A fully-conservative sliding grid algorithm for compressible flows using an Isogeometric Discontinuous Galerkin scheme

Stefano Pezzano, Régis Duvigneau

A fully-conservative sliding grid algorithm for compressible flows using an Isogeometric Discontinuous Galerkin scheme

Stefano Pezzano, Régis Duvigneau

Université Côte d'Azur, INRIA, CNRS, LJAD
INRIA Sophia-Antipolis, 2004 route des Lucioles - B.P. 93
06902 Sophia-Antipolis, France

Abstract

This work aims at developing a high-order, fully conservative, discretization of sliding grids with applications to compressible flows at different regimes (from subsonic to supersonic). The proposed approach combines a discontinuous Galerkin formulation for Navier-Stokes equations with rational representations originating from Isogeometric Analysis, which allows to design a watertight and fully conservative sliding grid algorithm. A verification exercise is first carried out to rigorously establish the convergence rate of the method. Then, the accuracy and robustness are demonstrated for a flow around a pitching ellipse at different regimes. A comparison with a grid deformation technique and a sensitivity study with respect to the location of the sliding interface are also investigated. Finally, the flow around a vertical-axis wind turbine configuration is simulated to show the potentiality of the proposed approach to deal with more complex geometries.

1 Introduction

A significant number of engineering flow problems involve rotating components. This is, for example, the case of wind turbines, compressors, helicopters, and many others. Often, in such applications, a part of the geometry remains fixed, whereas other parts rotate. A specific strategy is therefore necessary to represent the time evolution of the computational domain. The use of a continuous deformation technique is usually discarded due to the necessity of dynamic remeshing when confronted with large displacements of the boundary. For this reason, the simulation of flows involving rotating objects is usually carried out using sliding grids, in which the computational domains is subdivided into a fixed and a rotating region. However, the discontinuous mesh movement between the fixed and the moving subdomain generates hanging nodes along the sliding interface. Moreover, when piecewise linear grids are employed, the computational domain is characterized by gaps and overlaps. Therefore, specific numerical methods are required to treat the issues introduced by the sliding movement.

In this context, Discontinuous Galerkin (DG) schemes are appealing because they are natively capable of handling geometric non-conformities. Ferrer et al. (1) proposed the first DG method with sliding grids for incompressible flows. On the other hand, Isogeometric Analysis (IGA) (2) is well adapted to rotating grids because circular interfaces can be exactly represented, thanks to the Non-Uniform Rational B-Splines (NURBS) representation, avoiding the emergence of holes and overlaps. In the framework of IGA, DG formulations can be

adopted to handle geometric non-conformities between NURBS patches (3). In particular, Bazilevs et al. (4) combined a Continuous Galerkin (CG) scheme based on quadratic NURBS with a DG discretization of the sliding interface to solve the incompressible Navier-Stokes equations. An alternative strategy is to employ mortars to deal with the hanging nodes. The mortar approach is based on projections and it was first introduced in the context of spectral element methods to treat non-conforming meshes (5; 6). More recently, the mortar method has been adopted to simulate compressible flows with sliding meshes, combined with compact high-order discretizations (7; 8). Mortar based methods, however, suffer from loss of conservativity when dealing with curvilinear sliding interfaces, due to the nature of the projections (8). Moreover, as evidenced in (9), interpolation based DG methods are significantly less expensive than mortar approaches.

In this work, we propose an Isogeometric DG method with sliding meshes for compressible flow problems. The NURBS-based representation is exploited to define a watertight interface between the rotating and the fixed subdomain, whereas the DG formulation allows to obtain a fully conservative numerical scheme, suitable for compressible flows. The developed algorithm is explained in details in section 3. The methodology is then verified using a classic benchmark for compressible flows and a thorough error analysis is conducted. Section 5 is dedicated to the investigation of the viscous flow around a pitching ellipse, with the aim of testing the sliding mesh algorithm on a more demanding test case. In particular, a mesh sensitivity analysis is performed, with respect to the grid refinement level, the basis degree and the position of the sliding interface. Moreover, a comparison with a grid deformation approach is presented. We also simulate the pitching ellipse in a supersonic flow condition to demonstrate the ability of the sliding mesh algorithm to deal with shocks. Lastly, in order to prove the potential of the Isogeometric DG framework for more complex geometries, a 3-bladed vertical axis wind turbine configuration is considered in section 6. In particular, the generation of a suitable computational grid is discussed and a flow simulation is carried out. Finally, the main results are outlined in the conclusion and some research perspectives are discussed.

2 NURBS-based DG with moving meshes

In order to simulate fluid flow problems with moving geometries, we consider the Navier-Stokes equations in ALE form, with a computational domain which moves with a generic velocity $\mathbf{V}_g$. The two-dimensional case is presented here, but the proposed approach can be generalized to the three-dimensional case. The divergence form of the equations is the following:

$$\frac{\partial \mathbf{W}}{\partial t} + \nabla \cdot (\mathbf{F}_c - \mathbf{F}_v) - \mathbf{V}_g \cdot \nabla \mathbf{W} = 0, \tag{1}$$

with $\mathbf{W}$ being the vector of conservative variables, $\mathbf{F}_c$ the convective flux, and $\mathbf{F}_v$ the viscous flux:

$$\mathbf{W} = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho e \end{pmatrix}, \quad \mathbf{F}_{c,i} = \begin{pmatrix} \rho u_i \\ \rho u_1 u_i + p\delta_{1i} \\ \rho u_2 u_i + p\delta_{2i} \\ \rho u_i (e + \frac{p}{\rho}) \end{pmatrix}, \quad \mathbf{F}_{v,i} = \begin{pmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ u_k \tau_{ki} - q_i \end{pmatrix}, \tag{2}$$

where τ_{ij} is the viscous stress tensor and q_i is the thermal conduction flux, defined as:

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \frac{\partial u_k}{\partial x_k} \delta_{ij}, \tag{3}$$

$$q_i = -\gamma \frac{\mu}{Pr} \frac{\partial e}{\partial x_i}, \tag{4}$$

where $\gamma = 1.4$, $Pr = 0.72$ and μ is determined by the Reynolds number. For inviscid flows, the Euler equations are considered and the viscous flux $\mathbf{F}_v$ term is ignored.

The second order derivatives introduced by the viscous part of the equations are treated using the Local Discontinuous Galerkin (LDG) approach (10). To this end, we reduce eq. (1) to a system of first order equations by introducing the auxiliary variable $\mathbf{G}$:

$$\begin{cases} \dfrac{\partial \mathbf{W}}{\partial t} + \nabla \cdot \mathbf{F}_c(\mathbf{W}) - \nabla \cdot \mathbf{F}_v(\mathbf{W}, \mathbf{G}) - \mathbf{V}_g \cdot \nabla \mathbf{W} = 0, \\ \mathbf{G} - \nabla \mathbf{W} = 0. \end{cases} \tag{5}$$

In order to obtain a discrete solution, the NURBS-based DG method relies on Rational Bernstein functions(11):

$$R^p_{i_1 i_2}(\xi, \eta) = \frac{B^p_{i_1}(\xi)\, B^p_{i_2}(\eta)\, \omega_{i_1 i_2}}{\sum_{j_1=1}^{p+1} \sum_{j_2=1}^{p+1} B^p_{j_1}(\xi)\, B^p_{j_2}(\eta)\, \omega_{j_1 j_2}}, \tag{6}$$

where ξ and η are the parametric coordinates, defined in the parametric domain $\widehat{\Omega} = [0,1]^2$. The coefficients $\omega_{j_1 j_2}$ are the weights and B^p_j are the Bernstein polynomials of degree p. According to the isogeometric paradigm, a unified mathematical representation is adopted for the geometry $\mathbf{x}$ and the discrete solution fields $\mathbf{w}_h$ and $\mathbf{g}_h$. Thus, in each element, we can write:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{w}_h \\ \mathbf{g}_h \end{pmatrix} = \sum_{i=1}^{(p+1)^2} R_i(\xi, \eta) \begin{pmatrix} \mathbf{x}_i \\ \mathbf{w}_i \\ \mathbf{g}_i \end{pmatrix}. \tag{7}$$

The computational mesh is thus a tessellation of rational Bézier patches, whose geometry is described by the control points $\mathbf{x}_i$, whereas $\mathbf{w}_i$ and $\mathbf{g}_i$ are respectively the degrees of freedom (DOF) of the conservative variables and their gradients. The eqs. (5) are discretized using the direct ALE method developed in (12), leading to the following weak formulation:

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{w}_i \displaystyle\int_{\widehat{\Omega}} R_k R_i \, |\mathrm{J}_{\Omega_j}| \, \mathrm{d}\widehat{\Omega} \right) = \displaystyle\int_{\widehat{\Omega}} \nabla R_k \cdot (\mathbf{F}_c - \mathbf{F}_v - \mathbf{V}_g \mathbf{w}_h) \, |\mathrm{J}_{\Omega_j}| \, \mathrm{d}\widehat{\Omega} \\ \qquad\qquad - \displaystyle\oint_{\partial\widehat{\Omega}} R_k (\mathbf{F}^*_{ale} - \mathbf{F}^*_v) \, |\mathrm{J}_{\Gamma_j}| \, \mathrm{d}\widehat{\Gamma}, & \text{(8a)} \\ \mathbf{g}_i \displaystyle\int_{\widehat{\Omega}} R_k R_i \, |\mathrm{J}_{\Omega_j}| \, \mathrm{d}\widehat{\Omega} = \displaystyle\int_{\widehat{\Omega}} \nabla R_k \mathbf{w}_h \, |\mathrm{J}_{\Omega_j}| \, \mathrm{d}\widehat{\Omega} - \displaystyle\oint_{\partial\widehat{\Omega}} R_k \mathbf{W}^* \, |\mathrm{J}_{\Gamma_j}| \, \mathrm{d}\widehat{\Gamma}. & \text{(8b)} \end{cases}$$

