SISSA
Scuola
Internazionale
Superiore di
Studi Avanzati

Mathematics Area - PhD course in
Mathematical Analysis, Modelling, and Applications

Hybrid reduced-order methods for segregated fluid-structure interaction solvers in an ALE approach using the Finite Volume Method

Candidate:
Valentin Eric Brice
Nkana Ngan

Advisor:
Prof. Gianluigi Rozza
Co-advisors:
Prof. Giovanni Stabile
Dr. Andrea Mola

Academic Year 2023-24

A ma famille.

Acknowledgments

First and foremost: Blessed be the GOD, Father of our Lord Jesus-Christ, who has blessed us with all spiritual blessings in heavenly places in Christ Ephesians 1:3. I am grateful to GOD for providing me with the patience, tenacity, and endurance to complete my Ph.D. and advance in my career.

I would also like to express my gratitude to my advisor, Prof. Gianluigi Rozza, for allowing me to be his student during my PhD studies. I want to express my gratitude to him for his amazing tolerance, unwavering assistance, tremendous support, and love. He has always shown that the well-being of his students is his top priority, and he has been a great friend. His words of encouragement and reassurance were immensely appreciated when things weren't going as planned. Being a student in Prof. Rozza's lab allowed me to make significant progress that I am not sure if I would have been able to achieve otherwise.

I am grateful as well to my co-advisors Prof. Giovanni Stabile and Dr. Andrea Mola for their ongoing support. I'd want to thank them for those hours we spent together discussing my work's outcomes and debugging code. Their thoughts and guidance greatly improved this dissertation. Their insights significantly improved the quality of our work. I appreciate Giovanni's tolerance and Andrea's inspirational quest for perfection throughout the past four years.

I'd also want to thank my thesis committee members, Professor Annalisa Quaini and Professor Nicola Parolini, for agreeing to review this work.

I'd like to express my heartfelt gratitude to my family for all of their sacrifices during my stay overseas, as well as for always providing me with moral support and encouragement. Thank you for all the texts and phone calls, even though I was often too stressed to respond. A huge thank you to my wife, Ms Armelle Nkana Ngan, and my daughters Gabriella, Emmanuelle, and Raphaelle, for their unwavering encouragement and moral support, as well as patience and understanding, throughout all of my M.Sc. and Ph.D. studies, despite the fact that I have been away from home for nearly five years. Not every day was easy, but having you all as an unwavering support is why I can keep going and do my best.

I am also grateful to my second family: the Rozza's group for the wonderful conversations we had during the coffee breaks. Additionally, I would like to express my gratitude to my office mates, Marco Feder, Lorenzo Fabris, and Nirav Shah, for our many hours of togetherness. They are like brothers to me, and over the years, I have spent more time with them than with my own family.

Last but not least, I would like to thank myself for doing this hard-work and to have completed this additional step in this life. I can't tell you how many times I thought about quitting this Ph.D. program without saying it aloud.

Abstract

This dissertation intends to construct hybrid reduced-order models (ROMs) for segregated Fluid-structure interactions solvers in a Arbitrary Lagrangian-Eulerian (ALE) framework both in laminar and turbulent regime. The ROM methodology developed is based on full order solvers which make use of the Finite Volume Discretization method. The hybrid ROMs are built by combining data-driven techniques and classic model reduction approaches, such as proper orthogonal decomposition combined with Galerkin projection. This combination seems like a natural evolutionary process for engineering applications. The resulting hybrid ROMs, allow for both scalability and physics-based modelling. In this hybrid model, Proper Orthogonal Decomposition (POD) primarily calculates low-dimensional projections/features that can be evolved over time through the simultaneous use of classical Galerkin projection, neural networks/ recurrent neural networks, or radial basis functions. In this recipe, the Partial Differential Equations (PDEs) associated to mass and momentum balance are treated using a standard POD-Galerkin projection approach, while Radial Basis Functions (RBF) networks are used for the grid motion interpolation and Neural Networks (NNs) are employed for the approximation of the eddy viscosity field. Incorporating data-driven techniques, the reduced-order models are able to achieve higher accuracy, efficiency and flexibility of use, making them powerful tools for simulating complex Fluid-structure interactions problems which could be useful in industrial applications and in the development of digital twins.

In this work, we tested the hybrid ROMs on two different test cases. The first test is that of the flow around an oscillating cylinder in laminar regime ($Re = 200$) and the second test case on the Flow Induced Vibration (FIV) of a pitch-plunge airfoil at a high Reynolds number ($Re = 10^7$). The results confirm that in each test case the ROM methodology developed is able to reproduce the wake dynamics of the flow around the moving body and all the response characteristics of the system such as the lift and drag forces, amplitude/ frequency/phase of the displacement, and the features of the original dynamical system. The results show good convergence properties without any necessity of additional stabilization for what concerns pressure solutions. This is due to the fact that, even at the reduced order level, we use a segregated scheme which reproduces the full order model algorithm used to deal with the saddle-point structure of the Navier–Stokes equations.

Finally, to circumvent the expensive computational cost associated with the Galerkin projection methods, we introduce an additional hyper-reduction methodology based on Empiral Lagrangian Interpolation Method (ELIM) leveraging on the identification of some points at optimal locations in the domain of interest. Preliminary results on the reconstruction part of the algorithm are presented for the test case of Burger's equation over a backward facing step.

Keywords: Fluid-structure interaction, reduced-order model, Finite Volume Method, Proper orthogonal decomposition, Galerkin projection, radial basis network, mesh

motion, turbulence, long-short term memory, neural networks, flow-induced vibration, vortex-induced vibration, Empirical Lagrangian interpolation, time series, and machine learning.

Acronyms

AEs	Auto-Encoders.
ALE	Arbitrary Lagrangian-Eulerian.
CFD	Computational Fluid Dynamics.
CNNs	Convolutional Neural Networks.
DMD	Dynamic Mode Decomposition.
EIM	Empirical Interpolation Method.
ELI	Empirical Lagrangian Interpolation.
ELIM	Empiral Lagrangian Interpolation Method.
FDM	Finite Difference Method.
FEA	Finite Element Analysis.
FEM	Finite Element Method.
FIV	Flow Induced Vibration.
FOM	Full-order Models.
FSI	Fluid-structure interactions.
FVM	Finite Volume Method.
GCL	Geometrical Conservation Law.
GNNs	Generative Neural Networks.
HFM	High-Fidelity Models.
LSTM	Long Short-Term Memory.
ML	Machine Learning.
NNs	Neural Networks.
NSEs	Navier-Stokes Equations.
ODEs	Ordinary Differential Equations.
PCA	Principal Component Analysis.
PDEs	Partial Differential Equations.
PIMPLE	Pressure-Implicit with Splitting of Operators.
PINNs	Physics Informed Neural Networks.

PISO	Pressure Implicit with Splitting of Operators.
POD	Proper Orthogonal Decomposition.
RANS	Reynolds Averaged Navier-Stokes.
RBF	Radial Basis Functions.
RNNs	Recurrent Neural Networks.
ROM	reduced-order model.
ROMs	reduced-order models.
SIMPLE	Semi-Implicit Method for Pressure Linked Equations.
SVD	Singular Value Decomposition.
VIV	Vortex Induced Vibrations.

List Of Scientific Contributions

Preprints

- **Valentin Nkana Ngan**, Giovanni Stabile, Andrea Mola, and Gianluigi Rozza. "A hybrid reduced-order model for segregated fluid-structure interaction solvers in an ALE approach at high Reynolds number." arXiv preprint arXiv:2406.12701 (2024). **Submitted**.
- Zancanaro, Matteo, **Valentin Nkana Ngan**, Giovanni Stabile, and Gianluigi Rozza. "A segregated reduced order model of a pressure-based solver for turbulent compressible flows." arXiv preprint arxiv.org/abs/2205.09396 (2024). **In revision**.
- **Valentin Nkana Ngan**, Giovanni Stabile, Andrea Mola, and Gianluigi Rozza. "A reduced-order model for segregated fluid-structure interaction solvers based on an ALE approach." arXiv preprint arXiv:2305.13613 (2023). **In revision**.

Journal papers

Contents

Acknowledgments v

Abstract vii

List Of Scientific Contributions xi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Literature review . . . 2
  - 1.2.1 Projection-based ROMs . . . 2
  - 1.2.2 Related work . . . 3
  - 1.2.3 Machine learning for CFD . . . 5
- 1.3 Thesis Contributions . . . 7
- 1.4 Thesis Structure . . . 8

2 Full-order mathematical model 11
- 2.1 Introduction . . . 11
- 2.2 Fluid's Governing Equations . . . 12
  - 2.2.1 The incompressible Navier-Stokes Equations . . . 12
  - 2.2.2 Reynolds Average Navier-Stokes . . . 13
    - 2.2.2.1 Boussinesq assumption . . . 13
    - 2.2.2.2 Modelling turbulent viscosity . . . 14
- 2.3 Finite Volume Method . . . 15
  - 2.3.1 The pressure gradient term . . . 16
  - 2.3.2 Geometric Conservation Law . . . 16
  - 2.3.3 The Convective Term . . . 17
  - 2.3.4 The Diffusion Term . . . 17
- 2.4 Segregated Pressure-Based Solvers . . . 18
  - 2.4.1 Pressure-equation for incompressible flows . . . 18
    - 2.4.1.1 The SIMPLE algorithm . . . 18
    - 2.4.1.2 The PISO algorithm . . . 20
- 2.5 Structure's governing equations . . . 22
  - 2.5.1 Translation motion . . . 22
  - 2.5.2 Translation and rotational motion . . . 22
- 2.6 Coupling conditions . . . 22
- 2.7 Mesh motion techniques . . . 23
- 2.8 Summary . . . 24

3 Reduced-order models **25**
3.1 Introduction . . . 25
3.2 The Proper Orthogonal Decomposition . . . 26
3.3 The segregated reduced approaches . . . 29
3.3.1 Galerkin projection . . . 29
3.3.2 Reduced-PIMPLE for incompressible flows . . . 30
3.3.3 Hyper-Reduced-PIMPLE for incompressible flows . . . 30
3.3.4 POD with interpolation for mesh motion prediction . . . 32
3.4 Machine learning for eddy viscosity prediction . . . 35
3.4.1 POD-NNs . . . 36
3.4.2 POD-RNNs . . . 37
3.5 Efficient projection in moving domains problems . . . 39
3.6 Hyper-Reduction: From projection to interpolation . . . 40
3.6.1 From projection to interpolation . . . 40
3.6.2 Empirical Lagrangian Interpolation algorithm . . . 42
3.6.3 Error analysis . . . 43
3.7 Summary . . . 44

4 Applications and Numerical Results **45**
4.1 Physical problems of interest . . . 45
4.2 Unsteady Laminar Case . . . 46
4.2.1 Description of the configuration and boundary conditions . . . 46
4.2.2 Description of the configuration and boundary conditions . . . 46
4.2.2.1 Linear solvers for the fluid . . . 47
4.2.2.2 Structural solver . . . 48
4.2.3 Results and discussion . . . 48
4.2.3.1 Computational cost . . . 48
4.2.3.2 Reconstruction error . . . 49
4.2.3.3 ROM solution error . . . 50
4.3 Unsteady Turbulent Case . . . 56
4.3.1 Definition of test case, simulation results, and discussion . . . 56
4.3.1.1 Definition of the test case . . . 56
4.3.1.2 Simulation results . . . 58
4.3.1.3 Prediction quality . . . 58
4.3.1.4 Machine learning of the temporal eddy viscosity coefficients . . . 59
4.3.1.5 ROM online resolution time . . . 60
4.3.1.6 ROM online resolution quality . . . 61
4.4 Empirical Lagrangian Interpolation . . . 69
4.5 Summary . . . 70

5 Conclusions and Outlooks **71**
5.1 Conclusions . . . 71
5.2 Outlooks and Perspectives . . . 72

6 Appendices **75**
6.1 Machine learning of the temporal eddy viscosity coefficients . . . 75
6.2 Cylinder additional results . . . 75
6.2.1 Synchronization analysis . . . 75

6.2.2 Temporal coefficients analysis 76
6.2.3 Domain Filtering 76

List of Figures

2.1 Relation between two neighbour cells Ω_i and Ω_j of the tessellation $\mathcal{T}$ for a given variable $\bar{\boldsymbol{u}}$. 16
2.2 Flow chart of fluid-body motion 23

3.1 Machine Learning Algorithms [2]. 26
3.2 Schematic of a multi-layer neural network 37
3.3 Schematic of recurrent neural network 38

4.1 Cross flow vortex-induced vibrations 46
4.2 The mesh used in the simulations: (a) the initial mesh, (b) the deformed mesh in correspondence with a cylinder displacement of approximately 40% of the diameter. 47
4.3 From left to right, the eigenvalues decay and cumulative eigenvalues of the POD modes. Blue lines indicate velocity eigenvalues, green lines indicate pressure eigenvalues, and red lines indicate point displacement eigenvalues 50
4.4 FOM and ROM solutions comparison at $t = 20$ s from left to right: column (a) FOM solutions and column (b) predicted solutions. The first row represents the velocity fields, second-row pressure fields, and third-row grid nodes displacement fields of both FOM and ROM. The reduced solution used POD 20 modes for both velocity and pressure, and 1 mode for grid nodes displacement. 51
4.5 Velocity field Absolute ROM error in the L^2 norm as a function of time. 51
4.6 Pressure field Absolute ROM error in the L^2 norm as a function of time. 52
4.7 Time series comparison between the reference curve of the lift force acting on the cylinder in Newton unit with predicted curves 53
4.8 Time evolution of the absolute errors of the pressure reduced approximation. The error values in both graphs are in percentages. 53
4.9 Time series comparison between the reference curve of the drag force acting on the cylinder in Newton unit with predicted curves. 54
4.10 Time series of the absolute error analysis of the drag force (original and predicted signals) from Figure 4.9 54
4.11 Time series evolution of the centre of mass. 55
4.12 Time series evolution of the centre of mass absolute error. 55

4.13 (a) Schematic of the fluid-structure system considered: a foil allowed to undergo 2 degrees of freedom fully passive plunging and pitching motion with spring constraints [171], (b) a picture of the zoomed mesh with 12 556 cells (control volumes) and 26 316 node points near an airfoil of chord length 1.0 m, (c) picture showing the position of the foil in the computational domain 56
4.14 The decay of the POD modes eigenvalues for velocity, pressure, point-Displacement, and Eddy viscosity fields. Color code: blue – velocity, green –pressure, red – pointDisplacement, magenta – eddy viscosity . 59
4.15 Comparison of the velocity and pressure fields. First row velocities comparison and second row pressure comparison. Column (a) FOM fields, column (b) reduced solution with POD-NNs and column (c) reduced solution with POD-LSTM. The snapshots are captured in the second period i.e. $t = T = 0.1$ s 62
4.16 Comparison of the eddy viscosity and grid node displacement fields. First row eddy viscosity comparison and second row grid node displacement comparison. Column (a) FOM fields, column (b) reduced solution with POD-NNs and column (c) reduced solution with POD-LSTM. The snapshots are captured in the second period i.e. $t = T = 0.1$ s. 62
4.17 Sensitivity study of the error (log-scale) in the L^2-norm versus the time evolution of the velocity field. The red line shows the results obtained inside and outside the time window 63
4.18 Sensitivity study of the error (log-scale) in the L^2-norm versus the time evolution of the pressure field. The red line shows the results obtained inside and outside the time window 63
4.19 Sensitivity study of the error in the L^2 norm in log-scale versus the time evolution of the eddy viscosity field. The red line shows the results obtained inside and outside the time window 63
4.20 Time series comparison between the reference signal of the lift force acting on the foil in Newton unit with predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation). 65
4.21 Time series of the error analysis of the lift force (original and predicted signals) from Figure 4.20. The vertical line in red divides the error plots in two. Left: interpolation error and right: extrapolation error. . 65
4.22 Time series comparison between the reference signal of the drag force acting on the foil in Newton unit with predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation). 66
4.23 Time series of the error analysis of the drag force (original and predicted signals) from Figure 4.22. The vertical line in red divides the error plots in two. Left: interpolation error and right: extrapolation error. 66
4.24 Time series comparison between the reference signal of the plunge with different predicted signals. The vertical line divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation). 67

4.25 Plunge's error analysis versus time. 67
4.26 Time series comparison between the reference signal of the pitch (angle of attack) with different predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation). 68
4.27 Pitch's error analysis versus time. 68
4.28 The 50 Lagrangian points are represented in red, the stencils of the cells and associated degrees of freedom needed for the evaluation of the discrete differential operators are in light-green. The discarded nodes in the evolution of the dynamics are in blue. The stencil is made by 1 layer of cells. 69
4.29 Comparison of the velocity field. First row original solutions, second row reconstructed solutions using 50 Lagrangian points, and third row the reconstructed errors. Snapshots are given at $t = 0.1$ s, $t = 0.22$ s, and $t = 3$ s. 70

6.1 Training, validation, and test of the first four POD temporal coefficients of the eddy viscosity at the offline level using LSTM and NNs. The first half of the datasets (scaled between -1 and 1) is divided in two parts (30% for training and 20% for validation) for training and validation on both models. The remaining half is used for testing on both architectures. 76
6.2 First row, from left to right: the time histories of the lift and drag forces. The solid black lines are the FOM curves, and the dashed green lines are the ROM curves obtained with 16 modes for the velocity and 21 modes for the pressure. Second row, from left to right: Power spectra density comparison of the lift and drag coefficients. 77
6.3 First six temporal coefficients of the velocity: black original signals and red predicted signals. 77
6.4 First six temporal coefficients of the pressure: black original signals and blue predicted signals 78
6.5 Domain filtering: first row initial mesh, second row: (a) region of highest interest (near mesh), (b) region of lower interest (intermediate and (c) region not affected by the mesh motion (far field). 78

List of Tables

4.1 A summary of the boundary conditions imposed in the ALE fluid dynamic problem. Note that the * subscript indicates quantities that are computed by the rigid body structural solver. 47
4.2 Simulation parameters 48
4.3 Offline and Online times comparison varying the number of modes . . 49
4.4 Summary of simulation settings of the flow passing pitch-plunge airfoil 58
4.5 Normalized eigenvalues of the POD modes of the fields of interest . . 59
4.6 Offline and Online times comparison varying the number of modes . . 61

6.1 LSTM hyper-parameters 75

List of Algorithms

1 SIMPLE algorithm for compressible flows 20
2 PIMPLE algorithm with dynamic mesh. 21

3 POD algorithm 28
4 Reduced-PIMPLE algorithm with dynamic mesh 31
5 Hyper-Reduced-PIMPLE algorithm with dynamic mesh 33
6 Compute MLP output. 37
7 Compute the output sequence of an RNN 38
8 General scheme of the LSTM algorithm 39
9 Empirical Lagrangian Interpolation Method 42

Chapter 1

Introduction

This chapter is intended to provide some motivation on the development of ROMs for FSI problems and the need to develop reduced-order models (ROMs). Section 1.1 discusses the motivation behind the importance of the study of FSI problems. Next, in Section 1.2 some literature review is presented. First, SubSection 1.2.1 discusses projection-based ROMs, and second, some related works in SubSection 1.2.2 are mentioned. After, that, in SubSection 1.2.3, the literature on Machine Learning (ML) for CFD is presented. In the last two sections, namely Section 1.3 and Section 1.4, we talk about the thesis's contributions and structure.

