V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

A MESH TOPOLOGY CHANGE *ALE* FRAMEWORK FOR EFFICIENT BODY LARGE-DISPLACEMENT ADAPTIVE SIMULATIONS

G. Olivier*, F. Alauzet*

*INRIA GAMMA project,
11, Domaine de Voluceau - Rocquencourt B.P.105, 78153 Le Chesnay, France
e-mail: geraldine.olivier@inria.fr

Key words: Unsteady flow, moving mesh, metric-based mesh adaptation, ALE, changing topology, GCL

Abstract. *This paper presents two new methods to couple moving mesh ALE simulations with highly anisotropic adaptation. The first one allows to extend the classical ALE framework often used for simulation involving moving geometries to changing-connectivity meshes. The second is an extension of the mesh adaptation fixed-point algorithm for unsteady flows to the context of ALE simulations. Three CFD simulations in two dimensions illustrate these new advances.*

1 INTRODUCTION

The main goal of this paper is to perform anisotropic adaptive Arbitrary-Lagrangian-Eulerian (*ALE*) simulations. To be able to do this, we adopted a twofold strategy. First, we worked on a new variable-topology *ALE* scheme enabling the use of the swap operation in a fully *ALE* manner, thus relaxing the strong fixed-topology constraint imposed by the classical *ALE* framework. Secondly, we improved some existing moving mesh techniques in terms of *CPU* time and efficiency. Finally, we extended the so-called fixed-point adaptation algorithm to moving mesh simulations. This paper is built as follows: in Section 2, the modeled problem is described and in Section 3, the spatial numerical method is detailed; Section 4, *ALE* specific issues regarding numerical time integration are discussed, and a new variable-topology *ALE* formulation is presented. Section 5, the enhanced moving mesh techniques are explained. Finally, in Section 6, the extension of the so-called fixed-point mesh adaptation algorithm to moving mesh simulations is described. Numerical results are shown in Sections 4 and 6.

2 MODELLING OF THE PROBLEM

2.1 The Euler equations in the Arbitrary-Lagrangian-Eulerian formulation

The basic idea of the Arbitrary-Lagrangian-Eulerian method is that there is a priori no reason for which the movement of the vertices should remain either fix (Eulerian de-

scription) or should follow exactly the fluid particles (Lagrangian formulation). On the contrary, the vertices movement and the particles movement can be totally decorrelated. To do this, an intermediate reference configuration $\Omega_\xi = \Omega_\xi(t)$ is introduced. The important thing is that the new reference configuration can evolve with time, but not necessarily by following the particles like in the purely Eulerian case. If $\boldsymbol{\phi}_t : \Omega_\xi(t) \rightarrow \Omega_x(t)$ is the mapping between the *ALE* reference configuration and the current domain at t, the velocity of the mesh is given by $\boldsymbol{w}(\xi,t) = \dfrac{\partial \boldsymbol{\phi}_t}{\partial t}|_\xi(\xi,t)$, which represents the instantaneous velocity of the points of the domain. Assuming that the gas is perfect, inviscid and that there is no thermal diffusion, the compressible Euler equations for a Newtonian fluid in the *ALE* framework read, for any arbitrary closed volume $C(t)$ of boundary $\partial C(t)$[1]:

$$\begin{aligned} & \frac{\partial}{\partial t}|_\xi \left(\int_{C(t)} \mathbf{W}\, \mathrm{d}\mathbf{x} \right) && + \int_{\partial C} \left(\mathcal{F}(\mathbf{W}) - \mathbf{W} \otimes \boldsymbol{w} \right) \cdot \mathbf{n}\, \mathrm{d}\boldsymbol{s} && = \int_{C(t)} \mathbf{F}_{ext}\, \mathrm{d}\mathbf{x} \\ \Longleftrightarrow\ & \frac{\partial}{\partial t}|_\xi \left(\int_{C(t)} \mathbf{W}\, \mathrm{d}\mathbf{x} \right) && + \int_{\partial C} \left(\mathcal{F}(\mathbf{W}) \cdot \mathbf{n} - \mathbf{W}\ (\boldsymbol{w} \cdot \mathbf{n}) \right)\, \mathrm{d}\boldsymbol{s} && = \int_{C(t)} \mathbf{F}_{ext}\, \mathrm{d}\mathbf{x} \end{aligned} \quad (1)$$

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

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

2.2 Movement of the geometries

In this work, the *ALE* formulation is used to perform computations involving bodies interacting with a surrounding fluid. Bodies are assumed to be **rigid**, of constant mass and **homogenous**, *i.e.* their mass is uniformly distributed in their volume. Our bodies will never break into different parts.

In two dimensions, each rigid body B is fully described by:

- $\partial B = \boldsymbol{s}(v)$ the parametrized surface defining its boundary
- $\mathbf{n} = \mathbf{n}(\boldsymbol{s}(v))$ the **inward** normal to the object boundary ∂B
- $\mathbf{x_G} = \mathbf{x_G}(t) = (x(t), y(t))$ the position of its gravity center

[1] superscript ^{T} is used for the transposition operation on a vector, and $^{-T}$ holds for $\left((\,\cdot\,)^{-1} \right)^T$

- $\boldsymbol{\theta} = \theta(t)\mathbf{e}_z$ its angular displacement vector: its norm θ represents the angular displacement in the two dimensional plane $(\mathbf{e}_x, \mathbf{e}_y)$
- $\boldsymbol{\omega} = \dfrac{\mathrm{d}\theta}{\mathrm{d}t}\mathbf{e}_z$ its angular speed vector
- m its mass assumed to be constant
- $J_{(Gz)}$ its moment of inertia along axe $\mathbf{e}_z$ computed at G.

In two dimensions, the Euler equations for solid dynamics simplify to:

$$\left\{\begin{array}{lll} m\dfrac{\mathrm{d}^2x}{\mathrm{d}t^2} & = \mathbf{F}_{ext}\cdot\mathbf{e}_x & = \left(\displaystyle\int_{\partial B} p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_x + m\mathbf{g}\cdot\mathbf{e}_x \\ m\dfrac{\mathrm{d}^2y}{\mathrm{d}t^2} & = \mathbf{F}_{ext}\cdot\mathbf{e}_y & = \left(\displaystyle\int_{\partial B} p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_y + m\mathbf{g}\cdot\mathbf{e}_y \\ J_{(Gz)}\dfrac{\mathrm{d}^2\theta}{\mathrm{d}t^2} & = \mathbf{M}_G\left(\mathbf{F}_{ext}\right)\cdot\mathbf{e}_z & = \left(\displaystyle\int_{\partial B} (\boldsymbol{s}-\mathbf{x_G})\wedge p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_z \end{array}\right. \tag{2}$$

and the kinetic moment of the gravity forces is zero when computed at point G. If one point A of the object must remain fixed, the moment equation must be rewritten at fixed-point A and, as $\frac{\mathrm{d}x_A}{\mathrm{d}t} = 0$ and $\frac{\mathrm{d}y_A}{\mathrm{d}t} = 0$, the system is reduced to only one scalar equation:

$$J_{(Az)}\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = \mathbf{M}_A\left(\mathbf{F}_{ext}\right)\cdot\mathbf{e}_z = \left(\int_{\partial B}(\boldsymbol{s}-\mathbf{x_A})\wedge p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_z + m\left(\int_{\partial B}(\boldsymbol{s}-\mathbf{x_A})\wedge\mathbf{g}\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_z$$

with $\mathbf{M}_A$ the kinetic moment of the external forces applied on ∂B computed at point A and $J_{(Az)}$ the moment of inertia of the object related to axe (Az). According to Huygens theorem, we have:

$$J_{(Az)} = J_{(Gz)} + m\left\|\mathbf{GA}\right\|^2$$

3 SPATIAL DISCRETIZATION FOR THE FIXED-TOPOLOGY CASE

Semi-discretization. Domain Ω is discretized by a tetrahedral unstructured mesh $\mathcal{H}$. The vertex-centered Finite Volume formulation consists in associating with each vertex P_i of the mesh and at each time t a control volume or Finite Volume cell, denoted $C_i(t)$. 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 (bi-segments in two dimensions). The normal flux $\mathbf{F}_{ij}(t)$ along each cell interface is taken constant (not in time but in space), just like the solution $\mathbf{W}_{ij}$ on the interface.

Rewriting System (1) for $C(t) = C_i(t)$, we get the following semi-discretization at P_i:

$$\begin{aligned}
&\frac{\partial}{\partial t}|_\xi \left(\int_{C_i(t)} \mathbf{W}_i(t)\, \mathrm{d}\mathbf{x} \right) + \sum_{P_j \in V_i} \int_{\partial C_{ij}(t)} \Big(\mathcal{F}(\mathbf{W}_{ij}(t)) \cdot \mathbf{n}_{ij}(t) \; - \; \mathbf{W}_{ij}(t) \left[\boldsymbol{w}_{ij}(t) \cdot \mathbf{n}_{ij}(t) \right] \Big)\, \mathrm{d}\boldsymbol{s} \; = 0 \\
\iff & \frac{\partial \left(|\, C_i(t)\, | \mathbf{W}_i(t) \right)}{\partial t}|_\xi \quad + \sum_{P_j \in V_i} |\, \partial C_{ij}(t)\, | \left(\mathbf{F}_{ij}(t) \; - \; \mathbf{W}_{ij}(t) \sigma_{ij}(t) \right) \; = 0 \\
\iff & \frac{\partial \left(|\, C_i(t)\, | \mathbf{W}_i(t) \right)}{\partial t}|_\xi \quad + \sum_{P_j \in V_i} |\, \partial C_{ij}(t)\, | \, \boldsymbol{\Phi}_{ij} \left(\mathbf{W}_i(t),\, \mathbf{W}_j(t),\, \mathbf{n}_{ij}(t),\, \sigma_{ij}(t) \right) \; = 0
\end{aligned} \tag{3}$$

- $\mathbf{W}_i(t)$ is the mean value of state $\mathbf{W}$ in cell C_i at t
- V_i is the set of all neighboring vertices of P_i
- $\mathbf{n}_{ij}$ 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 $\partial C_{ij}(t)$[2]
- $\sigma_{ij}(t) = \dfrac{1}{|\, \partial C_{ij}(t)\, |} \displaystyle\int_{\partial C_{ij}(t)} \boldsymbol{w}_{ij}(t) \cdot \mathbf{n}_{ij}(t)\, \mathrm{d}\boldsymbol{s}$ is the normal velocity of $\partial C_{ij}(t)$
- $\boldsymbol{\Phi}_{ij} \left(\mathbf{W}_i(t), \mathbf{W}_j(t), \mathbf{n}_{ij}(t), \sigma_{ij}(t) \right) \approx \mathbf{F}_{ij}(t) \; - \; \mathbf{W}_{ij}(t) \sigma_{ij}(t)$ 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 mono-dimensionnaly in the direction normal to the each Finite Volume cell interface. Consequently, the numerical calculation of the flux function $\boldsymbol{\Phi}_{ij}$ at interface ∂C_{ij} can be achieved by the resolution at each time step of a one-dimensional Riemann problem in direction $\mathbf{n}_{ij} = \mathbf{n}$ with initial values $\mathbf{W}_L = \mathbf{W}_i$ on the left of the interface and $\mathbf{W}_R = \mathbf{W}_j$ on the right. The normal speed of the interface is temporarily noted σ for clarity reasons. To this aim, an approximate *HLLC* Riemann solver is used.

HLLC numerical flux. The methodology provided in [4] can be extended to the Euler equations in their *ALE* formulation. The *HLLC* flux is then described by three waves phase velocities:

$$S_L = \min \left(\eta_L - c_L, \tilde{\eta} - \tilde{c} \right) \quad \text{and} \quad S_R = \max \left(\eta_R + c_R, \tilde{\eta} + \tilde{c} \right)$$

$$S_M = \frac{\rho_R \eta_R (S_R - \eta_R) - \rho_L \eta_L (S_L - \eta_L) + p_L - p_R}{\rho_R (S_R - \eta_R) - \rho_L (S_L - \eta_L)}$$

[2]Our convention is that the flux is positive if it goes in the same direction as the normal. Thus, $\mathbf{F}_{ij} > 0$ means that the flux goes from cell C_i to C_j. If $\sigma_{ij} > 0$, geometrical flux $-\mathbf{W}_{ij}(t)\sigma_{ij}$ is negative and therefore oriented from C_j to C_i, which means that cell C_i steals mass from C_j.

and two approximate states:

$$\mathbf{W}_\mathbf{L}^* = \begin{cases} \rho_L^* = \rho_L \dfrac{S_L - \eta_L}{S_L - S_M} \\ p_L^* = p^* = \rho_L \left(\eta_L - S_L\right)\left(\eta_L - S_M\right) + p_L \\ \left(\rho\,\mathbf{u}\right)_L^* = \dfrac{\left(S_L - \eta_L\right)\rho\,\mathbf{u}_L + \left(p^* - p_L\right)\mathbf{n}}{S_L - S_M} \\ \left(\rho\,e\right)_L^* = \dfrac{\left(S_L - \eta_L\right)\rho\,e_L - p_L\eta_L + p^* S_M}{S_L - S_M} \end{cases} \quad \mathbf{W}_\mathbf{R}^* = \begin{cases} \rho_R^* = \rho_R \dfrac{S_R - \eta_R}{S_R - S_M} \\ p_R^* = p^* = \rho_R \left(\eta_R - S_R\right)\left(\eta_R - S_M\right) + p_R \\ \left(\rho\,\mathbf{u}\right)_R^* = \dfrac{\left(S_R - \eta_R\right)\rho\,\mathbf{u}_R + \left(p^* - p_R\right)\mathbf{n}}{S_R - S_M} \\ \left(\rho\,e\right)_R^* = \dfrac{\left(S_R - \eta_R\right)\rho\,e_R - p_R\eta_R + p^* S_M}{S_R - S_M} \end{cases}$$

The *HLLC* flux through the interface is finally given by:

$$\mathbf{\Phi}^{\text{Hllc}}(\mathbf{W}_L, \mathbf{W}_R, \sigma) = \begin{cases} \mathbf{F}_L - \sigma\mathbf{W}_L & \text{if } S_L - \sigma > 0 \\ \mathbf{F}_L^* - \sigma\mathbf{W}_L^* & \text{if } S_L - \sigma \leq 0 < S_M - \sigma \\ \mathbf{F}_R^* - \sigma\mathbf{W}_R^* & \text{if } S_M - \sigma \leq 0 \leq S_R - \sigma \\ \mathbf{F}_R - \sigma\mathbf{W}_R & \text{if } S_R - \sigma < 0 \end{cases}$$

The *HLLC* approximate Riemann solver has the following properties. It automatically: (i) satisfies the entropy inequality, (ii) resolves isolated contacts exactly, (iii) resolves isolated shocks exactly and (iv) preserves positivity.

