ and $t = 0.0025$ (top), $m = 100$ and $t = 0.0079$ (center), $m = 1000$ and $t = 0.025$ (bottom). The mobile piston is on the right

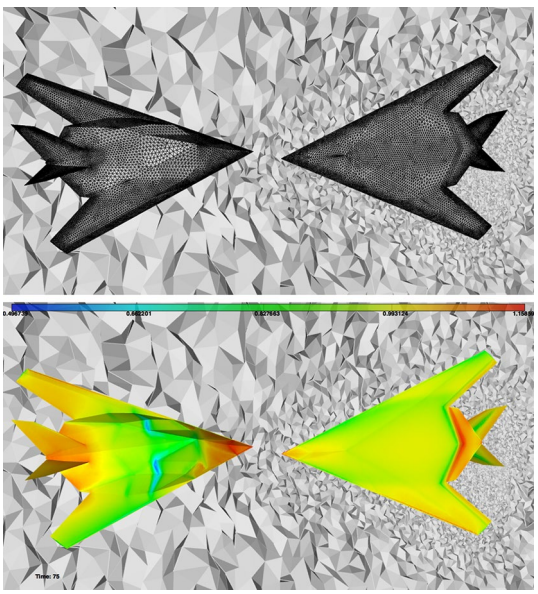

Fig. 13 Test case of the two F117s. Zoom on the aircraft mesh (top) and solution (bottom) just before they pass each other

which is less than 0.0002% of the number of elements of the mesh, and so is likely to have little influence on the solution.

7.2 Ejected cabin door

The last example is the FSI simulation of the ejection of the door of an over-pressurized aircraft cabin. This test case has been proposed by aircraft designers, and the aim is to evaluate when the door hinge will yield under cabin pressure. Generally, such experiments are done in a hangar and numerical simulations must be able to predict if the door will impact the observation area.

From a purely moving mesh point of view, the difficulty is that the geometry is anisotropic and rolls over the elements while progressing inside a uniform mesh composed of 965,000 vertices. Another difficulty lies at the beginning of the movement when the door is ejected from its frame. The gap between the door and the frame is very small, and the mesh is sheared in the interstice; see Fig. 15. However, no remeshing is needed. The connectivity-change moving mesh strategy is able to get rid of these skewed elements quite rapidly to finally achieve an excellent quality throughout the door displacement.

From the FSI point of view, we present a simplified version of the case—and thus probably not very representative of the actual physics involved. However, we make sure that the solution is physically coherent with the simplified initial conditions. At time $t = 0$, the volume is divided into two parts: outside and inside the cabin; see Fig. 15. The inside of the cabin and the outside are not insulated, so the pressurized air leaks out of the cabin. The small volume surrounding the door is considered initially as being outside the cabin. The outside of the cabin is initialized to classic atmospheric values at 10,000 m ($\rho = 0.44\ \mathrm{kg\,m^{-3}}, \mathbf{u} = (0,0,0)\ \mathrm{m\,s^{-1}}, p = 20\ \mathrm{kPa}$). At this altitude, the air of the cabin should be pressurized to simulate an altitude of 2500 m ($\rho = 0.96\ \mathrm{kg\,m^{-3}}, \mathbf{u} = (0,0,0)\ \mathrm{m\,s^{-1}}, p = 75\ \mathrm{kPa}$). To simulate a blast, those values are multiplied by 10 inside the cabin ($\rho = 9.6\ \mathrm{kg\,m^{-3}}, \mathbf{u} = (0,0,0)\ \mathrm{m\,s^{-1}}, p = 750\ \mathrm{kPa}$). The mass of the door is set to 100 kg.

Snapshots of the solution at different time steps are shown in Fig. 16. We can see features similar to mach diamonds at the rear of the door, due to the high speed of the door. As concerns the door movement, the door is blown away from the cabin as expected. However, it acquires a slow rotation movement. For the time frame considered ($T_{end} = 0.2s$), gravity has a negligible impact on the trajectory. The mesh is composed of 965,000 vertices and 5.6 millions of tetrahedra. Due to the difficult part where the door is moving in its frame, the whole simulation requires 29 elasticity solutions and 540 optimization steps for a total of 875,000 swaps. The final mesh average quality of 1.33 is excellent and we notice that 99.95% of the elements have a quality smaller than 2 and only 1 element has a quality higher than 5.