The convective numerical flux $\mathbf{F}^*_{ale}$ is computed through a modified HLL Riemann solver (12), whereas the LDG flux (10) is used for $\mathbf{F}^*_v$ and $\mathbf{W}^*$. Integration is carried out in the parametric domain $\widehat{\Omega}$ using Gauss-Legendre formulas. Due to the mapping between the physical and the parametric space, the Jacobians $|\mathrm{J}_{\Omega_j}|$ and $|\mathrm{J}_{\Gamma_j}|$ appear in the integrals. Equation (8b) is solved within each time iteration in a decoupled manner and the system of equations (8) can be rewritten as:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathcal{M}\mathbf{w}) = \mathcal{R}(\mathbf{w}_h, \mathbf{V}_g), \tag{9}$$

where $\mathcal{M}$ is the mass matrix and the residual $\mathcal{R}$ is the right-hand side of eq. (8a). The resulting set of ordinary differential equations (ODE) (9) is integrated in time using a 4th-order four-stage explicit Runge-Kutta (RK) scheme. Moreover, it is possible to further simplify the time

integration algorithm when the mesh movement is rigid. Indeed, it is trivial to show that the mass matrix is constant in time when the elements are rotating and translating. We thus obtain:

$$\mathcal{M}\frac{\mathrm{d}\mathbf{w}}{\mathrm{d}t} = \mathcal{R}(\mathbf{w}_h, \mathbf{V}_g). \tag{10}$$

In this case, the mesh movement contributions are limited to the flux terms due to the grid velocity $\mathbf{V}_g$. Integrating equation (10) is significantly less expensive from the computational standpoint, as the inverse of the mass matrix is computed in the pre-processing phase.

Finally, artificial viscosity terms are added in the case of discontinuous solutions, based on the subcell shock capturing procedure presented in (13) extended to rational representations (14; 15).

3 Sliding mesh algorithm

In this section we explain in details the treatment of sliding meshes. The algorithm is characterized by three phases. First, the geometry is updated and the control points of the mesh are moved. As the movement is performed, the connectivity between the interface elements is checked and updated. Once the grid connectivity is established, the interface is subdivided into smaller arcs shared by only two elements. Lastly, a special treatment of the sliding faces is required in order to accurately compute the flux integrals. These three phases are detailed in the following subsections.

3.1 Mesh movement and connectivity update

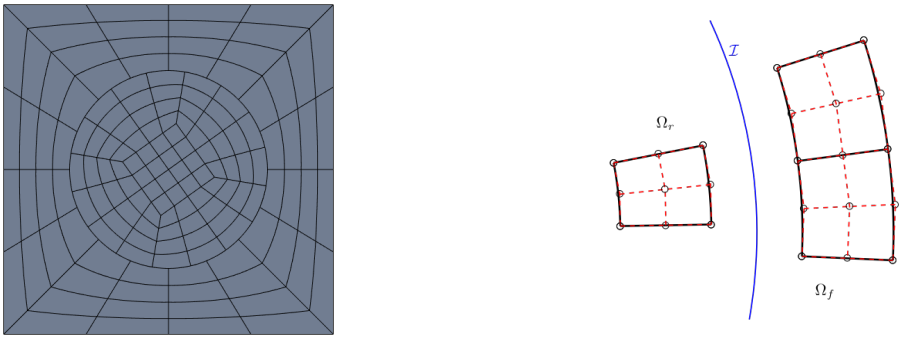
For the sake of simplicity, we analyse the case of a single rotating subdomain Ω_r surrounded by a fixed region Ω_f. The sliding interface is thus defined as $\mathcal{I} = \partial\Omega_r \cap \partial\Omega_f$. Following the isogeometric approach, the grid velocity field is represented using the same basis adopted for the DG formulation:

$$\mathbf{V}_g = \sum_{i=1}^{(p+1)^2} R_i(\xi, \eta)\mathbf{v}_{g,i}\,, \tag{11}$$

where the $\mathbf{v}_{g,i}$ is the velocity of the i-th control point $\mathbf{x}_i$. Since the mesh velocity is described as a DG field, it is naturally possible to represent discontinuous displacements, which is particularly useful in the context of sliding meshes. Indeed, the sliding interface is characterized by a discontinuity of the tangential velocity $\mathbf{V}_g \cdot \mathbf{t}$. On the other hand, the normal component $\mathbf{V}_g \cdot \mathbf{n}$ must be continuous, in order to keep the mesh watertight. For perfectly circular interfaces, the control point velocities $(u_{g,i},\, v_{g,i})$ can be easily computed thanks to the affine invariance property of Bézier patches (11). Defining the angular velocity ω and the axis of rotation $O = (x_0, y_0)$, we have, for each control point $(x_i,\, y_i)$ in the rotating subdomain Ω_r:

$$\begin{cases} u_{g,i} = -\omega(y_i - y_0), \\ v_{g,i} = \omega(x_i - x_0). \end{cases} \tag{12}$$

Assuming that the elements are initially aligned, the discontinuity in the mesh movement generates hanging nodes along the sliding interface, as it can be observed in Fig. 1. Consequently, the degrees of freedom of the elements adjacent to the interface are not aligned. As the inner domain rotates, the position of the hanging nodes changes as well. In order to



(a) Domain subdivision

(b) Degrees of freedom along the interface

Figure 1: Generation of hanging nodes due to the sliding movement

update the mesh connectivity, we rely on the computation of two angles, as represented in Fig. 2. $\Delta\theta_0$ is the angular amplitude of the element edge between $\mathbf{x}_1$ and $\mathbf{x}_2$, whereas $\Delta\theta_s$ is the angular amplitude of the circular arc between $\mathbf{x}_1$ and the hanging node $\mathbf{x}_s$. We then define the ratio s as:

$$s = \frac{\Delta\theta_s}{\Delta\theta_0}\,. \tag{13}$$

The current connectivity is valid when $0 < s < 1$, whereas it must be updated when the angular ratio s becomes negative or greater than the unity.

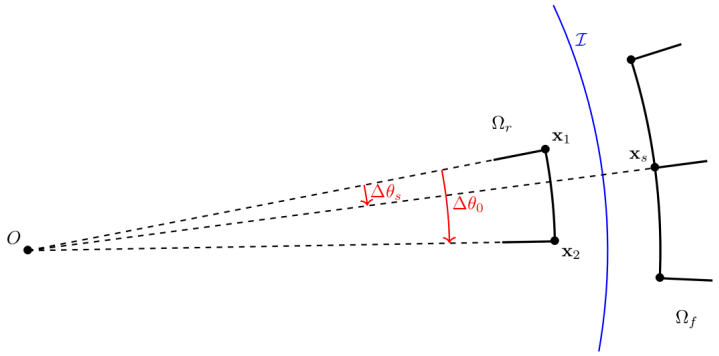


Figure 2: Definition of the angles for connectivity update

3.2 Interface splitting

The second step of the sliding algorithm is the subdivision of the interface into a set of elemental faces, which are shared uniquely by two elements. Let us name Ω_0 the rotating element, Ω_1 and Ω_2 the corresponding fixed elements, as illustrated in Fig. 3. The elemental sliding faces are therefore $\Gamma_1 = \partial\Omega_0 \cap \partial\Omega_1$ and $\Gamma_2 = \partial\Omega_0 \cap \partial\Omega_2$. We also define $\Gamma_0 = \partial\Omega_0 \cap \mathcal{I} = \Gamma_1 \cup \Gamma_2$.

The splitting procedure relies on the properties of the NURBS representation. Indeed, a rational Bézier curve of degree p defined on the parametric interval $(0, 1)$ can be interpreted

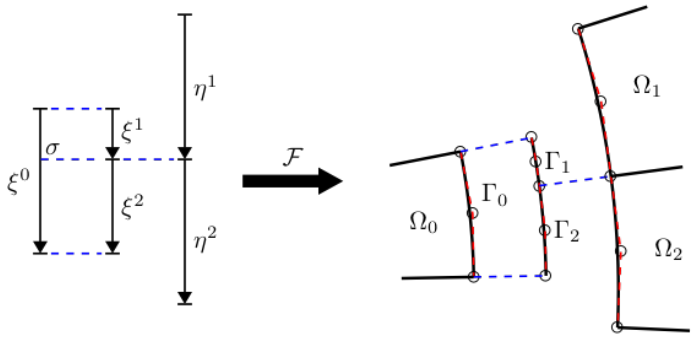


Figure 3: Splitting procedure in the parametric and physical spaces

as a NURBS curve with a the following knot vector:

$$\Xi = \{\underbrace{0, \ldots, 0}_{p+1}, \underbrace{1, \ldots, 1}_{p+1}\}. \tag{14}$$

Let us define the splitting parameter σ as the value of ξ^0 that corresponds to the physical position of the hanging node $\mathbf{x}_s$ on the face Γ_0. It is then possible to divide Γ_0 into the two elemental faces, Γ_1 and Γ_2, by applying $(p+1)$-times the knot insertion algorithm (11) at the parametric coordinate $\xi^0 = \sigma$. We thus obtain two Bézier curves with the following knot vectors:

$$\Xi^1 = \{\underbrace{0, \ldots, 0}_{p+1}, \underbrace{\sigma, \ldots, \sigma}_{p+1}\}, \quad \Xi^2 = \{\underbrace{\sigma, \ldots, \sigma}_{p+1}, \underbrace{1, \ldots, 1}_{p+1}\}. \tag{15}$$

The control points of the new curves are obtained by recursively using p times the knot insertion formula, specialized here for Bézier representations:

$$\mathbf{x}_i^{r+1} = (1-\alpha_i^r)\mathbf{x}_{i-1}^r + \alpha_i^r \mathbf{x}_i^r, \quad \alpha_i^r = \begin{cases} 1 & \text{if } i \leq r+1 \\ \sigma & \text{if } r+2 \leq i \leq p+1 \\ 0 & \text{if } i \geq p+2 \end{cases} \tag{16}$$

with $r = 0, \ldots, (p-1)$. The result of the splitting procedure is illustrated in Fig. 3 for a quadratic representation.