High-order scheme. The previous formulation reaches at best a first-order accuracy. The *MUSCL* type reconstruction method has been designed to increase the order of accuracy of the scheme. The idea is to use extrapolated values $\mathbf{W}_{ij}$ and $\mathbf{W}_{ji}$ of $\mathbf{W}$ at interface ∂C_{ij} to evaluate the flux. The following approximation is performed:

$$\mathbf{\Phi}_{ij} = \mathbf{\Phi}(\mathbf{W}_{ij}, \mathbf{W}_{ji}, \mathbf{n}_{ij})$$

with $\mathbf{W}_{ij}$ and $\mathbf{W}_{ji}$ linearly interpolated state values on each side of the interface:

$$\begin{cases} \mathbf{W}_{ij} &=& \mathbf{W}_i + \dfrac{1}{2}\left(\nabla\mathbf{W}\right)_{ij} \cdot \overrightarrow{P_iP_j}\,, \\ \mathbf{W}_{ji} &=& \mathbf{W}_j + \dfrac{1}{2}\left(\nabla\mathbf{W}\right)_{ji} \cdot \overrightarrow{P_iP_j}\,, \end{cases} \tag{4}$$

In contrast to the original *MUSCL* approach, the approximate "slopes" $\left(\nabla\mathbf{W}\right)_{ij}$ and $\left(\nabla\mathbf{W}\right)_{ji}$ are defined for each edge using a combination of centered, upwind and nodal gradients.
The centered gradient, which is related to edge $\overrightarrow{P_iP_j}$, is defined as:

$$\left(\nabla\mathbf{W}\right)_{ij}^C \cdot \overrightarrow{P_iP_j} = \mathbf{W}_j - \mathbf{W}_i\,.$$

Upwind and downwind gradients, which are also related to edge $\overrightarrow{P_iP_j}$, are computed using the upstream and downstream tetrahedra associated with edge $\overrightarrow{P_iP_j}$. These tetrahedra are

respectively denoted K_{ij} and K_{ji}. K_{ij} is the unique tetrahedron of the ball of P_i (resp. P_j) the opposite face of which is crossed by the straight line prolongating edge $\overrightarrow{P_iP_j}$, see Figure 1. Upwind and downwind gradients of edge $\overrightarrow{P_iP_j}$ are then defined as:

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

where $\nabla \mathbf{W}_{|K} = \sum_{P \in K} (\nabla \phi_P \otimes \mathbf{W}_P)$ is the $\mathbb{P}^1$-Galerkin gradient on element K. Parametrized nodal gradients are built by introducing the β-scheme:

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

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 $\beta = 1$.

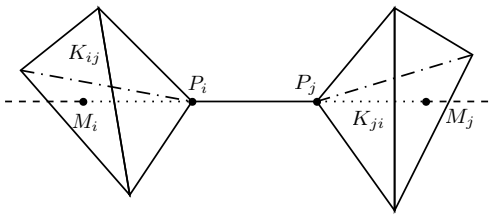

Figure 1: *Downstream K_{ij} and upstream K_{ji} tetrahedra associated with edge $\overrightarrow{P_iP_j}$.*

Numerical dissipation of fourth-order: V4-scheme. The most accurate β-scheme is obtained for $\beta = 1/3$. Indeed, it can be demonstrated that this scheme is third-order for the two-dimensional linear advection problem on structured triangular meshes [6]. In our case, for the non-linear Euler equations on unstructured meshes, a second-order scheme with a fourth-order numerical dissipation is obtained. These high-order gradients are given by:

$$(\nabla \mathbf{W})_{ij}^{V4} = \frac{2}{3}\,(\nabla \mathbf{W})_{ij}^C + \frac{1}{3}\,(\nabla \mathbf{W})_{ij}^U , \quad (\nabla \mathbf{W})_{ji}^{V4} = \frac{2}{3}\,(\nabla \mathbf{W})_{ij}^C + \frac{1}{3}\,(\nabla \mathbf{W})_{ij}^D .$$

Numerical dissipation of sixth-order: V6-scheme. An even less dissipative scheme has been proposed in [6]. It is a more complex linear combination of gradients using centered, upwind and nodal $\mathbb{P}^1$-Galerkin gradients. The nodal $\mathbb{P}^1$-Galerkin gradient of P_i is related to cell C_i and is computed by averaging the gradients of all the tetrahedra having P_i as a vertex:

$$(\nabla \mathbf{W})_i^N = \nabla \mathbf{W}_{|P_i} = \frac{1}{4\,|\,C_i\,|} \sum_{K \in C_i} \left(|\,K\,|\,\nabla \mathbf{W}_{|K}\right).$$

The upwind (resp. downwind) nodal gradients $(\nabla \mathbf{W})_i^{UN} = \nabla \mathbf{W}_{|M_i}$ (resp. $(\nabla \mathbf{W})_j^{DN} = \nabla \mathbf{W}_{|M_j}$) are computed by linear interpolation from the nodal gradients attached to the three vertices of

the face containing M_i (resp. M_j), where M_i and M_j have been defined above in Figure 1.
A sixth-order dissipation scheme is obtained by considering the following high-order gradient:

$$(\nabla \mathbf{W})_{ij}^{V6} = (\nabla \mathbf{W})_{ij}^{V4} - \frac{1}{30}\left((\nabla \mathbf{W})_{ij}^{U} - 2\,(\nabla \mathbf{W})_{ij}^{C} + (\nabla \mathbf{W})_{ij}^{D}\right) - \frac{2}{15}\left((\nabla \mathbf{W})_{i}^{UN} - 2\,(\nabla \mathbf{W})_{i}^{N} + (\nabla \mathbf{W})_{j}^{N}\right)$$

$$(\nabla \mathbf{W})_{ji}^{V6} = (\nabla \mathbf{W})_{ij}^{V4} - \frac{1}{30}\left((\nabla \mathbf{W})_{ij}^{D} - 2\,(\nabla \mathbf{W})_{ij}^{C} + (\nabla \mathbf{W})_{ij}^{U}\right) - \frac{2}{15}\left((\nabla \mathbf{W})_{j}^{DN} - 2\,(\nabla \mathbf{W})_{j}^{N} + (\nabla \mathbf{W})_{i}^{N}\right)$$

Limiter. The previous MUSCL schemes are not monotone. Therefore, limiting functions must be coupled with the previous high-order gradient evaluations to guarantee the Total-Variation-Diminishing (TVD) property of the scheme. To this aim, the gradient of Relation (4) is replaced by a limited gradient denoted $\left(\nabla \mathbf{W}^{lim}\right)_{ij}$. Here, we will always consider a three entries limiter introduced by Dervieux [5] which is a generalization of the SuperBee limiter proposed by Roe :

$$\left(\nabla \mathbf{W}^{lim}\right)_{ij} = Lim\left(\left(\nabla \mathbf{W}^{D}\right)_{ij}, \left(\nabla \mathbf{W}^{C}\right)_{ij}, \left(\nabla \mathbf{W}^{V4/V6}\right)_{ij}\right)$$

$$\begin{aligned} \text{with : } Lim\,(a,b,c) &= 0 && \text{if } ab \leq 0 \\ &= \text{sign}\,(a)\min\left(2\,|\,a\,|, 2\,|\,b\,|, 2\,|\,c\,|\right) && \text{otherwise} \end{aligned}$$

Mirror state boundary conditions. Mirror state boundary conditions consist in imposing slipping boundary conditions in a weak manner by prescribing a flux rather than directly enforcing a specific value for the variables on the boundary. As we are interested in the Euler equations, the fluid is inviscid and the physically consistent boundary condition on the moving bodies is a slipping boundary condition, *i.e.* $(\mathbf{u}\cdot\mathbf{n})_{|\partial B} = \sigma_{|\partial B}$. Mirror state $\overline{\mathbf{W}}$ associated with boundary state $\mathbf{W}$ is an imaginary state, virtually defined on the other side of the boundary. It is built such that the extrapolated value of $\mathbf{W}_{|\partial B} = (\mathbf{W} + \overline{\mathbf{W}})/2$ on the boundary satisfies $(\mathbf{u}\cdot\mathbf{n})_{|\partial B} = \sigma_{|\partial B}$. We consider a Finite Volume cell C_i in contact with the boundary of object B, and we note $\mathbf{n}_i^k$ the outward non-normalized normal to the k^{th} boundary facet and σ_i^k its normal speed, see Figure 2. The ALE mirror state of state $\mathbf{W}_i^k$ on the other side of boundary interface k of cell C_i is:

$$\overline{\mathbf{W}}_i^k = \left(\overline{\rho}_i = \rho_i,\, \overline{\mathbf{u}}_i = \mathbf{u}_i - 2\left(\mathbf{u}_i\cdot\mathbf{n}_i^k - \sigma_i^k\right)\mathbf{n}_i^k,\, \overline{\varepsilon}_i = \varepsilon_i\right)^T$$

The numerical $HLLC$ boundary flux between the boundary state and its mirror state across boundary interface k of cell C_i can then be calculated and we obtain:

$$\mathbf{\Phi}^{\text{Hllc}}\left(\mathbf{W}_i, \sigma_i^k, \mathbf{n}_i^k\right) = \|\mathbf{n}_i^k\| \begin{pmatrix} 0 \\ p_i^* \mathbf{n}_i^k \\ p_i^* \sigma_i^k \end{pmatrix}, \quad \text{where } p_i^* = \rho_i\left(\eta_i - \sigma_i^k\right)\max\left(c_i, \tilde{c_i} + \eta_i - \sigma_i^k\right) + p_i$$

$$\text{with } \tilde{c_i} = (\gamma - 1)\left(h_i - \sigma_i^k \eta_i - \frac{\left(\eta_i^{tan,k}\right)^2}{2} + \frac{\left(\sigma_i^k\right)^2}{2}\right)$$

and $\eta_i^{tan,k} = \|\mathbf{u} - \left(\mathbf{u}_i\cdot\mathbf{n}_i^k\right)\mathbf{n}_i^k\|$ the norm of the component of $\mathbf{u}$ tangent to the boundary.

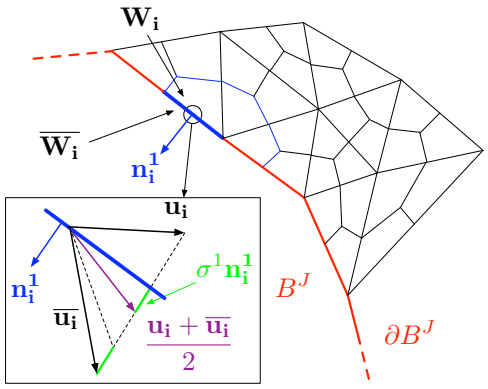

Figure 2: *Mirror state across boundary interface k of cell C_i.*

4 ALE SPECIFIC ISSUES REGARDING TIME DISCRETIZATION

4.1 The *GCL* law

We need to check that the movement of the mesh is not responsible for any artificial alteration of the physical phenomena at stake. 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 (1) is written for a constant state, assuming $\mathbf{F}_{ext} = 0$, we get, for any arbitrary closed volume $C = C(t)$ with boundary $\partial C(t)$:

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

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 $\partial C(t)$, this relation is integrated into:

$$|\,C(t+\Delta t)\,| - |\,C(t)\,| = \int_t^{t+\Delta t}\int_{\partial C(t)} (\boldsymbol{w}\cdot\mathbf{n})\,\mathrm{d}\boldsymbol{s}\,\mathrm{d}t, \quad \text{with } t \text{ and } t+\Delta t \in [0,T]\,. \tag{5}$$

From a geometrical viewpoint, this relation states that the algebraic variation of the volume of C between two instants equals the algebraic area swept by its boundary.

4.2 Accuracy preserving and *DGCL* temporal schemes

A *DGCL* property for each temporal scheme. The continuous *GCL* relation raises several important questions: should Relation (5) be satisfied at the discrete level? What are the effects of respecting this law at a discrete level on the consistency, stability and accuracy of the numerical scheme? How to enforce it at a discrete level? In the following, we will restrict ourselves to Finite Volume schemes.
Thomas and Lombard [24] were probably among the first to emphasize the importance of this law. Since then, the subject has often proved controversial. However, there are currently two things almost everybody agrees about:

1. The *GCL* serves as an additional constraint controlling the way the computation of the geometrical parameters is performed. Indeed, we *a priori* ignore how to compute geometrical parameters σ and $\mathbf{n}$ appearing in Relation (3). For instance, should one take $\mathbf{n}$ at t^n? at t^{n+1}? Or take a kind of averaged normal? The *GCL* can help answer this question.

2. Enforcing the *GCL* at the discrete level is mandatory. An exception can be made if sufficiently small time steps are used but, in practice, it is very time consuming and it is hard to know what "sufficiently small" means.

In the sequel, we detail the approach developped by Yang and Mavriplis to extend well-known temporal schemes to *ALE* simulations while enforcing the *GCL*. We show how this methodology works on a Strong-Stability-Preserving Runge-Kutta (*SSPRK*) scheme.