The total time of the simulation is 2 h on a machine with 20 hyperthreaded i7 cores at 2.5 GHz, for 1122 solver time steps, including 43 min of elasticity solution. The number of elasticity solutions is still small compared to the number of time steps (29 elasticity solutions and 1122 solver time steps), and elasticity solutions do not account

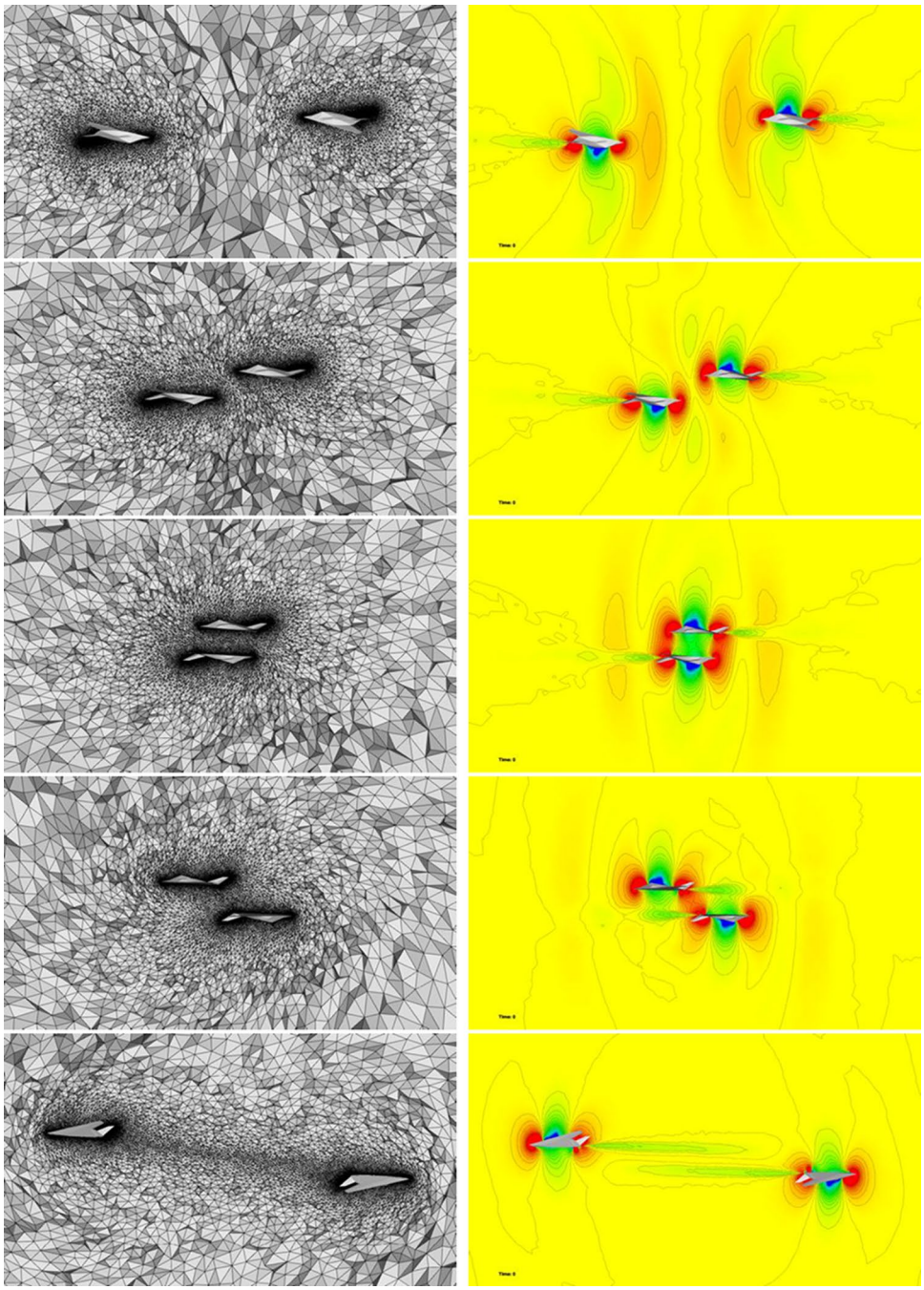

Fig. 14 Test case of the two F117s. Snapshots of the moving geometries and the mesh (left) and density (right)