In order to adopt the described splitting algorithm, the value of σ must be known. Due to the rational nature of the representation, computing σ is not trivial. The procedure that allows to determine the parameter corresponding to a given point of a rational Bézier curve is the point inversion algorithm (11). If we denote $\mathbf{C}$ as the Bézier representation of the Γ_0 curve, we can write:

$$\mathbf{C}(\sigma) = \frac{\sum_{i=1}^{p+1} B_i^p(\sigma)\,\omega_i\,\mathbf{x}_i}{\sum_{j=1}^{p+1} B_j^p(\sigma)\,\omega_j} = \mathbf{x}_s, \tag{17}$$

where ω_i and $\mathbf{x}_i$ are respectively the weights and the control points of the considered edge. The problem of determining σ is solved by minimizing the square of the distance between the curve $\mathbf{C}$ and the hanging node:

$$g(\xi^0) = ||\mathbf{C}(\xi^0) - \mathbf{x}_s||^2. \tag{18}$$

Imposing $g'(\sigma) = 0$, one obtains:

$$f(\sigma) = \mathbf{C}'(\sigma) \cdot \big(\mathbf{C}(\sigma) - \mathbf{x}_s\big) = 0. \tag{19}$$

The solution of eq. (19) is achieved by means of an iterative algorithm. Piegl et al. (11) adopted the Newton algorithm to solve the point inversion problem. Such an approach requires to analytically compute the first derivative of f, which contains the second derivative of the Bézier curve $\mathbf{C}$. In order to avoid the costly computation of $\mathbf{C}''$, we employ the secant algorithm, which results in the following recursive formula:

$$\sigma^i = \frac{\sigma^{i-2} f(\sigma^{i-1}) - \sigma^{i-1} f(\sigma^{i-2})}{f(\sigma^{i-1}) - f(\sigma^{i-2})}, \tag{20}$$

and we initialise the algorithm taking σ^1 equal to the angular ratio s used to update the connectivity, and σ^0 equal to the splitting parameter computed for the previous time step. The convergence conditions to stop the secant iterations are:

$$\begin{cases} f(\sigma^i) \leq 10^{-15}, \\ |\sigma^i - \sigma^{i-1}| \leq 10^{-14}. \end{cases} \tag{21}$$

Since the values used to initialise the algorithm already provide a close estimate of the actual splitting parameter, few iterations are required. In practice, we observed that the convergence conditions are always satisfied in less than 5 iterations.

3.3 Flux computation

Once the interface is split into the elemental faces, it is possible to compute the flux integrals of the DG scheme. For each elemental segment of the interface, two integrals have to be computed, one for the rotating element and one for the fixed element. The difficulty of computing the flux integrals is due to the non-matching parameterisation on the two sides of the interface. Indeed, due to the lack of alignment of the elements between Ω_r and Ω_f, the face Γ_1 is described using two different parameterisations and the same is true for Γ_2, as presented in Fig. 3. If we focus on Γ_1, the flux integrals for a numerical flux function $\mathbf{F}^*$ are:

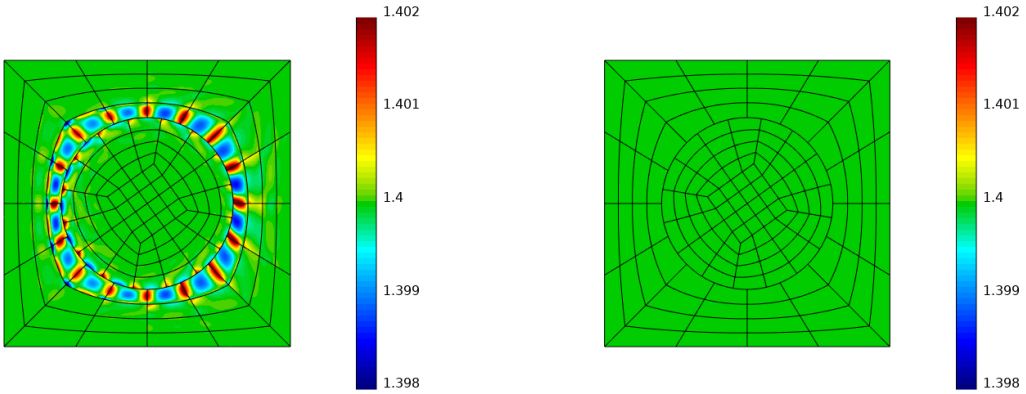
$$\mathbf{f}_1^- = \int_{\Gamma_1} R_k(\xi^1)\, \mathbf{F}^*\big(w_h^-(\xi^1), w_h^+(\eta^1), \mathbf{n}\big)\, \mathrm{d}\Gamma_1, \tag{22}$$

$$\mathbf{f}_1^+ = \int_{\Gamma_1} R_k(\eta^1)\, \mathbf{F}^*\big(w_h^-(\xi^1), w_h^+(\eta^1), \mathbf{n}\big)\, \mathrm{d}\Gamma_1. \tag{23}$$

One could naively try to use as quadrature points the Gauss-Legendre nodes on the interval $\xi^1 \in [0, \sigma]$ for $\mathbf{f}_1^-$ and on the interval $\eta^1 \in [1 - \sigma, 1]$ for $\mathbf{f}_1^+$. However, the positions of the integration points in the physical space do not match, due to the non-linearity of the coordinate transformation, resulting in a locally non-conservative scheme. The effect of the loss of conservativity is presented in Fig. 4a for a freestream preservation problem. We can clearly observe the spurious oscillations around the sliding interface.

We can instead obtain a fully conservative scheme by computing both flux integrals on the parametric interval $\xi_1 \in [0, \sigma]$. We therefore rewrite $\mathbf{f}_1^-$ and $\mathbf{f}_1^+$ as:

$$\mathbf{f}_1^- = \int_0^{\sigma} R_k(\xi^1)\, \mathbf{F}^*\big(w_h^-(\xi^1), w_h^+\big(\eta^1(\xi^1)\big), \mathbf{n}\big)\, |\mathrm{J}_{\Gamma_1}|\, \mathrm{d}\xi^1, \tag{24}$$



(a) Non-conservative interface

(b) Fully conservative interface

Figure 4: Freestream preservation with non-conservative and conservative sliding interfaces

$$\mathbf{f}_1^+ = \int_0^\sigma R_k(\eta^1(\xi^1))\,\mathbf{F}^*(w_h^-(\xi^1), w_h^+(\eta^1(\xi^1)), \mathbf{n})\,|\mathrm{J}_{\Gamma_1}|\,\mathrm{d}\xi^1, \tag{25}$$

where $|\mathrm{J}_{\Gamma_1}|$ is the Jacobian of the map between ξ_1 and Γ_1. In order to approximate expressions (25) and (24) with numerical quadrature, the reparameterisation $\eta^1(\xi^1)$ has to be computed for the quadrature points. If we denote ξ^1_{gp} as the parametric coordinate of a generic Gauss-Legendre point, then its physical coordinate is equal to:

$$\mathbf{x}_{gp} = \frac{\sum_{i=1}^{p+1} B_i^p(\xi^1_{gp})\,\omega_i\,\mathbf{x}_i}{\sum_{j=1}^{p+1} B_j^p(\xi^1_{gp})\,\omega_j} = \frac{\sum_{i=1}^{p+1} B_i^p(\eta^1_{gp})\,\upsilon_i\,\mathbf{y}_i}{\sum_{j=1}^{p+1} B_j^p(\eta^1_{gp})\,\upsilon_j}, \tag{26}$$

where $\mathbf{x}_i$ and ω_i are the control points and weights of the elemental face, whereas $\mathbf{y}_i$ and υ_i are the control points and weights of the edge of Ω_1. The reparameterisation η^1_{gp} is found by solving eq. (26) with the point inversion algorithm described in the previous section. The same treatment is then repeated for the integrals over the face Γ_2:

$$\mathbf{f}_2^- = \int_\sigma^1 R_k(\xi^2)\,\mathbf{F}^*(w_h^-(\xi^2), w_h^+(\eta^2(\xi^2)), \mathbf{n})\,|\mathrm{J}_{\Gamma_2}|\,\mathrm{d}\xi^2, \tag{27}$$

$$\mathbf{f}_2^+ = \int_\sigma^1 R_k(\eta^2(\xi^2))\,\mathbf{F}^*(w_h^-(\xi^2), w_h^+(\eta^2(\xi^2)), \mathbf{n})\,|\mathrm{J}_{\Gamma_2}|\,\mathrm{d}\xi^2. \tag{28}$$

In order to show the effectiveness of the proposed strategy, we report in Fig. 4b the result of the freestream preservation test and compare it with non-conservative integral computation. We can observe that the spurious oscillation around the interface are absent and that the constant solution is well preserved.

