Dipartimento di Scienze Matematiche "Giuseppe Luigi Lagrange"
Corso di Laurea in Ingegneria Matematica, Indirizzo Modelli Matematici e Simulazioni Numeriche

Tesi di Laurea Magistrale

A Data-Driven Reduced Order Optimization Approach for Cruise Ship Design

Advisors:

Prof. Claudio Canuto

Prof. Gianluigi Rozza

Co-advisor:

Dr. Nicola Demo

Industrial co-advisors:

Dr. Gianluca Gustin

Dr. Gianpiero Lavini

Candidate

Giulio Ortali

Academic year 2018/2019

Abstract

This Master Thesis describes the implementation of a data-driven shape optimization pipeline in a naval architecture application. We investigate the use of Reduced Order Methods (ROMs) in order to improve the efficiency and applicability of the techniques, also in an industrial setting.

The above-mentioned pipeline is applied to a realistic cruise ship in order to reduce the total drag applied. We begin by defining the design space, generated by deforming an initial unoptimized shape in a parametric way using Free Form Deformation (FFD). The evaluation of the performance of each new hull is determined by simulating the flux via finite volume discretization of a biphase (water and air) fluid. In order to improve the efficiency of the simulation over a new shape, we use ROMs, in particular Proper Orthogonal Decomposition with Interpolation (PODI). Finally, a genetic optimization algorithm is used to explore the design space, using PODI for efficient evaluation of the flows.

Other applications of ROMs are considered apart from the efficient resolution of the numerical problem, which are the approximation of dynamical systems via Dynamic Mode Decomposition (DMD) and reduction of the dimensionality of the parameter space.

Keywords— finite volume method, computational fluid dynamics, reduced order modeling, free form deformation, shape optimization

Contents

Acknowledgments

I would first like to thank my thesis advisor prof. Claudio Canuto of Politecnico di Torino for giving me the opportunity to work on this project and for his willingness to hear my doubts and problems. A very special thanks goes also to prof. Gianluigi Rozza of Scuola Internazionale Superiore di Studi Avanzati (SISSA) for the infinite helpfulness he has shown during the period I passed in SISSA and for having stimulated me and believed in my capacities.

A huge thank you goes also to Dr. Nicola Demo of SISSA, for being always available to listen to my doubts and solve my problems, and to all the mathLab group in SISSA, from associate researchers to PhD students and other master thesis students, for making me feel at home in the 4 months I passed in Trieste.

I would also like to thank the naval architecture office of Fincantieri in Trieste for welcoming me and having guided me in approaching a new discipline. In particular, I would like to thank Dr. Gianluca Gustin and Dr. Gianpiero Lavini for the precious and helpful hints they gave me during the work.

Finally, but most importantly, I want to express my very profound gratitude to my parents, for giving me the opportunity to pursue this goal, and to my girlfriend and friends, for providing me with support and continuous encouragement throughout my years of study.

Giulio Ortali.

List of Figures

List of Tables

Introduction

In this Master Thesis we investigate the role of Reduced Order Methods (ROMs) for the efficient resolution of a shape optimization problem applied to naval architecture. The object of the optimization is an already built and sailing cruise ship. The aim is to test the applicability of the algorithm and, in case of positive results, employ it in the future for the design of new ships.

This work has been done in collaboration with *Fincantieri S.p.A* and *Scuola Internazionale di Studi Superiori Avanzati (SISSA)*, in the context of Horizon 2020 ERC Consolidator Grants, inside the Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics (AROMA-CFD) framework, and originated as a continuation and extension of previous works, such as for example [1], [2] and [3].

A shape optimization problem consists of finding the optimal geometric configuration of an object in terms of some performance to be maximized and some constraints to be fulfilled. In particular, we design a complete shape optimization pipeline that, given an initial guess for the shape of the hull, returns an optimal deformation of it. We measure performance in terms of total resistance applied to the ship, or drag, and we apply some constraints on the possible deformations to be explored in the optimization process.

The generality in which we have defined shape optimization problems entails a vast field of applicability, ranging from basic and applied sciences to engineering and industrial applications. A challenging feature of this class of problems is that, in order to tackle it, one usually needs to make use of different tools coming from different branches of sciences and mathematics.

In fact, the basic aspects that need to be considered in resolving this class of problems are:

1. the geometrical description of the object of interest and the implementation of a technique able to generate the different designs to be tested, also considering the possible constraints imposed;

2. the evaluation of the performance of the newly generated shapes, that in our case, for example, consists of solving a fluid dynamics problem

by means of numerical simulation;

3. the exploration of the design space using an optimization algorithm, returning as a final output the solution of our problem.

All the steps mentioned are critical and require particular caution in order to establish which is the best methodology to use and how to implement it in a way that is more suited for the particular problem at hand. In this work, some possible choices are implemented and tested in order to select the most efficient configuration.

In choosing how to perform the steps declared above, an aspect needs not to be overlooked: the computational time required. This point is crucial in industrial applications, where the computation of a coarse numerical solution is often preferred to a more accurate, but more time-consuming, one.

Between the three steps mentioned above, the aspect of computational parsimony is particularly important in the second one. This is because the numerical simulation of an industrial fluid dynamics problem is, in most cases, a very complex task, requiring a heavy computational load in order to be performed. For this reason, a lot of efforts has been made both by the Computational Fluid Dynamics (CFD) community, formulating more sophisticated techniques to tackle the problem, and by the High Performance Computing (HPC) community, in order to speed up the computations by means of advanced serial and parallel programming techniques.

This is the framework in which ROMs can play an important role. Reduced order methods are a large and diverse class of techniques developed from different communities of research for different kinds of applications. The common factor between these methodologies is the purpose of reducing the complexity of a mathematical model and its numerical approximation.

The main application of ROMs is in the context of parametric Partial Differential Equations (PDEs). A parametric PDE is a partial differential equation in which one or more parameters are introduced in the formulation, making the solution of the problem parameter-dependent. In particular, the parameter introduced in our case is related, in a way that will be more clear in Chapter 2, to the geometrical shape of the vessel. If the resolution of the numerical problem associated to the PDE (that will be denoted as Full Order Model (FOM)) for a particular value of the parameter is particularly expensive, as it is in our case, and if this solution needs to be computed for different values of the parameter, the computational time required becomes inadmissible, especially from an industrial point of view. Using ROMs we can build a reduced version of the FOM (that will be denoted as Reduced Order Model (ROM)) that returns an approximate solution in a very short time.

Apart from the parametric PDEs application, ROMs can also be applied to other contexts. In this work, we consider an application to dynamical systems, using a technique called Dynamic Mode Decomposition (DMD), and an application to the reduction of the dimensionality of the parameter space.

The structure of this work is the following:

- Chapter 1 (Reduced Order Methods): In this chapter, we introduce the reader to ROMs, giving some motivations and some basic ideas. We present here the methodologies that we use in this work, that is Proper Orthogonal Decomposition (POD), with the Proper Orthogonal Decomposition with Interpolation (PODI) variant, and Dynamic Mode Decomposition (DMD).

- Chapter 2 (Geometrical Shape Parametrization): In this chapter, we present the case study. After that, we introduce Free Form Deformation (FFD), a technique used to obtain parametrical geometrical deformations of the unoptimized shape and, finally, a possible reduction of the parameter space that employs POD.

- Chapter 3 (Full Order Model): In this chapter, we present the FOM for the simulation of the physical problem. We discuss here the continuous model (Navier-Stokes equations with some modifications) and the discretization technique (Finite Volume Method).

- Chapter 4 (Numerical Results): In this chapter, we show how the different ingredients we have presented in the previous chapters perform. Finally, we introduce the optimization algorithm and present the final results obtained.

Trieste and Torino, September 2019.

Chapter 1

Reduced Order Methods: Theory and Applications

Reduced Order Methods (ROMs) are a broad category of methods and techniques used to reduce the complexity of a mathematical model and possibly the computational cost required to obtain its solution numerically. The contexts in which this reduction is convenient are numerous in engineering and basic or applied sciences. This chapter intends to give some examples of such cases, to discuss some important issues related to ROMs and, finally, to introduce from a mathematical point of view the methodologies used in the following chapters.

We begin with some motivations for ROMs and some examples of applications. After that, we discuss in more detail Reduced Basis ROMs and the technique that will be used extensively during this work, i.e. Proper Orthogonal Decomposition (POD), with the PODI (POD with interpolation) variant. For this purpose, we will briefly recall the theory of Singular Value Decomposition (SVD). Finally, we discuss Dynamic Mode Decomposition (DMD) as a tool for reduction in the case of time-dependent problems.

1.1 Motivations and Applications of ROMs

A possible reduction of the computational complexity operated in most Computational Fluid Dynamics (CFD) problems, in particular in the ones related to high Reynolds numbers, is done considering an averaged version of the Navier-Stokes equations using an approach called Reynolds-Averaged Navier-Stokes Equations (RANS) (see Section 3.4 or [4]). Another possible simplification considers a version of the Navier-Stokes equations in which viscosity effects are neglected, generating the potential equations (see for

example [5]). These techniques, that are widely used and known since more than one century, can be seen in our context as model reduction techniques, in which some physical or mathematical simplification is adopted in order to reduce the computational complexity of the problem at hand.

Our attention, however, is directed to a particular class of ROMs, i.e. Reduced Basis ROMs (RB-ROMs), ore more precisely Proper Orthogonal Decomposition ROMs (POD-ROMs). An introduction to this class of methods will be given in Section 1.2, while for a more comprehensive treatment of RB-ROMs and in general of the topics we will discuss in this chapter, we refer for example to [6] or [7].

The main application of ROMs we will consider in this chapter is the efficient solution of parametric Partial Differential Equations (PDEs). A parametric PDE is a Partial Differential Equation in which one or more parameters are introduced either to account for uncertainty or to apply some control/optimization to the physical or geometrical setting of the problem. The contexts in which parametric PDEs are useful are countless, going from solid and fluid mechanics, acoustics, electromagnetism, and problems in finance. In particular, our attention is, because of the problem we are considering, directed to CFD and, consequently, to Navier-Stokes equations.

An issue of primarily importance in the numerical resolution of PDEs is the ability to simulate with a certain accuracy the problem at hand. However, in general, a compromise between the precision of the numerical solution and the computational efforts required is mandatory, since many problems encountered are very complex and a precise solution of them would be prohibitive. This is for example the case of CFD problems with industrial applications, like the one we are considering in this work, in which very complex geometrical configurations and high velocities make the completely faithful resolution of the equations impracticable.

This is particularly true in the context of parametric PDEs since usually we want to explore the parameter space and we need to solve the PDE for different parameter points. A reduction of the computational complexity is therefore essential, allowing the resolution of a multi-queries paradigm (for example, an optimization process) in a reasonable time, with obvious implications on the interest of those methods in the industry. This is, in fact, the framework in which ROMs will be used in this work, in which a shape optimization process is carried out.

Other contexts in which we may need to reduce the computational effort for the resolution of a numerical problem are the cases in which the solution needs to be solved in real-time, even on low-resource local devices (laptop, tablet, smartphones). This is the case, for example, of control problems over dynamical systems ([8]), problems in environmental sciences ([9]) or problems

with applications in finance.

Finally, ROMs techniques are gaining importance in some fields of engineering and applied sciences in which the complexity of the problem at hand is becoming increasingly high, either because of multi-physics couplings or multi-scale settings. We have examples of this in nuclear engineering ([10]), for the design of the cooling systems, or in fluid-structure interaction problems ([11]).

To conclude this section, we mention that parametric PDEs is not the only field of applicability of ROMs techniques. For example, in this work, we will use them in the context of:

- geometrical shape parametrization, as a pre-processing tool used to reduce the dimensionality of the parametric space used for the PDE; for more details on this aspect, refer to Section 2.4;
- dynamical systems, as a post-processing tool used to evolve the system generated by the PDE to the asymptotic state; for more details on this aspect, refer to Section 1.5.

1.2 Reduced Basis Reduced Order Methods (RB-ROMs)

All the methodologies we will consider in this work fall into the category of Reduced Basis ROMs (RB-ROMs).

We begin with a formal definition of a system of parametric PDEs in the form:

$$\text{find } u(\mu) \in \mathbf{X} \text{ s.t.: } \; a(u(\mu), w; \mu) = F(w; \mu) \;\; \forall w \in \mathbf{X}, \tag{1.1}$$

where $\mu \in \mathbf{P}$ is the parameter and $u(\mu) \in \mathbf{X}$ is the *exact solution* of the problem.

We introduce the notion of the solution manifold, that is the set of all possible solutions of our parametric problem under the variation of the parameter:

$$\mathbb{M} = \{u(\mu), \;\; \mu \in \mathbb{P}\}. \tag{1.2}$$

The exact solution in most of the cases is not available in an analytic manner, and it is approximated numerically. The so-called *truth solution* can be obtained using different techniques, where the most popular ones are Finite Elements (FE) and Finite Volumes (FV). In this work, we consider a finite volume approach. For a brief description of FV, refer to Section 3.4.

We denote the truth solution to our problem with $u^{\mathcal{N}}$, where $\mathcal{N}$ represents the number of degrees of freedom associated with it. This number is related

to the discretization methodology employed and the dimension of the computational grid. A high value of $\mathcal{N}$ implies a high dimension of the resulting linear system and consequently a high computational cost.

The problem associated to the truth solution can be stated as:

$$\text{find } u^{\mathcal{N}}(\mu) \in \mathbf{X}^{\mathcal{N}} \text{ s.t.: } \; a(u^{\mathcal{N}}(\mu), w; \mu) = F(w; \mu) \;\; \forall w \in \mathbf{X}^{\mathcal{N}}, \tag{1.3}$$

where $\mathbf{X}^{\mathcal{N}}$ is a finite dimensional subspace of $\mathbf{X}$ of dimension $\mathcal{N}$. The manifold is then defined by:

$$\mathbb{M}^{\mathcal{N}} = \{u^{\mathcal{N}}(\mu) \;\; \mu \in \mathbb{P}\}. \tag{1.4}$$

The final goal of RB methods is to approximate any element of this manifold using a low number of basis functions, or modes, $\{\chi_i(x)\}_{i=1}^{N}$ that form what we call the reduced basis. These N functions are globally defined over the computational domain and are obtained using some pre-computed truth solutions for particular parameter values. For more details on the possible techniques that could be employed to compute the reduced basis, we refer to Section 1.3. In particular, the technique we will consider, named Proper Orthogonal Decomposition (POD), considers a hierarchical orthonormal basis generated using energetic considerations.

The *reduced solution* $u_N^{\mathcal{N}} \approx u^{\mathcal{N}}$ is composed by a suitable linear combination of these modes:

$$u_N^{\mathcal{N}}(\mu) = \sum_{i=1}^{N} \xi_i(\mu)\chi_i(x), \tag{1.5}$$

and the reduced formulation of the problem (1.3) becomes:

$$\text{find } u_N^{\mathcal{N}}(\mu) \in \mathbf{X}_N^{\mathcal{N}} \text{ s.t.: } \; a(u_N^{\mathcal{N}}(\mu), w; \mu) = F(w; \mu) \;\; \forall w \in \mathbf{X}_N^{\mathcal{N}}, \tag{1.6}$$

where $\mathbf{X}_N^{\mathcal{N}} = span(\{\chi_i(x)\}_{i=1}^{N})$.

Hence, while the degrees of freedom associated with the truth solution $\mathcal{N}$ are typically high, the degrees of freedom associated with the RB approximation are only N, where $N \ll \mathcal{N}$.

This projection-based approach in which the original equations (1.3) are projected onto the reduced basis obtaining (1.6) will be referred in the following as the *intrusive approach* (for more details see for example [12] or [13]). We will not go more in-depth on how this projection is performed since in this work another technique, named Proper Orthogonal Decomposition with Interpolation (PODI), will be used in order to obtain $u_N^{\mathcal{N}}$. PODI will be described in the following section and will be referred to as the *non-intrusive approach* for reasons that will be more clear in the following.

As a last remark, we highlight the fact that, as already said, to compute the reduced basis we need a set of what can be called, using machine learning nomenclature, training data, that is a set of truth solutions for particular values of the parameters computed using the high fidelity solver. This is usually the most expensive phase in the ROM pipeline, and the applicability of ROMs techniques is often related to the size of the sampling needed to obtain certain performances.

Connected to this last aspect, we mention here another important feature of ROMs, that is the offline-online decomposition. By this, we mean that the use of RB-ROMs techniques can be split, as we have already seen, in two phases:

- an *offline-phase*, in which a set of high fidelity solutions are collected and the reduced basis is computed by combining them;
- an *online-phase*, in which we project the problem onto the reduced space and the solutions for new parameters are computed in an efficient manner.

The first phase requires typically more computational time, having to rely on the high fidelity solver, and is usually executed in high performance computing clusters using parallel programming techniques. The second phase, however, is quite inexpensive from the computational point of view and could be done in low-end terminal or even tablet/smartphones, widening the field of applicability of this method.

1.3 Computation of the Proper Orthogonal Decomposition (POD) Reduced Basis

The first step of a reduced basis approach is the selection of a low dimensional basis capable of expressing as good as possible the set of all possible outcomes of our system of interest, whether it is a fluid dynamics problem, the output of a deformation process or a dynamical system. These modes could be then used to project the equation they were derived from and to obtain a reduced version of it, with much lower dimensionality.

Proper Orthogonal Decomposition (POD). A possible choice for the computation of the basis is proper orthogonal decomposition, which consists basically of a SVD applied to a set of high fidelity solutions, which we now describe.

Let us begin with some notation. We will call a possible outcome of our system $\mathbf{y}$ a *snapshot* and denote with m its dimensionality ($\mathbf{y} \in \mathbb{R}^m$). We will then denote with n the number of snapshots collected in the offline phase, and gather them in the *snapshots' matrix* $\mathbf{Y} \in \mathbb{R}^{m \times n}$:

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_n \\ | & | & & | \end{bmatrix}. \tag{1.7}$$

The matrix $\mathbf{Y}$ will, in general, have a rank $d \leq \min(m, n)$ but, WLOG, we will restrict ourselves to the case $d = n < m$.

The $n < m$ condition is motivated by the fact that we are treating high dimensional snapshots, in which $n \ll m$; for example, we may have values of m of the order of the hundreds of thousands, while the number of snapshots n will be of the order of hundreds.

The condition $d = n$ instead is motivated by the fact that we are considering the output of a very complex system and it's very unlikely that the snapshots matrix will not be of full rank, i.e. that the space generated by its columns could be approximated by a set of $d < n$ vectors up to the epsilon machine. However, the central part of POD is that, in some cases, a low-rank approximation of the snapshots matrix can be generated by accepting a value of the error higher than the epsilon machine, as we will soon show.

To proceed with notation, we call l the number of POD modes that will be used to construct this approximation, with $l < n < m$.

We now recall, in the following Theorem, the singular value decomposition:

Theorem 1 ***(SVD)*** *Given $\mathbf{Y} \in \mathbb{R}^{m \times n}$ of rank $d = n < m$:*

- *$\exists \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n > 0$ (singular values);*
- *$\exists \mathbf{\Psi} \in \mathbb{R}^{m \times m}$ orthogonal with columns $\{\psi_i\}_{i=1}^m$ (left singular vectors);*
- *$\exists \mathbf{\Phi} \in \mathbb{R}^{n \times n}$ orthogonal with columns $\{\phi_i\}_{i=1}^n$ (right singular vectors);*

with $\mathbf{Y} = \mathbf{\Psi \Sigma \Phi}^T$ and $\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & & \\ & & \end{pmatrix} \in \mathbb{R}^{m \times n}$.

The following properties hold:

1. $\mathbf{Y}\phi_i = \sigma_i\psi_i$, $i = 1, ..., n$;

2. $\mathbf{Y}^T\psi_i = \sigma_i\phi_i$, $i = 1, ..., m$;

3. ψ_i *is an eigenvector of* $\mathbf{Y}\mathbf{Y}^T$ *with eigenvalue* σ_i^2 *(for* $i = 1, ..., m$*);*

4. ϕ_i *is an eigenvector of* $\mathbf{Y}^T\mathbf{Y}$ *with eigenvalue* σ_i^2 *(for* $i = 1, ..., n$*).*

SVD is a very powerful tool used in a variety of applications. The third property reported in the theorem states that to obtain the matrix $\mathbf{\Psi}$ one could solve an eigenvalue problem on $\mathbf{Y}\mathbf{Y}^T$, that we can interpret as a covariance matrix on the snapshots. This observation suggests that in some sense, that will shortly be more clear, the left eigenvectors are directions that maximize the variance of the space spanned by these vectors.

To make the last assertion more precise, let us proceed with the formal definition of the POD basis:

Theorem 2 ***(POD basis)*** *Given* $\mathbf{Y} \in \mathbb{R}^{m\times n}$ *of rank* $d = n < m$, $\{\chi_i\}_{i=1}^l$, *for* $l \in \{1, .., n\}$ *is the POD basis of* $\mathbf{Y}$ *if and only if it is a solution of:*

$$\max_{\tilde{\psi}_1,\tilde{\psi}_2,..\tilde{\psi}_l} \sum_{i=1}^{l}\sum_{j=1}^{n} | < \mathbf{y}_j, \tilde{\psi}_i >_{\mathbb{R}^m} |^2 \quad s.t. < \tilde{\psi}_i, \tilde{\psi}_i >_{\mathbb{R}^m} = \delta_{i,j}, \quad for\ 1 \le i, j \le n. \tag{1.8}$$

This can be read as: the POD basis is the one that maximizes the similarity (as measured by the square of the scalar product) between the snapshots matrix and its elements, under the constraint of orthonormality. In this sense, when we obtain the l-rank POD basis, we have the set of dimension l capable of optimally express the variance in the snapshots.