for more than a third of the total time. Note that the a priori unpredictable trajectory of FSI problems generally results in a slightly higher number of elasticity steps than with analytic imposed motion. The optimization step only takes 3 min. Once again, only an average of 0.0001% swaps per tetrahedra and solver time step are performed, which is not a lot.

8 Conclusion

A strategy to run complex three-dimensional simulations with moving geometries has been presented in this paper. This strategy lies first on a robust connectivity-change moving mesh algorithm, which couples an elasticity-like mesh deformation method and mesh optimizations.

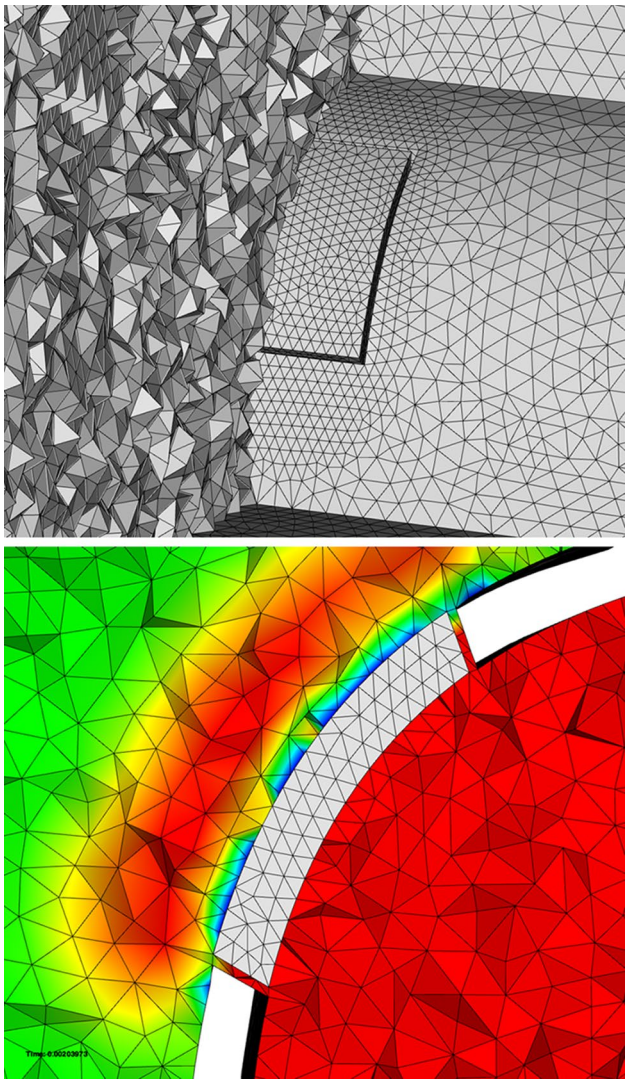

Fig. 15 Test case of the door ejection. Surface mesh and cut plane just after the beginning of the simulation. Note that the gap between the door and its frame is very small, and meshed with only one layer of elements, which increases the difficulty of the moving mesh problem

In particular, a reduced number of elasticity solutions improves the efficiency of the algorithm, while edge/face swapping makes it possible to deal with the strong shearing of the mesh that occurs when the geometries undergo large displacement.

To run CFD simulations on such moving meshes, a finite volumes flow solver for the compressible Euler equations has been extended to the ALE framework. As regards the temporal accuracy, the SSPRK schemes considered are based on the strict application of the discrete geometric conservation law (DGCL), which is supposed to preserve the accuracy of the schemes. The displacement of the geometries can be either imposed a priori, or governed by a 6-DOF fluid–structure coupling.

