CAD-Consistent Aerodynamic Design via the Isogeometric Paradigm

Stefano Pezzano, Régis Duvigneau, Mickael Binois

Association Aéronautique et Astronautique de France

56th 3AF International Conference on Applied Aerodynamics
28 — 30 March 2022, Toulouse – France

FP81-2022-duvigneau

CAD-CONSISTENT AERODYNAMIC DESIGN VIA THE ISOGEOMETRIC PARADIGM

S. Pezzano[1,2], R. Duvigneau[1] and M. Binois[1]

[1] *Université Côte d'Azur, Inria, CNRS, LJAD, 2004 route des lucioles 06902 Sophia-Antipolis, France*

[2] *CERFACS, 42 Avenue Gaspard Coriolis 31057 Toulouse, France*

ABSTRACT

This work aims at proposing an aerodynamic design optimization methodology entirely based on Computer-Aided Design (CAD) representations, yielding a fully integrated geometry-simulation-optimization framework. Specifically, the geometry to optimize is defined thanks to CAD standards, like Non-Uniform Rational B-Splines (NURBS); the resolution of compressible Euler / Navier-Stokes equation is achieved using a Discontinuous Galerkin (DG) method based on rational Bézier elements with adaptive mesh refinement, which is geometrically exact with respect to CAD boundaries; The global optimization of CAD control points is carried out using a Bayesian Optimization method. The whole design chain is therefore consistent with respect to the CAD geometry. The proposed approach is demonstrated for the optimization of an airfoil in transonic regime.

1. INTRODUCTION

Modeling complex phenomena by solving systems of Partial Differential Equations (PDEs) has become a classical approach in several scientific disciplines, such as fluid and structural mechanics, electromagnetics, etc. This results from both the increase of the computational facilities and the maturity of numerical methods. As a consequence, it is now possible to simulate industrial products before manufacturing, at early design phase, to analyze in detail the physical behaviors. Simultaneously, the need for optimization algorithms has grown, in order to maximize the product performance, beyond engineers intuition, and minimize the environmental costs.

However, the use of fully automated design optimization loops is still difficult, in particular due to the complexity of the simulation pipeline. The simulation of realistic systems indeed requires mastering a large number of techniques and specific software: geometric modeling using Computer-Aided Design (CAD) tools, automated grid generation, PDE solvers for the different physical problems and couplings, post-processing and visualization, design optimization. Not only a deep knowledge is necessary to make them run, but also the coordination of these components is a source of difficulties, due to a lack of interoperability. The geometry plays a central role in this complexity overhead, because it is described by different representations, depending on the disciplinary context: CAD tools use high-order representations like Non-Uniform Rational B-Splines (NURBS), whereas most PDE solvers are using grids, which are usually based on piecewise linear representations (triangles, tetrahedra, etc.). The coexistence of the two geometrical representations in the design loop yields a loss of accuracy and an overhead complexity due to numerous geometrical transformations. This situation is even more tedious in a multidisciplinary context, where different solvers are coupled, and in design optimization because all the steps have to be automated.

A first breakthrough has been proposed in 2005 by T. Hughes and coauthors, namely the IsoGeometric Analysis (IGA) method [9], which consists in solving PDEs directly using parametric surfaces or volumes from CAD as computational domain, in a Finite-Element formulation. Hence, the proposed approach bypasses the approximation of the geometry by grids and unifies the representations used in CAD and analysis, yielding a significant simplification of the design pipeline. However, several dificulties and limitations have arisen: i) Despite the developments achieved by the CAD communauty for 15 years, the capability of CAD software to generate ready-to-use computational domains is still very limited. ii) Multidimensional NURBS representations commonly

used in CAD rely on tensorial products which make local refinement tedious, due to uncontrolled propagation. This has motivated the development of more sophisticated representations, like T-Splines [1] or THB Splines, which are however far more complex to handle. iii) The use of IGA for hyperbolic systems suffers from the need for stabilization, which is not straightforward in this context. As a consequence, applications in compressible fluid mechanics for instance are very rare.

To overcome the limitations of the IGA method, an extension of the isogeometric analysis paradigm to Discontinuous Galerkin (DG) methods has been proposed [5], with application to non-linear hyperbolic systems. This requires a change of basis to generate discontinuities in the solution at element interfaces without altering the CAD geometry (for this, a CAD technique named Bézier extraction is used). The resulting approach demonstrates several interesting properties to improve the original formulation: conservativity, straightforward local refinement [6], CAD-consistency, easy extension to moving or deforming geometries [13].

This work explains how this methodology can be employed to construct a design optimization loop, in which the geometry is entirely described in terms of CAD representations, yielding gains in both accuracy and ease to deploy.

2. GEOMETRY PARAMETERIZATION

Design procedures of most modern engineering systems are based on CAD kernels, which allow a straightforward communication with the manufacturing process. Several representations can be adopted, but the most commonly used approach relies on the description of the object using parameterized boundaries (B-Rep approach). We will therefore adopt this point of view in the current work. However, for the sake of simplicity, we restrict here on two-dimensional problems in order to focus on the methodology. The extension to three-dimensional problems may not be straightforward, depending on the complexity of the selected case, and is out of the scope of this work. Nevertheless, we pay attention to propose methods that are not essentially restricted to two-dimensional cases.

