Received: 17 May 2023 | Revised: 14 November 2023 | Accepted: 24 December 2023

DOI: 10.1002/nme.7426

RESEARCH ARTICLE

WILEY

A shape optimization pipeline for marine propellers by means of reduced order modeling techniques

Anna Ivagnes | **Nicola Demo** | **Gianluigi Rozza**

SISSA Mathematics Area, mathLab, Trieste, Italy

Correspondence
Gianluigi Rozza, SISSA Mathematics Area, mathLab, Via Bonomea, 265, Trieste, 34136, Italy.
Email: gianluigi.rozza@sissa.it

Funding information
H2020 European Research Council; European Social Fund Plus

Abstract

In this article, we propose a shape optimization pipeline for propeller blades, applied to naval applications. The geometrical features of a blade are exploited to parametrize it, allowing to obtain deformed blades by perturbating their parameters. The optimization is performed using a genetic algorithm that exploits the computational speed-up of reduced order models to maximize the efficiency of a given propeller. A standard offline–online procedure is exploited to construct the reduced-order model. In an expensive offline phase, the full order model, which reproduces an open water test, is set up in the open-source software OpenFOAM and the same full order setting is used to run the CFD simulations for all the deformed propellers. The collected high-fidelity snapshots and the deformed parameters are used in the online stage to build the nonintrusive reduced-order model. This article provides a proof of concept of the pipeline proposed, where the optimized propeller improves the efficiency of the original propeller.

KEYWORDS

blade propeller design, data-driven reduced order modeling, genetic algorithm, industrial shape optimization, mesh parametrization

1 | INTRODUCTION

The problem of optimizing the efficiency of marine propellers is a topic of considerable importance in naval engineering applications. The correct understanding of how the geometry of the blades would impact the propagation of vibrations and noise is propaedeutic to the design of new propellers and for the improvement of the propulsion performances. Therefore, a preliminary but necessary step in the optimization of propellers' shape is the geometrical parametrization of a single blade, where both the global geometrical features, that is, pitch, rake, skew, chord lengths, and the sections' properties, that is, camber, thickness, are recognized as parameters.[1,2] Modifications in the parameters lead to different blade shapes, allowing the exploration of a large number of different shapes. In the project here presented, we started from a blade provided by Fincantieri S.p.A., but the starting blade can be in general defined starting from its parameters.

The step following the parametrization is the setup of the computational fluid dynamic (CFD) simulation in order to compute the propeller performances. In particular, different simulation settings have been investigated in the state of the art: *open water* tests,[3–5] usually studied at a reduced scale, and the inverse problem concerning the wake generated by the propeller, analyzed in previous works, such as References 6 and 7. In particular, this article focuses on the optimization of propeller efficiency in open-water tests. From a computational point of view, the main challenge is to find a compromise

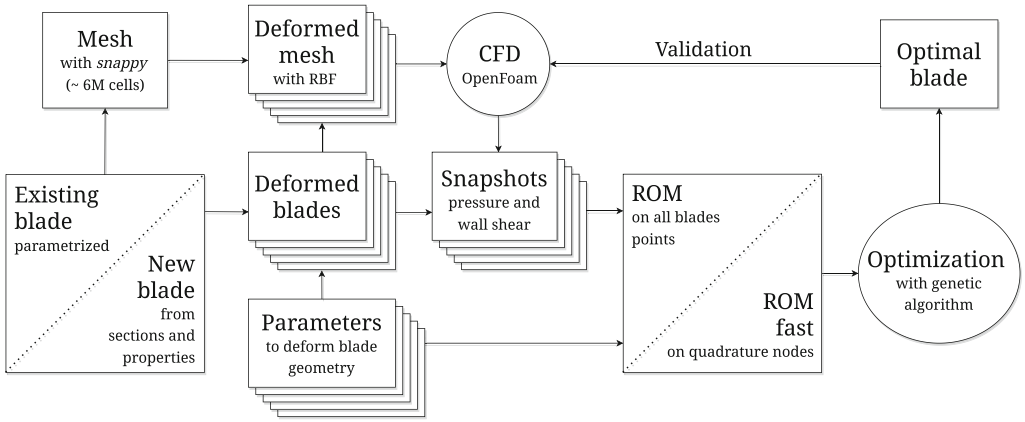


FIGURE 1 Pipeline of the shape optimization.

between the computational cost of high-fidelity simulations, that is, the mesh refinement, the computational resources, and the precision in the reconstruction of the most significant physical fields in experimental tests, that is, the torque and the thrust coefficients. Indeed, an accurate reconstruction of the experimental tests would require a significantly large number of degrees of freedom and the computational effort needed for the optimization process would become unfeasible.

To mitigate this issue, this contribution introduces an optimization pipeline exploiting the potentialities of *data-driven* or *nonintrusive* reduced order models (ROMs).[8–15] ROMs constitutes a consolidated method for real-time approximation of the numerical solutions in different problem configurations, that is, finite volume, finite element, and so on, and in naval applications. We can find evidence of the ROM potentiality in naval problems in previous works, such that References 16–24.

The standard approach when dealing with ROMs is the *offline–online* procedure. The offline stage consists of the computation of a large number of high-fidelity simulations, each one exploiting a different deformed blade; the full-order snapshots of the main flow fields, that is, pressure and wall shear stress are then collected. In the online stage, the reduced order model built from snapshots and parameters is exploited to predict the propeller efficiency in the optimization process.

This contribution proposes two different ROMs approaches: a standard one, where the snapshots are evaluated on all the mesh points of the blades, and a *fast* ROM, where the fields are evaluated only on a reduced number of blades points, that is, the Gauss quadrature nodes. To the knowledge of the authors, this approach is here experimented for the first time in a ROM fashion and allows for a significant reduction in the computational time without losing the ROM accuracy.

We now present the structure of the article, which follows the pipeline represented in Figure 1. Section 2 will analyze the geometry of a propeller and of a single blade (Section 2.1), the deformation of the original propeller (Section 2.2), and of the whole mesh (Section 2.3). The setting of the full order model (FOM), which is simulated in the open-source software *OpenFOAM*, is described in detail in Section 3. Then, the basic theory of reduced order models (ROM) is presented in Section 4, where two different ROMs approaches are described (Sections 4.1 and 4.2). Section 5 describes how the two ROMs techniques are exploited either in a genetic (Section 5.1) or in a gradient-based method (Section 5.2). The results obtained in all the optimization processes are there compared.

2 | PROPELLER GEOMETRY AND MESH DEFORMATION

This section is dedicated to the analysis of the geometry of a single blade, followed by the exploitation of its main parameters to obtain different deformed shapes (Sections 2.1 and 2.2).

The deformation of the blade would result in the deformation of the whole *OpenFOAM* mesh in our offline simulations. This specific issue is addressed in Section 2.3.

2.1 | Blade parameters

The first important step in the development of efficient propellers design is the geometrical parametrization of a single blade of the propeller.

The generic structure of a propeller is displayed in Figure 2, where the main components of the propeller are distinguished, the blades, the hub and the shaft. As can be seen from Figure 2, in this work we focus on a propeller a fixed number of blades, namely 6 blades.

The blades of marine propellers are characterized by different geometrical features, which are measured on different cylindrical sections and are associated with the values of the radius at which sections are taken. Figure 3 represents the values of the radii taken into account in our particular case study for blade parametrization. In particular, r_0 is the hub radius, whereas R is the maximum radius of the propeller. As a usual approach for the study of marine propellers, the reference radii r are all measured with respect to $\frac{r}{R}$, as can be seen from Figure 3. In particular, the blade is characterized by four faces: the sections at radius r_0 and R, named *root* and *tip* respectively, the blade *face* (or *pressure side*) and *back* (or *suction side*), whose borders are indicated in Figure 4. We can distinguish two classes of parameters: the *global* parameters, associated with each radius value, that is, pitch, rake, skew, chord length; the *section* parameters, that is, quantities defined at different chord percentages for each section, such as thickness and camber*.

The parameters which are taken into account for the blade deformation presented in this work are pitch, chord length, thickness, and camber, whereas the other parameters are fixed for all the deformed shapes. More in detail, the pitch of a propeller is the displacement that it makes in a complete spin of 360 degrees, we are considering a propeller with a different pitch distribution for each section considered. The other parameters considered, that is, chord length, thickness, and camber, are graphically defined in Figure 4A. In particular, for each section thickness and camber are defined as

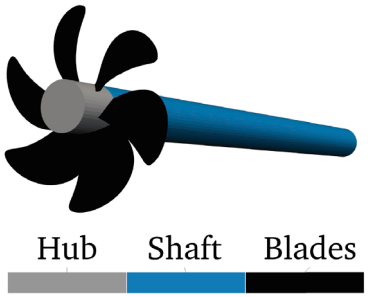


FIGURE 2 Generic structure of a propeller.

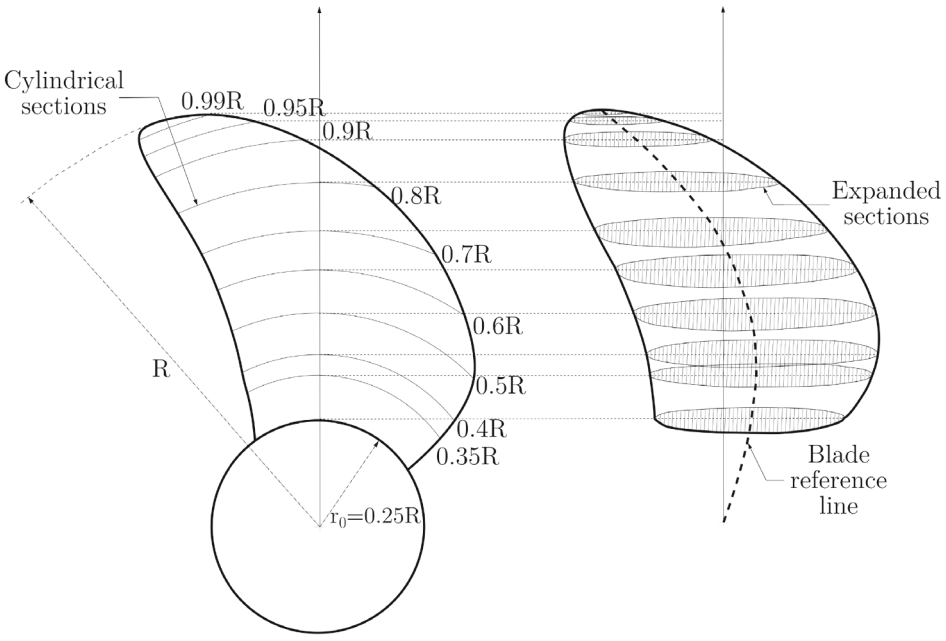


FIGURE 3 Radial view of a propeller's blade. The figure on the left shows the original cylinder sections of a blade, where R is the propeller radius and r_0 is the hub radius. The figure on the right represents the projections of the corresponding sections on a flat plane.