The link between POD and SVD is stated in the following theorem, that can be proven using Lagrangian penalization techniques (see for example [14]):

Theorem 3 *Given* $\mathbf{Y} \in \mathbb{R}^{m\times n}$ *of rank* $d = n < m$, *its l-rank POD basis is given by the set of the first l left singular vectors* $\{\psi_i\}_{i=1}^l$ *Moreover, we have:*

$$\max_{\tilde{\psi}_1,\tilde{\psi}_2,..\tilde{\psi}_l} \sum_{i=1}^{l}\sum_{j=1}^{n} | < \mathbf{y}_j, \tilde{\psi}_i >_{\mathbb{R}^m} |^2 = \sum_{i=1}^{l} \sigma_i^2. \tag{1.9}$$

What this theorem gives us are two important things: a practical way of resolving problem (1.8) and a way to quantify the variance contained in the l-rank POD basis. In particular, we can infer the quality of the l-rank approximation given by the POD basis considering the following ratio:

$$\frac{\sum_{i=1}^{l} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2}. \tag{1.10}$$

This is the equivalent of the statistical concept of R squared, which quantifies the quality of a statistical model in terms of the fraction of explained variance over total variance, that is what is reported in this formula.

For example, we can understand, looking at the decay of the values of σ_i, how well a system could be expressed by a reduced method and what is the value of l that is optimal to do so.

We report here two other interesting results that express the error generated by considering a truncated reconstruction of the snapshots matrix by considering only the first l modes expressed in two different matrix norms:

$$\begin{aligned} ||\mathbf{Y} - \mathbf{\Psi}_l \mathbf{\Sigma}_l \mathbf{\Phi}_l^T||_2^2 =& \sigma_{l+1}^2, \\ ||\mathbf{Y} - \mathbf{\Psi}_l \mathbf{\Sigma}_l \mathbf{\Phi}_l^T||_F =& \sqrt{\sum_{i=l+1}^{m} \sigma_i^2}, \end{aligned} \tag{1.11}$$

where $\mathbf{\Sigma}_l \in \mathbb{R}^{l \times l}$ is build considering the first l singular values and $\mathbf{\Psi}_l \in \mathbb{R}^{m \times l}$ and $\mathbf{\Phi}_l \in \mathbb{R}^{n \times l}$ are built considering the first l columns of respectively $\mathbf{\Psi}$ and $\mathbf{\Phi}$.

In (1.11), $||.||_F$ is the Frobenius norm, computed as:

$$||\mathbf{M}||_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij}^2}, \tag{1.12}$$

and $||.||_2$ is the norm induced by the Euclidean norm over $\mathbb{R}^n$, that can be shown to be equal to the maximum singular value of the matrix.

Greedy generation. Another possibility for the generation of the reduced basis is the greedy approach.

To understand the difference of this approach with respect to the standard POD approach, we highlight how, in the latter, our strategy is composed by:

- an exploration of the solution manifold, performed computing the snapshots matrix;

- an analysis of the space explored, using SVD.

In the POD greedy approach instead, the idea is to build the base during the exploration of the space. In particular, we begin with only one snapshot, corresponding to the first element of the base, and then iteratively add a new element to the base computing a new solution for a particular value of the parameter. Doing this, we only need to solve a total of N truth solutions to generate the N-dimensional reduced basis space.

However, to select in a proper way the new elements to be added to the reduced basis, we must have available an estimator $\eta(\mu)$ that can predict the error due to the model order reduction. Using this, the $n+1$ parameter to be added to the reduced basis space is the one that maximizes the error of the previous reduced order model:

$$\mu_{n+1} = argmax_{\mu \in \mathbb{P}} \eta(\mu). \tag{1.13}$$

The enrichment of the reduced basis is repeated until the maximal estimated error is below a required tolerance.

The pro of using this approach is that the number of offline snapshots to be computed is quite low and the offline phase is carried out more efficiently, computing a new solution in the zone of the parameter space where we have poor information. However, the reduced basis generated using this approach is not guaranteed to be optimal, as the standard POD one was.

Another interesting characteristic of the standard POD basis as defined in (1.8) with respect to the greedy basis is that it is a hierarchical basis, in which the elements are ordered in terms of the energy contained (or the variance explained). This characteristic authorizes us to truncate the basis at any point in an optimal and simple way.

1.4 Proper Orthogonal Decomposition with Interpolation (PODI)

PODI is an alternative to standard POD that is more used in the industrial setting because of its ease of use and its flexibility.

In PODI the first phase, the generation of the reduced basis, is performed in the same way as the standard approach. A significant difference, however, is that while in intrusive POD we need to rely upon an open-source software to compute the truth solutions, since in the online phase we will need to have access to the source code in order to project the equations. In the PODI approach, this is not necessary, and we can use commercial software or even experimental data to train our model.

The reason why in PODI we have this freedom in the offline phase is that the online phase is completely different from the standard POD one.

We recall here that the assumption of RB-ROMs is that the truth solution of our problem $u^{\mathcal{N}}$ can be approximated by the reduced solution $u_N^{\mathcal{N}}$ composed by linear combination of spatial modes $\chi_i(x)$ multiplied by coefficients $\xi_i(\mu)$, that is:

$$u^{\mathcal{N}}(\mu) \approx u_N^{\mathcal{N}}(\mu) = \sum_{i=1}^{N} \xi_i(\mu)\chi_i(x). \tag{1.14}$$

In PODI then we define an interpolator considering as a function the one that associates the value of the parameter μ to the modal coefficients of the related solution $\{\xi_i(\mu)\}_{i=1}^N$. This multi-dimensional interpolator is trained using the data coming from the snapshots matrix, in which both the parameter values and the modes coefficients are known, and is then used to infer the value of the coefficients associated with new parameters. The values of the coefficients are then used to reconstruct the approximated truth solution using (1.14).

This approach is entirely data-driven and is independent both on the equations and on the physics of the problem. This has its advantages and disadvantages. The ease of use and the complete freedom in the generation of the snapshots, that are crucial in an industrial setting in which commercial software are widely spread, correspond to a lower accuracy associated with the reduced model.

An aspect we have not yet discussed is how the interpolation is carried out. In this work, we have compared different choices of interpolation in term of prediction error and the result of this comparison are presented in Section 4.3.

1.5 Dynamic Mode Decomposition (DMD)

Dynamic Mode Decomposition (DMD) is a data-driven ROM technique that allows approximating the evolution of a complex dynamical system as the combination of structures, or modes, that evolve linearly in time.

The basic idea and the main assumptions made for DMD are the same as POD. We are given a dynamical system that at first glance exhibits complex behavior, possibly non-linear and high dimensional, and we try to express it as a linear superposition of modes, in a number that is typically much smaller than the original dimension.

The workflow associated with DMD is also quite similar to the one associated with POD: we first need to collect a time series of fields that constitutes

the snapshots matrix using which we can compute the POD modes (with the corresponding singular values). The difference with POD is that the POD modes extracted this way are then used in DMD to build a low dimensional approximation of the linear operator that defines the evolution of the system. We can then reconstruct the dynamics of the exact linear operator by considering an eigendecomposition of the approximated one, using the so-called *DMD modes*. The DMD modes are spatial fields that often can be identified with coherent structures in the flow and the eigenvalues associated define the behavior of the corresponding mode, its growth/decay or its oscillations.

We now present the mathematical formulation of DMD. We begin by expressing our dynamical system as an equally-spaced time series of data vectors $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^m$. The main assumption behind DMD is that the snapshots can be generated, or approximated in the case of non-linear dynamics, by the following linear dynamics:

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k, \tag{1.15}$$

for a suitable finite dimensional matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$. This is, of course, a hypothesis on the system and the applicability of DMD depend on it.

The aim of DMD is then to build a low-dimensional approximation of $\mathbf{A}$, using the values of $\mathbf{x}_i$ for $i = 0, ..., n$, and then use it to infer the dynamics for values of i greater than n. To do this, we proceed to arrange data into matrices:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_0 & \mathbf{x}_1 & \ldots & \mathbf{x}_{n-1} \\ | & | & & | \end{bmatrix}, \tag{1.16}$$

and

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_n \\ | & | & & | \end{bmatrix}, \tag{1.17}$$

where $\mathbf{Y}$ consists basically of a translation of $\mathbf{X}$.

We now have that the condition (1.15) can be expressed as:

$$\mathbf{Y} \approx \mathbf{A}\mathbf{X}. \tag{1.18}$$

At this point, we have that the best-fit matrix A is given by:

$$\mathbf{A} = \mathbf{Y}\mathbf{X}^{\dagger}, \tag{1.19}$$

where † indicates the Moore-Penrose pseudo-inverse, that is the matrix that minimizes the error:

$$||\mathbf{Y} - \mathbf{A}\mathbf{X}||_F, \tag{1.20}$$

where $||.||_F$ is the Frobenius norm, computed as in (1.12).

While in theory it is possible to obtain $\mathbf{A}$ using (1.19), this is discouraged in practice because of its considerable dimension and the difficulties that would arise in order to obtain it numerically. The idea is now to obtain information on $\mathbf{A}$, in particular its eigendecomposition, using a low-dimensional approximation $\tilde{\mathbf{A}}$.

To perform this, we compute the SVD of $\mathbf{X}$ obtaining[1]:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{1.21}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$, $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$. We assume here, as we already did in the section regarding POD, that X is full rank n and that $n < m$. For the sake of simplicity, we will denote with $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{\Sigma}$ the n-truncated versions $\mathbf{U}_n \in \mathbb{R}^{m\times n}$, $\mathbf{V}_n \in \mathbb{R}^{n\times n}$ and $\mathbf{\Sigma}_n \in \mathbb{R}^{n\times n}$, as defined in (1.11).

We then define the matrix $\tilde{\mathbf{A}}$ by projecting $\mathbf{A}$ on the first n left-singular vectors of $\mathbf{X}$, obtaining:

$$\tilde{\mathbf{A}} = \mathbf{U}^T\mathbf{A}\mathbf{U} = \mathbf{U}^T\mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}. \tag{1.22}$$

We have thus obtained a low-rank linear operator $\tilde{\mathbf{A}} \in \mathbb{R}^{n\times n}$ that approximates the high-rank operator $\mathbf{A} \in \mathbb{R}^{m\times m}$ without the need to explicitly computing the latter.

We now compute the eigendecomposition of $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}. \tag{1.23}$$

At this point it can be proven that only the first n eigenvalues of $\mathbf{A}$ are non-zero. The corresponding eigenvector, contained in the matrix $\mathbf{\Phi} \in \mathbb{R}^{m\times n}$, can be computed in two ways:

- by projecting the low-rank approximation $\mathbf{W}$ on the high dimensional space, via:
$$\mathbf{\Phi} = \mathbf{U}\mathbf{W}, \tag{1.24}$$
obtaining the so-called *projected DMD modes*;

- by computing:
$$\mathbf{\Phi} = \mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}, \tag{1.25}$$
obtaining the *exact DMD modes*.

[1] we restrict ourselves to the real case

We now show that the exact DMD modes are indeed the eigenvectors of $\mathbf{A}$ and that the corresponding eigenvalues are the same of $\tilde{\mathbf{A}}$. In fact, we have that, using the relations $\mathbf{A} = \mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T$ and $\tilde{\mathbf{A}} = \mathbf{U}^T\mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}$:

$$\begin{aligned} \mathbf{A}\mathbf{\Phi} &= \mathbf{\Phi}\mathbf{\Lambda} \\ \mathbf{A}\mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W} &= \mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}\mathbf{\Lambda} \\ \mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W} &= \mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}\mathbf{\Lambda} \\ \mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\tilde{\mathbf{A}}\mathbf{W} &= \mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}\mathbf{\Lambda} \\ \tilde{\mathbf{A}}\mathbf{W} &= \mathbf{W}\mathbf{\Lambda}. \end{aligned} \tag{1.26}$$

A similar procedure could also be considered for the projected modes.
Using the exact DMD modes, we can reconstruct the linear dynamics via:

$$\begin{aligned} \mathbf{Y} &\approx \mathbf{A}\mathbf{X} \\ &\approx \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\dagger\mathbf{X} \\ &\approx \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\dagger \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \dots & \mathbf{x_{n-1}} \end{bmatrix} \\ &\approx \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\dagger \begin{bmatrix} \mathbf{x}_0 & \mathbf{A}\mathbf{x}_0 & \mathbf{A}^2\mathbf{x}_0 & \dots & \mathbf{A}^{n-1}\mathbf{x}_0 \end{bmatrix} \\ &\approx \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\dagger \begin{bmatrix} \mathbf{x}_0 & \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\dagger\mathbf{x}_0 & \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\dagger\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\dagger\mathbf{x}_0 & \dots & (\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\dagger)^{n-1}\mathbf{x}_0 \end{bmatrix} \\ &\approx \mathbf{\Phi}\mathbf{\Lambda} \begin{bmatrix} \Phi^\dagger x_0 & \mathbf{\Lambda}\mathbf{\Phi}^\dagger\mathbf{x}_0 & \mathbf{\Lambda}^2\mathbf{\Phi}^\dagger\mathbf{x}_0 & \dots & \mathbf{\Lambda}^{n-1}\mathbf{\Phi}^\dagger\mathbf{x}_0 \end{bmatrix} \\ &\approx \mathbf{\Phi} \begin{bmatrix} \mathbf{\Lambda} & \mathbf{\Lambda}^2 & \dots & \mathbf{\Lambda}^n \end{bmatrix} \mathbf{\Phi}^\dagger\mathbf{x}_0. \end{aligned} \tag{1.27}$$

Finally, the state $\mathbf{x}_k$ can be approximated as:

$$\mathbf{x}_k = \mathbf{\Phi}\mathbf{\Lambda}^k\mathbf{\Phi}^\dagger\mathbf{x}_0. \tag{1.28}$$

We remark that the two possible approaches to build $\mathbf{\Phi}$ (exact or projected) are alternative and lead to slightly different DMD reconstructions.

CHAPTER 1. REDUCED ORDER METHODS: THEORY AND APPLICATIONS

Chapter 2

Geometrical Shape Parametrization

This chapter deals with geometrical shape parametrization applied to our case study, i.e. the hull of a cruise ship. More explicitly, we want to define a finite set of control parameters that, properly tuned, generate different deformations of an initial undeformed hull shape. The parameters are expressed as a vector of real-valued variables $\mu \in \mathbb{R}^N$, where N is the dimensionality of the parametrization.

We begin with some motivations and preliminary considerations on geometrical shape parametrization. Then we proceed with a description of the undeformed hull in order to fix some terminology and to define which are the constraints we will consider. Next, we briefly present Free Form Deformation (FFD), underlining some aspects which are relevant in our case. Finally, we mention a possible reduction of the parameter space that uses POD.

Because of the structure we decided to give to this work, the presentation of the specific settings used for the deformation, the results of the deformation itself and the results of the parameter space reduction are postponed to Section 4.1.

2.1 Motivations and Preliminary Considerations

Geometrical shape parametrization consists of the definition of a mathematical method, and consequently a numerical algorithm, that, given a particular value of some predefined parameters, returns a deformation of an initial undeformed geometrical object. This tool will be used in the shape optimization pipeline for the generation of the design space explored by the

optimization algorithm.

This step is of crucial importance since the generation of both a too small and a too big design space would imply bad performances of the entire pipeline. In fact, while in the case of a too small design space we would lose some possibly well performing shapes, in the case of a too big design space we would have difficulties in its exploration during the optimization phase.

For the definition of the parametrization, we will base ourselves on different kinds of considerations, coming both from naval architecture and mathematics. While the observations behind the methodology we will use, named Free Form Deformation, are completely general and applicable to different areas of engineering, the particular choices we will present pertain to the hull deformation setting. They are moreover dependent on the aim for which the deformation is performed and the particular ship considered. We note at this point that, if the number of points in which the undeformed ship is discretized is N_p, where this number is of the order of hundreds of thousands, and if any of these points could be moved independently in the 3 directions, the number of parameters, i.e. the dimensionality of the parametrization, would be $N = 3 \times N_p$. This number is prohibitive, leading to the need for a huge sampling and consequently of a huge computational cost[1]. Moreover, only a few of the configurations generated by this parameter space would be interesting from a practical point of view (for example, a symmetry constraint would halve the dimensionality of the parameter space). From this discussion, it is clear that a key feature our parametrization must possess is low dimensionality, and in general we must try to avoid the introduction of parameters that either have a low influence on the shapes generated or generate shapes that we are not interesting for us.

As a last preliminary remark, we note that in this chapter all the quantities (lengths, areas, volumes) are given on the full scale and not on the scale used in the simulations (see Section 3.1).

2.2 Case Study: A Fincantieri Cruise Ship

The hull we are considering here is taken from an already built and working cruise ship, illustrated in Figure 2.1.

The reference system we will consider, both in this chapter and in the followings, is the one reported in the figure: the x-axis agrees with the direction in which the ship is heading, the z direction is perpendicular to the

[1] the size of the sampling is connected to the error the ROM is able to generate. However, in general and independently of ROM, it is clear that the bigger the parameter space is, the higher is the computational cost that one must employ to explore it

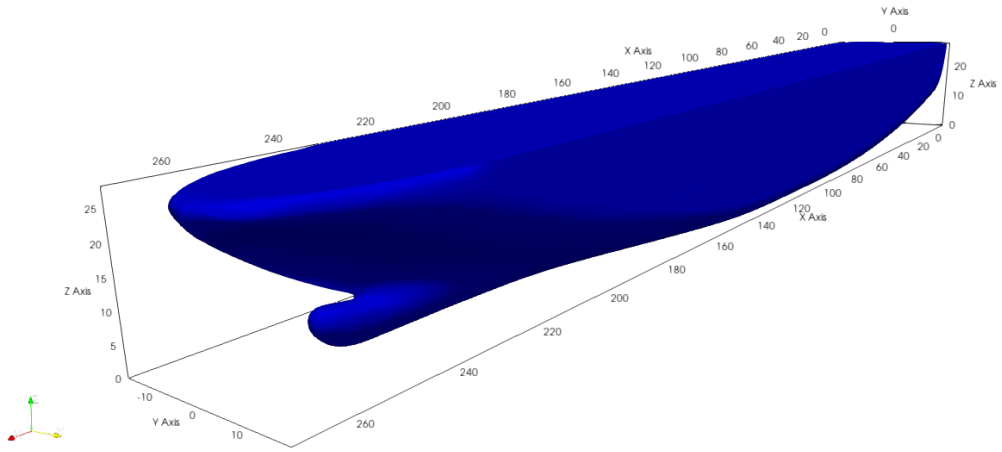


Figure 2.1: 3D view of the undeformed hull.

sea and directed to the sky and the y-axis is directed accordingly. As for the center, it is placed near the lower-bottom part of the ship.

An important feature of the considered hull is that the central part of the ship is made up of a rectangular parallelepiped that connects gradually and smoothly on the front to the bow and on the back to the stern of the ship. We anticipate at this point that our attention will be devoted to the frontal part of the ship and, for this reason, the deformations will take place there.

Notations and terminology. In naval architecture a boat is divided, no matter the size, in 20 chunks, generated by 21 equally spaced cuts obtained with planes perpendicular to the x-axis. The first cut corresponds to the origin of the x-axis, while the last one corresponds to the point of the waterline that is furthest from the origin of the x-axis. The intersections between our hull and these sections are illustrated in Figure 2.2, where on the left the first 11 sections and on the right the last 10 sections are displayed. These figures, together with analogous figures[2] obtained by intersections with planes perpendicular to y and z (Figures 2.3 and 2.4), are essential tools in naval architecture and will be used to confront the deformed hulls to the undeformed one.

Another representation that is very useful in visualizing a hull shape is the plot of areas, that represents in the abscissa the x value and in the ordinate

[2]regarding these sections, there is no standard convention on their number and locations. We have decided to consider 6 equally spaced sections in y and 9 equally spaced in z

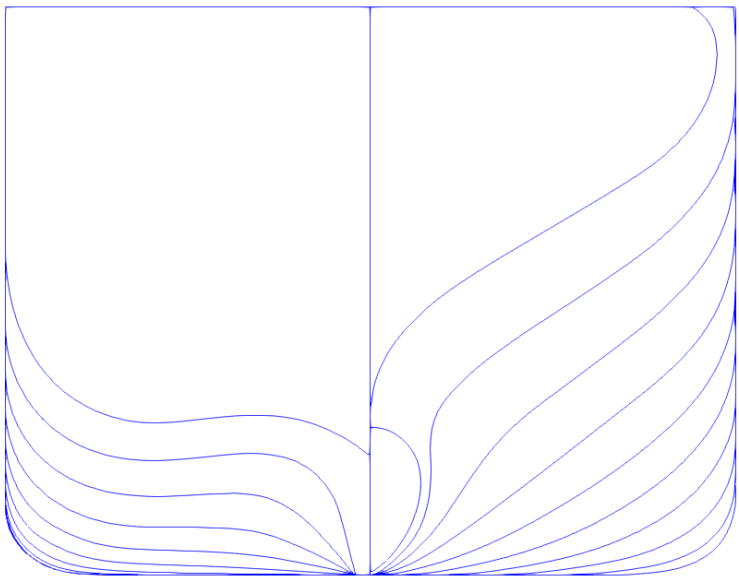


Figure 2.2: x-sections of the undeformed hull. Section 0 to 10 on the left, 11 to 20 on the right.

Figure 2.3: y-sections of the undeformed hull.

Figure 2.4: z-sections of the undeformed hull.

