
Efficient moving mesh technique using generalized swapping

F. Alauzet¹

INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay, France Frederic.Alauzet@inria.fr

Summary. Three-dimensional real-life simulations are generally unsteady and involve moving geometries. Industries are currently still very far from performing such simulations on a daily basis, mainly due to the robustness of the moving mesh algorithm and their extensive computational cost. The proposed approach is a way to improve these two issues. This paper brings two new ideas. First, it demonstrates numerically that moving three-dimensional complex geometries with large displacements is feasible using only vertex displacements and mesh-connectivity changes. This is new and presents several advantages over usual techniques for which the number of vertices varies in time. Second, most of the CPU time spent to move the mesh is due to the resolution of the mesh deformation algorithm to propagate the body displacement inside the volume. Thanks to the use of advanced meshing operators to optimize the mesh, we can reduce drastically the number of such resolutions thus impacting favorably the CPU time. The efficiency of this new methodology is illustrated on numerous 3D problems involving large displacements.

Key words: Moving mesh, mesh deformation algorithm, dynamic mesh, topology change, swapping, local reconnection, elasticity equation, large displacement

1 Introduction

The growing expectations of the industrial world for simulations involving moving geometries have given a boost to this research field for the last decades. Moving mesh simulations are now used in many research fields: ballistics, biomedical, aeronautics, blast studies, turbo-machinery, transports, etc. These simulations, which combine the difficulties associated with unsteadiness, mesh movement and fluid-structure coupling, are generally hard to perform and very costly in terms of CPU time.

Three leading methodologies have been designed in the literature to handle geometry displacements during numerical simulations: body-fitted approach [22], Chimera method [7] and immersed/embedded boundary methods [20]. Each of them has its own strengths and weaknesses. In this work, we consider the first class of method where the time-evolving computational domain is treated with a *body-fitted approach*, which means that the computational mesh follows time-evolving geometries in their movement. In other words, dynamic bodies have to be moved inside

the computational mesh using a deformation algorithm. Arbitrary-Lagrangian-Euler (ALE) numerical schemes are then considered to take into account vertices displacement inside the flow solver.

This paper focuses on improving moving mesh techniques for real-life three-dimensional problems involving moving geometries potentially undergoing large displacements. Only simplicial unstructured meshes are considered mainly because several fully-automatic and robust softwares are available to generate such meshes [12, 15, 18]. Another reason is that our final objective is to use highly anisotropic metric-based mesh adaptation [1, 3] for moving mesh simulations.

Problematics and state-of-the-art

Unfortunately, for body-fitted simulations, the fixed-topology constraint imposed by the classical ALE framework limits considerably the efficiency of moving mesh techniques. If the problem induces large displacements of the geometry, the resulting mesh distortions can adversely affect the accuracy and stability of the numerical solution process. approach unpractical. Classical approaches generally fail to achieve the mesh displacement leaving the engineer with no solution or untrustworthy ones.

So far, two different methods are usually adopted to handle large displacements moving mesh simulations.

The first one consists in moving the mesh as much as possible, keeping the topology fixed. Mesh is moved and equations are solved in a fully ALE manner until the quality of the mesh becomes too bad. The domain is then globally re-meshed with the current geometry configuration and the current solution is interpolated on the newly generated mesh. The ALE computation can then resume, see for instance [6, 14]. One asset of this method is that meshing and solver aspects are entirely decoupled. However, the number of global re-meshings can become very large especially when elements undergo shearing. These very frequent global re-meshings can lead to prohibitive CPU times and can even sometimes fail in three dimensions.

The second approach aims at maintaining the best possible mesh quality while moving using several local re-meshing operations such as vertex addition, vertex collapsing, connectivity changes and vertex displacements. This strategy is extremely robust and enables to maintain a good mesh quality throughout the simulation. However, local re-meshing using various meshing operations requires fully dynamical data structures inside the solver. Besides, this approach involves a large number of solution interpolations as they are performed on the fly after each mesh modification. Hence, the numerical method does not fully comply with the ALE framework. Illustrations of this method can be found in [8, 10].

Our approach

We have adopted a new approach for moving mesh simulations as compared to existing methodologies. The guideline to design our strategy is to propose an enhanced moving mesh method which is compatible with the ALE framework and able to handle anisotropic adapted meshes.

First, we propose to couple the mesh deformation algorithm with local mesh optimization using only vertex smoothing and generalized swapping, see Figure 1. This choice is due to the powerfulness of this mesh-topology-change operator which is especially appropriate to handle shears and large deformation movements. With

this strategy, only the number of tetrahedra and edges varies in 3D. The number of vertices remains fixed. The moving mesh algorithm can be summarized as follows:

1. solve the mesh deformation problem
2. optimize the mesh using vertex smoothing and local reconnection
3. move all the vertices.

The mesh deformation algorithm is based on the resolution of a linear elasticity system even if the proposed approach is compatible with other algorithms.

Second, the use of an efficient local mesh optimization process enables to maintain a good mesh quality throughout the geometry displacement. Therefore, it becomes conceivable to significantly reduce the number of requested mesh deformation problem resolutions impacting favorably the CPU time. In this context, as the mesh deformation algorithm is solved for a large time frame, it is of main interest to supply mesh vertices with non-uniform velocity curved trajectories.

Under these assumptions, we numerically demonstrate that *moving 3D complex geometries with large displacements is possible using only vertex displacements and connectivity changes*. Vertex addition or deletion is not mandatory.

This paper begins with a description of the mesh deformation method based on the linear elasticity. Next, it presents the considered mesh optimization operators. Section 4 is dedicated to our new changing-topology moving mesh algorithm and the modifications to enhance its efficiency. This paper ends with numerous 3D examples which demonstrate the powerfulness of this novel approach.

2 Linear elasticity mesh deformation method

For body-fitted simulations, the whole mesh must be deformed to follow the geometry movement while keeping a geometric conforming mesh. Two alternatives are generally considered: PDE based or interpolation mesh deformation methods.

Interpolation methods consist in interpolating deformations from boundary points to other points in space. These approaches are generally based on the radial basis function (RBF) interpolation method or variations of it [9, 17].

As regards PDE based methods, several models have been considered: Laplacian-based method [16], spring-analogy methods [5] and linear-elasticity-based method [4, 21, 23]. As stated in [23], and despite various attempts to improve them, spring-analogy techniques have shown less robust than elasticity-based methods, and also tend to deteriorate the quality of the mesh at a faster rate. Besides, linear-elasticity-based techniques enable vertices to go round moving bodies instead of bumping into them as compared to Laplacian-like approaches, thus playing in favor of quality.

Consequently, the linear-elasticity mesh deformation approach has been retained, even if all deformation methods can be combined with the changing topology approach proposed in this work. The inner vertices movement is obtained by solving an *elasticity-like equation* with a \mathbb{P}_1 Finite Element Method (FEM):

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

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 the Hooke's

law for isotropic homogeneous medium. So far, vertices located on domain boundaries cannot slip on these boundary surfaces to simplify the moving mesh algorithm. This is more restrictive thus making the mesh displacement a harder task. Therefore, only Dirichlet boundary conditions are used and the displacement of vertices located on the domain boundary is strongly enforced in the linear system. The FEM system is solved by a Conjugate Gradient algorithm coupled with an LU-SGS pre-conditioner.