1.1 Motivation

Fluid–structure interaction (FSI) problems arise in the design of many important systems and artifacts in mechanical, aerospace and civil engineering. More specifically, FSI is particularly important in applications such as automotive, aircraft and spacecraft design, engines performance prediction, bridges and building construction and even the study of blood flow in the human body. As such, the understanding of FSI problems is of paramount importance.

Clearly, FSI couples different phenomena that are related both to fluid dynamics and solid mechanics. These phenomena can then combine in different possible ways and result in extremely rich physical problems. In automotive and civil engineering, the flow past bluff body geometries (cylinders, prisms, cubes, etc.) is the one most often encountered. In such cases, periodic forces generated by vortex shedding can cause structural vibrations with a phenomenon called Vortex Induced Vibrations (VIV). A classical VIV example is represented by vibrations induced on a cylindrical structure exposed to wind. In aerospace engineering applications instead, the flow is typically attached to lift-generating bodies (such as wings, propeller and turbine blades, sails), and vibrations can be induced by different mechanisms such as flutter.

FSI problems in most cases are too complex to be solved analytically. Then, to properly model these problems, most researchers have resorted to the experimental method, and in particular made use of wind tunnels. However, there are several problems that can limit experiments feasibility and effectiveness. On one hand, wind tunnels are small, resulting in scalability problems for the experimental results. In the case of big structures such as bridges, this can lead to inaccurate results. In addition, in the experiments, it is quite difficult to get simultaneous velocity and pressure measurements. On the other hand, wind tunnel testing is associated

to high and sometimes prohibitive costs. Among these, the most relevant costs are associated to the production of accurately crafted testing models, and to the energy required to the operation of the wind tunnel. Thanks to the advancements of computer hardware and numerical algorithms, over the last decades it became more and more convenient to carry out numerical simulations to complement experiments towards a successful design of buildings, cars, aircraft.

At the numerical level, FSI problems are very difficult problems to solve, as they require coupling between different algorithms designed to solve problems of fluid dynamics and structural mechanics. Interfacing Finite Element Analysis (FEA) algorithms for structural mechanics to Computational Fluid Dynamics (CFD) solver can result in fact in quite complex numerical FSI problems. There are, however, several software tools or libraries that can be used to successfully solve FSI problems. We mention some popular solvers such as ANSYS Fluent and ANSYS Mechanical, OpenFOAM [76], STAR-CCM+, COMSOL Multiphysics [71], and ADINA to name a few. In the present context, we will refer to simulations carried out with these instruments as Full-order Models (FOM) or High-Fidelity Models (HFM). Although the results are accurate, the simulations are — even when carried out on advanced platforms — computationally demanding, not only in terms of computational time, but also in terms of data storage and handling. The complexity is further increased when dealing with FSI problems in turbulent regime, which require fine temporal and spatial resolution to capture the intricate dynamics accurately.

Given such a difficulty, there is a pressing need for building reduced-order models (ROMs) that can approximate the essential features of Fluid-structure interactions (FSI) problems with significantly less computational effort [58]. A reduced-order model is in fact a numerical model of the physical system of interest which is derived from the FOM system, but is characterized by significantly fewer degrees of freedom. As such, it can result in relatively inexpensive simulations. In most cases, reduced-order models (ROMs) leverage on data obtained by multiple resolution of the FOM systems to obtain the desired computational cost reduction. Reduced-order models have then been developed and used to obtain more efficient and computationally economical ways of investigating complex problems. The use of ROMs is primarily motivated by the desire to have detailed knowledge of the physics of the problem being investigated, together with an efficient and reliable prediction tool [105]. In the following section, we review some literature of projection-based ROMs, related work, and machine learning in Computational Fluid Dynamics (CFD).

1.2 Literature review

In this part of the dissertation, three subsections will be presented. SubSection 1.2.1 will introduce the concept of projection-based ROMs for CFD. In SubSection 1.2.2, will give a short literature review on ROMs developed for FSI problems in the direction of this work. Finally, SubSection 1.2.3 will focus on the literature on machine learning for CFD is given.

1.2.1 Projection-based ROMs

Various methods of building reduced-order models are available in the literature. However, the common underlying theme is to extract the key features in the flow

field, preferably from a high-fidelity experimental or computational data source. The extracted features are carefully chosen to represent dominant spatial and temporal dynamics as computed using the Navier-Stokes Equations (NSEs).

Projection-based reduced-order models provide an efficient and accurate way to simulate high-dimensional systems by leveraging low-dimensional representations. Techniques like Proper Orthogonal Decomposition (POD) enable the conversion of complex Partial Differential Equations (PDEs) into simpler Ordinary Differential Equations (ODEs), allowing for a reduction of the number of system degrees of freedom and consequently facilitating faster computations while maintaining precision. Extensive research and applications across various domains underscore the importance and utility of projection-based ROMs [27]. As for the projection techniques which can be employed, Galerkin projection [4, 8, 10, 13] and Petrov-Galerkin projection are considered the most used ones because they have succeeded in various research areas [7] and has been used in numerous CFD applications. Although the projection-based approaches are proven to be computationally efficient, they do not account for spatial variations in the flow and are known to become unstable under different conditions even for canonical cases. They are also intrusive in nature, which requires extensive modifications to the current state-of-the-art of CFD codes, so as to obtain a suitable implementation. For such reasons, several researchers have also explored non-intrusive approaches based on data-driven techniques. Among others, Eigensystem realization algorithm (ERA) is one such method which is purely data-driven and is used primarily for the stability analysis of dynamical systems [137,181].

Projection-based methodologies for reduced-order modelling involve projecting the high-dimensional system's governing equations onto a lower-dimensional subspace spanned by the reduced basis obtained from Proper Orthogonal Decomposition (POD) or other techniques. POD is one of the most widely used techniques to find lower dimensional linear subspace for reduced-order modelling of fluid flows [15, 59, 85, 97, 98]. The reason for this, is that the linear low order representations obtained by POD are mathematically optimal for any given dataset. However, within the ROM community, other reduction methodologies are also available in the literature such as the greedy algorithm [128, 134, 168], Proper Generalized Decomposition (PGD) [36, 117], Dynamic Mode Decomposition (DMD) [83, 133, 146]. For more details in the comparisons of these methodologies, the interested reader can refer to [14, 65]. In this dissertation, the POD technique is chosen for the generation of the reduced-order space and a Galerkin projection is applied for the construction of the projection-based ROM.

1.2.2 Related work

In the literature, several scholars have widely studied the application of ROMs for Fluid-structure interactions (FSI), and state-of-the-art counts already a lot of scientific contributions. An example can be found in [152] in which the full-order model of an F16 fighter-aircraft, with over 2.1 million degrees of freedom, was reduced to a model of just 90 degrees of freedom. In [116], the authors presented an overview of the combination of the reduced basis method (RBM) with two different approaches for FSI problems, namely, a monolithic and a partitioned approach. They have provided a detailed implementation of two reduction procedures and then applied them to the Turek-Hron benchmark test case within a fluid at Reynolds number $Re = 100$.

Also, researchers in [180] have made some improvements to extend the application of POD-Galerkin projection to a domain with moving solid boundaries. Researchers in [150] proposed a decoupled modelling of the FSI problems; they modelled the fluid and structure domains separately. They constructed a POD-ROM global basis function for the structure using the Singular Value Decomposition (SVD). A Galerkin-free ROM approach based on POD has been applied to FSI problems with large mesh deformations in [151]. The authors used a two-dimensional VIV on a cylinder and a three-dimensional shock wave boundary layer-induced panel flutter to demonstrate their methodology. However, due to numerical issues associated with Galerkin ROM and the difficulty of constructing a ROM for FSI problem with the moving mesh, the authors used two separate ROM for the fluid and structure domain. The study conducted in [178] discussed a non-intrusive reduced-order model (NIROM) for FSI. The authors based their methodology on POD and Radial Basis Functions (RBF) interpolation methods for unstructured meshes in the Finite Element Method (FEM). They validated their methodology on a one-way (flow past a cylinder) and two-way coupling (a free-falling cylinder in water), and Vortex Induced Vibrations (VIV) of an elastic beam. Moreover, the work of Liberge and Hamdouni [92], has been extended from their previous work on the 1-D Burgers equation [91], they defined global POD modes from a global fluid-solid velocity field, and successfully built a ROM for a flow passing a spring-attached cylinder oscillating at a small amplitude. In [55,93], the authors chose to apply the immersed interface method with POD on a flow passing an oscillatory cylinder. They simulated the interaction between a fluid and a rigid body with imposed rotation velocity. In the case of a rigid body motion as considered in this thesis, an interesting manner to avoid issues associated to mesh deformation was presented by Lewin et al. [90] and Placzek [131]. They performed the projection of the governing equations in a non-inertial reference frame to preserve the POD formulation's consistency. However, in their case, stability problems appear when considering highly non-linear flows. Troshin et al. [165] outlined an alternative POD methodology for a flow field in a domain with moving boundaries. The moving domain is mapped to a stationary domain by combining a transfinite interpolation and an algorithm for volume adjustment. Liberge et al. [93] implemented a multiphase method that allows the performance of the POD on a moving domain using characteristic functions to follow the fluid-structure interface. Falaize et al. [55] extended such formulation for flows induced by rigid bodies in forced rotation. Also, they included parametric changes in the proposed model. Longatte et al. [96] explored the behaviour of POD-multiphase ROM presented in [93] when the parameter values are different from those used to build the POD basis. Stankiewicz et al. [159, 160] deepen the study of Anttonen et al. [7] with test cases of increasing complexity also considering parametric changes. Moreover, in numerical examples involving a moving airfoil, Freno et al. [56] showed that when an index-based domain is used to build the ROM, similar to the one considered by Anttonen et al. [7], numerical simulations do not suffer from the mesh deformation limitation discussed above. Also, authors in [89] have presented a ROMs for simulating the flow around a heaving wing using POD.

Furthermore, although the concept of ML is not new, in the past years, the fluid dynamics and fluid-structure interaction communities have witnessed enough applications in machine learning (ML) thanks to the advances in algorithms, computing power, affordable memory, and abundance of data. In the literature, several

approaches for applying ML to FSI have been explored in [63, 107, 108, 127, 174]. In [174], scholars have applied ML to circumvent potential stability issues associated with Galerkin projection to build a novel hydro-elastic reduced-order model with applications to design optimization. Moreover, scholars in [126], have explored a ML based framework for the reduced-order modelling (ROM) of high-speed fluid-structure interaction. Here, Convolution Neural Networks(CNNs) were used to reconstruct the flow, leading to a more general formulation for predicting the temporal coefficients of the POD reduced basis. Other researchers have modelled most practical FSI problems in a highly unstructured and conformal body mesh.

The majority of the methodologies reported above, has been used the Finite Element Method (FEM) as full-order model for data collection. Although some works have been done in the framework of reduced-order models with the FVM, such as [28, 42, 42, 66, 73, 142, 156, 158, 161, 183]. However, none of these works have studied FSI problems in the FVM. To the best of our knowledge, only the following works [42, 44, 111] have considered the fluid-structure interactions problems in FVM. The work presented in [42] proposed a ROM approach for transient modelling accounting for the presence of multiple objects in nonlinear cross-flows. The authors employed a technique based on the moving domain and immersed boundary method to overcome the challenge of handling moving boundaries due to the movements of multiple objects. In [111], a fully coupled partitioned fluid–structure interaction methodology has been developed for transonic aeroelastic structures undergoing nonlinear displacements. The Euler equations, written in an arbitrary Lagrangian–Eulerian coordinate frame are solved in the fluid domain, whereas the structure is represented by a quadratic modal reduced-order model. In [44], a contribution to the development of model order reduction techniques to reduce the computational complexity of high-dimensional aeroelastic models has been carried out also for fluid-structure interaction taking into account moving and/or deforming meshes.

1.2.3 Machine learning for CFD

The ubiquity of Machine Learning (ML) techniques motivated several researchers in the fluid mechanics community to implement and adapt them for several Computational Fluid Dynamics (CFD) applications [50,170]. As a subset of ML, deep learning consists in the use of highly multilayered neural networks in order to obtain complex maps from large and rich datasets to selected quantities of interest. The tremendous success of deep learning [87] in various other fields such as imaging, finance, robotics prompted their potential applications to fluid mechanics problems [109, 172]. More recently, [141], two new non-linear manifold hyper-reduced methods with reduced over-collocation and teacher–student training of a compressed decoder based on least-squares Petrov-Galerkin have been developed. The capabilities of the methodologies were tested on a 2D non-linear conservation law and a 2D shallow water models, and the results obtained were compared with a purely data-driven method in which the dynamics is evolved in time with a LSTM network. Also, in [140], a new intrusive and explicable methodology for reduced-order modelling that employs neural networks for solution manifold approximation, without discarding the physical and numerical models underneath during the predictive/online stage was presented. The authors focused on the use of auto-encoders to compress further

the dimensionality of linear approximants of solution manifolds, achieving in the end a nonlinear dimension reduction. After having obtained an accurate nonlinear approximant, they sought for the solutions on the latent manifold with the residual-based nonlinear least-squares Petrov-Galerkin method, opportunely hyper-reduced in order to be independent of the number of degrees of freedom. Along with the above technique, new adaptive hyper-reduction strategies were developed together with the employment of local nonlinear approximants. They tested the effectiveness of their methodology on two nonlinear time-dependent parametric benchmarks, involving a supersonic flow past a NACA airfoil with varying Mach number and an incompressible turbulent flow around the Ahmed body with changing slant angle. Not long ago, convolutional neural networks (CNNs) [112] are used to develop a novel nonlinear modal decomposition method which performed superior to the traditional POD. Turbulence modelling has received great attention from the data-driven practitioners of fluid mechanics community and has lead to excellent works for which a detailed review is available in [49]. An attempt to construct the high-resolution flow field from relatively under-resolved turbulent flow field data is described in [57]. Maulik et al. [102] used neural networks to propose a data-driven turbulence closure framework, which is used for the sub-grid modelling of kraichnan turbulence. Radial basis feed forward neural networks have been used in modelling the turbulence of subsonic flows around NACA0012 airfoils [184]. The perturbations in eigenvectors of RANS model are represented via machine learning based methods in [177]. Recurrent Neural Networks (RNNs) [17] have been used in devising a novel modal predictive control framework which exploits low-rank features inherent in a fluid flow problem and predict the quantities relevant to the control of flow. Deep neural network architectures are widely used among the computer graphics community for flow field simulations. [84] described one of the initial attempts to integrate the traditional fluid solvers with machine learning techniques. Random forests were employed to approximate the governing equations of fluid flow with Lagrangian description to predict positions and velocities of the particles. The problem of liquid splash modelling and prediction is solved by using neural networks to learn the regression of splash formation [167]. A data-driven method based on LSTM networks is proposed to infer the temporal evolution of pressure fields coming from the Navier-Stokes solvers in [175]. [179] proposed a generative adversarial networks (GANs) for the problem of reconstructing super resolution smoke flows. Another applications of GANs were presented in [41, 77], where the authors [77] developed a novel generative model to synthesize fluid flows from a set of reduced parameters and authors in [41] presented "GAROM", a new approach for reduced-order modelling based on GANs. The complex behaviour of liquids over a parameter range is mapped onto a reduced representation based on space-time deformations in [19]. The above is achieved via Generative Neural Networks (GNNs) with the introduction of a novel deformation-aware loss function. The accuracy of deep learning model for the inference of Reynolds Averaged Navier-Stokes (RANS) solutions is investigated via state-of-the-art U-net architecture with numerous trained network architectures [162]. Bhatnagar et al. [16] used convolutional neural networks to predict the velocity and pressure field in unseen flow conditions and geometries for a given pixelated shape of the object. Physics Informed Neural Networks (PINNs) were developed in [136] by considering the residual of the NSEs as the loss function in the training procedure. An attempt to develop hybrid models combining the

capabilities of POD with deep learning models is described in [172]. Carlberg et al. [29] used Auto-Encoders (AEs) for reconstructing the missing CFD data from the data saved at few time instances of high-fidelity simulations. As it is a common knowledge now that Convolutional Neural Networks (CNNs) are very good at extracting features, in the work of Miyanawala et al. [106], the authors showcased the advantages of CNNs for the data-driven ROM of unsteady wake flows in terms of feature extraction. In the quest for more accurate reduced-order modelling, the concept of AEs has attained great success in image tracking, scene labelling is explored in a series of papers by fluid mechanics community [54, 88]. A more complex variant of AEs is the convolutional recurrent auto-encoder networks, which are obtained by the integration of CNN's with RNN's. Recently, such models have been applied for benchmark fluid flow problem of lid-driven cavity [60].

In addition to applications specific to fluid mechanics, the research community has explored the more general area of reduced-order modelling of nonlinear dynamics via data-driven methods such as ERA (eigensystem realization algorithm), Dynamic Mode Decomposition (DMD) and pure machine learning approaches. In that regard, Koopman theory has gained a lot of traction among the fluid mechanics community. It has been used for linear control [80, 129] and modal decomposition [94, 104]. An excellent analogy between the Singular Value Decomposition (SVD) with Koopman theory is presented in [81]. Marko et al. [24] presented a detailed review on the applications of Koopman theory for nonlinear dynamical systems. Building on the mathematical framework of Koopman theory, several attempts have been made to develop deep learning models for nonlinear dynamical systems [120, 121, 182]. The work presented in [119] and [99] have provided further evidence to the strong correlation between Koopman theory and ML based nonlinear model order reduction. Brunton et al. [22] have provided an excellent review on the applications of machine learning for CFD problems. However, in many situations, we may not have the knowledge to understand the underlying physical laws which govern a process. In that regard, there have been several instances where machine learning was applied to learn the hidden governing equations of fluid flow [23, 30, 39, 95, 120, 135, 143]. A completely new alternative to the traditional, Neural Networks (NNs) where the derivative of the hidden state is parametrized via a neural network instead of specifying a discrete sequence of hidden layers, was studied in [33]. This new family of deep neural network has been quickly adopted by the research community as a whole and also employed in applications specific in [101, 144].

1.3 Thesis Contributions

This thesis aims to construct a hybrid reduced-order model (ROM) for segregated fluid-structure interaction solvers in an Arbitrary Lagrangian-Eulerian (ALE) approach for incompressible flows both in laminar and turbulence regime. The prerequisites for achieving this goal are the following

- Firstly, the design and implementation of an algorithm based on Proper Orthogonal Decomposition (POD) that combines the classical Galerkin projection and radial basis networks. Benchmarking results in Section 4.2 show the stability and accuracy of the proposed method with respect to the high-dimensional model by capturing transient flow fields, more importantly, the

forces acting on the moving object, and also the structure displacement.

- Secondly, the attention is shifted to the treatment of turbulent flows. We expand upon the prior algorithm to construct an additional algorithm that combines the above standard Galerkin projection and two data-driven methods, namely the radial basis networks, and neural networks/ long short term memory. Simulation results on the pitch-plunge airfoil at a high Reynolds number ($Re = 10^7$) in Section 4.3 demonstrate the ROM's ability to accurately capture the physics of fluid-structure interaction phenomena.