2.1 NURBS Basis

We make therefore the assumption that the geometry of the system of interest is described by a set of parameterized curves. Non-Uniform Rational B-Splines (NURBS) curves are now considered as standard in CAD [7, 4], since they allow to represent exactly a broad class of curves commonly encountered in engineering, like conic sections. Moreover, they permit a very intuitive definition and modification of geometrical objects through the handling of *control points*. Hence NURBS curves are chosen to describe the geometry. They are defined using a so-called *knot* vector $\Xi = (\xi_1, \dots, \xi_l) \in \mathbb{R}^l$, which consists of nondecreasing l real numbers. This knot vector defines a discretization of the *parametric domain* $[\xi_1, \xi_l]$. NURBS basis functions are derived from B-Spline functions $(N_i^p)_{i=1,\cdots,n}$, which are polynomial and defined recursively as [4]:

$$N_i^0(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$N_i^p(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_i^{p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1}^{p-1}(\xi). \tag{2}$$

Note that the quotient 0/0 is assumed to be zero. The degree of the functions p, the number of knots l and functions n are related by the relation $l = n + p + 1$ [4]. Open knot vectors, i.e. knot vectors with first and last knots of multiplicity $p+1$, are usually used for representations of degree p to impose interpolation and tangency conditions at both extremities [4]. Therefore $\xi_1 = \ldots = \xi_{p+1}$ and $\xi_{n+1} = \ldots = \xi_{n+p+1}$. A set of B-Spline basis functions of degree two is illustrated in Fig. 1.

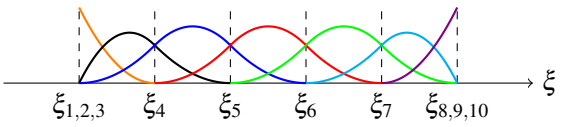

Figure 1: Seven quadratic B-Spline basis functions.

Then, the NURBS basis functions $(R_i^p)_{i=1,\cdots,n}$ are defined by associating a set of weights $(\omega_i)_{i=1,\cdots,n}$ to the B-Spline functions according to:

$$R_i^p(\xi) = \frac{w_i \, N_i^p(\xi)}{\sum_{j=1}^n w_j \, N_j^p(\xi)}. \tag{3}$$

Finally, a NURBS curve of coordinates $\mathbf{x}(\xi) = (x(\xi), y(\xi))$ is obtained by positioning a set of *control points* which are associated to the NURBS basis functions:

$$\mathbf{x}(\xi) = \sum_{i=1}^n R_i^p(\xi) \mathbf{x}_i, \tag{4}$$

where $\mathbf{X} = (\mathbf{x}_i)_{i=1,\cdots,n}$ are the coordinates of the control points in the *physical domain*. Note that the previous equation defines a mapping $\mathscr{F}$ from the parametric domain $\widehat{\Omega}$ to the physical one Ω, as illustrated in Fig. 2

2.2 Some properties

Projection A first important property concerns the relationship between B-Spline and NURBS curves: it can

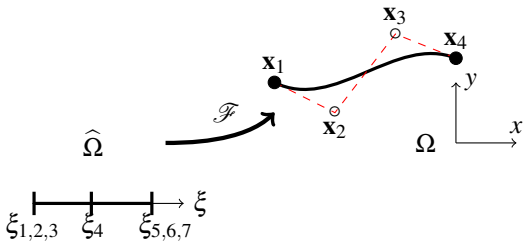

Figure 2: Example of quadratic NURBS curve with four control points.

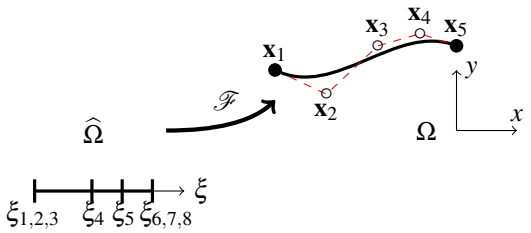

Figure 3: Example curve of Fig. 2 with a new inserted knot.

be shown [14] that any NURBS curve can be considered as the projection of a B-Spline curve defined in a space of higher dimension using its weights. For instance, a NURBS curve lying in the plane (x,y), characterized by a set of control points $\mathbf{X} = (\mathbf{x}_i)_{i=1,\cdots,n} = (x_i, y_i)_{i=1,\cdots,n}$ and weights $(\omega_i)_{i=1,\cdots,n}$, can be described as the projection of a B-Spline curve lying in a 3D space and defined by the control points $(x_i\omega_i, y_i\omega_i, \omega_i)_{i=1,\cdots,n}$. This representation of NURBS curves as B-Spline ones is useful for practical manipulations because one can apply transformations directly to B-Spline curves and project the result, instead of developing specific procedures specific to NURBS.

Knot insertion A second property of NURBS representations is the capability to insert a new knot, and thus a new basis function, without altering the geometrical object [14]. This procedure can be considered as a local h-refinement procedure [9] and is referred to as *knot insertion*.

Indeed, any NURBS curve with the associated knot vector $\Xi = (\xi_1, \ldots, \xi_l) \in \mathbb{R}^l$ can be identically represented using the knot vector $(\xi_1, \ldots, \xi_q, \bar{\xi}, \xi_{q+1} \ldots, \xi_l) \in \mathbb{R}^{l+1}$, that includes the additional knot $\bar{\xi}$ inserted between ξ_q and ξ_{q+1}. The new curve is then defined by $n+1$ control points. To compute their new locations, one can use the projection property described above and simply apply the procedure to the corresponding B-Spline curve defined in the space of higher dimension. With the additional knot, the B-Spline curve can be written as [14]:

$$\mathbf{x}(\xi) = \sum_{i=1}^{n} N_i^p(\xi)\mathbf{x}_i = \sum_{i=1}^{n+1} N_i^p(\xi)\bar{\mathbf{x}}_i, \tag{5}$$

with the new set of $n+1$ control points defined as $\bar{\mathbf{x}}_i = (1-\alpha_i)\mathbf{x}_{i-1} + \alpha_i\mathbf{x}_i$ with:

$$\alpha_i = \begin{cases} 1 & \text{if } i \leq q-p \\ \frac{\bar{\xi}-\xi_i}{\xi_{i+p}-\xi_i} & \text{if } q-p+1 \leq i \leq q \\ 0 & \text{if } i \geq q+1 \end{cases} \tag{6}$$

Fig. 3 illustrates the knot insertion procedure applied to the curve of the previous figure.