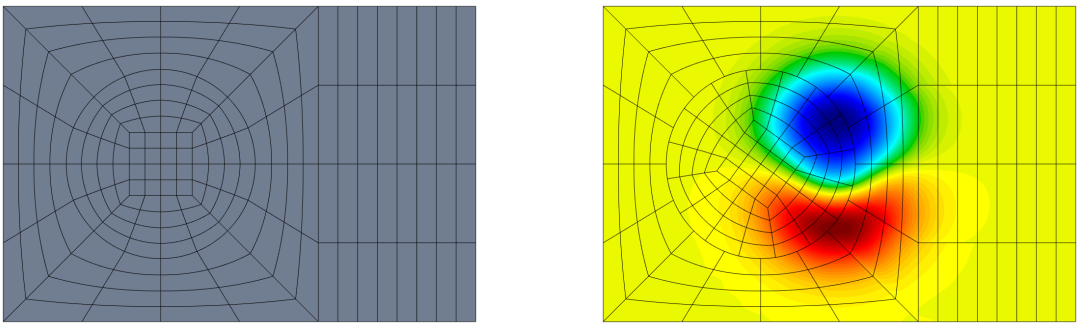
4 Verification: isentropic vortex

The developed sliding mesh technique is firstly tested on an analytic problem governed by the compressible Euler equations. The considered test case consists in the advection of an isentropic vortex, which is a common benchmark for assessing the accuracy of compressible

flow solvers (16). The time evolution of the flow is described by the following function:

$$\begin{cases} \rho = \left(1 - \dfrac{\gamma - 1}{16\gamma\pi^2}\beta^2 e^{2(1-r^2)}\right)^{\frac{1}{\gamma-1}} \\ u = 1 - \beta\dfrac{y - y_0}{2\pi}e^{1-r^2} \\ v = \beta\dfrac{x - t - x_0}{2\pi}e^{1-r^2} \\ p = \rho^\gamma \end{cases} \tag{29}$$

where $r = \sqrt{(x - t - x_0)^2 + (y - y_0)^2}$, with $x_0 = 5$, $y_0 = 0$ and $\beta = 5$. The computational domain is $[2.5, 10] \times [-2.5, 2.5]$ and the boundary fluxes are computed using the analytic solution (29).



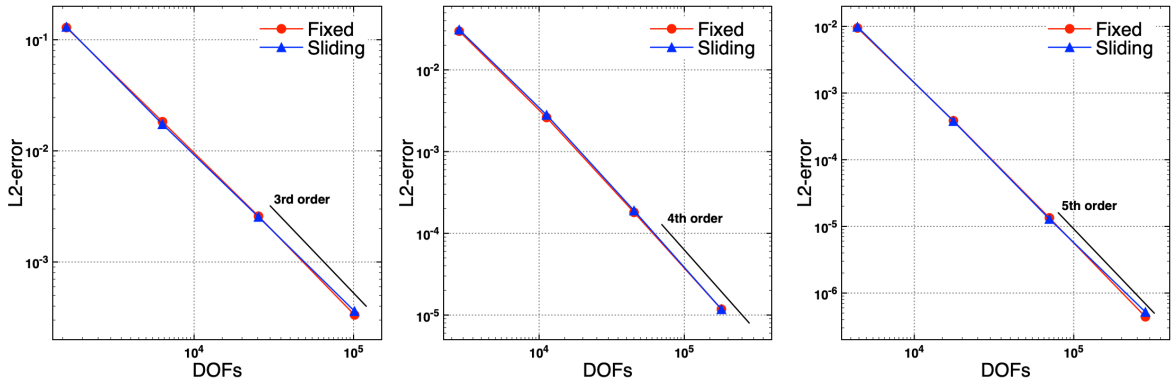
(a) Initial mesh

(b) Rotating mesh and solution field at t=1.2

Figure 5: Isentropic vortex, setup

The computational domain is divided into two regions, a rotating one and a fixed one, as illustrated in Fig. 5. The radius of the sliding interface is equal to 1.5, the centre of rotation of the inner region corresponds to the initial axis of the vortex, and a unitary rotational frequency is adopted. The core of the vortex is therefore inside the rotating subdomain in the initial phase of the simulation. As the vortex is advected downstream, it crosses the sliding interface to enter the fixed subdomain, as it can be observed in Fig. 5b. The proposed test case allows to investigate how well the sliding interface is capable of preserving the flow characteristics.

In order to perform a rigorous error analysis, a convergence study is carried out. Starting from the baseline mesh presented in Fig. 5a, isotropic refinement is applied to obtain four different levels of mesh resolution. Quadratic, cubic and quartic basis functions are tested. For each combination of degree and mesh level, a simulation is performed considering a fixed inner subdomain as well. By comparing the results obtained with the two approaches, it is possible to quantify the error introduced by the computation of the flux integrals on the sliding interface. The L^2-norm of the error for the total energy field is evaluated at $t = 2$ using numerical quadrature. The results of the convergence analysis are presented in Fig. 6. It can be observed that optimal convergence rates are obtained for all basis degrees for both fixed and sliding meshes. Furthermore, roughly the same errors are obtained with the two techniques, meaning that the error generated by the sliding interface is negligible and flow characteristics are perfectly preserved.



(a) p = 2 (b) p = 3 (c) p = 4

Figure 6: Isentropic vortex, error analysis

5 Case study: pitching ellipse flow

The second test case concerns the bidimensional flow around a pitching ellipse. Thanks to the use of rational Bézier functions, the ellipse can be exactly represented. Thus, there is no additional error introduced by the discretization of the computational domain. The aspect ratio of the ellipse is equal to 12 and its angle of attack evolves in time with the following law:

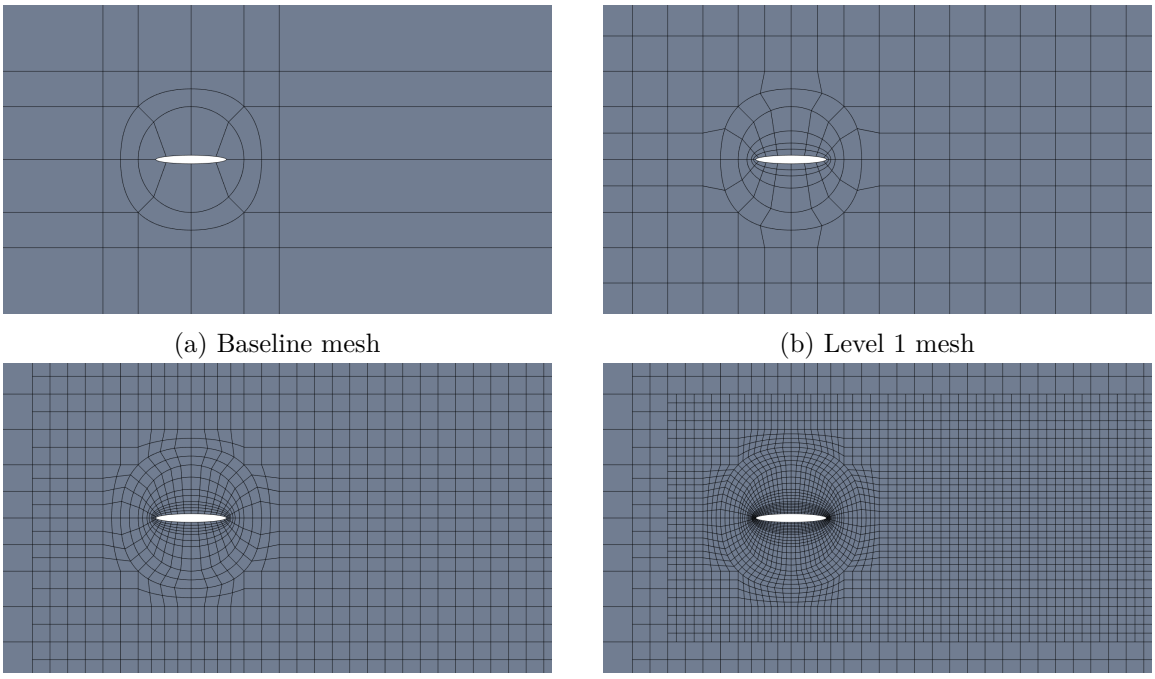
$$\alpha(t) = -A \sin(2\pi f t), \tag{30}$$

with reduced frequency $k = \pi f c / U_\infty = 0.5\pi$ and amplitude $A = 30°$. The ellipse is pitching about its center. We consider a laminar subsonic flow condition with freestream Mach number equal to 0.2 and a Reynolds number of 500. The flow configuration is characterized by a periodic vortical wake generated by the dynamic separation of the boundary layer around the ellipse. Therefore, the aim of the proposed test case is to study the behaviour of the developed sliding mesh technique for complex unsteady flows.

5.1 Mesh convergence

Firstly, a mesh convergence study is conducted. We consider a large computational domain, delimited by the rectangle $[-30c, 60c] \times [-30c, 30c]$, with c being the major axis of the ellipse. The far-field boundary conditions are weakly imposed using Riemann invariants. On the ellipse surface, a weakly prescribed no-slip adiabatic wall boundary condition is adopted. Since the isogeometric approach allows an exact representation of the boundary, an extremely coarse baseline mesh of degree two can be generated. As represented in Fig. 7, the meshes adopted for the actual computations are obtained applying a hierarchical refinement to the baseline configuration. The convergence analysis is carried out using 3 mesh refinement levels:

- level 1: 1464 total elements, 16 of which on the ellipse boundary,
- level 2: 2406 total elements, 32 of which on the ellipse boundary,
- level 3: 5046 total elements, 64 of which on the ellipse boundary.