The approach of Yang and Mavriplis. Since 2005, Mavriplis and Yang ([26, 18, 27]) have renewed the way of thinking the *GCL*. The objective was initially to further clarify the link between the *DGCL* nature of a temporal scheme and the preservation of its order of accuracy. The originality of this approach consists in defining precisely which *ALE* parameters are true degrees of freedom and which are not. In contrast with other approaches ([14, 11, 20]), 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 geometrical parameters, such as the cells interfaces normals or the upwind/downwind triangles 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 area swept.
Practically speaking, the interfaces normal speeds are found by simply rewriting the scheme for a constant discrete solution, which leads to a small linear system which can be inverted by hand. This procedure is detailed in the next section for one *SSPRK* scheme. Any fixed-mesh scheme can be extended to the case of moving meshes thanks to this methodology, and the resulting scheme is naturally *DGCL*. Even if this has not been proven theoretically, the expected order of convergence has also been observed numerically for several schemes designed with this method.

***SSPRK* schemes.** Runge-Kutta methods are famous multi-stages methods to integrate *ODE*s. In the numerical resolution of hyperbolic *PDE*s, notably the Euler equations which are widely used in aeronautics, it is desirable to exhibit among the huge family of Runge-Kutta schemes the ones satisfying the so-called Strong Stability Preserving (*SSP*) property.
A Runge-Kutta scheme is said to be *SSP* if we have $|\,\mathbf{W}^{n+1}\,| \leq |\,\mathbf{W}^{n}\,|$, $|\cdot|$ being here a chosen semi-norm. The semi-norm is classically the *TVD* norm defined by:

$$|\,\mathbf{W}^{n+1} - \mathbf{W}^{n}\,|_{TVD} = \sum_{i=1}^{N} |\,\mathbf{W}_i^{n+1} - \mathbf{W}_i^{n}\,|\,.$$

See [22, 9, 23, 12, 7] for more details on the subject.

In the sequel, we note SSPRK(s,p) the s-stages *SSPRK* scheme of order p. We adopt the following notations:

$$\mathbf{f}_i^s = \sum_{k=0}^{n_i-1} \mathbf{\Phi}(\mathbf{W}_i^s, \mathbf{n}_i^{k,s}, \sigma_i^{k,s}) \text{ with } \begin{cases} n_i & \text{the number of facets of cell } C_i \\ \mathbf{n}_i^{k,s} & \text{the outward } \textbf{non-normalized} \text{ normal to facet } k \text{ of cell } C_i^s \\ \sigma_i^{k,s} & \text{normal speed of facet } k \text{ of cell } C_i^s \ . \end{cases}$$

Superscript notation X^s indicates that the considered quantity is the X obtained at stage s of the Runge-Kutta process. For instance, C_i^s is the cell associated with vertex P_i when the mesh has been moved to its s^{th} Runge-Kutta configuration. Coefficients $(c_l)_{0\leq l\leq s}$ indicate the relative position in time of the current Runge-Kutta configuration: $t^s = t^n + c_s\,\mathrm{d}t$ with $\mathrm{d}t = t^{n+1} - t^n$. Finally, we note $A_i^{k,s}$ the area swept by facet k of cell C_i between the initial Runge-Kutta configuration and the s^{th} one.

Application of Mavriplis and Yang approach to SSPRK(4,3) scheme.

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

In the left part of the array, the SSPRK(4,3) scheme is given in its Butcher formulation, whereas it is given in its Shu-Osher representation [22] on the right. For this scheme to be *DGCL*, it must preserve a constant solution $\mathbf{W}_i = \mathbf{W}_0$. In this specific case, our conservative variable is $\mathbf{Y_i} = |\,C_i\,|\mathbf{W}_0$ and the purely physical fluxes vanish, leading to $\mathbf{f}_i^s = -\mathbf{W}_0 \sum_{k=0}^{n_i-1} \|\mathbf{n}_i^{k,s}\|\sigma_i^{k,s}$.

Therefore, the scheme writes:

$$
\begin{aligned}
& & |\,C_i^0\,| &= |\,C_i^n\,| & & \\
|\,C_i^1\,| - & |\,C_i^0\,| &= \sum_{k=0}^{n_i-1} A_i^{k,1} &= \frac{\mathrm{d}t}{2}\sum_{k=0}^{n_i-1} \|\mathbf{n}_i^{k,0}\|\sigma_i^{k,0} \\
|\,C_i^2\,| - & |\,C_i^0\,| &= \sum_{k=0}^{n_i-1} A_i^{k,2} &= \frac{\mathrm{d}t}{2}\sum_{k=0}^{n_i-1} \left(\|\mathbf{n}_i^{k,0}\|\sigma_i^{k,0} + \|\mathbf{n}_i^{k,1}\|\sigma_i^{k,1}\right) \\
|\,C_i^3\,| - & |\,C_i^0\,| &= \sum_{k=0}^{n_i-1} A_i^{k,3} &= \frac{\mathrm{d}t}{6}\sum_{k=0}^{n_i-1} \left(\|\mathbf{n}_i^{k,0}\|\sigma_i^{k,0} + \|\mathbf{n}_i^{k,1}\|\sigma_i^{k,1} + \|\mathbf{n}_i^{k,2}\|\sigma_i^{k,2}\right) \\
|\,C_i^4\,| - & |\,C_i^0\,| &= \sum_{k=0}^{n_i-1} A_i^{k,4} &= \frac{\mathrm{d}t}{2}\sum_{k=0}^{n_i-1} \left(\frac{1}{3}\|\mathbf{n}_i^{k,0}\|\sigma_i^{k,0} + \frac{1}{3}\|\mathbf{n}_i^{k,1}\|\sigma_i^{k,1} + \frac{1}{3}\|\mathbf{n}_i^{k,2}\|\sigma_i^{k,2} + \|\mathbf{n}_i^{k,3}\|\sigma_i^{k,3}\right)
\end{aligned}
$$

A necessary and natural condition for the above relations to be satisfied is to have, for each interface k of each Finite Volume cell C_i:

$$
\begin{pmatrix} A_i^{k,1} \\ A_i^{k,2} \\ A_i^{k,3} \\ A_i^{k,4} \end{pmatrix} = \mathrm{d}t \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \|\mathbf{n}_i^{k,0}\|\sigma_i^{k,0} \\ \|\mathbf{n}_i^{k,1}\|\sigma_i^{k,1} \\ \|\mathbf{n}_i^{k,2}\|\sigma_i^{k,2} \\ \|\mathbf{n}_i^{k,3}\|\sigma_i^{k,3} \end{pmatrix} \quad \Leftrightarrow \quad \begin{pmatrix} \|\mathbf{n}_i^{k,0}\|\sigma_i^{k,0} \\ \|\mathbf{n}_i^{k,1}\|\sigma_i^{k,1} \\ \|\mathbf{n}_i^{k,2}\|\sigma_i^{k,2} \\ \|\mathbf{n}_i^{k,3}\|\sigma_i^{k,3} \end{pmatrix} = \frac{1}{\mathrm{d}t} \begin{pmatrix} 2 & 0 & 0 & 0 \\ -2 & 2 & 0 & 0 \\ 0 & -2 & 6 & 0 \\ 0 & 0 & -2 & 2 \end{pmatrix} \begin{pmatrix} A_i^{k,1} \\ A_i^{k,2} \\ A_i^{k,3} \\ A_i^{k,4} \end{pmatrix}
$$

Therefore, the normal speed of interface k of cell C_i must be updated as follows in the Runge-Kutta process:

$$
\begin{aligned}
\sigma_i^{k,0} &= \frac{2}{\mathrm{d}t\|\mathbf{n}_i^{k,0}\|} A_i^{k,1}, & \sigma_i^{k,1} &= \frac{1}{\mathrm{d}t\|\mathbf{n}_i^{k,1}\|}\left(-2A_i^{k,1} + 2A_i^{k,2}\right) \\
\sigma_i^{k,2} &= \frac{1}{\mathrm{d}t\|\mathbf{n}_i^{k,2}\|}\left(-2A_i^{k,2} + 6A_i^{k,3}\right), & \sigma_i^{k,3} &= \frac{1}{\mathrm{d}t\|\mathbf{n}_i^{k,3}\|}\left(-2A_i^{k,3} + 2A_i^{k,4}\right)
\end{aligned}
$$

and the $\|\mathbf{n}_i^{k,s}\|$ are computed on the mesh once it has been moved to the s^{th} Runge-Kutta configuration. The $A_i^{k,s}$ are computed as described in [21].

4.3 A new changing-topology *ALE* formulation

Context and problematics. As already mentionned, we are interested in large deformation simulations, likely to involve shear movements. Usually, when the connectivity of the mesh must be changed between two time steps, local or global interpolation procedures are used to get the solution on the new mesh. It is standard practice to use a simple $\mathbb{P}^1$-Lagrangian projection, even if an increasing number of research teams try to investigate and improve the accuracy and conservativity properties of this interpolation step, [25, 17]. However, the effects of these repeated interpolations are still not well understood, especially when they are performed locally on the fly after each topology change.

Our opinion is that having a true changing topology *ALE* scheme would get us rid of this spoiling projection step and will better fit into the global *ALE* framework. To our knowledge,

only a few attempts to do this can be found in the literature, see for example [19, 13]. Indeed, it is a very difficult subject, as designing a fully ALE procedure with topologically-variable mesh implies that:

- the mesh data structures will be much more complex and dynamic. The number of edges and elements is the same in two dimensions but varies in three dimensions if connectivity changes occur. If edges collapsings and creations are authorized, the number of vertices also evolves in time and so do the number of cells, like the number of their interfaces.
- we are able to generate four dimensional space-time meshes in three dimensions, linking Finite Volume cells at t^n with cells at t^{n+1} (see Figure 3); incidentally, note that this kind of meshes is hard to handle because each element has a different number of faces and vertices
- the space-time meshes must be generated at each solver time step
- these space-time meshes must be hybrid, involving cell-based prismatic elements in regions were the connectivity of the mesh does not change (to be coherent with the classical ALE framework) and tetrahedra when connectivity changes occur, see Figure 3
- a new fully ALE scheme based on the cells movement rather than on the vertices and elements movement and involving space-time Finite Volume cells and space-time interfaces.

Such an approach cannot be considered at the moment as it would imply the rewriting of most of the Finite Volume ALE code and would necessitate the inclusion of a $4D$ mesher inside the solver. As a first step, we therefore chose to simplify the problem following the line below:

- we focused on changing-topology meshes only involving swaps, *i.e.* neither addition nor suppression of vertices. This choice is due to the powerfulness of this tool and the fact that it sometimes appears as the only solution to handle shears and large deformation movements
- once a swap has been performed, all the edges touching the swapped edge are set as blocked until the next optimization step. Note that this is not as restrictive as it can seem at first glance. Indeed, imagine an edge has been blocked due to a previous swap in its neighborhood while would have wished to swap it. It is generally not a problem as this swap might be done at the next step on a configuration which is generally not too different from the current one.
- the space-time mesh is generated implicitly, *i.e.* we never actually store its entities but they virtually exist through the way our new scheme works.

Our scheme. The swap operation is considered as a time continuous process during which some Finite Volume cells corners are duplicated and other collapsed. The idea is to consider a three-dimensional mesh linking the cells affected by the swap at t^n and those at t^{n+1}, the time being the third dimension. In the classical ALE framework, all the space-time interfaces are quadrangular and generally twisted, see Figure 3 (left). If a swap occurs, the space-time mesh

in the swap region exhibits a pseudo-tetrahedral element in two-dimensions, the faces of which are bi-planar and made of two triangles, see Figure 3 (right). Note that space-time interfaces are straight, *i.e.* the bi-segment normals keep the same direction while moving along these space-time triangles between two time steps.

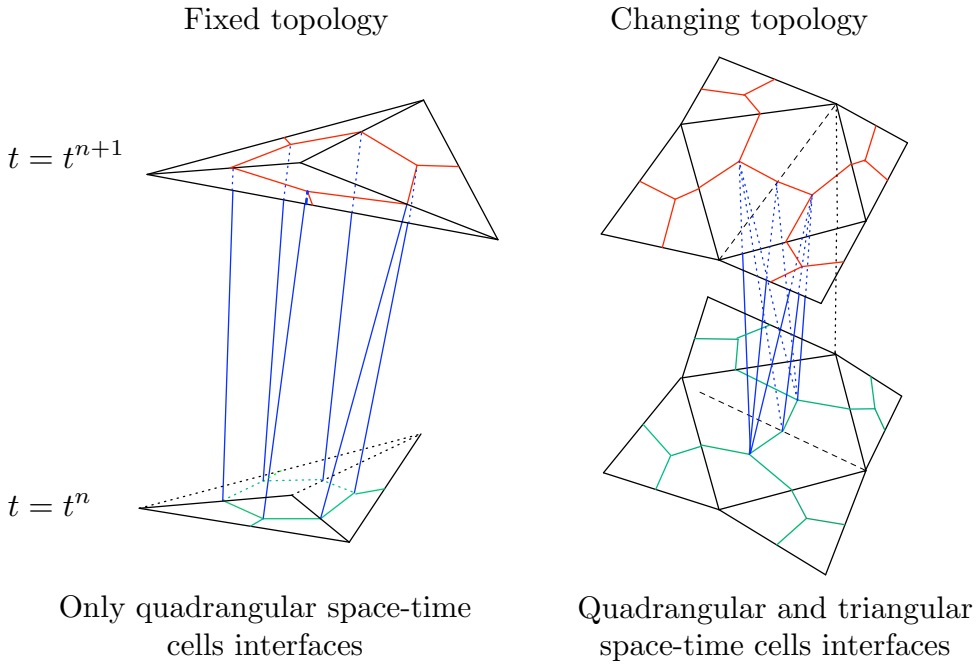

Figure 3: Space-time vision of the edge swapping operation and its effect on space-time Finite Volume cells interfaces (in blue). The pseudo-tetrahedral space-time element is shown on the right, in blue lines.