- Thirdly, we propose a hyper-reduction algorithm to further reduce the expensive computational cost associated with the Galerkin projection method. We introduce a ROM based on Empiral Lagrangian Interpolation Method (ELIM) for selecting some optimal locations in the domain of interest. The preliminary results in Section 4.4 are tested on the reconstruction of the solution of the Burger's equation over a backward facing step.

The novelty of the proposed reduced-order models (ROMs) is in its generality and versatility. It is designed to operate with Finite Volume Method (FVM) and can be extended to compressible flows and other Full-order Models (FOM) regardless of the mesh motion approaches and turbulence closure models used at the offline stage. In contrast, of most reduced-order models (ROMs) in the literature, this ROM has introduced a reduced-order approximation of the velocity $\boldsymbol{u}$, pressure p, eddy viscosity ν_t, and grid motion. The motivation behind having a reduced-order version of the pressure, the eddy viscosity, and grid node displacement fields can be summarized by the following items

- Having an accurate prediction of the pressure field at the reduced-order level is vital. This is because of the need to recover certain performance indicators, which highly depend on the pressure field. An example of such performance indicators is the fluid dynamics forces (lift, and drag) acting on the surface of bodies immersed in the flow.

- The predicted eddy viscosity and grid node displacement are essential for having a full reduced model for fluid-structure interaction problems.

The hybrid ROM proposed in this thesis assumes that each of the fluid dynamics variables of velocity $\boldsymbol{u}$, pressure p, eddy viscosity ν_t, and grid node displacement have a different set of reduced degrees of an analogous projection process should ideally be used for the particular turbulence model equations. But this would need the development of a turbulent ROM for every unique turbulence model. With so many turbulence models at our disposal, the final choice would be difficult. This is due to the consequences of the quantity of distinct ROMs that must be created and concurrently tracked for prospective updates. Thus, in contrast to the velocity and pressure example, the data-driven approaches approximate the eddy viscosity and the grid mesh motion in the hybrid reduced-order model suggested here.

1.4 Thesis Structure

This thesis comprises fourth chapters which are organized as it follows

- Chapter 1 addresses some motivations on the study of FSI and contents of this thesis. It also gives an idea of why it is relevant to develop reduced-order models for FSI problems in CFD using a Finite Volume Method (FVM). Additionally, it provides a discussion on projection-based ROMs, a literature on some related works on ROMs for FSI problems, and also in Machine Learning (ML) for Computational Fluid Dynamics (CFD).

- Chapter 2 deals with the problem of interest at the full-order level, which are the incompressible Navier–Stokes equations in 2D. It explains how the Finite Volume discretization of this problem is done, and then it lays out the algorithms employed for solving the governing equations. In the same chapter, the governing equations of a rigid structure are discussed in both 1D and 2D problems. As this thesis discussed FSI, the coupling conditions at the interface and mesh motion techniques are also considered.

- Chapter 3 proposes the Proper Orthogonal Decomposition (POD) methodology for the generation of reduced-order spaces. After that, the reduced algorithm based on the PIMPLE algorithm is discussed in SubSection 3.3.2. Afterwards, in the same chapter, several ROMs based on the POD are presented. The POD is combined with radial basis functions for the mesh motion prediction and with machine learning algorithms for the prediction of the eddy viscosity. These ROMs are used for the reduction of the full-order models used in this thesis. Later, a mathematical theory of an efficient projection for problems in moving domains is discussed. The chapter is closed with empirical Lagrangian interpolation (ELI) method in multi-dimension in Section 3.6.

- Chapter 4 presents the results of the ROMs developed in this thesis. The results are for benchmark cases in CFD. The main benchmarks include the flow passing an oscillation cylinder, flow passing a translational and rotational airfoil. Both test cases are studied for fluid-structure interaction problems. Lastly, 2D Burger's equation on a backward-facing step is used for testing as the preliminary results obtained for the hyper-reduction in SubSection 4.4.

Chapter 2

Full-order mathematical model

Contents

2.1 Introduction . . . 11
2.2 Fluid's Governing Equations . . . 12
2.2.1 The incompressible Navier-Stokes Equations . . . 12
2.2.2 Reynolds Average Navier-Stokes . . . 13
2.2.2.1 Boussinesq assumption . . . 13
2.2.2.2 Modelling turbulent viscosity . . . 14
2.3 Finite Volume Method . . . 15
2.3.1 The pressure gradient term . . . 16
2.3.2 Geometric Conservation Law . . . 16
2.3.3 The Convective Term . . . 17
2.3.4 The Diffusion Term . . . 17
2.4 Segregated Pressure-Based Solvers . . . 18
2.4.1 Pressure-equation for incompressible flows . . . 18
2.4.1.1 The SIMPLE algorithm . . . 18
2.4.1.2 The PISO algorithm . . . 20
2.5 Structure's governing equations . . . 22
2.5.1 Translation motion . . . 22
2.5.2 Translation and rotational motion . . . 22
2.6 Coupling conditions . . . 22
2.7 Mesh motion techniques . . . 23
2.8 Summary . . . 24

2.1 Introduction

Full-order simulation refers to the range of methods based on the high-dimensional discretization of Partial Differential Equations (PDEs). Well known discretization techniques are Finite Element Method (FEM), Finite Difference Method (FDM), and Finite Volume Method (FVM) to mention a few. In this dissertation, FVM has been used as a discretization technique. Such a numerical discretization, widely used in industrial applications [75, 147], exploits the fact that FVM locally enforces the balance of mass, momentum, and energy (in compressible flows). This chapter briefly recalls the mathematical models behind the incompressible Navier-Stokes

Equations (NSEs) in Section 2.2. In SubSection 2.2.1, the incompressible case is formulated in the ALE framework, as this case takes into account the mesh motion. Later on, an overview of the FVM is discussed in Section 2.3. In Section 2.4, a segregated pressure-based solver is discussed. After that, the notion of coupling conditions at the interface of the fluid-structure and mesh motion techniques are discussed in Section 2.6, and Section 2.7 respectively.

2.2 Fluid's Governing Equations

The equations that govern the fluid flow are based on the conservation laws of a fluid's physical properties. This section presents the systems of governing equations addressed in this thesis, the incompressible, Navier-Stokes Equations (NSEs) formulated in the ALE framework.

2.2.1 The incompressible Navier-Stokes Equations

In this Subsection, we will discuss fluid flow problems characterized by a low velocity, constant density, and a motion where viscous forces overwhelm inertia forces. We suppose to have the Mach number (Ma) below 0.3. The Ma is the ratio between the magnitude of the free-stream velocity of the fluid and the speed of sound c. The unsteady incompressible NSEs in the ALE framework for a *Newtonian fluid* are written as follows

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.1}$$

$$\frac{\delta \boldsymbol{u}}{\delta t} + \nabla \cdot [\boldsymbol{u} \otimes (\boldsymbol{u} - \boldsymbol{u}^g)] - \nu \nabla^2 \boldsymbol{u} = -\frac{1}{\rho} \nabla p. \tag{2.2}$$

$$\frac{\partial \Omega(t)}{\partial t} + \nabla \cdot \boldsymbol{u}^g = 0 \tag{2.3}$$

Eq. (2.1) is the continuity equation, Eq. (2.2) is the NSEs (momentum equation), and Eq. (2.3) is the Geometrical Conservation Law (GCL). In addition to Eqs. (2.1)–(2.3), *initial and boundary conditions* are incorporated for a well-defined problem.

Also, $\Omega(t)$ is the deformed domain at the time t, $\boldsymbol{u}$ is the velocity field, p is the pressure field, ν is the kinematic viscosity, and $\boldsymbol{u}^g$ is the velocity of the cell displacement. The time derivative in the ALE framework is given by the following relationship

$$\frac{\delta}{\delta t} = \frac{\partial}{\partial t} + \boldsymbol{u}^g \nabla. \tag{2.4}$$

The grid which moves in space must also obey the conservation law [166], which is stated as "the change in volume/ area of each control volume between time t^n and t^{n+1} must equal the volume/ area swept by the cell's boundary during $\Delta t = t^{n+1} - t^n$" as expressed in Eq. (2.3).

2.2.2 Reynolds Average Navier-Stokes

The exact unsteady Reynolds Averaged Navier-Stokes (RANS) Equations based on Reynolds decomposition, as shown in Eq. (2.7) are given as follows

$$\nabla \cdot \bar{\boldsymbol{u}} = 0, \tag{2.5}$$

$$\frac{\delta \bar{\boldsymbol{u}}}{\delta t} + \nabla \cdot (\bar{\boldsymbol{u}} \otimes (\bar{\boldsymbol{u}} - \boldsymbol{u}^g)) = -\frac{1}{\rho}\nabla \bar{p} + \nu \nabla^2 \bar{\boldsymbol{u}} - \frac{1}{\rho}\nabla \cdot (\rho \overline{\boldsymbol{u}'\boldsymbol{u}'}), \tag{2.6}$$

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}'(\boldsymbol{x}, t) + \underbrace{\frac{1}{T}\int_0^T \boldsymbol{u}(\boldsymbol{x}, t) dt}_{\bar{\boldsymbol{u}}(\boldsymbol{x})}, \tag{2.7}$$

where $\nu \nabla^2 \bar{\boldsymbol{u}}$ being the viscous stress, $\boldsymbol{u}'$ is the velocity fluctuations due to turbulence, and $\frac{1}{\rho}\nabla \cdot (\rho \overline{\boldsymbol{u}'\boldsymbol{u}'})$ is the Reynolds stress. For a detailed derivation of the RANS, the interested reader can refer to [38, 138].

2.2.2.1 Boussinesq assumption

Solving Eqs. (2.5)–(2.7) involves modelling the Reynolds stress. One of the methodologies used to model the Reynolds stresses is the Boussinesq assumption, which is expressed as follows,

$$-\rho \overline{u_i' u_j'} = 2\nu_t S_{ij} - \frac{2}{3}\rho k \delta_{ij}. \tag{2.8}$$

where

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right). \tag{2.9}$$

and $k = \frac{1}{2}\overline{u_i'^2}$. Hence, with this assumption, it is necessary to define or compute the turbulent viscosity scalar quantity ν_t in all the flow domains to close the system of equations. The final set of equations is given by

$$\nabla \cdot \bar{\boldsymbol{u}} = 0. \tag{2.10}$$

$$\frac{\delta \bar{\boldsymbol{u}}}{\delta t} + \nabla \cdot (\bar{\boldsymbol{u}} \otimes (\bar{\boldsymbol{u}} - \boldsymbol{u}^g)) = \nabla \cdot \left[\frac{1}{\rho}(\nu + \nu_t)\nabla \bar{\boldsymbol{u}})\right] - \frac{1}{\rho}\left(\nabla \bar{p} + \frac{2}{3}\rho \nabla k\right). \tag{2.11}$$

$$\frac{\partial \Omega(t)}{\partial t} + \nabla \cdot \boldsymbol{u}^g = 0. \tag{2.12}$$

$\frac{2}{3}\rho \nabla k$ being the normal stresses arising from the Boussinesq hypothesis, ν_t is the so-called turbulent viscosity, $\nu_{eff} = \nu + \nu_t$ is called the effective viscosity, and k is the turbulent kinetic energy. The quantity ν_t is suitably a viscosity only from the dimensional point of view, and it is called viscosity considering the analogy of the Boussinesq approximation and with the shear stress relations in a Newtonian fluid. The real viscosity is a property of the fluid and not its motion. A variety of methodologies are available in the literature to calculate the eddy viscosity.

The final equation to be solved is given as follows

$$\nabla \cdot \bar{\boldsymbol{u}} = 0, \tag{2.13}$$

$$\frac{\delta \bar{\boldsymbol{u}}}{\delta t} + \nabla \cdot (\bar{\boldsymbol{u}} \otimes (\bar{\boldsymbol{u}} - \boldsymbol{u}^g)) = \frac{1}{\rho}\nabla \cdot [(\nu + \nu_t)\nabla \bar{\boldsymbol{u}})] - \frac{1}{\rho}\nabla \bar{p}, \tag{2.14}$$

$$\frac{\partial \Omega(t)}{\partial t} + \nabla \cdot \boldsymbol{u}^g = 0. \tag{2.15}$$

$\bar{\boldsymbol{u}}$ is the average velocity field, and $\bar{p}$ the average pressure field. A variety of methodologies are available in the literature to calculate the eddy viscosity. This work uses the $k-\omega$ SST (shear stress transport) introduced in [103].

2.2.2.2 Modelling turbulent viscosity

The SST $k-\omega$ turbulence model is a two-equation eddy-viscosity model that has become very popular. It is based on $k-\epsilon$ and $k-\omega$ models to get the best of both. It uses the $k-\omega$ model in the boundary layer (BL) and the $k-\epsilon$ model in free stream [122]. A full description of the two aforementioned turbulence models can be found in [169, 176]. The relation between the dissipation rate ϵ and specific dissipation rate ω is given by

$$\epsilon = C_\mu k\omega \quad \text{with} \quad C_\mu = 0.09. \tag{2.16}$$

$k[\frac{m^2}{s^2}]$ being turbulent kinetic and $\epsilon \equiv -\frac{dk}{dt}[\frac{m^2}{s^3}]$ the energy dissipation rate. With these two quantities, it is possible to form a length scale $L = k^{\frac{3}{2}}\epsilon$ (being the size of large energy-containing eddies in a turbulent flow), a timescale $\tau = \frac{k}{\epsilon}$, and consequently the turbulent viscosity:

$$\nu_t = C_\mu \frac{k^2}{\epsilon} \quad \text{in the } k-\epsilon \text{ model.} \tag{2.17}$$

or

$$\nu_t = \rho \frac{k}{\omega} \quad \text{in the } k-\omega \text{ model.} \tag{2.18}$$

The SST $k-\omega$ model does produce a bit too large turbulence levels in regions with large normal strain, like stagnation regions and regions with strong acceleration. This tendency is much less pronounced than with a normal $k-\epsilon$ model, though. The connection of the apparent viscosity to the mean flow variables, in the SST $k-\omega$ model, is done as follows

$$\nu_t = \frac{a_1 k}{\max(a_1\omega, SF_2)} \tag{2.19}$$

■ To summarize, the model equations of the $k-\omega$ SST model are:

$$\frac{\bar{D}k}{\bar{D}t} = \frac{\partial}{\partial x_j}\left[\left(\frac{\nu_t}{\sigma_k} + \nu\right)\frac{\partial k}{\partial x_j}\right] + \tilde{\mathcal{P}}_k - \omega k \beta^*. \tag{2.20}$$

$$\frac{\bar{D}\omega}{\bar{D}t} = \frac{\partial}{\partial x_j}\left[\left(\frac{\nu_t}{\sigma_\omega} + \nu\right)\frac{\partial \omega}{\partial x_j}\right] + \mathcal{P}_\omega - C_{\omega 2}\omega^2 + 2(1-F_1)\frac{\sigma_{\omega 2}}{\omega}\frac{\partial \omega}{\partial x_j}\frac{\partial k}{\partial x_j}. \tag{2.21}$$

Where,

$$\tilde{\mathcal{P}}_k = \min(\mathcal{P}_k, 10\beta^* k\omega) \quad \text{with} \quad \mathcal{P}_k = \nu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) \frac{\partial \bar{u}_i}{\partial x_j} \tag{2.22}$$

$$\mathcal{P}_\omega = C_{\omega 1} S^2 \quad \text{with} \quad S = \sqrt{2 s_{ij} s_{ij}} \tag{2.23}$$

$$a_1 = \frac{5}{9} \tag{2.24}$$

$$\beta^* = \frac{9}{100}, \quad C_{\omega 1} = 0.44, \quad C_{\omega 2} = 0.0828, \quad \sigma_\omega = 0.5, \quad \sigma_{\omega 2} = 0.0856 \tag{2.25}$$

$$F_2 = \tanh\left(\left[\max\left(\frac{\sqrt{k}}{\omega \beta^* y}, \frac{500\nu}{y^2 \omega} \right) \right]^2 \right) \tag{2.26}$$

$$F_1 = \tanh\left(\left[\min\left(\max\left(\frac{2\sqrt{k}}{\omega \beta^* y}, \frac{500\nu}{y^2 \omega} \right), \frac{4 C_{\omega 2} k}{C D_{k\omega} y^2} \right) \right]^4 \right) \tag{2.27}$$

$$C D_{k\omega} = \max\left(2 \frac{\sigma_{\omega 2}}{\omega} \frac{\partial \omega}{\partial x_j} \frac{\partial k}{\partial x_j}, 10^{-10} \right) \tag{2.28}$$

In the incoming section, we discuss the FV discretization technique.

2.3 Finite Volume Method

In this section, the FVM for the incompressible NSEs is presented. The standard FVM aims to discretize the system of PDEs written in integral form following [110]. The present uses a 2-dimensional tessellation. N_h will represent the dimension of the Full-order Models (FOM) which is the number of control volumes in the discretized problem. The following addresses the discretization methodology of the momentum and continuity equations. In particular, a segregated approach is used to solve the momentum and continuity equations inspired by Rhie-Chow interpolation [139].

The first step towards FVM is to divide the domain Ω into a tessellation $\mathcal{T}$ composed by a certain number N_h of cells Ω_i, so that: $\mathcal{T} = \{\Omega_i(\cdot)\}_{i=1}^{N_h}$ and $\bigcup_{i=1}^{N_h} \Omega_i(\cdot) = \Omega$ and $\Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j$ so that every cell Ω_i is a non-convex polyhedron. In the following, to simplify the notation, $\Omega_i(\cdot) = \Omega_i$ and $S_i = \partial\Omega_i$. S_i being the total surface related to cell Ω_i. One can see in Figure 2.1 a visualization between two neighbour cells of the mesh. The unsteady momentum equation written in its integral form for every cell of the tessellation reads as follows,

$$\int_{\Omega_i} \frac{\delta \bar{\boldsymbol{u}}}{\delta t} d\Omega_i + \int_{\Omega_i} \nabla \cdot [\bar{\boldsymbol{u}} \otimes (\bar{\boldsymbol{u}} - \boldsymbol{u}^g)] d\Omega_i - \int_{\Omega_i} \nu_{eff} \nabla^2 \bar{\boldsymbol{u}} d\Omega_i + \int_{\Omega_i} \nabla \bar{p} d\Omega_i = 0. \tag{2.29}$$

$\nu_{eff} = \nu + \nu_t$ being the effective viscosity and the sum of the ν (molecular viscosity) and turbulent viscosity. In the sequel, the FOM is analysed term by term.

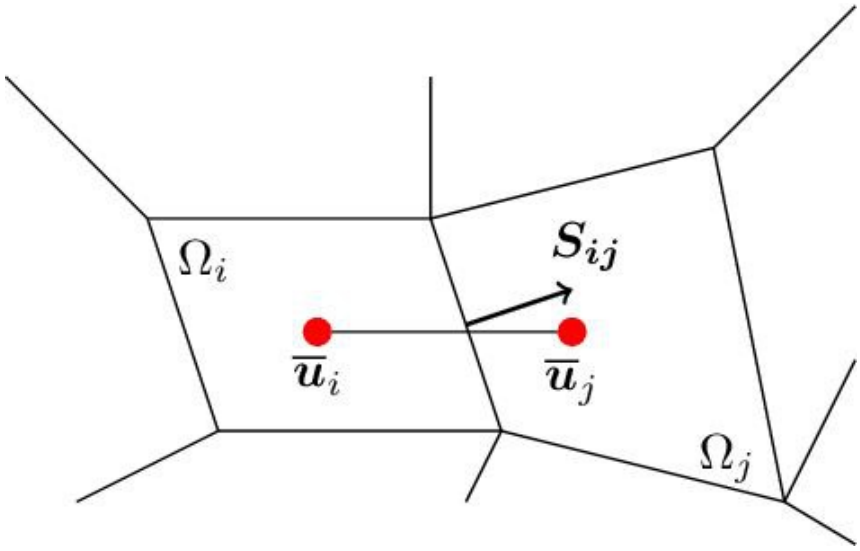


Figure 2.1: Relation between two neighbour cells Ω_i and Ω_j of the tessellation $\mathcal{T}$ for a given variable $\bar{\boldsymbol{u}}$.