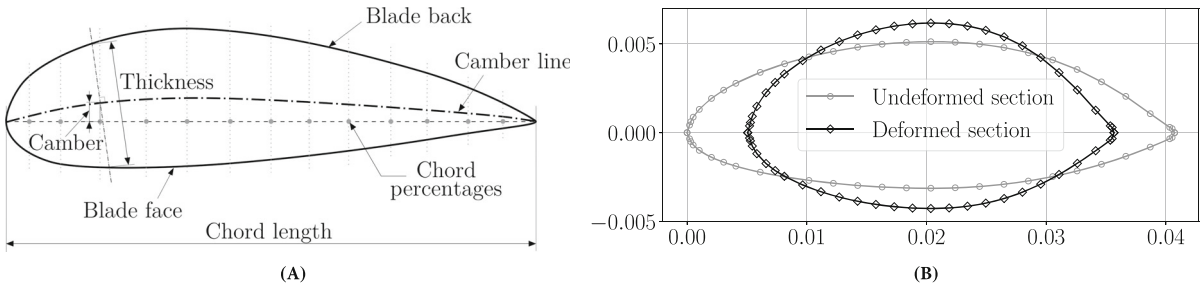


FIGURE 4 (A) Graphical definition of the parameters of a blade section: chord length, thickness, and camber corresponding to the preset chord percentages. (B) Example of deformed blade root section.

lists of values associated with predefined chord percentages, as represented in Figure 4A. The parameters of the initial shape are reserved data[†], but we highlight that the entire pipeline is reproducible with all the propellers defined by the above-mentioned parametrization.

Remark: It is worth remarking that we adopt the American convention in the definition of the thickness since it is measured along lines orthogonal to the camber line. In the British convention, instead, it is measured along vertical lines, orthogonal to the chord line at the specific chord percentages considered.

2.2 | Blade deformation

Once the geometrical parameters of interest are selected, changes in the blade geometry can be easily made by modifying the parameters. In this work, the deformation parameters are the multiplicative factor we impose on the original geometrical parameters of the original blade. For instance, if we consider a deformation parameter equal to 1.3 for the pitch, the resulting deformed blade will have the pitch geometrical parameter 30% greater than the original blade.

The deformation rates' values are set within the intervals:

- $[0.9,\ 1.1]$ (for pitch);
- $[0.8,\ 1.2]$ (for camber);
- $[0.7,\ 1.3]$ (for chord length);
- $[0.7,\ 1.3]$ (for thickness).

The ranges have been chosen such that the parametric blades satisfy the structural feasibility constraint. Figure 4B displays an example of deformation of the root section of the blade when the deformation parameter is $\mu = [1\ \ 0.95\ \ 0.75\ \ 1.27]$, where the components refer to pitch, camber, chord length, and thickness, respectively. Such deformation is obtained by employing the Python package `BladeX`.[25,26] The software builds the blades by creating a Non-Uniform Rational Basis Spline (NURBS) surface [27,28] passing through all the (deformed) blade sections. In this representation, the shape of the surfaces is defined by the control points belonging to the UV plane, which enables the following steps of the pipeline thanks to the mapping between such a (2D) plane and the 3D final surface.

Remark: It is worth highlighting that we considered the same deformation rates for each section and each chord percentage, that is, we are not taking into account different deformations for different sections and/or chord coordinates. Therefore, we are considering $p = 4$ parameters for the whole blade.

In the project here presented, we considered $M = 216$ deformed blades, where 200 blades are obtained imposing deformations with parameters selected with uniform random distribution in the intervals defined above, and 16 blades are deformed considering all the combinations at the extremes of the intervals. Three examples of deformed shapes are displayed in Figure 5.

2.3 | Mesh deformation

A challenging step in the pipeline of this project is to perform the mesh deformation. Indeed, after a large number of deformed blades is obtained, as described in Section 2.1, the computational mesh (schematized in Figure 11) has to be deformed as well, according to each blade deformation. However, it is important to highlight one important requirement

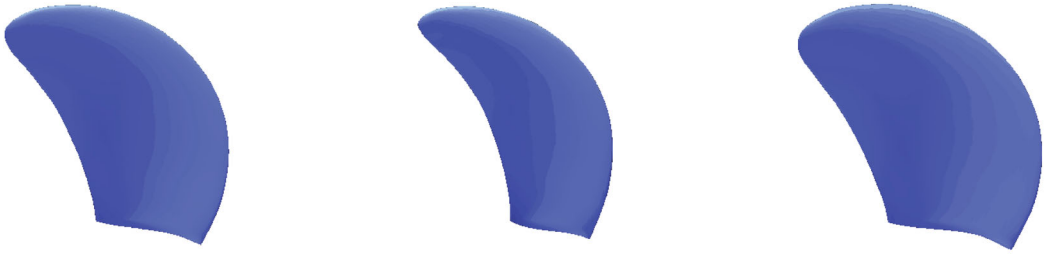


FIGURE 5 Graphical view of three deformed blades.

that all the deformed meshes have to satisfy: the mesh topology, as well as the number of degrees of freedom, should be preserved in all the FOM simulations. The statement above is a prerequisite for the construction of the reduced order model since all the pressure and velocity high-fidelity solutions should have the same degrees of freedom.

A mesh that has the same number of points and the same topology as the FOM simulation with the undeformed blades cannot be obtained from the *OpenFOAM* tool of mesh generation. Therefore, the mesh deformation is performed in this work by exploiting an interpolation technique. In particular, we followed two fundamental steps to obtain the points of each deformed mesh: the deformation of the propeller's shaft and the deformation of the entire mesh, described in detail in the following parts. The images reported in Sections 2.3.1–2.3.3 display results for the deformed blade with parameter $\mu = \begin{bmatrix} 1 & 0.72 & 0.75 & 0.95 \end{bmatrix}$.

Before specifying the operative details of the deformation, we recall in the following paragraph the basic concepts of the radial basis function (RBF) interpolation technique when used for mesh deformation.

2.3.1 | RBF for mesh deformation

The general problem is to find the deformed counterpart $\mathbf{x}^{\text{def}}$ of the original undeformed mesh $\mathbf{x}^{\text{undef}}$. As starting point we should consider a certain number N_{control} of undeformed and deformed control points $\mathbf{c}^{\text{undef}}$ and $\mathbf{c}^{\text{def}}$. The deformed mesh is found with the following expression:

$$x_j^{\text{def}} = f\left(x_j^{\text{undef}}\right) = \sum_{i=1}^{N_{\text{control}}} \alpha_i \phi\left(\left\| x_j^{\text{undef}} - c_i^{\text{undef}} \right\|\right), \quad j = 1, \ldots, N_{\text{points}}, \tag{1}$$

where $\phi_i = \phi\left(\left\| x_j^{\text{undef}} - c_i^{\text{undef}} \right\|\right)$ are the radial basis functions, which can have different shapes, such as thin plate splines, multiquadric or inverse multiquadric. In our case, we consider the *thin plate splines*, thus: $\phi(r) = r^2 \log(r)$, where r is the radius of the basis function, that is, $r = \|x_j^{\text{undef}} - c_i^{\text{undef}}\|$.

Remark: We highlight here that choosing a proper RBF kernel would ensure the preservation of the mesh quality and topology throughout the discretized domain. The weights α_i are found by *training* the interpolation with some known control points, that have to satisfy the following conditions:

$$c_j^{\text{def}} = f\left(c_j^{\text{undef}}\right) = \sum_{i=1}^{N_{\text{control}}} \alpha_i \phi\left(\left\| c_j^{\text{undef}} - c_i^{\text{undef}} \right\|\right), \quad j = 1, \ldots, N_{\text{control}}. \tag{2}$$

2.3.2 | Shaft deformation

For each deformed blade, the following list of passages is followed to obtain the deformed shaft:

1. identification of N_{shaft} points on the root face of the undeformed and deformed blades. Being the surface defined by NURBS, the points are generated on the reference UV plane and mapped to the blade surface;
2. collection of the shaft points which remain unchanged in deformation, that is, the two bases of the shaft. The union of the undeformed shaft bases and of the undeformed blades' root provides the *undeformed shaft control points*, whereas the union of the undeformed shaft bases and of the deformed blades' root provides the *deformed shaft control points*;

3. training of a *Radial Basis Function* (RBF) interpolating technique with the *undeformed* and *deformed shaft control points* (as in (2)), and then extraction of the deformed shaft by applying the RBF to the undeformed points of the shaft lateral surface (as in (1)). The control points and the final mesh points on the shaft are represented in Figure 6.

To conclude, we highlight that such an operation is needed in order to allow the correct deformation of the whole mesh. If the points on the shaft surface are fixed, this would lead indeed to unexpected behavior when the RBF system is assembled over all the surface points. In fact, in this case the blade points are moved, but not the adjacent ones belonging to the shaft. The propedeutic procedure here described induces indeed a deformation on the shaft points, translating them on the cylindrical surface accordingly to the blade root deformation.

2.3.3 | Global mesh deformation

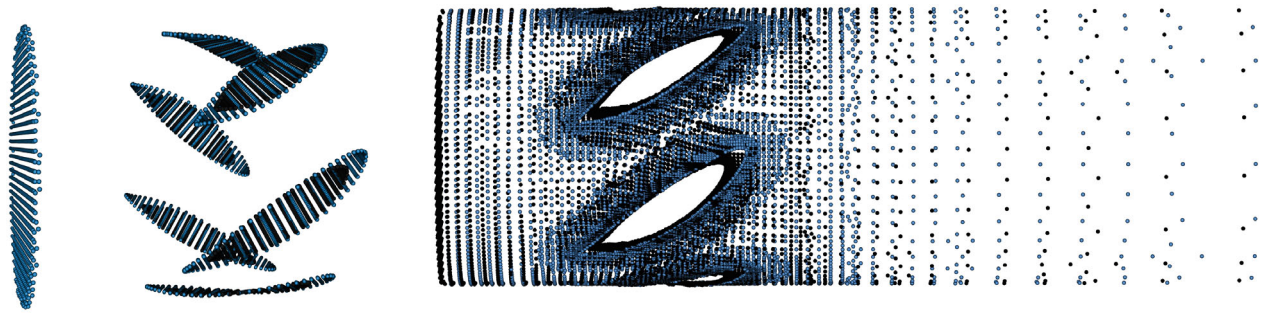
The deformation of the whole mesh is obtained with the following steps:

1. identification of $N_{\text{quadrature}}$ Gauss quadrature nodes on the faces of both the undeformed and deformed blades; we name these nodes as *undeformed* and *deformed blades' control points*. Here we chose as control points the Gauss quadrature nodes on the blades, but other selections that ensure a proper geometrical characterization of the blades are possible.
2. collection of all the points of the undeformed mesh which belong to the boundaries, that is, the lateral surface of the outer cylinder Γ_{outer}, the inlet Γ_{inlet}, the outlet Γ_{outlet}, and the surfaces of the propeller shaft Γ_{shaft}. The union of the undeformed blades' control points and the undeformed boundaries will provide the *undeformed mesh control points*, while the union of the deformed blades' control points and the undeformed boundaries will provide the *deformed mesh control points*;
3. training of a second RBF interpolation with the undeformed and deformed mesh control points (as reported in (2)); then, the RBF is applied on the entire internal mesh (again, as in (1)) to obtain the deformed internal mesh, which is represented in Figure 7.