By cutting the space-time mesh at an intermediate time between t^n and t^{n+1}, we see that an evanescent cell is created just after t^n and vanishes at t^{n+1}. Consequently, some fluxes are exchanged between this evanescent cell and its surrounding real cells between the two configurations. The cells present at $t^{n+\frac{1}{2}}$ are depicted in Figure 4. In Figure 5 (left), we have represented the directions of the purely geometrical fluxes entering and going out of this evanescent cell on a specific configuration. Two interfaces are now associated with the old green edge $\overrightarrow{P_0P_1}$ and the new red edge $\overrightarrow{P_2P_3}$ instead of just one, each of these interfaces having its own ALE parameters. In the sequel, we note $a^+ = \max(0, a)$ (resp. $a^- = \min(0, a)$) the positive (resp. negative) part of a real value a and we have trivially :

$$a = \frac{a^+ + a^-}{2}, \quad |a| = \frac{a^+ - a^-}{2}, \quad a^+ a^- = 0\,.$$

We also define new parameters associated with each of the four interfaces of the evanescent cell, each interface $i \in [\![0, 3]\!]$ being made of two bi-segments $k = 1$ and $k = 2$, see Figure 5 (right). In fact, two of these interfaces are associated with the old swapped edge and the two others are associated with the new edge.

- $\sigma_i^k,\ i \in [\![0,3]\!],\ k \in \{1,2\}$ is the normal speed of bi-segment k of interface i
- $\mathbf{n}_i^k$ is the **non-normalized** normal to bi-segment k associated with interface i
- $\mathbf{n}_i = \mathbf{n}_i^1 + \mathbf{n}_i^2$ is the global **non-normalized** normal to interface i
- $\sigma_i = \dfrac{\sigma_i^1 \|\mathbf{n}_i^1\| + \sigma_i^2 \|\mathbf{n}_i^2\|}{\|\mathbf{n}_i\|}$ the normal speed of interface i

Regarding the four bordering edges, we only consider the bi-segment, noted k^s, which is located inside the swap area. The normal speed of bi-segment k^s of **bordering** edge $\overrightarrow{P_iP_j}$ is noted $\sigma_{ij}^{k^s}$. For the moment, this normal speed does not take into account the redistribution of the mass due to the apparition and disappearance of the evanescent cell.

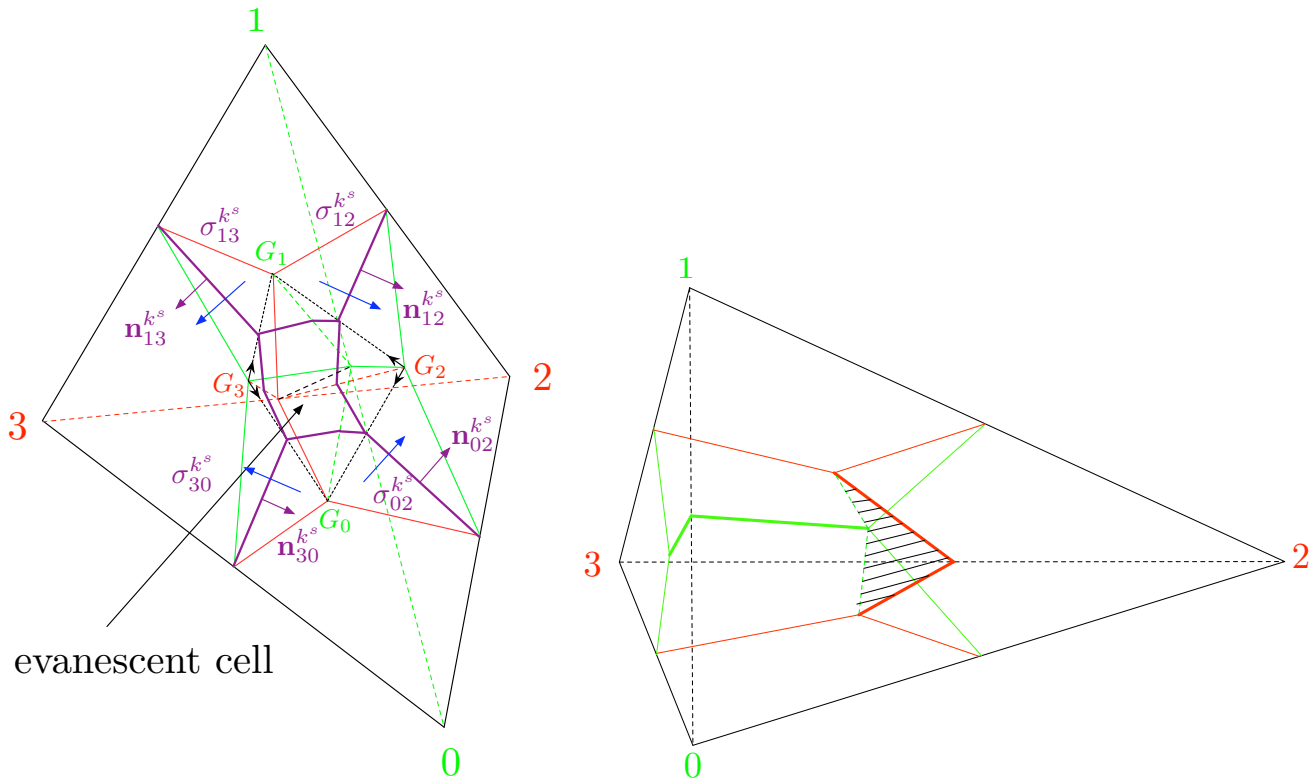

Figure 4: Left, evolution of the Finite Volume cells due to the swap operation: the two green bi-segments associated with the swapped green edge vanish while two red bisegments associated with the new red edge are created. The intermediate configuration at $t^{n+\frac{1}{2}}$ is in violet. Blue arrows represent the "standard" ALE geometrical fluxes exchanges between the real cells. Right, a trickier swap configuration.

First, due to the specific geometric configuration of the evanescent cell, see Figure 3, the reversed Thales theorem applies and we have the following relations:

$$\mathbf{n}_1^1 = -\mathbf{n}_0^1, \quad \mathbf{n}_1^2 = -\mathbf{n}_0^2, \quad \|\mathbf{n}_0\| = \|\mathbf{n}_1\|, \quad \mathbf{n}_3^1 = -\mathbf{n}_2^1, \quad \mathbf{n}_3^2 = -\mathbf{n}_2^2, \quad \|\mathbf{n}_2\| = \|\mathbf{n}_3\|$$

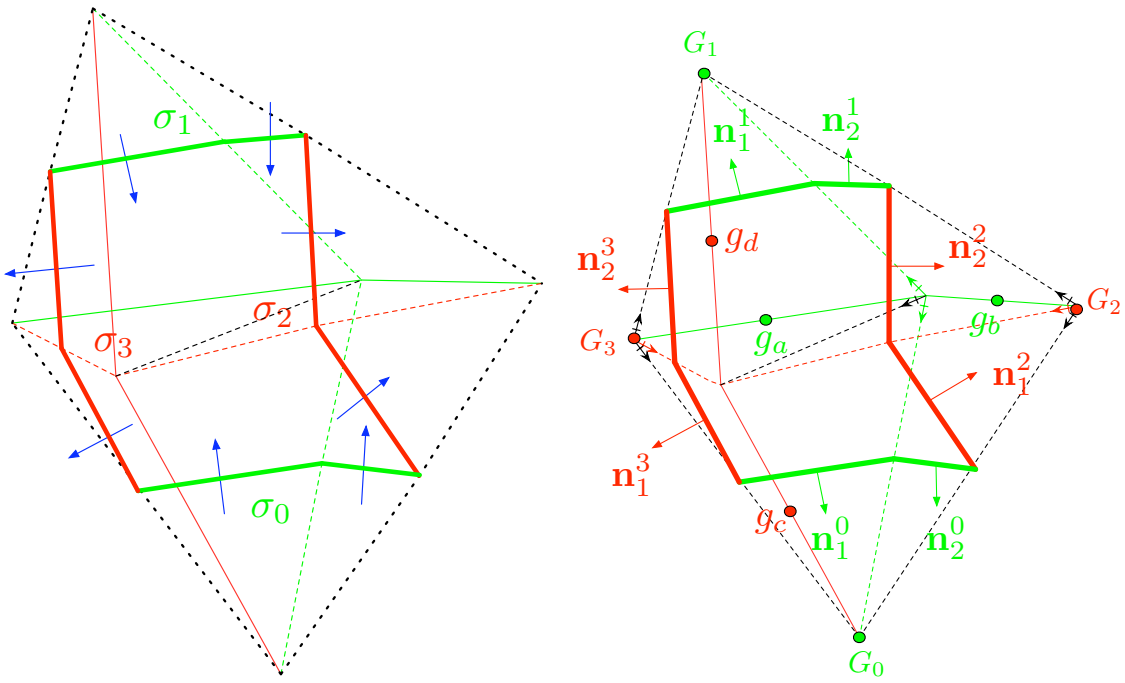

Figure 5: Left, zoom on the evanescent cell appearing during the edge-flipping operation. The geometrical flux exchanges between the evanescent cell and its neighbors are represented with blue arrows. Right, notations for the *ALE* swap scheme.

Second, the normal speeds of the bi-segments are not computed directly but rather through $A_i^k = \sigma_i^k \|\mathbf{n}_i^k\|$, the area swept by the bi-segment during the swap process. These eight algebraic swept areas are computed as follows [3]:

$$A_0^1 = \frac{1}{2}\left(\boldsymbol{w}_{g_a \to G_0} \cdot \mathbf{n}_0^1\right), \quad A_0^2 = \frac{1}{2}\left(\boldsymbol{w}_{g_b \to G_0} \cdot \mathbf{n}_0^2\right), \quad A_1^1 = \frac{1}{2}\left(\boldsymbol{w}_{g_a \to G_1} \cdot \mathbf{n}_1^1\right), \quad A_1^2 = \frac{1}{2}\left(\boldsymbol{w}_{g_b \to G_1} \cdot \mathbf{n}_1^2\right),$$

$$A_2^1 = \frac{1}{2}\left(\boldsymbol{w}_{G_2 \to g_c} \cdot \mathbf{n}_2^1\right), \quad A_2^2 = \frac{1}{2}\left(\boldsymbol{w}_{G_2 \to g_d} \cdot \mathbf{n}_2^2\right), \quad A_3^1 = \frac{1}{2}\left(\boldsymbol{w}_{G_3 \to g_c} \cdot \mathbf{n}_3^1\right), \quad A_3^2 = \frac{1}{2}\left(\boldsymbol{w}_{G_3 \to g_d} \cdot \mathbf{n}_3^2\right). \tag{6}$$

where we have noted $\boldsymbol{w}_{M^n \to N^{n+1}} = \dfrac{\mathbf{x}_N^n - \mathbf{x}_M^{n+1}}{\mathrm{d}t}$ and we get:

$$\sigma_0 = 2\frac{A_0^1 + A_0^2}{\|\mathbf{n}_0\|}, \quad \sigma_2 = 2\frac{A_2^1 + A_2^2}{\|\mathbf{n}_2\|}, \quad \sigma_1 = 2\frac{A_1^1 + A_1^2}{\|\mathbf{n}_1\|}, \quad \sigma_3 = 2\frac{A_3^1 + A_3^2}{\|\mathbf{n}_3\|}. \tag{7}$$

Note that the evanescent nature of the middle cell is contained in the formula:

$$\sum_{i,k} A_i^k = 0 \iff (\sigma_0 + \sigma_1)\|\mathbf{n}_0\| + (\sigma_2 + \sigma_3)\|\mathbf{n}_2\| = 0\,.$$

[3] The factors 2 appearing in Relations (6) and (7) is due to the definition of the $\mathbf{n}_i^k$. Indeed, the $\mathbf{n}_i^k$ are the bi-segments normals taken on the configuration at t^n to match with Mavriplis and Yang approach, whereas the swept area must be computed using the normals on the configuration at $t^{n+\frac{1}{2}}$, which, according to Thales theorem, equal $\frac{\mathbf{n}_i^k}{2}$.

To finish with, we define area A by:

$$\begin{aligned} A &= A_0^1 + A_0^2 + A_1^1 + A_1^2 = -\left(A_2^1 + A_2^2 + A_3^1 + A_3^2\right) \\ &= \left(\sigma_0^+ + \sigma_1^+\right)\|\mathbf{n}_0\| + \left(\sigma_2^+ + \sigma_3^+\right)\|\mathbf{n}_2\| = -\left(\sigma_0^- + \sigma_1^-\right)\|\mathbf{n}_0\| - \left(\sigma_2^- + \sigma_3^-\right)\|\mathbf{n}_2\|. \end{aligned}$$

We now explain how the normal speeds of the six edges involved in the swap operations (the four bordering edges, the old swapped edge and the new one) are corrected. The geometrical parameters of the four surrounding edges of the swap region are corrected while the new and swapped edges are given specific parameters to take the mass redistribution between the vertices into account. We note $\sigma_{ij}^{k^s,*}$ the corrected normal speed of bi-segment k^s of **bordering** edge $\overrightarrow{P_iP_j}$, bi-segment k^s still being the one located inside the swap region. Our scheme writes as follows:

$$\begin{array}{lcll} \sigma_{02}^{k^s,*} = \sigma_{02}^{k^s} + (\sigma_2^+\sigma_0^- - \sigma_2^-\sigma_0^+)\dfrac{\|\mathbf{n}_0\|\|\mathbf{n}_2\|}{\|\mathbf{n}_{02}\|A} & & \mathbf{n}_{02}^* = \mathbf{n}_{02}^n \\ \sigma_{03}^{k^s,*} = \sigma_{03}^{k^s} + (\sigma_3^+\sigma_0^- - \sigma_3^-\sigma_0^+)\dfrac{\|\mathbf{n}_0\|\|\mathbf{n}_3\|}{\|\mathbf{n}_{03}\|A} & & \mathbf{n}_{03}^* = \mathbf{n}_{03}^n \\ \sigma_{12}^{k^s,*} = \sigma_{12}^{k^s} + (\sigma_2^+\sigma_1^- - \sigma_2^-\sigma_1^+)\dfrac{\|\mathbf{n}_1\|\|\mathbf{n}_2\|}{\|\mathbf{n}_{12}\|A} & & \mathbf{n}_{12}^* = \mathbf{n}_{12}^n \\ \sigma_{13}^{k^s,*} = \sigma_{13}^{k^s} + (\sigma_3^+\sigma_1^- - \sigma_3^-\sigma_1^+)\dfrac{\|\mathbf{n}_1\|\|\mathbf{n}_3\|}{\|\mathbf{n}_{13}\|A} & & \mathbf{n}_{13}^* = \mathbf{n}_{13}^n \\ \sigma_{01}^* = (\sigma_0^-\sigma_1^+ - \sigma_0^+\sigma_1^-)\dfrac{\|\mathbf{n}_0\|}{A} & & \mathbf{n}_{01}^* = \mathbf{n}_{01}^n \\ \sigma_{23}^* = (\sigma_2^-\sigma_3^+ - \sigma_2^+\sigma_3^-)\dfrac{\|\mathbf{n}_2\|}{A} & & \mathbf{n}_{23}^* = \mathbf{n}_{23}^{n+1} \end{array} \tag{8}$$

By way of example, in Figure 4, $\sigma_0 > 0$, $\sigma_1 > 0$, $\sigma_2 < 0$ and $\sigma_3 < 0$, thus $\sigma_{01}^* = 0$ and $\sigma_{23}^* = 0$. This means that all the mass sucked from cell C_0 is fully redistributed to C_2 and C_3 and nothing is given to C_1. The same happens for C_1 which receives nothing from C_0. However, this case is a very favorable one used for pedagogic purposes. For instance, for the configuration shown in Figure 4 (right), σ_2 and σ_3 are of opposite signs and thus σ_{23}^* is non zero. In this particular case, mass is crossing the interface from C_2 to C_3.