2.3.1 The pressure gradient term

The pressure gradient term is discretized using Gauss's theorem.

$$\int_{\Omega_i} \nabla \bar{p} d\Omega_i = \int_{S_i} \bar{p} d\boldsymbol{S} \approx \sum_j \boldsymbol{S}_{ij} \bar{p}_{ij}, \tag{2.30}$$

where $\boldsymbol{S}_{ij}$ is the oriented surface dividing the two neighbour cells Ω_i and Ω_j. $\bar{p}_{ij}$ is the pressure evaluated at the centre of the face $\boldsymbol{S}_{ij}$.

2.3.2 Geometric Conservation Law

The GCL can also be written in integral form.

$$\frac{\delta}{\delta t} \int_{\Omega_i} d\Omega_i + \int_{S_i} \boldsymbol{u}^g \cdot \boldsymbol{n} dS_i = 0 \tag{2.31}$$

$\boldsymbol{n}$ being the outward unit normal vector on the boundary surface. By multiplying Eq. (2.31) by ρ and using the incompressibility constraint, we obtain the following relationship

$$\int_{\partial\Omega_i} \boldsymbol{u}^g \cdot \boldsymbol{n} dS_i = 0. \tag{2.32}$$

This means there is no need to consider the grid velocity in the continuity Eq. (2.1). Eq. (2.31) can be rewritten in its discrete form as follows

$$\frac{\delta \Omega_i}{\delta t} = -\sum_{j \in S_i} \boldsymbol{u}^g \cdot \boldsymbol{S}_{ij}. \tag{2.33}$$

2.3.3 The Convective Term

The convective term can be discretized as follows, using Gauss's theorem.

$$\int_{\Omega_i} \nabla \cdot [\boldsymbol{u} \otimes (\boldsymbol{u} - \boldsymbol{u}^g)] d\Omega_i = \int_{\Omega_i} \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) d\Omega_i - \int_{\Omega_i} \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}^g) d\Omega_i \tag{2.34}$$

$$= \int_{S_i} d\boldsymbol{S}_i \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) - \int_{S_i} d\boldsymbol{S}_i \cdot (\boldsymbol{u} \otimes \boldsymbol{u}^g) \tag{2.35}$$

$$= \sum_{j \in S_i} \boldsymbol{u}_{ij} F_{ij} - \sum_{j \in S_i} \boldsymbol{u}_{ij} (\boldsymbol{u}^g{}_{ij} \cdot \boldsymbol{S}_{ij}). \tag{2.36}$$

Here, $\bar{\boldsymbol{u}}_{ij}$ is the velocity evaluated at the centre of the face $\boldsymbol{S}_{ij}$, and $\boldsymbol{F}_{ij} = \bar{\boldsymbol{u}}_{ij} \cdot \boldsymbol{S}_{ij}$ is the flux of the velocity through the face $\boldsymbol{S}_{ij}$. This procedure underlines two considerations. The first one is that $\bar{\boldsymbol{u}}_{ij}$ is not straightly available, in the sense that all the variables of the problem are evaluated at the centre of the cells. At the same time, the velocity is evaluated at the centre of the face. Many different techniques are available to obtain it. However, the basic idea behind them all is that the face value is obtained by interpolating the values at the centre of the cells. The second clarification is about fluxes: during an iterative process for the resolution of the equations, they are calculated using the velocity obtained at the previous step so that the non-linearity is easily resolved.

2.3.4 The Diffusion Term

The diffusion term is discretized as follows,

$$\int_{\Omega_i} \nu_{eff} \nabla^2 \bar{\boldsymbol{u}} d\Omega_i = (\nu_{eff})_i \int_{S_i} d\boldsymbol{S} \cdot (\nabla \bar{\boldsymbol{u}})\, d\Omega_i \approx \sum_j (\nu_{eff})_{ij} \boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij}, \tag{2.37}$$

where $(\nu_{eff})_i$ is the effective viscosity of the $i-th$ cell, $(\nu_{eff})_{ij}$ is the effective viscosity evaluated at the centre of the face S_{ij}, and $(\nabla \bar{\boldsymbol{u}})_{ij}$ is the gradient of $\bar{\boldsymbol{u}}_{ij}$ evaluated at the centre of the face S_{ij}. As for the evaluation of the term $\boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij}$ in Eq. (2.37), its value depends on whether the mesh is orthogonal or non-orthogonal. Notice that the gradient of the velocity is not known at the face of the cell. The mesh is orthogonal if the line that connects two cell centres is orthogonal to the face that divides these two cells. For orthogonal meshes, the term $\boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij}$ is evaluated as follows,

$$\boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij} \approx \|\boldsymbol{S}_{ij}\| \frac{\bar{\boldsymbol{u}}_i - \bar{\boldsymbol{u}}_j}{\|\boldsymbol{d}_{ij}\|}, \tag{2.38}$$

where $\boldsymbol{d}_{ij}$ represents the vector connecting the centres of cells of index i and j. For visualization, see Figure 2.1. If the mesh is non-orthogonal, then a correction term has to be added to Eq. (2.38). In that case, one has to consider computing a non-orthogonal term to account for the non-orthogonality of the mesh, as given by the following relation [74]

$$\boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij} = \|\boldsymbol{\pi}_{ij}\| \frac{\bar{\boldsymbol{u}}_i - \bar{\boldsymbol{u}}_j}{\|\boldsymbol{d}_{ij}\|} + \boldsymbol{k}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij}. \tag{2.39}$$

Herein, $\boldsymbol{S}_{ij} = \boldsymbol{\pi}_{ij} + \boldsymbol{k}_{ij}$ and $\boldsymbol{\pi}_{ij}$ is chosen to be parallel to $\boldsymbol{S}_{ij}$ and $\boldsymbol{k}_{ij}$ to be orthogonal to $\boldsymbol{d}_{ij}$. The term $(\nabla\bar{\boldsymbol{u}})_{ij}$ is obtained through interpolation of the values of the gradient at the cell centres $(\nabla\bar{\boldsymbol{u}}_i)$ and $(\nabla\bar{\boldsymbol{u}}_j)$. The discretized forms of Eqs. (2.1)–(2.3) are written in a compact as follows:

$$\begin{bmatrix} \mathbf{A}_u & \mathbf{B}_p \\ \nabla(\cdot) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{u}}_h \\ \bar{\boldsymbol{p}}_h \end{bmatrix} = \mathbf{0}. \tag{2.40}$$

The above system matrix has a saddle point structure which is usually difficult to solve using a coupled approach. For this reason, a segregated approach is used in this dissertation, where the momentum equation is solved with a tentative pressure and later corrected by exploiting the divergence-free constraint.

2.4 Segregated Pressure-Based Solvers

After having introduced the discretization of the different terms in the NSEs, we proceed by addressing the *algorithms* used to solve the discretized system. This dissertation uses a segregated approach based on the PIMPLE algorithm. This means that the equations for each variable characterizing the system (the velocity, the pressure, and the variables characterizing turbulence) are solved sequentially and the solution of the previous equations is inserted into the subsequent equation. This aspect has to be kept in mind for the finite volume discretization strategy. *A main advantage of a segregated algorithm is the memory-efficiency, since the discretized equations need only to be stored in memory one at a time. However, a shortcoming of the segregated approach is the slow convergence of the solution as the equations are solved in a decoupled manner.* Also, with the usage of segregated approaches, the saddle-point formulation is somehow circumvented in the sense that the Ladyzhenskaya–Babushka–Bezzi condition [18, 46] is not strictly required any more since no coupled problems are here to be solved. This aspect is strongly relevant especially for the reduced-order part since no stabilization is needed. For more details and understanding, the interested reader can refer to the following studies [158] and [9].

2.4.1 Pressure-equation for incompressible flows

In this paragraph, we discuss the main idea of the PIMPLE algorithm. The PIMPLE algorithm is a mix of SIMPLE [125], and PISO [72] algorithms. This algorithm is mostly used for unsteady problems requiring a high Courant number or a dynamic mesh such as the one considered in this study. To better understand the procedure of the PIMPLE algorithm, some crucial points about both algorithms are reported in the following, as they will be useful for building the reduced-PIMPLE algorithm.

2.4.1.1 The SIMPLE algorithm

Starting with the SIMPLE algorithm, the first step is to solve the discretized momentum equation considering the pressure field of the previous iterations. The momentum matrix is divided into diagonal and extra-diagonal parts so that the

following holds:

$$\mathbf{A}_u \boldsymbol{u}_h^{n*} = \mathbf{A}\boldsymbol{u}_h^{n*} - \mathbf{H}(\boldsymbol{u}_h^{n*}), \tag{2.41}$$

with n being an index to identify a generic iteration and $\mathbf{A}_u$ satisfying the following relation:

$$\mathbf{A}_u \boldsymbol{u}_h^{n*} = -\mathbf{B}_p \boldsymbol{p}_h^{n-1}. \tag{2.42}$$

The momentum equation can be rewritten as follows:

$$\mathbf{A}\boldsymbol{u}_h^{n*} = \mathbf{H}(\boldsymbol{u}_h^{n*}) - \mathbf{B}_p \boldsymbol{p}_h^{n*} \Rightarrow \boldsymbol{u}_h^{n*} = \mathbf{A}^{-1}\mathbf{H}(\boldsymbol{u}_h^{n*}) - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h^{n-1}. \tag{2.43}$$

In an iterative algorithm, the next step is introducing a small correction to the velocity and pressure field inside the *inner loop*. Then, one can define the following relations:

$$\boldsymbol{u}_h^{n} = \boldsymbol{u}_h^{n*} + \boldsymbol{u}', \quad \boldsymbol{p}_h^{n} = \boldsymbol{p}_h^{n-1} + \boldsymbol{p}'. \tag{2.44}$$

where $\boldsymbol{u}_h^{n*}$ does not satisfy the continuity equation, and $\boldsymbol{u}_h^{n}$ does. $\square'$ are the corrections for both terms. By inserting Eq. (2.44) in Eq. (2.43), and rearranging terms give:

$$\boldsymbol{u}_h^{n} - \boldsymbol{u}' = \mathbf{A}^{-1}[\mathbf{H}(\boldsymbol{u}_h^{n}) - \mathbf{H}(\boldsymbol{u}') - \mathbf{B}_p \boldsymbol{p}_h^{n} + \mathbf{B}_p \boldsymbol{p}']. \tag{2.45}$$

From Eq. (2.45), one deduces a relation between $\boldsymbol{u}'$ and $\boldsymbol{p}'$:

$$\boldsymbol{u}' = \tilde{\boldsymbol{u}}' - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}', \tag{2.46}$$

with

$$\tilde{\boldsymbol{u}}' = \mathbf{A}^{-1}\mathbf{H}(\boldsymbol{u}'). \tag{2.47}$$

The following relation holds thanks to Eq. (2.42) :

$$\boldsymbol{u}_h^{n} = \mathbf{A}^{-1}[\mathbf{H}(\boldsymbol{u}_h^{n}) - \mathbf{B}_p \boldsymbol{p}_h^{n}]. \tag{2.48}$$

With the use of Eq. (2.46) and the divergence operator $\nabla(\cdot)$ applied to $\boldsymbol{u}_h^{n}$ in Eq. (2.44) knowing $\boldsymbol{u}'$ from Eq. (2.46), one obtain an equation that directly relates $\boldsymbol{p}'$ and $\boldsymbol{u}_h^{n*}$:

$$[\nabla(\cdot)]\left(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'\right) = [\nabla(\cdot)]\boldsymbol{u}_h^{n*} + [\nabla(\cdot)]\tilde{\boldsymbol{u}}'. \tag{2.49}$$

Which is basically the discretized Poisson Equation for Pressure (PPE) expressed in terms of the velocity and pressure corrections. In the SIMPLE algorithm, the velocity correction $\tilde{\boldsymbol{u}}'$ is unknown as $\mathbf{H}(\boldsymbol{u}')$, hence neglected, implying the following relation:

$$[\nabla(\cdot)]\left(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'\right) = [\nabla(\cdot)]\boldsymbol{u}_h^{n*}. \tag{2.50}$$

Therefore, $\boldsymbol{p}'$ is expressed as the only function of $\boldsymbol{u}_h^{n*}$ in Eq. (2.50). Then the corrected pressure is entered again in Eq. (2.43) in order to obtain a new velocity field $\boldsymbol{u}_h^{n*}$ and repeat the procedure until the pressure correction falls below a given *tolerance* and the velocity satisfy both the continuity and momentum equation.

As the $\tilde{\boldsymbol{u}}'$ is neglected, the SIMPLE algorithm converges slowly and is used mainly for steady-state simulations. Furthermore, to avoid instabilities, relaxation

factor α_p and α_u are introduced in the computation of $\boldsymbol{p}_h^n$ and $\boldsymbol{u}_h^{n*}$ as follows:

$$\boldsymbol{p}_h^n = \boldsymbol{p}_h^{n-1} + \alpha_p \boldsymbol{p}', \tag{2.51}$$

$$\boldsymbol{u}_h^{n*} = \mathbf{A}^{-1}\mathbf{H}(\boldsymbol{u}_h^{n*}) - \alpha_u \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h^{n-1}. \tag{2.52}$$

The main steps of the SIMPLE algorithm are reported in Algorithm 1.

Algorithm 1: SIMPLE algorithm for compressible flows

Input : $\boldsymbol{u}_h^{n*}$, *and* $\boldsymbol{p}_h^{n-1}$;

while $n \leq NoIs$ **do**

- Assemble momentum equation, **relax** it employing α_u;
- Solve $\mathbf{A}_u \boldsymbol{u}_h^{n*} = -\mathbf{B}_p \boldsymbol{p}_h^{n-1}$ ▷ Momentum predictor Eq. (2.42) to obtain $\boldsymbol{u}_h^{n*}$;
- Assemble Eq. (2.50) ▷ *Pressure correction step* ;
- Solve $[\nabla(\cdot)]\left(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'\right) = [\nabla(\cdot)]\boldsymbol{u}_h^{n*}$ ▷ PPE to obtain $\boldsymbol{p}'$;
- $\boldsymbol{p}_h^n \leftarrow \boldsymbol{p}'$;
- Relax $\boldsymbol{p}_h^n$ employing α_p ;
- $\boldsymbol{u}_h^n \leftarrow \mathbf{A}^{-1}\mathbf{H}(\boldsymbol{u}_h^{n*}) - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h^n$: ▷ Momentum corrector ;
- $n \leftarrow n + 1$ ▷ n is the current iteration ;

Output: $\boldsymbol{u}_h^n$, and $\boldsymbol{p}_h^n$;

2.4.1.2 The PISO algorithm

The PISO algorithm comes to play to speed up the convergence after neglecting $\tilde{\boldsymbol{u}}'$ and *computing the pressure correction* $\boldsymbol{p}'$ using Eq. (2.46). $\boldsymbol{u}'$ is computed as follows:

$$\boldsymbol{u}' = -\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'. \tag{2.53}$$

Allowing the computation of $\tilde{\boldsymbol{u}}'$ using Eq. (2.47). One defines a second velocity correction equation mirroring Eq. (2.46) as follows:

$$\boldsymbol{u}'' = \tilde{\boldsymbol{u}}' - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}''. \tag{2.54}$$

As $\boldsymbol{u}''$ in Eq. (2.54) satisfy the continuity equation, one define also a second pressure correction equation as:

$$[\nabla(\cdot)]\left(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}''\right) = [\nabla(\cdot)]\tilde{\boldsymbol{u}}'. \tag{2.55}$$

To sum up, what the PISO algorithm does more than the SIMPLE algorithm is to add a second inner loop to correct pressure and velocity. This speeds up the convergence, allowing this algorithm to be used in a transient simulation. Following the procedure described by Eqs. (2.53)–(2.55) further corrections, steps can be added, increasing both the algorithm's convergence and computational cost.

The essential steps of the PIMPLE algorithm involving mesh motion is reported in Algorithm 2. In Algorithm 2, the iterations within one time-steps are called *outer iterations*, they are performed in an *outer loop* in which the coefficients and the source matrix of the discretized equations are updated. The operations performed on linear systems with fixed coefficients are called instead *inner iterations*, and they occur in the so-called *inner loop*.

Algorithm 2: PIMPLE algorithm with dynamic mesh.

Input : Initial fields $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$, ν_t^0, and $\boldsymbol{\delta}^0$ ▷ $\boldsymbol{\delta}^0$ initial displacement;

while $n \leq n_{Max}$ **do**

  while *No. outer corrections* $\geq$ *2 and Tol* $\geq$ *maxTol* **do**

    Compute the forces; ▷ Using $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$;

    Solve the rigid body problem Eqs. (2.57) and (2.58);

    Solve the mesh motion problem ▷ To obtain $\boldsymbol{\delta}^n$ see Section 2.7;

    $\mathbf{A}_u \boldsymbol{u}_h^{n*} = -\mathbf{B}_p \boldsymbol{p}_h^{n-1}$ ▷ Assembling the momentum equation Eq. (2.50);

    Solve $\mathbf{A}_u \boldsymbol{u}_h^{n*} = -\mathbf{B}_p \boldsymbol{p}_h^{n-1}$ ▷ Momentum predictor Eq. (2.42) to obtain $\boldsymbol{u}_h^{n*}$;

    $[\nabla(\cdot)]\,(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'_h) = [\nabla(\cdot)]\boldsymbol{u}_h^{n*}$ ▷ Assembling the matrix of PPE Eq. (2.42);

    Solve $[\nabla(\cdot)]\,(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'_h) = [\nabla(\cdot)]\boldsymbol{u}_h^{n*}$ ▷ PPE to obtain $\boldsymbol{p}'$;

    $\boldsymbol{u}'_h \leftarrow -\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'_h$ ▷ Momentum corrector Eq. (2.53) ;

    while *No. inner corrections* **do**

      $[\nabla(\cdot)]\,(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}''_h) = [\nabla(\cdot)]\tilde{\boldsymbol{u}}'$ ▷ Assembling the matrix for PPE Eq. (2.55);

      Solve $[\nabla(\cdot)]\,(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}''_h) = [\nabla(\cdot)]\tilde{\boldsymbol{u}}'$ ▷ Recursively to obtain $\boldsymbol{p}''_h$;

      $\boldsymbol{u}''_h \leftarrow \mathbf{A}^{-1}\mathbf{H}(\boldsymbol{u}_h^{n*}) - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}''_h$ ▷ Momentum corrector Eq. (2.54);

    Solve turbulence and other transport quantities to obtain ν_t^n;

    $\boldsymbol{u}_h^n \leftarrow \boldsymbol{u}''_h$;

    $\boldsymbol{p}_h^n \leftarrow \boldsymbol{p}'_h$;

  Update tolerance;

  $n \leftarrow n + 1$ ▷ n here is the current time step;

Output: $\boldsymbol{u}_h^n$, $\boldsymbol{p}_h^n$, ν_t^n, and $\boldsymbol{\delta}^n$;

2.5 Structure's governing equations

In this section, the two motions structure addressed in this thesis are presented. The first one is a translational motion. The second one is a two-degree of freedom problem which take into account the rotational motion in addition to the translation one.