(a) Baseline mesh (b) Level 1 mesh

(c) Level 2 mesh (d) Level 3 mesh

Figure 7: Different refinement levels for the ellipse simulation

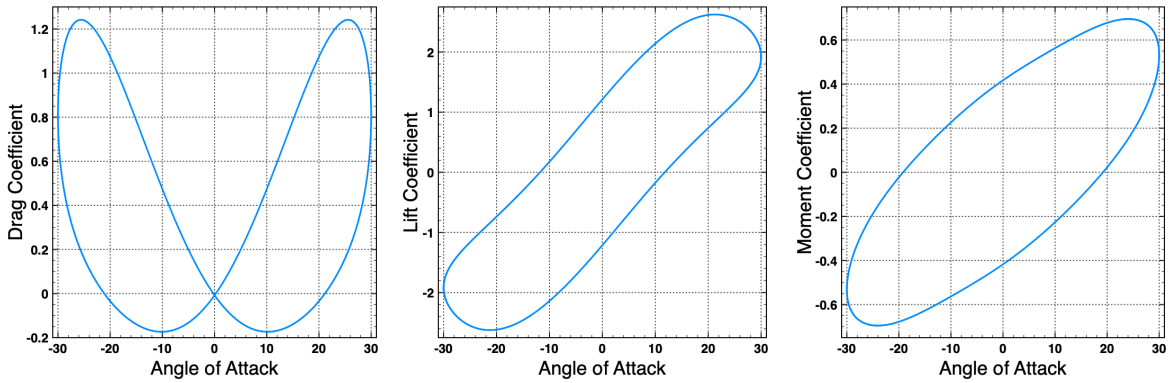
The study is performed using rational basis defined using quadratic, cubic and quartic polynomials. The radius of the sliding interface is equal to $0.75c$, meaning that the minimum distance between the ellipse and the interface is of a quarter of the major axis length c.

The initial condition of the simulations corresponds to the uniform freestream flow state. After a numerical transient, the solution converges to a stable limit cycle in which the flow fluctuations are synchronized with the pitching movement of the ellipse. We present in Fig. 8 the evolution of the aerodynamic coefficients over one pitching cycle. We compute, for each cycle, the average drag coefficient $\bar{C}_d$, the peak lift coefficient $\widehat{C}_l$ and the energy transfer coefficient E. The latter being defined as the area delimited by the $C_m - \alpha$ curve:

$$E = \frac{1}{\frac{1}{2}\rho_\infty u_\infty^2 c^2} \oint M_z \, \mathrm{d}\alpha = \oint C_m \, \mathrm{d}\alpha, \tag{31}$$

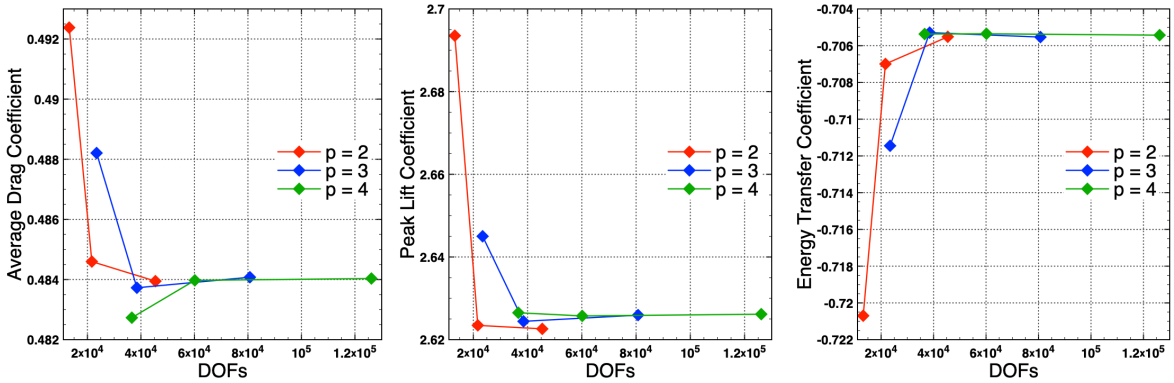
where M_z is the moment of the aerodynamic forces with respect to the pitching axis. The simulations are stopped when the considered coefficients are sufficiently converged in time.

The results of the mesh convergence analysis are reported in Fig. 9. All the tested configurations of degrees and refinement levels are able to reproduce the synchronization between the flow and the pitching movement. As expected, a significantly faster convergence is observed when the degree of the basis is increased. At the same time, the coarsest mesh is not capable of predicting with accuracy all the coefficients, even with quartic polynomials. Indeed, the level 2 grid with a cubic representation provides a better estimation of $\bar{C}_d$, with a similar number of degrees of freedom. Therefore, we conclude that the cubic level 2 mesh represents a satisfactory compromise between accuracy and computational cost. In general, the best trade-off is found by combining h and p refinements. We can also remark that



(a) Drag coefficient (b) Lift coefficient (c) Moment coefficient

Figure 8: Pitching ellipse, limit cycle solution, cubic level 2 mesh



(a) Average drag (b) Peak lift (c) Energy transfer

Figure 9: Pitching ellipse, convergence of the aerodynamic coefficients

the value of E is negative, which means that, from a physical point of view, the energy is transferred from the ellipse to the fluid.

5.2 Comparison with deforming mesh

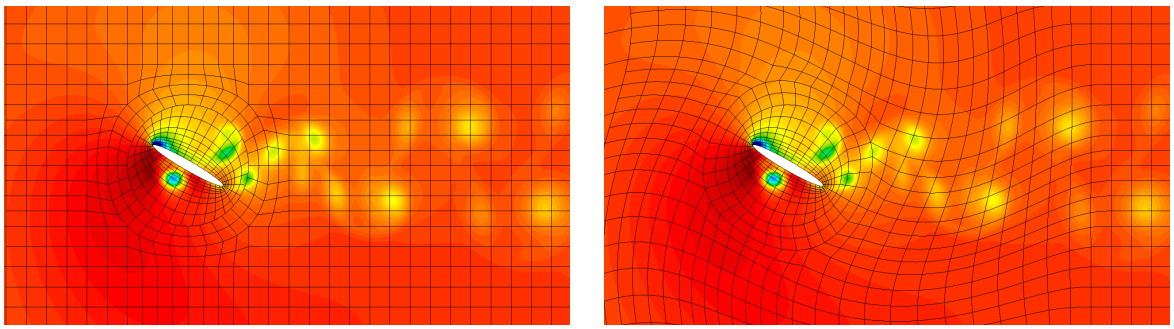
In the second part of the ellipse case study, we compare the sliding mesh approach with the deformation technique validated in (12). Since the pitching amplitude is limited to 30°, it is still possible to continuously deform the computational domain without incurring into an excessive distortion of the mesh, which could potentially decrease the accuracy of the solution. In order to compare the two techniques, we repeat the convergence study with the grids of Fig. 7 using the ALE methodology. To this end, a mesh velocity field $\mathbf{V}_g$ needs to be defined. The computational domain is thus divided in three regions: an inner one that oscillates rigidly with the ellipse, an outer fixed zone and a transition region. The velocity of the control point net is computed with the following law:

$$\begin{cases} u_g(\mathbf{x}, t) = -\tilde{\omega}\, y \\ v_g(\mathbf{x}, t) = \tilde{\omega}\, x \end{cases} \tag{32}$$

with $\tilde{\omega} = \dot{\alpha}\,\sigma(\mathbf{x}, t)$, where σ is a piecewise-defined blending function:

$$\sigma(\mathbf{x}, t) = \begin{cases} 1, & \text{if } R \leq R_{int} \\ \frac{1}{2}\left[1 + \cos\left(\pi \frac{R - R_{int}}{R_{ext} - R_{int}}\right)\right], & \text{if } R_{int} < R < R_{ext} \\ 0, & \text{if } R \geq R_{ext} \end{cases} \tag{33}$$

with $R = \sqrt{x^2 + y^2}$. The distance R_{int} is equal to the radius of the sliding interface and $R_{ext} = 4c$. Intuitively, the sliding mesh movement can be interpreted as the limit case where $R_{ext} = R_{int}$. We illustrate in Fig. 10 the differences between the deforming and the sliding meshes. Besides, we can observe that, despite the significant differences between the two grids, the density fields obtained with the two approaches are visually identical.



(a) Sliding mesh (b) Deforming mesh

Figure 10: Comparison of sliding and deforming meshes, density field, quartic level 2 mesh

In order to better compare the two approaches, $\bar{C}_d$, $\widehat{C}_l$ and E are computed for the deforming mesh as well. The numerical values of the three aerodynamic coefficients are reported in Tables 1, 2 and 3 respectively. We can observe that the same converged values are obtained with both the techniques. Therefore, we conclude that the proposed sliding mesh approach is correctly capable of simulating complex flows. Although some small discrepancies are observed between the values obtained using the level 1 meshes, there is no evidence to infer that one methodology is more accurate than the other. However, the sliding approach is less expensive from a computational point of view, because the mass matrix is constant in time, and is less limited in terms of movement amplitude.

	Sliding			Deformation		
Degree	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
2	0.4924	0.4846	0.4839	0.4916	0.4833	0.4838
3	0.4882	0.4837	0.4841	0.4835	0.4836	0.4841
4	0.4827	0.4840	0.4840	0.4820	0.4840	0.4840

Table 1: Average drag coefficient for sliding and deforming meshes

	Sliding			Deformation		
Degree	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
2	2.694	2.623	2.623	2.691	2.619	2.622
3	2.645	2.624	2.626	2.630	2.624	2.626
4	2.627	2.626	2.626	2.624	2.626	2.626

Table 2: Peak lift coefficient for sliding and deforming meshes

	Sliding			Deformation		
Degree	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
2	-0.7207	-0.7070	-0.7055	-0.7204	-0.7059	-0.7054
3	-0.7114	-0.7053	-0.7055	-0.7065	-0.7052	-0.7055
4	-0.7054	-0.7054	-0.7054	-0.7047	-0.7054	-0.7054