Practically speaking, the old and the new edges coexist in the mesh between t^n and t^{n+1}. The old edge is removed only once the flux across it between t^n and t^{n+1} has been computed. These edges are treated exactly like the others, *i.e.* a complete (geometrical **and** physical) *ALE* flux is computed across these edges. The extremities of the edges involved in the swap operation can also move, and it will automatically be taken into account by Scheme (8). Eventually, this scheme is of course *DGCL*.

Space-accuracy order can be enhanced using the *MUSCL* technique coupled with a limiter, therefore guaranteeing that this truly *ALE* scheme is intrinsically *TVD*, contrary to the classical projection approach which requires a repairing step. However, if low dissipation is desired,

one must determine how to compute upwind/downwind gradients which are necessary for the V4/V6 schemes. It is clear that the upwind/downwind triangles must be triangles belonging to the configuration at t^n, at least if we keep on following Mavriplis and Yang method. For the old edge, we naturally take its upwind/downwind triangles at t^n and for the new one, we do as if it belonged to the configuration at t^n and compute its upwind/downwind triangles on this configuration. Once the upwind/downwind triangles are found, the V4/V6 schemes is naturally extended for our topology-changing *ALE* formulation. Therefore, this scheme is of order two in space, with a low numerical dissipation.

Perspectives. Finally, as being *DGCL* is a sufficient condition for a scheme to be at least first-order time-accurate on moving meshes according to [10], our scheme is at least of order one in time. However, there is *a priori* no reason for it to be of order higher than one. The extension to multi-step integration schemes like the implicit Backward Differentiation Formula schemes seems quiet easy, even if we have not tested it yet. As regards the extension of *ALE DGCL* Runge-Kutta schemes of Section 4.2 to meshes with time-dependent connectivity, it is currently underway. The generalization of this scheme to the three dimensional swap is a very difficult problem we are currently working on.

4.4 Numerical results and perspectives

Test case description. To show the usefulness and efficiency of the swap operation, we performed a simplified turbo-machinery simulation. A fluid is emitted radially from a central hub of radius $r_0 = 0.3$. The emitted fluid applies a pressure on the blades, which makes them rotate quicker and quicker. By centrifugal effects, the fluid is propelled toward the exterior. The initial conditions, written in cylindrical coordinates $(\mathbf{e}_r, \mathbf{e}_\theta)$ are as follows:

$$\rho = \rho(r) = \frac{r_0}{r}\rho(r_0) = \frac{r_0}{r}\rho_{in}, \quad \rho\mathbf{u} = \rho(r) q_{unif}\mathbf{e}_r, \quad \rho e = \frac{p_{unif}}{\gamma - 1} + \frac{1}{2}\rho(r) q_{unif}^2$$

with $\rho_{in} = \rho(r_0) = 1$ the inflow density on the hub, $p_{unif} = 1$ the initial uniform pressure and $\mathbf{u}_{unif}$ the initial uniform radial velocity. This initial state ensures that we initially have a constant, uniform radial flow. Indeed, if we take a ring delimited by radius r_0 and $r > r_0$, all the mass entering the ring across the inner circle is expelled through the outer circle, which means that the initial state satisfies the stationary conservation equation $\operatorname{div}(\rho\,\mathbf{u}) = 0$.

$$\int_{\theta=0}^{2\Pi} \rho(r)\mathbf{u}(u)\cdot\mathbf{e}_r\, r\mathrm{d}\theta = \int_{\theta=0}^{2\Pi} \rho(r_0)\mathbf{u}(r_0)\cdot\mathbf{e}_r\, r_0\mathrm{d}\theta$$

An inflow boundary condition is imposed on the hub, a slipping boundary condition on the blades and a transmitting boundary condition on the bounding box. The initial mesh is shown on Figure 6. Several sub-domains are defined: sub-domain ① is linked to the hub and is set to be static, sub-domain ② is linked to the blades and so turns while sub-domain ③ is simply the rest of the computational domain and is also set to be static.

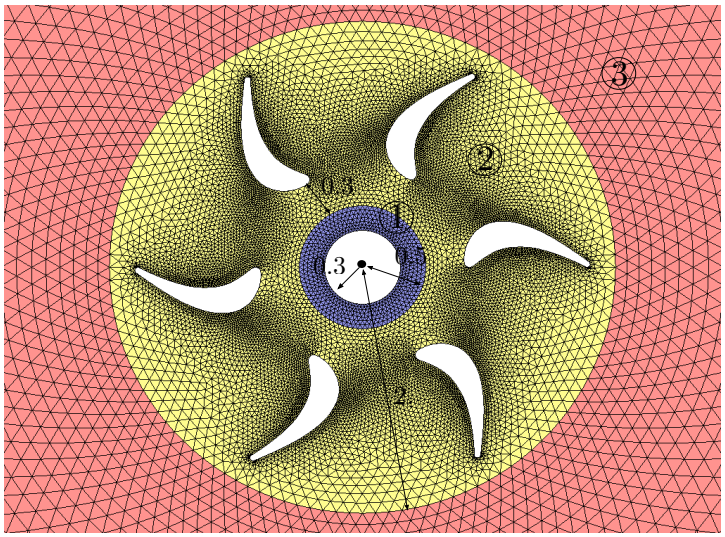

Figure 6: The test case geometry and the associated mesh. The three sub-domains of the mesh are represented in violet, yellow and pink, respectively.

Results analysis. This simulation constitutes an excellent example of the efficiency of the swap when the mesh is sheared. Indeed, if we forbid the use of the swap operation, the mesh deteriorates very quickly because while trying to follow the blades movement, the elements progressively stretch until the minimal altitude of the mesh is so small that the simulation cannot advance in time anymore. Indeed, the *CFL* condition makes the solver time step tend to 0. In our simulation, the initial mesh has an excellent quality and during the movement, only the layers of elements separating moving sub-domain ① from the two other fixed sub-domains change in time. The vertices attached to the blades are the only ones to move in time, and, as they do it in a completely rigid manner, the initial quality of the mesh is preserved, Figure 8. Therefore, the quality of the initial mesh has been maintained throughout the computation and the simulation can evolve as long as desired. Besides, we note that the changing-topology *ALE* scheme works well and gives a result which at least seems in accordance with the physical intuition, Figure 7. The conservativity has also been positively checked.

5 MOVING MESH STRATEGIES

Moving mesh issues are the most problematic in large displacements simulations, especially with non-uniform or adapted meshes. Indeed, the moving mesh algorithm must fulfill the following requirements:

- it must be very efficient as it is called at each solver time step.
- it must preserve the validity of the mesh, *i.e.* it should not generate reversed elements. Remeshing must indeed remain occasional because this operation, if constantly repeated, becomes costly and spoils the solution accuracy due to the interpolation stage it requires
- it must preserve elements quality to maintain the solution accuracy and sufficiently big

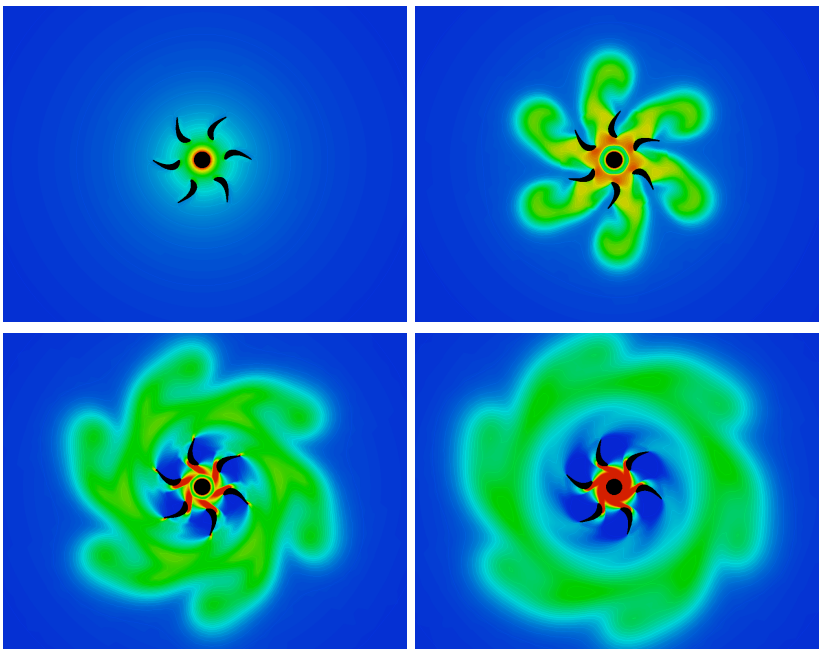

Figure 7: The density of the fluid obtained by resolution of the complete fluid-rigid-body interaction problem, using our changing-topology *ALE* scheme. Solution at $t = 0.05s$, $6s$, $12s$ and $14s$.

time steps

- it must handle small mesh size regions, shear and large movements
- the movement must preserve the (anisotropic) adaptation, if any.

5.1 Prescription of the movement inside the physical domain

In moving domain simulations, the whole mesh must move in order to follow the geometry movement while keeping a valid mesh, *i.e.* without reversed elements. The problem is the following: knowing the displacement of the vertices located on the moving boundaries, which displacement should be prescribed to the inner vertices to respect at the best the above criteria?

Elasticity analogy. We chose to prescribe the vertices movement by solving the *linear elasticity equation* with a $\mathbb{P}^1$ Finite Element method (*FEM*) as suggested in [3]:

$$\operatorname{div}(\mathcal{S}(\mathcal{E})) = 0\,, \quad \text{with } \mathcal{E} = \frac{\nabla \mathbf{d} +^t \nabla \mathbf{d}}{2}\,, \tag{9}$$

where $\mathcal{S}$ and $\mathcal{E}$ are respectively the constraint and the deformation tensors and $\mathbf{d} = (d_1, d_2, d_3)^T$ is the Lagrangian displacement of the vertices. The constraint tensor follows the linear elasticity behavior law, where ν is the Poisson ratio, E is Young's modulus and λ_1, λ_2 are the Lamé coefficients of the material:

$$\mathcal{S}(\mathcal{E}) = \lambda_1 \operatorname{trace}(\mathcal{E})\, \mathcal{I}_d + 2\,\lambda_2\, \mathcal{E}\,, \text{ or } \quad \mathcal{E}(\mathcal{S}) = \frac{1+\nu}{E}\mathcal{S} - \frac{\nu}{E}\operatorname{trace}(\mathcal{S})\, \mathcal{I}_d$$

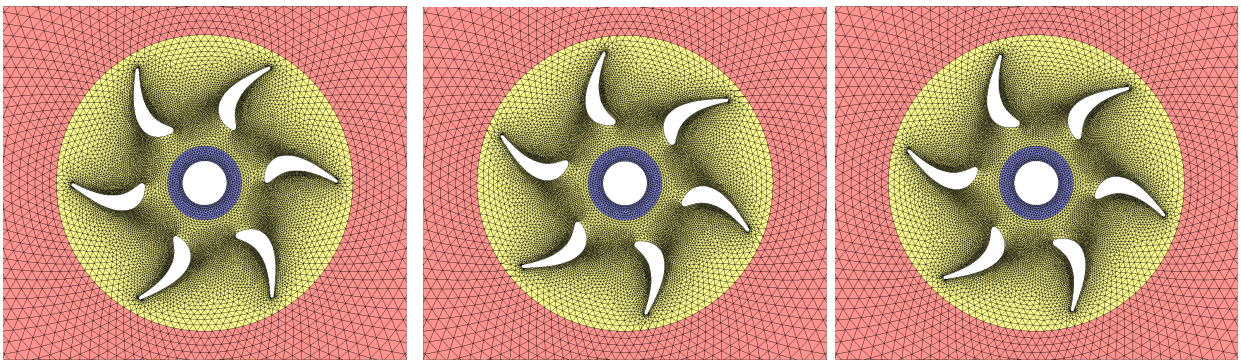

Figure 8: The variable-topology mesh at $t = 0.05s$, $12s$ and $14s$.

The higher the Young modulus, the stiffer the material. In our context, E is typically of the order 10^5 and $\nu \approx 0.49$, which corresponds to a very soft, nearly incompressible material. The Finite Element system is solved by a *GMRES* algorithm coupled with an *ILU* preconditioner. Note that the solution computed at the previous elasticity resolution step is used as initial guess for the new resolution in order to reduce the number of iterations needed to converge the elasticity system.

