HiSST: 4th International Conference on High-Speed Vehicle Science Technology

22-26 September 2025, Tours, France

A Dynamic CFD Mesh Algorithm with Joint Resolution to Compute Relative Motion and Efforts on Body Parts.

Margot Guiho[1], *Philippe Grenard*[2]

Abstract

The accurate simulation of mobile components in hypersonic vehicles presents major challenges, particularly during critical flight phases involving geometric reconfigurations—such as control surface deflection, nozzle vectoring, or stage separation. At these speeds, aerodynamic and mechanical response times may become comparable, requiring fully dynamic computations that go beyond quasi-static assumptions.

To address these challenges, two dynamic mesh modules have been developed within the ONERA CEDRE simulation framework. The mesh intersection module enables robust and fully parallel computation of overlapping moving geometries. In realistic use cases such as the Ariane 64 lift-off configuration, this module was shown to handle approximately 100,000 intersection problems per time step over one million time steps—representing 100 billion elementary computations—without performance degradation.

The mesh displacement module incorporates mechanical joints into the simulation, accounting for load transfer and kinematic constraints between moving parts. Eleven standardized joint types were implemented, with support for joint limits and failure modeling in simple configurations. Elementary test cases validated the robustness of the approach, even for over-constrained (hyperstatic) assemblies.

These tools were successfully applied to a realistic launcher scenario and are adaptable to a wide range of configurations involving transient motion, such as fin deployment or launcher touchdown. Future developments will include modeling of contact, friction, rebound, and technological joints such as springs and dampers.

Keywords: *CFD, moving meshes, joints resolution*

Nomenclature

Latin

$\underline{b}$ – bias vector (stabilization)
$\underline{C}(\underline{s})$ – Constraint vector
$\underline{G}$ – center of mass
$\underline{\underline{I_A}}$ – inertia tensor at point A
$\underline{\underline{J}}$ – Jacobian matrix of a constraint
$\underline{\underline{M}}$ – Global mass matrix
m – mass
q – a quaternion
q_0,$\underline{q}$ – scalar and vector part of **q**
$\underline{r} = \underline{GP}$ – vector between $\underline{G}$ and arbitrary point $\underline{P}$
$\underline{\underline{R}}$ – rotation matrix associated with the quaternion
$\underline{s}$ – Generalized position vector
$\underline{\underline{S}}$ – Matrix linking position
t – time
$\underline{\mathcal{T}^{ext}}$ – Torsor of external (+inertial) forces
$\underline{v_A}$ – velocity of point A
$\underline{x_A}$ – coordinates of point A

Greek

β – stabilization parameter
Δt – time step
$\underline{\lambda}$ – Lagrange multiplier
θ_a – Angle around a axis
$\underline{\omega}$ – angular velocity

[1] *ONERA, Chemin de la Hunière, Palaiseau, France, margot.guiho@onera.fr*
[2] *ONERA, Chemin de la Hunière, Palaiseau, France, philippe.grenard@onera.fr*

1. Introduction

The dynamic behavior of hypersonic vehicles often involves complex geometric changes during flight. These can include control surface deflections, nozzle vectoring for upper rocket stages, or the activation of moving parts such as aerodynamic surfaces and actuators. In the case of missile or launch vehicle systems, such changes are further complicated by events like booster separation or stage detachment.

At such high velocities, the timescales of aerodynamic and mechanical phenomena can be of the same order of magnitude, potentially leading to significant interactions or hysteresis effects. This creates a strong need for dynamic simulations—beyond quasi-static approximations—during these critical flight phases, where structural motion and aerodynamic loads are tightly coupled.

Furthermore, mechanisms such as control surface or nozzle deflection can transmit significant loads to the main vehicle body, possibly inducing unwanted angular responses or pitch motions. Accurately capturing these effects requires not only a dynamic treatment of moving geometries but also an appropriate modeling of mechanical joints and force transmissions.

In this context, developments have been carried out within the ONERA CEDRE code, a high-fidelity simulation tool for energetic systems. Two dynamic mesh modules have been introduced:

- A mesh intersection module for handling topological changes and interactions between overlapping moving bodies (chapter 2).
- A mesh displacement module that supports the integration of mechanical joints and the associated force transfer mechanisms (chapter 3).

These capabilities have been tested on both elementary and realistic configurations, including launcher lift-off scenarios, and are designed to extend the simulation framework to accurately handle mobile structures during transient flight phases.

2. Parallel mesh intersection module

2.1. General principle of the mesh intersection

In order to compute movable objects in a CFD framework, along with their associated meshes, the most popular method would probably be the Chimera mesh method, originally developped for structured aerodynamics applications [1, 2, 3]. This method relies on the superposition of two grids, with some overlapping where solutions are interpolated between the two grids in order to transfer some informations. This interpolation step has a major drawback: for a conservative interpolation, the algorithm may have a very high computational cost. On the other hand, a non-conservative interpolation would help saving CPU ressources, but can lead to dramatic loss of mass, momentum and energy conservativity if the local fields located at the overlapping is non uniform. In the context of aerodynamics application, and with an upstream mesh overlapping in a homogeneous region, that method has proved to be efficient and precise.

Some applications however focuses on both downstream flow evolution or highly non linear flow at the frontier between the overlapping grids, where the chimera method might introduce either high computational costs or high conservativity losses. To overcome these difficulties, an interction mesh method for moving bodies was developped [4, 5, 6] within a hexaedra framework for the meshes. This approaches uses an exact intersection between the two grids to build a conformable mesh to ensure conservativity. This approach allows to remove the costly interpolation steps as the solution is now solved on a moving mesh, however the intersection algorithm has its own cost, which can be reduced by evaluating the steps at which the intersection should be re-computed.

Here, the later technique, with intersection, has been reworked with generalized polyhedron and within a highly parallel environment. It will be described hereafter with emphasis on parallel treatement and general polyhedron intersection. The geometric problem can be summarized as: find the intersection between a volumic mesh and a closed surfacic mesh.

2.2. Parallel approach

The parallel uses intensively the numerical library ”ParaDiGM” [**?**]. The main idea is to extract from the full meshes a set of ”elementary intersection problems” that are dispatched on all CPU. This search is