2.5.1 Translation motion

The one-dimensional equation of motion for a general rigid-body system can usually be written in the following form

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = \frac{F_y}{m}. \tag{2.56}$$

Where $\omega_n = \sqrt{k/m} = 2\pi f_n$ is the natural pulsation of the system, k is the spring's stiffness, m is the mass of the rigid body, $\zeta = \dfrac{c}{2m\omega_n}$ is the fraction of structural damping c with respect to critical or simply damping ratio, y is the displacement of the structure's in the transverse direction, and F_y is the lift force in the free-stream transverse direction. The fluid force F_y per unit length of the structure drives the motion of the structure.

2.5.2 Translation and rotational motion

In this dissertation, the aeroelastic structural model as depicted in Figure 4.13 is governed by a two-degree freedom pitch and plunge system. The equations of the structure's motion are

$$m\ddot{h} + c_h\dot{h} + k_h h - mb\ddot{\theta}\cos\theta + mb\dot{\theta}^2\sin\theta = F_h(t). \tag{2.57}$$

$$\boldsymbol{I}_\theta\ddot{\theta} + c_\theta\dot{\theta} + k_\theta\theta - mb\ddot{h}\cos\theta = \boldsymbol{M}_\theta(t). \tag{2.58}$$

m being the mass of the airfoil per unit span, $F_h(t)$ the sectional lift per unit span, $\boldsymbol{I}_\theta$ the sectional moment of inertia of the airfoil, $\boldsymbol{M}_\theta(t)$ is the pitching moment, $\theta(t)$ the pitch rotation, h(t) the plunge displacement, b the distance between the pivot location and the center of mass. The structural stiffness of the plunge and pitch is designated by k_h and k_θ; the related damping coefficients are c_h and c_θ. The structural frequencies are $f_\theta = (2\pi)^{-1}\omega_\theta$, and $f_h = (2\pi)^{-1}\omega_h$ with $\omega_\theta = \sqrt{k_\theta/I_{zz}}$ and $\omega_h = \sqrt{k_h/m}$. I_{zz} being the moment of inertia $\boldsymbol{I}_\theta$ in the z-direction. For a thorough introduction to structural dynamics and aero-elasticity refer to [68].

2.6 Coupling conditions

The coupling conditions of the fluid-structure is achieved at the boundary conditions on the common interface $\Gamma(t) = \partial\Omega(t)$ which all stem from simple physical principles namely the *kinematic condition* (the fluid velocity, grid velocity, and structure's velocity are continuous at the interface), *dynamic condition* (the normal stresses of the fluid and structure are continuous on the interface), and the *geometric condition*

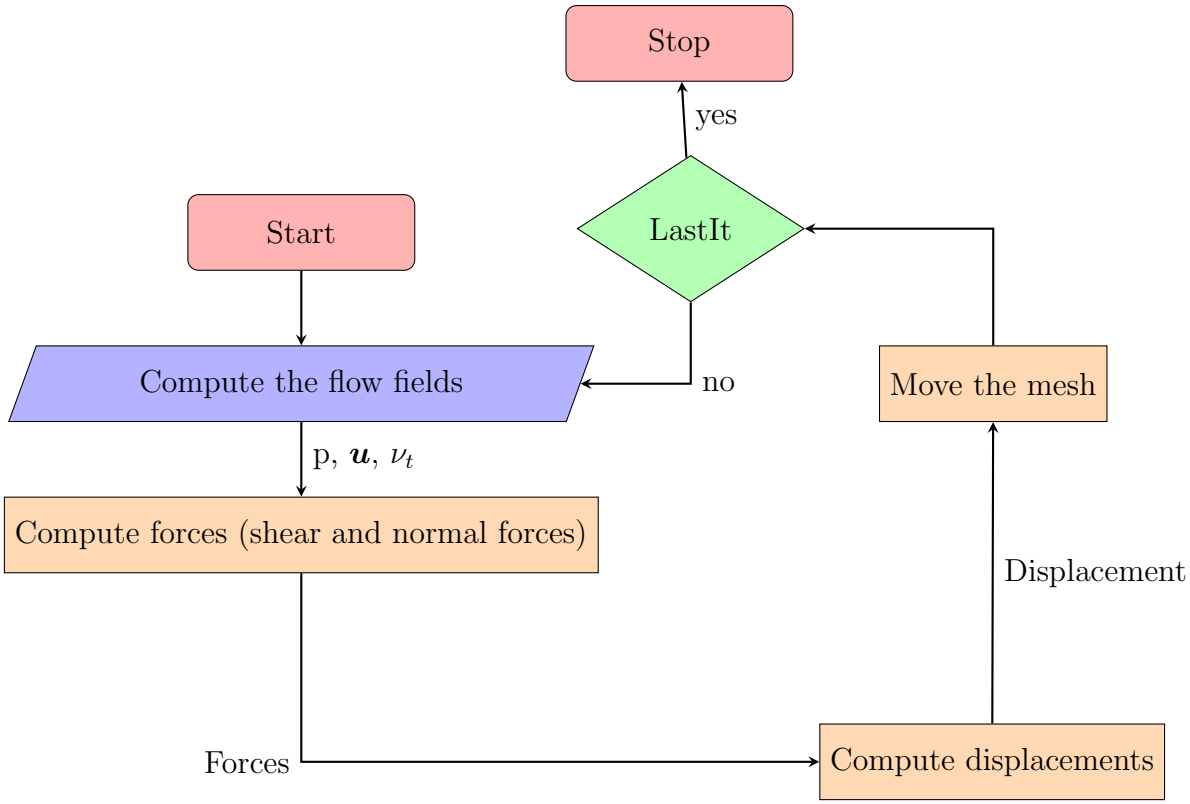


Figure 2.2: Flow chart of fluid-body motion

(the fluid and structure domain should always match. The following equations summarize the above conditions.

$$\boldsymbol{u} \cdot \boldsymbol{e}_y = \boldsymbol{u}^g \cdot \boldsymbol{e}_y = \dot{y} \quad \text{and} \quad \int_{\Gamma(t)} (\boldsymbol{\sigma}(\boldsymbol{x}, t) \cdot \boldsymbol{n}) \cdot \boldsymbol{n}_y d\Gamma + F_y = 0, \tag{2.59}$$

with $\boldsymbol{\sigma}(\boldsymbol{x}, t) = -p(\boldsymbol{x}, t)\mathbf{I} + 2\mu\boldsymbol{D}$ as, it is assumed that the fluid is Newtonian. Here, $\mathbf{I}$ is the identity tensor, and $\boldsymbol{D} = \frac{1}{2}\left(\nabla \cdot \boldsymbol{u}(\boldsymbol{x}, t) + (\nabla \cdot \boldsymbol{u}(\boldsymbol{x}, t)^T\right)$. Next, the Section 2.7 discuss the mesh motion strategy.

2.7 Mesh motion techniques

Translation and/or rotational motion of the centre of gravity (COG) are accounted by solving Newton's second law Eqs. (2.57) and (2.58) or Eq. (2.56) in the global inertial reference frame. After the linear and angular accelerations have been computed using in the case of 1D or 2D motion, translation and/or rotational kinematics are used to update the body linear and angular velocities. After calculating the motion of the rigid body, it is necessary to move the boundary as well as the mesh surrounding the body in order to maintain a good quality mesh. In this work, the mesh deformation technique used is the so-called Slerp (Spherical Linear Interpolation) as it is better at handling translation and rotational (of a solid body in the three arises XYZ) mesh deformation when it comes to cell shearing [76] and has great applications in computer vision.

2.8 Summary

In this chapter, we have traversed through five critical sections, each contributing to a comprehensive understanding of our core subject. Beginning with an in-depth analysis of foundational concepts, we laid the groundwork necessary to appreciate the subsequent intricate discussions. Section 2.2 provided a theoretical framework of the governing equations driving the fluid flows, the incompressible NSEs. The governing equations in the incompressible flows were presented in the ALE form. In Section 2.3, we delved into the Finite Volume discretization technique, and we discussed the segregated algorithms that we use in this thesis, mainly the SIMPLE, PISO, and PIMPLE algorithms. In Section 2.5, two cases of the structure's governing equations are presented. The first one considered a translation motion in the transverse direction and the second one take in addition to translation also rotational motion, leading to a two-degree motion problem of a pitching and plunging motion. In Section 2.6 the coupling conditions at the interface of the interaction between the fluid and structure is presented in the nutshell. Finally, in Section 2.7 the mesh motion techniques used in this thesis are addressed and some references for other mesh strategies for the interested reader are pointed out.

In conclusion, this chapter has provided a thorough exploration of the basic requirements, from foundational theories, of the FOM. Each section has contributed uniquely, building a layered understanding that is both deep and broad. As we move forward, the insights gained here will serve as a valuable guide, informing our ongoing inquiry and application in the dynamic landscape of our field. However, standard high-fidelity simulations are quite expensive both in time and computationally resources. In the following chapter, reduced-order models can be used to reduce computational resources, obfuscate proprietary models and speed up development cycles.

Chapter 3

Reduced-order models

Contents

3.1 Introduction . . . 25
3.2 The Proper Orthogonal Decomposition . . . 26
3.3 The segregated reduced approaches . . . 29
3.3.1 Galerkin projection . . . 29
3.3.2 Reduced-PIMPLE for incompressible flows . . . 30
3.3.3 Hyper-Reduced-PIMPLE for incompressible flows . . . 30
3.3.4 POD with interpolation for mesh motion prediction . . . 32
3.4 Machine learning for eddy viscosity prediction . . . 35
3.4.1 POD-NNs . . . 36
3.4.2 POD-RNNs . . . 37
3.5 Efficient projection in moving domains problems . . . 39
3.6 Hyper-Reduction: From projection to interpolation . . . 40
3.6.1 From projection to interpolation . . . 40
3.6.2 Empirical Lagrangian Interpolation algorithm . . . 42
3.6.3 Error analysis . . . 43
3.7 Summary . . . 44

3.1 Introduction

Reduced-order models (ROMs) can be broadly categorized into linear and non-linear reduction techniques. Both approaches aim to simplify complex systems while retaining essential dynamics, but they handle the system's behaviour differently. ROMs are simplifications of high-fidelity, complex models. They capture the behaviour of these source models so that engineers can quickly study a system's dominant effects using minimal computational resources. ROMs have become popular in the product development industry because engineers are facing market demands for shorter design cycles that produce higher-quality products. In this dissertation, linear reduction technique based on Proper Orthogonal Decomposition (POD) ([59, 85, 97, 98]) has been used to compute the subspace for the approximated solution. However, among POD other methodologies are available in the literature such as: the greedy algorithm [128, 134, 168], Proper Generalized Decom-

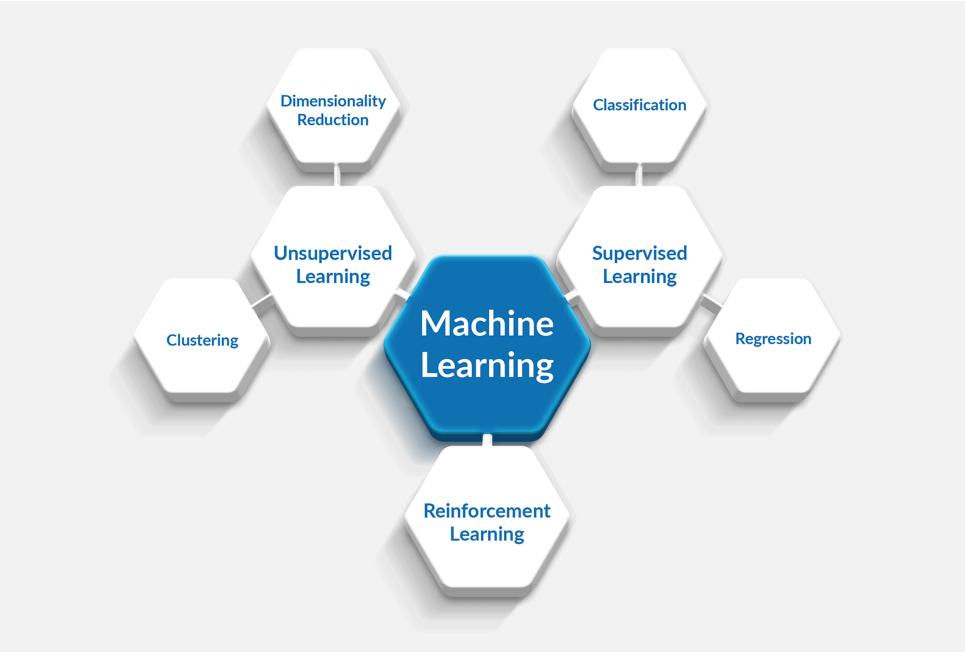


Figure 3.1: Machine Learning Algorithms [2].

position [36, 117], Dynamic Mode Decomposition [83, 133, 146]. For more details in the comparisons of these methodologies, the interested reader can refer to [14, 65].

The choice behind POD is motivated by its optimality, efficiency, and robust mathematical foundation. Additionally, its ability to significantly reduce computational complexity while maintaining high accuracy makes it an attractive option for a wide range of engineering applications. However, some limitations related to the POD should not be neglected. For example, the energy truncation of the POD basis still remains arbitrary and there is not yet a guarantee about a fixed value of the rate of "energy" captured by the first m modes which can absolutely ensure a good representation of the relevant scales of the system and therefore a good accuracy of the ROM. Finally, POD basis functions are closely related to the *reference data* from which they have been derived rather than to the *operators of the governing dynamical system*. In the following, POD details are recalled, together with the concept of POD-Galerkin projection, POD-Interpolation (POD-RBF) using Radial Basis Functions (RBF), and POD-ML.

3.2 The Proper Orthogonal Decomposition

The POD method [97, 98], also known as the Principal Component Analysis (PCA) method in machine learning, is a well-established algorithm. It is part of most machine learning toolkits and can be used for various applications such as data visualization or, for our purposes, dimensionality reduction. Figure 3.1 outlines the concept of dimensionality reduction as an algorithm of machine learning. The Auto-Encoders (AEs) can be interpreted as a flexible and nonlinear generalization of POD [26]. In [53], Baldi and Hornik showed that an AE network with one hidden layer and linear activation functions could closely resemble the standard POD/PCA.

In this thesis, POD is used to construct the low-dimensional space. POD is a compression technique where a set of numerical realizations (in time or parameter

space) is reduced into a number of orthogonal basis (spatial modes) that capture the essential information suitably combined from previously acquired system data [6]. We apply POD to a group of realizations called snapshots. It consists of computing a certain number of full-order solutions $\boldsymbol{s}_i = \boldsymbol{s}(t_i)$ where $t_i \in \boldsymbol{T}$ for $i = 1, \cdots, N$. $\boldsymbol{T}$ being the training collection of a certain number N of the time values, to obtain a maximum amount of information from this costly stage to be employed later on for a cheaper resolution of the problem. Those snapshots can be resumed at the end of the resolution all together into a matrix $\boldsymbol{S} \in \mathbb{R}^{N_h \times N}$

$$\boldsymbol{S} = \left[\boldsymbol{s}(\boldsymbol{x}, t_1), \ldots, \boldsymbol{s}(\boldsymbol{x}, t_N)\right], \tag{3.1}$$

when dealing with time-dependent problem or

$$\boldsymbol{S} = \left[\boldsymbol{s}(\boldsymbol{x}, \mu_1), \ldots, \boldsymbol{s}(\boldsymbol{x}, \mu_N)\right], \tag{3.2}$$

in the case of parameter dependency. The idea is to compute a ROM solution that can minimize the error between the obtained realization of the problem and its high-fidelity counterpart, see Eq. (3.6). In the POD-Galerkin scheme, the reduced-order solution is expressed as follows

$$\boldsymbol{s}(\boldsymbol{x}, t) \approx \boldsymbol{s}^{ROM}(\boldsymbol{x}, t) = \sum_{i=1}^{N_r} a_i(t)\boldsymbol{\phi}_i(\boldsymbol{x}) \tag{3.3}$$

or

$$\boldsymbol{s}(\boldsymbol{x}, \pi) \approx \boldsymbol{s}^{ROM}(\boldsymbol{x}, \pi) = \sum_{i=1}^{N_r} a_i(\pi)\boldsymbol{\phi}_i(\boldsymbol{x}). \tag{3.4}$$

Where $N_r \ll N_h$ (N_h is the number of cells in the computational domain) is a predefined number, $\boldsymbol{\phi}_i$ is a generic pre-calculated orthonormal basis depending only on the space while $a_i(t)$ is the temporal modal coefficients satisfying the following conditions

$$a_j(t) = \left(\boldsymbol{\phi}_j, \boldsymbol{s}(\boldsymbol{x}, t)\right)_{L^2(\Omega)}, \qquad \boldsymbol{\phi}_i \boldsymbol{\mathcal{V}}_h \boldsymbol{\phi}_j^T = \boldsymbol{\delta}_{ij}. \tag{3.5}$$

$\boldsymbol{\mathcal{V}}_h$ being the mass matrix defined by the chosen inner product. In the case of L^2 norm and FVM, $\boldsymbol{\mathcal{V}}_h$ is a diagonal matrix containing the cell volumes. The best performing functions, $\boldsymbol{\phi}_i$ in this case, are the ones minimizing the L^2 norm error Eq. (3.6) between all the reduced solutions $\boldsymbol{s}_i^{ROM}$ $\forall$ $i = 1, \cdots, N$ and their high fidelity counterparts,

$$E = \sum_{i=1}^{N} \|\boldsymbol{s}_i^{ROM} - \boldsymbol{s}_i\|_{L^2(\Omega)} = \sum_{i=1}^{N} \|\boldsymbol{s}_i - \sum_{i=1}^{N_r} \left(\boldsymbol{s}_i, \boldsymbol{\phi}_i\right)_{L^2(\Omega)} \boldsymbol{\phi}_i\|_{L^2(\Omega(t_0))}. \tag{3.6}$$

$\Omega(t_0)$ being the reference configuration of the computational domain. Note that the projection is performed with respect to, $L^2(\Omega(t))$ while POD is computed with respect to $L^2(\Omega(t_0))$. It can be shown that solving the minimization problem based

on Eq. (3.6) is equivalent to solving the following eigenvalue problem [82]

$$\boldsymbol{C}\mathbf{V} = \mathbf{V}\boldsymbol{\lambda}. \tag{3.7}$$

$\boldsymbol{C} = \boldsymbol{S}\boldsymbol{S}^T \in \mathbb{R}^{N\times N}$ being the correlation matrix between all the different training solutions of the snapshot matrix $\boldsymbol{S}$, $\mathbf{V} \in \mathbb{R}^{N\times N}$ is the matrix whose columns are the eigenvectors, and $\boldsymbol{\lambda} \in \mathbb{R}^{N\times N}$ is a diagonal matrix whose diagonal entries are the eigenvalues. The entries of the correlation matrix are defined as follows

$$\boldsymbol{C}_{ij} = \boldsymbol{s}_i^T \boldsymbol{\mathcal{V}}_h \boldsymbol{s}_j. \tag{3.8}$$

The solution of the eigenvalue problem Eq. (3.7) is equivalent to the computation of an Eq. (3.9) of the matrix $\boldsymbol{S}$, so that

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T. \tag{3.9}$$

Where $\boldsymbol{\Sigma}$ is a diagonal matrix which contains the singular values of $\boldsymbol{S}$, listed in order of decreasing magnitude. $\boldsymbol{U}$, and $\boldsymbol{W}$ are orthonormal matrices. Indeed, given Eq. (3.9), the correlation matrix can be given as follows

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T. \tag{3.10}$$

When using a POD strategy, the required basis functions are obtained through the resolution of the eigenproblem mentioned in Eq. (3.7), obtained with the method of snapshots by solving Eq. (3.6). One can compute the required basis functions as follows

$$\boldsymbol{\phi}_i = \left(N\sqrt{\lambda_i}\right)^{-1} \sum_{j=1}^{N} \boldsymbol{s}_j \boldsymbol{V}_{ji} \quad \forall i = 1, \cdots, N. \tag{3.11}$$

All the basis functions can be collected into a single matrix

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_{N_r}] \in \mathbb{R}^{N_h \times N_r}, \tag{3.12}$$

which will be used to project the high-fidelity problem onto the reduced subspace so that the final system's dimension is N_r. This procedure leads to a problem requiring a computational cost that could be much lower than the original problem. The Algorithm 3 summarizes the main steps for computing the POD basis matrix.