Bézier extraction It is important to underline now a particular case: if a new knot is inserted at an existing knot location, the regularity of the curve is decreased. More generally, the curve at the knot ξ_q has the regularity C^{p-r}, where r is the multiplicity of the knot q [14]. Therefore, if one inserts p knots at an existing knot location, the geometry of the curve is preserved but the curve is now divided in two independent parts. If this multiple-knot insertion is achieved for all inner knots, the curve is decomposed into a set of independent *Bézier curves*, each of them composed of $p+1$ control points [14], as depicted in Fig. 4.

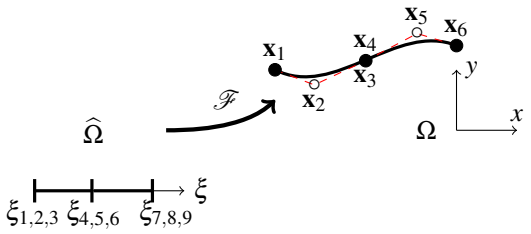

Figure 4: Example curve of Fig. 2 after Bézier extraction procedure.

2.3 Surfaces and volumes

All the previous concepts and properties can be extended to surfaces (respectively volumes) by using bivariate (respectively trivariate) tensor products. In particular, a NURBS surface of degree p is defined as:

$$\mathbf{x}(\xi,\eta) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} R_i^p(\xi)R_j^p(\eta)\mathbf{x}_{ij}, \tag{7}$$

where $(\mathbf{x}_{ij})_{i=1,\cdots,n}$ are the coordinates of the control point indexed ij in the lattice. Such a surface defined in the plane is shown in figure 5. This tensorial construction is especially important because it will be the baseline of the CAD-consistent computational domain.

Table 1: Airfoil case: NURBS Control point coordinates.

index	1	2	3	4	5	6	7	8	9	10
x	0.	0.	0.125	0.25	0.375	0.5	0.626	0.75	0.875	1.
y^+	0.	0.031	0.068	0.074	0.065	0.048	0.034	0.039	0.025	0.
y^-	0.	-0.027	-0.031	-0.044	-0.069	-0.061	-0.039	-0.013	-0.010	0.

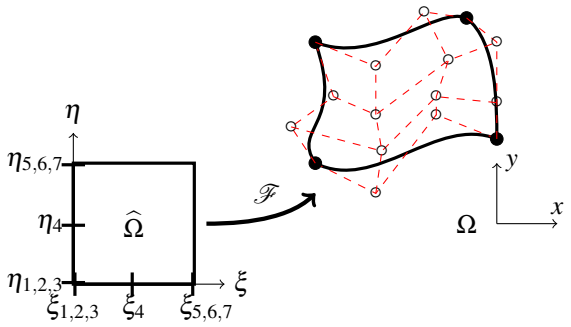

Figure 5: Example of quadratic NURBS surface with four control points.

3. COMPUTATIONAL DOMAIN

3.1 Principles

For each geometry explored during the design optimization phase, a corresponding computational domain must be constructed to allow the physical analysis using a dedicated PDE solver. According to the proposed paradigm, we aim at defining a mesh which is consistent with the CAD geometry, yielding a geometrically exact analysis. In this perspective, we have to satisfy the following constraints:

1. The mesh should cope with the solver requirements;
2. The mesh boundary should exactly match the NURBS curves that define the geometry;
3. The mesh should remain valid for large variations of the geometry;
4. The mesh generation process should be completely automated.

Let's consider first the question of the solver requirements. Indeed, our objective of geometrically exact analyses yields a strong dependency between the mesh generation task and the PDE resolution, contrary to more classical approaches. As deeply investigated in previous works [5], we do not follow the original isogeometric paradigm proposed by T. Hughes and coauthors [9], which consists in defining the mesh as a set of NURBS patches and apply a Finite-Element (FE) formulation, for two main reasons. Firstly, FE schemes are not so well suited to aerodynamic applications due to the necessity to adjust a set of parameters to stabilize the hyperbolic terms and avoid oscillations in the vicinity of shocks. Secondly, local mesh refinement requires the use of sophisticated and somehow cumbersome representations like T-Splines [1] to avoid a large propagation of the refinement, due to the tensorial nature of NURBS patches. To overcome these difficulties, a Discontinuous Galerkin (DG) formulation based on rational Bézier representations is preferred [5]. Indeed, the DG formulation is very well adapted to the resolution of hyperbolic systems of conservation laws and the use of a discontinuous representation makes refinement essentially local.

However, this choice makes the construction of the mesh a bit more complicated. Indeed, the proposed approach necessitates to generate a mesh composed of rational Bézier elements that coincide with the NURBS boundary. Hopefully, this can be achieved quite easily, thanks to the Bézier extraction property described in the previous section. This allows to split the boundary NURBS curves into a set of rational Bézier curves that will compose the boundary edges of the elements.