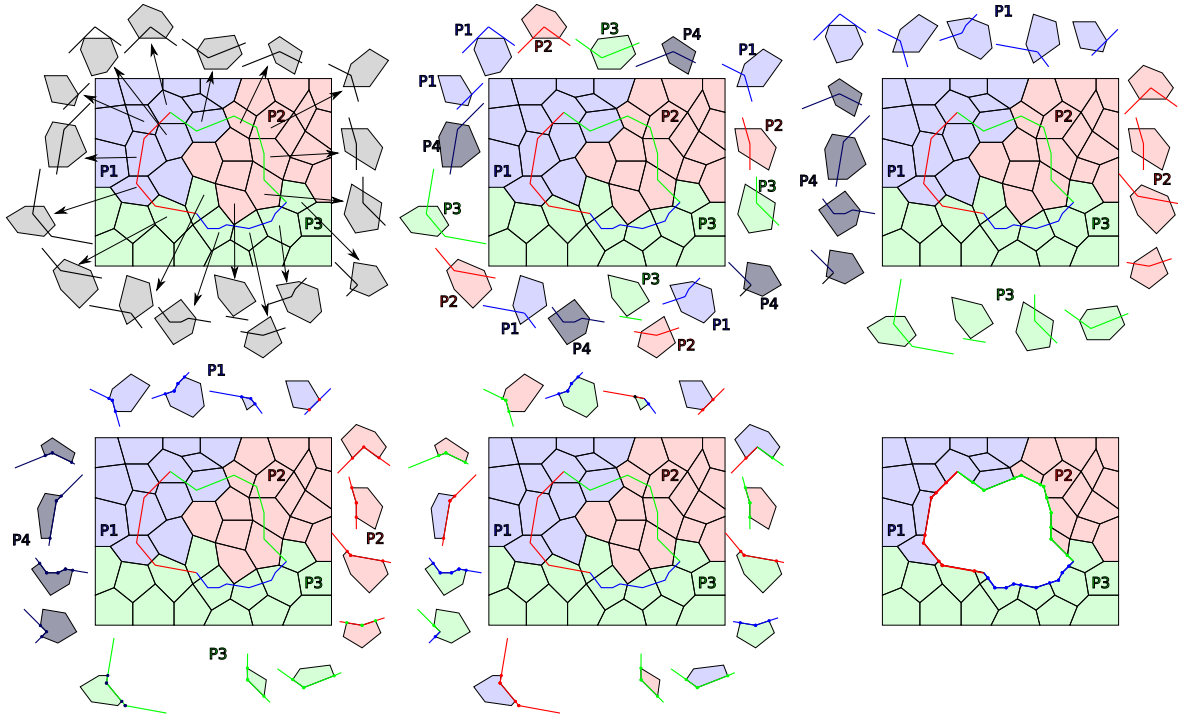

Fig 1. Parallel algorithm steps.

made through a parallel bounding box intersection algorithm. The main steps are presented (from left to right and from top to bottom) in the figure 1, with colors representing the CPU on which the data are during each steps, and consists in the following:

- on every CPU, for every volumetric cells, a cartesian bounding box is computed as the min/max coordinates of every points of each cells. An additional margin of a few percent is added.
- on every CPU, every surfacic element in the intersecting surface, a cartesian bounding box is also computed.
- those bounding boxes are given to the ParaDiGM library, which gives in return on every CPU, a balanced list of volume elements along with, for each volumeic cell, the list of surfacic elements whose bounding boxes intersect the bounding box of the volumic cell (see Fig. 1, step 1 and 2).
- the elementary intersection problem is then solved independently for each volumic element (see sec. 2.3 and Fig. 1, step 3 and 4)
- intersection results are then given back to the ParaDiGM library that send data back to their original CPUs (see Fig. 1, step 5)
- from the intersected elements, hidden and visible ones are deduced
- the final computational mesh is rebuilt from those information and transfered to the CFD solver (Fig. 1 step 6).

2.3. Mesh intersection algorithm

The mesh intersection algorithm has been developped for both 2D and 3D grids. For the sake of clarity and simplicity, it is only described hereafter for 2D meshes, for which cells an faces (in a CFD nomenclature) are respectively polygons and segments.

The input of this elementary intersection problem is, as described previously, one cell (polygon) along with all the surfacic faces (segments) close enough to potentially intersect it (but not necessarily) (Fig.3, step 1). All the faces are given with a normal vector pointing out the visible (not hided) part, chosen

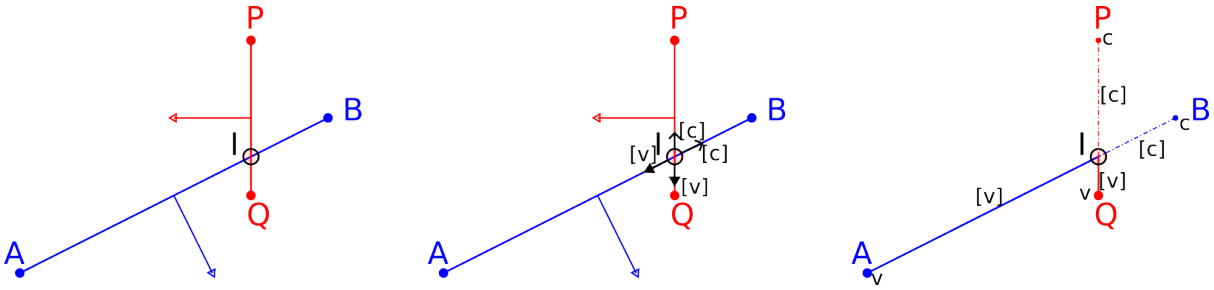

Fig 2. 2D face-face intersection principle.

from the mesh intersection setup. The surfacic faces are given, in 2D, through one or several broken lines. Each of their vertex has one or two segments associated. In 3D, each face edge is linked to at most two faces. Thus, by construction, vertices (or edges in 3D) linked to only one face are necessarily covert ("c"). Moreover, each face-face intersection gives birth to a new vertex belonging to both faces, where the covert or visible directions can be defined thanks to the visible normal direction of each face (Fig. 2). This permits to initiate the covert/visible aspect of some vertices and/or faces (Fig. 3, step 2).

From these initializations, the visible or covert aspect of faces and vertices is propagated through the connectivity of the diffent elements consituting the elementary problem. Covert parts are enventually removed from the chopped cell (Fig. 3, step 3 and 4).

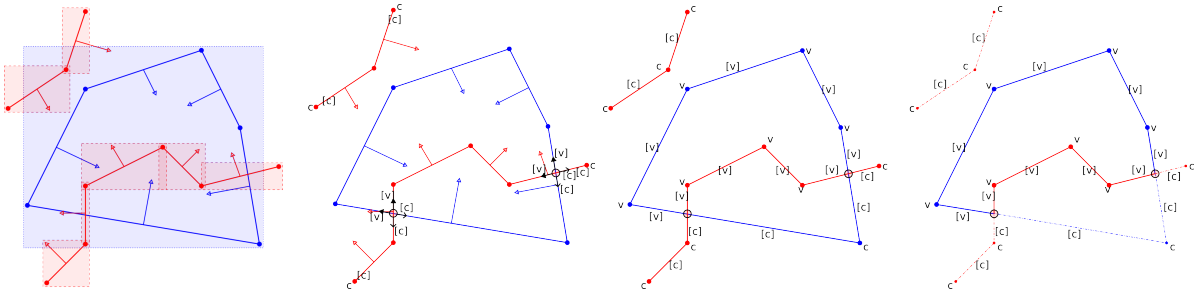

Fig 3. The different steps for the cell intersection by a list of faces in 2D.

2.4. Limitations and difficulties

Obviously, 3D is not equal to 2D+1D. The 3D algorithm is far more complex, including the necessity to rebuild ordered set of points to define closed faces, treating edge/face intersection and all the singular cases.

It must also be noted that the intersection process can lead to several ill-shaped cells, for which limitations and corrections procedure must apply:

- very small cells, which can lead to numerical difficulties in the CFD solver.
- non convex or non-connected polygonal faces or cells, for which the spatial discretization order might drop to order one locally.