Elasticity-dedicated mesh. The resolution is performed on an elasticity dedicated mesh, which is taken to be uniform and much coarser than the one used to solve the Euler equations. The displacement of the inner vertices of the finer Euler mesh is then obtained by a $\mathbb{P}^2$-Lagrangian interpolation. This simple trick has several assets, notably in the context of anisotropic mesh adaptation:

- it would be very hard to solve the Finite Element elasticity system on an anisotropic mesh adapted to the physics of the Eulerian flow, due to the creation of an artificial extra-stiffness hindering the convergence of the linear system resolution; using an elasticity-dedicated mesh naturally gets rid of this problem
- it avoids to restructure the Finite Element elasticity matrix each time a connectivity change is performed in the mesh, which is very frequent in our case where anisotropic adaptation tends to create stretched elements along shock waves
- it reduces the size of the elasticity linear system, thus favorably impacting *CPU* time and storage

Note that the elasticity mesh must be moved along with the computational mesh, unless otherwise specified. However, this is generally cheap in terms of *CPU* time as the elasticity mesh is coarser and uniform, which makes it much easier to move.
Eventually, the elasticity system is not solved at each solver iteration but rather at some chosen moments. The duration between two elasticity computations is called the elasticity time step. The trajectory between two computations is then considered as linear. The simplest choice is to solve the elasticity system only every $m \geq 1$ iterations of the flow solver but this can be problematic if a sharp change occurs in the trajectory of some inner vertex. Besides, in [26] this

kind of artifice is inadvisable, as a loss of temporal accuracy can stem from the lack of regularity of the vertices numerical trajectories. A clever way to proceed consists in adapting the elasticity time step according to the smoothness of the vertices trajectories, the most winding trajectory driving the adaptation.

Inhomogeneous mesh. Another advantage of elasticity-like methods is to offer the opportunity to adapt the local material properties of the mesh, especially its stiffness, according to the distortion and efforts born by each element. In our case, only basic tools are used.
On the one hand, the pure rigidification of some regions turns out to be amazingly beneficial, as it maintains the mesh in its initial (presumably good) state in chosen critical regions. It shows particularly efficient near objects with complex geometry features or when the mesh is adapted near a body (see Subsection 6.5, page 27). Some regions can even be maintained completely motionless. It also important to note that it is not necessary to solve the elasticity equation in fully rigidified or motionless areas: these regions do not even need to appear in the elasticity mesh, which results in an additional gain in *CPU* time and memory requirements. This strategy actually reduces the general problem of moving arbitrary geometries in a mesh to the much simpler problem of moving simpler objects (spheres, boxes, convex objects) encompassing them. However, this trick cannot be used if contacts between bodies are enabled.
On the other hand, when full rigidification is not possible, element stiffness is set inversely proportional to the shortest distance to one of the moving bodies. This enables to partly redistribute the deformation of the mesh away from the bodies, thus relieving the elements in the surrounding of the objects which are generally the ones undergoing the main damages. With all these techniques, an average gain of 20% of *CPU* time can be observed (even more in 3D) as compared to the classical methodology.

Mesh movement. Once the vertices displacements are known, vertices must be moved to their new location. One of the safest way to proceed is to use a naive dichotomous algorithm. This is safe but extremely slow. For this reason, we preferred to move the mesh in one shot, checking only once that the resulting mesh is correct. The domain is directly remeshed if some false elements have been created, without trying to find the maximal authorized displacement before reversing an element.

5.2 Optimization process

After the mesh has been moved, the new mesh is slightly optimized to enhance its quality.

Quality criterium. The quality criterium we use is the minimal altitude of the mesh, which is a good hint of the simulation degradation: slowdown of the simulation, degradation of the mesh and thus of the accuracy. A more classical and also efficient criterium Q for an arbitrary element K with edges $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ is a shape criterium, [8]:

$$Q(K) = \frac{12}{\sqrt{3}} \frac{|\,K\,|_{\mathcal{M}}}{\sum_{k=1}^{3} \ell^2_{\mathcal{M}}(\mathbf{e}_k)}$$

and index $_{\mathcal{M}}$ means that the quantities are computed in metric $\mathcal{M}$ if metric-based mesh adaptation is performed (otherwise $\mathcal{M} = \mathcal{I}_d$), see Sub-section 6.2, page 23.

Optimization. During the optimization phase, edge swaps and nodes relocations by Laplacian smoothing as described in [8] alternate. Only the elements having a quality higher than a prescribed threshold have one of their edge swapped. When mesh adaptation is used, the optimization routines must be adapted, especially for node relocation, to take the metric size prescription into consideration.

6 METRIC-BASED UNSTEADY ANISOTROPIC MESH ADAPTATION FOR *ALE* SIMULATIONS

For a given continuous function u, we denote by $\Pi_{\mathcal{H}} u$ the $\mathbb{P}^1$-interpolation of u. In this study, we start with the usual assumption in metric-based methods: *mastering the* $\mathbf{L}^p$ *norm of the* P_1**-interpolation error** *of solution field* u *is enough to control the global* **approximation error**.

6.1 Metric-based generation of anisotropic adapted meshes

The metric-based generation of anisotropic adapted meshes uses the notion of Riemannian metric space [8]. For a computational domain $\Omega \subset \mathbb{R}^d$, a Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is a spatial field that defines at any point of Ω a metric tensor $\mathcal{M}(\mathbf{x})$, *i.e.* a $d \times d$ symmetric definite positive matrix. A mesh generator can work in this specific Riemannian metric space instead of the usual Euclidean space, with a new dot product defined locally by: $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}} = \langle \mathbf{u}, \mathcal{M}\mathbf{v} \rangle$ for $(\mathbf{u}, \mathbf{v}) \in \mathbb{R}^d \times \mathbb{R}^d$. The Riemmannian metric tensor $\mathcal{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ can be seen as a mathematical object defined at each point $\mathbf{x}$ of Ω, that will prescribe a density, a set of anisotropy directions and the stretching along these directions bound to govern the generation of a new mesh. In that case, the length of edge $\mathbf{ab}$ and the volume of element K are computed by:

$$\ell_{\mathcal{M}}(\mathbf{ab}) = \int_0^1 \sqrt{\mathbf{ab}^T \cdot \mathcal{M}(\mathbf{a} + t\,\mathbf{ab}) \cdot \mathbf{ab}}\,\mathrm{d}t\,, \quad |K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}(\mathbf{x})}\,\mathrm{d}\mathbf{x}\,. \tag{10}$$

It is important to note that, in a Riemannian metric space, computing the length of a segment (*i.e.* an edge) differs from evaluating the distance between the extremities of this segment. Indeed, the shortest path between two points is not the straight line anymore, but a generally curved geodesic.

The main idea of metric-based mesh adaptation is to generate *a unit mesh* in the prescribed Riemannian metric space, *i.e.* a mesh of $\Omega \subset \mathbb{R}^3$ such that each edge has a unit length and each element is regular for $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$: $\forall \mathbf{e},\ \ell_{\mathcal{M}}(\mathbf{e}) = 1$ and $\forall K,\ |K|_{\mathcal{M}} = \frac{\sqrt{2}}{12}$. Eventually, a metric $\mathcal{M}$ is also characterized by its complexity $N\big[\mathcal{M}\big]$, which is an indication of the number of vertices of a unit mesh in this metric. The complexity operator $N\big[\cdot\big]$ acting on metrics is defined by:

$$N\big[\mathcal{M}\big] = \int_\Omega \big\{\det\big(\mathcal{M}(\mathbf{x})\big)\big\}^{\frac{1}{2}}\,\mathrm{d}\mathbf{x}\,.$$

6.2 Metric-based multi-scale anisotropic mesh adaptation

The considered problem of mesh adaptation consists in finding the mesh $\mathcal{H}$ of Ω that minimizes the linear interpolation error $u - \Pi_{\mathcal{H}} u$ controlled in $\mathbf{L}^p$ norm. If posed directly in terms of discrete meshes, this problem is intractable. To overpass this difficulty, a new framework has been designed in [16] which allows to pose the problem in terms of Riemannian metrics. The problem of the minimization of the interpolation error in $\mathbf{L^p}$ norm under a given complexity N constraint reformulated with metrics can be solved analytically. It leads to the following $\mathbf{L^p}$ optimal metric operator $\mathcal{M}^{\Omega}_{\mathbf{L^p}}\left[\cdot\right]$, here applied on sensor u:

$$\mathcal{M}^{\Omega}_{\mathbf{L^p}}\left[u\right] = D_{\mathbf{L^p}}\ \left(\det\left|\, H^{\Omega}\left[u\right]\,\right|\right)^{-\frac{1}{2p+d}}\ \left|\, H^{\Omega}\left[u\right]\,\right| \ \text{with}\ D_{\mathbf{L^p}} = N^{\frac{2}{d}}\left(\int_{\Omega}\left(\det\left|\, H^{\Omega}\left[u\right](\mathbf{x})\,\right|\right)^{\frac{p}{2p+d}}\,\mathrm{d}\mathbf{x}\right)^{-\frac{2}{d}}. \tag{11}$$

$H^{\Omega}\left[\cdot\right]$ is the hessian operator on domain Ω. The metrics constructed with low-p norms are more sensitive to weaker variations of the solution whereas the $\mathbf{L}^{\infty}$ norm mainly concentrates on the strongest singularities. Now, to obtain an optimal discrete mesh $\mathcal{H}_{\mathbf{L}^p}$, it is sufficient to generate a unit mesh with respect to $\mathcal{M}_{\mathbf{L}^p}$ thanks to Relation (10). This process enables to recover a global second-order asymptotic mesh convergence for the considered sensor variable u, even if singularities are present.

6.3 The fixed-point mesh adaptation algorithm for unsteady flows.

The unsteady fixed-point mesh adaptation algorithm [1] is used to avoid generating a new mesh at each solver iteration (which would imply that a remesher is coded inside the solver). It is also an answer to the lag problem occurring when computing the solution t^n and accordingly adapt the mesh at each time step. Indeed, by doing this, the mesh is always late compared to the solution as it is not adapted for the displacement of the solution between t^n and t^{n+1}.
The fixed-point mesh adaptation algorithm works as follow. The simulation time interval $[0, T]$ is cut into m identical adaptation sub-intervals of size ΔT. The solution is computed on the first adaptation sub-interval $[0, \Delta T]$ and sampled at regular time intervals of length δt during this period. We now have n samples of the solution between 0 and ΔT. The metric associated

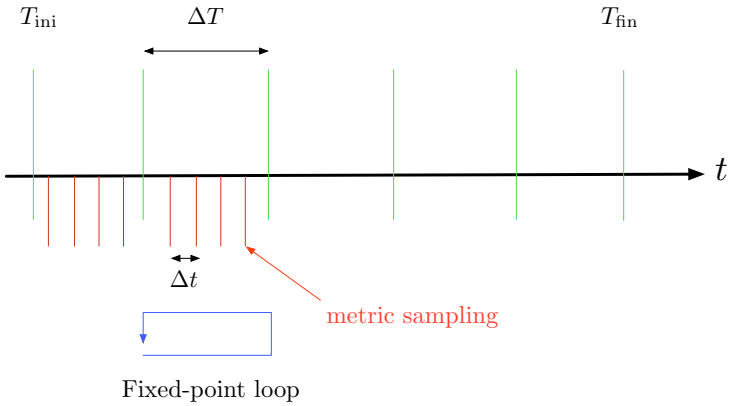

Figure 9: Sampling of the solution during each adaptation sub-interval and fixed-point loop.

with each of these samples is computed according to Formula (11). We now have n metric fields. These metric fields are intersected to get a single resulting metric that will prescribe to each vertex the maximal acceptable size guaranteeing the control of the spatial error in $\mathbf{L^p}$ norm on the whole sub-interval. A new mesh adapted to this sub-interval is then generated and we start again the sampling procedure on the same period. We loop until we reach convergence of the couple mesh-solution for this period, hence the name "fixed-point" algorithm. When the solution has been well computed on $[0, \Delta T]$, we go to the next sub-interval $[\Delta T, 2\Delta T]$ and do the same thing. The m sub-intervals are treated the same way.
An enhanced version of this algorithm which tries to better take into account the global space-time error when adapting is currently under development, see [2].

6.4 Extension to the *ALE* framework

In this section, we want to adapt the mesh to a sensor function defined on a deforming domain. Computational domain $\Omega = \Omega(t)$ is now time-dependent and, generally, $\Omega(t^n) \neq \Omega(t^{n+1})$ between two instants t^n and t^{n+1} in $[0, T]$. The following notations will be used in the sequel:

- Ω^n and Ω^{n+1} denote the spatial domain at t^n and t^{n+1}, respectively
- $\nabla^n \left[\cdot \right]$ denotes the gradient operator[4] performed on domain Ω^n
- $H^{n+1} \left[\cdot \right]$ denotes the hessian operator performed on domain Ω^{n+1}
- $\mathcal{M}^{n+1}_{\mathbf{L^p}} \left[\cdot \right]$ denote the $\mathbf{L^p}$ optimal metric operator calculated on Ω^{n+1}.

We also note: $N\left(\mathcal{M}^{n+1}_{\mathbf{L^p}} \left[u^{n+1}\right]\right) = N^{n+1}$.