2.1 Elasticity-dedicated mesh strategy

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/Navier-Stokes equations. The displacement computed on the elasticity-dedicated mesh is then interpolated on the finer adapted CFD mesh [2] using a \mathbb{P}_2 -Lagrangian scheme, which is sufficiently accurate considering the intrinsic smoothness of the solution of the above elasticity equation. This is mandatory because, as already said, we intend to couple moving mesh simulations and anisotropic mesh adaptation. But, it would be very hard (or impossible) to solve the FEM elasticity linear system on an anisotropic mesh adapted for the CFD resolution due to the creation of an artificial extra-stiffness hindering the convergence. This is also the case for viscous meshes when high aspect ratio structured elements are used in the boundary layer region.

Furthermore, this elasticity-dedicated mesh strategy enables a significant reduction of the CPU time and memory requirement. Indeed, it reduces the size of the elasticity linear system to be solved and it avoids to restructure the FEM elasticity matrix each time a connectivity change is performed in the CFD mesh. Even if the elasticity mesh must be moved and sometimes optimized along with the computational mesh, the additional cost is negligible as compared to the gain in terms of CPU for the linear elasticity resolution.

2.2 Local material properties

Another 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 born by each element. Following [21], the way the Jacobian of the transformation from the reference element to the current element is accounted for in the FEM matrix assembly is modified. Classically, the \mathbb{P}_1 FEM formulation of the linear elasticity matrix leads to the evaluation of quantities of the form:

$$\int_K s \frac{\partial \varphi_J}{\partial x_k} \frac{\partial \varphi_I}{\partial x_l} d\mathbf{x} = s |K| \frac{\partial \varphi_J}{\partial x_k} \frac{\partial \varphi_I}{\partial x_l}$$

where s is either λ , μ or $\lambda + 2\mu$ and $|K|$ is the volume of tetrahedron K . The above quantity is replaced by:

$$\int_K s \left(\frac{|\hat{K}|}{|K|} \right)^\chi \frac{\partial \varphi_J}{\partial x_k} \frac{\partial \varphi_I}{\partial x_l} d\mathbf{x} = s |K| \left(\frac{|\hat{K}|}{|K|} \right)^\chi \frac{\partial \varphi_J}{\partial x_k} \frac{\partial \varphi_I}{\partial x_l},$$

where $\chi > 0$ is the stiffening power and \hat{K} is the reference element. This technique comes to locally multiply λ and μ by a factor proportional to $|K|^{-\chi}$. χ determines the degree by which smaller elements are rendered stiffer than larger ones.

2.3 Rigidifying regions

Finally, in some situations - geometries involving tiny complex details, boundary layer meshes, etc. - we have found useful to rigidify certain regions. This can greatly simplify the moving mesh problem and reduces significantly the size of the elasticity system. A first strategy consists of rigidifying a certain number of elements layers around moving objects or a given region. Elements belonging to a rigid layer are moved in a completely rigid manner using the angular and the translation displacement of the associated body. A second strategy aims at reducing 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.

3 Mesh optimization

Mesh quality tends to decrease while the mesh is moving, thus elements may become inverted or highly skewed impacting negatively the stability, efficiency and accuracy of the flow solver. Therefore, it is advantageous to perform frequent mesh optimizations to fight against this adverse effect. For 3D adapted meshes, the quality is measured in terms of element shapes by the following quality function [11]:

$$Q_{\mathcal{M}}(K) = \frac{\sqrt{3}}{216} \frac{\left(\sum_{i=1}^6 \ell_{\mathcal{M}}^2(\mathbf{e}_i) \right)^{\frac{3}{2}}}{|K|_{\mathcal{M}}} \in [1, +\infty], \quad (2)$$

where $\ell_{\mathcal{M}}(\mathbf{e})$ and $|K|_{\mathcal{M}}$ are edge length and element volume in metric \mathcal{M} . Metric \mathcal{M} is a 3×3 symmetric positive definite tensor prescribing element sizes, anisotropy and orientations to the mesh generator. $Q_{\mathcal{M}}(K) = 1$ corresponds to a perfectly regular element while a high value of $Q_{\mathcal{M}}(K)$ indicates a nearly degenerated element. For non-adapted meshes, the identity matrix \mathcal{I}_3 is chosen as metric tensor.

3.1 Vertex smoothing

Vertex smoothing consists in relocating each vertex inside its ball of elements, *i.e.*, the set of elements having P_i as vertex. For each tetrahedron K_j of the ball of P_i , the face of K_j opposite to vertex P_i , denoted F_j , proposes a new optimal position:

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

where G_j is the gravity center of face F_j and \mathbf{n}_j is the inward normal to F_j . $h_{\mathcal{M}}(\mathbf{n})$ denotes the size prescribed by metric \mathcal{M} in direction \mathbf{n} which is defined by:

$$h_{\mathcal{M}}(\mathbf{n}) = \frac{\|\mathbf{n}\|}{\ell_{\mathcal{M}}(\mathbf{n})} \implies P_j^{opt} = G_j + \sqrt{\frac{2}{3}} \frac{\mathbf{n}_j}{\ell_{\mathcal{M}}(\mathbf{n}_j)}.$$

Other formulations for P_j^{opt} can be proposed. The final optimal position P_i^{opt} is computed as a weighted average of all these optimal positions $\{P_j^{opt}\}_{K_j \supset P_i}$. If a weight coefficient depending on the quality of K_j is chosen, we set:

$$P_i^{opt} = \frac{\sum_{K_j \in \text{Ball}(P_i)} \max(Q_{\mathcal{M}}(K_j), Q_{max}) P_j^{opt}}{\sum_{K_j \in \text{Ball}(P_i)} \max(Q_{\mathcal{M}}(K_j), Q_{max})},$$

and Q_{max} is a parameter to be defined. In this way, an element of the ball is all the more influent as 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. Otherwise, successive relaxed positions $P_i^{new} = P_i + \alpha P_i^{opt}$, with progressively decreasing values of α are checked.

Mesh smoothing helps to preserve nicely shaped elements but it is really less efficient in 3D than in 2D. Moreover, it cannot solve the shearing issue. To significantly enhance mesh optimization, topology changes are absolutely needed.

3.2 Generalized swapping

In 2D, edge swapping is a rather simple topological operation which consists in flipping an edge shared by two triangles, see Figure 1, top left. This operation changes neither the number of triangles nor the number of edges of the mesh.

The direct extension of the 2D swap in three dimensions is the swap of a face shared by two tetrahedra. Let α and β be the two tetrahedra's vertices opposite to the common face $P_1P_2P_3$. Then, the swap operator consists in suppressing this face and creating the edge $e = \alpha\beta$, see Figure 1, top right. In this case, the two original tetrahedra are deleted and three new tetrahedra are created. This swap is called $2 \rightarrow 3$. The reverse operator can also be defined which consists in deleting three tetrahedra sharing such a common edge $\alpha\beta$ and creating two new tetrahedra sharing face $P_1P_2P_3$, see Figure 1, top right. This swap is called $3 \rightarrow 2$.