3. Mesh displacement module: joints resolution

3.1. Single body motion resolution

Let's consider a single rigid body, defined by :

- a mass m.
- a center of mass $\underline{G^0}$.
- an inertia tensor (at G) $\underline{\underline{I_G^0}}$.

- external forces, represented by a torsor $\underline{\mathcal{T}^{ext}}$ defined by external forces $\underline{\mathcal{F}^{ext}}$ and moments applied at the center of mass $\underline{\mathcal{M}_G^{ext}}$.

The superscript $\cdot^0$ represents the fact that these quantities are expressed in the body frame, while the absence of superscript will describe the quantities in the main (Galilean) frame.

The displacement of this body is described through:

- the position of its center of mass $\underline{x_G}(t)$.
- the velocity of this center of mass $\underline{v_G}(t) = \underline{\dot{x}_G}(t)$.
- the attitude of the rigid body defined by a unit quaternion $\mathbf{q} = (q_0; \underline{q})$.
- the angular velocity $\underline{\omega}(t)$.

Several attitude representations can be used [7, 8], but they can all define a rotational matrix $\underline{\underline{R}}(t)$, which can be used to switch between body frame and main frame. Indeed, for all point P defined by $\underline{GP}(t) = \underline{r}(t)$, one can write the following relations:

$$\underline{r}(t) = \underline{\underline{R}}(t) \cdot \underline{r^0} \qquad \underline{\dot{r}}(t) = \underline{\omega}(t) \times \underline{r^0} \qquad \underline{\underline{I}}(t) = \underline{\underline{R}}(t)\underline{\underline{I^0}}\,\underline{\underline{R^T}}(t) \tag{1}$$

The inertia tensor, contrary to the mass, depends on the frame in which it is expressed: while it is constant in the body frame, it depends on the time on the main frame. Finally, let's introduce the matrix $\underline{\underline{\hat{a}}}$, linked to a vector $\underline{a}$, such that, for all vector $\underline{v}$, we have:

$$\underline{\underline{\hat{a}}} \cdot \underline{v} = \underline{a} \times \underline{v} \qquad \underline{\underline{\widehat{a}}} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \tag{2}$$

If $*$ denotes the quaternion products, we also can write the following relations:

$$\underline{\omega} = 2\,\dot{\mathbf{q}} * \widetilde{\mathbf{q}} \qquad \dot{\mathbf{q}} = \frac{1}{2}(0, \underline{\omega}) * \mathbf{q} = \underline{\underline{Q}} \cdot \underline{\omega} \qquad \underline{\dot{\underline{R}}} = \underline{\underline{\widehat{\omega}}} \cdot \underline{\underline{R}} \tag{3}$$

with:

$$\underline{\underline{Q}} = \frac{1}{2}\begin{pmatrix} -q_x & -q_y & -q_z \\ q_0 & q_z & -q_y \\ -q_z & q_0 & q_x \\ q_y & -q_x & q_0 \end{pmatrix} = \begin{pmatrix} -\underline{q}^T \\ q_0\underline{\underline{\mathbb{I}}} - \underline{\hat{q}} \end{pmatrix} \tag{4}$$

The conservation of impulsion and angular momentum reads:

$$\frac{d}{dt}\left(m\underline{v_G}\right) = \underline{\mathcal{F}^{ext}} \tag{5}$$

$$\frac{d}{dt}\left(\underline{\underline{I}} \cdot \underline{\omega}\right) = \underline{\mathcal{M}_G^{ext}} \tag{6}$$

Deriving the inertia tensor with respect to time leads to the relation

$$\underline{\dot{\underline{I}}} \cdot \underline{\omega} = \underline{\omega} \times \left(\underline{\underline{I}} \cdot \underline{\omega}\right)$$

Thus:

$$m^0\underline{\dot{v_G}} = \underline{\mathcal{F}^{ext}} \tag{7}$$

$$\underline{\dot{x_G}} = \underline{v_G} \tag{8}$$

$$\underline{\underline{I}} \cdot \underline{\dot{\omega}} = \underline{\mathcal{M}_G^{ext}} + \left(\underline{\underline{I}} \cdot \underline{\omega}\right) \times \underline{\omega} \tag{9}$$

$$\dot{\mathbf{q}} = \underline{\underline{Q}} \cdot \underline{\omega} \tag{10}$$

Introducing the generalized position vector $\underline{s}$, the velocity one $\underline{v}$, the global mass matrix $\underline{\underline{M}}$ and the $\underline{\underline{S}}$ matrix linking position derivatives to velocity through the relation $\underline{\dot{s}} = \underline{\underline{S}} \cdot \underline{v}$:

$$\underline{s} = \begin{pmatrix} \underline{x_G} \\ \mathbf{q} \end{pmatrix} \qquad \underline{v} = \begin{pmatrix} \underline{v_G} \\ \underline{\omega} \end{pmatrix} \qquad \underline{\underline{M}} = \begin{pmatrix} m^0 \underline{\underline{\mathbb{I}_{33}}} & 0 \\ 0 & \underline{\underline{I}} \end{pmatrix} \qquad \underline{\underline{S}} = \begin{pmatrix} \underline{\underline{\mathbb{I}_{33}}} & 0 \\ 0 & \underline{\underline{Q_{43}}} \end{pmatrix} \tag{11}$$

The fundamental principles of dynamics on a single body can be written as (including the inertial term $\left(\underline{\underline{I}} \cdot \underline{\omega}\right) \times \underline{\omega}$ into the external torsor $\underline{\mathcal{T}^{ext}}$) :

$$\underline{\underline{M}} \cdot \underline{\dot{v}} = \underline{\mathcal{T}^{ext}} \tag{12}$$

$$\underline{\dot{s}} = \underline{\underline{S}} \cdot \underline{v} \tag{13}$$

which can be integrated through a Verlet scheme [9]

3.2. Multiple body motion resolution with joints

For multiple bodies, the formalism can be extended by including position and quaternion of the N bodies in the $\underline{s}$ vector, velocity and angular velocity of the N bodies into the $\underline{v}$ vector, the mass and inertia tensor of the N bodies in a diagonal block matrix $\underline{\underline{M}}$ and an extended diagonal block matrix $\underline{\underline{S}}$:

$$\underline{s} = \begin{pmatrix} \underline{x_{G1}} \\ \mathbf{q_1} \\ \cdots \\ \underline{x_{GN}} \\ \mathbf{q_N} \end{pmatrix} \qquad \underline{v} = \begin{pmatrix} \underline{v_{G1}} \\ \underline{\omega_1} \\ \cdots \\ \underline{v_{GN}} \\ \underline{\omega_N} \end{pmatrix} \qquad \underline{\dot{s}} = \underline{\underline{S}}\, \underline{v} \tag{14}$$

$$\underline{\underline{S}} = \begin{pmatrix} \underline{\underline{\mathbb{I}}} & 0 & \cdots & 0 & 0 \\ 0 & \underline{\underline{Q_1}} & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \underline{\underline{\mathbb{I}}} & 0 \\ 0 & 0 & \cdots & 0 & \underline{\underline{Q_N}} \end{pmatrix} \qquad \underline{\underline{M}} = \begin{pmatrix} m_1^0 \underline{\underline{\mathbb{I}}} & 0 & \cdots & 0 & 0 \\ 0 & \underline{\underline{I_1}} & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & m_N^0 \underline{\underline{\mathbb{I}}} & 0 \\ 0 & 0 & \cdots & 0 & \underline{\underline{I_N}} \end{pmatrix} \tag{15}$$