Optimal *ALE* metric. Even if $\Omega^n \neq \Omega^{n+1}$ in general, we assume that these two spatial domains can be mapped one onto the other, which means there exists a mapping $\boldsymbol{\phi}$ such that:

$$\begin{array}{rccl} \boldsymbol{\phi}: & \Omega^n & \longrightarrow & \Omega^{n+1} \\ & \mathbf{x}^n & \longmapsto & \mathbf{x}^{n+1} = \boldsymbol{\phi}\left(\mathbf{x}^n\right) \end{array}$$

and, as $\boldsymbol{\phi}$ is a diffeomorphism, we have, for any infinitesimal vector $\mathrm{d}\mathbf{x}^n \in \Omega^n$:

$$\mathrm{d}\mathbf{x}^{n+1} = \left[\nabla^n \left[\boldsymbol{\phi}\right]\left(\mathbf{x}^n\right)\right]^T \cdot \mathrm{d}\mathbf{x}^n \,. \tag{12}$$

Mapping $\boldsymbol{\phi}$ and mesh displacement field $\mathbf{d}$ are linked by the following relation:

$$\mathbf{x}^{n+1} = \boldsymbol{\phi}\left(\mathbf{x}^n\right) = \mathbf{x}^n + \mathbf{d}\left(\mathbf{x}^n\right) \Longrightarrow \nabla^n \left[\boldsymbol{\phi}\right]\left(\mathbf{x}^n\right) = \mathcal{I}_d + \nabla^n \left[\mathbf{d}\right]\left(\mathbf{x}^n\right), \ \forall\, \mathbf{x}^n \in \Omega^n$$

For the purpose of simplicity, sensor function u is assumed to be scalar, the extension to vectorial functions being straightforward. Finally, we note $\widehat{H^{n+1}}$ the hessian of u^{n+1} (computed on Ω^{n+1} transported on domain Ω^n. This mathematically writes:

$$\begin{array}{rccl} \widehat{H^{n+1}}: & \Omega^n & \longrightarrow & \mathbb{R} \\ & \mathbf{x}^n & \longmapsto & H^{n+1}\left[u^{n+1}\right]\left(\boldsymbol{\phi}\left(\mathbf{x}^n\right)\right). \end{array}$$

[4]Here, the gradient is not the Jacobian, *i.e.* for an arbitrary vector field $\mathbf{f} = (f_1, \ldots, f_d)$, its gradient matrix is $\nabla \mathbf{f} = \left(\frac{\partial f_j}{\partial x_i}\right)_{ij}$

As stated in the previous sections, no vertex is added or suppressed during the mesh movement. Consequently, the mesh complexity remains constant in time, *i.e.* the complexity of the metric field defined on Ω^n must be the same as the one of the metric field defined on Ω^{n+1}. Otherwise, the problem is not consistent.
In the sequel, the following assertion is demonstrated:

Theorem 1 (Optimal *ALE* $\mathbf{L^p}$ metric) *Let metric $\mathcal{M}_{\mathbf{L^p}}^{n,ALE}[u]$ be defined on Ω^n by:*

$$\begin{aligned}
\mathcal{M}_{\mathbf{L^p}}^{n,ALE}[u](\mathbf{x}^n) &= D_{\mathbf{L^p}}^{ALE}\left\{\det\big(\widehat{H^{n+1}}(\mathbf{x}^n)\big)\right\}^{-\frac{1}{2p+d}}\ \nabla^n\left[\boldsymbol{\phi}\right](\mathbf{x}^n)\cdot\widehat{H^{n+1}}(\mathbf{x}^n)\cdot\nabla^n\left[\boldsymbol{\phi}\right]^T(\mathbf{x}^n)\\
&= D_{\mathbf{L^p}}^{ALE}\left\{\det\big(\nabla^n\left[\boldsymbol{\phi}\right]\big)\right\}^{\frac{2}{2p+d}}\left\{\det\big(\widetilde{H^{n+1}}\big)\right\}^{-\frac{1}{2p+d}}\widetilde{H^{n+1}}\\
\textit{with}\quad \widetilde{H^{n+1}} &= \nabla^n\left[\boldsymbol{\phi}\right](\mathbf{x}^n)\cdot\widehat{H^{n+1}}(\mathbf{x}^n)\cdot\nabla^n\left[\boldsymbol{\phi}\right]^T(\mathbf{x}^n)\\
\textit{and}\quad D_{\mathbf{L^p}}^{ALE} &= \big(N^{n+1}\big)^{\frac{2}{d}}\left(\int_{\Omega^{n+1}}\left\{\det\big(H^{n+1}\left[u^{n+1}\right]\big)\right\}^{\frac{p}{2p+d}}\mathrm{d}\mathbf{x}^{n+1}\right)^{-\frac{2}{d}}
\end{aligned}\tag{13}$$

The following properties hold:

i) Let us assume metric $\mathcal{M}_{\mathbf{L^p}}^{n,ALE}[u]$ is used to generate a unit mesh $\mathcal{H}^n$ of Ω^n and let us denote by $\mathcal{H}^{n+1}$ the mesh of Ω^{n+1} which is the image of mesh $\mathcal{H}^n$ by mapping $\boldsymbol{\phi}$. Then, mesh $\mathcal{H}^{n+1}$ is optimal to control the interpolation error in $\mathbf{L^p}$ norm of sensor u^{n+1} on Ω^{n+1}.

ii) Metric $\mathcal{M}_{\mathbf{L^p}}^{n,ALE}[u]$ has the same complexity N^{n+1} as metric $\mathcal{M}_{\mathbf{L^p}}^{n+1}\left[u^{n+1}\right]$.

iii) There is no reason for mesh $\mathcal{H}^n$ to be optimal for the control of sensor u^n's interpolation error at t^n.

Proof of i). According to the metric-based mesh adaptation theory for steady problems, the optimal metric in $\mathbf{L^p}$ norm for u^{n+1} is $\mathcal{M}_{\mathbf{L^p}}^{n+1}\left[u^{n+1}\right]$ as defined in (11). Thus, an optimal mesh of Ω^{n+1} adapted to u^{n+1} can be built by generating a unit mesh $\mathcal{H}^{n+1}$ with respect to $\mathcal{M}_{\mathbf{L^p}}^{n+1}\left[u^{n+1}\right]$:

$$1 = \left(\mathbf{e}^{n+1}\right)^T\cdot\mathcal{M}_{\mathbf{L^p}}^{n+1}\left[u^{n+1}\right]\cdot\mathbf{e}^{n+1},\quad \text{for each } \mathbf{e}^{n+1} \text{ of mesh } \mathcal{H}^{n+1}\,. \tag{14}$$

For any arbitrary edge $\mathbf{e}^n$ of $\mathcal{H}^n$ having $\mathbf{e}^{n+1}$ as image by $\boldsymbol{\phi}$ in $\mathcal{H}^{n+1}$, we write:

$$\widehat{\mathbf{e}^{n+1}}\left(\mathbf{x}^n\right) = \mathbf{e}^{n+1}\left(\,\boldsymbol{\phi}(\mathbf{x}^n)\,\right) = \left[\nabla^n\left[\boldsymbol{\phi}\right](\mathbf{x}^n)\right]^T\mathbf{e}^n\left(\mathbf{x}^n\right)\,.$$

As we are only interested in controlling the prevailing term of the interpolation error, we can use the above relation, which is true at first order, in the demonstration.
The idea of this proof is to unravel how Condition (14) writes when transposed onto mesh $\mathcal{H}^n$.

For any arbitrary edge $\mathbf{e}^n$ of $\mathcal{H}^n$ having $\mathbf{e}^{n+1}$ as image in $\mathcal{H}^{n+1}$, we write, using the definition of operator $\mathcal{M}^{n+1}_{\mathbf{L^p}}\left[\cdot\right]$ deduced from Relation (11):

$$
\begin{aligned}
1 &= \Big[\mathbf{e}^{n+1}\left(\boldsymbol{\phi}(\mathbf{x}^n)\right)\Big]^T \cdot \mathcal{M}^{n+1}_{\mathbf{L^p}}\left[u^{n+1}\right]\left(\boldsymbol{\phi}(\mathbf{x}^n)\right) \cdot \mathbf{e}^{n+1}\left(\boldsymbol{\phi}(\mathbf{x}^n)\right) \\
&= \Big[\widehat{\mathbf{e}^{n+1}}(\mathbf{x}^n)\Big]^T \cdot \mathcal{M}^{n+1}_{\mathbf{L^p}}\left[u^{n+1}\right]\left(\boldsymbol{\phi}(\mathbf{x}^n)\right) \cdot \widehat{\mathbf{e}^{n+1}}(\mathbf{x}^n) \\
&= \left(\Big[\nabla^n\left[\boldsymbol{\phi}\right](\mathbf{x}^n)\Big]^T \cdot \mathbf{e}^n(\mathbf{x}^n)\right)^T \cdot \mathcal{M}^{n+1}_{\mathbf{L^p}}\left[u^{n+1}\right]\left(\boldsymbol{\phi}(\mathbf{x}^n)\right) \cdot \left(\Big[\nabla^n\left[\boldsymbol{\phi}\right](\mathbf{x}^n)\Big]^T \cdot \mathbf{e}^n(\mathbf{x}^n)\right) \\
&= \Big[\left(\mathbf{e}^n\right)(\mathbf{x}^n)\Big]^T \cdot \Bigg\{\left(N^{n+1}\right)^{\frac{2}{d}} \left(\int_{\Omega^{n+1}} \Big\{\det\big(H^{n+1}\left[u^{n+1}\right]\big)\Big\}^{\frac{p}{2p+d}} \mathrm{d}\mathbf{x}^{n+1}\right)^{-\frac{2}{d}} \\
&\qquad\qquad \times \quad \Big\{\det\big(\widehat{H^{n+1}}\big)\Big\}^{-\frac{1}{2p+d}} \ \nabla^n\left[\boldsymbol{\phi}\right] \cdot \widehat{H^{n+1}} \cdot \nabla^n\left[\boldsymbol{\phi}\right]^T\Bigg\} \cdot \mathbf{e}^n(\mathbf{x}^n)\,.
\end{aligned}
$$

If we create a unit mesh of Ω^n with respect to metric $\mathcal{M}^{n,ALE}_{\mathbf{L^p}}[u]$, the mesh generator will enforce:

$$
\Big[\mathbf{e}^n(\mathbf{x}^n)\Big]^T \cdot \mathcal{M}^{n,ALE}_{\mathbf{L^p}}(\mathbf{x}^n) \cdot \mathbf{e}^n(\mathbf{x}^n) = 1, \quad \text{for all edge } \mathbf{e}^n \text{ of } \mathcal{H}^n\,.
$$

Rewritting the above calculus upside down, we get the following implication:

$$
\Big[\mathbf{e}^n(\mathbf{x}^n)\Big]^T \cdot \mathcal{M}^{n,ALE}_{\mathbf{L^p}}(\mathbf{x}^n) \cdot \mathbf{e}^n(\mathbf{x}^n) = 1 \Longrightarrow \Big[\widehat{\mathbf{e}^{n+1}}\Big]^T \cdot \mathcal{M}^{n+1}_{\mathbf{L^p}}\left[u^{n+1}(\,\boldsymbol{\phi}(\mathbf{x}^n)\,)\right] \cdot \widehat{\mathbf{e}^{n+1}} = 1\,.
$$

Therefore, the deformed mesh is unit for the optimal metric associated with sensor u^{n+1}, meaning that it is optimal to control the interpolation error in $\mathbf{L^p}$ norm of the sensor at t^{n+1}. □

Proof of ii). Using $\det(\alpha\,\cdot) = \alpha^d \det(\cdot)$ for any scalar α, we get:

$$
\begin{aligned}
N\left(\mathcal{M}^{n,ALE}_{\mathbf{L^p}}\right) &= \int_{\Omega^n} \Big\{\det\big(\mathcal{M}^{n,ALE}_{\mathbf{L^p}}(\mathbf{x}^n)\big)\Big\}^{\frac{1}{2}}\,\mathrm{d}\mathbf{x}^n \\
&= \left(D^{ALE}_{\mathbf{L^p}}\right)^{\frac{d}{2}} \left(\int_{\Omega^n} \Big\{\det\big(\nabla^n\left[\boldsymbol{\phi}\right](\mathbf{x}^n)\big)\Big\} \Big\{\det\big(\widehat{H^{n+1}}(\mathbf{x}^n)\big)\Big\}^{-\frac{d}{2(2p+d)}} \Big\{\det\big(\widehat{H^{n+1}}(\mathbf{x}^n)\big)\Big\}^{\frac{1}{2}}\,\mathrm{d}\mathbf{x}^n\right) \\
&= N^{n+1}\left(\int_{\Omega^n} \Big\{\det\big(\nabla^n\left[\boldsymbol{\phi}\right]\big)\Big\} \Big\{\det\big(\widehat{H^{n+1}}(\mathbf{x}^n)\big)\Big\}^{\frac{p}{2p+d}}\,\mathrm{d}\mathbf{x}^n\right)^{-1} \\
&\qquad\qquad \times \left(\int_{\Omega^n} \Big\{\det\big(\nabla^n\left[\boldsymbol{\phi}\right](\mathbf{x}^n)\big)\Big\}^{\frac{d}{2p+d}} \Big\{\det\big(\widehat{H^{n+1}}(\mathbf{x}^n)\big)\Big\}^{\frac{p}{2p+d}}\,\mathrm{d}\mathbf{x}^n\right) \\
&= N^{n+1} \\
&= N\left(\mathcal{M}^{n+1}_{\mathbf{L^p}}\left[u^{n+1}\right]\right)
\end{aligned}
$$

The *ALE* fixed-point algorithm. Only few things need to be modified to extend the fixed-point algorithm to *ALE* simulations. The simulation time interval $[0, T]$ is still cut into m identical sub-intervals of length ΔT. The moving mesh simulation is run on the first adaptation sub-interval and sampled at regular time intervals of length Δt. Now that we have our n samples of the solution between 0 and ΔT, we compute their metrics doing as if these solutions had been computed on the initial mesh and we locally correct the metric . The metric is actually no more associated with a fixed location in space like in the fixed mesh case, but it is rather attached to the moving vertices. These metric fields are intersected like in the fixed-mesh case and a mesh adapted for this period of time is generated. We are now sure that the moved adapted mesh at t^{n+1} is at least unity for $\mathcal{M}^{n+1}$. The rest of the algorithm does not change: this procedure is repeated inside the current sub-interval until convergence. The other sub-intervals are handled the same way.