Remark: It is important to remark that, if we want to perform a deformation of a certain domain, there is the need to include in the control points the boundaries of the domain, that is, the bases of the shaft in point (1) and the boundaries of the whole domain in point (2).

2.3.4 | Results for mesh deformation

This part is dedicated to the graphic results of the mesh deformation step. In particular, here we show different mesh views for the original and the deformed blades and for the meshes of the corresponding FOM simulations. For a detailed description of the FOM setting we refer the reader to Section 3.



(A) (B)

FIGURE 6 (A) Zoom on the control points on the propeller's shaft. (B) Shaft points. Undeformed and deformed shaft points are represented in blue and black, respectively.

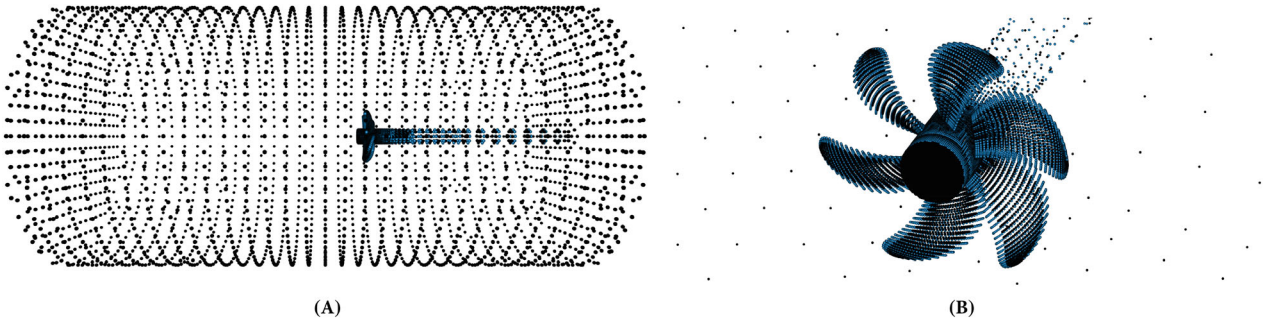


FIGURE 7 (A) Representation of all the control points in the mesh, including points at boundaries and on the propeller's surface; (B) Zoom on the undeformed (blue) and deformed (black) control points on blades.

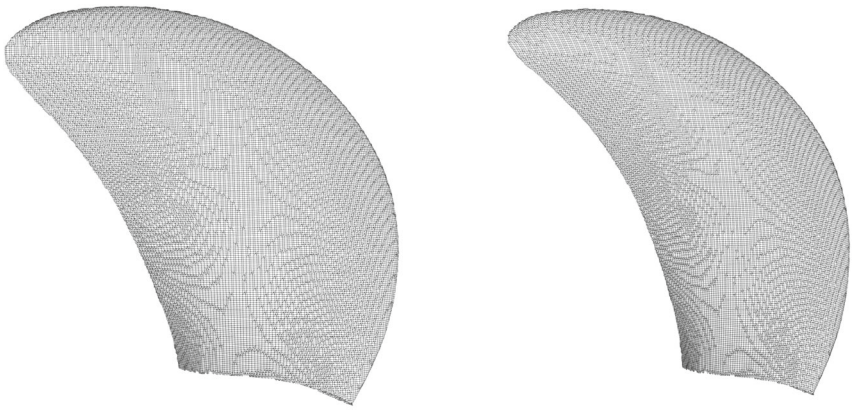


FIGURE 8 Mesh on the surface of the original undeformed blade and of a selected deformed blade.

Figure 8 displays the mesh on the surfaces of the undeformed and deformed blades, Figures 9 and 10 represent the slices of the mesh on two different planes, parallel and orthogonal to the propeller axis, respectively. All the deformed meshes obtained through the above-mentioned interpolation technique provide good results in terms of mesh quality.

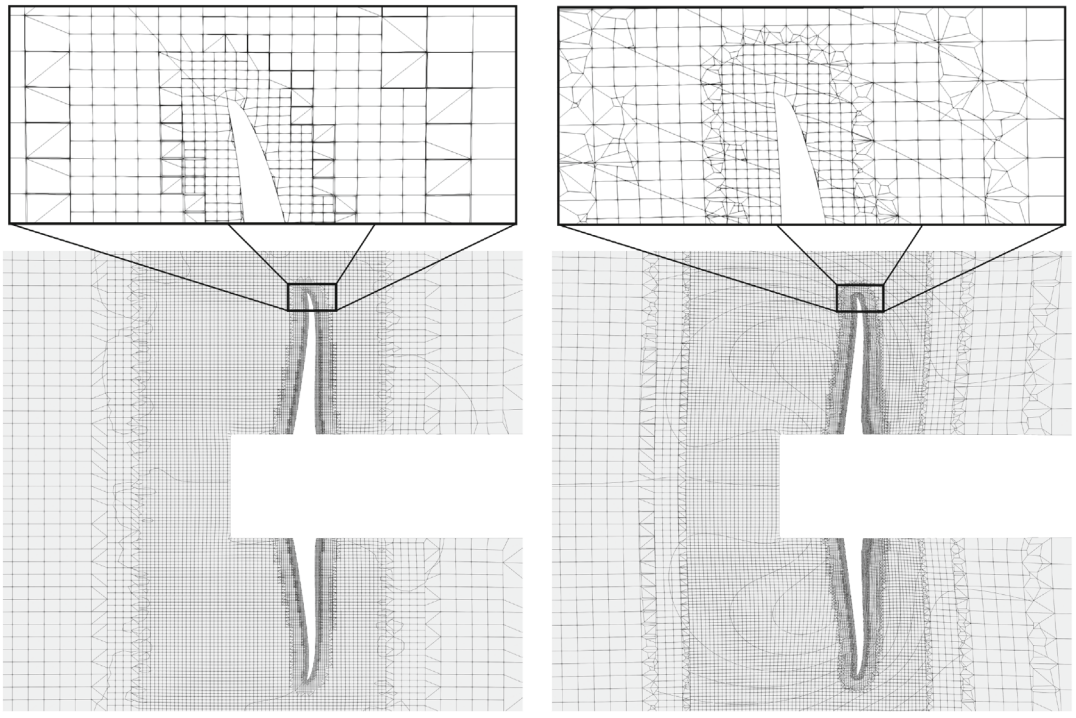
The time needed to perform the mesh deformation using a RBF interpolation method is ∼2–2.5 h in a serial computation[‡].

3 | FULL ORDER MODEL

A necessary step in our pipeline is the setting of the full order model (FOM) for high-fidelity simulations. The mathematical model here considered is the Unsteady Reynolds Averaged Navier–Stokes (U-RANS) Equations, that are numerically discretized and solved making use of the open-source software *OpenFOAM*,[29] which exploits the finite-volume method.[30] Section 3.1 describes the U-RANS approach coupled with the turbulence modeling, whereas Section 3.2 explains the technique used to generate the rotation of the propeller in the FOM simulations.

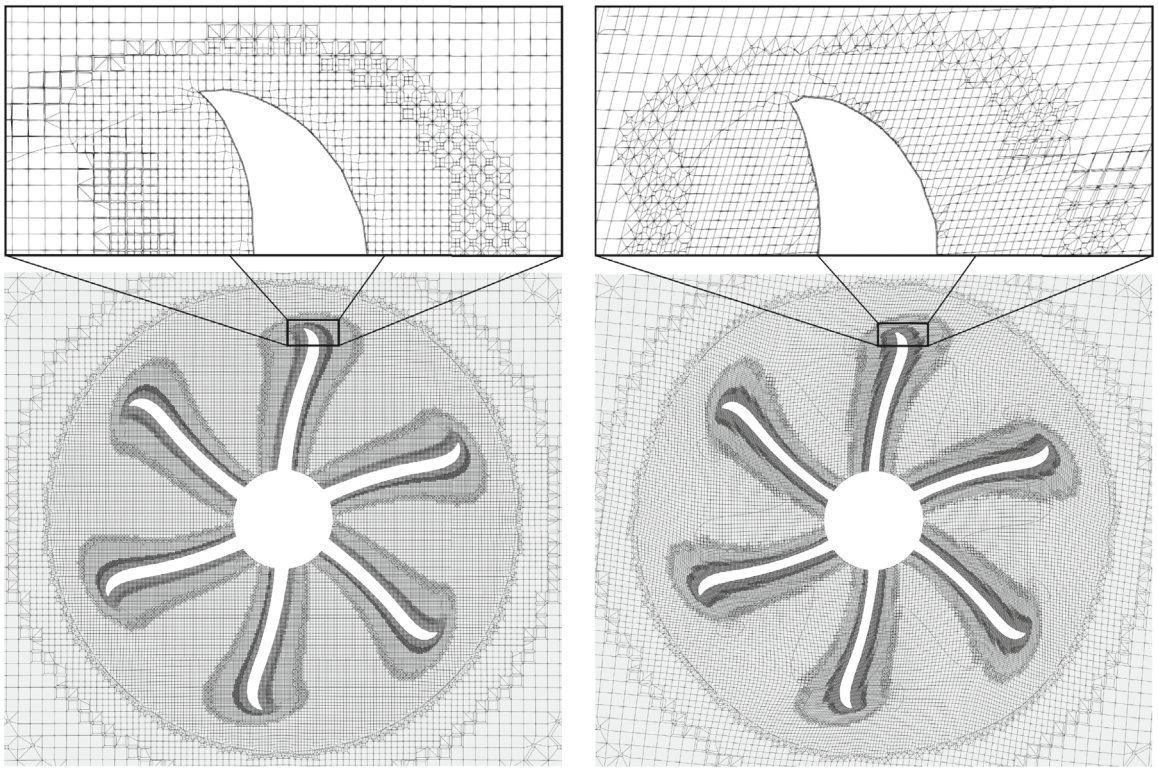
3.1 | The U-RANS approach and the turbulence modeling

In this article, we adopt the following notation: $\Omega \in \mathbb{R}^d$ with $d = 2$ or 3 is the fluid domain, Γ its boundary, $t \in [0, T]$ the time, $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ the velocity field, $p = p(\mathbf{x}, t)$ the normalized pressure scalar field divided by the fluid density, and ν the fluid kinematic viscosity.



(A) Undeformed mesh slice **(B)** Deformed mesh slice

FIGURE 9 Zoom of a slice of the mesh with corresponding normal parallel to the propeller axis.



(A) Undeformed mesh slice **(B)** Deformed mesh slice

FIGURE 10 Zoom of a slice of the mesh with corresponding normal orthogonal to the propeller axis.

Figure 11 represents a slice of our computational domain and mesh. In particular, the mesh is built inside the outer cylinder, whose bases are the inlet Γ_{inlet}, the outlet Γ_{outlet} and the lateral surface Γ_{outer}. The surface of the propeller is named here $\Gamma_{\text{propeller}}$. In the internal domain, four different cylinders are built, whose surfaces are indicated as $(\Gamma_i)_{i=1}^4$. We introduce here the internal cylinders in order to generate different mesh refinements; in particular, the mesh gets gradually more refined from the outer surface to the blades' surface in order to provide an accurate reconstruction of the flow fields acting on the blades, as can be seen from Figure 11.