In fact, a generalization of this operation exists and acts on shell of tetrahedra [11]. For an internal edge $e = \alpha\beta$, the shell of e is the set of tetrahedra having e

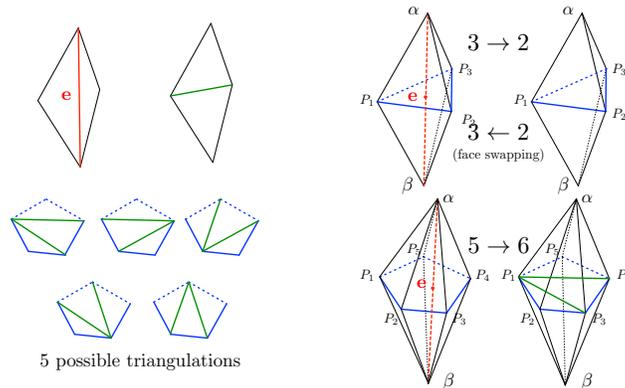


Fig. 1. Top left, the swap operation in 2D. Top right, edge swap $3 \rightarrow 2$ and face swap $2 \rightarrow 3$. Bottom left, the 5 possible triangulations of the pseudo-polygon for a shell having 5 elements. Bottom right, an example of $5 \rightarrow 6$ edge swap. For all these figures, shells are in black, old edges are in red and new edges in green.

as common edge, see Figure 1, right. From a shell of size n , a non-planar *pseudo-polygon* formed by n vertices $P_1 \dots P_n$ - shown in blue in Figure 1 - can be defined. Performing a three-dimensional swap of edge $\alpha\beta$ comes (i) to suppress edge $\alpha\beta$ and all tetrahedra of the shell, (ii) then to define a triangulation of the pseudo-polygon $P_1 \dots P_n$ and (iii) finally to create new tetrahedra by joining each triangle of the pseudo-polygon with vertices α and β . The number of possible triangulations depends on n the number of vertices of the pseudo-polygon: $N_{tri} = Cat(n-1)$ where Cat is the Catalan's number defined by: $Cat(n) = \frac{(2n-2)!}{n!(n-1)!}$. Figure 1 depicts the possible triangulations for a pseudo-polygon with $n = 5$ (bottom left) and one possible swap configuration (bottom right). The different edge swaps are generally denoted $n \rightarrow m$ where 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. Except for the swap $4 \rightarrow 4$, all the 3D swaps change the number of edges and tetrahedra of the mesh. But, the number of vertices remains constant.

The generalized swap operator is used to improve mesh quality. The following strategy is considered. Tetrahedra are treated in quality order from the worst to the best one. For each element, the four face swaps and the six edge swaps are simulated, then the best of all the configurations is selected and performed if it satisfies all the specified criteria in terms of size and quality. A key to perform efficient swaps in the moving mesh context is to authorize a slight quality degradation.

4 Changing-topology moving mesh algorithm

4.1 Basic algorithm

When the classical approach is considered, the changing-topology moving mesh algorithm is called at each solver time step (or for few solver steps) to move the mesh from t to $t + \delta t$:

Algorithm 1 Basic Changing-Topology Moving Mesh Algorithm

- Mesh deformation algorithm:
 1. $\mathbf{d}_{|\partial\Omega_h}^{body}$ = Compute bodies vertices displacements from current bodies translation \mathbf{d}^{body} and rotation $\boldsymbol{\theta}^{body}$ vectors
 2. \mathbf{d}^{els} = Solve elasticity system $(\mathbf{d}_{|\partial\Omega_h}^{body}, \delta t)$
 - Mesh optimization:
 1. \mathcal{H}^n = Swaps optimization $(\mathcal{H}^n, Q_{target}^{swap})$
 2. \mathbf{d}^{opt} = Vertices smoothing $(\mathcal{H}^n, Q_{target}^{smoothing}, Q_{max})$
 - \mathcal{H}^{n+1} = Move the mesh $(\mathcal{H}^n, \delta t, \mathbf{d} = \mathbf{d}^{els} + \mathbf{d}^{opt})$
-

For rigid bodies, the Newton-Euler equation for six-degree-of-freedom rigid body motion are considered. Each body displacement is defined by a translation \mathbf{d}^{body} and a rotation $\boldsymbol{\theta}^{body}$ vectors. These vectors are obtained either as analytical user data or as the results of a fluid-structure-interaction (FSI) problem. From these data, the displacement of each body vertex P_i , with coordinates \mathbf{x}_i , is defined by:

$$\mathbf{d}^{body}(\mathbf{x}_i) = \mathbf{d}^{body} - \mathbf{r}_i + (\mathbf{r}_i \cdot \boldsymbol{\theta}_u) \boldsymbol{\theta}_u + \sin \theta (\mathbf{r}_i \times \boldsymbol{\theta}_u) + \cos \theta (\mathbf{r}_i - (\mathbf{r}_i \cdot \boldsymbol{\theta}_u) \boldsymbol{\theta}_u),$$

where $\mathbf{r}_i = \mathbf{g}\mathbf{x}_i$ with \mathbf{g} the gravity center of the body, $\theta = \|\boldsymbol{\theta}^{body}\|$ and $\boldsymbol{\theta}_u = \frac{\boldsymbol{\theta}^{body}}{\|\boldsymbol{\theta}^{body}\|}$.

In the case of deformable bodies, the motion of each vertex on the boundary $\mathbf{d}^{body}(\mathbf{x}_i)$ is directly obtained from the FSI problem resolution.

The displacement on bodies surfaces provided by these vectors set the boundary conditions of the linear elasticity system. The resolution of the mesh deformation algorithm provides a displacement vector $\mathbf{d}^{els}(\mathbf{x}_i)$ for all mesh vertices.

To remain in the ALE framework, the mesh optimization - swapping and smoothing - is seen like a correction of the mesh deformation algorithm. To this end, the mesh smoothing phase does not optimize the current mesh but the future vertices location at $t + \delta t$ providing a motion correction $\mathbf{d}^{opt}(\mathbf{x}_i)$ for time frame $[t, t + \delta t]$. Therefore, the effective displacement of each vertex from t to $t + \delta t$ is given by:

$$\mathbf{d}(\mathbf{x}_i) = \mathbf{d}^{els}(\mathbf{x}_i) + \mathbf{d}^{opt}(\mathbf{x}_i) \quad \text{and} \quad \mathbf{x}_i(t + \delta t) = \mathbf{x}_i(t) + \mathbf{d}(\mathbf{x}_i).$$

These mesh modifications change the numerical approximation space thus they have to be taken into account in the ALE numerical scheme by correcting mesh vertices ALE velocities, see [19] for details. Furthermore, to be able to formulate the changing-topology ALE numerical scheme, the following constraint on the swap operator must be imposed: *once a swap has been performed, all edges/faces belonging to a new element of the cavity are set as blocked (i.e., cannot be swapped) until the next swap optimization step.* This constraint is not as restrictive as it may seem at first glance. This is clearly emphasized in the numerical examples section. Moreover, it has the advantage of simplifying the management of data structures.

4.2 Improving mesh deformation algorithm efficiency

Mesh deformation algorithms, here the elasticity problem, are known to be an expensive part of dynamic mesh simulations as their resolution is generally required at each solver time step (or each few solver time step). Therefore, one way to reduce the CPU time dedicated to this part is to reduce the number of such resolutions.

In our case, instead of solving the elasticity system at each flow solver time step δt , its resolution is done once for a large time frame of length Δt . There is a risk to get a less effective mesh displacement solution but, this is a worthwhile strategy if our methodology is able to handle such large displacement while preserving the mesh quality. To this end, we will take the advantage of the mesh optimization methods and we will consider accelerated velocity curved trajectories for vertices.