Algorithm 3: POD algorithm

Input : Snapshot matrix $\boldsymbol{S} \in \mathbb{R}^{dN_h \times N_t}$;

1 Construct the correlation matrix for $\boldsymbol{S}$;

2 Solve the eigen problem Eq. (3.7);

3 construct the modal basis functions Eq. (3.11);

4 extract the first r basis functions as columns of the functions matrix $\boldsymbol{\Phi}$;

Output: Basis functions matrix $\boldsymbol{\Phi}$ containing the modes basis $\boldsymbol{\phi}_i(\boldsymbol{x})$.

The next section discusses the reduced approaches, which include Galerkin projections and the data-driven techniques employed in this thesis.

3.3 The segregated reduced approaches

In fluid dynamics simulations, particularly, when dealing with incompressible flows, pressure-velocity coupling algorithms are essential for solving the Navier-Stokes Equations (NSEs). The most widely used algorithms for this purpose are the Semi-Implicit Method for Pressure Linked Equations (SIMPLE), Pressure Implicit with Splitting of Operators (PISO), and the Pressure-Implicit with Splitting of Operators (PIMPLE). When implementing ROMs, these algorithms can also be adapted to maintain efficiency and accuracy. This current section discusses the Galerkin projection and reduced algorithm based on the PIMPLE summarized in Algorithm 2 discussed above, exploring their adaptations and benefits in the context of ROMs.

3.3.1 Galerkin projection

In this section, a Galerkin projection is constructed for velocity, and pressure for incompressible flows. In the following, it is assumed that the momentum, and Poisson Equation for Pressure (PPE) are discretized and written in the following form

$$\mathbf{A}_u \boldsymbol{u}_h = \boldsymbol{b}_u, \quad \mathbf{B}_p \boldsymbol{p}_h = \boldsymbol{b}_p, \tag{3.13}$$

where

$$\mathbf{A}_u \in \mathbb{R}^{dN_h \times dN_h}, \text{ and } \mathbf{B}_p \in \mathbb{R}^{N_h \times N_h}, \tag{3.14}$$

indicate the matrices containing the terms related to velocity, and pressure for the in their discretized forms respectively.

$$\mathbf{b}_u \in \mathbb{R}^{dN_h}, \text{ and } \mathbf{b}_p \in \mathbb{R}^{N_h}, \tag{3.15}$$

are the related source terms. In addition, $\boldsymbol{u}_h$ and $\boldsymbol{p}_h$ are the vectors where all the $\bar{\boldsymbol{u}}_i$, and $\bar{p}_i$ variables are collected. $d = 2$ is the space dimension, and N_h being the number of control volumes (cells) in the mesh. In the sequel, Galerkin projection (on the fully discrete equations) is used for the construction of the reduced systems. We assume the following decomposition's introduced in Section 3.2

$$\boldsymbol{u}_h \simeq \sum_{i=1}^{N_u} a_i(\cdot)\boldsymbol{\phi}_i(\boldsymbol{x}) = \boldsymbol{\Phi}\boldsymbol{a}, \tag{3.16}$$

or,

$$\boldsymbol{p}_h \simeq \sum_{i=1}^{N_p} b_i(\cdot)\boldsymbol{\xi}_i(\boldsymbol{x}) = \boldsymbol{\Xi}\boldsymbol{b}. \tag{3.17}$$

Where, $a_i(\cdot)$ and $b_i(\cdot)$ are modal coefficients which can time-dependent, parameter-dependent or both. $\boldsymbol{\phi}_i$, and $\boldsymbol{\xi}_i$ are the basis functions corresponding to the POD modes of the velocity, and pressure fields stored respectively in

$$\boldsymbol{\Phi} \in \mathbb{R}^{dN_h \times N_u}, \tag{3.18}$$

and

$$\boldsymbol{\Xi} \in \mathbb{R}^{N_h \times N_p}. \tag{3.19}$$

In addition, N_u, and N_p being the numbers of basis functions selected for the predicted velocity, and pressure solutions respectively. $\boldsymbol{a} \in \mathbb{R}^{N_u}$, and $\boldsymbol{b} \in \mathbb{R}^{N_p}$ are the vectors containing the coefficients for the velocity expansion, while the same reads for the pressure.
The linear systems in Eq. (3.13) are projected using the respective basis functions defined in Eq. (3.16) leading to

$$\boldsymbol{A}_u^r \boldsymbol{a} = \boldsymbol{b}_u^r, \quad \boldsymbol{A}_p^r \boldsymbol{b} = \boldsymbol{b}_p^r, \tag{3.20}$$

Where $\boldsymbol{A}_u^r = \boldsymbol{\Phi}^T \mathbf{A}_u \boldsymbol{\Phi} \in \mathbb{R}^{N_u \times N_u}$, and $\boldsymbol{A}_p^r = \boldsymbol{\Xi}^T \mathbf{A}_p \boldsymbol{\Xi} \in \mathbb{R}^{N_p \times N_p}$. The resulting reduced linear systems in Eq. (3.20) can be solved using any method for dense matrices. For example, the Householder rank-revealing QR decomposition of a matrix with full pivoting is used, and it is available in the Eigen library [62].

3.3.2 Reduced-PIMPLE for incompressible flows

In this subsection, the reduced algorithm is based on the PIMPLE algorithm as described in Algorithm 2. As the main idea here is to rely on a method capable of being as coherent as possible concerning the high-fidelity problem (Algorithm 2), in the following the main steps for the reduced algorithm related to incompressible turbulent flows with moving mesh are reported in Algorithm 4.

3.3.3 Hyper-Reduced-PIMPLE for incompressible flows

In this subsection, we propose a hyper-reduced algorithm, mimicking the reduced-PIMPLE algorithm, as described in Algorithm 4. The main idea is to rely on a method capable of being as consistent as possible with respect to the reduced-PIMPLE (Algorithm 4). In the present section, we assume that the interpolation points of the Empirical Interpolation Method (EIM) method of choice [31, 32, 34, 35, 45, 114] are already available. Thus, the following paragraph focuses on how the problem is hyper-reduced. Details on how the interpolation points are obtained is discussed in SubSection 3.6.2.

From the reduced systems (3.20), one can observe that the orthogonal projection matrices $\boldsymbol{\Phi}$, and $\boldsymbol{\Xi}$ depend on the number of the degree of freedom of the full-order model system N_h as indicated in Eq. (3.18), and Eq. (3.19). Let $J = \{j_1, \cdots, j_M\} \subset \{1, 2, \cdots, N_h\}$, we define the selector operator $\boldsymbol{\mathcal{P}}_{h,M} = [\boldsymbol{e}_{j1}, \cdots, \boldsymbol{e}_{jM}] \in \mathbb{R}^{N_h \times M} : \mathbb{R}^{N_h} \to \mathbb{R}^M$ as that which projects any vector $\boldsymbol{z}_h \in \mathbb{R}^{N_h}$ on M of its N_h coordinates, so that

$$\boldsymbol{z}_{h,M} = \boldsymbol{\mathcal{P}}_{h,M} \boldsymbol{z}_h. \tag{3.21}$$

$\boldsymbol{e}_{jl}$ being the canonical vector with 1 being placed at j_l position, and M is a given number of optimal location on a given domain denoted in the literature as "best points" in [114], "magic points" in [100], or "interpolation points" in interpolation

Algorithm 4: Reduced-PIMPLE algorithm with dynamic mesh

Input : $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$, ν_t^0, $\boldsymbol{\delta}_h^0$, $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$, and $\boldsymbol{\Xi}$;

while $t \leq t_{end}$ **do**

  while *No. outer corrections* $\geq$ *2 and Tol* $\geq$ *maxTol* **do**

    Compute the forces; ▷ Using $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$;

    Solve the rigid body problem Eqs. (2.57) and (2.58) ▷ To obtain the new COG y_{new}^C;

    Compute $\boldsymbol{c} = RBF(y_{new}^C)$ Eq. (3.34);

    Reconstruct $\boldsymbol{\delta}^n = \boldsymbol{\Psi}\boldsymbol{c}$ Eq. (3.35);

    $\mathbf{A}_u \boldsymbol{u}_h^{n*} = \mathbf{b}_u$ ▷ Assembling the momentum matrix Eq. (2.41);

    Solve $\boldsymbol{\Phi}^T \mathbf{A}_u \boldsymbol{\Phi} \boldsymbol{a}^* = \boldsymbol{\Phi}^T \mathbf{b}_u$ ▷ To obtain $\boldsymbol{a}^*$ with $\mathbf{b}_u = -\mathbf{B}_p \boldsymbol{p}_h^{n-1}$;

    Reconstruct $\boldsymbol{u}_h^{n*}$ ▷ Using $\boldsymbol{a}^*$;

    $[\nabla(\cdot)]\,(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h') = [\nabla(\cdot)]\boldsymbol{u}_h^{n*}$ ▷ Assembling the matrix of PPE Eq. (2.50);

    Solve $\boldsymbol{\Xi}^T \mathbf{A}_p \boldsymbol{\Xi} \boldsymbol{b}' = \boldsymbol{\Xi}^T \boldsymbol{b}_p$ ▷ To obtain $\boldsymbol{b}'$;

    Reconstruct $\boldsymbol{p}_h'$ ▷ Using $\boldsymbol{b}'$;

    $\boldsymbol{u}_h' \leftarrow -\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h'$ ▷ Momentum corrector Eq. (2.53) ;

    while *No. inner corrections* **do**

      $[\nabla(\cdot)]\,(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h'') = [\nabla(\cdot)]\tilde{\boldsymbol{u}}'$ ▷ Assembling the matrix for PPE Eq. (2.55);

      Solve $\boldsymbol{\Xi}^T \mathbf{A}_p \boldsymbol{\Xi} \boldsymbol{b}'' = \boldsymbol{\Xi}^T \boldsymbol{b}_p$ ▷ Recursively to obtain $\boldsymbol{b}''$ where $\boldsymbol{b}_p = [\nabla(\cdot)]\tilde{\boldsymbol{u}}'$;

      Reconstruct $\boldsymbol{p}_h''$ ▷ Using $\boldsymbol{b}''$;

      $\boldsymbol{u}_h'' \leftarrow \tilde{\boldsymbol{u}}' - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h''$: ▷ Momentum corrector Eq. (2.54) ;

    Evaluate the networks using NNs or LSTM ;

    Reconstruct the new turbulent viscosity ν_t^n;

    $\boldsymbol{u}_h^{n*} \leftarrow \boldsymbol{u}_h''$;

    $\boldsymbol{p}_h^{n-1} \leftarrow \boldsymbol{p}_h^{n-1} + \boldsymbol{p}_h'$;

Output: $\boldsymbol{u}_h^n$, $\boldsymbol{p}_h^n$, ν_t^n, and $\boldsymbol{\delta}_h^n$;

theory to be more general. For example, let's reconsider Eq. (3.16) given by,

$$\boldsymbol{u}_h \simeq \sum_{i=1}^{N_u} a_i(\cdot)\boldsymbol{\phi}_i(\boldsymbol{x}) = \boldsymbol{\Phi}\boldsymbol{a}. \tag{3.22}$$

If we apply the selector operator $\boldsymbol{\mathcal{P}}_{h,M}$ on Eq. (3.16) the following holds

$$\boldsymbol{u}_{h,M} \simeq \sum_{i=1}^{N_u} a_i(\cdot)\boldsymbol{\mathcal{P}}_{h,M}^T\boldsymbol{\phi}_i(\boldsymbol{x}) = \boldsymbol{\mathcal{P}}_{h,M}^T\boldsymbol{\Phi}\hat{\boldsymbol{a}} = \boldsymbol{\mathcal{Q}}\hat{\boldsymbol{a}}. \tag{3.23}$$

With $\boldsymbol{\mathcal{Q}} \in \mathbb{R}^{M\times N_u}$ being a sub matrix of the matrix $\boldsymbol{\Phi}$ which does not depend on N_h. We can obtain the following *subsystem* when considering the momentum matrix

$$\boldsymbol{Q}^T[\boldsymbol{\mathcal{P}}_{h,M}^T\mathbf{A}_u\boldsymbol{\mathcal{P}}_{h,M}]\boldsymbol{Q}\hat{\boldsymbol{a}} = \boldsymbol{Q}^T[\boldsymbol{\mathcal{P}}_{h,M}^T\mathbf{b}_u]. \tag{3.24}$$

The above subsystem (3.24) can be written as follows

$$\boldsymbol{Q}^T\mathbf{A}_u^M\boldsymbol{Q}\hat{\boldsymbol{a}} = \boldsymbol{Q}^T\mathbf{b}_u^M. \tag{3.25}$$

With,

$$\boldsymbol{Q}^T = \boldsymbol{\Phi}^T\boldsymbol{\mathcal{P}}_{h,M}, \quad \mathbf{A}_u^M = \boldsymbol{\mathcal{P}}_{h,M}^T\mathbf{A}_u\boldsymbol{\mathcal{P}}_{h,M}, \quad \mathbf{b}_u^M = \boldsymbol{\mathcal{P}}_{h,M}^T\mathbf{b}_u. \tag{3.26}$$

One can notice that the hyper-reduced subsystem (3.25) has the same structure as the reduced systems (3.20) but now the matrices, thanks to the selection operator, end up being assembled on a space with smaller degree of freedom (as many as the magic points).

The system Eq. (3.24), can be easily derive as follows

$$\begin{aligned}\mathbf{A}_u\boldsymbol{u}_h = \boldsymbol{b}_u &\Longleftrightarrow \boldsymbol{\Phi}^T\mathbf{A}_u\boldsymbol{\Phi}\boldsymbol{a} = \boldsymbol{\Phi}^T\boldsymbol{b}_u \\ &\Longleftrightarrow \boldsymbol{\Phi}^T\boldsymbol{\mathcal{P}}_{h,M}[\boldsymbol{\mathcal{P}}_{h,M}^T\mathbf{A}_u\boldsymbol{\mathcal{P}}_{h,M}]\boldsymbol{\mathcal{P}}_{h,M}^T\boldsymbol{\Phi}\hat{\boldsymbol{a}} = \boldsymbol{\Phi}^T\boldsymbol{\mathcal{P}}_{h,M}[\boldsymbol{\mathcal{P}}_{h,M}^T\boldsymbol{b}_u].\end{aligned} \tag{3.27}$$

The above derivation can be also obtained for the PPE. It is worth mentioning that the selector operator $\boldsymbol{\mathcal{P}}_{h,M}^T$ should take also into account the neighbour cells required to complete the stencils. In the following, the main steps for the hyper-reduced algorithm related to incompressible turbulent flows with moving mesh are reported in Algorithm 5. It is worth mentioning that, the selector operator can be applied to Eq. (3.48) as another technique to further reduce the computational cost.

3.3.4 POD with interpolation for mesh motion prediction

This section presents a method to further reduce the computational cost associated with the mesh motion part in the system. Along with reducing the number of degrees of freedom within the system, the advantage of this methodology is that it does not require dependence on the mesh motion technique used at the full-order level in order to calculate mesh deformation. As will be discussed, the methodology combines proper orthogonal decomposition with radial basis functions (RBF) networks applied to the grid node's displacement field.