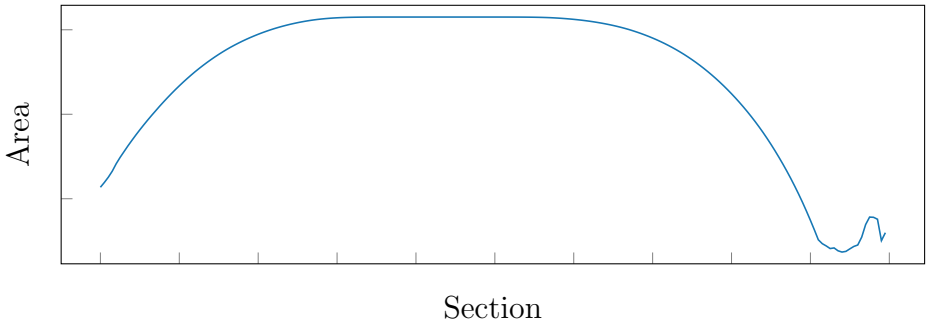


Figure 2.5: Plot of areas of the undeformed hull.

Length between perpendiculars (LPP)	242.000 m
X-coordinate of aft perpendicular	0.000 m
Length on waterline (LWL)	251.520 m
Length overall submerged (LOS)	260.029 m
Breadth moulded on WL (B)	36.00 m
Draught moulded on FP (TF)	8.000 m
Draught moulded on AP (TA)	8.000 m

Table 2.1: Geometrical characteristics of the undeformed hull.

the submerged area of the x-section. In Figure 2.5 we report the plot of areas of the undeformed hull. A feature characterizing the hull that can be visualized in this plot is the flex point of the curve in the frontal part of the ship. We mention that the fluctuations that can be observed in the frontal part of the curve, in the zone of the bulbous bow, are relative to errors generated by the numerical integration. We have decided not to consider higher order integration techniques, that could have returned smoother results, since this part of the curve is not interesting for our study.

Finally, in Table 2.1, we report some geometrical quantities associated with the boat (for a precise definition of the properties listed, see [15]).

Deformation constraints. Clearly, not all the deformations of the original hull are desirable or feasible, and so we want to impose some constraints. These are useful to restrict the dimension and size of the parameter space, to help us focus on portions of the hull that are, for the study we are performing, more interesting than others, and, finally, to fulfill some construction constraints, given by laws or by the shipowner.

A very simple constraint, as already mentioned, is that we want symmetric deformations in the y direction with respect to the longitudinal plane.

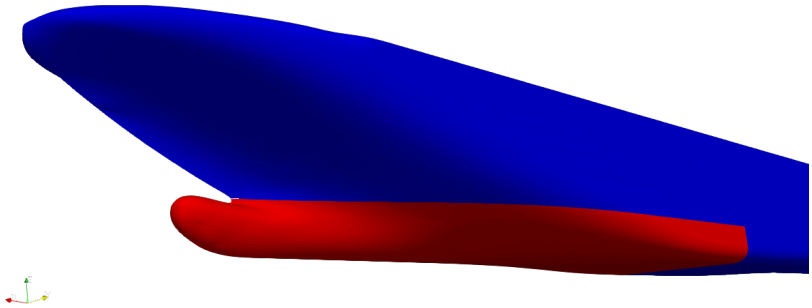


Figure 2.6: Indication, in red, of the part of the undeformed hull where the deformation will take place.

Another constraint is that we want to modify only the part of the ship below the waterline. The reason for this is that the emerged part of the hull is designed with other targets in mind, such as customer comfort and shipowner indications. Moreover, being the water much more dense than air, the main part of the resistance in a ship is given by water resistance. We also have to note that, in the simulations we computed, the upper part of the ship is not modeled in all its details, and so it would be useless to consider its contributions to the drag.

As already mentioned, another requirement we impose is that we only want to modify the frontal part of the ship, going from section 10 to 20, where section 10 is more or less the part in which the parallelepipedal part of the ship begins to tighten to link up to the bow. Clearly, the reason why we impose this constraint is that we do not want to modify the parallelepipedal part of the boat, being this part fixed by previous design steps. Concerning this, we also mention that it is essential that the modifications do not tend to enlarge the ship on y-direction, making it larger than its size at section 10. We will see how this is ensured in Section 4.1.

In Figure 2.6 we show the part of the ship we are going to deform highlighted in red.

To proceed, another important constraint is the volume constraint, meaning we do not want the boat to lose volume below a certain tolerance. This tolerance, in our case, is the 1‰ of the total submerged volume. How this last constraint is imposed is discussed in Section 4.4.2.

The last constraint we mention here is that the deformation must keep the continuity and smoothness of the shape of the hull, not generating sharp edges. This condition can be imposed at the level of FFD and can be proven

formally.

Of course, there are a lot of other constraints that need to be imposed, for example, stability constraints, but we choose not to consider further terms to simplify the analysis. These constraints should be checked a posteriori on the optimized hull, and in case of non-fulfillment, one should modify the definition of the parameter space. However, we will not take into account these in our study.

2.3 Free Form Deformation: Theory and Properties

FFD is a geometric tool, extensively employed in computer graphics, used to deform a rigid object based on the movement of some predefined control points. Introduced in [16], it has seen various improvements over the years. The reader can refer for example to [17] for Extended-FFD, [18] for a more recent review and [19] and [20] for a coupling with ROM techniques. In this work, we used the basic formulation, which we now briefly describe.

While most of the deformation techniques directly manipulate the geometrical object considered, the main idea behind FFD is to define a regular lattice of points around the object (or part of it) and manipulate the whole enclosed space by applying a motion to the points belonging to the lattice.

This characteristics of FFD entails some advantages:

- possibility to consider general shapes;
- possibility for the user to define the parameters explicitly;
- good performances and sensitivity even with low dimensional parametrizations.

As a last significant benefit of using FFD, we mention that the computation may be divided into an offline stage, more time-consuming, and an online stage, much cheaper, matching the need for ROM techniques.

Mathematical formulation. We begin with the definition of a differentiable and invertible map ψ that goes from the physical space to a reference space:

$$\psi : (x_1, x_2, x_3) \to (s, t, p). \tag{2.1}$$

Inside the physical space, we define a region D that is where the deformation will take place. The map ψ is then defined so that it maps this region

D to the unit cube:

$$\psi(D) = (0,1) \times (0,1) \times (0,1) = D_0. \tag{2.2}$$

Inside the unit cube we define a cubic lattice of control points, with L,M and N points respectively in x,y and z directions:

$$P^0_{l,m,n} = \begin{pmatrix} l/L \\ m/M \\ n/N \end{pmatrix} \in D_0, \;\; l = 0, ..., L, \; m = 0, ..., M, \; n = 0, ..., N. \tag{2.3}$$

We define a motion of those points $P^0_{l,m,n} \to P_{l,m,n}$ via:

$$P_{l,m,n} = P^0_{l,m,n} + \mu_{l,m,n}, \quad \mu_{l,m,n} \in \mathbb{R}^{3\times(L+1)\times(M+1)\times(N+1)}. \tag{2.4}$$

The parametric map T that performs the deformation of the physical space is then defined by:

$$T(\psi(x); \mu) = \psi^{-1}(\sum_{l=0}^{L}\sum_{m=0}^{M}\sum_{n=0}^{N} b^{L,M,N}_{l,m,n}(\psi(x))P_{l,m,n}), \tag{2.5}$$

where:

$$b^{L,M,N}_{l,m,n}(s,t,p) = \binom{L}{l}\binom{M}{m}\binom{N}{n}(1-s)^{(L-l)}s^l(1-t)^{(M-m)}t^m(1-p)^{(N-n)}p^n. \tag{2.6}$$

These are tensor products of trivariate Bernstein polynomials:

$$b^L_l(s) = \binom{L}{l}(1-s)^{(L-l)}s^l, \tag{2.7}$$

$$b^M_m(t) = \binom{M}{m}(1-t)^{(M-m)}t^m, \tag{2.8}$$

$$b^N_n(p) = \binom{N}{n}(1-p)^{(N-n)}p^n. \tag{2.9}$$

In the theory we presented, the subset D can be any set included in $\mathbb{R}^3$. However, for simplicity, we restrict to the case in which D is a rectangular parallelogram immersed in $\mathbb{R}^3$. In this case, the map ψ is simply composed of a dilatation, a translation and a rotation, and the grid of control points $P^0_{l,m,n}$ is simply a grid over the parallelogram. The particular definition of the parallelogram, of the grid and of the motions μ will be made in Section 4.1.

The parametric nature of Free Form Deformation is expressed in the vector $\mu_{l,m,n}$. However, the dimensionality of this is still too high for our purpose, hence we anticipate here that not all the movements of the points in the lattice will be independent and some constraints will be imposed.

Properties of FFD. We now present some properties of Free Form Deformation and some practical issues that will be used later in the description of the deformation setting.

We begin by noting that, since Bernstein polynomials vanish on the boundary, all the deformations take place inside the lattice.

Moreover, because of the definition of the map, the deformations inside the domain are continuous and smooth no matter how complex the motion of the points is.

However, since FFD could also be used to deform only part of the solid object, in general we have no assurance that continuity and smoothness are maintained at the boundary of the lattice. To ensure this, it can be proven that it is enough that one or two layers of control points for respectively continuity and smoothness are maintained fixed at the boundaries. This point is of crucial importance for us since, as we already mentioned, the motions we want to impose on the hull are only restricted to the submerged frontal part.

As a last remark, we mention that FFD, because of its definition, could be applied to both geometrical shape and computational mesh, allowing us to generate the mesh only once on the undeformed hull and thus saving computational time. However, we decided not to follow this approach because we noticed that the deformations we imposed, to both mesh and geometry, produced non-conformities on the mesh generated, particularly on the boundary layer.

2.4 ROM on Shape Parametrization

In this section, we discuss a possible model order reduction that could be employed to reduce the dimensionality of the parameter space we have built in the preceding sections.

This reduction is based on a Proper Orthogonal Decomposition applied to the deformed hulls, which have been obtained using FFD, giving us the principal modes of deformation, ordered from the most to the less "energetic"[3]. Considering then only a small number of these modes, we can define a new parametric map using a technique known in literature as the *basis shape approach* (see [18]).

In a basis shape approach, a new deformed hull R is obtained starting

[3]the energy of the modes in this setting is related to how much they contribute in generating the deformed shapes

from the undeformed hull r via:

$$R = r + \sum_{i=1}^{N_r} v_i U_i, \tag{2.10}$$

where N_r is the new number of parameters used, U_i, for $i = 1 \dots N_r$ are the first N_r deformation modes extracted from the POD and v_i are randomly generated coefficients.

From a practical point of view, the following steps are performed:

- Generation of N_s deformed hulls using FFD under uniform random variation of the parameters;
- Assembly of a snapshots matrix $\mathbf{S}$ of dimensions $2N_p \times N_s$, in which N_p is the number of points in which the hull is discretized. The snapshots' matrix contains in the first N_p rows the x-displacements of the points for the different hulls (deformed configuration - undeformed configuration) and in the N_p rows that follow the y-displacements. z-displacements are not considered since in the FFD setting we will consider in Section 4.1, no point will be moved in this direction;
- SVD decomposition on snapshots matrix $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$:
  - $\mathbf{\Sigma}$ contains the singular values, ordered from the highest to the lowest;
  - $\mathbf{U}$ contains the modes ordered from the one corresponding to the highest singular value to the one corresponding to the lowest.
- Projection of the snapshots over the modes $\mathbf{S}_{red} = \mathbf{U}^T\mathbf{S}$ and determination of the ranges of variation of the coefficient v_i for the linear combinations.

The results obtained using this methodology will be discussed in Section 4.1.3.

Chapter 3

Full Order Model Formulation

We now discuss the Full Order Model (FOM), which generates what we call the high fidelity solution.

We begin by pointing out that the purpose of this work is neither the validation of a physical model built to describe a real-life phenomenon nor a discussion of the discretization techniques that could be employed to simulate it numerically. In fact, our interest is mainly a study of the applicability of ROMs to an industrial setting. Nonetheless, it is clear that a robust and reliable FOM is essential for the generation of any meaningful result. With this in mind, we will limit ourselves to a brief presentation of the choices made, both for the physical model and the discretization techniques, with few remarks on critical aspects.

We begin by introducing the main hypothesis behind the full order model and presenting some definitions that will be useful in discussing the results. After that, we present the continuous model employed to represent the problem at hand, i.e. Volume Of Fluid (VOF), and the discretization technique used to numerically simulate the PDEs, i.e. Finite Volume Method (FVM).

As for the preceding chapter, the results of the FOM will be illustrated later in Section 4.2.

3.1 Main Features and Assumptions of the Full Order Model (FOM)

The simulations we are running pertain to an early stage in the design of a cruise ship. The purpose of this phase is the design of the hull shape, aiming to minimize the drag and disregarding:

- the detailed design of the superior part of the ship, which is especially

important for customer's comfort;

- the possible movements (pitch and draw) of the ship in response to the forces applied to it;
- the structural response of the boat to the mechanical stresses suffered;
- the coupling between the motion of the ship and the propeller;
- possible rough sea conditions.

In particular, because of these assumptions, we simulate the flow around the hull with a river-like approach, where the boat is fixed in space and the fluid is flowing around it with constant inlet velocity in the x direction[1].

Another hypothesis we are considering is the symmetry of the flow with respect to the plane that cuts the boat in half longitudinally; this assumption allows us to simulate only half of the fields by applying appropriate symmetry boundary conditions and thus halving the computational time.

As will be specified more precisely in the following sections, the model employed for the flow around the boat is that of an incompressible biphase flow, composed of water and air. An interesting remark we mention here is that the reason why a proper modellation of air and water interaction is needed is not that we do not want to miss air contribution to the total resistance, being this contribution much lower with respect to water resistance, but that we need to properly model the wave formation. To understand the need for a modellation of waves, we present an interesting example. A very famous paradox in fluid dynamics is D'Alambert Paradox. D'Alambert proved that, if a body is immersed in a flow modeled as incompressible, inviscid, potential and with constant velocity, it experiences zero drag force. If instead we consider a modification of the standard setting of the paradox, with a multiphase (for example air and water) flow with the same hypothesis made above and a body immersed in it (imagine for example a sphere moving with the surface of water passing through its center) it can be proven that the sphere is indeed subject to a drag. This is so because the sphere is moving the surface of the water and, to produce this motion, uses a part of its kinetic energy.

Ship resistance. The total force exerted by the flow on the ship, computed taking the integral of the surface stresses, can be decomposed in different ways.

[1] for a description of the reference system, see Section 2.2

3.1. MAIN FEATURES AND ASSUMPTIONS OF THE FULL ORDER MODEL (FOM)

In the first place, we can decompose the total force with respect to the global reference system, obtaining three components. In our case, only the x component, which we call *resistance* or *drag*, is relevant since:

- the y component is zero because of the symmetry hypothesis;
- the z component is irrelevant since we are not considering ship movements such as pitch and draw.

More precisely we are considering what is often referred to as *calm water resistance*, since the inflow is completely unperturbed (we are not considering rough sea).

Ship resistance can be then decomposed in two terms depending on the component of the surface stress that is integrated, generating pressure term (normal to the surface) and friction (or viscous) term (tangential to the surface).

Transient behavior. In the simulations we are considering, the drag is characterized by a transient behavior. In fact, we will observe oscillations around a mean value that eventually converge to a steady state in which the resistance is constant over time. While in theory one could directly run the simulation until the oscillations dissipate and the resistance reaches the steady state, this would take a very long time, resulting in a prohibitive computational cost. A much faster approach relies instead on DMD, as introduced in Section 1.5, allowing us to derive the asymptotic dynamics using a few seconds of simulation. More details on the use of DMD in our test case are given in Section 4.3.3.

For the sake of completeness, we remark that the steady-asymptotic behavior of drag implies neither the steadiness of the physical flow (in fact we are disregarding the fluctuating part of the flow using a RANS turbulence model), nor the steadiness of the mean flow (we are only considering the effect of the flow on ship resistance).

Similarity laws and Froude number. Similarity laws are a tool used to relate quantities (in our case ship resistance) obtained from a scaled model to the full scale phenomenon.

The scaled model is determined by defining a fixed length of the scaled boat, related to the complexity of the problem we are willing to solve, and by computing the scaled velocity from the full scale velocity by imposing the equality of an adimensional quantity, called Froude number. Froude number is defined as $Fr = \frac{u}{\sqrt{gL}}$, where g is the gravity acceleration, and u and L

are a characteristic flow velocity and a linear dimension of the problem, and represents the ratio between flow inertia and the gravitational field. This number is used in naval architecture since the dynamics of vessels that have the same Froude number (like our scaled and full-size model) can be easily compared as they produce a similar wake.

These laws have been originally formulated in order to conduct experimental tests on scaled hulls. However, they can also be used in CFD, allowing us to run a simulation not on the full scale model but on a scaled hull, in our case with a 1:25 ratio. While in an experimental setting these laws had the purpose of being able to test a hull without the need to build the real ship, in CFD the purpose of scaling is to reduce the number of cells needed. This is motivated by the fact that the problem at the model scale possesses a Reynolds number that is lower than the one at the full scale (for a definition of the Reynolds number see Section 3.2).

3.2 Navier-Stokes Equations

In this section, we introduce the analytical model apt to describe the behavior of an incompressible viscous Newtonian fluid, i.e. Navier-Stokes equations. We will consider, in this section and the following, two modifications of the basic formulation that will allow us to consider, respectively, turbulent flows (with the RANS model) and multiphase flows (with the VOF model).

The basic model reads:

$$\begin{cases} \frac{\partial u}{\partial t} + (u \cdot \nabla)u + \frac{1}{\rho}\nabla p - \nabla \cdot \nu \nabla u = 0, \\ \nabla \cdot u = 0, \end{cases} \tag{3.1}$$

where u, p, ρ and ν denote respectively fluid velocity, pressure, density and kinematic viscosity.

The first equation in (3.1))expresses the momentum balance in the infinitesimal volume and is essentially a reformulation of Newton'st 2^{nd} law of dynamics expressing the Eulerian fluid's acceleration ($\frac{\partial u}{\partial t}$) as a result of convection ($(u \cdot \nabla)u$), a gradient of pressure ($\frac{1}{\rho}\nabla p$) and diffusion ($\nabla \cdot \nu \nabla u$). The second equation instead expresses the mass conservation, emerging as solenoidality of the velocity field because of the incompressibility hypothesis.

Turbulence handling. The Navier-Stokes equations, as formulated in (3.1), are inoperative from a numerical point of view whenever the flow considered exhibit high velocities and complex geometrical features. The phe-

nomenon behind this is called turbulence and is one of the most ambitious and interesting open problems of applied mathematics.

A precise definition of turbulence is not available at the moment, and its characterization is usually made in terms of a dimensionless quantity called Reynolds Number, defined as:

$$Re = \frac{uL}{\nu}, \tag{3.2}$$

where ν is the kinematic viscosity of the fluid and u and L are a characteristic flow velocity and a linear dimension of the problem (for more details see [4]).

Reynolds number quantifies the relative importance of the convection term (in the numerator) with respect to the diffusive term (in the denominator) for the flow considered. A high value of the Reynolds number corresponds to a convection dominated turbulent flow, characterized by:

- highly irregular movement of the particles in the flow, making a statistical rather than deterministic treatment necessary;
- presence of chaotic changes in the flow fields, where small initial deviations result in enormous final deviations;
- presence of a strong three-dimensional vortex generation mechanism;
- enhanced mixing of the flow, resulting in a dissipative process, particularly important at the small scales.

Because of the importance of simulating high Reynolds flows, that are ubiquitous in industrial applications, and the impossibility to treat them using (3.1), because of the problems we mentioned, several models have been formulated over the years. The approach we considered in our work, named Reynolds-Averaged Navier-Stokes (RANS), despite being one of the earliest and simplest models available, is still (with a number of tweaks and improvements) the most used in an industrial setting.

The basic idea of a RANS model is decomposing the velocity field $u(x,t)$ expressing it by a sum of mean and fluctuating part:

$$u(x,t) = \bar{u}(x,t) + \tilde{u}(x,t). \tag{3.3}$$

Inserting this decomposition (and an analogous one made for pressure) inside the original N-S equations (3.1), we obtain:

$$\begin{cases} \frac{\partial \bar{u}}{\partial t} + (\bar{u}\cdot\nabla)\bar{u} + \frac{1}{\rho}\nabla\bar{p} - \nabla\cdot\nu\nabla\bar{u} - \nabla\cdot(\tilde{u}\otimes\tilde{u}) = 0 \\ \nabla\cdot\bar{u} = 0. \end{cases} \tag{3.4}$$

The additional term appearing in the first equation is expressed as a divergence of the tensor $\tilde{u} \otimes \tilde{u}$, known in the literature as Reynolds stresses tensor R. This term expresses the additional diffusivity in the flow generated by the turbulent behavior, and needs to be modeled as a function of the mean velocity field $\bar{u}$ in order to close the equations. The particular model chosen in this work will be declared in Section 4.2.

3.3 Volume Of Fluid (VOF)

Volume of fluid is a free-surface modeling technique that allows to describe a multiphase fluid composed of two incompressible, isothermal immiscible fluids (water and air, in our case). For a more in-depth discussion on VOF, we refer to [21].

This method is based on a phase-fraction technique: a new scalar variable, denoted by α, is added to N-S equations, representing the fraction of water contained in the infinitesimal volume (or in the finite volume, when we will discretize the equations). This variable belongs to the interval $[0, 1]$, where $\alpha = 1$ represents a point in which water is present, $\alpha = 0$ a point in which air is present and $\alpha \in (0, 1)$ represents interface points. By its nature, α is a discontinuous variable, and so the discretization method must ensure that the interface is captured a small number of cells.