However, this time, some internal efforts can exist due to the presence of joints between bodies. These efforts are represented through the use of a constraint torsor $\underline{\mathcal{T}^c}$, which is an unknown of the dynamic problem.

$$\begin{aligned} \underline{\underline{M}} \cdot \underline{\dot{v}} &= \underline{\mathcal{T}^{ext}} + \underline{\mathcal{T}^c} \\ \underline{\dot{s}} &= \underline{\underline{S}} \cdot \underline{v} \end{aligned} \tag{16}$$

3.3. Joints resolution

A joint introduces constraints between two bodies. The globality of these constraints can be expressed as an ”position-constraint” equation:

$$\underline{C}(\underline{s}) = 0 \tag{17}$$

The vector $\underline{C}$ contains one component for each constrained degree of freedom of a joint. The general idea is that if the position-constraint equation is respected at initial time and if its temporal derivative is also null, then the constraint is always true. Deriving the position-constraint equation leads to velocity-constraint equation:

$$\underline{\dot{C}}(\underline{s}) = \frac{d\underline{C}}{d\underline{s}} \underline{\dot{s}} = \left(\frac{d\underline{C}}{d\underline{s}} \underline{\underline{S}} \right) \underline{v} = \underline{\underline{J}}\, \underline{v} \tag{18}$$

The $\underline{\underline{J}}$ matrix is called the jacobian matrix. As numerical integration is never perfect, this derivative is generally solved as:

$$\underline{\dot{C}}(\underline{s}) = \underline{\underline{J}}\, \underline{v} + \underline{b} = 0 \qquad \underline{b} = \frac{\beta}{\Delta t} \underline{C}(\underline{s}) \quad \beta \in [0, 1] \tag{19}$$

where $\underline{b}$ is a bias velocity vector that permits stabilization of the solution (Baumgarte stabilization [10]).

The constraint equation is solved through the use of Lagrange multiplier $\underline{\lambda}$ such that:

$$\underline{\mathcal{T}^c} = \underline{\underline{J}}^T \underline{\lambda} \tag{20}$$

which ensure that those internal constraints do not work. Writing Eq. (19) with the new velocity vector $\underline{v^{n+1}}$ (integrating Eq. 16) gives:

$$\begin{aligned} \underline{\underline{J}}\,\underline{v^{n+1}} + \underline{b} &= 0 \\ \underline{\underline{J}}\left[\underline{v^n} + \Delta t \underline{\underline{M}}^{-1}\left(\underline{\mathcal{T}^{ext}} + \underline{\mathcal{T}^c}\right)\right] + \underline{b} &= 0 \\ \Delta t \underline{\underline{J}}\,\underline{\underline{M}}^{-1}\underline{\underline{J}}^T\underline{\lambda} &= -\underline{\underline{J}}\left(\underline{v^n} + \Delta t \underline{\underline{M}}^{-1}\underline{\mathcal{T}^{ext}}\right) - \underline{b} \end{aligned} \tag{21}$$

Introducing $\underline{\underline{K}} = \underline{\underline{J}}\,\underline{\underline{M}}^{-1}\underline{\underline{J}}^T$ (which is a square matrix) and $\underline{\bar{v}^{n+1}} = \underline{v^n} + \Delta t \underline{\underline{M}}^{-1}\underline{\mathcal{T}^{ext}}$, the velocity solution without constraints:

$$\underline{\underline{K}}\,\underline{\lambda} = -\frac{1}{\Delta t}\left(\underline{\underline{J}}\,\underline{\bar{v}^{n+1}} + \underline{b}\right) \tag{22}$$

The system resolution then implies to build the jacobian matrix for all possible constraints, build the $\underline{\underline{K}}$ matrix, and solve the linear system (22). This resolution is done via the sequential impulses solver ([11, 12, 13]) or, similarily, through a Projected Gauss Seidel method which is quite similar ([14]). Another option, which is not always possible, is to compute directly the inverse matrix $\underline{\underline{K^{-1}}}$ with a Gauss method.

Constraints equations, as well as jacobian matrices, have been analytically developped for all the 11 normalized constraints (ISO-3952), and implemented into ONERA's CEDRE solver.

4. Validation

4.1. Elementary test cases for joints resolution

4.1.1. Fixed joint

A fixed joint does not allow any relative motion, neither translation nor rotation, between two bodies. The elementary test case used to validate this joint involves two cubes with different masses subjected to gravity, rotating at identical speeds with opposing forces (see Fig. 4). The aim of this case study is to demonstrate that the two cubes remain locked together and rotate around the center of the joint. If the joint constraint were not enforced, the two cubes would rotate around their own centers of mass and gradually separate.

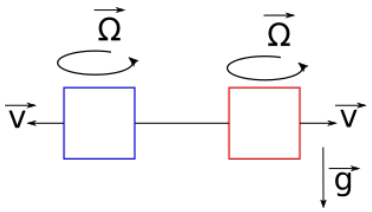

Fig 4. Representation of the fixed joint case

The analytical solution, derived using the principle of fundamental dynamics (PFD), yields the following equations:

$$\begin{aligned} \theta_z(t) &= \omega_z(0)t + \theta_z(0) \\ x_i(t) &= \cos(\theta_z(t))(x_i(0) - x_G(0)) + x_G(0) \\ y_i(t) &= -\sin(\theta_z(t))(x_i(0) - x_G(0)) + \frac{m_1 V_{y1}(0) + m_2 V_{y2}(0)}{m1 + m2}t \\ z_i(t) &= -\frac{1}{2}gt^2 + v_z(0)t + z(0) \end{aligned}$$

with $m_1 = 10$, $m_2 = 1$, $\underline{OG_1}(0) = (1,0,0)$, $\underline{OG_2}(0) = (-1,0,0)$, $\underline{V_1}(0) = (0,2,0)$, $\underline{V_2}(0) = (0,-2,0)$, $\theta_z(0) = 0$ and $\omega_z(0) = 2$. Fig. 5 shows the evolution of the centers of gravity of the two cubes obtained from the simulation and the analytical solution, respectively.