The goal of the paper was to demonstrate that this strategy allows us to run complex three-dimensional simulations. Challenging examples of such simulations, were presented and analyzed, with imposed motion or fully FSI. Despite the large displacement of the geometries, no global remeshing was necessary to perform the simulations, while preserving a good mesh quality. The unsteady solutions of these examples are physically coherent, and the total CPU time of the simulations is reasonable. The use of high-order Runge–Kutta schemes was shown to help reduce the cost of the simulation. Whereas the handling of the moving mesh usually takes up a large proportion of the simulation total CPU time, our connectivity-change moving mesh algorithm only accounts for a small fraction of that CPU time, most of the time being spent on actually solving the equations.

Some issues still need to be addressed. A thorough analysis of the impact of swaps on the solution accuracy needs to be carried out. We have shown that a conservative interpolation method improves significantly the accuracy of the solution in the presence of discontinuities. A comparison of this approach with the one described in [51] would probably be revealing. The elasticity time step is currently constant and fixed a priori for the whole simulation. The efficiency of the moving mesh part would be improved if this time step could be automatically adapted to the movement of the geometries. Finding such an optimal time step is harder than it seems. The case of non rigid bodies also has to be addressed: the moving mesh algorithm can handle deformable bodies [1, 7], but the FSI coupling would be much more complex.

Finally, this entire strategy was devised to fit within our metric-based mesh adaptation framework [42]. An unsteady version of the adaptation algorithm already exists [3]. It was extended to the case of moving meshes in 3D recently [8], and a goal-oriented version of the process is developed.

9 Software used

The solver described in this paper and used to run all the simulations is `Wolf`. The meshes were generated using `GHS3d` [27] and `Feflo.a` [43]. The visualization of the meshes and solutions was done with `Vizir` [41].