We now report the equations that define the VOF method:

$$\begin{cases} \frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u) + \nabla p - \rho g - \nabla \cdot \nu \nabla u - \nabla \cdot R - f_\sigma = 0, \\ \nabla \cdot u = 0, \\ \frac{\partial \alpha}{\partial t} + \nabla \cdot (u\alpha) = 0, \end{cases} \tag{3.5}$$

where the density ρ and the kinematic viscosity ν are defined using an algebraic formula expressing them as a convex combination of the corresponding water and air properties:

$$\rho = \alpha \rho_W + (1 - \alpha)\rho_A, \tag{3.6}$$

$$\nu = \alpha \nu_W + (1 - \alpha)\nu_A. \tag{3.7}$$

The first two equations in (3.5) represent the classical continuity and momentum balance equations for an incompressible fluid. We make here some remarks:

- the inertial and the convective terms have not been expanded as in (3.1) since ρ, in this case, is not a constant;

- a gravitational contribution is included, appearing as an external volume force field ρg;

- a new term is present in the momentum equation, namely f_σ, that models surface tension as an external surface force. This term needs further modeling to become usable; however, since we decided to neglect it, we will not proceed with the discussion.

The third equation in the system (3.5) is a new equation required to close the system after the addition of the variable α. In fact, this equation is simply a transport equation for the fraction of fluid in which only the convective term is considered.

As already mentioned, the main difficulty in using a VOF approach is the possible smearing of the free-surface, caused by excessive diffusion introduced in the discretization of the transport equation. For this reason, a multitude of schemes has been tested over the years in order to obtain a sharp, yet stable, solution. Another possibility to tackle this problem is using a finer grid in the regions where the variable α is presumed to be discontinuous; in Section 4.2.2 we will present the strategy we have used in order to achieve this. Another approach relies on an alternative formulation on the VOF method, resulting in an additional term in the transport equation for the fraction of fluid, which will be presented in the next section.

Before passing to a description of the discretization technique we have employed, we mention that we have voluntarily omitted the specification of the initial and boundary conditions for the problem (3.5) because we will declare them after the definition of the computational domain in Section 4.2.

3.4 Discretization of the Equations - Finite Volume Method (FVM)

In this section, we briefly present the discretization technique used to numerically approximate the equations presented in the preceding section, which is Finite Volume Method (FVM). Being a complete discussion of FVM far beyond the scope of this section, we refer to classical texts such as [22] or [23].

FVM, like Finite Element Method (FEM) and Finite Difference Method (FDM), is a method used to translate a system of PDE into a system of algebraic equations. The solution of such system provides the value of the solution of the problem considered at discrete positions in the computational domain.

The first step in a finite volume discretization is the generation of a computational mesh over the domain of interest. The mesh is made up of a tessellation of non-overlapping polyhedra, with complete freedom on the type of polyhedra and their size. The quality of the mesh employed is crucial in order to obtain a good approximation of the fields, and its generation is an issue of primarily importance, especially in industrial problems because of complex geometrical features. We will return to this aspect in Section 4.2.2 when we will discuss the procedure employed to generate the mesh.

Once a suitable tessellation is given, the system of PDEs is written in integral form over each cell. The degrees of freedom of the resulting discretized system of algebraic equations are defined as the averages of the fields over the finite volumes, and their is indicated by $\mathcal{N}$.

We now discuss FVM in the context of classical Navier-Stokes equations, as reported in (3.1), considering the discretization of the terms appearing in the momentum and continuity equation. After that, we will consider how the terms and equations added in the VOF method can be treated.

Momentum equation. The momentum balance equation is written for the i^{th} finite volume V_i as:

$$\int_{V_i} \frac{\partial}{\partial t} u \, dV + \int_{V_i} (u \cdot \nabla) u \, dV - \int_{V_i} \nabla \cdot \nu \nabla u \, dV + \int_{V_i} \nabla p \, dV = 0. \tag{3.8}$$

The gradient of pressure term can be expressed using the Gauss' theorem as:

$$\int_{V_i} \nabla p \, dV = \int_{S_i} p \, dS \approx \sum_f S_f p_f, \tag{3.9}$$

where $S_i = \partial V_i$ is the boundary of the finite volume. The approximation of the surface integral is built by taking the sum over the faces f of the control volume of the area vector of each face S_f times the value of the pressure at the center of the faces p_f.

The convective term can be discretized as follows:

$$\int_{V_i} (u \cdot \nabla) u \, dV = \int_{S_i} (u_f \cdot \, dS) u_f \approx \sum_f (u_f \cdot S_f) u_f = \sum_f F_f u_f, \tag{3.10}$$

where u_f is the velocity vector evaluated at the center of each face of the finite volume and $F_f = S_f \cdot u_f$ is the mass flux through each face.

We note here that the values of p_f and u_f, pressure and velocity at the center of each face of the finite volume, must be obtained from the cell center values using suitable interpolation schemes. Possible alternatives are central, upwind, second order upwind differencing schemes.

The diffusive term is discretized as:

$$\int_{V_i} \nabla \cdot \nu \nabla u \, dV = \int_{S_i} \nu \nabla u \cdot \, dS \approx \sum_f \nu (\nabla u)_f \cdot S_f, \tag{3.11}$$

where $(\nabla u)_f$ is the gradient of u computed in the finite volume faces.

In the case of orthogonal meshes, in which the face dividing two cells is orthogonal with respect to the distance connecting the two cell centers, the term $(\nabla u)_f \cdot S_f$ could be computed via:

$$(\nabla u)_f \cdot S_f = |S_f| \frac{u_N - u_P}{|d|}, \tag{3.12}$$

in which the face f divides cell P from the cell N and d is the distance vector connecting the two cell centers.

Pressure equation. The mass balance equation, in the case of incompressible flows, boils down to the condition of divergence zero of the velocity field $\nabla \cdot u = 0$.

The standard for a Finite Volume discretization procedure is instead to work on a modified version of this equation, called the Poisson equation for pressure, obtained taking the divergence of the momentum equation and using the divergence-free constraint mentioned above. This equation reads:

$$\Delta p = -\nabla \cdot (u \cdot \nabla) u. \tag{3.13}$$

Together with this equation, we define suitable boundary conditions for p, with:

$$\nabla p \cdot n = 0, \tag{3.14}$$

imposed in all the boundaries except the outlet, and:

$$p = 0, \tag{3.15}$$

imposed in the outlet.

The benefit of working with this equation instead of the original $\nabla \cdot u = 0$ is that the pressure appears explicitly in the former, allowing us a more direct coupling between this equation and the momentum equation.

Pressure-velocity coupling. The pressure-velocity coupling in Navier-Stokes equations is indeed an issue of primarily importance, and a considerable number of approaches have been formulated over the years, together with tweaks and modifications such as under-relaxation and other stabilization techniques. However, most of the techniques to treat this coupling fall into two categories:

- coupled algorithms, in which both equations are discretized and are solved at the same time, assembled in a single linear system and solved only once;

- segregated algorithms, in which the two equations are resolved once at a time, evaluating the coupling terms contained in them using the currently available solution, typically corresponding to the previous time-step.

A segregated solver is basically made up of successive substitutions, with no theoretical guarantee of convergence of the solution. However, the smaller size of the matrices involved makes a segregated approach in general more accessible from the point of view of computational time with respect to a coupled approach. This is why most of the modern solvers for incompressible flows, like the one we are using, implement segregate approaches.

As an example, we report the steps that compose the SIMPLE (Semi-Implicit Algorithm for Pressure-Linked Equations) algorithm, one of the earliest segregated pressure-velocity coupling algorithm formulated:

1. guess the pressure field;

2. solve the momentum equation using the available pressure (momentum predictor step);

3. calculate the new pressure based on the available velocity field (pressure correction step);

4. repeat 2. and 3. to convergence.

Many assumptions need to be made in order to reach a stable solution using this algorithm and, for this reason, a series of tweaks (such as under-relaxation) and corrections (such as repeated pressure correction in the PISO algorithm) has been formulated over the years.

Discretization of VOF terms. We interrupt at this point the discussion of the finite volume discretization of the Navier-Stokes equations and we briefly discuss how to deal with the additional terms contained in the VOF equations (3.5).

The treatment we will discuss is based on the implementation of the multiphase solver contained in OpenFOAM, the CFD software we used to perform the simulations. As a bibliography reference we cite the OpenFOAM user guide [24] and a more specific manual regarding the multiphase solver interFoam [25].

3.4. DISCRETIZATION OF THE EQUATIONS - FINITE VOLUME METHOD (FVM)

As already mentioned, one of the critical issues encountered using the VOF model is obtaining a sharp resolution of the free surface. One possibility to tackle this problem, implemented inside the interFoam solver, relies on an alternative two-fluid formulation of the conventional VOF equation. In this model, an additional term, called the compression term, is introduced in the transport equation for α, originated from modeling the velocity in terms of a weighted average of water and air velocity. This term, while vanishing in the continuum formulation, has a non-zero contribution in the discretized equations, allowing for a sharp interface resolution.

More specifically, the model makes use of a two-fluid Eulerian model for the two-phase flow, solving separately the phase fraction equation for each individual phase, with:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (u_W \alpha) = 0, \tag{3.16}$$

for water, and:

$$\frac{\partial (1-\alpha)}{\partial t} + \nabla \cdot (u_A (1-\alpha)) = 0, \tag{3.17}$$

for air.

In these equations, u_W and u_A indicate the velocity for the two phases, water and air, and an additional equation is introduced expressing the velocity of the effective fluid as a weighted average of the two:

$$u = \alpha u_W + (1-\alpha) u_A. \tag{3.18}$$

Equation (3.16) can be reformulated introducing the *compression velocity*, defined as the relative velocity of the two phases $u_r = u_W - u_A$:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (u\alpha) + \nabla \cdot (u_r \alpha (1-\alpha)) = 0. \tag{3.19}$$

In this equation the last term, denominated *compression term*, is non-vanishing only in the interface region, where $\alpha \neq 0$ and $\alpha \neq 1$, and is responsible for an artificial compression of the free surface, generating a sharper one.

As already mentioned, in the continuum setting we have:

$$\nabla \cdot (u_r \alpha (1-\alpha)) = 0, \tag{3.20}$$

everywhere in the domain since the function α takes only the values of 1 (for water) or 0 (for air), with a discontinuous behavior at the interface. In a finite volume setting, however, the discontinuity is resolved in a certain number of

cells, and in those cells the contribution (3.20) becomes non-zero. An issue that needs to be discussed at this point is how the value of the compression velocity u_r can be modeled in those cells. Different options are possible, and we refer to [25] for a detailed description of the method implemented in interFoam, that is based on the gradient of the fraction of fluid $\nabla\alpha$.

Chapter 4

Numerical Experiments: Definitions and Results

In this chapter we collect, for each distinct phase that composes our optimization pipeline, the specific description of the numerical experiment's setting, the results obtained running the experiments and, finally, some considerations on the results.

The phases will be presented in distinct sections in this order:

- Geometrical Shape Parametrization;
- Full Order Model;
- Reduced Order Model;
- Optimization.

4.1 Geometrical Shape Parametrization

In this section, we present the choices made for the setting of the free form deformation. After that, we show the results obtained and make some considerations. Finally, we present and comment on the results obtained from the parameters space reduction described in Section 2.4.

4.1.1 Description of the FFD Setting

In this section, we use the notation and remarks presented in Section 2.3. In particular, we are going to define a domain D, a grid of points $P^0_{l,m,n}$ and a tensor of points' motions $\mu_{l,m,n}$ that satisfy the constraints imposed.

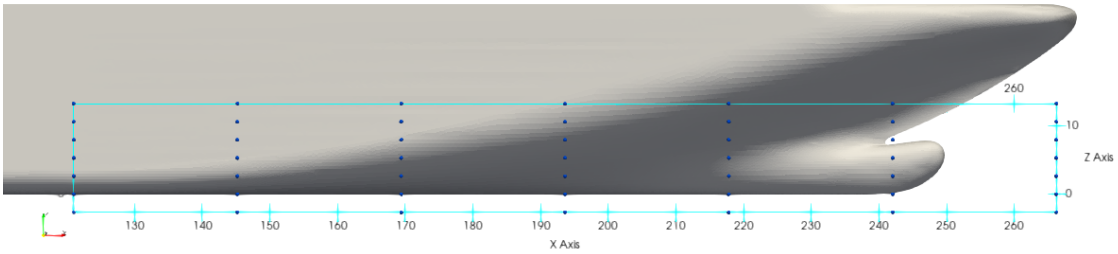


Figure 4.1: x-normal view of the set D (in cyan) and the lattice of points $P^0_{l,m,n}$ (in blue) over the undeformed hull.

The set D is illustrated in Figures 4.1 and 4.2, together with the grid of undeformed points $P^0_{l,m,n}$.We see there that the lattice is positioned, in x direction, on sections 10, 12, 14, 16, 18, 20 and 22[1]. As for z direction, we can see that a layer of points is positioned on the waterline (indicated in dark blue in Figure 4.2) with two more layers above it, and another layer is positioned on the bottom of the ship, with one more layer under it. As for y direction, we have a grid of points on the longitudinal symmetry plane of the boat, one tangent to the parallelepipedal side and one more layer outside the latter. The numbers of points for the x, y and z sides are respectively 7, 11 and 7, for a total of 539 points.

Concerning $\mu_{l,m,n}$, as anticipated, only part of the points in the lattice are displaced and, in general, the displacements are not independent.

More specifically, the layers corresponding to sections 10, 12, 20 and 22 remain fixed, together with the two upper and lower layers, the two far left and the two far-right layers and, finally, the layer over the longitudinal symmetry plane. Except for this last one, that is kept fixed to maintain symmetry, the other layers are kept fixed in order to achieve the continuity and smoothness of the deformations, required especially in the x direction where the deformation must link in a smooth way to the rest of the boat, as prescribed by the constraints.

As for the moving layers, we have that:

- the layer corresponding to section 14 moves (except for the points kept fixed) simultaneously in x direction, generating 1 parameter;

- the layer corresponding to section 16 moves (except for the points kept fixed) simultaneously in x direction, generating 1 parameter;

[1]section 22 is a fictitious section, not used in naval architecture

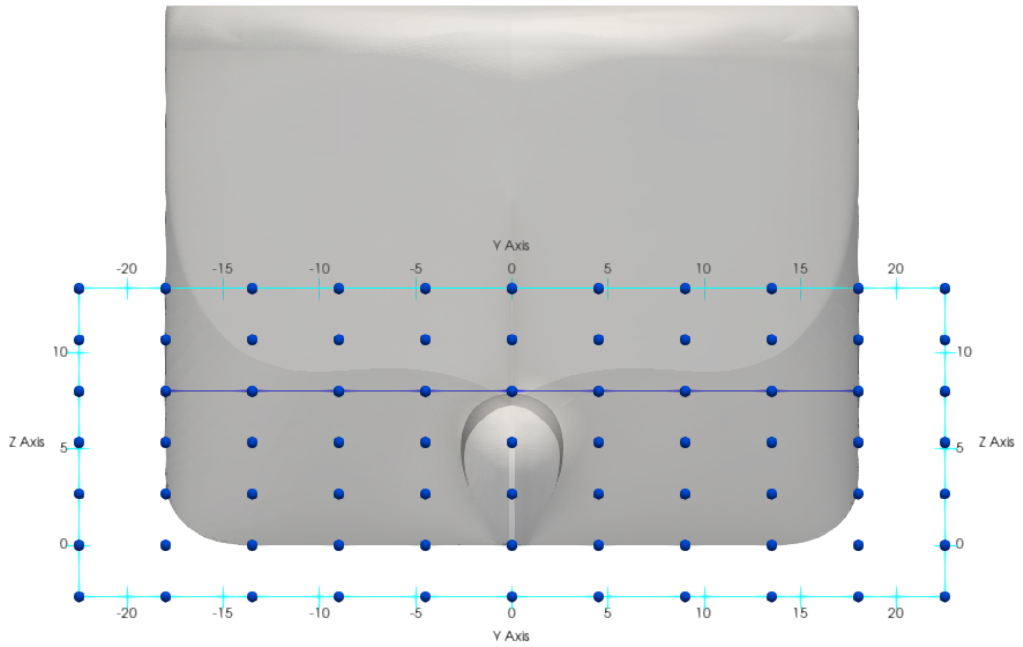


Figure 4.2: y-normal view of the set D (in cyan) and the lattice of points $P^0_{l,m,n}$ (in blue) over the undeformed hull.

- the layer corresponding to section 18 moves (except for the points kept fixed) simultaneously in x direction, generating 1 parameter, and the 3 different levels corresponding to different z-quotas move simultaneously in y direction, generating additional 3 parameters.

The total number of parameters is then 6, where the first 3 correspond to the x-movements of sections 14, 16 and 18 and the last 3 correspond to y-movements of the different z-quotas of section 18.

As for the maximum extension of the points' movements, corresponding to the domain of definition of the parameters, we express it as a fraction of the size of the FFD lattice. That is, if for example a parameter connected to an x-movement of the lattice has value β and the total length of the FFD-lattice in the x direction is B, the corresponding points move in the x direction of a quantity $\beta \times B$.

Using this definition, the ranges of the 6 parameters are:

- x-movements of section 14: $[-0.08, 0.08]$;
- x-movements of section 16: $[-0.08, 0.08]$;
- x-movements of section 18: $[-0.06, 0.06]$;

- y-movements of quota 1 of section 18: $[-0.08, 0.08]$;
- y-movements of quota 2 of section 18: $[-0.08, 0.08]$;
- y-movements of quota 3 of section 18: $[-0.08, 0.08]$.

The choices made are, as already mentioned, arbitrary and generated by a procedure of trial and error. An initial guess of the definition and range of the parameters needs to be checked and validated by considering if the output given by the FFD meets the prescribed constraints and objectives.

The reason why sections 14 and 16 do not move in the y direction is that we do not want the boat to enlarge in this direction and reach a breadth that is larger than the breadth of the parallelepipedal part. The rationale behind the movements of sections 14 16 and 18 in the x direction is instead to vary the plot of areas (Figure 2.5) maintaining the basic shape of the sections, that varies only in section 18 where we impose y movements.

4.1.2 Results and Considerations

Before presenting the results, we remark that the implementation of FFD we used is based on PyGEM, an open-source Python library (see [26]).

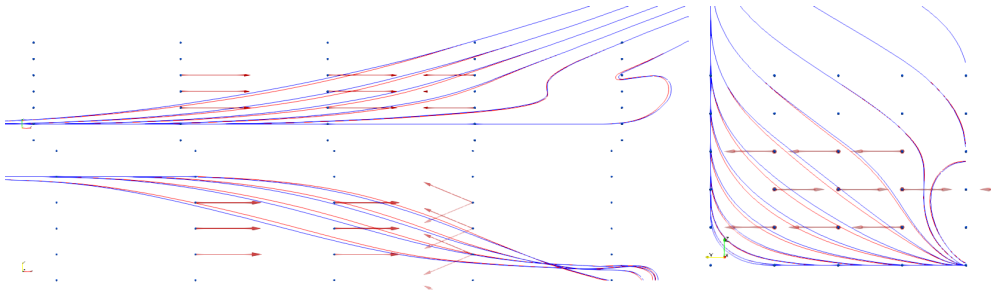
We report, in Figures 4.3 and 4.4, some deformations obtained using FFD with the setting described in the preceding section. In particular in Figure 4.3 we show, for each different deformation, the x, y and z sections, in blue for the undeformed hull and in red for the deformed one, together with the lattice points and their motions. In Figure 4.4 instead, we represent the plot of areas, in blue for the undeformed hull and in red for the deformed one. Each deformation is specified in terms of the parameter that generated it, indicated in the corresponding caption.

We can see that the new hulls generated satisfy the constraints imposed, being continuous and smooth and not enlarging on y direction over the parallelepipedal part. We can also see that, as requested, the plot of areas varies in the new shapes, where particularly interesting are the changes in the flex point of the curve.

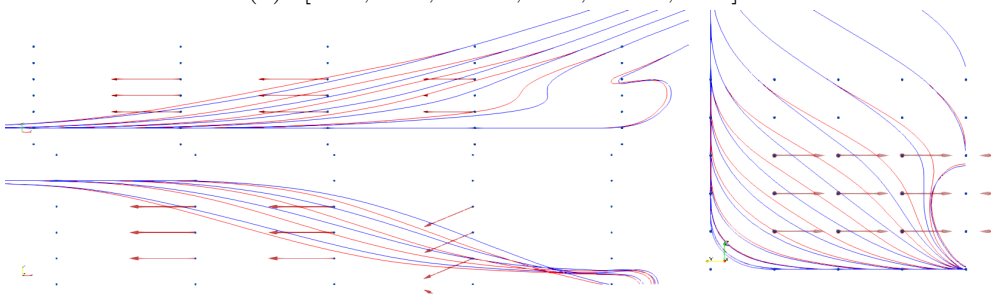
4.1.3 ROM on Shape Parametrization: Results

In this section, we present the results obtained from the parameters space reduction described in Section 2.4.