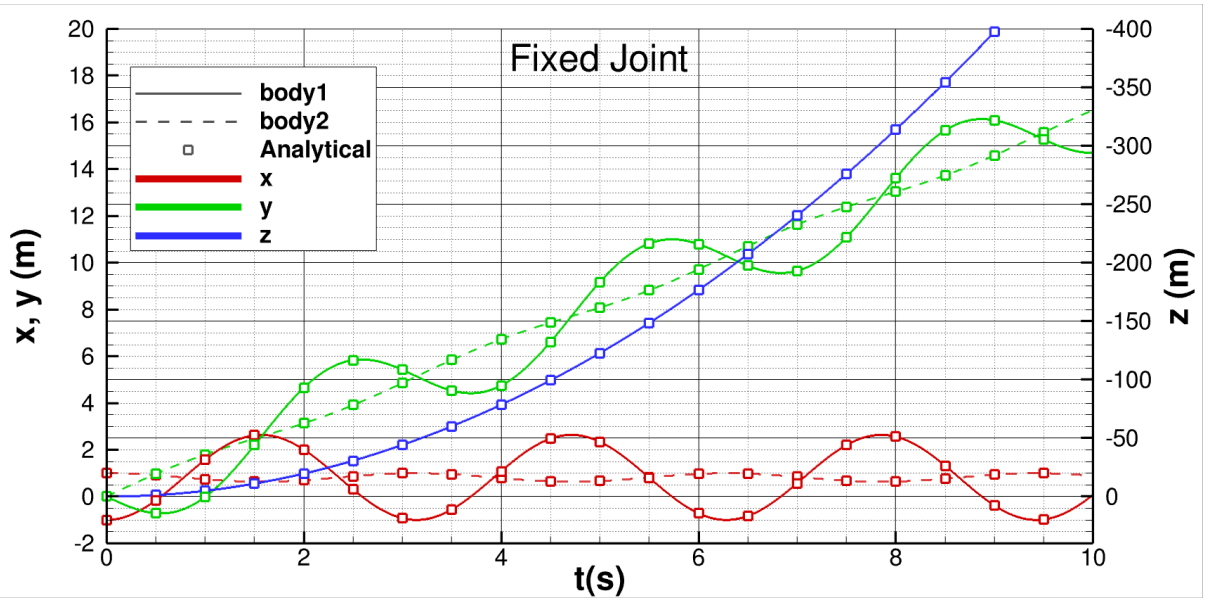

Fig 5. Evolution of the center of gravity position of both cubes (lines) along the three axes (x, y, z) for the fixed joint test case. Comparison to the analytical solution (symbols).

The evolution of the position is similar between the simulation and the analytical solution. Along the z-axis, the evolution is identical for the two cubes, and the difference in mass has no influence. Moreover, the evolution along the x and y axes shows sinusoidal and periodic characteristics, demonstrating that the cubes rotate around the center of the joint.

4.1.2. Hinge joint

A hinge joint allows only one degree of freedom in rotation. The elementary test case used to validate this joint represents a pendulum: one cube is fixed and the other is subjected to gravity (see Fig. 6).

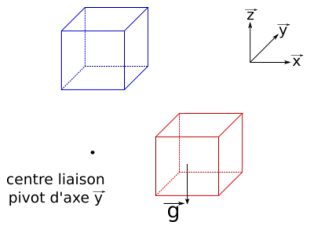

Fig 6. Representation of the hinge joint case

The analytical solution is quite complicated if not linearized around small angles. It also depends on the inertia tensor of the cube since some energy is transferred to its rotational kinetic energy.

Fig. 7 shows the evolution of the coordinates of the centers of gravity of the moving cube obtained from the simulation and from a simple numerical integration of the analytical differential equation for different inertia tensor. I_0 reprensent the inertia tensor of a 1 m side cube, the other curves corresponding to 10 and 100 times this value. As expected, the higher the inertia value, the lower the angular frequency.

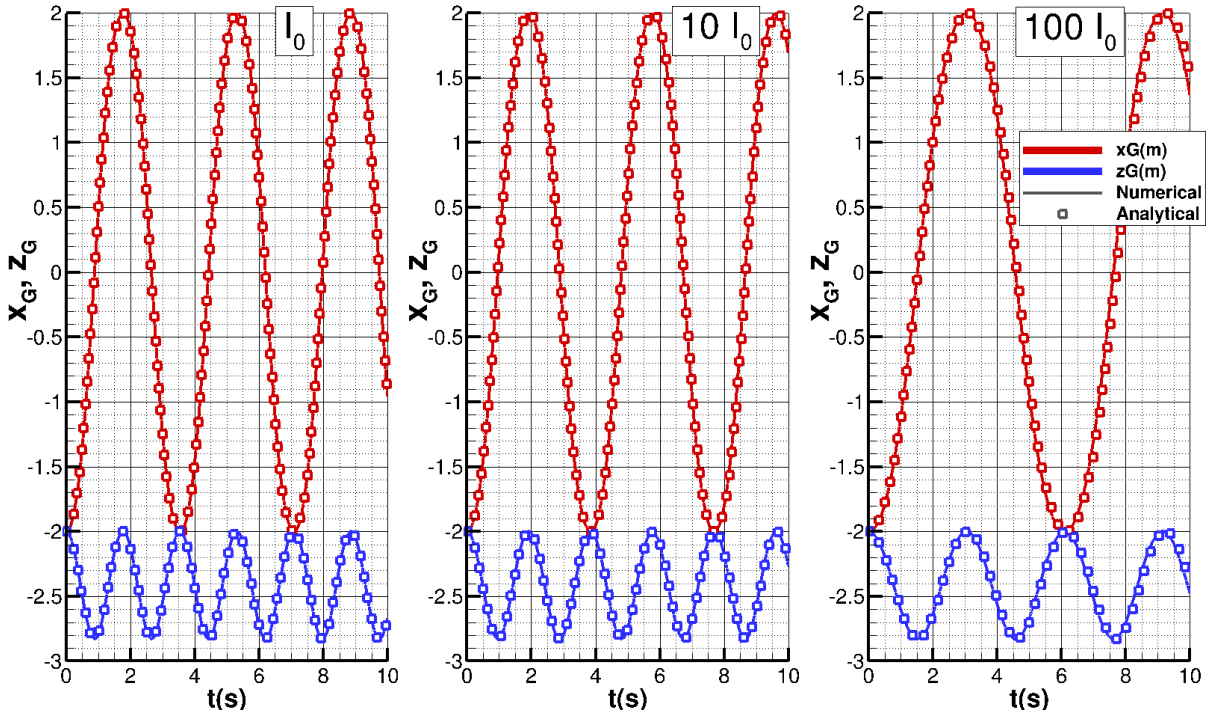

Fig 7. Evolution of the center of gravity position of cubes along the three axes (x, y, z) for the hinge joint test case.

4.1.3. Slider joint

A slider joint allows only one degree of freedom in translation. This validation case involves four cubes connected in pairs by four slider joints, as illustrated in Fig. 8. One of the cubes is subjected to a force with components along the x- and z-axes. Another cube is fixed to prevent it from being displaced by this force, by imposing a huge mass for the cube 3. The mass of cube 1 is 1 kg, 2 kg for cube 2 and 4 kg for cube 4, while the force components are $F = (-3, 0, 2)$ (N).