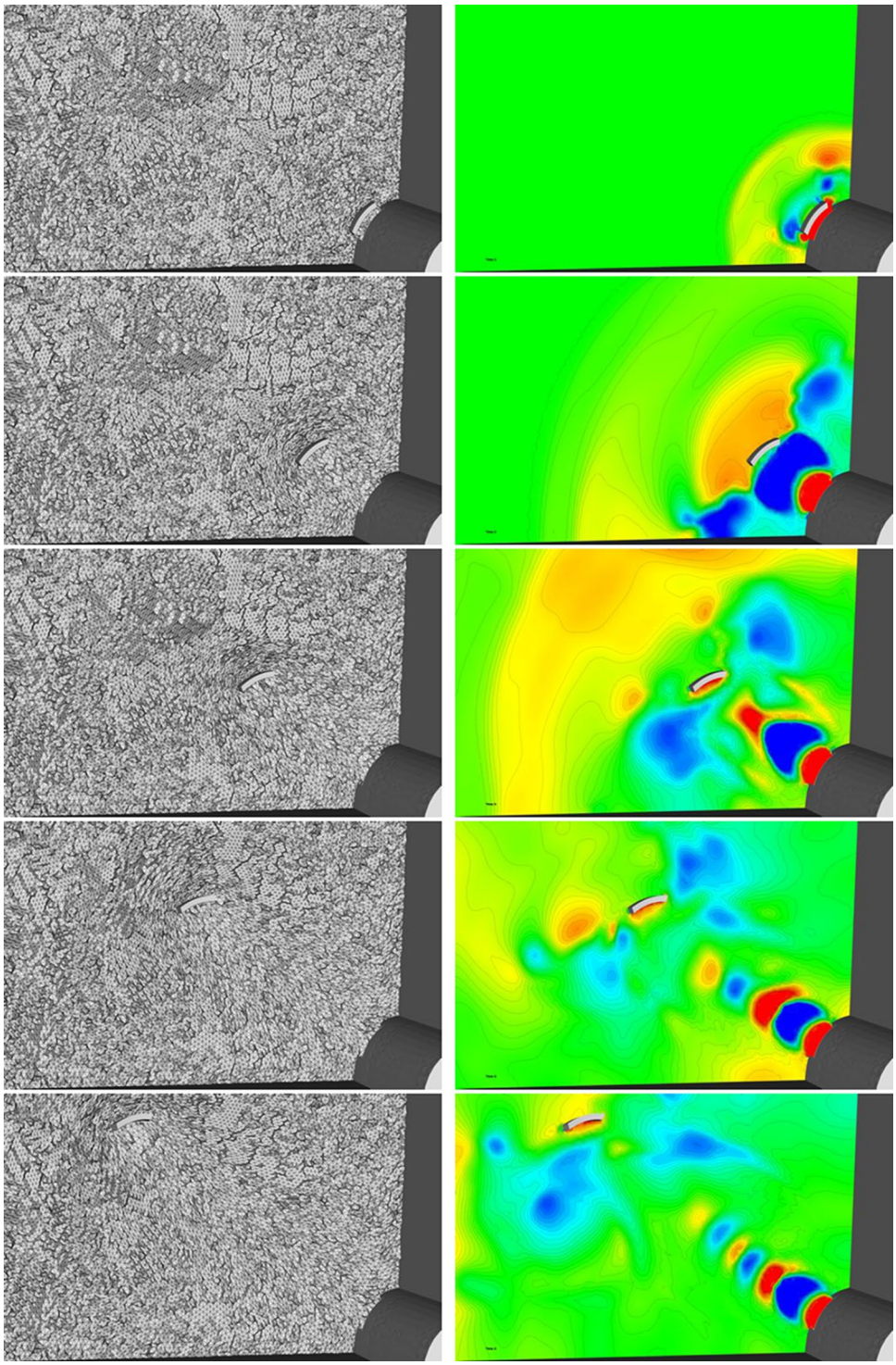

Fig. 16 Test case of the door ejection. Snapshots of the moving geometry and the mesh (left) and pressure (right)

Acknowledgements This work was partially funded by the Airbus Group Foundation. The authors also gratefully acknowledge the support of EPRSC Grant EP/R029423/1.