Accelerated velocity curved trajectory

Solving the previously described mesh deformation problem once for large time frame could be problematic in two cases:

- curved trajectories of the boundary vertices
- accelerated bodies.

Indeed, boundary vertices trajectories may intersect inner vertices ones if they have a constant velocity linear trajectory leading to invalid elements.

To enhance inner vertices path, we propose to supply each vertex with an initial speed $\mathbf{v}(t)$ and a constant acceleration \mathbf{a} thus defining an accelerated curved trajectory. During time frame $[t, t + \Delta t]$, the position and the velocity of a vertex are updated as follow:

$$\begin{aligned}\mathbf{x}(t + \delta t) &= \mathbf{x}(t) + \delta t \mathbf{v}(t) + \frac{\delta t^2}{2} \mathbf{a} \\ \mathbf{v}(t + \delta t) &= \mathbf{v}(t) + \delta t \mathbf{a}\end{aligned}$$

Prescribing a velocity and an acceleration vector to each vertex require to solve two elasticity systems. For both systems, the same matrix, thus pre-conditionner, is considered. Only boundary conditions change. If inner vertices 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 the resolution of the second linear system is really cheaper than the first one. Indeed, with a good prediction of the expected solution thanks to the solution of the first linear system, its cost is generally less than 10% of the first resolution. Now, to define the trajectory of each vertex, the velocity and acceleration are deduced from evaluated middle and final positions:

$$\begin{aligned}\Delta t \mathbf{v}(t) &= -3 \mathbf{x}(t) + 4 \mathbf{x}(t + \Delta t/2) - \mathbf{x}(t + \Delta t) \\ \frac{\Delta t^2}{2} \mathbf{a} &= 2 \mathbf{x}(t) - 4 \mathbf{x}(t + \Delta t/2) + 2 \mathbf{x}(t + \Delta t)\end{aligned}$$

In this context, it is mandatory to certify that the mesh motion remains valid for the whole time frame $[t, t + \Delta t]$.

Validity of the mesh throughout the whole displacement

After computing the trajectory of each vertex, mesh motion validity throughout the displacement time frame $[t, t + \Delta t]$ has to be analyzed as an element may become invalid in between while being valid at the beginning and at the end of its motion, see Figure 2. The considered curved trajectory provides element volume functions of the time that are sixth order polynomial. Let $K = [P_0, P_1, P_2, P_3]$ be a tetrahedron, then each vertex of K has for position in time

$$\begin{aligned}P_i(t + \delta t) &= P_i(t) + \delta t \mathbf{v}(t) + 0.5 \delta t^2 \mathbf{a} \\ &= (1 - \tau)(1 - 2\tau) P_i(t) + 4\tau(1 - \tau) P_i(t + \Delta t/2) + \tau(2\tau - 1) P_i(t + \Delta t) \\ &= (1 - \tau)^2 \mathbf{P}_{i,0} + 2\tau(1 - \tau) \mathbf{P}_{i,1} + \tau^2 \mathbf{P}_{i,2}\end{aligned}$$

with $\tau = \frac{\delta t}{\Delta t} \in [0, 1]$ and the control points of the Bezier curve are

$$\mathbf{P}_{i,0} = P_i(t), \quad \mathbf{P}_{i,2} = P_i(t + \Delta t) \quad \text{and} \quad 2\mathbf{P}_{i,1} = 4P_i(t + \Delta t/2) - P_i(t) - P_i(t + \Delta t).$$

Let us introduce the following notation to express the volume of K at any moment in the time frame:

$$\mathcal{V}_{knm} = \mathbf{P}_{0,k} \mathbf{P}_{1,k} \cdot [\mathbf{P}_{0,m} \mathbf{P}_{2,m} \times \mathbf{P}_{0,n} \mathbf{P}_{3,n}], \quad \text{with } k, m, n \in [0, \dots, 2].$$

The \mathcal{V}_{kmn} are tetrahedron volumes defined by control points vector. In particular, we have $|K(t)| = \mathcal{V}_{000}$ and $|K(t + \Delta t)| = \mathcal{V}_{222}$. The volume of K expressed in function of the \mathcal{V}_{knm} and τ at any time is given by:

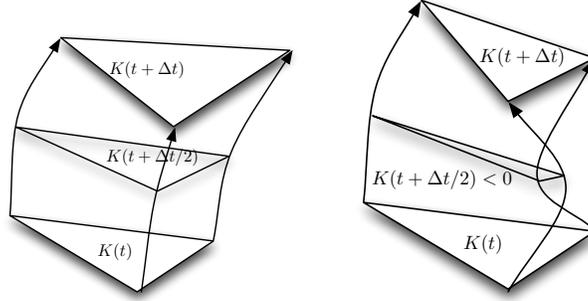


Fig. 2. Left, valid accelerated curved trajectory. Right, invalid accelerated curved trajectory while initial and final position are valid.

$$\begin{aligned}
\mathcal{V}(\tau) &= P_0 P_1(t + \delta t) \cdot [P_0 P_2(t + \delta t) \times P_0 P_3(t + \delta t)] \\
&= (1 - \tau)^6 \mathcal{V}_{000} \\
&\quad + 6(1 - \tau)^5 \tau \frac{1}{6} (2\mathcal{V}_{001} + 2\mathcal{V}_{010} + 2\mathcal{V}_{100}) \\
&\quad + 15(1 - \tau)^4 \tau^2 \frac{1}{15} (\mathcal{V}_{002} + \mathcal{V}_{020} + \mathcal{V}_{200} + 4\mathcal{V}_{110} + 4\mathcal{V}_{101} + 4\mathcal{V}_{011}) \\
&\quad + 20(1 - \tau)^3 \tau^3 \frac{1}{20} (2\mathcal{V}_{012} + 2\mathcal{V}_{021} + 2\mathcal{V}_{102} + 2\mathcal{V}_{120} + 2\mathcal{V}_{201} + 2\mathcal{V}_{210} + 8\mathcal{V}_{111}) \\
&\quad + 15(1 - \tau)^2 \tau^4 \frac{1}{15} (\mathcal{V}_{220} + \mathcal{V}_{202} + \mathcal{V}_{022} + 4\mathcal{V}_{112} + 4\mathcal{V}_{121} + 4\mathcal{V}_{211}) \\
&\quad + 6(1 - \tau)^1 \tau^5 \frac{1}{6} (2\mathcal{V}_{221} + 2\mathcal{V}_{212} + 2\mathcal{V}_{122}) \\
&\quad + \tau^6 \mathcal{V}_{222}
\end{aligned} \tag{3}$$

The study of the sign of this polynomial enables to check the validity of the mesh motion when vertices have an accelerated velocity curved path. It requires the evaluation of 27 tetrahedra volumes.