To make the procedure fully automated and allow large defomations of the boundary, we proceed in three steps for the construction of the grid: first, a small set of fixed rational Bézier elements are defined inside the computational domain, but far from its deforming boundary. Then, an extremely coarse grid is generated by connecting the boundary edges to surrounding fixed points, yielding a *primary mesh* composed of a few rational Bézier elements. The edges define the control points and weights at the border of the elements. The interior ones are initialized using a discrete Coons patch construction[7]. Note that the fixed part of this mesh is far enough from the deforming boundary to permit large geometry modifications. Even if this mesh is not suitable for analysis due to its coarseness, it is anyway consistent with the CAD definition of the geometry. Finally, some local refinement steps are achieved by using multiple knot insertions to obtain a *secondary mesh*, which will be considered as the initial grid for an adaptive analysis. The properties of the refinement process maintain both the boundary geometry and the viability of the elements (e.g. positive Jacobian).

3.2 Illustration

The proposed approach is illustrated for the specific case of an airfoil. This example is very simple on purpose, to put in light the methodological steps. The airfoil is parameterized by two NURBS curves of degree $p = 3$, one for the suction side and one for the pressure side, composed of $n = 10$ control points each. The knot vector of length $k = 14$ is defined with a uniform distribution for the two curves $\Xi = (0,0,0,0,1,2,3,4,5,6,7,7,7,7)$. Four repeated knots at start and end ensure that the curve interpolates the control points at the two extremities. The positions of the control points are provided in Tab. 1. Note that the alignment of control points at the leading edge ensures a C^1 continuity. For the sake of simplicity, unitary weights are employed. This defines completely the two curves illustrated in Fig. 6.

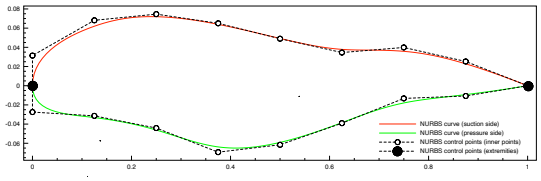

Figure 6: Airfoil geometry defined by two NURBS curves.

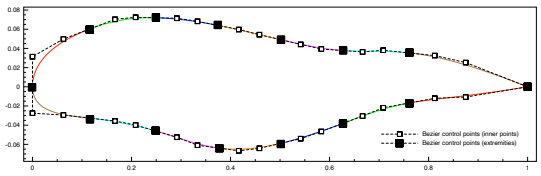

Figure 7: Airfoil geometry defined by a set of rational Bézier edges, after Bézier extraction procedure.

Then, we apply the Bézier extraction procedure which allows to split the curves into a set of rational Bézier edges without modifying the geometry, as illustrated in Fig. 7. The filled markers represent the extremities of the rational Bézier edges, where the control points are interpolated. The first step of the mesh construction consists in defining a set of rational elements surrounding the airfoil, independently from its shape, as depicted in Fig. 8. This task is achieved manually, given the small number of elements. Note that the tiling is chosen to coincide with the number of edges on the airfoil boundary. The gap between the airfoil and these fixed elements is obviously large enough to define very different airfoil shapes. During the second step, the extremities of the Bézier edges on the airfoil are connected to the fixed elements, filling the computational domain and yielding the so-called primary mesh, as shown in Fig. 9. Although extremely coarse, this mesh is CAD-consistent.

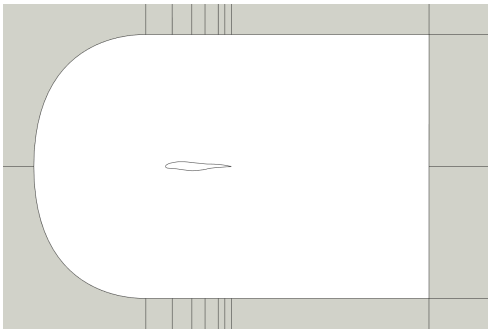

Figure 8: Set of fixed rational Bézier elements surrounding the airfoil.

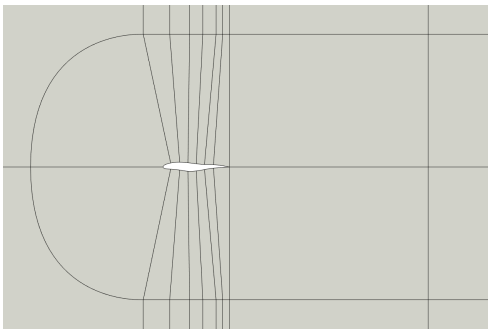

Figure 9: Primary mesh after connections.

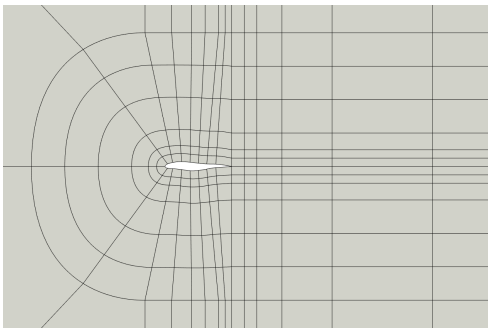

Figure 10: Secondary mesh after refinements.

The third step consists in refining the grid in the vicinity of the airfoil to obtain a mesh suitable for analysis. This can be done automatically by applying knot insertion algorithm. The resulting grid is denoted as secondary mesh shown in Fig. 10. Note that this mesh is still coarse because we intend to employ an Adaptive Mesh Refinement (AMR) strategy during the analysis and, therefore,

it is only an initial grid. Again, this task is achieved maintaining the geometry exact with respect to the baseline definition of the airfoil using two NURBS curves.

4. FLOW RESOLUTION

We present in this section the methods employed to solve flow equations using rational Bézier elements, yielding geometrically exact analyses. Most of these methods have already been described in previous works [5, 6, 13], therefore we focus here on their integration in the design optimization loop.