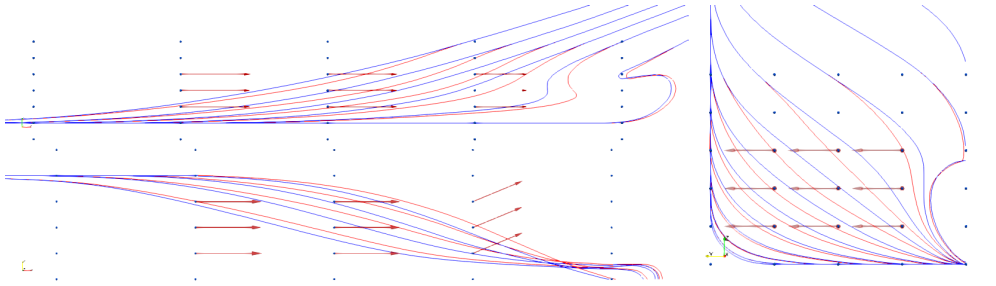
We begin by reporting, in Table 4.1, the first 10 singular values $\{\sigma_i\}_{i=1}^{10}$, ordered from the highest to the lowest, obtained from the SVD decomposition



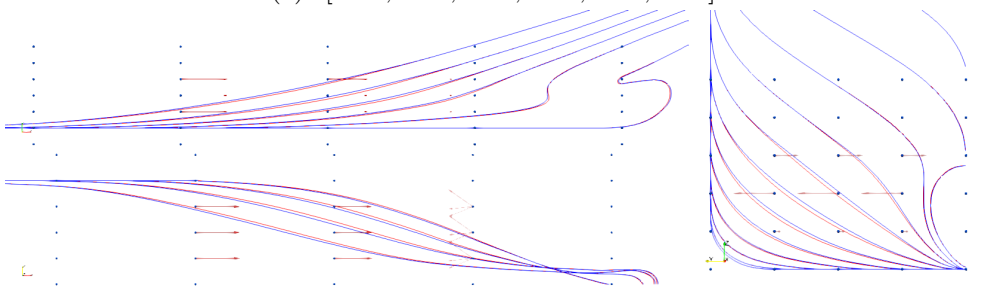
(a) $[0.08, 0.08, -0.06, 0.08, -0.08, 0.08]$



(b) $[-0.08, -0.08, -0.06, -0.08, -0.08, -0.08]$

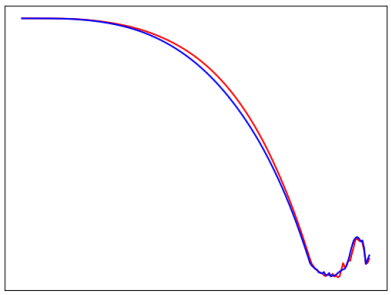


(c) $[0.08, 0.08, 0.06, 0.08, 0.08, 0.08]$

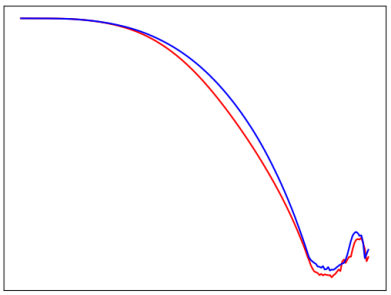


(d) $[0.0540, 0.0461, -0.0296, -0.0943, 0.0667, -0.0387]$

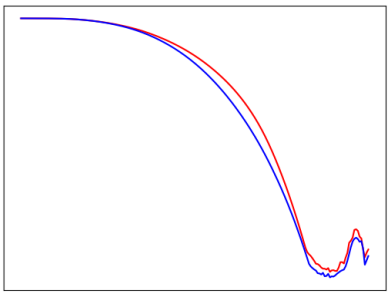
Figure 4.3: View of the x (right), y (upper left) and z (lower left) sections for the FFD deformation obtained with the value of the parameter indicated in the corresponding caption; in blue the undeformed hull, in red the deformed hull, in blue points the FFD lattice and in red arrows the corresponding motions of the points.



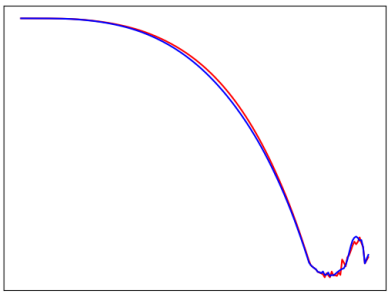
(a) $[0.08, 0.08, -0.06, 0.08, -0.08, 0.08]$



(b) $[-0.08, -0.08, -0.06, -0.08, -0.08, -0.08]$



(c) $[0.08, 0.08, 0.06, 0.08, 0.08, 0.08]$



(d) $[0.0540, 0.0461, -0.0296, -0.0943, 0.0667, -0.0387]$

Figure 4.4: View of the plot of areas for the FFD deformation obtained with the value of the parameter indicated in the corresponding caption; in blue the undeformed hull and in red the deformed hull.

σ_i
3.570×10^3
1.481×10^3
4.286×10^2
2.705×10^2
9.707×10^1
2.312×10^1
1.202×10^{-3}
1.201×10^{-3}
1.200×10^{-3}
1.199×10^{-3}

Table 4.1: Values of the first 10 singular values, ordered from the highest to the lowest, from the POD applied to the FFD deformed shapes.

applied to the snapshots matrix, together with a plot representing the values of $\frac{\sigma_i}{\sigma_0}$ on a log_{10} scale in Figure 4.5.

From these results, we can see that the SVD can detect the 6 parameters used in the FFD phase. In fact, after the 6^{th} singular value, we notice a steep descent, implying that to describe the whole set of N_s snapshots no more than 6 modes are useful and, for example, the 6^{th} mode contributes to the displacement 2 orders of magnitude less than the first one. (to understand the reason for this, see for example equation (1.11)).

To gather some insights on the deformation modes extracted in the previous step, we report:

- in Figure 4.6 the x-displacements of modes 1, 2 and 3;
- in Figure 4.7 the y-displacements of modes 4, 5 and 6.

We decided not to report the y-displacements of modes 1, 2 and 3 and the x-displacements of modes 4, 5 and 6 because they are negligible with respect to the values we reported. In fact, this can be interpreted by asserting that the first three modes correspond to x-displacements while the last three correspond to y-displacements. Moreover, we can see that, in general, the first modes correspond to larger displacements than the following ones (note that the scales of the plots are not the same).

This analysis suggests the possibility of a reduction of the parameters space dimensionality. This reduction could be performed using a small number of modes as basis functions in a basis shape approach, as described in Section 2.4. The model generated with this methodology has a parameters

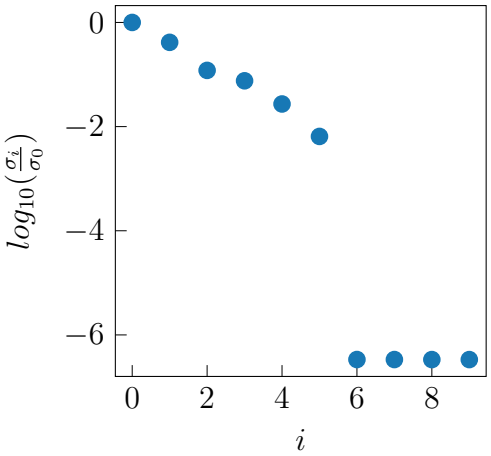


Figure 4.5: Values of $\frac{\sigma_i}{\sigma_0}$ on a log_{10} scale for the first 10 singular values, ordered from the highest to the lowest, from the POD applied to the FFD deformed shapes.

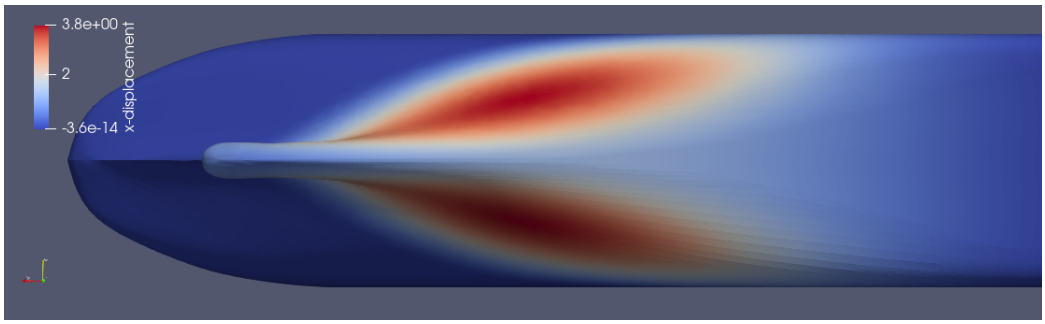
space dimensionality equal to the number of modes used, that we indicate here by N_r, and will be hence denoted as the N_r-dimensional model, in contrast to the original FFD model.

To test the ability of an N_r-dimensional model to reproduce the results of the FFD model, we consider some shapes generated by the latter and visualize, considering the frontal x-sections, the differences obtained considering only the first 3, 4, 5 and 6 modes (i.e. $N_r = 3, 4, 5, 6$). To perform this, we project the newly generated FFD shapes onto the modes and consider only the deformation obtained by truncating at the desired level. The results are illustrated in Figure 4.8 where the FFD deformation is illustrated in red and the 3, 4, 5 and 6 truncated deformations are illustrated in blue respectively on the upper left, upper right, lower left and lower right.

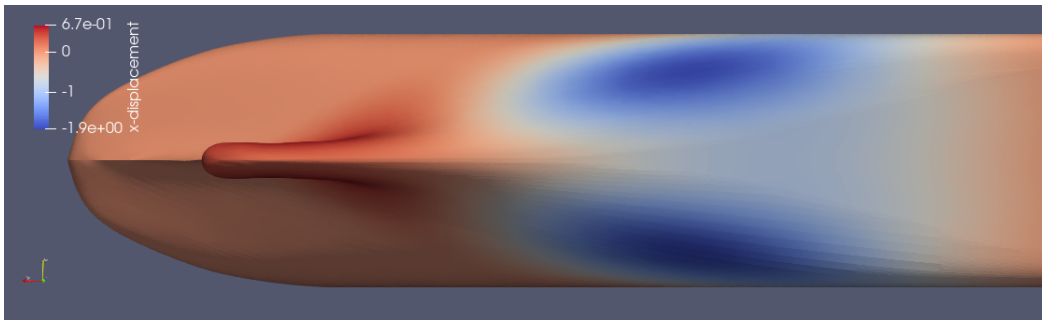
We can see that, while the first 3 modes are in general not enough to capture the deformation precisely and the total of the 6 modes are too much and generate a deformation that is indistinguishable from the original one, the first 4 modes (corresponding to the upper right image) are a good compromise between accuracy and simplicity.

Because of this, we choose a value of N_r of 4. We avoid showing a set of deformed shapes computed using the 4-dimensional model, as done for the FFD model, since the differences obtained are not noticeable to the naked eye, as we have just shown.

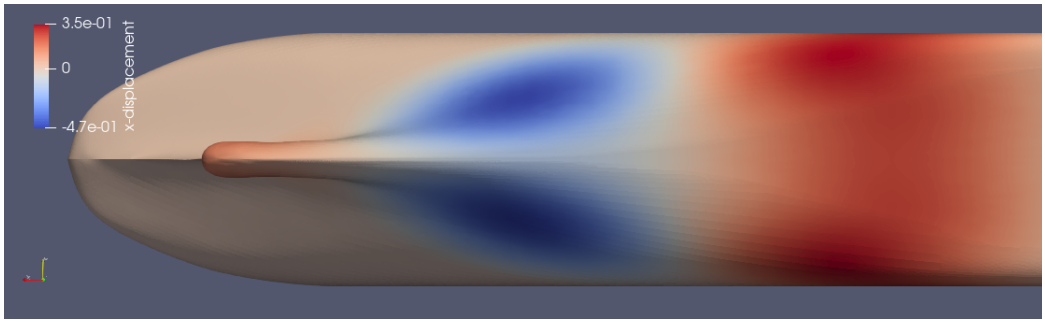
In the following sections, the two models (FFD and 4-dimensional) are tested alongside to understand which of the two is preferable over the other and for which reasons.



(a) mode #1

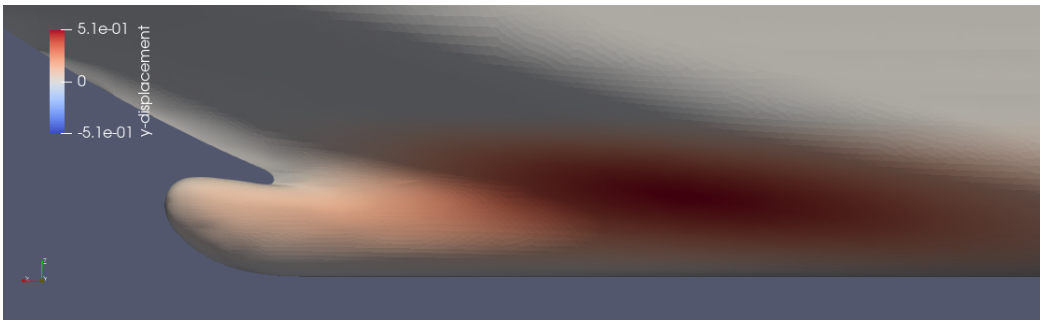


(b) mode #2

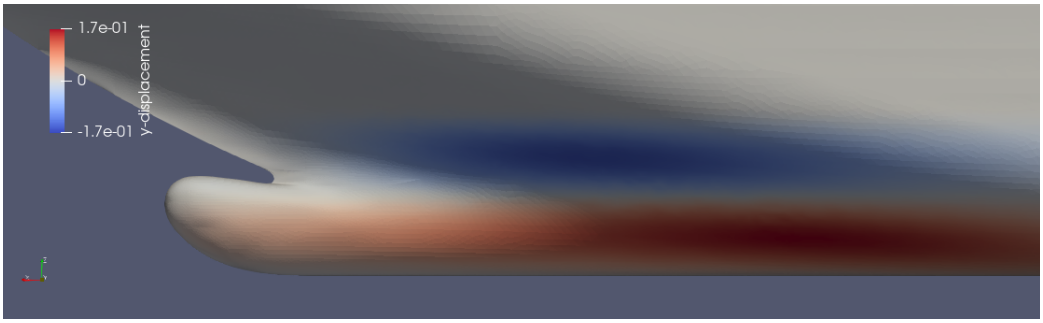


(c) mode #3

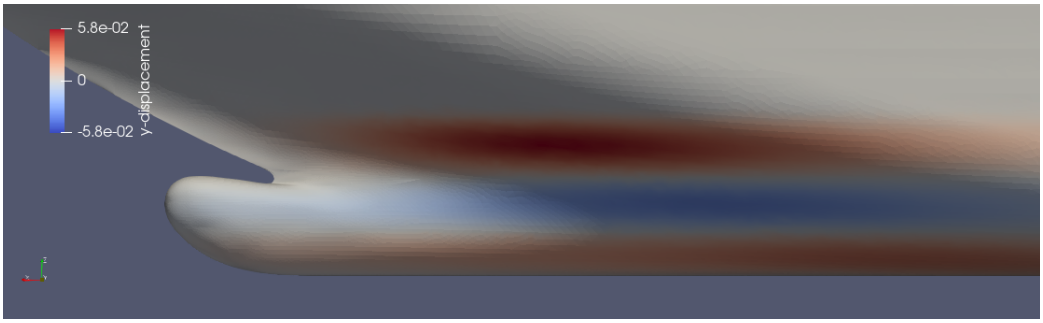
Figure 4.6: x-displacements of the first 3 modes of the POD applied to the FFD geometric deformation; the y-displacements are negligible and not reported.



(a) mode #4

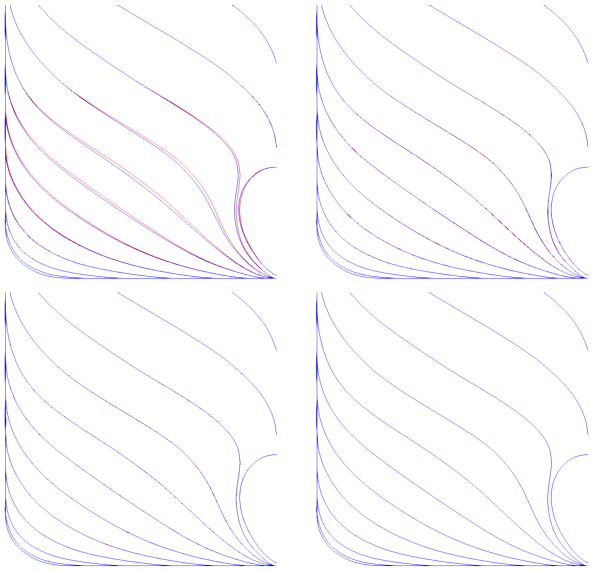


(b) mode #5

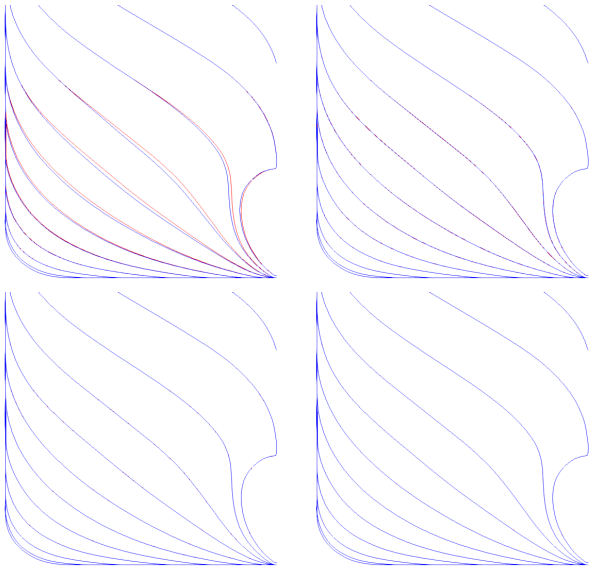


(c) mode #6

Figure 4.7: y-displacements of modes 4, 5 and 6 of the POD applied to the FFD geometric deformation; the x-displacements are negligible and not reported.



(a) $\mu = [-0.0019, -0.0651, -0.0493, -0.0205, -0.0602, -0.0553]$



(b) $\mu = [0.0154, 0.0700, 0.0182, -0.0369, -0.0571, -0.0708]$

Figure 4.8: In red the deformation with FFD (with the value of the parameters indicated in the corresponding caption) and in blue deformation with basis shape approach considering the first 3,4,5 and 6 POD modes respectively on the upper left, upper right, lower left and lower right.

4.2 Full Order Model

In this section, we specify more precisely the FOM in terms of the particular schemes employed for the discretization and resolution of the equations presented in Chapter 3. After that, we describe the computational domain in which the problem is solved and the characteristics and generation procedure of the computational mesh. Then, we declare the boundary and initial conditions imposed and, finally, we present the results obtained from the simulation.

We remark at this point that, as motivated in Section 3.1, we have run the simulations on a scaled 1:25 setting. Because of this, the numerical values of the initial and boundary conditions, the computational domain and the computational mesh are presented on this scale. We also note that all the geometrical quantities characterizing the undeformed hull reported in Table 2.1 must be reported on this scale for comparison.

4.2.1 Description of the FOM setting

OpenFOAM. As already mentioned, the FOM is implemented using OpenFOAM. OpenFOAM (Open Field Operation And Manipulation) is an open-source C++ toolbox used for the discretization and resolution of systems of Partial Differential Equations based on a Finite Volume approach. It is used in the context of continuum mechanics, especially in computational fluid dynamics, and contains a series of utilities and tools useful to define and customize user-specific solvers for different kinds of problems, together with pre-built solvers for the most common cases. We avoid discussing the technical functioning of OpenFOAM and refer the user to [22] or [24] for this.

In our problem, we used the interFoam solver, a pre-built solver implementing the VOF method. Most of the settings given to the solver are taken from the standard example contained in OpenFOAM, named DTCHull, resolving our problem with changes in the geometry and computational mesh. Because of this, we only mention some choices regarding time and space discretization and the transport and turbulence models employed.

Time/space discretization. As for the temporal discretization, we use a 1^{st} order implicit Euler scheme. The integration in time is carried out going from time 0 to 40 seconds, with an initial time-step of 0.001 seconds and adjustable time-stepping in terms of the maximum Courant number. In particular, we impose the following conditions:

- $max(Co) < 5$;

- $max(Co_\alpha) < 3$;

where $Co = \frac{u\Delta t}{\Delta x}$ is the Courant number and Co_α is an analogous quantity derived from the transport equation for α.

The time-scheme we use is implicit and thus unconditionally stable, hence we do not have theoretical restrictions on the Courant number as for the CFL condition (see [22]). However, the restriction in the Courant number is imposed in order to obtain a sufficiently accurate solution.

The adjustable time-stepping is then defined as to keep the maximum Courant numbers around the value prescribed above, with a maximum time-step allowed of 0.01.

As for the spatial discretization schemes, we only mention that for the convective term $(u \cdot \nabla)u$, one of the main bottlenecks in the discretization of convection dominated problem like ours, we have used a linear upwind scheme.

Transport and turbulence models. The numerical values of the transport properties of water and air are:

- $\rho_W = 1.09 \times 10^{-6} \frac{m^2}{s}$;
- $\rho_A = 1.09 \times 10^{-5} \frac{m^2}{s}$;
- $\nu_W = 998.8 \frac{kg}{m^3}$;
- $\nu_A = 1 \frac{kg}{m^3}$.

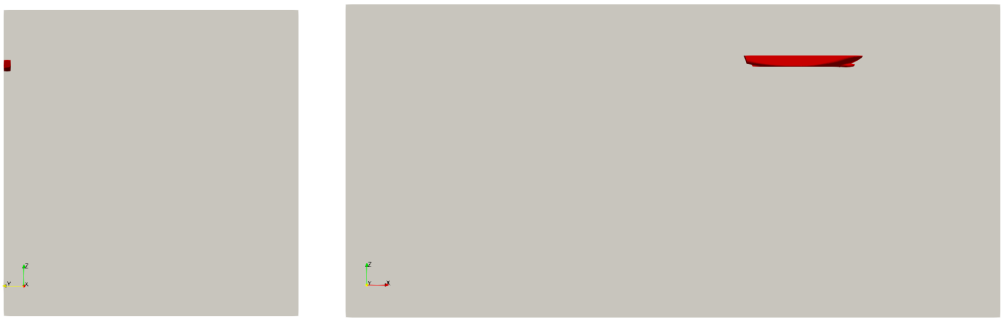
As for the treatment of turbulence, we have used a RANS k-ω SST model, with wall functions for the resolution of the boundary layer.