We briefly recall the basic concepts of the U-RANS approach, that is used in this work for the FOM simulations. The main hypothesis that characterizes the RANS approach is the *Reynolds decomposition*.[31] This theory is based on the assumption that each flow field can be expressed as the sum of its mean and fluctuating parts. Considering a generic space and time-dependent field $\sigma(\mathbf{x}, t)$:

$$\sigma(\mathbf{x}, t) = \overline{\sigma}(\mathbf{x}, t) + \sigma'(\mathbf{x}, t).$$

The U-RANS formulation consists in a time-averaged version of the NSE, that can be written as follows:

$$\begin{cases} \dfrac{\partial \overline{u_i}}{\partial x_i} = 0, \\ \dfrac{\partial \overline{u_i}}{\partial t} + \overline{u_j}\dfrac{\partial \overline{u_i}}{\partial x_j} = -\dfrac{\partial \overline{p}}{\partial x_i} + \dfrac{\partial (2\nu \overline{\mathbf{E}}_{ij} - \mathcal{R}_{ij})}{\partial x_j}, \end{cases} \tag{3}$$

where the Einstein notation has been adopted, $\mathcal{R}_{ij} = \overline{u_i' u_j'}$ is the Reynolds stress tensor, and $\overline{\mathbf{E}}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right)$ is the averaged strain rate tensor.

The U-RANS formulation in (3) needs to be coupled with a turbulence model in order to close system (3). In particular, here we adopt the $\kappa - \omega$ *Shear Stress Transport* (SST) model.[32] It belongs to the class of *eddy viscosity models*, whose main assumption is the Boussinesq hypothesis, that is, that the turbulent stresses are related to the mean velocity gradients as follows:

$$\mathcal{R}_{ij} = 2\nu_t \mathbf{E}_{ij} - \frac{2}{3}\kappa \delta_{ij},$$

where $\kappa = \frac{1}{2}\overline{u_i' u_i'}$ is the turbulent kinetic energy, ν_t is the eddy viscosity.

The final formulation is expressed as follows:

$$\begin{cases} \dfrac{\partial \overline{\mathbf{u}}}{\partial t} + \nabla \cdot (\overline{\mathbf{u}} \otimes \overline{\mathbf{u}}) = \nabla \cdot \left[-\overline{p}\mathbf{I} + (\nu + \nu_t)\left(\nabla \overline{\mathbf{u}} + (\nabla \overline{\mathbf{u}})^T\right)\right] & \text{in } \Omega \times [0, T], \quad (4a) \\ \nabla \cdot \overline{\mathbf{u}} = 0 & \text{in } \Omega \times [0, T], \quad (4b) \\ + \text{Boundary conditions} & \text{on } \partial\Omega \times [0, T], \quad (4c) \\ + \text{Initial conditions} & \text{in } (\Omega, 0). \quad (4d) \end{cases}$$

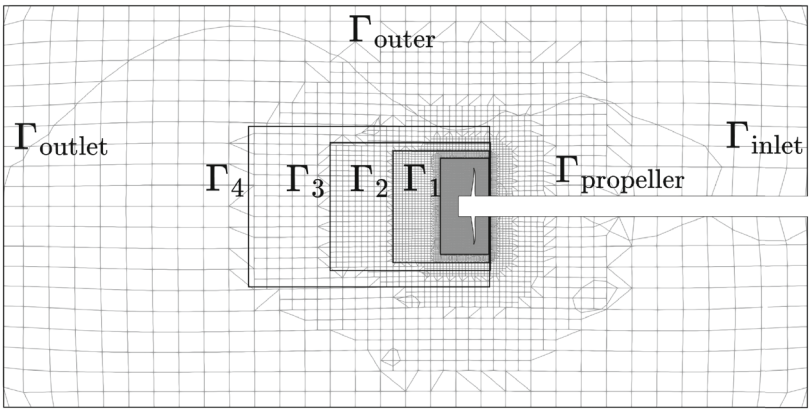


FIGURE 11 A slice of the computational domain Ω and mesh of the FOM in *OpenFOAM*.

The boundary and initial conditions (4c) and (4d) are here reported. For the pressure field, we consider:

$$\begin{cases} p(\mathbf{x},0)=0, & \mathbf{x}\in\Omega, \\ \dfrac{\partial p}{\partial \mathbf{n}}(\mathbf{x},t)=0, & \mathbf{x}\in\Gamma_{\text{inlet}},\Gamma_{\text{outer}}, \\ p(\mathbf{x},t)=0, & \mathbf{x}\in\Gamma_{\text{outlet}}. \end{cases}$$

For the velocity field:

$$\begin{cases} \mathbf{u}(\mathbf{x},0)=\mathbf{0}, \mathbf{x}\in\Omega, \\ \mathbf{u}(\mathbf{x},t)=\mathbf{u}_0(\mathbf{x})=(0,u_0,0), & \mathbf{x}\in\Gamma_{\text{inlet}} \\ \dfrac{\partial \mathbf{u}}{\partial \mathbf{n}}(\mathbf{x},t)=0, & \mathbf{x}\in\Gamma_{\text{outlet}} \\ \mathbf{u}(\mathbf{x},t)=\mathbf{0}, \mathbf{x}\in\Gamma_{\text{outer}},\Gamma_{\text{propeller}}. \end{cases}$$

In our case, the value of the inlet velocity u_0 is set such that the advance ratio $J=\frac{u_0}{nD}=0.85$, where $D=2R$ (if we refer to R in Figure 3) is the diameter of the propeller, and $n=15$ rounds/second is the number of revolutions per second of the propeller.

Moreover, the $\kappa-\omega$ SST model is coupled with a $\gamma-Re_{\theta_t}$ transition model, which simulates the laminar-turbulent transition in our simulation setting and was proposed in Reference 33. The initial and boundary conditions for the additional variables κ and ω, γ and Re_{θ_t} are set in the FOM following standard formulas and rules. In particular, the turbulent kinetic energy κ and the turbulence specific dissipation rate ω are initialized to:

$$\kappa=\frac{3}{2}(I\ u_0)^2, \omega=\frac{\kappa^{0.5}}{C_\mu^{0.5}L}$$

in all the domain, where the I is the turbulence intensity, that is $\sim 9\%$ in our case, $C_\mu=0.09$ and $L=D$ is the reference length scale. The intermittency variable γ and the transition momentum thickness Reynolds number Re_{θ_t} are initialized to:

$$\gamma=1, Re_{\theta_t}=\begin{cases} 1175.51-589.428I+\frac{0.2196}{I^2} & \text{if } I<1.3, \\ \frac{331.5}{(I-0.5658)^{0.671}} & \text{if } I>1.3. \end{cases}$$

For what concerns the boundary conditions, we have for all variables a fixed-value condition at the inlet, and a zero-gradient condition at the outlet and at the walls.

As stated before, in this article we adopt the finite volume discretization technique, which consists in performing a polyhedral discretization of the computational domain, where each finite volume is called *control volume*. After that, the equations in (4) are written in integrated on each control volume of the domain interest. The divergence theorem is then used to convert the volume integrals to surface integrals, which are finally discretized as sums of the fluxes at the boundary faces of each control volume.

3.2 | MRF approach

For what concerns the motion of the propeller, the rotation of the propeller in *OpenFOAM* is obtained using the *Moving Reference Frame* (MRF) approach. The MRF technique is a steady-state method widely spread in industrial CFD problems that involve rotating parts. The principle of this technique is the creation of a thin volumetric region of mesh cells around the rotating body during the meshing phase. This region, namely the *MRF zone*, is in our test case the smallest cylinder which surrounds the propeller's blades (Γ_1 in Figure 11). During the simulation, the MRF zone is rotated about the axis of the body and the body is kept stationary. The simulation is performed until a steady state is reached, such that the thrust and the torque forces reach a stationary value.

This stage aims to build a CFD model which provides accurate results compared to the experimental open-water tests. In particular, our metrics for accuracy measurement are the relative error on the values of the thrust and torque

coefficients (k_T and k_Q). These adimensional numbers are characteristic features in the fluid dynamics of marine propellers and are defined as follows:

$$k_T = \frac{T}{\rho n^2 D^4}, \quad k_Q = \frac{Q}{\rho n^2 D^5}. \tag{5}$$

In expression (5), ρ is the fluid density. T and Q are the thrust and the torque force experimented on the blades of our propeller. These forces are evaluated as the sum of a pressure-based and a viscous contribution, as follows:

$$\begin{aligned} T &= T_{\text{pressure}} + T_{\text{viscous}} = \rho\left(\int_{\Gamma_{\text{blades}}} p\mathbf{n}dA + \int_{\Gamma_{\text{blades}}} \boldsymbol{\Sigma}\mathbf{n}dA\right); \\ Q &= Q_{\text{pressure}} + Q_{\text{viscous}} = \rho\left(\int_{\Gamma_{\text{blades}}} p\mathbf{n}\times\mathbf{r}dA + \int_{\Gamma_{\text{blades}}} \boldsymbol{\Sigma}\mathbf{n}\times\mathbf{r}dA\right). \end{aligned} \tag{6}$$

In expression (6), Γ_{blades} indicated the blades' surface, $\mathbf{r} = (x, y, z)$ is the position vector, $\mathbf{n}$ is the unitary normal vector to the surface, $\boldsymbol{\Sigma}$ is the wall-shear stress tensor.

In the project presented in this article, the FOM performed in *OpenFOAM* reached an accuracy of $\sim 1\%$ and of $\sim 3\%$ for what concerns k_T and k_Q, respectively. Moreover, the time needed to perform one high-fidelity simulation is $24 - 48$ hours in a parallel setting.[§] The variance in the simulation time is due to the different number of iterations needed to reach a stable regime in all the computed simulations.

4 | NON-INTRUSIVE REDUCED ORDER MODEL

This section is dedicated to the explanation of the theory behind the model order reduction technique here employed to reduce the computational effort. This technique belongs to the framework of *nonintrusive reduced order methods*, that is, approaches exploiting the information provided by the high-fidelity simulations. In these kinds of models, the CFD governing equations are only used at the full order level to perform the offline simulations, but not at the reduced order level. Nonintrusive reduced order models are composed of two fundamental stages, the *reduction* and the *approximation* step, described in the following paragraphs. The techniques here explained are implemented in the Python package `EZyRB`.[34,35]

4.1 | Reduction

The first step consists of the compression of the original matrix of high-fidelity solutions, into a matrix of reduced dimension. In our case, we consider as snapshots the pressure and wall shear stress at the final time instant of each offline simulation, that is, when a steady state has been reached. The fields are evaluated not on all the points of the computational mesh, but only on the points of the blades.

It is important to remark that the flow fields are evaluated only on the blades because the propeller efficiency only depends on the fluid-dynamics behavior at the blades. In fact, the efficiency is evaluated as:

$$\eta_{\text{propeller}} = \frac{T}{Q}\frac{u_0}{2\pi n}, \tag{7}$$

where T and Q only depend on the fields on the blades, as can be evinced from (6).