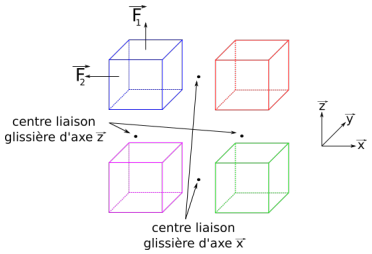

Fig 8. Representation of the slider joint case

The analytical solution, derived using the principle of fundamental dynamics (PFD), yields the following equations:

$$x_1(t) = 0.5\frac{F(x)}{m_1 + m_4}t^2 + x_1(0) \qquad x_4(t) = 0.5\frac{F(x)}{m_1 + m_4}t^2 + x_4(0)$$

$$y1(t) = 0$$

$$z_1(t) = 0.5\frac{F(z)}{m_1 + m_2}t^2 + z_1(0) \qquad z_2(t) = 0.5\frac{F(z)}{m_1 + m_2}t^2 + z_2(0)$$

Fig. 9 shows the evolution of the centers of gravity of the four cubes obtained from the simulation and the analytical solution, respectively.

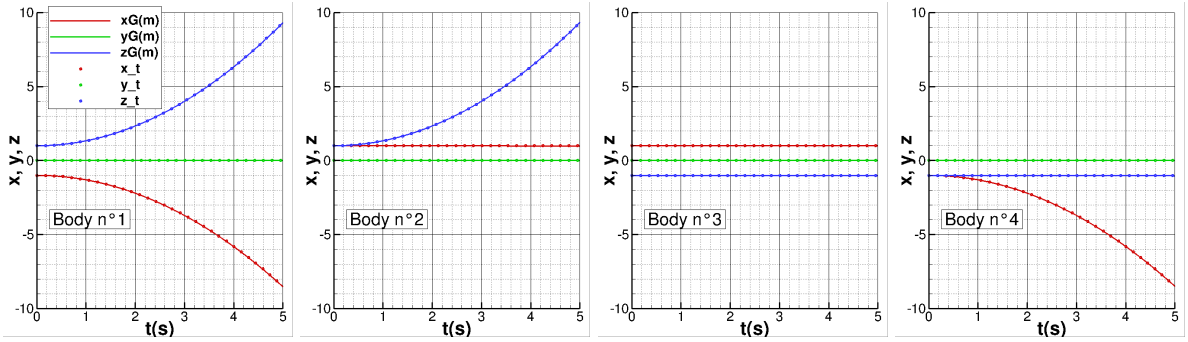

Fig 9. Evolution of the center of gravity position of cubes along the three axes (x, y, z) for the slider joint test case (numerical and theoritical results).

The positions evolve in the same way in both the analytical method and the CEDRE simulation. Therefore, this case is validated, and the correct functioning of the prismatic joint is confirmed. The implementation of the joints is thus stable in a hyperstatic scenario.

4.1.4. Joint failure

One of the additional functionalities developed is joint failure. To validate this feature, the hinge joint case study is reused with an imposed stress-type break condition on the joint. The connection is released when the joint force along the z-axis, Z_{21}, reaches 14 N.

The evolution of the positions and the forces applied on the joint are shown in Fig. 10.

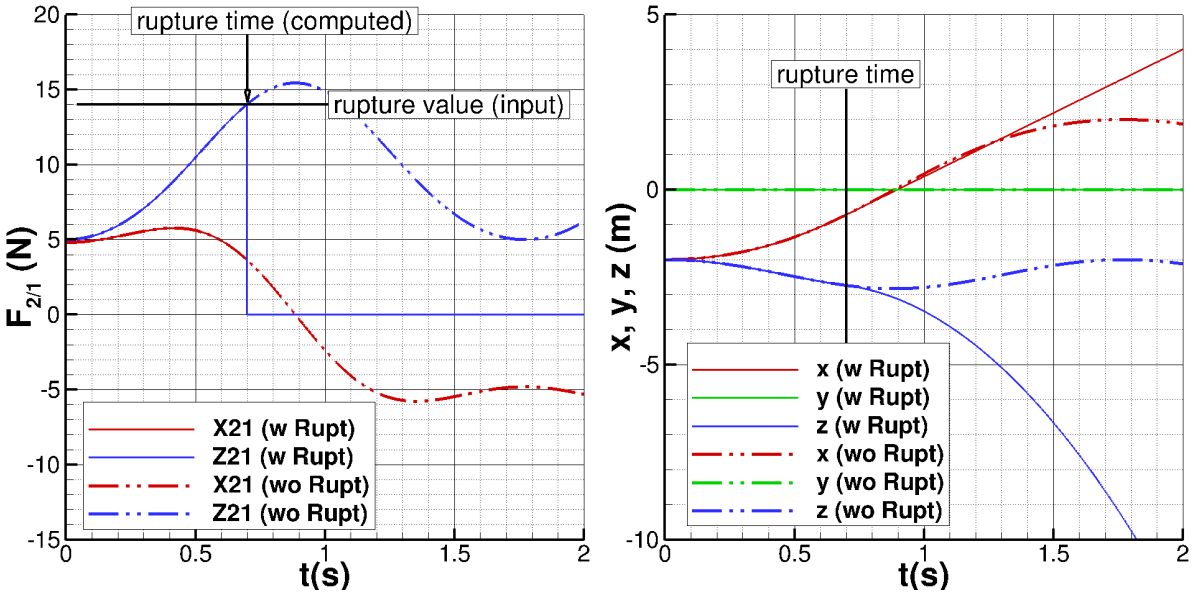

Fig 10. Efforts on the joint (left, x and z components) with gravity center coordinates evolution (right).

The force value reaches 14 N, triggering the release of the joint and satisfying the failure condition. The break occurs at $t = 0.7$ s. Up to that point, the behavior of B2 is similar to that observed during the hinge joint validation. Subsequently, the position along the z-axis is free to fall, while the position along the x-axis evolves linearly, confirming the expected phenomenon. Thus, the joint force failure condition is respected.

4.2. Ariane 64 lift-off

5. Test case description

This test cases computes the Ariane 64 (Ariane 6 with 4 solid rocket boosters) lift-off. Multiple moving (or fixed) bodies exists in this test case:

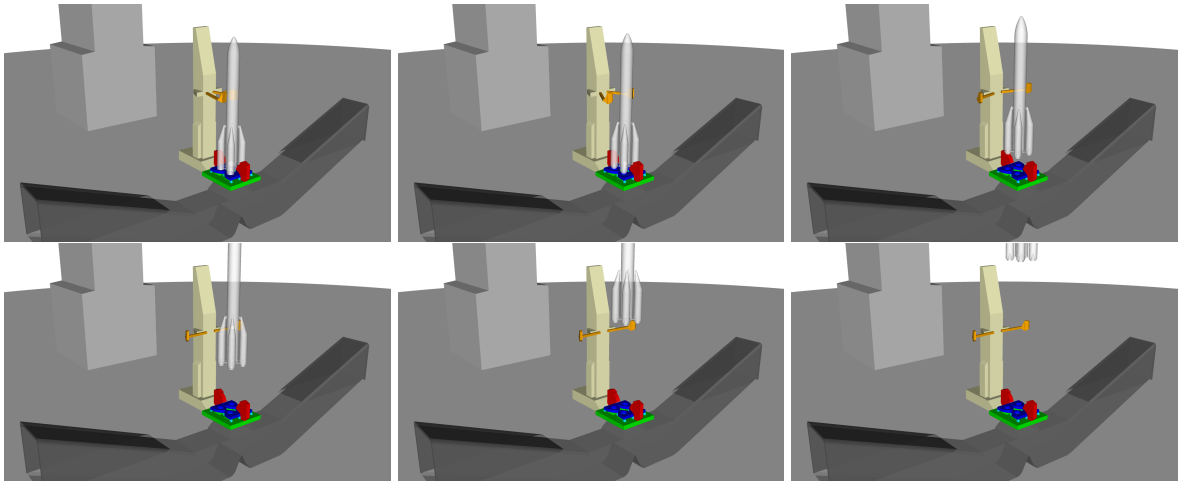

Fig 11. Evolution of the moving geometry with time (every second from t0+2s to t0+7s).