The mesh deformation is valid if all tetrahedra have a positive volume until the end of the displacement time frame, *i.e.*, if Polynomial (3) is always positive. To study its sign, we study the Bézier curve $\gamma(\tau) = (\tau, \mathcal{V}(\tau))$ associated with the volume polynomial denoted $\mathcal{V}(\tau)$. Indeed, Polynomial (3) can be rewritten:

$$\mathcal{V}(\tau) = \sum_{i=0}^6 B_i^6(\tau) V_i \quad \text{or} \quad \gamma(\tau) = \sum_{i=0}^6 B_i^6(\tau) \mathbf{Q}_i$$

where $B_i^n(\tau) = C_i^n \tau^i (1 - \tau)^{n-i} = \frac{n!}{i!(n-i)!} \tau^i (1 - \tau)^{n-i}$ are the Bernstein polynomials and V_i (resp. \mathbf{Q}_i) are the control coefficients (resp. points) of the curve. A sufficient condition for positivity is to have, see Figure 3 (left),

$$V_0 > 0 \quad \text{and} \quad \forall i = 1, 5, \quad V_i \geq 0 \quad \text{and} \quad V_6 > 0. \tag{4}$$

But, Condition (4) may be too restrictive. To refine the analysis, the well-known refinement algorithm of De Casteljau is considered, see for instance [13], that consists in splitting the curve $\gamma(\tau)$ into two curves $\gamma_1(\tau)$ and $\gamma_2(\tau)$, see Figure 3 (right). If Condition (4) is not verified, the refinement algorithm for a value t (here $t = \frac{1}{2}$) is

- Set $V_i^0 = V_i$, $\forall i = 0, \dots, 6$
- For $j=1, \dots, 6$
 Compute $V_i^j = (1-t)V_i^{j-1} + tV_{i+1}^{j-1}$, $\forall i = 0, \dots, (6-j)$
 EndFor
- $\mathcal{V}(\frac{1}{2}) = V_0^6$, $\mathcal{V}_1(\tau) = \sum_{i=0}^6 B_i^6(\tau) V_0^i$ and $\mathcal{V}_2(\tau) = \sum_{i=0}^6 B_i^6(\tau) V_i^{(6-i)}$

Then, the resulting refinement is analyzed:

- If $\mathcal{V}(\frac{1}{2}) = V_0^6 < 0$ then the tetrahedron volume becomes negatives between t and $t + \Delta t/2$. The mesh motion is not valid. The De Casteljau's refinement algorithm is re-applied on $\mathcal{V}_1(\tau)$ to find a $\Delta t/2^n$ for which the mesh deformation is valid.
- If Condition (4) is satisfied for $\mathcal{V}_1(\tau)$ and $\mathcal{V}_2(\tau)$ then the volume is always positive and the mesh motion is valid.
- Otherwise, none of the above conditions are verified, the validity is still unknown. The De Casteljau's refinement algorithm is re-applied to each curve that does not comply with Condition (4).

Nevertheless, this sign study is not sufficient to guarantee the validity of the mesh in any case. This is true if and only if all vertices have a trajectory with the considered parametrization. But, vertices located on moving objects are always moved exactly. Thus, their trajectory may not be quadratic. In consequence, we still have to check the validity of the mesh at each moving time step, at least for elements having one vertex on a moving boundary.

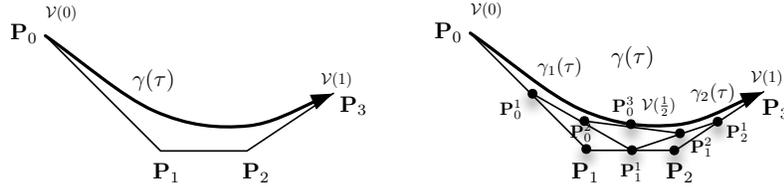


Fig. 3. Left, volume curve and its control point. Right, refinement of the volume curved which is split down the middle.

Enhance moving mesh algorithm

When the proposed strategy is considered, we have now a time step for the elasticity Δt and a time step for each move (*i.e.*, each solver time step) δt . To control the mesh quality through the considered time frame, mesh optimization is applied at each move and vertices speed \mathbf{v}^{opt} due to mesh smoothing are seen like corrections of vertices displacements for this step:

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

Elasticity time step. A maximal time step Δt_{max} between two elasticity resolutions is set at the beginning of the simulation. However, this time step can be too large depending on geometric or body motion constraints. Therefore, the chosen Δt is

adapted depending on the current situation. After each linear elasticity system resolution, the mesh motion is analyzed as described above. If the mesh motion is not valid, then the elasticity time step is reduced accordingly by dichotomy until validity is achieved: $\Delta t = \Delta t_{max}/2^n$. Eventually, the interval between two elasticity resolutions can be reduced when bad elements qualities are anticipated or shortened when mesh quality degrades abruptly.

Geometric time step. A good restriction to be imposed to the mesh movement is that vertices cannot cross too many elements on a single move. 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. As each moving step is coupled with an optimization procedure, cutting a large displacement into several smaller displacements by reducing the geometric CFL enables to ease mesh movement. The moving geometric time step is given by:

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

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

Algorithm. The enhance moving mesh strategy is:

Algorithm 2 Moving Mesh Algorithm with Curved Trajectories

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

1. $\{\mathbf{d}_{|\partial\Omega_h}^{body}(t + \Delta t/2)\}$ = Compute bodies vertices displacements from current translation speed \mathbf{v}^{body} , rotation speed $\boldsymbol{\theta}^{body}$ and acceleration \mathbf{a} for $[t, t + \Delta t/2]$
 $\mathbf{d}(t + \Delta t/2)$ = Solve elasticity system $(\mathbf{d}_{|\partial\Omega_h}^{body}(t + \Delta t/2), \Delta t/2)$
2. $\{\mathbf{d}_{|\partial\Omega_h}^{body}(t + \Delta t)\}$ = Compute bodies vertices displacements 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 $(\mathbf{d}_{|\partial\Omega_h}^{body}(t + \Delta t), \Delta t)$
3. $\{\mathbf{v}, \mathbf{a}\}$ = Deduce inner vertices speed and acceleration from both displacements $\{\mathbf{d}(t + \Delta t/2), \mathbf{d}(t + \Delta t)\}$
4. If mesh motion is invalid then $\Delta t = \Delta t/2^n$ and return to 1.
 Else $T^{els} = t + \Delta t$
5. While ($t < T^{els}$)
 - a) δt = Get moving mesh time step $(\mathcal{H}^k, \mathbf{v}, CFL^{geom})$
 - b) \mathcal{H}^k = Swaps optimization $(\mathcal{H}^k, Q_{target}^{swap})$
 - c) \mathbf{v}^{opt} = Vertices smoothing $(\mathcal{H}^k, Q_{target}^{smoothing}, Q_{max})$
 - d) \mathcal{H}^{k+1} = Move the mesh and update vertices speed $(\mathcal{H}^k, \delta t, \mathbf{v}, \mathbf{v}^{opt}, \mathbf{a})$
 - e) $t = t + \delta t$
 EndWhile

EndWhile

5 Numerical illustrations

Numerous analytical examples on academic geometries have been addressed to demonstrate the feasibility of performing 3D moving mesh simulation while keeping the number of vertices constant. This means that large displacements can be handled **without any remeshing** using only vertices displacements and edges/faces swaps. Here, three of them are presented: a moving rotating cube, a swirling cylinder and two cylinders interpenetrating. Then, first applications to industrial problems are shown. All examples have been run in *serial* on a 64-bits MacPro with an IntelCore2 chipsets with a clockspeed of 2.8 GHz with 32 Gb of RAM.