4.2.2 Computational Domain and Mesh

Computational domain. The computational domain, illustrated in Figure 4.9, is a three-dimensional rectangular parallelepiped with the following geometrical extensions[2]:

- x-axis: [-39, 24];
- y-axis: [-29, 0];
- z-axis: [-24, 6].

[2]for a description of the reference system, see Section 2.2



(a) (b)

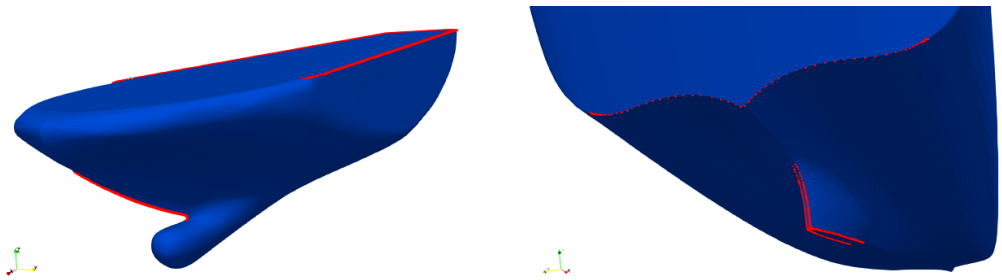
Figure 4.9: Computational domain for the full order model, x-section (left) and y section (right); in red the undeformed hull.

Computational mesh. The meshing tool used to generate the computational mesh is provided inside the OpenFOAM framework. More specifically, OpenFOAM contains a set of utilities useful for generating, modifying and checking the computational mesh. The ones we mention are:

- SurfaceFeatures: extract complex features of the geometry to be meshed;
- BlockMesh: generates a structured hexahedral mesh;
- topoSet: selects points and cells;
- refineMesh: refines selected cells;
- snappyHexMesh: generates the mesh around the object of interest.

The first step to be performed in order to generate the computational mesh is the extraction of surface features using surfaceFeatures tool. The surface features are portions of the geometry that needs some refinements in order to be captured in the correct way. They are specified by indicating some conditions on the angles generated by the various elements of the CAD file. In Figure 4.10, we can see the features that are identified for our particular hull.

After this preliminary step, we can begin with the construction of the mesh using blockMesh utility. This tool allows us to generate a structured mesh made of hexahedrons that is more and more refined, in z direction, as it approaches the waterline. The blocks generated are more extended in x and y direction and will be then refined in these directions in the next step. This choice has been made in order to capture the discontinuity of alpha at the waterline in a sharp way.



(a) (b)

Figure 4.10: Geometrical features captured by the SurfaceFeatures tool over the undeformed hull.

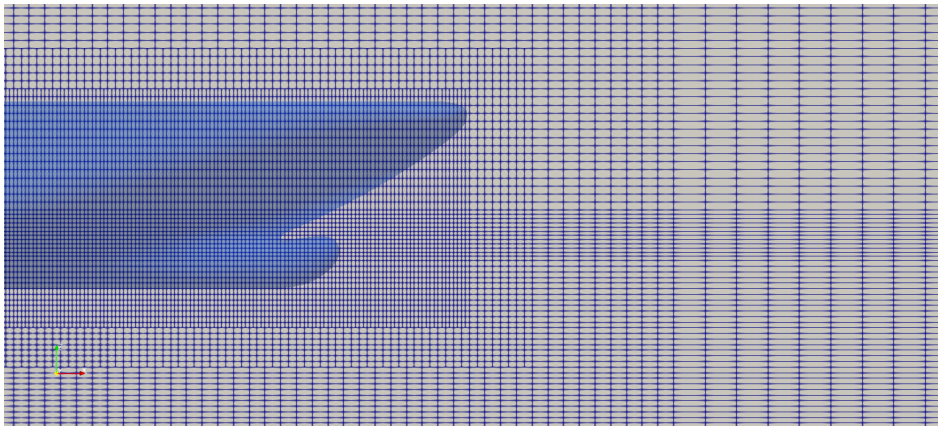


Figure 4.11: Output of the refinement phase over the undeformed hull.

The following step is the refinement of the mesh generated by blockMesh in the x and y direction in the zone in which the boat will be placed. This refinement phase is made up of 6 consecutive steps in which the cells to be refined are selected using topoSet and are refined using refineMesh. The result of the first two phases is shown in Figure 4.11 (together with the boat, still not meshed, for a reference).

After this, we run snappyHexMesh. This utility performs a series of steps. The first one consists of another refinement phase in which the cells that intersect the features obtained by surfaceFeatures are refined[3]. After these refinements, the actual geometry is imported, and all the cells that are internal to the geometry are removed from the mesh, generating what is called the castellated mesh (Figure 4.12). The next phase is the snapping

[3]one can also indicate other zones to be refined, either indicating a region like in topoSet or selecting all the cells that intersect a certain part of the surface

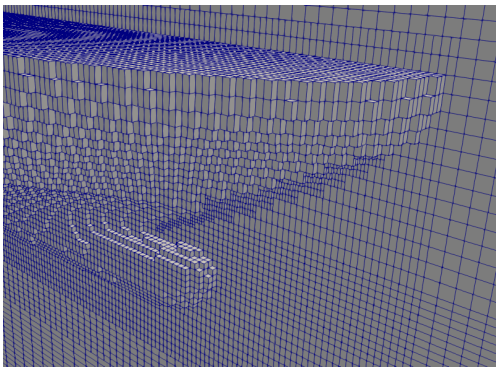


Figure 4.12: Castellated mesh over the undeformed hull.

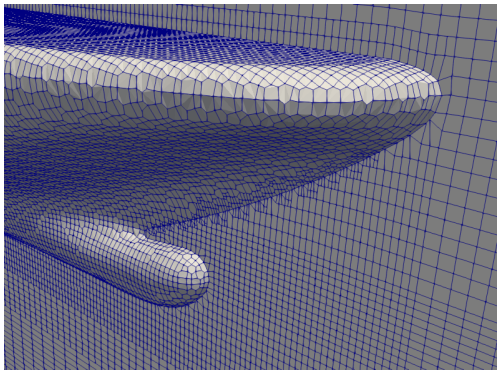


Figure 4.13: Snapped mesh over the undeformed hull.

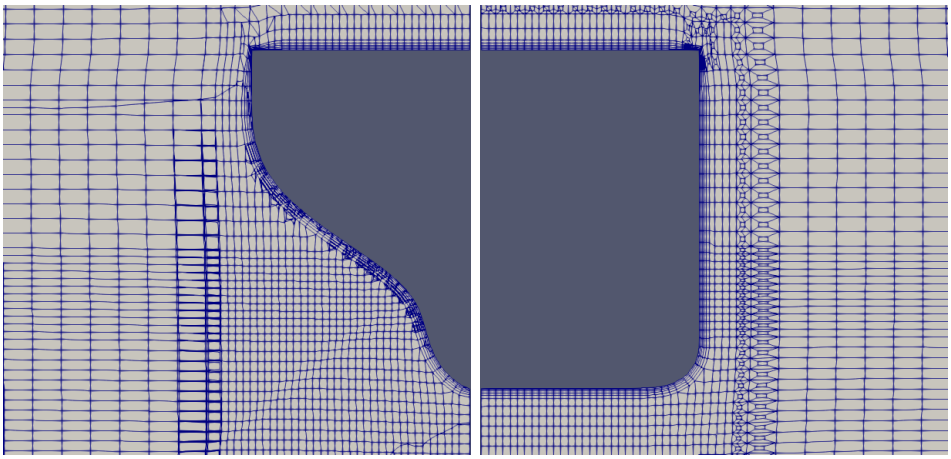


Figure 4.14: Final mesh over the undeformed hull, view from section 17 (left) and 10 (right).

phase in which the points of the cells that intersect the surface are moved so that the mesh follows the exact geometry of the surface (Figure 4.13). The last phase consists of the boundary layer cells addition (Figure 4.14).

4.2.3 Initial and boundary conditions.

We report here the boundary and initial conditions we have imposed on the system (3.5).

Before proceeding, we remark that, in the following calculations, we will need to consider the value of the kinematic viscosity ν of the fluid. In our case, however, we are considering a multiphase fluid and, in general, the value of ν will depend on the point in which it is calculated. For the sake of

simplicity, we will consider as a reference the value associated to the water phase.

As for the boundary conditions, we have imposed the following:

- in the **inlet** of the domain:
  - constant velocity (Dirichlet BC);
  - *fixed flux* condition on the pressure (deducted of the hydrostatic componend) $p-\rho gh$, where the value to be imposed is determined by checking that the flux generated is compliant with the condition on the velocity;
  - discontinuous fixed profile for α, where below the level of the unperturbed waterline the value is 1 (water) and above is 0 (air);
  - fixed values for the turbulent variables ω, k and ν_T;
- in the **outlet** of the domain:
  - constant average velocity;
  - zero gradient condition on the pressure $p - \rho gh$ (homogeneous Neumann BC);
  - *variable height flow rate* condition to α, where we indicate a lower bound (equal to 0) and an upper bound (equal to 1) and:
    - if α is greater than the upper bound, we apply a fixed value condition, with a uniform level of the upper bound;
    - if α is inside the range, we apply a zero-gradient condition;
    - if α is lower than the lower bound, we apply a fixed value condition, with a uniform level of the lower bound;
  - zero gradient condition for ω, k and ν_T;
- in the **bottom** and in the two **sides** of the domain, symmetry conditions for all the variables;
- in the upper plane of the domain, denoted as **atmosphere**:
  - *pressure inlet outlet velocity* condition for the velocity, imposing a zero gradient on the outlet and a fixed value in the inlet;
  - total pressure of 0 for the pressure $p - \rho gh$;
  - α equal to 0 (air);
  - zero gradient condition for ω, k and ν_T;

- finally, in the **hull**:
  - fixed value of 0 for the velocity (*non-slip condition*);
  - fixed flux condition on the pressure $p - \rho gh$;
  - zero gradient condition on α;
  - for the turbulent variables ω, k and ν_T, conditions depending on the *wall functions*.

The initial conditions imposed are defined as to be compliant with the inlet. In particular, we have a discontinuous profile for the variable α, fixed values for the velocity and the turbulent variables, and $p - \rho gh$ equal to 0.

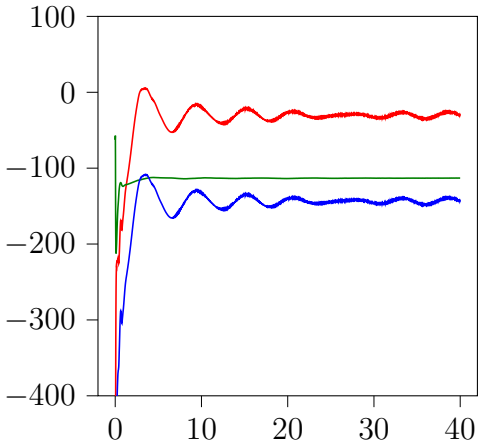
While most of the conditions declared here are quite standard for a ship resistance simulation, particularly interesting is the choice we have made of imposing an initial non-zero velocity on the whole domain. In fact, this choice results in an initial non-realistic strong force applied to the vessel that dissipates in few seconds, returning the real physical force. Alternatives to avoid this are possible, like for example the imposition of an initial velocity equal to 0. This would remove the non-physical behavior encountered in the first seconds of our simulations but would result in longer times needed to reach the steady state. Another interesting possibility is to solve a potential problem and to use the results obtained (for velocity and pressure) as initial conditions.

We now specify the numerical values we have assigned to the velocity and turbulent variables at the inlet.

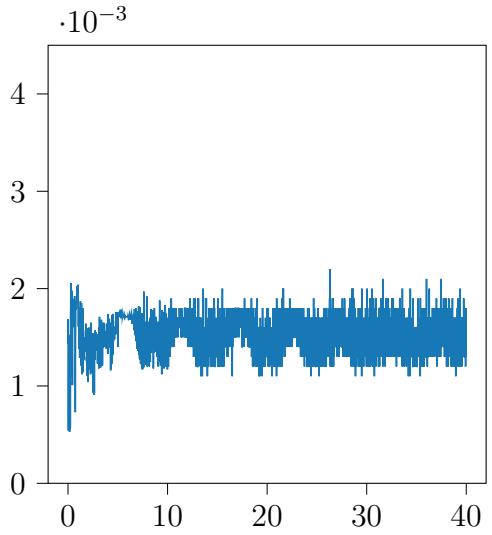
The inlet velocity imposed is $v = -2.26336\frac{m}{s}$, obtained from a full scale velocity of 22 knots (corresponding to approximately $11.3178\frac{m}{s}$). Considering this velocity, the length between perpendiculars (LPP) of the undeformed hull, equals to $9.68m$ and the kinematic turbulence of the water, equal to $1.09 \times 10^{-6}\frac{m^2}{s}$, the Reynolds number generated is around 2×10^7. This number is considerably lower than the Reynolds number generated by the model at the real scale, that is around 2.6×10^9 ($v = 11.3178\frac{m}{s}$, $LPP = 242m$, $\nu = 1.09 \times 10^{-6}\frac{m^2}{s}$), as we have anticipated in Section 3.1 . We also report the Froude numbers calculated for the scaled model and for the full-scale model (equal for hypothesis), which is 0.232.

As for the boundary conditions in the inflow for the turbulent variables ν_T, ω and k, they are computed starting from the freestream velocity $v = -2.26336\frac{m}{s}$, the turbulence intensity, which we imposed to 1%, and the eddy viscosity ratio ν_T/ν, imposed to 10. The values of the turbulent variables associated with these conditions are thus:

- $\nu_T = 1.09 \times 10^{-5}\frac{m^2}{s}$;



(a) Value of the resistance over the undeformed hull as a function of the simulation time; in red the pressure component, in green the viscous component and in blue the sum of the two.



(b) Value of the adaptative time-step as a function of the simulation time

Figure 4.15

- $\omega = 70.4972 \frac{1}{s}$;
- $k = 7.6841 \times 10^{-4} \frac{m^2}{s^2}$.

4.2.4 Results and Considerations

We report in this section some plots and figures obtained by simulating the FOM over the undeformed hull.

The simulations have been run in parallel on 40 processors by dividing the domain in equally sized chunks. The computational time required by the runs is around half an hour for the generation of the computational mesh (operated serially) and around 8 hours for the simulation.

We begin by reporting, in Figure 4.15, the plot of the drag, decomposed in the two contributes of pressure and viscosity, over the 40 seconds of the simulation, together with the value of the adaptative time-step. The values of resistance reported are equal to two times the values obtained in the simulations because of the symmetry assumption.

From the plot 4.15a, we can observe a very high value of drag in the first seconds of simulations, especially in the pressure component. This is due to the impulsive initial conditions imposed on the velocity field, as anticipated in Section 3.3. However, as anticipated there, just a few seconds of simulation

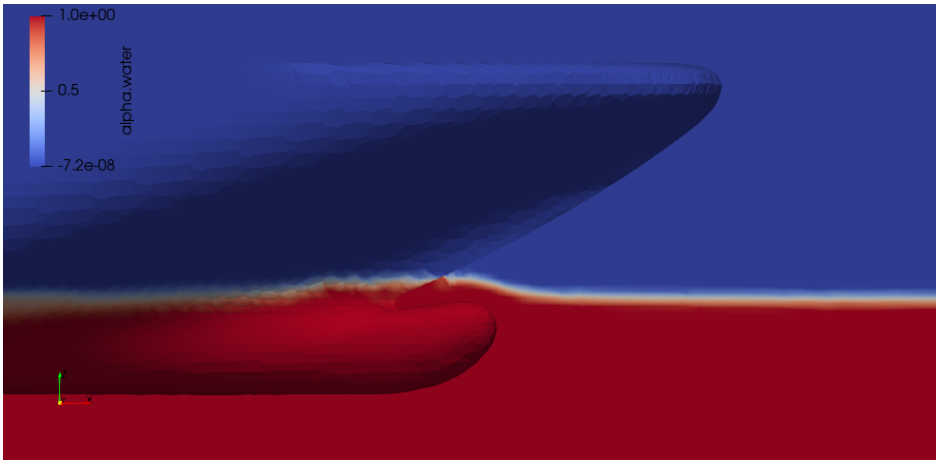


Figure 4.16: Value of the fraction of water α in the front of the ship; blue corresponds to air, red to water and gray to interface points.

are sufficient in order to arrive to a fluctuating state in which the pressure component oscillates around a mean value and, eventually, converges to it. The final time of the simulation, however, is too short to observe this convergence and we rely on DMD in order to obtain it. A discussion on the application of DMD to our test case is postponed to Section 4.3.3.

We now report some figures that show the value of the fraction of water α in order to describe the behavior of the ship in terms of wave generation. In Figure 4.16 and 4.17 we report the value of α on the front and on the back of the ship and in Figure 4.18 we report the value of α on a z-normal plane placed on the waterline. All the figures correspond to the final time of integration (40 seconds), and we have that a value of α of 0, corresponding to air, is represented in blue, a value of 1, corresponding to water, is represented in red, while the intermediate values, corresponding to interface points, are represented in gray.

Finally, we show in Figure 4.19 the value of the pressure p subtracted of the hydrostatic component ρgh on the front of the ship. In this figure, red values correspond to overpressure, observed in particular in the frontal part of the bow, and blue values correspond to pressure loss, associated with increases in the velocity of the flux and observed in the lateral part of the bow.

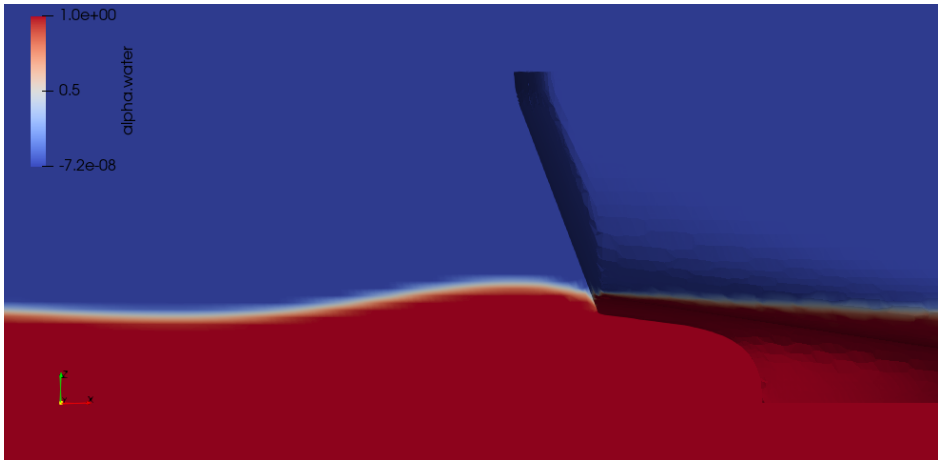


Figure 4.17: Value of the fraction of water α in the back of the ship; blue corresponds to air, red to water and gray to interface points.

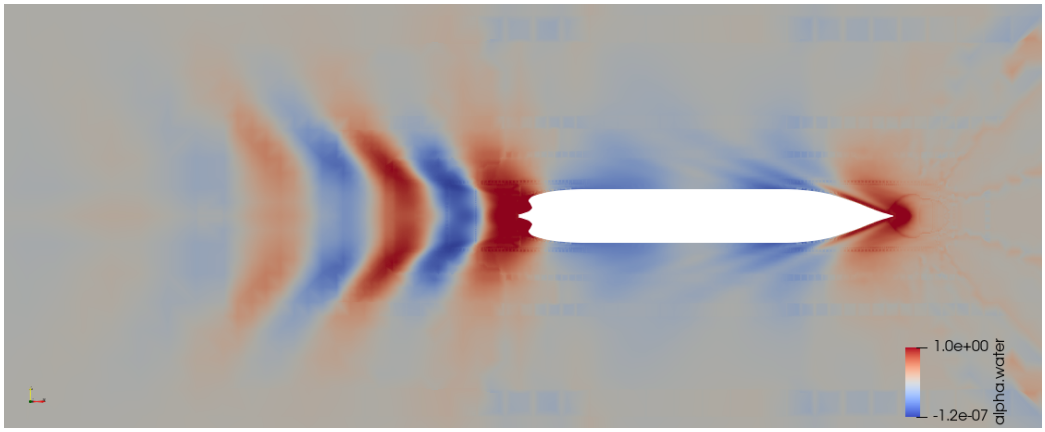


Figure 4.18: Value of the fraction of water α in the z-normal plane placed on the waterline; blue corresponds to air, red to water and gray to interface points.

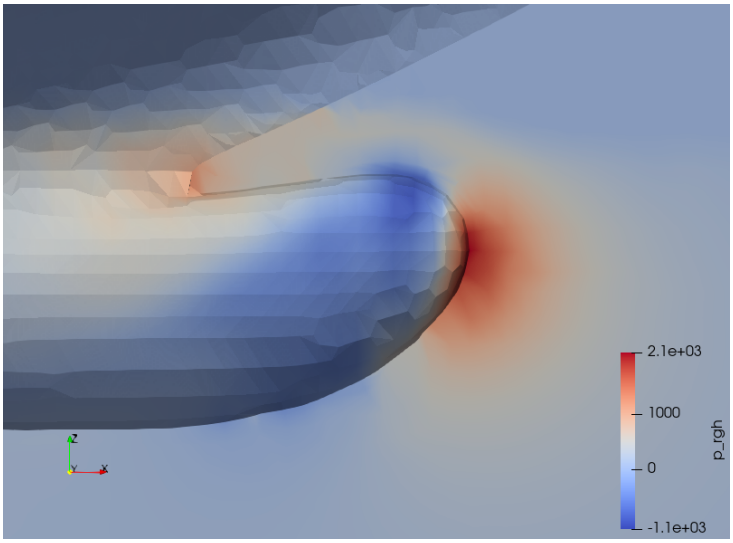


Figure 4.19: Value of the pressure p subtracted of the hydrostatic component ρgh on the front of the ship.