4.1 Isogeometric Discontinous Galerkin

As explained above, we adopt in this work a Discontinuous Galerkin formulation based on rational Bézier elements, which permits both to use CAD-consistant grids and offers a suitable framework for the resolution of hyperbolic systems. Due to the applications targeted, we consider as state equations the compressible Euler / Navier-Stokes equations. The extensions to an Arbitrary Lagrangian-Eulerian formulation can be found in [13]. In this context, the physical flux can be expressed as:

$$\mathbf{F} = \mathbf{F}_c(\mathbf{W}) - \mathbf{F}_v(\mathbf{W}, \nabla\mathbf{W}), \tag{8}$$

where $\mathbf{F}_c$ is the convective flux, $\mathbf{F}_v$ the viscous flux, $\mathbf{W}$ the conservative variables:

$$\mathbf{W} = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho e \end{pmatrix}, \mathbf{F}_{c,i} = \begin{pmatrix} \rho u_i \\ \rho u_1 u_i + p\delta_{1i} \\ \rho u_2 u_i + p\delta_{2i} \\ \rho u_i \left(e + \frac{p}{\rho}\right) \end{pmatrix}, \mathbf{F}_{v,i} = \begin{pmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ u_k \tau_{ki} - q_i \end{pmatrix}, \tag{9}$$

where ρ denotes the density, p the pressure, e the total energy and u_i the velocity vector. τ_{ij} is the viscous stress tensor and q_i is the thermal conduction flux, defined as:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}, \tag{10}$$

$$q_i = -\gamma\frac{\mu}{Pr}\frac{\partial e}{\partial x_i}, \tag{11}$$

where $\gamma = 1.4$, $Pr = 0.72$ and μ is determined by the Reynolds number. When Euler equations are considered, the viscous flux $\mathbf{F}_v$ vanishes. The second order derivatives are discretized with the Local Discontinuous Galerkin (LDG) approach [3]. We thus write the state equations as a system of first order equations:

$$\begin{cases} \dfrac{\partial \mathbf{W}}{\partial t} + \nabla\cdot\mathbf{F}_c(\mathbf{W}) - \nabla\cdot\mathbf{F}_v(\mathbf{W},\mathbf{G}) = 0, \\ \mathbf{G} - \nabla\mathbf{W} = 0. \end{cases} \tag{12}$$

where $\mathbf{G}$ is the gradient of the conservative variables. According to the DG formulation, these equations are multiplied by a rational Bézier trial function R_k (for the sake of clarity we drop degree p) and integrated by part on each element Ω_j. Using the map defined by the rational Bézier functions, the integrals are transposed from the physical space to the parametric domain $\widehat{\Omega}$, yielding the following *isogeometric DG formulation*:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{w}_i \int_{\widehat{\Omega}} R_k R_i \,|\mathrm{J}_{\Omega_j}|\,\mathrm{d}\widehat{\Omega}\right) &= \int_{\widehat{\Omega}} \nabla R_k \cdot \left(\mathbf{F}_c - \mathbf{F}_v\right)|\mathrm{J}_{\Omega_j}|\,\mathrm{d}\widehat{\Omega} \\ &\quad - \oint_{\partial\widehat{\Omega}} R_k\left(\mathbf{F}_c^* - \mathbf{F}_v^*\right)|\mathrm{J}_{\Gamma_j}|\,\mathrm{d}\widehat{\Gamma} \end{aligned} \tag{13}$$

$$\begin{aligned} \mathbf{g}_i \int_{\widehat{\Omega}} R_k R_i \,|\mathrm{J}_{\Omega_j}|\,\mathrm{d}\widehat{\Omega} &= \int_{\widehat{\Omega}} \nabla R_k \cdot \mathbf{w}_h \,|\mathrm{J}_{\Omega_j}|\,\mathrm{d}\widehat{\Omega} \\ &\quad - \oint_{\partial\widehat{\Omega}} R_k \mathbf{W}^* \,|\mathrm{J}_{\Gamma_j}|\,\mathrm{d}\widehat{\Gamma} \end{aligned} \tag{14}$$

This discretization is qualified as isogeometric because the same rational Bézier representation 7 is employed for the geometry $\mathbf{x}$, the local solution fields $\mathbf{w}_h$ and $\mathbf{g}_h$:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{w}_h \\ \mathbf{g}_h \end{pmatrix} = \sum_{i=1}^{(p+1)^2} R_i(\xi,\eta) \begin{pmatrix} \mathbf{x}_i \\ \mathbf{w}_i \\ \mathbf{g}_i \end{pmatrix}, \tag{15}$$

The convective flux function $\mathbf{F}^*$ is evaluated using the HLL Riemann solver, whereas the diffusive flux functions $(\mathbf{F}_v^*, \mathbf{W}^*)$ are computed using the LDG approach [3]. Moreover, integrals are evaluated by numerical quadrature based on Gauss-Legrendre rules and time integration is carried out using standard Runge-Kutta methods. The resulting method has been verified on a set of analytical problems, exhibiting optimal $p+1$ convergence rate [5, 6], and validated on classical benchmarks [13]. To complete this description, we mention the use of the subcell shock capturing method [12] adapted to the rational Bézier representation [5, 6]. As summary, we employ very classical techniques to solve the flow equations, except that they are based on rational Bézier representations intead of standard nodal Lagrange functions, to cope with CAD-consistent grids.