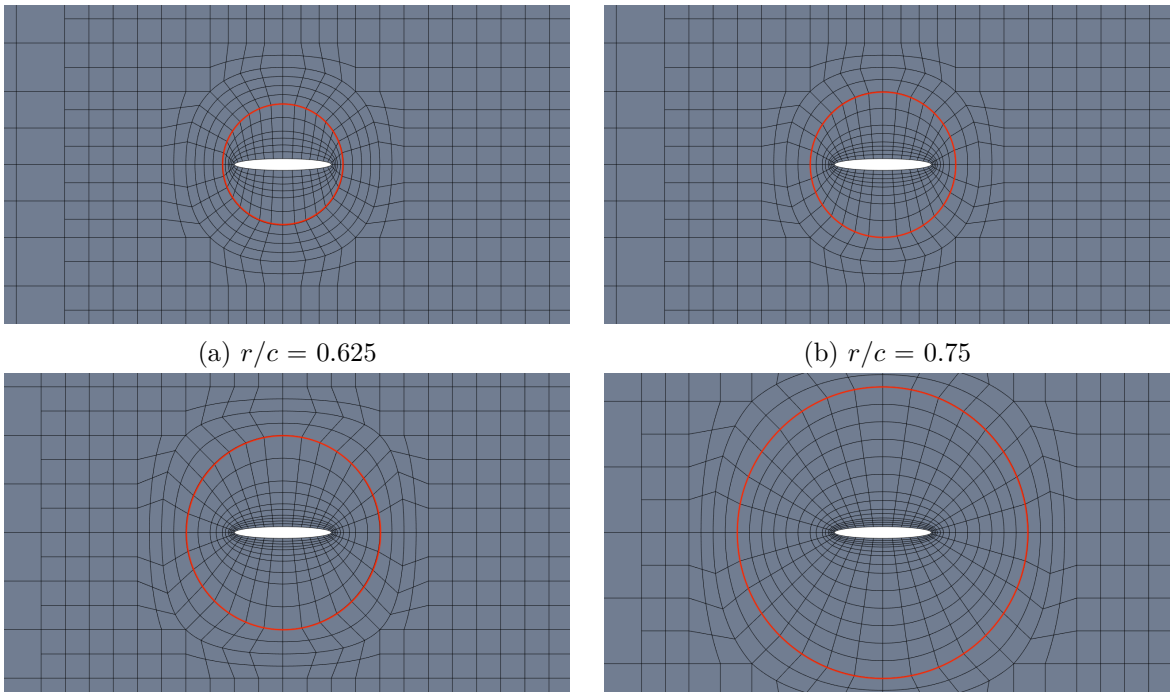
Table 3: Energy transfer coefficient for sliding and deforming meshes

5.3 Influence of the sliding interface radius

We then employ the ellipse case study to investigate the effect of the placement of the sliding interface on the numerical results. Low-order sliding mesh approaches can introduce numerical artifacts in the approximate solution, due to a loss of conservativity and poor interpolation. Ideally, the sliding interface should not be in close proximity of the boundary in order to limit the effects of spurious phenomena on the aerodynamic coefficients. On the other hand, a smaller rotating region allows to reduce the computational costs, because the smaller tangential velocity of the mesh allows to increase the time step. Therefore, the generation of meshes with sliding interfaces is the result of a trade-off and the process heavily relies on user experience. In this context, the use of a geometrically exact movement coupled with a high-order interpolation can significantly improve the accuracy of sliding grids and simplify the mesh generation task.

The previous convergence analysis was carried out considering a sliding interface radius equal to $0.75c$. By slightly modifying the baseline grid 7a, we were able to obtain three additional meshes, with different placements of the interface: one with a smaller radius of $0.625c$, and two with a larger rotating region, with r equal to c and $1.5c$, as illustrated in Fig. 11. In order to provide a fair comparison, the refinement process has been adapted to obtain a similar resolution in the wall area for all the considered grids. A cubic representation is employed and the overall mesh resolutions are comparable to the level 2 refinement adopted in the convergence study.

The values of $\bar{C}_d$, $\widehat{C}_l$ and E obtained for the different positions of the sliding interface are reported in Table 4. We do not observe a significant variation of the aerodynamic coefficients with respect to the radius of the interface. One can notice that the most sensitive quantity is the average drag coefficient. However, the relative variation between the maximum and the minimum value is less than 0.13%, which is completely negligible in a practical application. We can therefore state that, thanks to the accuracy of the developed sliding mesh technique, the results are nearly independent with respect to the position of the sliding interface. This simplifies the mesh generation problem, as the subdivision of the computational domain will be primarily dictated by the geometrical features of the boundary.



(a) $r/c = 0.625$ (b) $r/c = 0.75$

(c) $r/c = 1.00$ (d) $r/c = 1.50$

Figure 11: Meshes with different sliding interface placements

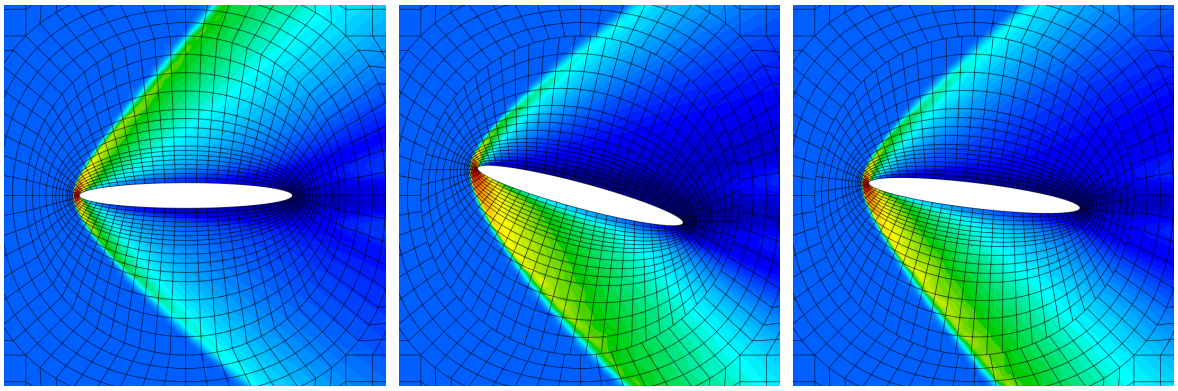
r/c	0.625	0.75	1.00	1.50
$\bar{C}_d$	0.4834	0.4837	0.4839	0.4840
$\widehat{C}_l$	2.6246	2.6244	2.6248	2.6247
E	-0.7052	-0.7053	-0.7054	-0.7054

Table 4: Variation of the aerodynamic coefficients with respect to r/c

5.4 Supersonic flow

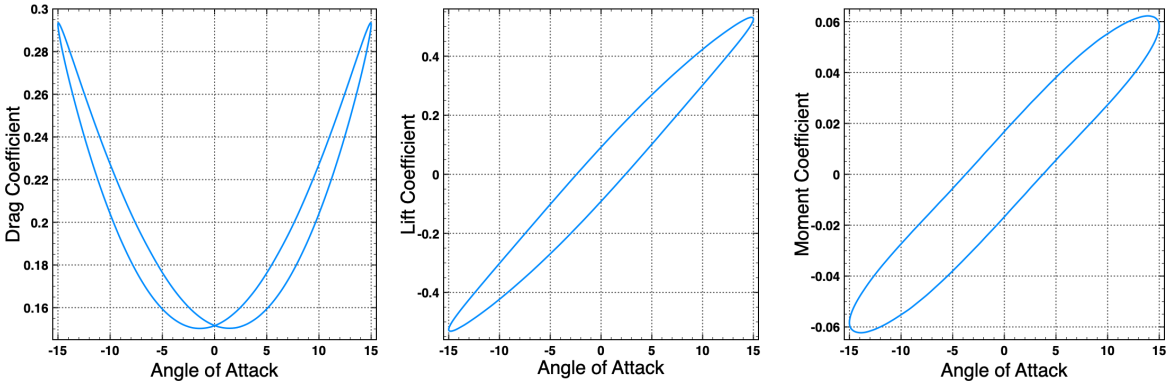
We conclude the ellipse case study with a simulation in the supersonic regime. The amplitude of the pitching motion is reduced to $A = 15°$ and a freestream Mach number equal to 2 is considered. The computation is performed using quadratic rational basis functions and the level 3 mesh, illustrated in Fig. 7d. The simulation of supersonic flows requires robust numerical algorithms, because of the presence of strong shocks. Furthermore, non-conservative schemes can lead to incorrect estimations of the shock velocity.

The considered flow is characterised by a bow shock in front of the ellipse. As the angle of attack α oscillates, the shape and intensity of the bow shock evolve. The density fields for three different values of α are illustrated in Fig. 12. As we can observe, the discontinuity is well resolved and the sliding interface does not interfere with the movement of the shock, since there is no loss of conservativity. As in the subsonic case, the solution converges to a stable limit cycle after the initial transient. In Fig. 13 we report the limit cycle curves for the aerodynamic coefficients C_l, C_d and C_m. We also estimate $\bar{C}_d$, $\widehat{C}_l$ and E for the supersonic flow condition. The computed values are: $\bar{C}_d = 0.223$, $\widehat{C}_l = 0.532$, and $E = -0.0146$.