Algorithm 5: Hyper-Reduced-PIMPLE algorithm with dynamic mesh

Input : $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$, ν_t^0, $\boldsymbol{\delta}_h^0$, $\boldsymbol{\Phi}$, $\boldsymbol{\Xi}$, $\boldsymbol{\Psi}$, $\boldsymbol{\mathcal{P}}_u$, and $\boldsymbol{\mathcal{P}}_p$;

while $t \leq t_{end}$ **do**

  while *No. outer corrections* ≥ 2 *and* $Tol \geq maxTol$ **do**

    Compute the forces; ▷ Using $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$;

    Solve the rigid body problem Eqs. (2.57) and (2.58) ▷ To obtain the new COG y_{new}^C;

    Compute $\boldsymbol{c} = RBF(y_{new}^C)$ Eq. (3.34);

    Reconstruct $\boldsymbol{\delta}^n = \Psi \boldsymbol{c}^T$ Eq. (3.35);

    $\mathbf{A}_u \boldsymbol{u}_h^{n*} = \mathbf{b}_u$ ▷ Assembling the momentum matrix Eq. (2.41);

    Solve $\boldsymbol{Q}_u^T[\boldsymbol{\mathcal{P}}_u^T \mathbf{A}_u \boldsymbol{\mathcal{P}}_u]\boldsymbol{Q}_u \boldsymbol{a}^* = \boldsymbol{Q}_u^T[\boldsymbol{\mathcal{P}}_u^T \mathbf{b}_u]$ ▷ $\boldsymbol{Q}_u = \boldsymbol{\mathcal{P}}_u^T \boldsymbol{\Phi}$;

    $\boldsymbol{u}_{h,M}^{n*} = \boldsymbol{Q}_u \boldsymbol{a}^*$ ▷ Reconstruct on the M Lagrangian points;

    Interpolate $\boldsymbol{u}_h^{n*}$ ▷ Using $\boldsymbol{u}_{h,M}^{n*}$ and Eq. (3.54) ;

    $[\nabla(\cdot)]\,(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}') = [\nabla(\cdot)]\boldsymbol{u}_h^{n*}$ ▷ Assembling the matrix of PPE Eq. (2.50);

    Solve $\boldsymbol{Q}_p^T[\boldsymbol{\mathcal{P}}_p^T \mathbf{A}_p \boldsymbol{\mathcal{P}}_p]\boldsymbol{Q}_p^T \boldsymbol{b}' = \boldsymbol{Q}_p^T[\boldsymbol{\mathcal{P}}_p^T \boldsymbol{b}_p]$ ▷ $\boldsymbol{Q}_p = \boldsymbol{\mathcal{P}}_p^T \boldsymbol{\Xi}$;

    $\boldsymbol{p}'_{h,M} = \boldsymbol{Q}_p \boldsymbol{b}'$ ▷ Reconstruct on the M Lagrangian points;

    Interpolate $\boldsymbol{p}'_h$ ▷ Using $\boldsymbol{p}'_{h,M}$ and Eq. (3.54) ;

    $\boldsymbol{u}'_h \leftarrow -\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'_h$ ▷ Momentum corrector Eq. (2.53) ;

    while *No. inner corrections* **do**

      $[\nabla(\cdot)]\,(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}''_h) = [\nabla(\cdot)]\tilde{\boldsymbol{u}}'$ ▷ Assembling the matrix for PPE Eq. (2.55);

      Solve $\boldsymbol{Q}_p^T[\boldsymbol{\mathcal{P}}_p^T \mathbf{A}_p \boldsymbol{\mathcal{P}}_p]\boldsymbol{Q}_p^T \boldsymbol{b}'' = \boldsymbol{Q}_p^T[\boldsymbol{\mathcal{P}}_p^T \boldsymbol{b}_p]$ ▷ $\boldsymbol{Q}_p = \boldsymbol{\mathcal{P}}_p^T \boldsymbol{\Xi}$;

      $\boldsymbol{p}''_{h,M} = \boldsymbol{Q}_p \boldsymbol{b}''$ ▷ Reconstruct on the M Lagrangian points;

      Interpolate $\boldsymbol{p}''_h$ ▷ Using $\boldsymbol{p}''_{h,M}$ and Eq. (3.54) ;

      $\boldsymbol{u}''_h \leftarrow \tilde{\boldsymbol{u}}' - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}''_h$: ▷ Momentum corrector Eq. (2.54) ;

    Evaluate the networks using NNs or LSTM ;

    Reconstruct the new turbulent viscosity ν_t^n;

    $\boldsymbol{u}_h^{n*} \leftarrow \boldsymbol{u}''_h$;

    $\boldsymbol{p}_h^{n-1} \leftarrow \boldsymbol{p}_h^{n-1} + \boldsymbol{p}'_h$;

Output: $\boldsymbol{u}_h^n$, $\boldsymbol{p}_h^n$, ν_t^n, and $\boldsymbol{\delta}_h^n$;

So, the first step of the mesh deformation reduction strategy is that of computing the POD modes of the grid nodes displacement field. To this end, we assemble a snapshot matrix with the grid nodes displacements obtained at different time steps

$$\boldsymbol{S}^g = [\boldsymbol{d}^g(\boldsymbol{x}, t_1), \ldots, \boldsymbol{d}^g(\boldsymbol{x}, t_N)]\,. \tag{3.28}$$

As in the case of pressure and velocity unknowns, the matrix $\boldsymbol{S}^g$ is then processed to obtain a correlation matrix using Eq. (3.8) and, from the solution of an eigenvalue problem as in Eq. (3.7), a set of POD modes. The reduced-order solution for the grid displacement field is then represented as,

$$\boldsymbol{\delta}_h(\boldsymbol{x}, t) \approx \sum_{i=1}^{N_r^d} c_i(t)\boldsymbol{\psi}_i(\boldsymbol{x}), \tag{3.29}$$

where N_r^d is the amount of modes considered for the grid displacement field. Along with the modal functions $\boldsymbol{\chi}_i(\boldsymbol{x})$, the solution of eigenvalue Eq. (3.7) provides the values of the modal coefficients in correspondence with each time step included in the snapshot matrix. For such a reason, a natural choice for computing the grid deformation at time instants not included in the snapshots would be that of interpolating the modal coefficients based on the time variable. However, given the fact that the grid nodes displacement is induced by the — rigid — translation of the cylinder boundary y^C, a more meaningful way to obtain the modal coefficients at each time step is to consider that

$$c_i(t) = \widehat{c}_i(y^C(t)) \quad i = 1, \ldots, N_r^d, \tag{3.30}$$

and interpolate the c_i values based on the cylinder vertical displacement variable — as obtained at each of the time steps at which the solution snapshots have been collected.

In this work, the interpolation step of the data driven POD strategy used for the reduction of the grid nodes displacement field is carried out by means of the Radial Basis Function [86] method. In the present framework, modal coefficient c_i at a generic value y^C is obtained evaluating the expression

$$c_i(y^C) = \sum_{k=1}^{N} w_k \rho(||y^C - y_k^C||), \tag{3.31}$$

in which $\rho : \mathbb{R} \to \mathbb{R}$ is in the so-called *radial basis*. ρ is a function of the Euclidean distance. In the present case it is a one dimensional function, but in more general cases it maps the m dimensional parameter space in $\mathbb{R}$. The weights w_k appearing in Eq. (3.31) are determined imposing the interpolation condition at the snapshots, in which the modal coefficients are known from Eq. (3.7). The conditions used are then

$$(c_i)_j = c_i(y_j^C) = \sum_{k=1}^{N} w_k \rho(||y_j^C - y_k^C||) \quad j = 1, \ldots, N \tag{3.32}$$

resulting in the system

$$\boldsymbol{c} = \mathbf{G}\boldsymbol{w}^T, \tag{3.33}$$

in which $\mathbf{G} = (g_{kj}) = \rho(||y_j^C - y_k^C||)$ is the Gram matrix. So, once the weights have been computed solving system (3.33) in the offline phase, in the online computations the modal coefficients are obtained evaluating Eq. (3.31).

We point out that, despite in the present case the c_i coefficients only depend on a scalar variable, in the case of multidimensional dependence, RBF interpolation can be used with no algorithmic modifications. Thus, more complex cases in which the cylinder exhibits rigid motions with more translational and rotational degrees of freedoms, could still be treated with the methodology described. Even in the case in which the FSI problem involves a deformable body which alters the shape of one or more boundaries of the fluid domain, RBF could be still used to interpolate the fluid mesh nodal displacements based on the structural displacements reduced coefficients.

The new coefficient at the online stage is computed by

$$c(y_{new}^C) = \sum_{j=k}^{N} w_k \rho(||y_{new}^C - y_k^C||). \tag{3.34}$$

The new point displacement is predicted as follows

$$\boldsymbol{d}^g(\boldsymbol{x}, t_{new}) = \sum_{i=1}^{N} c(y_{new}^C)\boldsymbol{\psi}_i(\boldsymbol{x}) = \Psi \boldsymbol{c}. \tag{3.35}$$

Finally, we remark that the RBF interpolation can also be interpreted as a network in which N is the number of neurons in the hidden layer, y_k^C is the centre vector for neuron k, $\boldsymbol{w}_k$ being the weight of neuron k in the linear output neuron. Given this analogy, we point out that different and more efficient networks can substitute RBF — which has a $O(N^2)$ computational cost — in future works.

3.4 Machine learning for eddy viscosity prediction

In this part, a deep neural network is used for the prediction of the eddy viscosity, mimicking the works done in [43, 66, 183]. In [66] radial basis functions has been used to predict the temporal coefficients of the eddy viscosity based on the temporal coefficients of the velocity. The study carried out in [183] used a fully connected neural network to predict the parameterized coefficients of the eddy viscosity in a steady simulation. Scholars in [43] studied the effects of the effective viscosity in projection-based ROM simulations and employed a simple spline interpolation for the prediction of the first dominant mode of the temporal coefficient of the eddy viscosity from known values.

In the same spirit, a fully connected neural network and a recurrent neural network based on the LSTM are presented here to predict temporal coefficients of the eddy viscosity. The choice of the aforementioned methodologies relies on the

idea of building a reduced problem independent of the turbulent technique ($k-\epsilon$, $k-\omega$, etc.) that might be used at the offline stage to evaluate the eddy viscosity. The first method uses the physical relation between the velocity field and eddy viscosity, thanks to the Boussinesq hypothesis in Eq. (2.8). The second method assumes a one-to-one dynamical mapping between the low-dimensional states $\boldsymbol{n}^i$ and $\boldsymbol{n}^{i-1}$ defined in Eq. (3.36). The POD is applied to the snapshot matrix $\boldsymbol{S}_{\nu_t} \in \mathbb{R}^{N_h \times N_s}$ and the low-dimensional eddy viscosity is approximated using the POD as described in Section 3.2.

$$\nu_t(\boldsymbol{x}, \cdot) \approx \sum_{i=0}^{N_{\nu_t}} n_i(\cdot)\boldsymbol{\psi}_i(\boldsymbol{x}) = \boldsymbol{\Psi}\boldsymbol{n}^T. \tag{3.36}$$

$\boldsymbol{\psi}_i(\boldsymbol{x})$ and $n_i(\cdot)$ are the POD spatial and temporal coefficients modes for the eddy viscosity, respectively. $N_{\nu_t} \ll N_s$ denotes the selected number of modes to predict the eddy viscosity. In contrast to the temporal coefficients modes of the velocity, and pressure obtained by projecting the FOM onto the respective POD spatial modes and subsequently solving the reduced problem as it is done in SubSection 3.3.1, the predicted temporal coefficients modes for the eddy viscosity are modelled via machine learning technique such as a multi-layer feed-forward neural network or recurrent network.

3.4.1 POD-NNs

Within this method, the neural network is fed with temporal coefficients of the velocity field $\boldsymbol{a}$ and mapped to the predicted temporal coefficient of the turbulent viscosity, denoted $\tilde{\boldsymbol{n}}$. For a comprehensive description on the theory behind the Neural Networks (NNs), the reader is redirected to Goodfellow et al. [64]. In a feed-forward neural network, the *output* of a single layer of a given input $\mathbf{A} \in \mathbb{R}^N$ is given by $\mathbf{Y} = f(\mathbf{W}\mathbf{A} + \mathbf{b})$. $\mathbf{W} \in \mathbb{R}^{M \times N}$ being the weight matrix, $\mathbf{b} \in \mathbb{R}^M$ is a bias term, and f (.) is a *nonlinear function* that acts element-wise on its inputs.

The Multi-layer neural networks are generated by feeding the output $\mathbf{h}_l = f_{l+1}(\mathbf{W}_l\mathbf{A}_l + \mathbf{b}_l)$ of a layer l as the input of the next layer. The general steps of a multi-layer neural network are summarized in Algorithm 6. The vector $\mathbf{h}_l$ is often referred to as the hidden state or feature vector at the $l-th$ layer. Generally, training a network involves finding the parameters $\boldsymbol{\theta} = \{\mathbf{W}_l, \mathbf{b}_l\}_{l=0}^{l=L-1}$ such that the expected loss $\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})$ between the output $\mathbf{Y}$ and the target value $\hat{\mathbf{Y}}$ is minimized, i.e.

$$\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta}}[\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})], \tag{3.37}$$

where $\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})$ is some measure of discrepancy between the predicted and target outputs given in some cases by,

$$\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y}) = \left[\frac{1}{n-1}\sum_{i=1}^{n}\frac{||\boldsymbol{Y}_i - \boldsymbol{g}(\boldsymbol{Y}_{i-1}; \boldsymbol{\theta})||_2^2}{||\boldsymbol{Y}_i||_2^2}\right], \tag{3.38}$$

for example in the case of RNNs. With $\boldsymbol{g}$ a one-to-one mapping given by $\hat{\boldsymbol{Y}}_i = \boldsymbol{g}(\boldsymbol{Y}_{i-1}; \boldsymbol{\theta})$. Figure 3.2 depicts the schematic of simple feed forward neural network

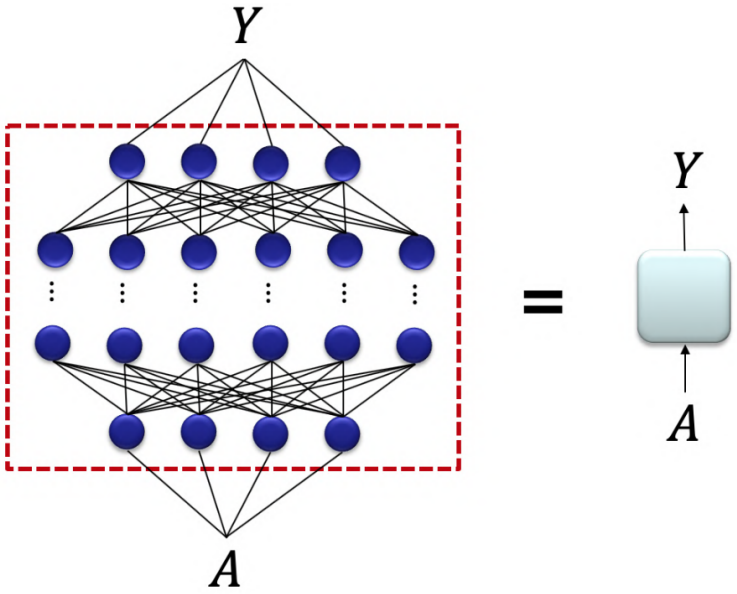


Figure 3.2: Schematic of a multi-layer neural network

with multiple layers. Such architectures are typically used in classification problems. In the following SubSection 3.4.2, we will discuss POD-RNNs based on the Long Short-Term Memory (LSTM).

Algorithm 6: Compute MLP output.

Input : $\boldsymbol{\chi}$;
set $\boldsymbol{\zeta}_0 \leftarrow \boldsymbol{\chi}$;
for $i \leftarrow 1$ *to* $l+1$ **do**
  $\boldsymbol{h}_i \leftarrow \boldsymbol{W}_i \zeta_{i-1} + \boldsymbol{b}_i$;
  $\zeta_i \leftarrow g_i(\boldsymbol{h}_i)$;
return $\boldsymbol{\zeta} \leftarrow \boldsymbol{\zeta}_{l+1}$;
Output: $\boldsymbol{\zeta} = f(\boldsymbol{\chi})$;

3.4.2 POD-RNNs

This methodology assumes the following relationships

$$\tilde{\boldsymbol{n}}^{i+1} = \boldsymbol{g}(\tilde{\boldsymbol{n}}^{i}). \tag{3.39}$$

$\boldsymbol{g}$ being an unknown mapping between the flow fields of adjacent time steps. The goal consists of learning a dynamical operator $\boldsymbol{g}$ by finding the parameters $\boldsymbol{\theta}$ (weights and biases) that allow to recurrently predict finite time series of the low-dimensional states. Recurrent Neural Networks (RNNs) are designed specifically to deal with sequential data. Data coming from transient CFD simulations or any nonlinear dynamical system are a set of time series that is sequential. This makes the application of RNNs to non-linear dynamical systems quite relevant [25].

RNNs differ from the traditional feed-forward neural network with the presence of a recurrent connection. This particular recurrent connection is responsible for

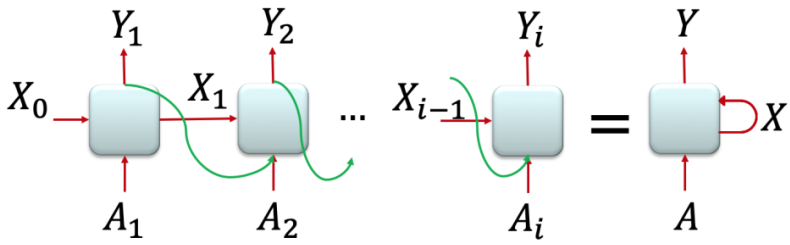


Figure 3.3: Schematic of recurrent neural network

storing the history of previous inputs $\boldsymbol{X}_i$ via hidden states. The basic mathematical structure of a simple RNNs cell for an input $\mathbf{A}_i \in \mathbb{R}^N$ and output $\mathbf{Y}_i \in \mathbb{R}^M$ given as follows

$$\mathbf{X}_i = f(\mathbf{W}_x \mathbf{X}_{i-1} + \mathbf{W}_a \mathbf{A}_i + \mathbf{b}) \quad \text{and} \quad \mathbf{Y}_i = g(\mathbf{W}_y \mathbf{X}_i). \tag{3.40}$$

$\mathbf{W}_x \in \mathbb{R}^{N_x \times N_x}$, $\mathbf{W}_a \in N_x \times N$, and $\mathbf{W}_y \in \mathbb{R}^{N_x \times M}$ being the hidden, input and output weight matrices respectively. $\mathbf{b} \in \mathbb{R}^M$ represents the bias term, and $\boldsymbol{X}_{i-1}$ the cell state at time $i-1$. The general steps of a RNN are summarized in Algorithm 7.

Figure 3.3 depicts the schematic of simple recurrent neural network. They differ from the traditional feed-forward neural network with the presence of a recurrent connection. This particular recurrent connection is responsible for the storing the history of previous inputs via hidden states. This work considers RNNs equipped with LSTM units [145] where the main steps are described in Algorithm 8. The straightforward prediction of $\tilde{\nu}_t$ at the online level is given using Eq. (3.36) i.e.

$$\tilde{\nu}_t = \Psi \tilde{\boldsymbol{n}}^T. \tag{3.41}$$

Algorithm 7: Compute the output sequence of an RNN

Input : Sequence $x_1, \cdots, x_p$;

$\boldsymbol{h}_0 \leftarrow 0$;

for $t \leftarrow 1$ *to* p **do**

  $\boldsymbol{h}_t \leftarrow g_h(\boldsymbol{W}_{hX} \boldsymbol{X}_t + \boldsymbol{W}_{hh} \boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$;

  $\boldsymbol{\zeta}_t \leftarrow g_\zeta(\boldsymbol{W}_{\zeta h} \boldsymbol{h}_t + \boldsymbol{b}_\zeta)$;

Output: Sequence $\zeta_1, \cdots, \zeta_p$

The RNNs are typically trained using stochastic gradient descent (SGD), or some variant, but the gradients are calculated using the backpropagation through time (BPTT) algorithm [173]. For a detailed discussion on BPTT the reader might refer to [12]. In BPTT, the RNN is first unrolled in time, stacking one copy of the RNN per time step. Training RNNs has long been considered to be challenging [12], especially when learning sequences with long-term dependencies as the gradients either vanish or explode. The vanishing or exploding gradient problem is typically addressed by using gated RNNs, including LSTM networks [145]. and networks based on the gated recurrent unit (GRU) [37]. These networks have additional paths through which gradients neither vanish nor explode, allowing gradients of the loss function to back-propagate across multiple time steps and thereby making the appropriate parameter updates.