Thus, each $i-$th snapshot is a flow field evaluated on the deformed blades corresponding to the $i-$th set of deformation parameters taken into account. For instance, if we consider the generic field s, we have the following matrix of snapshots:

$$\mathbf{S} = \begin{bmatrix} | & | & & | \\ \mathbf{s}_1(\mathbf{x}) & \mathbf{s}_2(\mathbf{x}) & \dots & \mathbf{s}_{\text{M}}(\mathbf{x}) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{N_{\text{dof}}\times \text{M}},$$

where N_{dof} is:

- exactly the number of cells of the blades in the computational mesh, that is $\sim 28 \times 10^5$ in the first type of ROM we consider (the ***standard*** *ROM*). In this case, the snapshots are evaluated at the cells' centers;
- a certain number of quadrature points retained on the blades, in our case 12×10^5, since we consider a grid of 100×100 nodes for the back and the face of each blade, in the second type of ROM (the ***fast*** *ROM*). In this case, the snapshots are evaluated on the quadrature points.

The basic principles of the two theories will be explored in Sections 4.1 and 4.2.

The reduction technique adopted in this work is the Proper Orthogonal Decomposition (POD). This technique is based on the projection of the snapshots into a reduced space, spanned by a limited number of the so-called *modes*, which are computed directly starting from the snapshots matrix in the offline stage. Each *reduced* snapshot $\mathbf{s}_i$ can be approximated as a linear combination of the modes:

$$\mathbf{s}_i \simeq \sum_{j=1}^{L} a_j \boldsymbol{\phi}_j,$$

where $\{\boldsymbol{\phi}_j\}_{j=1}^{L}$, are the modes, $L \ll N_{\text{dof}}$ is the reduced dimension, that has to be established a priori. $\{a_j\}_{j=1}^{L}$ are the reduced coefficients associated to the modes. The POD modes can be evaluated using a Singular Value Decomposition technique (SVD) or via the correlation matrix. In the first case, for instance, the snapshots matrix is decomposed in $\mathbf{S} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, where the columns of $\mathbf{U} \in \mathbb{R}^{N_{\text{dof}} \times L}$ are the POD modes. Thus, the reduced coefficients can be computed as $\mathbf{U}^T\mathbf{S}$.

4.2 | Approximation

In the second step, the goal is to predict the reduced coefficients associated with unknown values of the parameters, which do not belong to the original dataset. In our case, the snapshots are $\mathbf{s}_i = \mathbf{s}(\boldsymbol{\mu}_i)$, $i = 1, \dots, M$ and the parameters are $\boldsymbol{\mu}_i \in \mathbb{R}^p$, $p = 4$. The goal here is to evaluate $\mathbf{s}(\boldsymbol{\mu}^*)$, that is, the pressure field for deformation parameters that are not in our initial set of parameters.

Different techniques can be employed to reach this task. Here we consider one of the most used techniques, the RBF interpolation. The RBF allows representing our field at unknown parameter $\boldsymbol{\mu}^*$. In particular, the unknown coefficients $\mathbf{a}^*(\boldsymbol{\mu})$ can be computed in the following way:

$$\mathbf{a}(\boldsymbol{\mu}^*) = \sum_{i=1}^{M} \omega_i \phi(\|\boldsymbol{\mu}^* - \boldsymbol{\mu}_i\|),$$

where $\phi(\|\boldsymbol{\mu}^* - \boldsymbol{\mu}_i\|)$ are the radial basis functions with center $\boldsymbol{\mu}_i$ and weight ω_i. The weights are found from the conditions:

$$\mathbf{a}(\boldsymbol{\mu}_j) = \sum_{i=1}^{M} \omega_i \phi(\|\boldsymbol{\mu}_j - \boldsymbol{\mu}_i\|), \quad j = 1, \dots, M.$$

In this work, radial basis functions of multiquadric shape, of expression $\phi(r) = \sqrt{1 + (\varepsilon r)^2}$, with $r = \|\boldsymbol{\mu}^* - \boldsymbol{\mu}_i\|$. Alternative approaches that can be used as *approximation part* of ROMs are the Gaussian Process Regression (GPR),[36] or the K-Neighbors Regression (KNR).

4.3 | Standard ROM: Full mesh

In the standard-ROM model we introduce, we consider as snapshots the pressure and wall shear stress fields evaluated on all the mesh points of the blades.

The question that automatically arises is: *What is the procedure to predict the efficiency of a propeller starting from the deformation parameters* $\boldsymbol{\mu}^*$?

The efficiency of a propeller can be predicted by performing the following steps:

1. deform the original blades according to the deformation parameters $\boldsymbol{\mu}^*$;

2. deform the blades' points using a RBF technique and triangulate the mesh;
3. compute the normal unitary vectors to the cells and their areas, that are necessary for the computation of thrust and torque forces;
4. exploit the *standard* ROM to predict the pressure and wall-shear stress fields on the blades;
5. compute the thrust and torque forces approximating the integrals in (6), and then the efficiency (following expression (7)).

Remark: We specify here that in step 2 a triangulation is applied on the deformed mesh to make straightforward the computation of the normals and areas to the cells.

The integrals 6 are approximated as discrete sums on all the mesh cells, as follows:

$$T_{\text{approx}} = \rho \sum_{c=1}^{N_{\text{dof}}} (p_c \mathbf{n}_c a_c + \boldsymbol{\Sigma}_c \mathbf{n}_c a_c);$$
$$Q_{\text{approx}} = \rho \sum_{c=1}^{N_{\text{dof}}} (p_c \mathbf{n}_c \times \mathbf{x}_c a_c + \boldsymbol{\Sigma}_c \mathbf{n}_c \times \mathbf{x}_c a_c), \tag{8}$$

where $\mathbf{x}_c$ is the position of the centre of cell c, the approximated reduced fields are $p_c = p_{\text{rom}}(\mathbf{x_c})$, $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}_{\text{rom}}(\mathbf{x_c})$, $\mathbf{n}_c$ and a_c are the normal to cell c and its area.

The critical issue of this analysis is the computational time required to go through steps 1–5. In particular, the most computationally expensive step is 2 and the total time to predict the efficiency of a deformed propeller is $\sim$ 6 min.

4.4 | Fast ROM: Quadrature points

We here consider the second type of ROM, named *fast* ROM. In this case, the snapshots are evaluated on the coordinates of a predefined number $N_{\text{quadrature}}$ of Gauss quadrature nodes on the blades.

The 3-dimensional quadrature points are found mapping the 2D cartesian Gauss-Legendre quadrature nodes into the UV local reference system on the NURBS surfaces, and then mapping the UV coordinates in a 3D cartesian domain. In particular, we considered as different NURBS the suction and pressure side of all blades.

Remark: The number of quadrature nodes on the blades has been selected after conducting a sensitivity study. This study is conducted on the predictions of the thrust forces and torque momentum. In particular, we measured the accuracy of the prediction made considering quadrature formulas with respect to the FOM value, for different degrees of quadrature nodes. The results are represented in Figure 12. We can notice that the accuracy converges on steady values for a degree ≥ 100. For this reason, we selected 100 quadrature nodes on each face of the blades.

In Figure 13, two representations of the Gauss quadrature nodes on the undeformed and deformed blades are displayed.

Therefore, there are in the offline stage two additional steps with respect to the standard ROM:

- the computation of the quadrature nodes on all the deformed blades in the dataset;
- the evaluation of the snapshots on the quadrature nodes through *interpolation* or *regression* techniques. Here we consider a K-Neighbor Regression technique based on 5 nearest neighbors, where the fields on the quadrature nodes are predicted by local interpolation of the targets associated with the nearest neighbors in the training set.

Moreover, a ROM is also exploited not only for the prediction of the pressure and wall shear stress fields but also for the computation of the normal vectors to the blades' surfaces.

We illustrate here the complete procedure to predict the efficiency of a deformed propeller with parameters $\boldsymbol{\mu}^*$:

1. deform the initial blades according to parameters, as in 1;
2. generate a number $N_{\text{quadrature}}$ of Gauss quadrature nodes on the deformed blades;
3. predict the normal vectors exploiting ROM and compute the jacobians of the deformation;
4. exploit the ROMs to predict the pressure and wall shear stress on the blades' quadrature points;
5. evaluate the torque and thrust forces, and consequently the efficiency.

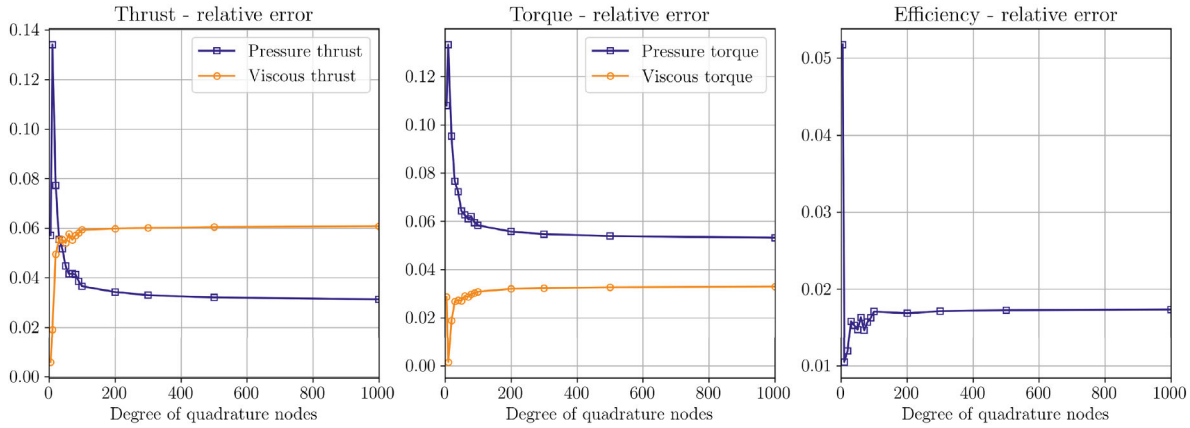
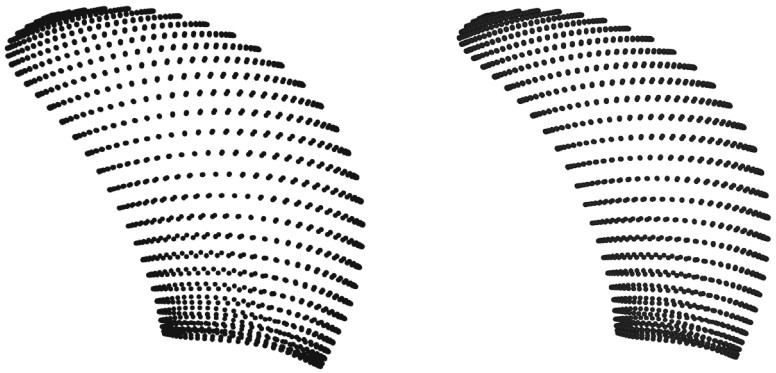


FIGURE 12 Accuracy of predicting forces, momentum and efficiency using quadrature formulas, for different number of quadrature samples. The accuracy is measured as a relative error with respect to the reference FOM value.