6.5 Numerical results

Pitching NACA 0012 airfoil (AgardCT5). Figure 10 shows the fixed-point mesh adaptation algorithm applied to a pitching NACA airfoil the angle of attack AoA of which is prescribed analytically by:

$$AoA(t) = AoA(t_0) + \max(AoA)\,\sin(p\,t),\ \text{with}\ \begin{cases} AoA(t_0) & = 0.016^\circ \\ \max(AoA) & = 2.51^\circ \\ p & = 0.1628\ \text{rad}.s^{-1}\,. \end{cases}$$

The period of the movement is $T = 2\Pi/p = 38.5945\,s$. The inflow Mach is 0.755. The simulation is of spatial order 2, with a V4 scheme. The RKSSP(3,3) scheme has been chosen for the temporal discretization. While the angle of attack increases to its maximum, the upper shock wave moves towards the trailing edge and becomes sharper. On the contrary, the shock located on the intrados moves toward the leading edge and its amplitude diminishes. Once the maximum value of the angle of attack has been reached, it decreases and the shock wave located on the extrados moves back to the leading edge while its amplitude is dropping. The intrados shock wave moves to the trailing edge and becomes sharper and sharper. The same phenomenon occurs for the second-half of the period.
We have stiffened all the elements located inside a disc containing the airfoil. All the vertices located inside this rigid disc have therefore the same pitching movement as the NACA. First, the rigidification process allows to preserve the quality of the initial mesh and reduces to its bare minimum the *CPU* time devoted to mesh movement and optimization. Second, in the context of anisotropic mesh adaptation, it enables to get rid of the difficulty of moving a mesh containing highly stretched elements. The movie associated with this simulation shows how the shock wave always evolves inside an adapted strip-shaped area of the mesh.
The adaptation is anisotropic and the density of the flow is taken as sensor. While the angle of attack increases toward an extremum, the adaptation process naturally tends to privilege more and more the side of the airfoil on which the shock wave becomes sharper. On the contrary, when the angle of attack approaches zero, the flow is nearly symmetric and both shocks have a similar amplitude. Therefore, the adaptation effort is balanced between the intrados and the extrados. Finally, note that in rigidified and motion less areas, *ALE* metric $\mathcal{M}_{\mathbf{L^p}}^{i,ALE}\left[\hat{u}^k\right]$

is exactly $\mathcal{M}^k_{\mathbf{L^P}}\left[u^k\right]$, with $k \in [0, n_i - 1], i \in [0, m - 1]$ and u^k a sample of the solution between t^i and t^{i+1}, as the sensor function u^k has not been deformed by the transport on mesh at t^i.

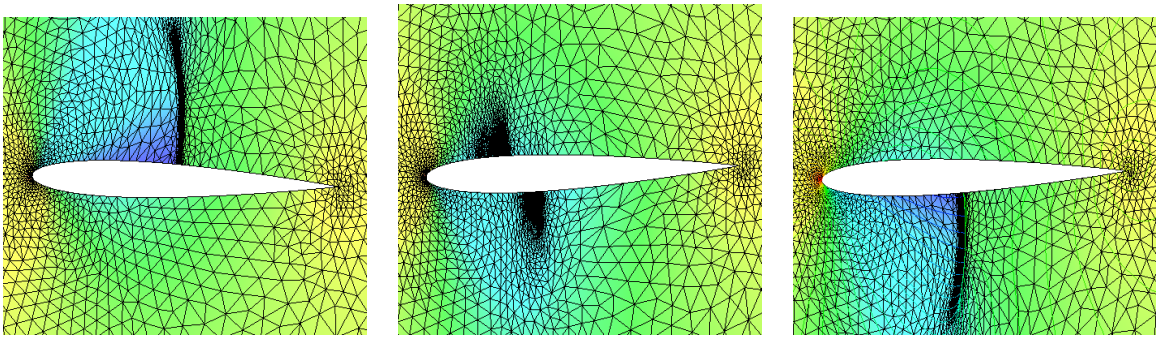

Figure 10: Anisotropic unsteady adaptation around a pitching NACA 0012 airfoil when the area surrounding the NACA is rigidified. Left at $\frac{T}{4}$, middle at $\frac{T}{2}$, right at $\frac{3T}{4}$.

Blast test case. This unsteady adaptation algorithm was tested on a two-dimensional blast test case proposed by Lohner [15]. A very strong shock wave at Mach 10 impacts a rectangular object, which is blown up. The object is first maintained fixed at the lower right corner and released when the vertical speed exceeds a given threshold. The results obtained by performing an isotropic adaptation on the density of the flow are shown in Figure 11.
First, the *ALE* formulation of the swap turns out to be very useful in this simulation to handle the mesh shearing between the ground and the bottom of the object when it is released. Second, and it is much more striking on the movie, the *ALE* fixed-point algorithm really enable to take the mesh movement into account in the adaptation process. Indeed, the reflected shock wave not only evolves inside the adapted strip-shaped area, but the strip-shaped area itself moves toward the shock wave.

7 CONCLUSION

There are two main new things in this paper. First, we have described a new variable-topology *ALE* scheme, which enables the use of the very powerful swap operation in moving mesh simulations where it was forbidden by the classical *ALE* framework. The conservativity and efficiency of this scheme have been illustrated on a turbo-machinery simple test case. Secondly, we have generalized the so-called fixed-point algorithm used in unsteady mesh adaptation to moving mesh simulations. Again, the feasibility and efficiency of the method have been shown on two test cases.
Future investigations will focus on the validation of the new changing connectivity *ALE* formulation, both in terms of accuracy and *CPU* time. The new *ALE* metric-based adaptation methodology will be further tested, first on analytical examples and next on flows around complex three-dimensional moving geometries.

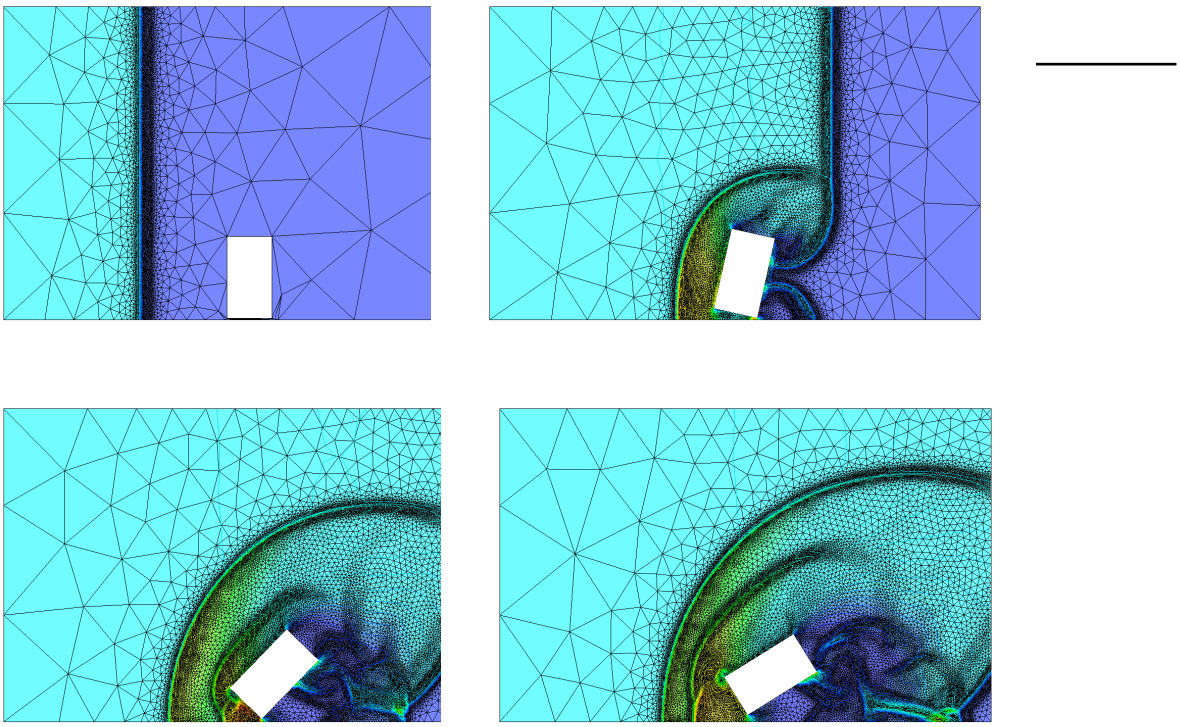

Figure 11: Isotropic adaptation with the *ALE* fixed-point algorithm on a blast test case. The solution evolves inside the adapted region and the adapted region moves towards the solution.

REFERENCES

[1] F. Alauzet, P.J. Frey, P.-L. George and B. Mohammadi, 3D Transient Fixed Point Mesh Adaptation for Time-Dependent Problems: Application to CFD Simulations, *J. Comp. Phys.*, **222** 592–623 (2007).

[2] F. Alauzet and G. Olivier, An Linf-Lp Space-Time Anisotropic Mesh Adaptation Strategy for Time-Dependent Problems, *ECCOMAS Proceeding*, Lisbonne (2010).

[3] T. Baker and P Cavallo, Dynamic Adaptation for Deforming Tetrahedral Meshes, *AIAA paper*, **1999-3253** (1999).

[4] P. Batten, N. Clarke, C. Lambert and D.M. Causon, On the Choice of Wavespeeds for the HLLC Riemann Solver, *SIAM J. Sci. Comp.*, **18**, 1553–1570 (1997).

[5] P.H Cournede, B. Koobus and A. Dervieux, Positivity Statements for a Mixed-Element-Volume Scheme on Fixed and Moving Grids, *Europ. J. Comput. Mech.*, **15**, 767–798 (2006).

[6] C. Debiez and A. Dervieux, Mixed-Element-Volume MUSCL Methods with Weak Viscosity for Steady and Unsteady Flow Calculations, *Comput. Fluids.*, **29**, 89–118 (1999).

[7] L. Ferracina and M.N Spijker, Stepsize Restrictions for the Total-Variation-Diminishing Property in General Runge-Kutta Methods, *Appl. Numer. Math.*, **40**, 265–279 (2005).

[8] P.J. Frey and P.L. George, Mesh Generation. Application to finite elements, *ISTE Ltd and John Wiley & Sons*, **2nd edition** (2008)

[9] S. Gottlieb and C. Shu, Total Variation Diminishing Runge-Kutta Schemes, *Math. Comp.*, **67**, 73–85 (1998).

[10] H. Guillard and C. Farhat, On the Significance of the Geometric Conservation Law for Flow Computations on Moving Meshes, *Comput. Meth. Appl. Mech. Engrg.*, **190**, 1467–1482 (1999).

[11] B. Koobus and C. Farhat, Second-order Time-Accurate and Geometrically Conservative Implicit Schemes for Flow Computations on Unstructured Dynamic Meshes, *Comput. Meth. Appl. Mech. Engrg.*, **170**, 103–129 (1999).

[12] J. F. B. M. Kraaijevanger, Contractivity of Runge-Kutta Methods, *BIT Numer. Math.*, **31**, 482–528 (1991).

[13] M. Kucharik, M. Shashkov, Extension of Efficient, Swept-Integration-Based Conservative Method for Meshes with Changing Connectivity, *Int. J. Numer. Meth. Fluids*, **56**, 1359–1365 (2007).

[14] M. Lesoinne and C. Farhat, Geometric Conservation Laws for Flow Problems with Moving Boundaries and Deformable Meshes and their Impact on Aeroelastic Computations, *Comput. Meth. Appl. Mech. Engrg.*, **134**, 71–90 (1996).

[15] R. Löhner, Adaptive Remeshing for Transient Problems, *AIAA Journal*, **75** 195–214 (1989).

[16] A. Loseille and F. Alauzet, Continuous Mesh Framework. Part I: Well-Posed Continuous Mesh and Interpolation Error Models, *SIAM J. Numer. Anal.*, accepted for publication (2009).

[17] L.G. Margolin and M. Shashkov, Remapping, Recovery and Repair on a Staggered Grid, *Comput. Meth. Appl. Mech. Engrg.*, **193** 4139–4155 (2004).

[18] D. Mavriplis and Z. Yang, Construction of the Discrete Geometric Conservation Law for High-Order Time Accurate Simulations on Dynamic Meshes, *J. Comp. Phys.*, **213**, 557–573 (2006).

[19] M. Meriaux and S. Piperno, Méthodes en Maillage Mobiles Auto-adaptatifs pour des Syteèmes Hyperboliques en Une et Deux Dimensions d'Espace, *Phd thesis* (2005).

[20] B. Nkonga, On the Conservative and Accurate CFD Approximations for Moving Meshes and Moving Boundaries, *J. Comp. Phys.*, **190**, 1801–1826 (2000).

[21] B. Nkonga and H. Guillard, Godunov Type Methods on Non-Structured Meshes for Three-Dimensional Moving Boundary Problems, *RR INRIA*, **1883**, (1993).

[22] C.W. Shu and S. Osher, Efficient Implementation of Essentially Non-oscillatory Shock-Capturing Schemes, *J. Comp. Phys.*, **77**, pp 439–471(1988).

[23] R. Spiteri and S. Ruuth, A New Class of Optimal High-Order Strong-Stability-Preserving Time Discretization Methods, *SIAM J. Numer. Anal.*, **40**, 469–491 (2003).

[24] P.D. Thomas and C.K. Lombard, Geometric Conservation Law and its Application to Flow Computations on Moving Grids, *AIAA Journal*, **17**, 1030–1037 (1979).

[25] V. Venkatakrishnan and D. Mavriplis, Implicit Method For the Computation Of Unsteady Flows On Unstructured Grids, *ICASE Report*, **95-60** (1995).

[26] Z. Yang and D. Mavriplis, Unstructured Dynamic Meshes with Higher-order Time Integration Schemes for the Unsteady Navier-Stokes Equations, *AIAA Paper*, **2005-1222** (2005).

[27] Z. Yang and D. Mavriplis, Higher-order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes, *AIAA Journal*, **45**, 138–150 (2007).