(a) $\alpha = 0°$, ascending phase (b) $\alpha = 15°$ (c) $\alpha = 7.5°$, descending phase

Figure 12: Supersonic pitching ellipse flow, density fields



(a) Drag coefficient (b) Lift coefficient (c) Moment coefficient

Figure 13: Supersonic pitching ellipse flow, limit cycle solution

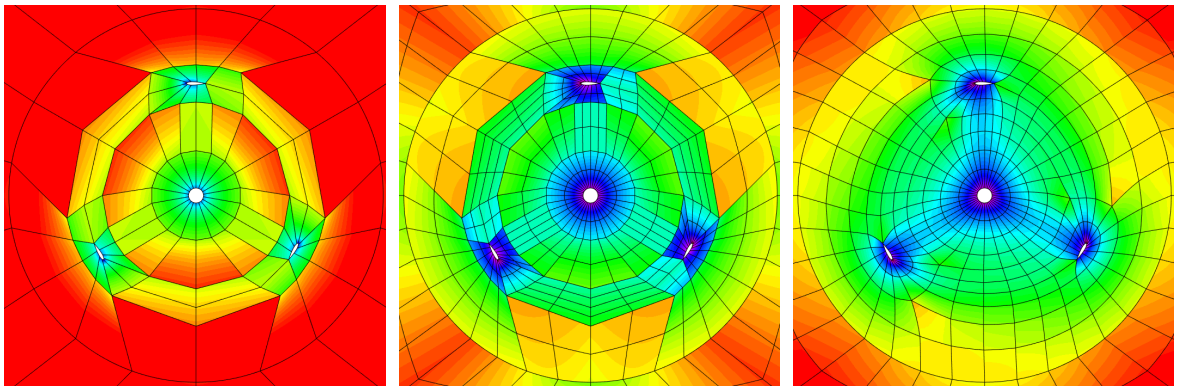
6 Application: vertical axis wind turbine

Sliding meshes are commonly employed to simulate rotating machinery such as helicopter rotors (17), axial compressors (18) and wind turbines (19). These kind of applications are characterized by complex geometries and topologies. For this reason the present section is dedicated to the investigation of a more challenging test case. In particular, we consider a Vertical Axis Wind Turbine (VAWT).

6.1 Construction of the computational domain

The first challenge of the proposed application is the generation of a suitable high-order mesh to perform the flow simulation. We start by defining the geometry of the turbine. Among the several possible VAWT designs, we consider an H-Darrieus rotor type (20) with three NACA 0018 blades. Being N the number of blades, the radius of the turbine R and the chord of the blades airfoils c are defined by means of the solidity $\sigma = Nc/R$, which is equal to 0.5 for our application. A cylindrical mast of radius $r = 0.2c$ completes the geometry of the rotor. The outer boundaries are sufficiently far from the turbine in order to allow the development of the wake and to avoid unwanted acoustic perturbations due to spurious reflections. As for

the ellipse case study, the far-field boundary conditions are based on Riemann invariants and the rotor surfaces are considered to be adiabatic no-slip walls.



(a) Baseline mesh (b) First refinement step (c) Smoothed mesh

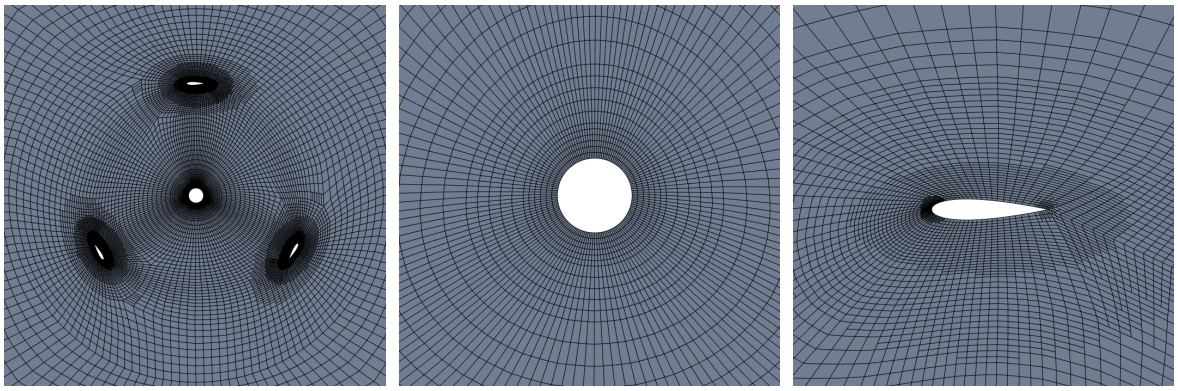
Figure 14: Initial VAWT mesh, refinement and smoothing. The colorscale represents the Jacobian of the Isogeometric map.

A rational quadratic representation is adopted for the rotor geometry and the sliding interface. The mast is therefore exactly described, whereas the NACA airfoil is approximated via least square fitting. In order to generate the mesh, we exploit the periodic pattern of the geometry. We first create a very coarse block-structured grid, presented in Fig. 14a. The baseline grid must respect the constraint of having a uniform subdivision of the sliding interface. Moreover, due the scale differences between the mast radius, the blade chord and turbine diameter, the initial patches are visibly irregular. We illustrate in Fig. 14b the mesh obtained after one isotropic refinement step and the Jacobian of the isogeometric map. It can be observed that strong discontinuities in the value of the Jacobian are present between the mesh blocks.

In order to improve the mesh regularity, we adopt a simple elliptic relaxation technique. The developed approach is an adaptation to high-order Bézier grids of the barycentric smoothing algorithm described by Falsafioon et al. (21). In each step of the relaxation procedure, the updated position of the j-th control point is equal to the barycentre of the neighbouring control points:

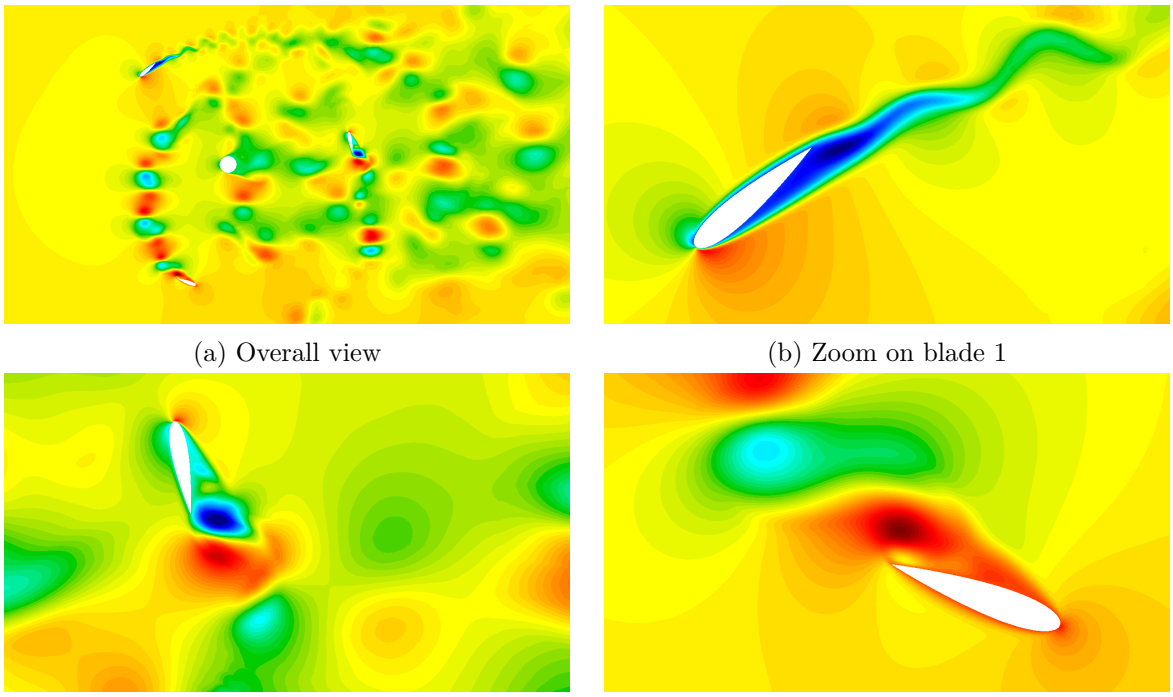
$$\mathbf{x}_j^{k+1} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i^k, \tag{34}$$

with n being the number of neighbours of $\mathbf{x}_j$. The sequence defined by eq. (34) converges towards the solution of the Laplace equation for the coordinate fields. The mesh illustrated in Fig. 14c is obtained applying the barycentric smoothing algorithm for 50 iterations starting from the grid of Fig. 14b. We notice that the regularity is significantly improved and that the gradient of the Jacobian is globally reduced. The mesh used for the actual computations is obtained by locally refining the smoothed grid in the rotor region, in particular around the mast and the blades, as presented in Fig. 15. The resulting grid consists of 18545 elements in total.



(a) Final rotor mesh (b) Detail of the mast region (c) Detail of the blade region

Figure 15: Final VAWT mesh



(a) Overall view (b) Zoom on blade 1

(c) Zoom on blade 2 (d) Zoom on blade 3

Figure 16: VAWT simulation, velocity field

6.2 Flow simulation

The typical working conditions of VAWTs are characterized by turbulent flows. Indeed, the blades need to have a sufficiently high lift-to-drag ratio to generate power, meaning that the flow regime of the airfoil has to be at least transitional. Simulating such a flow configuration requires turbulence modelling, high mesh resolution and implicit time integration, which is out of the scope of the current work. For these reasons, the present simulation is only carried out in the laminar regime, instead of considering fully realistic flow conditions.

The freestream Mach number is equal to 0.1 to avoid unwanted compressibility effects.