4.3 Reduced Order Model

In this section, we present some possible variations for the reduced order model and compare them against the full order model in order to establish the best setting. This optimal ROM will be then used in the optimization process in the next section.

We begin by defining how the ROM is applied to our test case. After that, we present the terms in which the comparison is made, and we explicit the differences between the different ROMs. Next, we present the results of the tests, making some considerations. Finally, we give some details on the use of DMD in our test case.

All of the possibilities described below have been tested both on the FFD and the 4-parameters models presented in Section 4.1.

4.3.1 Description of the ROM Setting

In Chapter 1, we presented reduced order methods in a general setting. In our case, however, we decided not to apply the decomposition $u_N^{\mathcal{N}} = \sum_{i=1}^{N} \xi_i \chi_i(x)$ to the whole solution field (pressure and velocity), as would be needed in the case an intrusive approach is used. Instead, we considered as a snapshot the field of total resistance computed over the hull at the steady state, defined as:

$$R = \tau_x \rho - p n_x, \tag{4.1}$$

where τ_x is the x component of the total (viscous+turbulent) tangential stresses, ρ is the density of the fluid (computed as in (3.7)), p is pressure and n_x is the x component of the normal to the surface.

A technical remark must be made at this point since, because of the fact that each deformed hull correspond to a different computational mesh, we have that each snapshot have a slightly different dimensionality[4] $\mathcal{N}$. The approach we used to tackle this problem is to project the field of total resistance of each solution to the deformed hull shape, expressed as a CAD file, and consider this projection as a snapshot. In fact, all the CAD files, despite being different because of the application of FFD, have the same number of nodes.

To test the effectiveness of a ROM, we decided to run 100 high fidelity simulations (for both the FFD and the 4-parameters models, for a total of 200 simulations) and to use them to both train and test the models. In particular, of the 100 total simulations, 80 are used as training set, forming the snapshots matrix over which the SVD is performed, and the remaining 20 are used as a test set. This separation between train and test set is performed randomly and repeated 15 times, considering the final average as the performance of the corresponding method.

The quantity used to evaluate the performance of a reduced order model is the relative L^2 error between the ROM and the FOM of the field of total resistance computed over the hull, calculated as:

$$\left(\frac{\int_A (u^{\mathcal{N}} - u_N^{\mathcal{N}})^2 dA}{\int_A u_N^{\mathcal{N}2} dA} \right)^{\frac{1}{2}}. \tag{4.2}$$

The variations in the ROM setting that generate the different models we will test are relative to:

- the number of modes used N;
- the interpolation scheme used in the PODI to compute the modal coefficients $\{\xi_i(\mu)\}_{i=1}^N$.

As for the number of modes used, we have that, if the snapshots matrix contains a total of N_s snapshots, and if the dimensionality of the snapshots is $\mathcal{N} > N_s$, applying a POD we extract a total of N_s modes $\{\chi_i(x)\}_{i=1}^{N_s}$. We can then choose a number $N \leq N_s$ of modes and express the reduced solution as:

$$u_N^{\mathcal{N}} = \sum_{i=1}^{N} \xi_i \chi_i(x). \tag{4.3}$$

[4]note that $\mathcal{N}$ in this case is unrelated to the linear system coming from the discretization of the differential equations, and only represents the dimensionality of the snapshots

The reason why we may want not to use all the modes to express our reduced solution is that, despite losing some accuracy disregarding the part of energy contained in the truncated modes $\{\chi_i(x)\}_{i=N+1}^{N_s}$, we may gain accuracy reducing the error related to the interpolation process. In fact, it is well known in numerical analysis that the interpolation error is strongly related to the dimensionality of the space in which the interpolation takes place. Our aim is then in general to reach a good tradeoff between the two errors introduced by the truncation of the modes and by the interpolation, considering the different models generated for different values of N and selecting the best one in terms of relative error.

Regarding the interpolation scheme, we considered three different possibilities:

- linear interpolation;
- interpolation with Radial Basis Functions (RBF);
- interpolation with Gaussian Processes (GP);

We remark that, for the last two methods, we are not considering an interpolation problem but rather a data approximation problem, using, for example, maximum distance techniques (for RBF) or regression techniques (for GP). However, we continue to refer to the term interpolation in reference to the acronym PODI (Proper Orthogonal Decomposition with Interpolation).

We now proceed to briefly describe the interpolation techniques just mentioned. Our aim is neither to present the implementation details, nor to discuss the general theory behind these techniques, but instead to give some insights. As for the practical implementation, for the linear interpolation and the RBF interpolation we relied on the implementations contained in the Python package scipy (see [27]), while for the Gaussian process interpolation we used the Python package GPy (see [28]).

Linear interpolation. The linear interpolation is conducted in a two-stage approach, in which initially a triangulation is built on the input data using Delaunay techniques, and the inference of the value of a new point x_j is made by applying linear barycentric interpolation on the triangles.

The main downside of using this approach is that only points belonging to the convex hull of the input data can be inferred. Because of this, it is necessary that at least all the vertices of the parametric space are contained in the training set, that in the case of an N dimensional parallelepipedal parameter space implies that the training set must be at least composed of 2^N elements.

RBF interpolation. In radial basis function interpolation, the interpolant is built as a weighted sum of Radial Basis Functions (RBFs).

A RBF is a function ϕ whose value depends only on the distance of the point from the origin of the axes, that is $\phi(\mathbf{x}) = ||\mathbf{x}||$, or from some other point y, that is $\phi(\mathbf{x}) = ||\mathbf{x} - \mathbf{y}||$. The distance in our case is expressed as Euclidean distance.

To perform the intepolation we must first define the shape of the RBF to be used. In our case we used a multiquadric function:

$$\phi(\mathbf{x}) = \sqrt{\frac{||\mathbf{x} - \mathbf{y}||_2}{\epsilon} + 1}, \tag{4.4}$$

where the centers $\mathbf{y}$, radius r and ϵ needs to be defined for each basis function.

The RBF interpolant is then constructed by obtaining the values of the coefficients that multiply each basis function in the final weighted sum. These values can be determined using different kinds of techniques, from the imposition of the interpolation condition to more sophisticated techniques that involve, for example, maximum distance criteria.

In particular, inside the class implemented in Scipy, there is the possibility to indicate a parameter, named *smoothness*, that, as the name suggests, quantifies the smoothness of the final approximation obtained. A value of smoothness equal to 0 corresponds to plain interpolation, generating a highly oscillating approximation, and higher values of the smoothness parameter correspond to smoother and smoother approximations.

GP interpolation. GP interpolation, or better GP regression, is an approximation technique in which the data to be interpolated is modeled by a Gaussian process. The interpolant is built, similarly to the RBF case, as a weighted sum of basis functions (we used RBF functions also for the GP case) and the method gives the best linear unbiased prediction of the data to be predicted.

We avoid going into the details of the method since we would need to introduce many statistical concepts. We refer to [29] for an in-depth presentation of this class of techniques.

4.3.2 Results and Considerations

Before presenting the results, we remark that the implementation we have used for the PODI is based on EZyRB, an open-source Python library for data-driven model order reduction (see [30]).

We report here, for both the 4-parameters in Figure 4.20 and the FFD model in Figure 4.21, two plots representing the average relative L^2 error

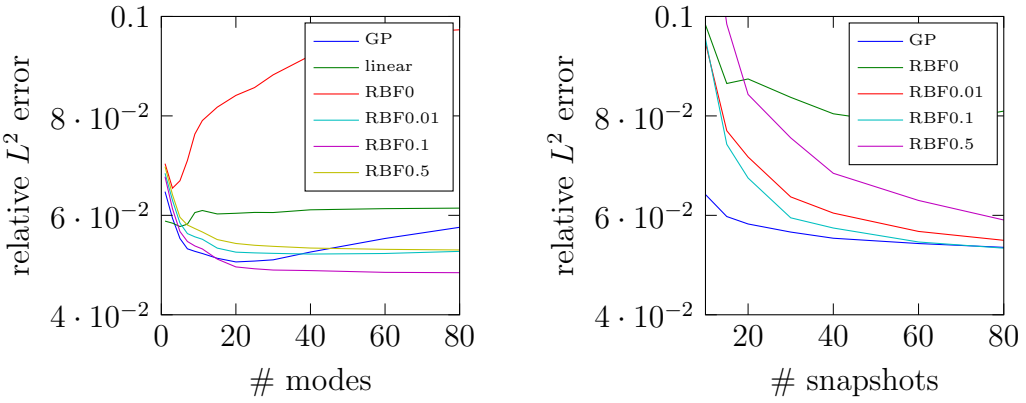


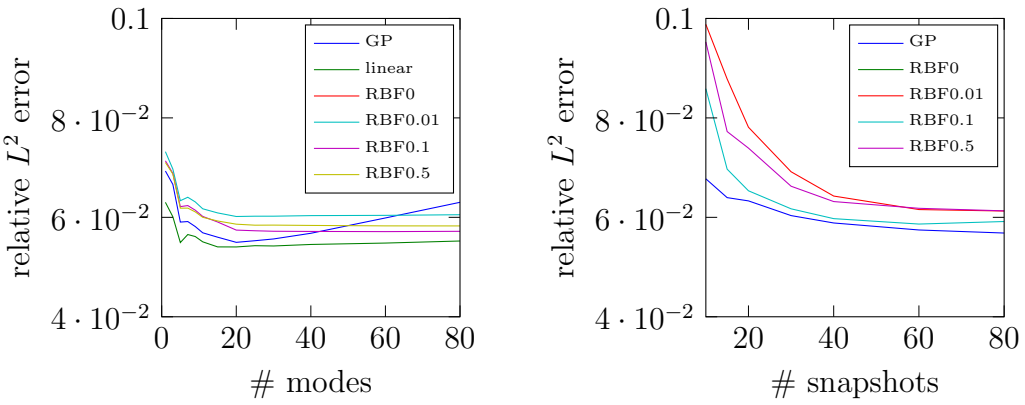
Figure 4.20: Average relative L^2 error between the FOM and the ROM as a function of the number of modes (left) and the number of snapshots used (right). Shapes generated using the FFD model.

(4.2) as a function of the number of modes (on the left) and the number of snapshots used (on the right). In the former, we have used the total number of snapshots available to train the model (i.e. 80) and varied the truncation parameter N, from 1 to 80, and the interpolation scheme, considering the linear, the GP and the RBF with four different smoothness parameters($= 0, 0.01, 0.1, 0.5$). In the plot on the right instead we considered the different models generated by considering only N_s snapshots, with N_s going from 10 to 80, with GP and RBF interpolation schemes[5] and with $N = 10$.

A first interesting result to be noted is that the 4-parameters model does not perform better than the FFD model. This was not expected a priori since a reduction of the dimensionality of the parameters space should have benefic effects on the performances of the ROM model. However, this result can be explained by the fact that, in transforming the parameters space, we are not in any sense simplifying or reducing the complexity of the design space generated, being the shapes obtained from the 4-parameters model quite identical to the ones generated by the FFD model. In this sense, an improvement of the ROM model could be obtained only by effectively reducing the design space, considering more simple deformations or more strict constraints and, consequently, obtaining a possibly worst optimum from the optimization process.

To proceed, from figures 4.20a and 4.21a we can see that, as expected, the

[5]we have not considered the linear scheme since, as already mentioned, it requires at least all the vertices of the domain to be included in the snapshots matrix



(a) (b)

Figure 4.21: Average relative L^2 error between the FOM and the ROM as a function of the number of modes (left) and the number of snapshots used (right). Shapes generated using the 4-parameters model.

error tends to be large for low values of N, because of the truncation error, and for high values of N, because of the interpolation error, the optimal value for the truncation being around 20. To interpret this result, we show, in Figure 4.22, a plot representing the singular values obtained from the SVD decomposition of the snapshots matrix, ordered from the highest to the lowest and expressed as a ratio $\frac{\sigma_i}{\sigma_1}$ in a log_{10} scale. Considering this plot and equation (1.11), we can see that the modes after the 20^{th} contribute to the reduced solution around two orders of magnitude less with respect to the first one.

Regarding the error as a function of the number of snapshots used, displayed in figures 4.20b and 4.21b, we can see that it posseses a decreasing monotone behavior. While, as expected, the optimal number of snapshots to be used is 80, an interesting information we get from these plots is that using a number of snapshots considerably lower, even halved, the error obtained is almost the same. This is very interesting from the point of view of industrial applications and is only possible when considering non-linear interpolation schemes.

To continue with the analysis, we can see from figures 4.20a and 4.21a that the best performances are returned by the GP and RBF (with smoothness 0.1) interpolation schemes, with a small difference between these two.

Based on these considerations, we now define the optimal ROM setting we will use in the optimization process:

- model for the generation of the deformed shapes: FFD 6-parameters

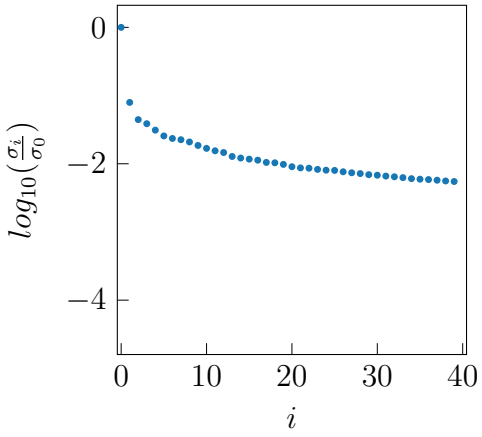


Figure 4.22: Values of $\frac{\sigma_i}{\sigma_0}$ on a log10 scale for the first 40 singular values, ordered from the highest to the lowest, from the POD applied to the 6-dimensional parametric problem.

model;

- interpolation scheme: GP interpolation;
- N_s: 100;
- N: 20.

To quantify the profit in computational time, we have that while, as mentioned in Section 4.2, the FOM requires around 9 hours to run, the online-phase of the ROM model defined above is almost real-time, requiring less than one second. Obviously, we need to consider the time required to perform the SVD (around 1 minute) and, most importantly, the time required in the offline phase for the computation of the snapshots, where in our case 100 CFD solutions need to be computed. However, this number is way lower than the number of CFD simulations that should be calculated in an optimization process, this number being around 3000. We also mention here that the time required by the deformation process (via FFD), by the dynamic mode decomposition and by the optimization process is negligible with respect to the time required by the computation of the offline phase of POD.

Finally, we show in Figure 4.23 the total resistance over the bulbous bow, computed for a new value of the parameters, for the FOM (on the left) and the ROM (on the right).

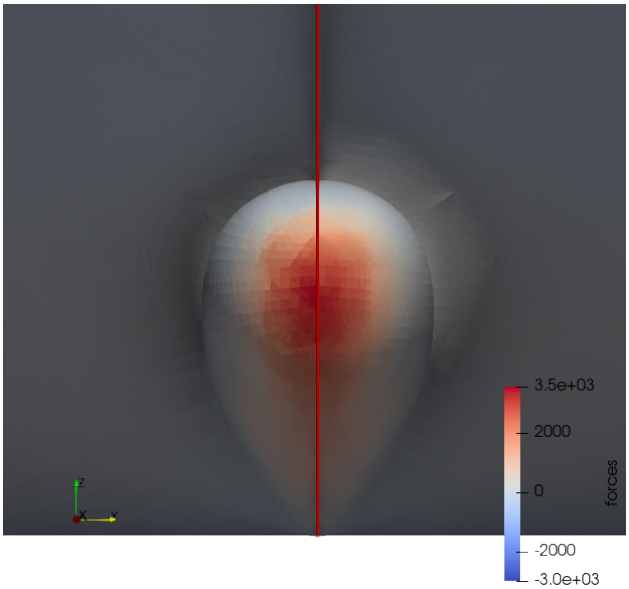


Figure 4.23: Value of total resistance over the bulbous bow for the FOM (on the left) and for the ROM (on the right).

4.3.3 Dynamic Mode Decomposition: Setting and Results

In this section, we present some considerations on the use of Dynamic Mode Decomposition in our test case, as introduced in Section 1.5, and discuss the results obtained. Before proceeding, we remark that the implementation we have used for the DMD is based on PyDMD, an open-source Python library (see [31]).

We can consider two possible choices for the use of DMD:

- apply DMD to the solution fields (velocity and pressure) defined over the whole computational domain, obtain the asymptotic solution and calculate from this the associated total resistance over the hull;

- calculate the total resistance field over the hull for each time-step and apply DMD to it, obtaining the asymptotic total resistance

Since we are not interested in the solution fields but only on resistance and since the total resistance field is much smaller than the solution fields, resulting in a much faster DMD, we decided to opt for the second approach.

More precisely for the time series $\{\mathbf{x}_0, ..., \mathbf{x}_n\}$ we considered the total resistance field calculated from second 20 to 40 every 0.5 seconds.

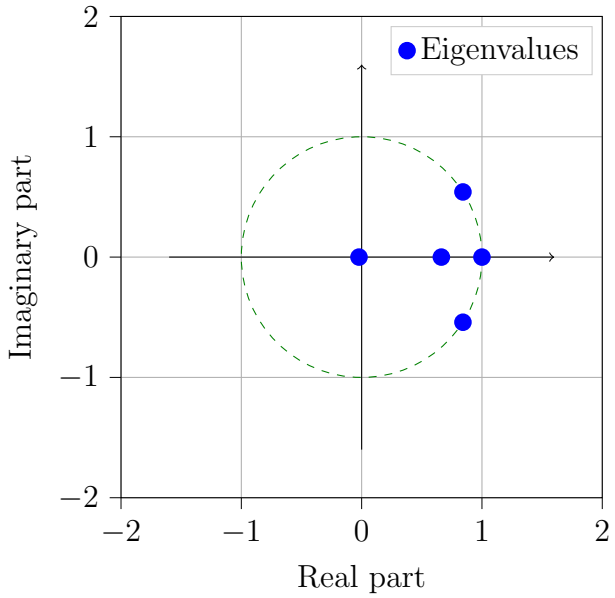


Figure 4.24: First 5 DMD eigenvalues.

Using this, we can assemble matrix X and Y and compute the DMD modes (we considered the projected modes). In Figure 4.24 we show the first 5 eigenvalues obtained by DMD in the complex plane.

In order to obtain the asymptotic state of our system, the standard approach consists of reconstructing its dynamics using the exact or projected DMD modes, as presented in Section 1.5. In this work, however, we decided to test an alternative approach, that consists of considering the dynamics described by only the first DMD mode. This is justified by the fact that, as shown in Figure 4.24, the first DMD mode has imaginary part 0 and thus represents a non-oscillating dynamics. Because of this, we assumed that the system obtained by the reconstruction process performed considering only the first mode represents the asymptotic behavior of our system and that the other modes made up the oscillations we can observe in the simulated resistance.

4.4 Optimization

In this section, we present the choices we have made for the optimization algorithm used to explore the design space. In particular, after a brief introduction to the general class of algorithm we have used, namely genetic algorithms (GA), we present the setting we have used for our test case. Fi-

nally, we show the results of the optimization process.

In exploring the parameter space, we will make extensive use of the tools we have discussed previously in this work. In particular, for every new value of the parameter, we need to compute the new deformed hull, using FFD, and we need to compute the value of the resistance around the new deformed hull using PODI. Finally, the optimized hull will be validated using the FOM.

4.4.1 Genetic Algorithms

As already mentioned, the family of algorithms we decided to use for the optimization process are called Genetic Algorithms (GA). The main idea behind these methodologies is the application of natural selection principles in which, starting from an initial random population, we breed new generations using mating and mutation mechanisms, selecting at each step the best individuals.

Going a bit deeper into the details of GAs, we begin by defining what is an individual, that in our case is represented by a vector of parameters[6] $\nu \in \mathbb{R}^N$, where N is the dimensionality of the parameters space. The first generation G_0 is then composed of n_0 individuals generated varying the parameters randomly in their ranges (as defined in Section 4.1).

Then we use a set of rules to breed new generations and to select the best individuals inside each generation. In particular, we need to define the following functions:

- *evaluate*: a function that takes an individual and returns the value of the objective function;
- *mate*: a function that takes two or more individuals and returns one or more new individuals that are generated mixing in some way the characteristics of the parents;
- *mutate*: a function that takes an individual and returns a new individual that is a random mutation of the first one;
- *select*: a function that takes a collection of individuals, usually a generation G_i, and returns the subset of the best individuals in the population based on the previously defined function *evaluate*.

The specific way a GA uses these functions depends on the particular algorithm considered. In our case, we used an algorithm named "$\mu + \lambda$", that can be summarized as follows:

[6] in this section, to indicate a value of the parameters, we will use the greek letter ν in substitution to μ

1. Initialize $G_0 = \{\nu_j^0\}_{j=0}^{\mu}$;
2. for $i = 0, 1, ..., n_{max}$:
   (a) Starting from G_i, generate a collection of λ new individuals $G_{i+\frac{1}{2}} = \{\nu_j^{i+\frac{1}{2}}\}_{j=0}^{\lambda}$ using *mate* and *mutate*. In particular, each new individual $\nu_j^{i+\frac{1}{2}}$ is generated by mating (of random individuals in G_i) with probability CXPB and by mutation (of a random individual in G_i) with probability MUTPB;
   (b) Evaluate the objectives functions of $G_{i+\frac{1}{2}}$ using *evaluate*;
   (c) generate G_{i+1} by selecting the best μ individuals in $G_{i+\frac{1}{2}} \bigcup G_i$ using *select*;
3. return as output of the algorithm the final generation $G_{n_{max}}$.