- first of all, the launchpad (the ground), with an infinite mass and inertia tensor: this is the fixed frame.
- second, the launcher, along with its own mesh. The launcher is linked to the launchpad through a slide joint with a mechanical stop at the table contact on the launchpad.
- on the launchpad, two moving arms containing the feeding lines of the upper cryogenic stage. They are both linked to the launchpad through a vertical revolute joint.
- on each arm, two moving hatches to protect the feeding lines during lift-off. Thoses hatches are linked to the arms through revolute joints as well.

The arms as well as their hatches are subject to tabulated torque, while the launcher is subject to multiple forces:

- its own weight (considered as constant here.
- the launchpad table reaction if there is a contact between launcher and launchpad (which is the case as long as the engines thrust is not sufficient to ensure lift-off).
- the engines thrust: the Vulcain engine is started 2 seconds before the solid rocket boosters, but its thrust is not sufficient to ensure lift-off. Once the boosters are started, the total thrust is larger than the launcher weight, which leads to lift-off.

The following time frame is computed:

- t0: beginning of the computation. Vulcain engine is started.
- t0+1s: Vulcain engine nominal pressure is reached.
- t0+2s: ESR ignition ; a torque is applied on both arms and hatches to retract arms and close hatches.
- t0+2.5s: ESR nominal pressure is reached.
- t0+10s: end of simulation

The time of lift off as well as the altitude evolution of the launcher are computational results since the efforts are integrated on the launcher surfaces and integrated through the MMD module.

6. Results

An evolution of the geometry (launcher lift-off, arms opening, hatches closing) is presented in Fig. 11

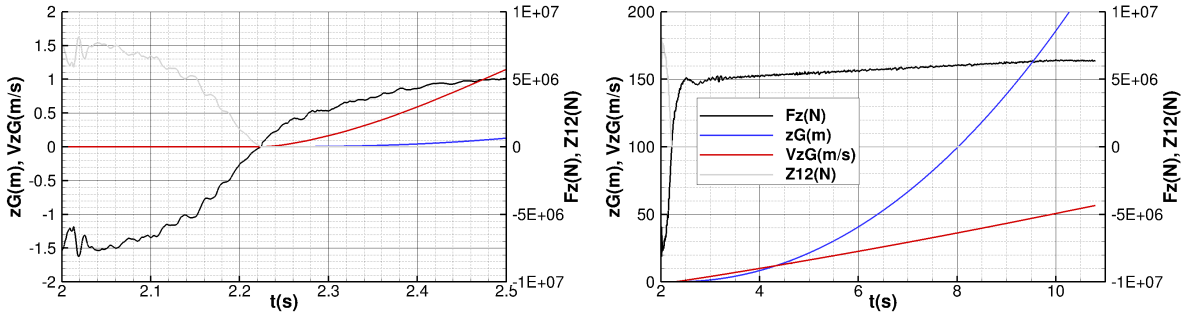

Fig 12. Forces (weight and thrust), altitude and velocity of the launcher with time. First stages after booster ignition (left) and full computation (right).

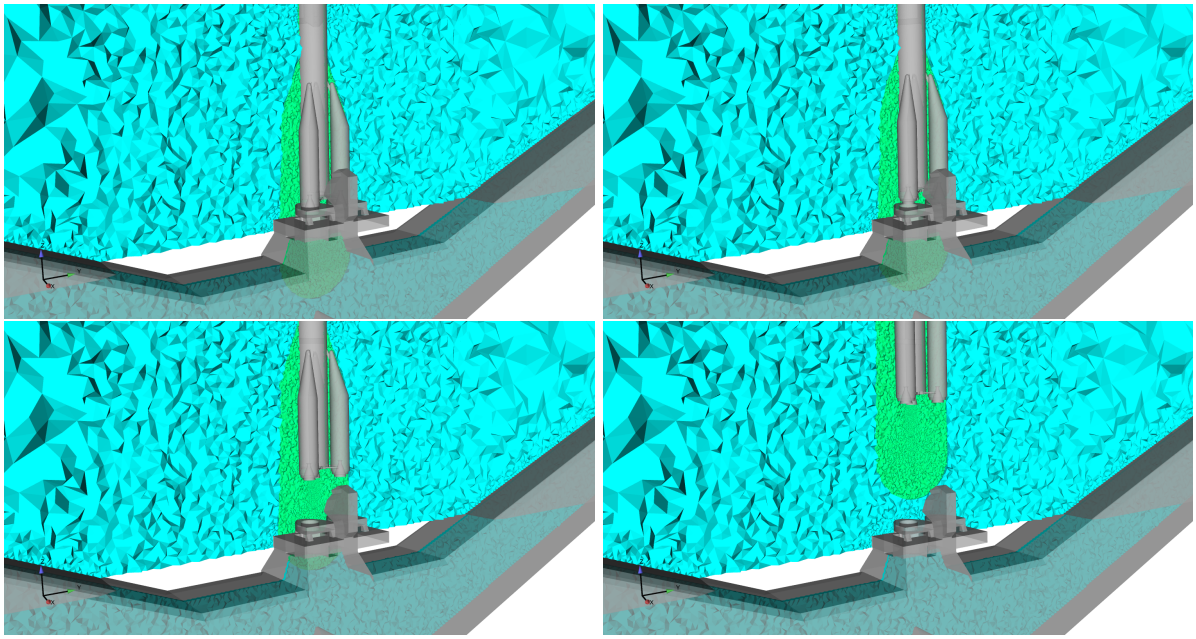

Fig 13. Mesh evolution with time (blue: launchpad, green: Ariane64).

The integrated forces (along with the weight of the vehicle) are plotted in Fig. 12. On the left of this figure, the early stage after booster ignition shows that the altitude and velocity of the moving launcher are zero while the forces F_z are negative, which means that the launcher relies on the launch table: the reaction force, computed by the joint resolution method, is plotted in grey, and is exactly the opposite of the weight+thrust components. When the thrust is sufficient to compensate the launcher weight, the velocity and altitude begins to rise, and the reaction force becomes zero as there are no more contact between the launcher and the launch table.

The computation mesh is shown in Fig. 13 during the first 3 seconds of the launch. The launchpad mesh (in blue) is intersected by the launcher mesh (in green), but during the early phase of the launch, the walls boundary conditions of the launchpad mesh also intersect the green, moving mesh.