References

1. Alauzet F (2014) A changing-topology moving mesh technique for large displacements. Eng Comput 30(2):175–200
2. Alauzet F (2016) A parallel matrix-free conservative solution interpolation on unstructured tetrahedral meshes. Comput Methods Appl Mech Eng 299:116–142
3. Alauzet F, Frey P, George P, Mohammadi B (2007) 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. J Comput Phys 222:592–623
4. Alauzet F, Loseille A (2009) On the use of space filling curves for parallel anisotropic mesh adaptation. In: Proceedings of the 18th international meshing roundtable. Springer, pp 337–357
5. Alauzet F, Mehrenberger M (2010) P1-conservative solution interpolation on unstructured triangular meshes. Int J Numer Methods Eng 84(13):1552–1588
6. Baker T, Cavallo P (1999) Dynamic adaptation for deforming tetrahedral meshes. AIAA J 19:2699–3253
7. Barral N, Luke E, Alauzet F (2014) Two mesh deformation methods coupled with a changing-connectivity moving mesh method for CFD applications. In: Proceedings of the 23th international meshing roundtable, vol 82. Elsevier, pp 213–227
8. Barral N, Olivier G, Alauzet F (2017) Time-accurate anisotropic mesh adaptation for three-dimensional time-dependent problems with body-fitted moving geometries. J Comput Phys 331(Supplement C):157–187
9. Batina J (1990) Unsteady Euler airfoil solutions using unstructured dynamic meshes. AIAA J 28(8):1381–1388
10. Batten P, Clarke N, Lambert C, Causon D (1997) On the choice of wavespeeds for the HLLC Riemann solver. SIAM J Sci Comput 18(6):1553–1570
11. Baum J, Luo H, Löhner R (1994) A new ALE adaptive unstructured methodology for the simulation of moving bodies. In: 32th AIAA aerospace sciences meeting. AIAA Paper 1994-0414, Reno, NV, USA
12. Benek J, Buning P, Steger J (1985) A 3D chimera grid embedding technique. In: 7th AIAA computational fluid dynamics conference. AIAA Paper 1985-1523, Cincinnati, OH, USA
13. Boer AD, van der Schoot M, Bijl H (2007) Mesh deformation based on radial basis function interpolation. Comput Struct 85:784–795
14. Bruchon J, Digonnet H, Coupez T (2009) Using a signed distance function for the simulation of metal forming process: formulation of the contact condition and mesh adaptation. From Lagrangian approach to an Eulerian approach. Int J Numer Methods Eng 78(8):980–1008
15. Chacón L, Delzanno G, Finn J (2011) Robust, multidimensional mesh-motion based on Monge–Kantorovich equidistribution. J Comput Phys 230(1):87–103
16. Compere G, Remacle JF, Jansson J, Hoffman J (2010) A mesh adaptation framework for dealing with large deforming meshes. Int J Numer Methods Eng 82(7):843–867
17. Cournède PH, Koobus B, Dervieux A (2006) Positivity statements for a Mixed-Element-Volume scheme on fixed and moving grids. Eur J Comput Mech 15(7–8):767–798
18. Debiez C, Dervieux A (2000) Mixed-Element-Volume MUSCL methods with weak viscosity for steady and unsteady flow calculations. Comput Fluids 29:89–118
19. Degand C, Farhat C (2002) A three-dimensional torsional spring analogy method for unstructured dynamic meshes. Comput Struct 80(3–4):305–316
20. Dobrzynski C, Frey P (2008) Anisotropic Delaunay mesh adaptation for unsteady simulations. In: Proceedings of the 17th international meshing roundtable. Springer, pp 177–194
21. Donea J, Giuliani S, Halleux JP (1982) An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid–structure interactions. Comput Methods Appl Mech Eng 33(1):689–723
22. Etienne S, Garon A, Pelletier D (2009) Perspective on the geometric conservation law and finite element methods for ALE simulations of incompressible flow. J Comput Phys 228(7):2313–2333
23. Etienne S, Garon A, Pelletier D, Cameron C (2010) Philiadium gregarium versus Aurelia aurita: on propulsion propulsion of jellyfish. In: 48th AIAA aerospace sciences meeting. AIAA Paper 2010-1444, Orlando, FL, USA
24. Farhat C, Geuzaine P, Grandmont C (2001) The discrete geometric conservation law and the nonlinear stability of ALE schemes for the solution of flow problems on moving grids. J Comp Phys 174(2):669–694
25. Formaggia L, Nobile F (2004) Stability analysis of second-order time accurate schemes for ALE-FEM. Comput Methods Appl Mech Eng 193(39–41):4097–4116
26. Frey P, George P (2008) Mesh generation. Application to finite elements, 2nd edn. ISTE Ltd and John Wiley & Sons, New York
27. George P, Borouchaki H (2003) "Ultimate" robustness in meshing an arbitrary polyhedron. Int J Numer Methods Eng 58(7):1061–1089
28. Guardone A, Isola D, Quaranta G (2011) Arbitrary Lagrangian Eulerian formulation for two-dimensional flows using dynamic meshes with edge swapping. J Comput Phys 230(20):7706–7722
29. Hassan O, Sørensen K, Morgan K, Weatherill NP (2007) A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing. Int J Numer Methods Fluids 53(8):1243–1266
30. Hay A, Yu K, Etienne S, Garon A, Pelletier D (2014) High-order temporal accuracy for 3D finite-element ALE flow simulations. Comput Fluids 100:204–217
31. Hirt CW, Amsden AA, Cook JL (1974) An arbitrary Lagrangian–Eulerian computing method for all flow speeds. J Comput Phys 14(3):227–253
32. Huang W, Russell RD (2010) Adaptive moving mesh methods, vol. 174. Springer Science & Business Media
33. Hughes TJR, Liu WK, Zimmermann TK (1981) Lagrangian–Eulerian finite element formulation for incompressible viscous flows. Comput Methods Appl Mech Eng 29(3):329–349
34. Kitamura K, Fujimoto K, Shima E, Kuzuu K, Wang ZJ (2011) Validation of arbitrary unstructured CFD code for aerodynamic analyses. Trans Jpn Soc Aeronaut Space Sci 53(182):311–319
35. Koobus B, Farhat C (1999) Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes. Comput Methods Appl Mech Eng 170(1–2):103–129
36. Kucharik M, Shashkov M (2008) Extension of efficient, swept-integration-based conservative method for meshes with changing connectivity. Int J Numer Methods Fluids 56(8):1359–1365
37. Lefrançois E, Boufflet J (2010) An introduction to fluid–structure interaction: application to the piston problem. SIAM Rev 52(4):747–767
38. Lesoinne M, Farhat C (1996) Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. Comput Methods Appl Mech Eng 134(1–2):71–90
39. Löhner R (2001) Applied CFD techniques. An introduction based on finite element methods. Wiley, New York
40. Löhner R, Yang C (1996) Improved ALE mesh velocities for moving bodies. Commun Numer Methods Eng 12(10):599–608
41. Loseille A, Guillard H, Loyer A (2016) An introduction to Vizir: an interactive mesh visualization and modification software. EOCOE, Rome, Italy