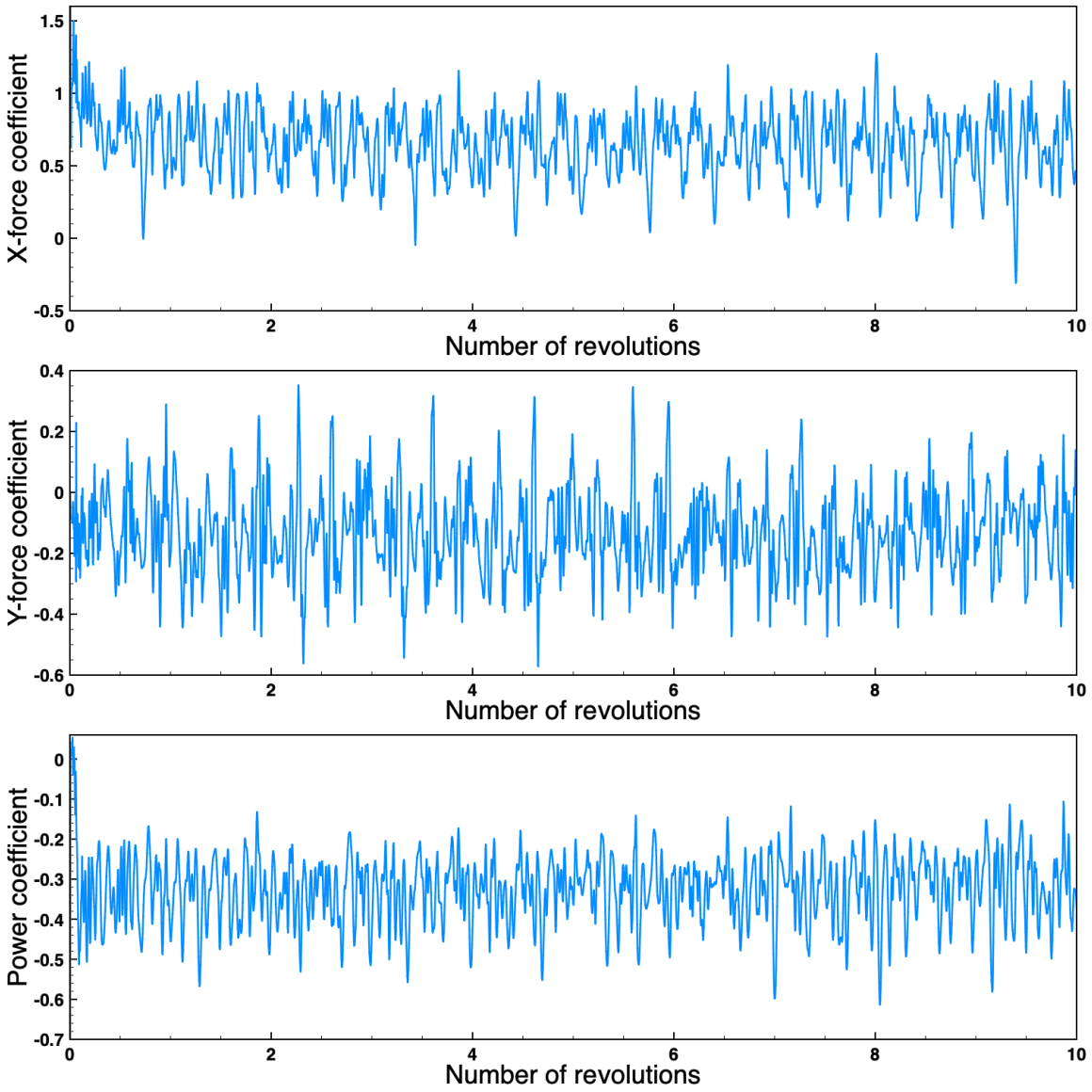


Figure 17: VAWT simulation, evolution of the aerodynamic coefficients

The rotational speed ω of the turbine is defined using the tip-speed ratio $\lambda = \omega R/u_\infty$. For the present simulation $\lambda = 2$, meaning that the tangential velocity of the blades is twice the freestream speed. The Reynolds number Re_{rotor} computed using u_∞ and the turbine radius is equal to 3000. In order to estimate the flow regime around the airfoils, we can consider the maximum possible relative velocity of the blades with respect to the freestream condition and compute:

$$Re_{blade} = \frac{c\,(u_\infty + \omega R)}{\nu} = \frac{\sigma}{N}\,(\lambda + 1)\,Re_{rotor} = 1500. \tag{35}$$

The flow around the blades is therefore laminar. In a realistic condition $Re_{blade} \approx 10^5$ at least. The result of the laminar regime is the separation of the blades boundary layer very close to the leading edge and the consequent formation of large vortical structures, as it can be observed in

Fig. 16. As the large structures are advected, interactions occur between the eddies and both the mast and the blades, resulting in a complex chaotic flow. A total of 10 revolutions are simulated. The evolution of the power coefficient C_p and of the aerodynamic force coefficients of the turbine are reported in Fig. 17. Due to the chaotic nature of the solution it is not possible to extract a periodic pattern for the C_p curve. Indeed, the interactions between the rotor and the vortical structures generates strong fluctuations in the power coefficient. Also, we can notice that the value of C_p is negative, meaning that the movement of the turbine is absorbing energy. This is a direct consequence of the poor aerodynamic performance of the blades in the laminar regime. Nevertheless, this study illustrates the capability of the proposed sliding mesh algorithm to deal with complex configurations.

7 Conclusion

In this paper we developed a high-order accurate sliding mesh technique for compressible flows. The proposed methodology is based on the isogeometric paradigm: a CAD-consistent DG formulation has been employed to discretize the Navier-Stokes equations. Then, starting from a general ALE framework, we detailed the numerical formulation and the treatment of the sliding interface. In particular, the properties of the NURBS representations were employed to implement an accurate and fully conservative sliding mesh algorithm.

The proposed approach has been firstly verified on a classic benchmark problem for inviscid flows and optimal convergence rates were observed for all the tested basis degrees. Furthermore, the comparison with an equivalent fixed grid showed that the error introduced by the sliding interface is negligible. As a second test case, we studied the viscous flow around a pitching ellipse, and the obtained results confirmed the precision and robustness of the sliding algorithm. We were able to compare the sliding mesh technique with a previously validated ALE formulation based on mesh deformation. We then showed that the outcome of the simulation is independent with respect to the position of the sliding interface. Lastly, we demonstrated the versatility of the developed methodology considering a supersonic flow condition.

As last case study, we simulated the flow around a 3-blade VAWT configuration, allowing us to evaluate the potential of the proposed approach on a more complex geometry. We especially focused on the construction of the computational grid, starting from a set of coarse Bézier patches. In order to obtain a better mesh quality, a smoothing algorithm has been employed, showing that ideas originating from unstructured grid generation can be beneficial in the context of isogeometric analysis as well.

Code Repository

The developed methodology is implemented in the **Igloo** software suite, which has been employed to perform all the presented computations. The source code and data are available, under the GNU General Public Licence v3, at the following repository: `https://gitlab.inria.fr/igloo/igloo/-/wikis/home`.

Acknowledgements

The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

References

[1] E. Ferrer, R. H. Willden, A high order Discontinuous Galerkin – Fourier incompressible 3D Navier–Stokes solver with rotating sliding meshes, Journal of Computational Physics 231 (2012) 7037–7056.

[2] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4135 – 4195.

[3] F. Zhang, Y. Xu, F. Chen, Discontinuous Galerkin methods for Isogeometric Analysis for elliptic equations on surfaces, Communications in Mathematics and Statistics 2 (2014) 431–461.

[4] Y. Bazilevs, T. J. R. Hughes, Nurbs-based isogeometric analysis for the computation of flows about rotating components, Computational Mechanics 43 (2008) 143–150.

[5] C. Mavripilis, Nonconforming discretizations and a posteriori error estimators for adaptive spectral element techniques, Ph.D. thesis, Massachusetts Institute of Technology, 1989.

[6] D. A. Kopriva, A conservative staggered-grid Chebyshev multidomain method for compressible flows. ii. A semi-structured method, Journal of Computational Physics 128 (1996) 475–488.

[7] B. Zhang, C. Liang, A simple, efficient, and high-order accurate curved sliding-mesh interface approach to spectral difference method on coupled rotating and stationary domains, Journal of Computational Physics 295 (2015) 147–160.

[8] J. Dürrwächter, M. Kurz, P. Kopper, D. Kempf, C.-D. Munz, A. Beck, An efficient sliding mesh interface method for high-order discontinuous Galerkin schemes, Computers & Fluids 217 (2021) 104825.

[9] E. Laughton, G. Tabor, D. Moxey, A comparison of interpolation techniques for non-conformal high-order discontinuous galerkin methods, Computer Methods in Applied Mechanics and Engineering 381 (2021) 113820.

[10] B. Cockburn, C.-W. Shu, The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems, SIAM Journal on Numerical Analysis 35 (1998) 2440–2463.

[11] L. Piegl, W. Tiller, The NURBS Book, second ed., Springer-Verlag, New York, NY, USA, 1996.

[12] S. Pezzano, R. Duvigneau, A NURBS-based discontinuous Galerkin method for conservation laws with high-order moving meshes, Journal of Computational Physics 434 (2021) 110093.

[13] P.-O. Persson, J. Peraire, Sub-Cell Shock Capturing for Discontinuous Galerkin Methods, 2006.

[14] R. Duvigneau, Isogeometric analysis for compressible flows using a Discontinuous Galerkin method, Computer Methods in Applied Mechanics and Engineering 333 (2018) 443 – 461.

[15] R. Duvigneau, CAD-consistent adaptive refinement using a NURBS-based discontinuous Galerkin method, Int. J. for Numerical Methods in Fluids (2020).

[16] J. S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, 1st ed., Springer, 2007.

[17] R. Steijl, G. Barakos, Sliding mesh algorithm for CFD analysis of helicopter rotor–fuselage aerodynamics, International Journal for Numerical Methods in Fluids 58 (2008) 527–549.

[18] N. Gourdain, Prediction of the unsteady turbulent flow in an axial compressor stage. Part 1: Comparison of unsteady RANS and LES with experiments, Computers & Fluids 106 (2015) 119–129.

[19] R. Patil, L. Daróczy, G. Janiga, D. Thévenin, Large eddy simulation of an H-Darrieus rotor, Energy 160 (2018) 388–398.

[20] L. Daróczy, G. Janiga, K. Petrasch, M. Webner, D. Thévenin, Comparative analysis of turbulence models for the aerodynamic simulation of H-Darrieus rotors, Energy 90 (2015) 680–690.

[21] M. Falsafioon, S. Arabi, R. Camarero, F. Guibault, Comparison of Two Mesh Smoothing Techniques for Unstructured Grids, 2014.