4.2 Adaptive mesh refinement

The use of Adaptive Mesh Refinement (AMR) techniques is a critical ingredient of the design procedure because it minimizes the computational time required for each analysis and makes the automated grid generation process reliable. Indeed, constructing *a priori* a mesh adapted to the flow features is far from being straightforward, especially when the process has to be fully automated. Therefore, it is far more efficient and robust to construct first a baseline grid, denoted as secondary mesh in section 3.1, and then use AMR techniques at runtime to obtain an appropriate resolution for each region of the computational domain. We underline that the proposed isogeometric

DG approach is a perfect framework in this perspective because the geometry is essentially preserved during the isogeometric refinement and the DG formulation allows a natural handling of non conformities.

A quadtree-like approach is adopted to refine rational Bézier patches. Whenever an element of Level-k is marked for refinement, it is split into 4 child elements of Level-(k+1) by inserting multiple knots at $\xi = 0.5$ and $\eta = 0.5$, as in the Bézier extraction procedure. The father element is stored and it can be recovered if its child elements are selected for coarsening. Thanks to the isogeometric paradigm, the same approach applies for both the geometry and the solution.

An error estimator specific to DG discretization is introduced to decides which elements should be refined, based on the measure of interface jumps. This indicator has the ability to identify regions with under-resolved elements. In practice, for each element Ω_j, one evaluates the solution jump at the interface between Ω_j and the neighbouring elements:

$$\varepsilon_j = \sum_{k \in \mathscr{N}_j} \int_{\Gamma_{jk}} \| \mathbf{W}|_{\Omega_j} - \mathbf{W}|_{\Omega_k} \| \, d\Gamma, \tag{16}$$

where $\mathscr{N}_j$ represents the set of elements around Ω_j and Γ_{jk} the interface between Ω_j and Ω_k. Then, the element Ω_j is flagged for refinement, respectively coarsening, if the indicator ε_j exceeds a used-defined criterion ε_{ref}, respectively ε_{coa}. In the case of coarsening, the solution should be approximated in the father element, from data located in the four sons elements. This is achieved by using a conservative least-squares projection. The whole adaptation procedure is detailed in [6].

We illustrate the proposed AMR approach for the case of an airfoil in transonic regime. The geometry and the corresponding secondary mesh are the ones described in section 3.1. The flow is governed by Euler equations, with a free-stream Mach number $M_\infty = 0.7$ and an incidence $\alpha = 1^\circ$. Cubic elements are used and three refinement levels are allowed starting from the secondary mesh. As can be seen in Fig. 11, the adaption procedure targets areas in the vicinity of shocks as well as leading and trailing edges, while maintaining a coarse grid where refinement is not required. As initially targeted, the whole evaluation procedure, starting from the NURBS-based geometry construction to the adapted flow simulation, is fully automated and CAD-consistent.

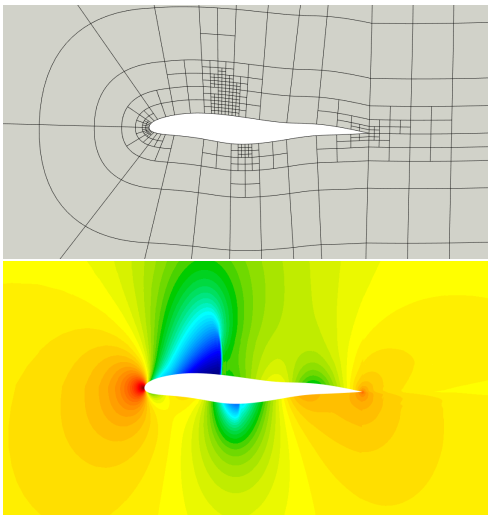

Figure 11: Adapted Mesh Refinement (top: adapted mesh, bottom: density field).

5. BAYESIAN OPTIMIZATION

The last component of the design loop is the optimization algorithm, that has to provide a new tentative of design parameters at each optimization step according to the knowledge acquired. We aim at solving complex problems, based on different physical behaviors, possibly characterized by the presence of multiple local minima, active constraints and including several optimization criteria. As a consequence, we choose to use a Bayesian Optimization (BO) approach as a constrained global optimization framework.

The BO methods [8] have been popularized by Jones [10]. They rely on the construction of Gaussian Processes [15] (GPs), also denoted Kriging models, as surrogates for the objective functions and constraints. These models are initialized on the basis of values obtained during a Design of Experiment (DoE) exploring phase [11], and then iteratively enriched with new values during the optimization phase. In this latter stage, the new points to be evaluated are determined via a merit function, that balances the optimization of the surrogate model and the improvement of its accuracy. The most commonly used merit function for single-objective problems is the *Expected Improvement* [10] (EI), that is computed from the expectation and variance of the GPs. Constraints are taken into account by introducing the *Expected Feasible Improvement* [17] (EFI) criterion.

6. OVERVIEW OF THE DESIGN LOOP

The assembly of the different methods described in the previous sections yields the design optimization loop depicted in Fig. 12. From a mathematical point of view, we seek for solving the following problem:

$$\begin{aligned} \min_{\mathbf{z} \in \mathscr{D}} \quad & g(\mathbf{z}), \\ \text{subject to} \quad & h(\mathbf{z}) \leq 0, \end{aligned} \tag{17}$$

where g is the cost function and h is the constraint function. The design variables $\mathbf{z}$ constitute a d-dimensional

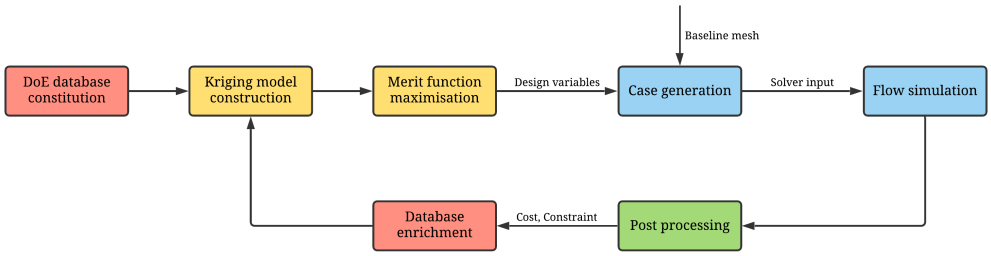

Figure 12: Design optimization loop.