42. Loseille A, Alauzet F (2011) Continuous mesh framework. Part I: well-posed continuous interpolation error. SIAM J Numer Anal 49(1):38–60
43. Loseille A, Menier V (2013) Serial and parallel mesh modification through a unique cavity-based primitive. In: Proceedings of the 22th international meshing roundtable. Springer, Orlando, pp 541–558
44. Luke E, Collins E, Blades E (2012) A fast mesh deformation method using explicit interpolation. J Comput Phys 231:586–601
45. Maréchal L (2018) Handling unstructured meshes in multi-threaded environments with the help of hilbert renumbering and dynamic scheduling. Parallel Comput (**under review**)
46. Mavriplis D, Yang Z (2006) Construction of the discrete geometric conservation law for high-order time accurate simulations on dynamic meshes. J Comput Phys 213(2):557–573
47. Murman S, Aftosmis M, Berger M (2003) Simulation of 6-DOF motion with cartesian method. In: 41th AIAA aerospace sciences meeting. AIAAsps2003-1246, Reno, NV, USA
48. Nkonga B (2000) On the conservative and accurate CFD approximations for moving meshes and moving boundaries. Comput Methods Appl Mech Eng 190(13):1801–1825
49. Nkonga B, Guillard H (1994) Godunov type method on non-structured meshes for three-dimensional moving boundary problems. Comput Methods Appl Mech Eng 113(1–2):183–204
50. Olivier G (2011) Anisotropic metric-based mesh adaptation for unsteady CFD simulations involving moving geometries. Ph.D. thesis, Université Pierre et Marie Curie, Paris VI, Paris, France
51. Olivier G, Alauzet F (2011) A new changing-topology ALE scheme for moving mesh unsteady simulations. In: 49th AIAA aerospace sciences meeting. AIAA Paper 2011-0474, Orlando, FL, USA
52. Roe P (1981) Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 43:357–372
53. Shu C, Osher S (1988) Efficient implementation of essentially non-oscillatory shock-capturing schemes. J Comput Phys 77:439–471
54. Sod G (1978) A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. J Comput Phys 27:1–31
55. Spiteri R, Ruuth S (2002) A new class of optimal high-order strong-stability-preserving time discretization methods. SIAM J Numer Anal 40(2):469–491
56. Thompson JF, Soni BK, Weatherill NP (1998) Handbook of grid generation. CRC Press, Boca Raton
57. Yang Z, Mavriplis D (2005) Unstructured dynamic meshes with higher-order time integration schemes for the unsteady Navier–Stokes equations. In: 41th AIAA aerospace sciences meeting. AIAA Paper 2005-1222, Reno, NV, USA
58. Yang Z, Mavriplis D (2007) Higher-order time integration schemes for aeroelastic applications on unstructured meshes. AIAA J 45(1):138–150