5.1 Academic examples

A moving rotating cube. We translate through the domain a rotating cube in a mesh composed of 34 567 vertices and 188 847 tetrahedra. The cube movement is shown in Figure 4. 26 elasticity systems have been solved during the movement while 25 have been initially prescribed. A total of 325 moving steps have been done. In Table 1, we observe that at the end of the move the quality of the mesh is excellent with an average quality of 1.4158, 99.40% of the elements with a quality less than 2 and a worst quality of 4.78. The worst created element during the whole movement has a quality of 7.07. The quality is thus excellent throughout the simulation. A total of 293 978 swaps have been performed which represent 904 swaps per moving steps. As regards the CPU time, it took 3m05s to move the cube, 61% of the CPU time being spent in elasticity resolutions.

Swirling cylinder. In this example, a cylinder is swirling inside a domain performing two turns. This displacement mainly involve shear inside the mesh. It is thus very appropriate to illustrate the efficiency of the swap topologic operation. The initial mesh is composed of 25 135 vertices and 139 540 tetrahedra. The cylinder rotation is depicted in Figure 5. 36 elasticity systems have been solved during the movement as initially prescribed. A total of 388 moving steps have been done. In Table 1, we note that at the end of the move the quality of the mesh is excellent with an average quality of 1.4458, 97.86% of the elements with a quality less than 2 and a worst quality of 4.81. The quality remains excellent throughout the simulation as the worst created element during the whole movement has a quality of 7.25. A total of 875 702 swaps have been performed which represent 2 256 swaps per moving steps. The number of swaps per move is quite large because a shear movement occurs in a large part of the domain. As regards the CPU time, it took 2m28s to rotate the cylinder, 48% of the CPU time being spent in elasticity resolutions.

Interpenetrating cylinders. The last example is more challenging, two cylinders interpenetrate each other and during this movement there is only one layer of elements

Case	Average Q_{end}	$1 < Q_{end} < 2$	Worst Q_{end}	Worst Q_{all}	# of swaps	CPU
Cube	1.4158	99.40%	4.78	7.07	293 978	3m05s
Cylinder	1.4458	97.86%	4.81	7.25	875 702	2m28s
Two cylinders	1.4645	97.74%	5.57	595	760 130	6m38

Table 1. Mesh statistics and CPU time for the three academic test cases.

between both cylinders, *i.e.* internal edges connect both cylinders. Hence, a shear layer appears between the two cylinders. This movement is shown in Figure 6. The initial mesh is composed of 34 567 vertices and 188 847 tetrahedra. A total of 106 elasticity systems have been solved during the movement while initially 100 resolutions have been prescribed. The added elasticity resolutions are done when both cylinders start to interpenetrate. At that moment, the worst quality elements appear because some tetrahedra are crushed by the two cylinders displacement. A total of 398 moving steps have been done. In Table 1, we note that at the end of the move the quality of the mesh is excellent with an average quality of 1.4645, 97.74% of the elements with a quality less than 2 and a worst quality of 5.57. But, during the movement bad shape elements appear, the worse of them has a quality equal to 595. However, the optimization process is able to get rid of these elements to finally achieve an excellent quality. A total of 760 130 swaps have been performed which represent 1 909 swaps per moving steps. As regards the CPU time, it took 6m38s to perform the displacement, 75% of the CPU time being spent in elasticity resolutions.

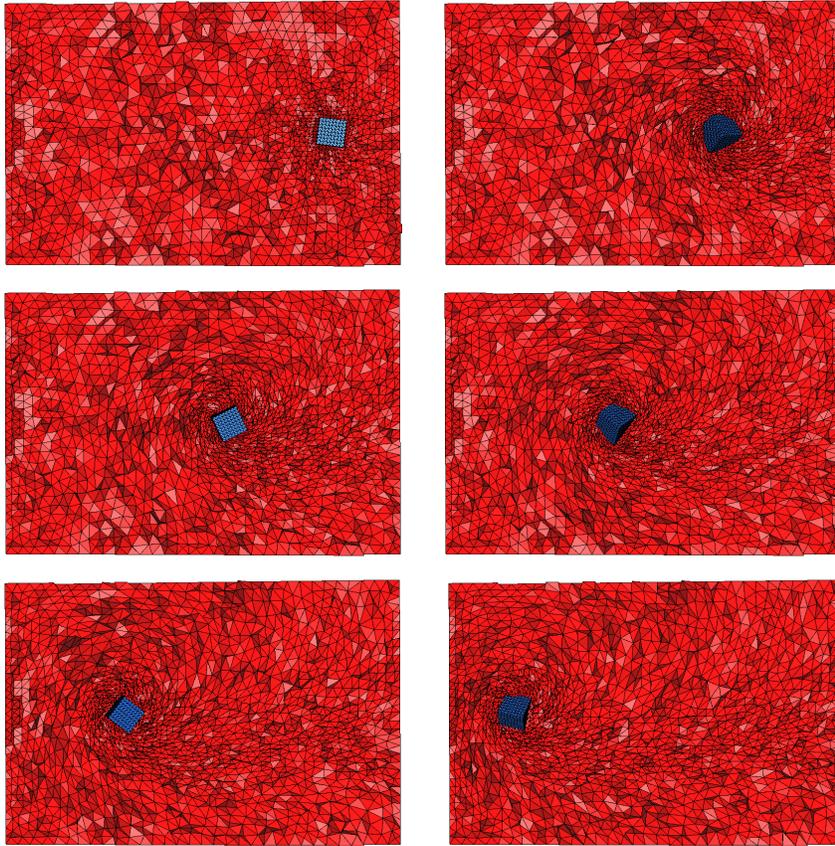


Fig. 4. Snapshots of a moving rotating cube.

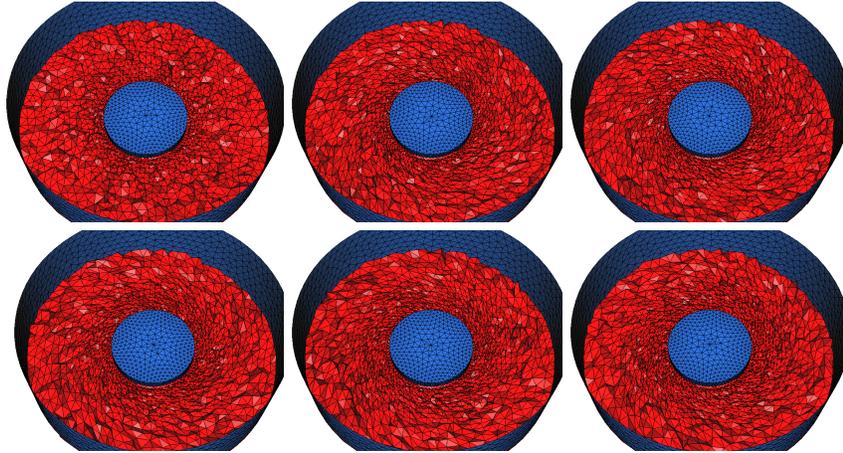


Fig. 5. Snapshots of a swirling cylinder. From left to right and from top to bottom, the cylinder after a rotation of 0 , 0.4π , 0.8π , 1.2π , 1.6π and 2π .