(A) Undeformed blades' quadrature nodes **(B)** Deformed blades' quadrature nodes

FIGURE 13 Quadrature nodes on undeformed and deformed blades. In this picture, $N_{\text{quadrature}} = 10,800$.

The technique here presented, which—to the best of our knowledge—is novel for this application, allows for a reduction in the computational time. Indeed, the efficiency evaluation for a single propeller takes approximately 8–15 s.[¶] The reason for this gain is that the mesh generation, that is, the expensive step 2 in standard ROM, is here replaced by step 2, which is much faster than the mesh generation.

The most important difference between the two approaches is the computation of the forces: in the standard case, we approximate the integrals in (6) with discrete sums on all the mesh cells; in the second case, the quadrature formulas on the Gauss nodes are considered to approximate the integrals. However, the use of integration formulas to compute the forces does not lead to a minor accuracy in the reconstruction of the efficiency.

4.5 | Results and comparison of ROMs

As specified in the previous paragraphs, the reduced order model can be built considering different techniques for the reduction and approximation stages. In this work, we always exploit the POD as a reduction method, since it is a consolidated technique in the field of industrial applications. On the other side, the approximation technique has been chosen as the result of an analysis of the accuracy of the resulting ROM. In particular, the RBF, GPR and KNR approximating techniques are here compared. As for the RBF, the thin plate spline is chosen as basis functions; in the KNR the regression is implemented considering $k = 5$ and uniform weights. The accuracy is here measured in terms of *k-fold cross validation* error. It is computed by splitting the database into k consecutive folds (we chose $k = 10$ in our case),

TABLE 1 Accuracy analysis for the **standard** ROM.

ROM field	RBF	GPR	KNR
Pressure	0.776%	0.632 %	9.651 %
Wall shear stress (x)	1.542 %	3.571 %	3.348 %
Wall shear stress (y)	3.293 %	6.305 %	5.310 %
Wall shear stress (z)	1.335 %	2.156 %	3.968 %

Note: The accuracy is measured in terms of the mean of the *k-fold cross validation* errors with $k = 10$.

TABLE 2 Accuracy analysis for the **fast** ROM.

ROM field	RBF	GPR	KNR
Pressure	2.673 %	5.857 %	12.117 %
Wall shear stress (x)	2.913 %	6.208 %	4.852 %
Wall shear stress (y)	6.252 %	12.233 %	7.489 %
Wall shear stress (z)	2.959 %	5.907 %	5.152 %
Normals (x)	4.688 %	11.312 %	4.984 %
Normals (y)	4.595 %	11.03 %	4.487 %
Normals (z)	4.511 %	10.837 %	4.888 %

Note: The accuracy is measured in terms of the mean of the *k-fold cross validation* errors with $k = 10$.

and each fold is used once as validation while the $k - 1$ remaining folds form the training set. We remind that the database is composed of M = 216 snapshots, corresponding to the initially deformed blades (details about the sampling are provided in 2.1.

The mean values of the *k-fold* errors for the standard and fast reduced order models are reported in Tables 1 and 2. The results show that the GPR is the best technique for the standard approach, whereas the RBF provides the best results for the fast ROM. The best methods are chosen for the efficiency prediction in the shape optimization in order to obtain good results in the FOM-ROM validation.

Figure 14 displays a comparison between the full-order fields acting on the blade and the reconstructed reduced-order fields. In the case of the fast ROM approach (Figure 14C,F), the fields are computed and represented on the Gauss quadrature nodes. The figure shows a good agreement between the reconstructed fields and their original high-fidelity counterpart.

5 | OPTIMIZATION PROCESS

This section is dedicated to the shape optimization process. In particular, Section 5.1 briefly recalls the theory on the genetic algorithm, here exploited for optimization, and Section 5.2 is dedicated to a comparison among the results obtained from the genetic optimization and those obtained using a gradient-based approach.

5.1 | Genetic optimization

This subsection is dedicated to the explanation of the genetic algorithm (GA) used for the optimization step. It was introduced for the first time in Reference 37 and it is inspired by the evolution mechanism.

The general scheme of how it works is displayed in Figure 15. In particular, it starts from an initial population of N_{pop} individuals and the *fitness*, that is, the objective function we want to minimize/maximize, is evaluated for all the individuals. Then, the algorithm iteratively performs three steps: *selection* of the best individuals, *crossover*, recombination of the genes of the individuals, and *mutation* of the genes, allowing the evaluation of new individuals.

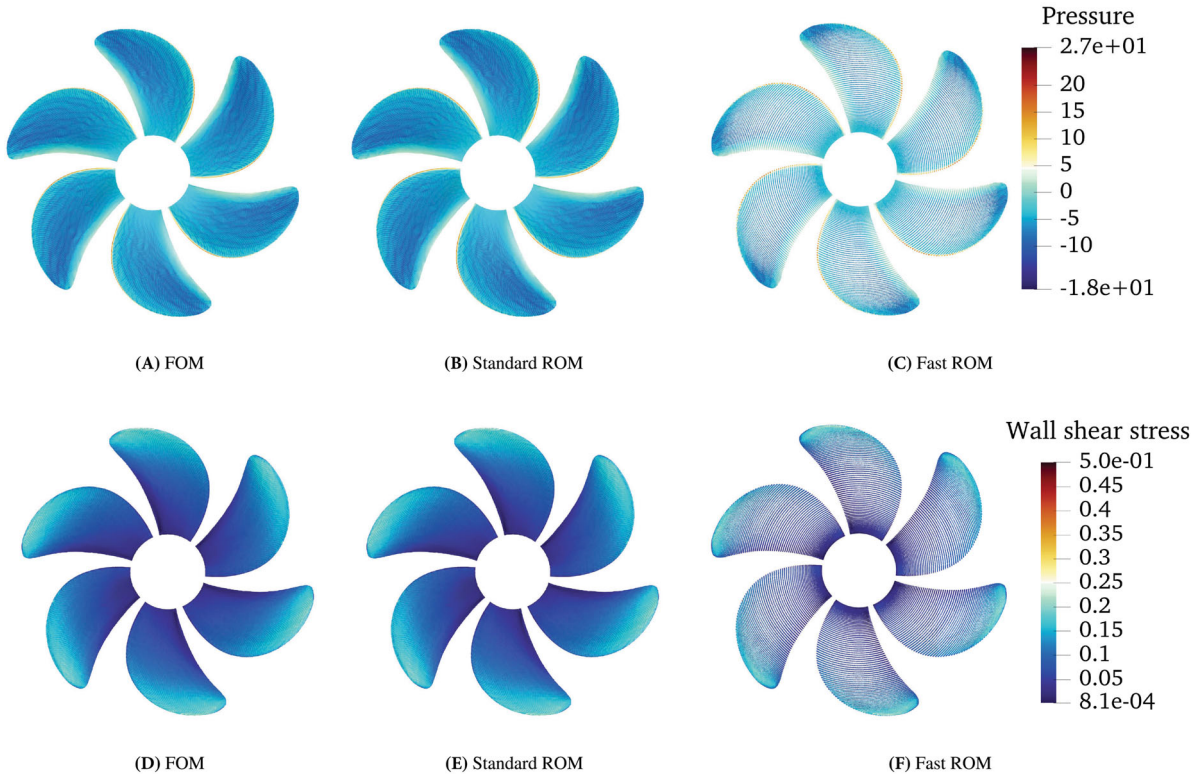


FIGURE 14 High-fidelity and reduced order fields corresponding to the initial blade: relative pressure in the first row and wall shear stress magnitude in the second row.

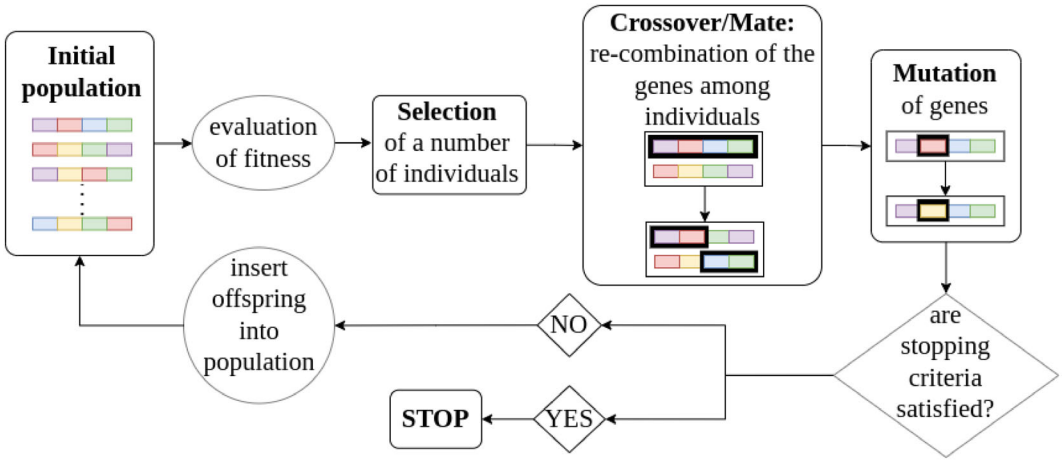


FIGURE 15 Schematic representation of the genetic algorithm.

The GA is computed in Python making use of the *Deap* evolutionary computational framework [38,39] and the following hyper-parameters are considered:

- the initial population is composed by 30 and 150 individuals for the standard and fast ROMs, respectively;
- in the *mate* stage, a one-point crossover is performed, that modifies in-place the input individuals;
- in the *mutation* step, a Gaussian mutation of mean $\mu = 1$ and standard deviation $\sigma = 0.1$ is applied to the input individual. The independent probability for each attribute to be mutated is set to 0.5 and 0.8, when the standard and fast ROMs are exploited, respectively;

- the *evolutionary algorithm* employed in the genetic process is the $(\mu + \lambda)$ algorithm, which selects the μ best individuals for the next generation ($\mu = 5$ for the standard ROM, 50 in the fast approach) and produces λ children at each generation ($\lambda = 10$ in the standard approach, 80 in the fast one). The probabilities that an offspring is produced by crossover and by mutation are 0.4 and 0.5, respectively.
- the previous steps are repeated for 10 and 20 generations for the standard and fast cases, respectively.

Remark: In this article, we chose the genetic algorithm for optimization instead of a gradient-based algorithm since it allows us to evaluate a larger number of individuals, in our case of deformed shapes, and avoids getting stuck into local minima.

In our particular test case, the individuals are the parameters, so each individual $\mathbf{i}$ has 4 genes, corresponding to the deformation rates of the geometrical parameters. The fitness function $f(\mathbf{i})$ we want to maximize is the efficiency. We consider two different optimization processes:

- basic optimization, where the efficiency $\eta_{\text{propeller}}$ is the fitness;
- constrained optimization, where a series of geometrical and physical constraints applies to the optimization process. In this case, we consider a penalized fitness function:

$$f(\mathbf{i}) = \eta_{\text{propeller}}(\mathbf{i}) - \sum_{j=1}^{N_{\text{constraints}}} w_j \text{ penalty } (\mathbf{i})_j, \tag{9}$$

where penalty $(\mathbf{i})_j$ and w_j are the penalty and the weight associated with the jth constraint, $N_{\text{constraints}}$ is the number of constraints taken into account.