Table 2: Airfoil optimization: bound constraints and optimal configuration.

index	2	3	4	5	6	7	8	9
y^+_{max}	0.040	0.070	0.080	0.080	0.080	0.070	0.050	0.040
y^+_{min}	0.030	0.040	0.040	0.040	0.040	0.030	0.020	0.010
y^+_{best}	0.032	0.050	0.057	0.064	0.063	0.059	0.042	0.018
y^-_{max}	-0.020	-0.030	-0.040	-0.040	-0.040	-0.020	-0.000	-0.000
y^-_{min}	-0.030	-0.050	-0.060	-0.070	-0.070	-0.050	-0.030	-0.020
y^-_{best}	-0.020	-0.050	-0.052	-0.049	-0.060	-0.040	-0.015	-0.007

vector contained in the design space $\mathscr{D} \subset \mathbb{R}^d$. Typically, the components of $\mathbf{z}$ represent the coordinates of the NURBS control points defining the geometry. Introducing the dependency of the cost function with respect to the flow solution $\mathbf{w}_h$ and the parametric representation of the geometry γ, the cost function writes:

$$g(\mathbf{z}) = \mathscr{G}\left(\gamma(\mathbf{z}), \mathbf{w}_h(\gamma(\mathbf{z}))\right). \tag{18}$$

Obviously, the same formulation holds for the constraint function. This formula underlines the main advantage of the proposed approach: the geometry $\gamma(\mathbf{z})$ used during the resolution of the state equations and the evaluation of the cost and constraint functions is not discretized (and approximated), contrary to methods based on classical piecewise-linear grids for which the computations rely on an approximated geometry $\gamma_h(\mathbf{z})$. This results in an increased accuracy, as demonstrated in previous works [5, 13]. Moreover, this simplifies the software coupling, thanks to the absence of a real mesh generation / deformation algorithm.

These advantages are put in light in the following sections, which deal with the application of the proposed methodology to the optimization of an airfoil shape.

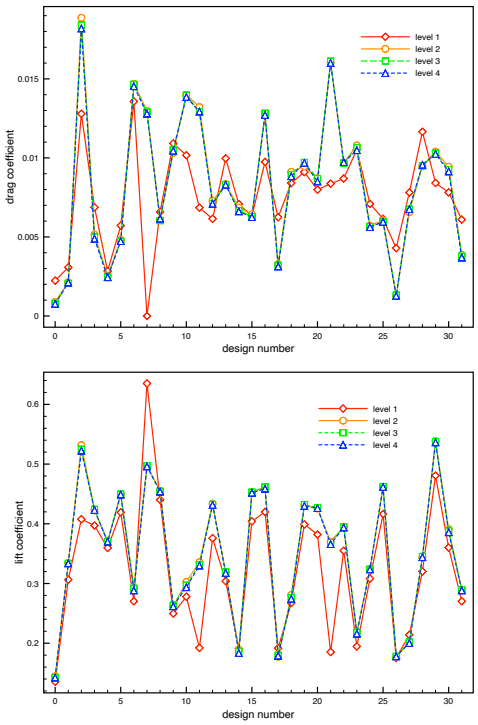

Figure 13: Drag and lift values in the DoE phase, for different refinement levels.

7. APPLICATION

We consider the optimization of the shape of an airfoil in transonic regime. The shape is defined using two NURBS curves of degree $p = 3$ and $n = 10$ control points, whose characteristics have been presented in section 3.2. The first and last control points are fixed during the optimization to maintain the leading and trailing edges. Inner con-

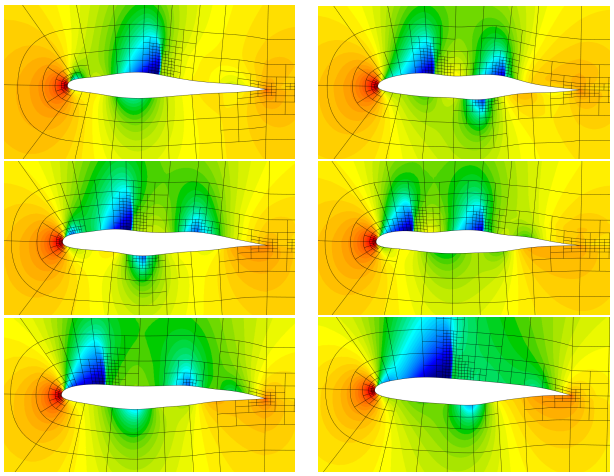

Figure 14: Some configurations from the DoE phase (adapted meshes and density fields).

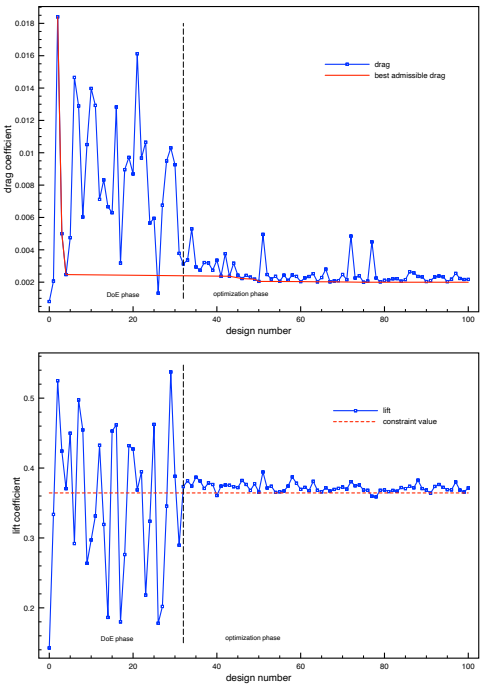

Figure 15: Drag and lift values during the design optimization procedure.

trol points are allowed to move vertically, yielding a set of $d = 16$ design parameters. The automated construction of the grid according to each design parameters set follows the steps described in section 3.1 and illustrated in 3.2.