In general, we have no guarantee that the algorithm just described will reach the optimum we are looking for. In fact, it is not even assured that different run of the GA give the same final result, as one would expect in theory, because of the random nature of GA contained in the generation of the initial population and in the mating and mutation functions. This could in fact be considered as a way of certifying the reliability of the optimization process: if, starting from different initial populations and considering different mating and mutations, we arrive, more or less, to the same optimum, we can say with a certain confidence that this is indeed the optimum we were searching for.

Another aspect must be considered to assert the optimality of our solution: the error introduced by the ROM. It is possible in fact that an optimum detected by the ROM is not an optimum when considering the FOM. It is also possible in fact that an optimum detected by the FOM is not an optimum when considering the continuum formulation or when considering the real physical phenomenon. We will not consider however these last two steps since, as we already specified in the initial remarks of Chapter 3, the aim of our work is not a study of modeling or discretization techniques, and we will only confront our results using the FOM.

4.4.2 Description of the Optimization Setting

We now list the explicit form of the functions and parameters that specify the optimization run. Apart from *evaluate*, that is the objective function of the optimization (or some modification of it, as we will see in our case) and is thus fixed, the choice of the other functions is entirely arbitrary.

The functions are defined as:

- *evaluate*: returns the value of the total resistance over the considered hull plus a penalization for volume loss. The penalization is imposed in a sharp manner, in which if a hull loses volume over the 1‰, a very big penalization is imposed (of numerous order of magnitude bigger than the resistance value), removing the hull from the possibility of being chosen in the select phase;

- *mate*: one point crossover, in which taken two individuals $[\nu_1^A, \nu_2^A, ..., \nu_N^A]$ and $[\nu_1^B, \nu_2^B, ..., \nu_N^B]$ and a random value i between 1 and N, the new individual is formed by the first i parameters of A and the last $N-i$ parameters of B, i.e. for example for $i = 3$ we have $[\nu_1^A, \nu_2^A, \nu_3^A, \nu_4^B, ..., \nu_N^B]$;

- *mutate* gaussian mutations, in which each parameter of the considered hull is mutated by a Gaussian of standard deviation $\sigma = 0.1\times$(upper bound of the considered parameter) with independent probability;

- *select*: select the best μ individuals in terms of the output of evaluate.

As for the other parameters characterizing the genetic algorithm, the values of MUTPB and CXPB we imposed are respectively 0.2 and 0.8, the values of λ and μ are 200 and 30, the value of σ is 0.1, the probability of mutation for each parameter is 0.3, and the total number of generations is $n_{max} = 15$.

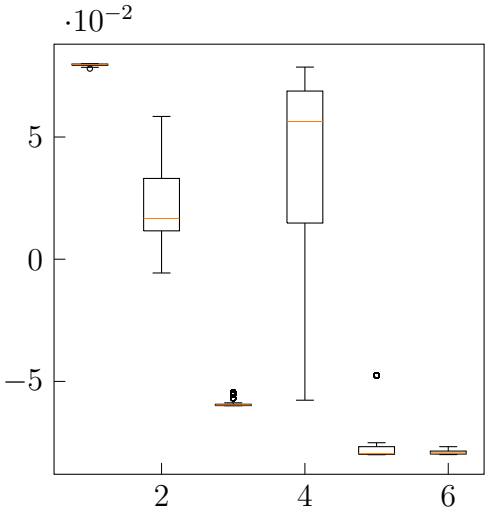
Other choices are possible for the imposition of the volume constraint. For example, we could penalize an illegal hull with a small value, proportional to the volume loss, thus allowing for a less sharp distinction between illegal and legal hulls. Another possibility is multi-objective optimization, in which the *select* function considers optimal hulls in terms of minor resistance and maximum volume, with different weights on the two contributions.

4.4.3 Results and Considerations

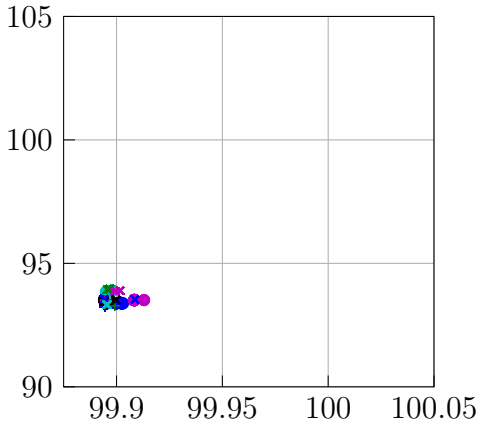
Before discussing the results, we remark that to implement the algorithm presented above, we use DEAP, an open-source Python library for evolutionary optimization (see [32]).

The optimization algorithm for the computation of total resistance employs the optimal ROM setting found in Section 4.3.

Because of the considerations made previously regarding the random nature of GA, we have decided to run the optimization algorithm 15 times.



(a) Boxplot of the value of the 6 coefficients for the individuals belonging to the last generation of all the different optimization runs.



(b) Individuals belonging to the last generation of all the different optimization runs represented as points in a plane with the total resistance in the y-axis and volume in the x-axis, both expressed as percentage over the undeformed hull.

Figure 4.25: First 15 optimization runs.

The final generation of each different run, and in particular the best individual found in this set, is then considered as the output of the optimization process.

We report in Figure 4.25 two graphs: the first one, on the left, representing a boxplot of the values of the 6 parameters, and the second one, on the right, representing hulls as points in a plane where in the y-axis is reported the value of the resistance of the deformed hull and in the x-axis is reported its volume, both calculated as a percentage of the undeformed hull. In both the plots we represent the entire final generation (composed of 30 hulls) of all the 15 runs of the optimization (represented with different symbols/colors in the second graph).

From Figure 4.25a we can see how the algorithm converges for almost all parameters (except the 4^{th} and to a lesser extent the 2^{nd}). This convergence can also be seen in 4.25b where we can also see that, as expected, the optimized hulls tend to lose volume to the maximum allowed by the constraint (that is 1‰) and tend to reduce the resistance of a considerable amount (around 6%).

At this point we proceed to test some of the optimized hulls obtained in the previous phase against the FOM. In particular, we consider the best hull

run #	ROM	FOM
1	272.076	291.328
2	272.010	281.610
3	273.394	290.195
4	272.278	288.863
5	272.460	288.581
6	271.746	n.a.
7	271.652	283.204
8	273.086	289.926

run #	ROM	FOM
9	272.364	284.880
10	271.904	290.954
11	273.664	287.503
12	272.468	287.412
13	271.868	292.869
14	273.418	288.013
15	272.424	285.435

Table 4.2: Value of the total resistance computed (via the FOM or via the ROM) for the optimal hull of each of the 15 optimization runs (non-available data are related to simulations that did not converge).

(in terms of total resistance) of each run of the optimization, obtaining a total of 15 new hulls to be tested. The values of the resistance obtained, for the ROM and for the FOM, for these 15 hulls are reported in Table 4.2.

For a comparison of these values, we mention that the total resistance computed over the undeformed hull (via the FOM) is equal to 291.308.

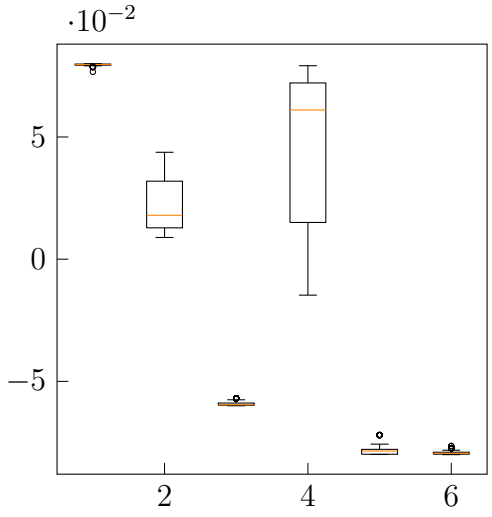
We can see that, even though the ROM tends to underestimate the value of the resistance considerably, the optimal hulls are, in most cases, an improvement of the undeformed hull.

The next step is now to add the new solutions computed to the snapshots matrix and to build a new ROM, performing the same steps as above. This addition has the effect of improving the information the ROM has available in the zone where we have localized the minimum in the previous step. This will in turn enhance the ability of the ROM to predict the value of resistance for new parameters in this zone of the parameter space.

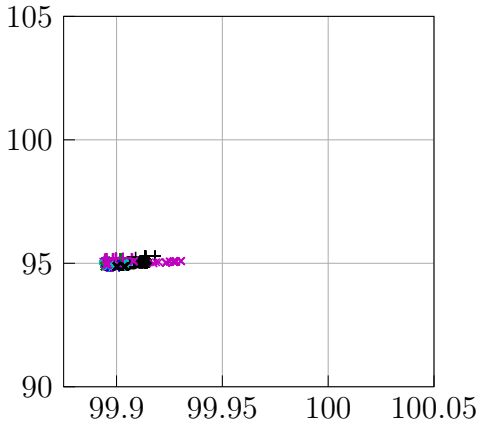
The ROM obtained in this way is then used to run 15 new optimizations as before, obtaining Figure 4.26.

From these figures, we can see that the optimization returns more or less the same optimum values. A first difference with the previous optimization run can be noted in the fact that the resistances returned by the ROM are in this case slightly higher. This is connected to the fact that the reduced order model is improved by the addition of the new snapshots, as we had expected. Another difference can be noted in Figure 4.26a where the dispersion of the 2^{nd} and 4^{th} variables has decreased compared to the previous run.

This last observation, and the fact that the optimum returned by both the optimization runs are essentially the same, gives us the confidence to take this as the result of our optimization pipeline. The final step consists



(a) Boxplot of the value of the 6 coefficients for the individuals belonging to the last generation of all the different optimization runs.



(b) Individuals belonging to the last generation of all the different optimization runs represented as points in a plane with the total resistance in the y-axis and volume in the x-axis, both expressed as percentage over the undeformed hull.

Figure 4.26: Second 15 optimization runs (with the addition of the previous 15 optima).

run #	ROM	FOM
1	276.870	283.978
2	276.600	283.754
3	276.848	288.963
4	276.472	n.a.
5	276.574	286.259
6	277.254	282.343
7	276.382	286.859
8	276.434	292.972

run #	ROM	FOM
9	277.294	286.304
10	277.474	284.329
11	276.388	285.248
12	276.372	284.564
13	276.462	289.081
14	276.552	286.810
15	276.342	282.436

Table 4.3: Value of the total resistance computed (via the FOM or via the ROM) for the optimal hull of each of the 15 optimization runs (with the addition of the previous 15 optima) (non-available data are related to simulations that did not converge).

then to test the new 15 optimal hulls using the full order model to certify the results, obtaining Table 4.3.

As a conclusion of this section, we report the output of our pipeline, which is the best hull, evaluated in terms of FOM total resistance, between all the ones returned by the optimization algorithm. This optimal configuration corresponds to the run # 2 of Table 4.2, with values of the parameters equal to: $[0.7957, 0.1242 - 0.5874, 0.7220, -0.7943 - 0.7946]$. The improvement on total resistance for this value of the parameters, measured on the FOM, is around 3.3%.

To visualize the result, we report in Figure 4.28 the plot of areas and in Figure 4.27 the x, y and z sections corresponding to this value of the parameters.

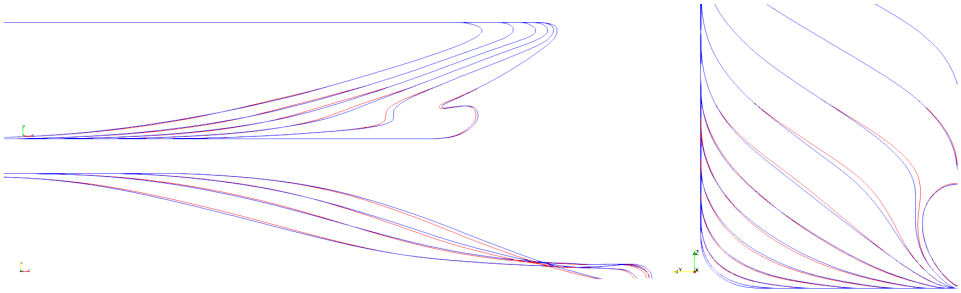


Figure 4.27: View of the x (right), y (upper left) and z (lower left) sections for the optimal hull shape; in blue the undeformed hull, in red the deformed hull.

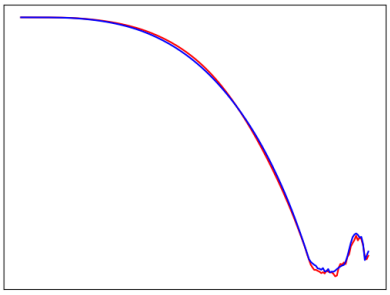


Figure 4.28: View of the plot of areas for the optimal hull shape; in blue the undeformed hull and in red the deformed hull.

Chapter 5

Conclusion and Perspectives

In this Master Thesis we have designed a data-driven modular shape optimization pipeline that can obtain a significant improvement on the efficiency of the hull of a cruise ship in a time that is reasonable also in an industrial context.

Going into more details, we have seen that the use of free form deformation to obtain deformations of an initial unoptimized hull is a good option, because of its flexibility of use and the positive results obtained. In defining the setting of FFD, one must assure the compliance with the constraint and aims of the deformation and, most importantly, establish the complexity and extension of the design space generated. In fact, we have seen that a reduction of the complexity of the problem using POD at this stage is not possible. As for the perspectives, one could consider more general deformation settings, enlarging the design space and thus possibly obtaining a better optimum. However, this possible improvement would lead to an increasing of the complexity of the problem and, hence, of the computational time associated.

As for the FOM model, we could consider possible enhancements both from the point of view of the discretization, considering a more refined grid or more precise schemes, and from the point of view of the model, removing one or some of the hypothesis exposed in Section 3.1.

From the point of view of reduced order models, we can conclude that they are of great use both in the context of reduction of the dynamical system associated to the FOM (via DMD), and, especially, in the reduction of the parametric problem (via PODI). The error associated with the latter in fact, even if not negligible (around 5%), enables us to run an optimization algorithm that gives us a reliable optimum, where the reliability has been tested on the FOM. A significant improvement that should be tested is the implementation of intrusive POD. In fact, despite losing some flexibility on

the generation of the snapshots matrix, a crucial aspect from an industrial point of view, we expect that the error associated with an intrusive approach could be noticeably minor with little increase of the computational time.

As for the optimization algorithm, genetic algorithms seem to behave well in this case study and are able to return solutions that are, with a certain confidence, quite near to the real optimum. Different alternatives could be tested, like for example a coupled approach "GA + gradient descent" in order to obtain a more precise determination of the optimum solution.

We want to emphasize two interesting features of the pipeline we designed, especially from the industrial point of view, that are its modularity and its data-driven nature. In fact, one or more ingredients between the ones we described (geometrical deformation, FOM, optimization phase) could be replaced in order to meet the requirements needed for the particular application at hand. Moreover, the flexibility given by the data-driven approach allows us to consider various possibilities for the generation of the FOM snapshots, also considering commercial CFD software or experimental data.

The presented pipeline can be further extended in an easy manner, thanks to the already mentioned modularity, in order to use more advanced and sophisticated algorithms to approximate the solution manifold. Machine learning techniques, that we partially explored in this thesis by using the gaussian process regression, can be an option to maintain the equation-free nature of our approach and at same time increase the accuracy of the approximated output of interest. Moreover, thanks to the huge computational reduction, we can adopt this pipeline to create a digital twin of complex system , allowing the generation of virtual model that replicate in real-time the behaviour of the original system.

Finally, to quantify the results we have obtained, we recall that the improvement over the initial hull we have obtained is around 3.3% on the total resistance, with a total loss of submerged volume of around the 1‰, as requested. The computational time required by our pipeline is around five weeks of CPU time, corresponding to the computation of 100 high fidelity solutions of 8 hours each (the time required by the other phases is negligible). As a comparison, the use of the FOM for the execution of one run of the optimization process (in contrast to the 30 runs we have considered for the ROM) would require around 140 weeks. We also remark that, as we have seen in Figure 4.20b, we could decrease considerably the computational time required by the pipeline by generating fewer snapshots in the offline phase, obtaining an error that is still comparable to the one we achieved.

Bibliography

[1] N. Demo, M. Tezzele, G. Gustin, G. Lavini, and G. Rozza, "Shape optimization by means of proper orthogonal decomposition and dynamic mode decomposition," in *Technology and Science for the Ships of the Future: Proceedings of NAV 2018: 19th International Conference on Ship & Maritime Research*, pp. 212–219, IOS Press, 2018.

[2] N. Demo, M. Tezzele, A. Mola, and G. Rozza, "A complete data-driven framework for the efficient solution of parametric shape design and optimisation in naval engineering problems." 2019.

[3] M. Tezzele, N. Demo, M. Gadalla, A. Mola, and G. Rozza, "Model order reduction by means of active subspaces and dynamic mode decomposition for parametric hull shape design hydrodynamics," in *Technology and Science for the Ships of the Future: Proceedings of NAV 2018: 19th International Conference on Ship & Maritime Research*, (Trieste, Italy), IOS Press, IOS Press, 2018.

[4] P. Davidson, *Turbulence: An Introduction for Scientists and Engineers*. Oxford University Press, 2015.

[5] G. K. Batchelor, *An introduction to fluid dynamics*. Cambridge University Press, 1967.

[6] J. S. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer Briefs in Mathematics, Switzerland: Springer, 1 ed., 2015.

[7] A. Quarteroni and G. Rozza, *Reduced Order Methods for Modeling and Computational Reduction*, vol. 9 of *MS&A*. Springer, 1 ed., 2014.

[8] A. Quarteroni, G. Rozza, and A. Quaini, "Reduced basis methods for optimal control of advection-diffusion problems," tech. rep., RAS and University of Houston, 2007.

[9] M. Strazzullo, F. Ballarin, R. Mosetti, and G. Rozza, "Model reduction for parametrized optimal control problems in environmental marine sciences and engineering," *SIAM Journal on Scientific Computing*, vol. 40, 10 2017.

[10] S. Georgaka, G. Stabile, K. Star, G. Rozza, and M. J. Bluck, "A hybrid reduced order method for modelling turbulent heat transfer problems," *ArXiv*, vol. abs/1906.08725, 2019.

[11] F. Ballarin, G. Rozza, and Y. Maday, ch. Reduced-order semi-implicit schemes for fluid-structure interaction problems, pp. 149–167. Springer International Publishing, 2017.

[12] F. Ballarin, A. Manzoni, G. Rozza, and S. Salsa, "Shape optimization by free-form deformation: Existence results and numerical solution for Stokes flows," *Journal of Scientific Computing*, vol. 60, no. 3, pp. 537–563, 2014.

[13] G. Stabile and G. Rozza, "Finite volume pod-galerkin stabilised reduced order methods for the parametrised incompressible navier–stokes equations," *Computers & Fluids*, vol. 173, pp. 273–284, 2018.

[14] S. Volkwein, "Proper orthogonal decomposition: Theory and reduced-order modelling," *Lecture Notes, University of Konstanz*, 01 2012.

[15] E. Tupper, "Introduction to naval architecture," *Introduction to Naval Architecture*, 01 2013.

[16] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *IN: SIGGRAPH, 1986. PROCEEDINGS*, pp. 151–160, 1986.

[17] S. Coquillart, "Extended free-form deformation: A sculpturing tool for 3d geometric modeling," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, (New York, NY, USA), pp. 187–196, ACM, 1990.

[18] A. Koshakji, A. Quarteroni, and G. Rozza, "Free form deformation techniques applied to 3d shape optimization problems," *Communications in Applied and Industrial Mathematics*, 2013.

[19] T. Lassila and G. Rozza, "Parametric free-form shape design with PDE models and reduced basis method," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 23-24, pp. 1583–1592, 2010.

[20] F. Salmoiraghi, A. Scardigli, H. Telib, and G. Rozza, "Free-form deformation, mesh morphing and reduced-order methods: enablers for efficient aerodynamic shape optimisation," *International Journal of Computational Fluid Dynamics*, vol. 32, no. 4-5, pp. 233–247, 2018.

[21] C. Hirt and B. Nichols, "Volume of fluid (vof) method for the dynamics of free boundaries," *J. Comput. Phys.*, vol. 39, pp. 201–225, 1 1981.

[22] F. Moukalled, L. Mangani, and M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab*. Springer Publishing Company, Incorporated, 1st ed., 2015.

[23] J. H. Ferziger and M. Perić, *Computational Methods for Fluid Dynamics*. Berlin: Springer, 2nd ed., 1999.

[24] OpenCFD, *OpenFOAM - The Open Source CFD Toolbox - User's Guide*. OpenCFD Ltd., United Kingdom, 6 ed., 7 2018.

[25] S. M. Dàmian, *Description and utilization of the interFoam multiphase solver*. Available at `http://infofich.unl.edu.ar/upload/3be0e16065026527477b4b948c4caa7523c8ea52.pdf`.

[26] PyGEM, "PyGEM: Python geometrical morphing." Available at `https://github.com/mathLab/PyGeM`.

[27] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. Available at `http://www.scipy.org/`.

[28] GPy, "GPy: A gaussian process framework in python," since 2012. Available at `http://github.com/SheffieldML/GPy`.

[29] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[30] N. Demo, M. Tezzele, and G. Rozza, "EZyRB: Easy Reduced Basis method," *The Journal of Open Source Software*, vol. 3, no. 24, p. 661, 2018.

[31] N. Demo, M. Tezzele, and G. Rozza, "PyDMD: Python Dynamic Mode Decomposition," *The Journal of Open Source Software*, vol. 3, no. 22, p. 530, 2018.

[32] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.