Finally, the temperature field evolution during the lift off is presented in Fig. 14. The mesh intersection zone is not visible on these fields, which

7. Conclusion

This article presented two dynamic mesh modules, focusing on their most recent developments.

The mesh intersection module can now operate in fully parallel mode. In the Ariane 64 lift-off configuration, approximately 100,000 intersection problems were solved at each of the 1,000,000 time steps,

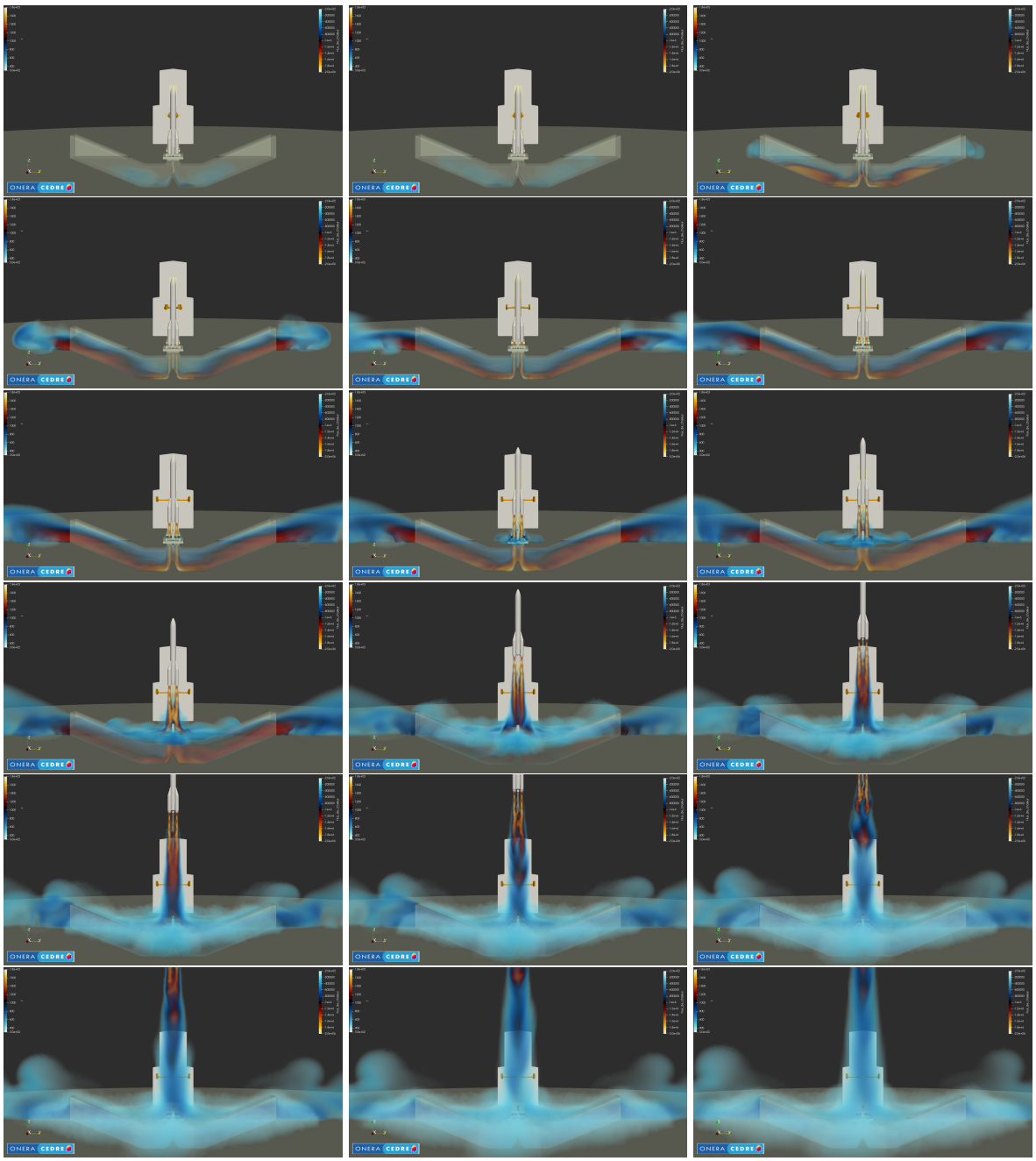

Fig 14. Volumetric temperature field evolution during the launch.

resulting in an estimated 100 billion elementary computations—completed without any issues.

The mesh displacement module was also presented, along with the theoretical background needed to handle joints between moving bodies. Several elementary test cases were introduced, demonstrating the robustness of the method, even in hyperstatic configurations. Eleven standardized joint types were implemented in the CEDRE code. Additionally, features such as joint limits and joint failure handling were developed for simple configurations

These modules were subsequently applied to a realistic launcher lift-off scenario. This approach is applicable to a wide range of configurations, such as moving parts under flight conditions (e.g., nozzle braking, grid-fin deployment), launcher touchdown with leg deployment, and evaluating the load levels on joints that may be heavily stressed during these dynamic phases.

Future work could focus on enhancing the joint module by introducing features such as friction, contact detection, and rebound, or by incorporating technological joints like springs and shock absorbers.

References

[1] J.L. Steger, F.C. Dougherty, and J.A. Benek. A chimera grid scheme. *Advances in Grid Generation*, 5, June 1983.

[2] J.A. Benek, J.L. Steger, F.C. Dougherty, and P.G. Buning. Chimera: A grid-embedding technique. Technical Report AEDC-TR-85-64, 1986.

[3] Joseph L. Steger and John A. Benek. On the use of composite grid schemes in computational aerodynamics. *Computer Methods in Applied Mechanics and Engineering*, 64(1):301–320, 1987.

[4] P. Brenner. Three-dimensional aerodynamics with moving bodies applied to solid propellant. In *AIAA paper 91-2304*, 1991.

[5] P. Brenner. Numerical simulation of 3d and unsteady aerodynamics about bodies in relative motion applied to tsto separation. In *AIAA paper 93-5142*, 1991.

[6] P. Brenner. Simulation du mouvement relatif de corps soumis a un ecoulement instationnaire par une mdthode de chevauchement de maillages. In *AGARD FDP Symposiums : Progress and Challenges in CFD Methods and Algorithms*, 1995.

[7] W. F. Phillips, C. E. Hailey, and G. A. Gebert. Review of attitude representations used for aircraft kinematics. *Journal of Aircraft*, 38(4):718–737, July 2001.

[8] W. F. Phillips, C. E. Hailey, and G. A. Gebert. Review of attitude representations used for aircraft kinematics - errata. *Journal of Aircraft*, 40(1):223–223, January 2003.

[9] Loup Verlet. Computer ”experiments” on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98–103, Jul 1967.

[10] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1(1):1–16, 1972.

[11] E. Catto. Iterative dynamics with temporal coherence. In *Games Developers Conference*, 2005.

[12] E. Catto. Fast and simple physics using sequential impulses. In *Games Developers Conference*, 2006.

[13] E. Catto. Modeling and solving constraints. In *Games Developers Conference*, 2009.

[14] Marijn Tamis and Giuseppe Maggiore. Constraint based physics solver - v1.02, 2015.