The flow is modeled by compressible Euler equations and is characterized by a free-stream Mach number $M_\infty = 0.7$ and an incidence $\alpha = 1^\circ$. The adaptive DG method presented above is used to solve the state equations until a steady state is reached. The objective of the optimization problem is the minimization of the drag coefficient C_D subject to a constraint on the lift coefficient $C_l \geq C_l^0$ with $C_l^0 = 0.3644$.

The assessment of the flow computation is more tedious in the context of design optimization than for a single flow analysis. Indeed, a mesh that is satisfactory for a certain configuration may not be suitable for others which have different flow characteristics. This difficulty is alleviated by the use of AMR techniques. Nevertheless, one should select the maximum number of refinement levels allowed by the automated procedure. To assess as much as possible this choice, the DoE phase is achieved using different values, ranging from one to four refinement levels, then the lift and drag values obtained with the different levels are compared. Since the DoE explores the whole design space, this gives a more robust assessment than just considering a single starting configuration. The size of the DoE is set to $N_{DoE} = 32$. Therefore, 32 airfoil geometries are generated randomly, according to a Latin Hypercube sampling [11], in the domain detailed in Tab. 2. Fig. 13 depicts the lift and drag values obtained with the different refinement levels. As can be observed, the results corresponding to one refinement level are clearly far from those obtained with more refined grids. The values obtained with three and four levels are almost indistinguishable, therefore the optimization phase is achieved using three refinement levels.

Some configurations are extracted from the DoE and shown in Fig. 14, for three refinement levels. As can be seen, the shape modifications strongly impact the flow characteristics, in particular the number, the intensity and the location of the shocks. Therefore, the use of AMR techniques is critical here to define a suitable grid for each airfoil shape, while maintaining a low computational cost.

The evolution of the cost (drag) and constraint (lift) functions during the whole design optimization procedure can be seen in Fig. 15. As observed, the lift and drag values are quite well learnt after the DoE phase, yielding a fast convergence towards interesting areas. One can notice that most trials in the optimization phase verify the lift constraint, whereas only minor improvements of the drag are reported after design 50. The design parameters defining the best shape, corresponding to the design 75, are provided in Tab. 2. As seen, two control points have reach their bounds, at the suction side. Fig. 16 shows the density field (same colormap as Fig. 14), which is characterized by a quasi-absence of shock on the pressure side. As expected, the drag minimization results in a shock reduction.

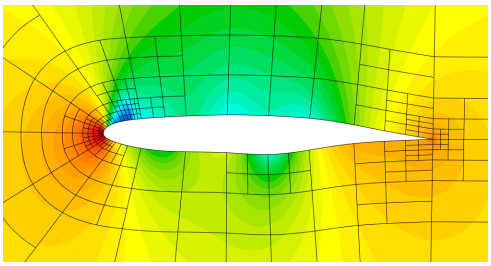

Figure 16: Optimal shape (adapted mesh and density field).

CONCLUSION

A fully integrated geometry-simulation-optimization framework has been presented, which entirely relies on CAD representations, yielding a CAD-consistent aerodynamic design procedure. The proposed methodology has been applied to the shape optimization of an airfoil in transonic regime, demonstrating the potentiality of the approach. Future developments will target the resolution of more complex configurations, from physical and geometrical viewpoints.

CODE REPOSITORY

The isogeometric analysis method is implemented in the **Igloo** software suite. 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`.

The optimization algorithm used is implemented in a set of open-source R packages [16, 2].

REFERENCES

[1] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5–8):229–263, 2010.

[2] M. Binois and V. Picheny. GPareto: An R package for Gaussian-process-based multi-objective optimization and analysis. *Journal of Statistical Software, Articles*, 89(8):1–30, 2019.

[3] B. Cockburn and C.-W. Shu. The local discontinuous galerkin method for time-dependent convection-diffusion systems. *SIAM Journal of Num. An.*, 35(6):2440–2463, 1998.

[4] C. De Boor. *A Practical Guide to Splines*. Springer Verlag, 1978.

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

[6] R. Duvigneau. CAD-consistent adaptive refinement using a nurbs-based discontinuous galerkin method. *Int. J. for Numerical Methods in Fluids*, February 2020.

[7] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, 1989.

[8] R. Garnett. *Bayesian Optimization*. Cambridge University Press, 2022.

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

[10] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[11] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[12] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous galerkin methods. In *AIAA Paper 2006-112*, 2006.

[13] S. Pezzano and R. Duvigneau. A NURBS-based Discontinuous Galerkin method for conservation laws with high-order moving meshes. *Journal of Computational Physics*, 434(1), 2021.

[14] L. Piegl and W. Tiller. *The NURBS book*. Springer-Verlag, 1995.

[15] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[16] O. Roustant, D. Ginsbourger, and Y. Deville. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51(1):1–55, 2012.

[17] M. Schonlau, W.J. Welch, and D.R. Jones. Global versus local search in constrained optimization of computer models. 1998.