We specify here only the generic formulation to apply constraints in the optimization procedure since the constraints imposed in this work are protected by nondisclosure agreements. It derives that a punctual discussion about the geometrical features of the optimal blade cannot be pursued; we consider the constrained optimization as an additional test with a different objective function, limiting our considerations to the precision comparison between ROM and FOM.

5.1.1 | Results of genetic optimization

Table 3 displays the best individual obtained from the genetic algorithm (GA) in the standard and fast ROM approaches.

In particular, we measure the results of the optimization processes in terms of absolute increases of the percentage efficiency with respect to the starting propeller, that is, the one corresponding to parameter $\mu = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$. Table 3 reports the following metrics:

$$\Delta_{\text{FOM}} = \eta_{\text{FOM}}^{\text{opt}} - \eta_{\text{FOM}}^{\text{start}}, \quad \Delta_{\text{ROM}} = \eta_{\text{ROM}}^{\text{opt}} - \eta_{\text{ROM}}^{\text{start}},$$

where $\eta_{\text{FOM}}^{\text{start}}$ and $\eta_{\text{ROM}}^{\text{start}}$ are the FOM and ROM efficiencies of the starting propeller, whereas $\eta_{\text{FOM}}^{\text{opt}}$ and $\eta_{\text{ROM}}^{\text{opt}}$ are the efficiencies for the optimal propeller. Moreover, we underline here that the efficiency of the starting propeller is captured with an absolute error of 0.01% in the standard ROM approach and of 0.33% in the fast ROM approach.

TABLE 3 Results of genetic optimization for the unconstrained and constrained cases, for both the standard and fast ROMs.

		Parameters					
	Optimization	**Def. pitch**	**Def. camber**	**Def. chord length**	**Def. thickness**	Δ_{ROM}	Δ_{FOM}
Standard	Unconstrained	0.92	1.03	0.70	0.75	+5.15 %	+5.13 %
	Constrained	1.01	0.81	0.82	0.99	+0.79 %	+0.81 %
Fast	Unconstrained	0.91	0.89	0.74	0.70	+3.76 %	+3.24 %
	Constrained	0.99	0.90	0.77	1.02	+1.16 %	+0.80 %

Note: Δ_{ROM} and Δ_{FOM} are the efficiency improvements with respect to the starting propeller of the optimization processes.

Table 3 shows that the standard and fast optimization processes lead to similar values of the final optimal efficiency, although the algorithms converge to different results in terms of individuals in the parameter space.

In general, the optimized propeller obtained with an unconstrained algorithm is characterized by higher efficiency with respect to the one resulting from the constrained process. Indeed, as can be seen from the Table, the unconstrained optimization processes lead to blades with small thickness, since it would result in lower torque values, and hence to more efficient propellers. On the other hand, thin blades can reduce the robustness of the propeller's structure and, hence, make the propeller more vulnerable to external stress. This is the reason why geometrical constraints (on the chord length and on the thickness, for example) are required by industries to ensure the robustness of the propeller's mechanical structure.

For what concerns the accuracy in the prediction of the efficiency, Table 3, together with Tables 1 and 2, show that the standard technique is more accurate, with an error behind the 0.1% on the efficiency value. However, the fast algorithm allows for reaching parametric points with comparable full-order efficiencies in a significantly reduced amount of time. Indeed, as already pointed out in Sections 4.3 and 4.4, the evaluation of the efficiency for a single individual lasts about 6 min exploiting the standard ROM, while it lasts less than 15 s exploiting the fast approach.

Another consideration that can be withdrawn is that the two techniques do not converge to the same optimal parametric points. The reason for this fact is that the objective function is computed following a different approximated procedure. The same holds true for the penalty terms related to forces' constraints in the constrained optimization. The optimal blades are graphically represented in Figure 16 and compared in shape with the original starting blade of the pipeline (colored in gray).

5.2 | A comparison with gradient-based algorithms

In this article, the authors chose to adopt a genetic algorithm for the optimization process, since it allows the evaluation of a large number of individuals, increasing the probability that the global optimum is reached. On the other hand, gradient-based algorithms suffer from high sensitivity to the initial guess of the optimization and this can lead the algorithm to freeze into the closest local optimum. Moreover, the genetic process is characterized by a large number hyper-parameters, specified in Section 5.1, and changing those parameters allowed us to do a large number of optimization experiments and to choose the values resulting in the highest objective function.

Here we show different experiments done using a gradient-based algorithm varying the initial guess of the optimization process. This part focuses on the results obtained using two different techniques: the *conjugate gradient* (*CG*) method, first described in Reference 40, and the *L-BFGS-B* method, namely the *Limited-Memory* version of the bound-constrained *BFGS* (*Broyden–Fletcher–Goldfarb–Shanno* algorithm, treated in References 41–43.

In this subsection, we focus on the **constrained fast optimization** and made a comparison between the results obtained with the above-cited gradient-based methods and the genetic algorithm. The results are reported in Table 4 for

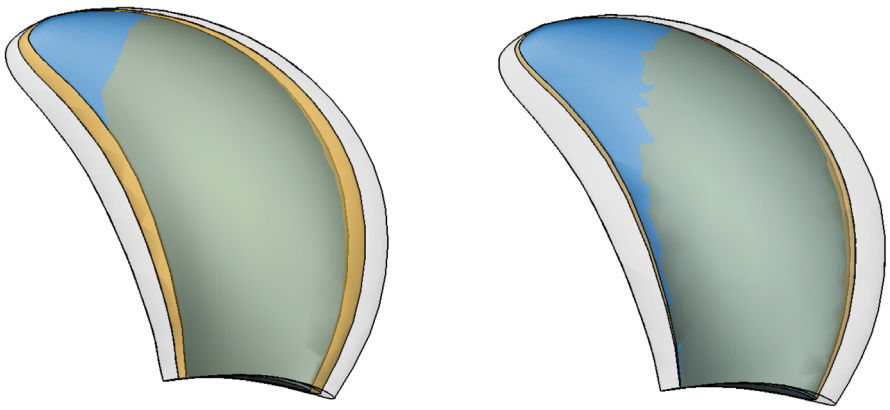


FIGURE 16 Graphical representation of the final optimal shapes obtained from the standard (A) and the fast (B) optimization processes. The figure displays the starting blade in gray, the optimal blade from the unconstrained optimization process in light blue, and the optimal blade from the constrained optimization in orange.

TABLE 4 Results of gradient optimization for different initial guesses for the CG and L-BFGS-B methods.

Test	Method	Initial guess	Func. evals	Grad. evals	Δ_{ROM}	Final point
1a	CG	$[1\ \ 1\ \ 1\ \ 1]$	160	30	+0%	$[1\ \ 1\ \ 1\ \ 1]$
1b	L-BFGS-B		190	38		
2a	CG	$[0.9\ \ 1.1\ \ 0.9\ \ 1.1]$	255	49	−0.4%	$[0.9\ \ 1.1\ \ 0.9\ \ 1.1]$
2b	L-BFGS-B		305	61		
3a	CG	$[0.99\ \ 0.90\ \ 0.77\ \ 1.02]$	247	48	+1.1%	$[0.99\ \ 0.90\ \ 0.77\ \ 1.02]$
3b	L-BFGS-B		120	24		
4a	CG	$[0.9\ \ 0.8\ \ 0.8\ \ 0.9]$	151	28	−32.6%	$[0.9\ \ 0.8\ \ 0.76\ \ 0.9]$
4b	L-BFGS-B		215	43	−0.1%	$[1\ \ 1\ \ 0.75\ \ 1.15]$

Note: The results are reported in terms of number of function and gradient evaluations, initial and final points of the optimization, and fitness improvement with respect to the starting propeller.

different starting points and for the CG and L-BFGS-B approaches. We specify that the fitness increments Δ_{ROM} are always measured with respect to the initial propeller, corresponding to parameters $[1\ \ 1\ \ 1\ \ 1]$, and do not depend on the starting guess of each optimization process.

The gradient-based approaches are stuck in the initial guess in most of the cases here analyzed. In particular, in Tests 3a and 3b, where the starting point is the optimum obtained from the GA, both the gradient-based approaches do not improve the results obtained with the genetic. Only for Test 4b the gradient method converges to a point that does not coincide with the initial guess, but it does not improve the genetic results. Moreover, it reaches an efficiency value lower than the efficiency of the starting propeller of our pipeline. We specify also that the values of the fitness reported in Table 4 correspond to the *penalized* efficiency of the propeller, that is, the one expressed in (9).

Remark: It is important to highlight that the genetic algorithm is characterized by a larger number of function evaluations with respect to gradient-based algorithms, resulting in a bigger computational time. The tuning of the hyper-parameters, specified in Section 5.1, leads to a number of function evaluations equal to 180 and 2750, for the optimizations employing the *standard* and *fast* ROM, respectively. Considering the fact that a function evaluation takes the same amount of time for gradient-based and genetic algorithms, the genetic computational time is *one order of magnitude bigger* than the gradient-based optimization. In fact, Table 4 shows that, in the constrained and *fast* ROM optimization, the maximum number of function evaluations is 305. However, the gradient-based method produces results (in Table 4) that are often stuck in local minima and never approach the results of the genetic algorithm (reported in Table 3). Therefore, we can finally say that the genetic algorithm provides a robust and efficient method, that has been validated in many previous industrial shape optimization projects, such as References 16,17,44.

6 | CONCLUSIONS

The manuscript has illustrated a shape optimization pipeline exploiting data-driven ROM for improving the efficiency. The complexity of the here-discussed application, both in the geometry of the propeller and in the formulation of the original model, highlights the modularity and generality of the ROM based on data.

The parameterization of the blade, discussed in Section 2, has been propagated to the mesh nodes, obtaining a series of discretized spaces that share the same topology, allowing us to apply the POD technique. As presented in Section 2.3, the deformation applied to the mesh nodes maintains an overall quality similar to the original mesh and avoids reconstructing a new mesh for any new deformation. It must be said that due to the RBF technique applied in this phase, the computational cost of the mesh deformation is not negligible, but lower than the one required for constructing the mesh from scratch. This possible computational bottleneck will be however better addressed in future works in order to mitigate its impact over the entire pipeline.

A problem related to the computational burden of the mesh generation also arises in the online stage, when using a *standard ROM* approach, as discussed in Section 4.1. One possible solution—that resolves the problem only at the online stage—is explored here with the *Fast ROM*, introduced in Section 4.2. The data-driven model is indeed built over

the quadrature nodes of the blade faces, avoiding the deformation of the mesh since the objective function to minimize depends only on integral quantities over the blade surface.