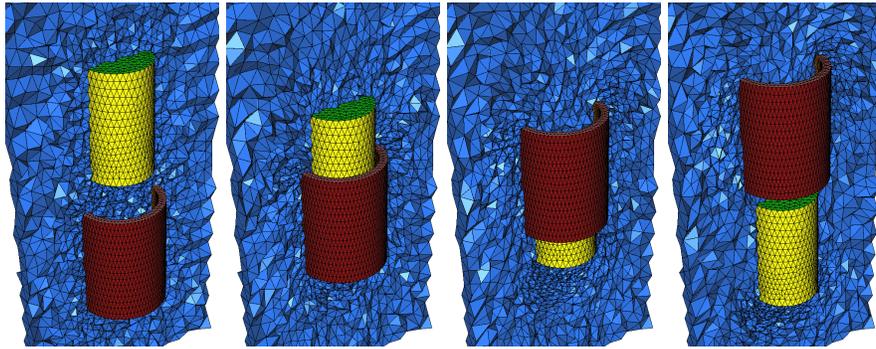


Fig. 6. Snapshots of two cylinders interpenetrating.

5.2 Industrial configurations

To illustrate that the presented method can be applied in an industrial context, two realistic problems are imitated. The geometry displacement is given by an analytical function. Be aware that there is no FSI computation for these test cases.

Ejected cabin door. The first problem is the door ejection of an over-pressurized aircraft cabin. This is a usual industrial benchmark for aircraft designers whose aim is to evaluate when the door hinge will yield under cabin pressure. Indeed, the trend for new aircrafts is to lower the cabin altitude.

The difficulty is that the geometry is anisotropic and rolls over the elements while progressing inside a uniform mesh composed of 98k vertices. The door movement is shown in Figure 7. At the end of the move, the quality of the mesh is excellent with an average quality of 1.2924, 99.75% of the elements with a quality lower than 2 and a worst quality of 6.48. But, at the beginning of the movement badly shaped elements appear when the door is ejected from its sash where the free space is tiny, see Figure 8, left. The worse of them having a quality equal to 83. The

number of mesh deformation resolutions automatically adapts to the complexity of the problem. Initially, a number of 20 elasticity resolutions has been specified but at the end a total 50 have been resolved and 1379 moving steps have been performed. Again, the optimization process is able to get rid of these elements to finally achieve an excellent quality. In this example, 394 420 swaps have been done and the total CPU time is 14m45s.

Space Shuttle ejecting its two Solid Rocket Boosters. The second example is the Space Shuttle ejecting its two Solid Rocket Boosters (SRB) after burnout. SRBs separation occurs just after two minutes into the flight, at an altitude close to 50 km and a Mach number above 4. The objective of such numerical simulation is to understand the separation dynamics of the SRBs attached to the space shuttle, notably to know if booster separation motor plume will hit the orbiter wind-shield.

This example deals with a very complex geometry, see Figure 8, right. The two SBRs progress inside a uniform mesh composed of 930k vertices, they have a parabolic trajectory and they rotate at the same time. The SBRs long-time movements are depicted in Figure 9. A total of 300 elasticity resolutions have been done as initially prescribed. At the end of the simulation the quality of the mesh is excellent with an average quality of 1.366, 99.92% of the elements with a quality lower than 2 and a worst quality of 7.89. Again, the quality is excellent throughout the simulation as the worst created element during the whole movement has a quality of 39. For this larger mesh application, a total of 2 302 moving steps and 8 899 889

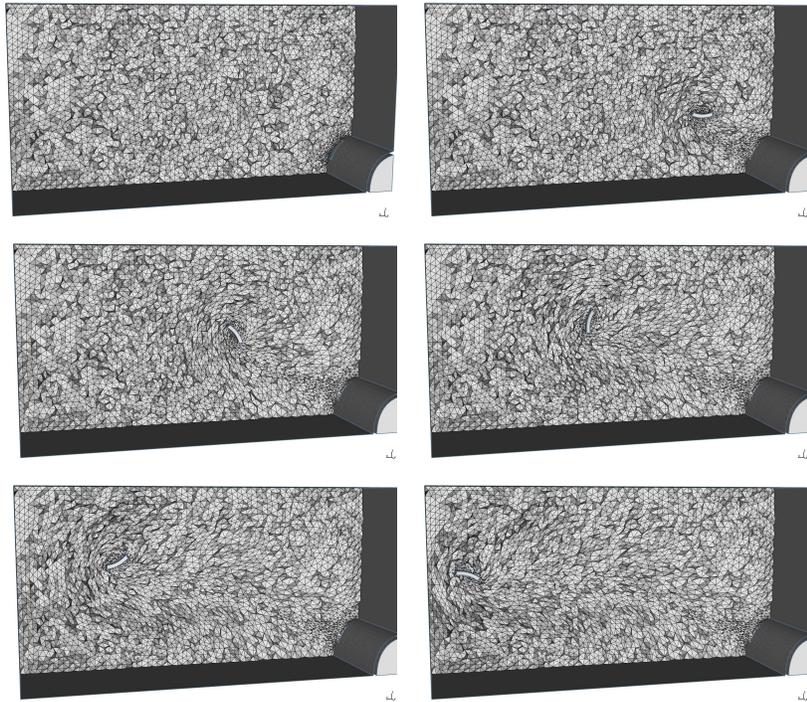


Fig. 7. Snapshots of the ejection of an aircraft cabin door.

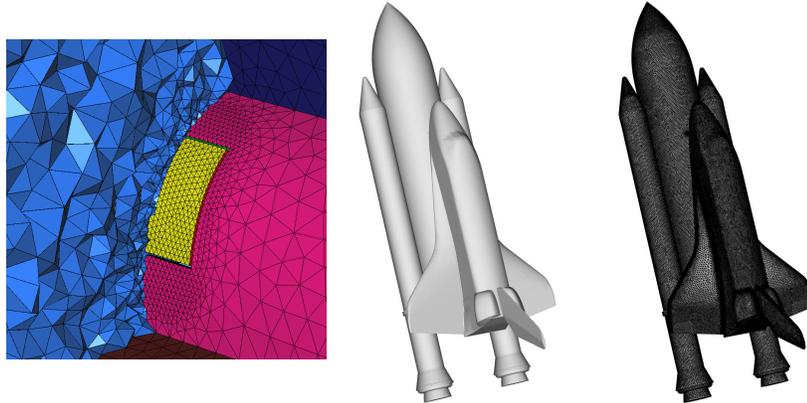


Fig. 8. Left, ejection door surface mesh and cut plane. Right, space shuttle geometry and surface mesh.

swaps have been carried out to achieve the SRBs displacement. The global CPU time is 8h.

6 Conclusion

In this paper, numerical evidence have been given to prove that 3D large displacements are possible with a mesh deformation algorithm coupled with mesh optimization using only swaps and vertex movements. No re-meshing are required. This clever strategy clearly improves the robustness, efficiency and accuracy of the moving mesh method. This strategy has been successfully applied to rigid body large displacement but also to deformable bodies.

However, several problematics have not yet been addressed. We need to define an optimal strategy for moving geometries simulations where the volume of the domain varies considerably, e.g. the simulation of a four-stroke engine. In that case, it seems mandatory to add or remove vertices. Similarly to metric-based anisotropic mesh adaptation [1, 3], a local re-meshing strategy will be employed. Other issues are the simulations of deformable geometries with possible topology change and the efficient treatment of contact problems.

References

1. F. Alauzet, A. Belme, and A. Dervieux. Anisotropic goal-oriented mesh adaptation for time dependent problems. In *Proceedings of the 20th International Meshing Roundtable*, pages 99–121. Springer, 2011.
2. F. Alauzet and M. Mehrenberger. P1-conservative solution interpolation on unstructured triangular meshes. *Int. J. Numer. Meth. Engng*, 84(13):1552–1588, 2010.
3. F. Alauzet and G. Olivier. Extension of metric-based anisotropic mesh adaptation to time-dependent problems involving moving geometries. In *49th AIAA Aerospace Sciences Meeting*, AIAA Paper 2011-0896, Orlando, FL, USA, Jan 2011.

4. T.J. Baker and P. Cavallo. Dynamic adaptation for deforming tetrahedral meshes. *AIAA Journal*, 19:2699–3253, 1999.
5. J. Batina. Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1388, 1990.
6. J.D. Baum, H. Luo, and R. Löhner. 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, Jan 1994.
7. J.A. Benek, P.G. Buning, and J.L. Steger. A 3D chimera grid embedding technique. In *7th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 1985-1523, Cincinnati, OH, USA, Jul 1985.
8. G. Compere, J-F. Remacle, J. Jansson, and J. Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *Int. J. Numer. Meth. Engng*, 82(7):843–867, 2010.
9. A. de Boer, M. van der Schoot, and H. Bijl. Mesh deformation based on radial basis function interpolation. *Comput. & Struct.*, 85:784–795, 2007.
10. C. Dobrzynski and P.J. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th International Meshing Roundtable*, pages 177–194. Springer, 2008.
11. P.J. Frey and P.L. George. *Mesh generation. Application to finite elements*. ISTE Ltd and John Wiley & Sons, 2nd edition, 2008.
12. P.L. George. Tet meshing: construction, optimization and adaptation. In *Proceedings of the 8th International Meshing Roundtable*, South Lake Tao, CA, USA, 1999.
13. P.L. George and H. Borouchaki. Construction of tetrahedral meshes of degree two. *Int. J. Numer. Meth. Engng*, 90:1156–1182, 2012.
14. O. Hassan, E. J. Probert, K. Morgan, and N. P. Weatherill. Unsteady flow simulation using unstructured meshes. *Comput. Methods Appl. Mech. Engrg.*, 189:1247–1275, 2000.
15. R. Löhner. Extensions and improvements of the advancing front grid generation technique. *Communications in Numerical Methods in Engineering*, 12:683–702, 1996.
16. R. Löhner and C. Yang. Improved ALE mesh velocities for moving bodies. *Comm. Numer. Meth. Engrg*, 12(10):599–608, 1996.
17. E. Luke, E. Collins, and E. Blades. A fast mesh deformation method using explicit interpolation. *J. Comp. Phys.*, 231:586–601, 2012.
18. D.L. Marcum. Unstructured grid generation using automatic point insertion and local reconnection. *Revue Européenne des Éléments Finis*, 9:403–423, 2000.
19. G. Olivier and F. Alauzet. A new changing-topology ALE scheme for moving mesh unsteady simulations. In *49th AIAA Aerospace Sciences Meeting*, AIAA Paper 2011-0474, Orlando, FL, USA, Jan 2011.
20. C.S. Peskin. Flow patterns around heart valves: a numerical method. *J. Comp. Phys.*, 10:252–271, 1972.
21. K. Stein, T. Tezduyar, and R. Benney. Mesh moving techniques for fluid-structure interactions with large displacements. *Jour. Appl. Mech.*, 70:58–63, 2003.
22. P.D. Thomas and C.K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037, 1979.
23. Z. Yang and D.J. Mavriplis. Higher-order time integration schemes for aeroelastic applications on unstructured meshes. *AIAA Journal*, 45(1):138–150, 2007.

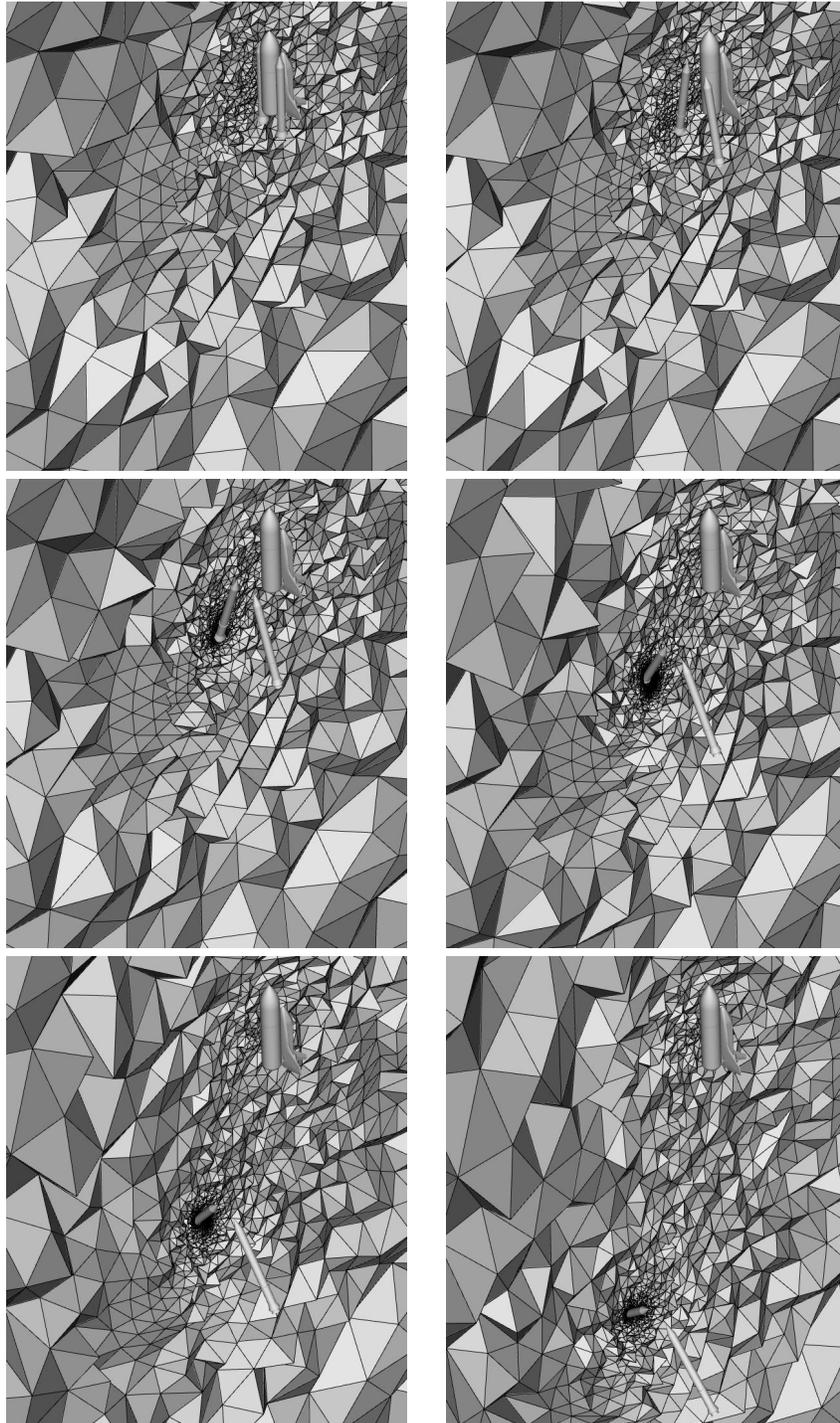


Fig. 9. Snapshots of the Space Shuttle ejecting its two SRBs after burnout.