As the reader can see in Section 4.3, the ROMs demonstrate in this work great precision, even thanks to the large offline simulation campaign performed to collect the initial high-fidelity snapshots. The computational cost required by the creation of this database is of course not negligible but has allowed a big reduction during the optimizations, allowing to test several methods and for the hyper-parameters tuning thanks to the huge gain in simulation time (24 h $= 8.64 \times 10^4$ s for the FOM against 15 s for the Fast-ROM, more than 5000 times faster). Future works will perform anyway the sensitivity analysis at varying the dimension of the solutions database, trying to provide guidelines for increasing accuracy in fixed computational budget contexts.

The efficiency of the reduced models allowed for testing different optimization procedures in Sections 5, comparing in this case the employment of methods based on the gradient of the objective functions in Section 5.2 and methods based on genetic strategies in Section 5.1. Even if the gradient-based approach is the most rigorous one, its employment over a nonconvex manifold (here computed by the RBF technique) produces poor results in terms of performance and shows its sensitivity to the selected starting point. The genetic optimization, requiring a larger number of evaluations, is able instead to produce better shapes in all the tests we performed.

AUTHOR CONTRIBUTIONS

Conceptualization: Anna Ivagnes, Nicola Demo; *Methodology*: Anna Ivagnes, Nicola Demo; *Formal analysis and investigation*: Anna Ivagnes; *Writing–original draft preparation*: Anna Ivagnes; *Writing–review and editing*: Nicola Demo; *Funding acquisition*: Gianluigi Rozza; *Supervision*: Gianluigi Rozza.

ACKNOWLEDGMENTS

We sincerely thank the industrial partners' Ing. Gianluca Gustin, Ing. Gianpiero Lavini, Ing. Edoardo Tagliamonte and Ing. Nicola Iona (from FINCANTIERI S.P.A.). We also thank Francesco Andreuzzi and Gianmarco Gurioli for their preliminary work on this project. This work was partially funded by INdAM-GNCS 2020–2021 projects, by European High-Performance Computing Joint Undertaking project Eflows4HPC GA N. 955558, by PRIN "Numerical Analysis for Full and Reduced Order Methods for Partial Differential Equations" (NA-FROM-PDEs) project by European Union Funding for Research and Innovation—Horizon 2020 Program—in the framework of European Research Council Executive Agency: H2020 ERC CoG 2015 AROMA-CFD project 681447 "Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics" P.I. Professor Gianluigi Rozza.

CONFLICT OF INTEREST STATEMENT

No potential competing interest was reported by the authors.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the authors, Anna Ivagnes, Nicola Demo, and Gianluigi Rozza, upon reasonable request with the permission of Fincantieri S.p.A.

ENDNOTES

*The generality of the pipeline allows to select different and more parameters.

†For nondisclosure agreements.

‡The mesh deformation is performed using one processor core only on SISSA HPC cluster Ulysses (200 TFLOPS, 2TB RAM, 7000 cores).

§The FOM simulations are performed using 55 processor cores on SISSA HPC cluster Ulysses (200 TFLOPS, 2TB RAM, 7000 cores).

¶Both the ROMs are performed on an Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz 16GB RAM on only one processor core.

ORCID

Anna Ivagnes https://orcid.org/0000-0002-2369-4493
Nicola Demo https://orcid.org/0000-0003-3107-9738
Gianluigi Rozza https://orcid.org/0000-0002-0810-8812

REFERENCES

1. Carlton J. *Marine Propellers and Propulsion*. 4th ed. Elsevier; 2019.

2. Kerwin JE. Marine propellers. *Annu Rev Fluid Mech*. 1986;18(1):367-403.
3. Baltazar J, Falcão de Campos JAC, Bosschers J. Open-water thrust and torque predictions of a ducted propeller system with a panel method. *Int J Rotat Machine*. 2012;2012.
4. Boswell RJ. *Design, cavitation performance, and open-water performance of a series of research skewed propellers*. tech. rep., David W Taylor Naval Ship Research and Development Center; 1971.
5. Bhattacharyya A, Krasilnikov V, Steen S. Scale effects on open water characteristics of a controllable pitch propeller working within different duct designs. *Ocean Eng*. 2016;112:226-242.
6. Muscari R, Di Mascio A, Verzicco R. Modeling of vortex dynamics in the wake of a marine propeller. *Comput Fluids*. 2013;73:65-79.
7. Zhang Q, Jaiman RK. Numerical analysis on the wake dynamics of a ducted propeller. *Ocean Eng*. 2019;171:202-224.
8. Salmoiraghi F, Scardigli A, Telib H, Rozza G. Free-form deformation, mesh morphing and reduced-order methods: enablers for efficient aerodynamic shape optimisation. *Int J Computat Fluid Dyn*. 2018;32(4-5):233-247. doi:10.1080/10618562.2018.1514115
9. Quarteroni A, Manzoni A, Negri F. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. 1st ed. Springer; 2015.
10. Hesthaven JS, Rozza G, Stamm B. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer Briefs in Mathematics. 1st ed. Springer; 2015.
11. Rozza G, Hess M, Stabile G, Tezzele M, Ballarin F. Basic ideas and tools for projection-based model reduction of parametric partial differential equations. In: Benner P, Grivet-Talocia S, Quarteroni A, Rozza G, Schilders WHA, Silveira LM, eds. *Model Order Reduction. 2*. De Gruyter; 2020:1-47.
12. Rozza G, Stabile G, Ballarin F, eds. *Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics*. Society for Industrial and Applied Mathematics; 2022.
13. Benner P, Grivet-Talocia S, Quarteroni A, Rozza G, Schilders W, Silveira LM. *System-and Data-Driven Methods and Algorithms*. De Gruyter; 2021.
14. Benner P, Schilders W, Grivet-Talocia S, Quarteroni A, Rozza G, Miguel SL. *Model Order Reduction: Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter; 2020.
15. Benner P, Schilders W, Grivet-Talocia S, Quarteroni A, Rozza G, Miguel SL. *Model order reduction: Volume 3: Applications*. De Gruyter; 2020.
16. Demo N, Tezzele M, Mola A, Rozza G. Hull shape design optimization with parameter space and model reductions, and self-learning mesh morphing. *J Marine Sci Eng*. 2021;9(2):185.
17. Demo N, Ortali G, Gustin G, Rozza G, Lavini G. An efficient computational framework for naval shape design and optimization problems by means of data-driven reduced order modeling techniques. *Bollettino dell'Unione Matematica Italiana*. 2020;14:211-230. doi:10.1007/s40574-020-00263-4
18. Tezzele M, Salmoiraghi F, Mola A, Rozza G. Dimension reduction in heterogeneous parametric spaces with application to naval engineering shape design problems. *Adv Model Simulat Eng Sci*. 2018;5(1):1-19.
19. Tezzele M, Demo N, Stabile G, Mola A, Rozza G. Enhancing CFD predictions in shape design problems by model and parameter space reduction. *Advanced Modeling and Simulation. Eng Sci*. 2020;7(40). doi:10.1186/s40323-020-00177-y
20. Tezzele M, Demo N, Mola A, Rozza G. An integrated data-driven computational pipeline with model order reduction for industrial and applied mathematics. *Novel Mathematics Inspired by Industrial Challenges*. Springer; 2022:179-200.
21. Tezzele M, Fabris L, Sidari M, Sicchiero M, Rozza G. A multifidelity approach coupling parameter space reduction and nonintrusive POD with application to structural optimization of passenger ship hulls. *Int J Numer Methods Eng*. 2023;124(5):1193-1210.
22. D'Agostino D, Serani A, Diez M. On the combined effect of design-space dimensionality reduction and optimization methods on shape optimization efficiency. *Multidisciplinary Analysis and Optimization Conference*. Aerospace Research Center. American Institute of Aeronautics and Astronautics; 2018.
23. Çelik C, Danişman DB, Khan S, Kaklis P. A reduced order data-driven method for resistance prediction and shape optimization of hull vane. *Ocean Eng*. 2021;235:109406.
24. Serani A, Diez M, Wackers J, Visonneau M, Stern F. Stochastic shape optimization via design-space augmented dimensionality reduction and rans computations. *AIAA SciTech 2019 Forum*. Aerospace Research Center; 2019:2218.
25. Gadalla M, Tezzele M, Mola A, Rozza G. BladeX: Python Blade Morphing. *J Open Source Software*. 2019;4(34):1203. doi:10.21105/joss.01203
26. BladeX on Github. https://github.com/mathLab/BladeX.
27. Piegl LA, Richard AM. Tessellating trimmed NURBS surfaces. *Comput Aided Des*. 1995;27(1):16-26.
28. Piegl L, Tiller W. *The NURBS book*. Springer Science & Business Media; 1996.
29. OpenFOAM website. https://openfoam.org/.
30. Moukalled F, Mangani L, Darwish M. *The Finite Volume Method in Computational Fluid Dynamics*. Vol 113. Springer; 2016.
31. Reynolds O IV. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philos Trans R Soc Lond*. 1895;186:123-164. doi:10.1098/rsta.1895.0004
32. Menter FR. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA J*. 1994;32(8):1598-1605. doi:10.2514/3.12149
33. Langtry RB, Menter FR. Correlation-based transition modeling for unstructured parallelized computational fluid dynamics codes. *AIAA J*. 2009;47(12):2894-2906. doi:10.2514/1.42362
34. Demo N, Tezzele M, Rozza G. EZyRB: Easy reduced basis method. *J Open Source Software*. 2018;3(24):661. doi:10.21105/joss.00661
35. EZyRB Github package. https://github.com/mathLab/EZyRB.
36. Williams CK, Rasmussen CE. *Gaussian Processes for Machine Learning*. MIT press; 2006.

37. Holland JH. Genetic algorithms and the optimal allocation of trials. *SIAM J Comput*. 1973;2(2):88-105. doi:10.1137/0202009
38. De Rainville FM, Fortin FA, Gardner MA, Parizeau M, Gagné C. Deap: A python framework for evolutionary algorithms. *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. Aerospace Research Center; 2012:85-92.
39. Fortin FA, De Rainville FM, Gardner MAG, Parizeau M, Gagné C. DEAP: Evolutionary algorithms made easy. *J Mach Learn Res*. 2012;13(1):2171-2175.
40. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving. *J Res Natl Bur Stand*. 1952;49(6):409.
41. Broyden CG. The convergence of a class of double-rank minimization algorithms 1. General considerations. *IMA J Appl Math*. 1970;6(1):76-90.
42. Robitaille B, Marcos B, Veillette M, Payre G. Quasi-Newton methods for training neural networks. *WIT Trans Informat Commun Technol*. 1970;2.
43. Byrd RH, Lu P, Nocedal J, Zhu C. A limited memory algorithm for bound constrained optimization. *SIAM J Sci Comput*. 1995;16(5):1190-1208.
44. Demo N, Tezzele M, Rozza G. A supervised learning approach involving active subspaces for an efficient genetic algorithm in high-dimensional optimization problems. *SIAM J Sci Comput*. 2021;43(3):B831-B853.

How to cite this article: Ivagnes A, Demo N, Rozza G. A shape optimization pipeline for marine propellers by means of reduced order modeling techniques. *Int J Numer Methods Eng*. 2024;125(7):e7426. doi: 10.1002/nme.7426