Algorithm 8: General scheme of the LSTM algorithm

Input : Sequence $x_1, \cdots, x_p$;

$\boldsymbol{h}_0 \leftarrow 0$;

$\boldsymbol{C}_0 \leftarrow 0$;

for $t \leftarrow 1$ *to* p **do**

  $\boldsymbol{f}_t \leftarrow \sigma(\boldsymbol{W}_f[\boldsymbol{x}_t, \boldsymbol{\zeta}_{t-1}] + \boldsymbol{b}_f)$;

  $\boldsymbol{i}_t \leftarrow \sigma(\boldsymbol{W}_i[\boldsymbol{x}_t, \boldsymbol{\zeta}_{t-1}] + \boldsymbol{b}_i)$;

  $\tilde{\boldsymbol{C}}_t \leftarrow \tanh(\boldsymbol{W}_f[\boldsymbol{x}_t, \boldsymbol{\zeta}_{t-1}] + \boldsymbol{b}_f)$;

  $\boldsymbol{C}_t \leftarrow \boldsymbol{f}_t \otimes \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \otimes \tilde{\boldsymbol{C}}_t \quad \boldsymbol{o}_t \leftarrow \sigma(\boldsymbol{W}_o[\boldsymbol{x}_t, \boldsymbol{\zeta}_{t-1}] + \boldsymbol{b}_o)$;

  $\boldsymbol{\zeta}_t \leftarrow \boldsymbol{o}_t \otimes \tanh(\boldsymbol{C}_{t-1})$;

Output: Sequence $\zeta_1, \cdots, \zeta_p$

3.5 Efficient projection in moving domains problems

Following the derivation of [47], when considering an entrained and/or deformable computational domain $\Omega(t)$, of boundary $\partial\Omega(t)$ with the unit external normal $\boldsymbol{n}(t)$ and velocity $\boldsymbol{u}_g$ respective to an absolute reference domain frame, the following ALE integral form for the Navier-Stokes equations can be written as follows

$$\frac{\delta}{\delta t}\int_{\Omega_i} \boldsymbol{u} d\Omega_i = -\sum_{j\in S_i} [\boldsymbol{F}_c(\boldsymbol{u}, \boldsymbol{u}_g) + \boldsymbol{F}_d(\boldsymbol{u})] \cdot \boldsymbol{n} dS_j - \nabla p. \tag{3.42}$$

where $\partial\Omega_i = S_i$ represents the $i-th$ face of the cell Ω_i considered in the mesh. $\boldsymbol{F}_c$ and $\boldsymbol{F}_d$ are the convective and diffusive fluxes. Then, the average of the conservative variables $\boldsymbol{u}$ in the cell Ω_i can be defined as

$$\boldsymbol{u} = \frac{1}{\mathcal{V}(\Omega_i(t))}\int_{\Omega_i(t)} \boldsymbol{u}(\boldsymbol{x}, t) d\Omega_i \tag{3.43}$$

Where $\mathcal{V}(\Omega_i(t))$ is the time-dependent volume of the related cell. By substituting Eq. (3.43) in Eq. (3.42), the following relation holds

$$\frac{\delta \mathcal{V}(\Omega_i(t))\boldsymbol{u}}{\delta t} = \boldsymbol{f}(\boldsymbol{u}, t). \tag{3.44}$$

Finally, we obtain

$$\mathcal{V}(\Omega_i(t))\frac{\delta \boldsymbol{u}}{\delta t} = \boldsymbol{f}(\boldsymbol{u}, t) - \boldsymbol{u}\frac{\delta \mathcal{V}(\Omega_i(t))}{\delta t}. \tag{3.45}$$

An additional source term is present in the right-hand side of the FOM Eq. (3.45) to take into account the variation of the cell volumes. For a preliminary study, this last source term can be neglected so that the following formulation is obtained

$$\frac{\delta \boldsymbol{u}}{\delta t} = \frac{\boldsymbol{f}(\boldsymbol{u}, t)}{\mathcal{V}(\Omega_i(t))}. \tag{3.46}$$

The previous equation takes into account the time dependence of the mesh metric for the calculation of the flow balance. The projection in L^2 sense of Eq. (3.46) is given as follows,

$$\left(\boldsymbol{\Phi}^T, \frac{\delta \boldsymbol{u}}{\delta t}\right)_{L^2(\Omega_i(t)} = \left(\boldsymbol{\Phi}^T, \frac{\boldsymbol{f}(\boldsymbol{u}, t)}{\mathcal{V}(\Omega_i(t))}\right)_{L^2(\Omega_i(t)}. \tag{3.47}$$

This leads to

$$\boldsymbol{\Phi}^T \mathcal{V}(\Omega_i(t)) \dot{\boldsymbol{a}} = \boldsymbol{\Phi}^T \boldsymbol{f}(\boldsymbol{u}, t). \tag{3.48}$$

This form will have a practical advantage when constructing a surrogate model with hyper-reduction for future research. In Section 3.6, to construct an efficient hyper-reduction model Eq. (3.48) will be considered.

3.6 Hyper-Reduction: From projection to interpolation

The Empirical Lagrangian Interpolation Method (ELIM) was proposed in 2004 [11] as a way of identifying a good set of interpolation nodes on multidimensional unstructured meshes and has found numerous applications. For a very short list, the interested reader can refer to [1, 11, 31, 32, 51, 158, 164]. Recently, the concept of neural empirical interpolation method (NEIM) has been introduced in [67]. NEIM is a greedy algorithm which accomplishes the reduction by approximating an affine decomposition of the nonlinear term of the ROM, where the vector terms of the expansion are given by neural networks depending on the ROM solution, and the coefficients are given by an interpolation of some "optimal" coefficients. In the following, we discuss the concept of "projection to interpolation", and the ELIM algorithm.

3.6.1 From projection to interpolation

In Section 3.2, we discussed that the optimal linear representation in any weighted L^2 norm, given a basis (assuming for simplicity and conditioning that it is orthonormal) of cardinality, n is through an expression of the form

$$\boldsymbol{u}(\boldsymbol{x}, t) \approx \mathcal{P}_n[\boldsymbol{u}] = \sum_{i=1}^{n} a_i(t) \boldsymbol{\phi}_i(\boldsymbol{x}) \tag{3.49}$$

where a_i and $\boldsymbol{\phi}_i$ satisfy Eq. (3.5) and properties mentioned in Section 3.2.

The approximation through projection onto a reduced basis, Eq. (3.49), is replaced by interpolation (in the physical dimension(s), $\boldsymbol{x}$) as follows. First, a basis is chosen, for example through a POD or greedy approach. The interpolant then is sought to have the form:

$$\mathcal{I}_n[\boldsymbol{u}](\boldsymbol{x}, t) = \sum_{i=1}^{n} \alpha_i(t) \boldsymbol{\phi}_i(\boldsymbol{x}) \tag{3.50}$$

where α_i are defined to be solutions of the interpolation problem; namely that the interpolant exactly agrees with the function at the Lagrangian nodes (the construction of which we discuss below) as follows

$$\mathcal{I}_n[\boldsymbol{u}](\boldsymbol{X}_i, t) = \boldsymbol{u}(\boldsymbol{X}_i, t), \quad \forall \quad i = 1, \cdots, n. \tag{3.51}$$

For the moment, we shall assume that the Lagrangian nodes $\boldsymbol{X}_i$ are known and proceed to describe how to use them to find the Lagrangian interpolant. This can be somewhat misleading, since it is not the way it works in practice, which is: the first Lagrangian node is found, its associated interpolant built, the second Lagrangian node is found, the interpolant enriched to take it into account, and so on. They are not disjoint processes, unlike standard spectral methods, where all the Gaussian nodes are found and afterwards the interpolant built. This is not only a procedural difference, but highlights a big difference of the Empiral Lagrangian Interpolation Method (ELIM): namely that the approach (nodes and interpolant) is *hierarchical*. Hopefully this gradual presentation is intuitive and pedagogical, later in this section we will present the full, coupled algorithm together with its numerical intricacies.

Solving Eq. (3.51) is equivalent to solving the n-by-n system:

$$\sum_{i=1}^{n} \boldsymbol{V}_{ji}\alpha_i(t) = \boldsymbol{u}(\boldsymbol{X}_j, t), \tag{3.52}$$

where the interpolation matrix $\boldsymbol{V}_{ji} = \{\boldsymbol{\phi}_i(\boldsymbol{X}_j)\}$ is a generalization of the Vandermonde matrix. It is worth mentioning that when polynomial basis is used, the Vandermonde matrix can easily be very *ill-conditioned* if the nodes are chosen, for example, equally spaced, not to mention if they are scattered. So one can anticipate that the solution to the problem Eq. (3.50) is already non-trivial and one of the goals of the ELIM is to make sure that it does not lead to an ill conditioned problem, as well as providing a high accuracy interpolant.

The choice of empirical nodes given by the ELIM together with the linear independence of the reduced basis ensures that $\boldsymbol{V}$ is invertible, so that:

$$\alpha_i = \sum_{j=1}^{n} \boldsymbol{V}_{ij}^{-1}\boldsymbol{u}(\boldsymbol{X}_j, t). \tag{3.53}$$

is the unique solution to Eq. (3.52). It then follows upon substituting Eq. (3.53) into Eq. (3.50) that the empirical interpolant is

$$\mathcal{I}_n[\boldsymbol{u}](\boldsymbol{x}, t) = \sum_{j=1}^{n} \boldsymbol{B}_j(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{X}_j, t). \tag{3.54}$$

where

$$\boldsymbol{B}_j(\boldsymbol{x}) = \sum_{j=1}^{n} \boldsymbol{\phi}_j(\boldsymbol{x})\boldsymbol{V}_{ij}^{-1} \tag{3.55}$$

is independent of t. Note that the coefficients $\{\boldsymbol{B}_j\}_{j=1}^n$ satisfy $\boldsymbol{B}_j(\boldsymbol{X}_i) = \delta_{ij}$ and are built directly from the reduced basis. Next we present the general algorithm how to

compute the ELI nodes in a rather qualitative way.

3.6.2 Empirical Lagrangian Interpolation algorithm

We present the Empirical Lagrangian Interpolation (ELI) algorithm as introduced in [100]. The ELI algorithm is remarkably simple. It relies on a greedy selection process, both in the choice of basis functions and interpolation points. In a greedy algorithm, the next step is determined by some local optimality criterion based on the current situation. This may in some cases lead to a globally optimal algorithm, but in most cases, as for the ELI method, this is not generally true. For a more thorough explanation of the concept of greedy algorithms, see e.g. [40]. We now present the ELI algorithm, given as Algorithm 9.

Algorithm 9: Empirical Lagrangian Interpolation Method

Input : Basis or set $\mathcal{U} = \{\boldsymbol{u}_j\}_{j=1}^{n}$;

1 $\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}\in\mathcal{U}} ||\boldsymbol{u}(.)||_{L^\infty(\Omega)}$;

2 $x_1 = \arg\max_{\boldsymbol{x}\in\bar{\Omega}} |\boldsymbol{u}_1(\cdot)|$;

3 $\boldsymbol{q}_1 = \boldsymbol{u}_1(\cdot)/\boldsymbol{u}_1(\boldsymbol{x}_1)$;

4 $\mathbf{B}_{11}^1 = \boldsymbol{q}_1(\boldsymbol{X}_1)$;

5 **for** $M \leftarrow 2 \quad to \quad M_{max}$ **do**

6 $\sum_{j=1}^{M-1} \boldsymbol{q}_j(\boldsymbol{x}_i)\alpha_{M-1,j}[\boldsymbol{u}] = \boldsymbol{u}(\boldsymbol{x}_i), \quad i = 1, \cdots, M-1$ ▷ Solve the interpolation problem;

7 Compute the interpolant $\mathcal{I}_{M-1}[\boldsymbol{u}(\cdot)] = \sum_{j=1}^{M-1} \alpha_{M-1,j}[\boldsymbol{u}]\boldsymbol{q}_j(\cdot)$;

8 $\epsilon_{M-1}(\boldsymbol{u}) = ||\boldsymbol{u} - \mathcal{I}_{M-1}[\boldsymbol{u}]||_{L^\infty(\Omega)}$;

9 $\boldsymbol{u}_M = \arg\max_{\boldsymbol{u}\in\mathcal{U}} ||\epsilon_{M-1}(\boldsymbol{u})||_{L^\infty(\Omega)}$;

10 $x_M = \arg\max_{\boldsymbol{x}\in\bar{\Omega}} |\boldsymbol{u}_M(\boldsymbol{x}) - \mathcal{I}_{M-1}[\boldsymbol{u}_M](\boldsymbol{x})|$;

11 $\boldsymbol{r}_M(\cdot) = \boldsymbol{u}_M(\cdot) - \mathcal{I}_{M-1}[\boldsymbol{u}_M](\cdot)$;

12 $\boldsymbol{q}_M(\cdot) = \boldsymbol{r}_M(\cdot)/\boldsymbol{r}_M(x_M)$;

13 $\mathbf{B}_{ij}^M = \boldsymbol{q}_j(x_i), \quad 1 \leq i, j \leq M$;

Output: ELI nodes $\{x_j\}_{j=1}^{M}$ and the interpolant $\mathcal{I}_M$

Note that if for some reason, the set of functions $\mathcal{U}$ were to be given, all linearly independent, then the procedure of finding the interpolation points through the above process is also well-defined and leads to a set of interpolation points that have similar properties as above. It is worth mentioning that, instead of choosing the interpolation node as is done in step 10 in Algorithm 9, it is possible to choose the interpolation node that fully minimize the condition number of the interpolation-matrix and/ or Lebesgue constant of the interpolant. For more details, the reader can refer to [163].

The rationale for the greedy approach is that it allows us to get a better sense

of the interpolation properties since

$$||\boldsymbol{u} - \mathcal{I}_n[\boldsymbol{u}]||_{L^\infty(\Omega)} \leq ||\boldsymbol{u}_{n+1} - \mathcal{I}_n[\boldsymbol{u}_{n+1}]||_{L^\infty(\Omega)} = \epsilon_n(\boldsymbol{X}_{n+1}). \tag{3.56}$$

and this last quantity is one of the outputs of the construction process. How we may limit the required number of interpolation nodes through the use of a posteriori error estimates is discussed in SubSection 3.6.3.

3.6.3 Error analysis

In this section, we make use of the Lebesgue constant, defined as

$$\Lambda_n = \sup_{\boldsymbol{x}\in\Omega} \sum_{j=1}^{n} |\mathcal{L}_j(\boldsymbol{x})|. \tag{3.57}$$

For the Empirical Lagrangian Interpolation (ELI) method, an upper bound for Λ_n is given by $2^n - 1$ [11]. This is however a very pessimistic estimate, and in practice the observed behaviour is usually far better. A classical result in approximation theory, also known as Lebesgue's lemma [11], gives the following bound for the interpolation error.

Lemma 3.6.1. *Assume X is a Banach space, and $X_n \subset X$, $dim(X_n) = n$. For any $\boldsymbol{u} \in X$ the interpolation error satisfies:*

$$||\boldsymbol{u} - \mathcal{I}_n[\boldsymbol{u}]||_X \leq (1 + \Lambda_n)||\boldsymbol{u} - \mathcal{P}_n[\boldsymbol{u}]||_X. \tag{3.58}$$

Here, the projection error $||\boldsymbol{u} - \mathcal{P}_n[\boldsymbol{u}]||_X$ is the best possible approximation of $\boldsymbol{u}$ in the approximation space X_n. For the ELI method, Lemma 3.6.1 can be made considerably more precise if a few conditions are fulfilled. It can in fact be shown that the upper bound for the interpolation error from the greedy algorithm is given by

Theorem 3.6.1. *Assume $\mathcal{U} \subset X \subset L^\infty(\Omega)$ and the existence of a (possibly unknown) sequence of finite dimensional spaces*

$$\mathcal{Z}_1 \subset \mathcal{Z}_2 \subset \cdots \subset \mathcal{Z}_n \subset span(\mathcal{U}) \quad dim(X_n) = n, \tag{3.59}$$

such that there exists $c > 0$ and $\alpha > \log(4)$ with,

$$\forall \boldsymbol{u} \in \mathcal{U}, ||\boldsymbol{u} - \mathcal{P}_n[\boldsymbol{u}]||_X \leq ce^{-\alpha n}. \tag{3.60}$$

Then

$$||\boldsymbol{u} - \mathcal{I}_n[\boldsymbol{u}]||_{L^\infty(\Omega)} \leq ce^{-(\alpha - \log(4))n}. \tag{3.61}$$

Proof. see [100]. □

This theorem has some immediate implications. First, if there exists a finite dimensional space allowing for an exponential approximation, the ELI method will

achieve an exponential rate of convergence. Also, if the spaces $\mathcal{Z}_i$ are not predetermined, the greedy algorithm provides us with such a sequence through the ELI space X_n. A third and final advantage of the greedy algorithm is that it allows easy access to an a posteriori error estimate, as we know that the argmax definition ϵ_{n+1} in Algorithm 9 ensures that

$$||\boldsymbol{u} - \mathcal{I}_n[\boldsymbol{u}]||_{L^\infty(\Omega)} \leq ||\boldsymbol{u}_{n+1} - \mathcal{I}_n[\boldsymbol{u}_{n+1}]||_{L^\infty(\Omega)} = \epsilon_n(\boldsymbol{X}_{n+1}). \tag{3.62}$$

This implies that we always know the maximum error for the previous interpolation space. This can be used to end Algorithm 9 at a predefined tolerance level and thus avoid computing all n_{max} stages.

3.7 Summary

This chapter gave a general tool for model-order reduction used in this thesis. Here, six sections have been discussed. Each contributing to a comprehensive understanding of the core of this thesis, which was to build a surrogate model based on the POD for segregated solvers with moving domains. In Section 3.2, the POD is presented for time-dependent problems for moving domains for incompressible flows. However, it is worth mentioning that ROM can be suitable also compressible flows with moving domains or boundaries. Moving forward, Section 3.3 addressed the reduced algorithm based on the PIMPLE algorithm, which combined the projection-based on the governing equations and data-driven techniques. After that, in Section 3.3.4 and Section 3.4 the POD has been combined with radial basis functions and machine learning algorithms respectively to predict the grid displacement and eddy viscosity field at the online stage. In Section 3.5, efficient projection in moving domains problems is presented to circumvent the expensive projection on the linear systems coming from the discretized governing equations. Lastly, in Section 3.6, hyper-reduction is introduced. The hyper-reduction is based on the ELIM for choosing the interpolation nodes in the mesh domain.

Chapter 4

Applications and Numerical Results

Contents

4.1 Physical problems of interest . . . 45
4.2 Unsteady Laminar Case . . . 46
4.2.1 Description of the configuration and boundary conditions . . . 46
4.2.2 Description of the configuration and boundary conditions . . . 46
4.2.2.1 Linear solvers for the fluid . . . 47
4.2.2.2 Structural solver . . . 48
4.2.3 Results and discussion . . . 48
4.2.3.1 Computational cost . . . 48
4.2.3.2 Reconstruction error . . . 49
4.2.3.3 ROM solution error . . . 50
4.3 Unsteady Turbulent Case . . . 56
4.3.1 Definition of test case, simulation results, and discussion . . . 56
4.3.1.1 Definition of the test case . . . 56
4.3.1.2 Simulation results . . . 58
4.3.1.3 Prediction quality . . . 58
4.3.1.4 Machine learning of the temporal eddy viscosity coefficients . . . 59
4.3.1.5 ROM online resolution time . . . 60
4.3.1.6 ROM online resolution quality . . . 61
4.4 Empirical Lagrangian Interpolation . . . 69
4.5 Summary . . . 70

4.1 Physical problems of interest

There are numerous physical problems in an engineering system. Common physical problems solved using the standard FVM include: heat transfer, acoustics, fluid mechanics. In this section, we will present most applications on fluid mechanics, especially in fluid dynamics. The first physical problem of interest is the two-dimensional flow passing a translating circular cylinder in a laminar regime. As in industrial and

real-world applications, many fluid dynamics problems involve turbulent flows interacting with moving boundaries, the second problem of interest is a two-dimensional problem of the flow passing an airfoil section which is able to rotate and translate in the turbulence regime. The second test case is a classical benchmark in the field of aerospace engineering. Lastly, the third case is a 2D Burger's equation on a backward-facing step. The 2D Burgers' equation on a backward-facing step is a significant test case for validating hyper-reduction techniques. By efficiently solving this problem, one can gain insights into more complex fluid-structure interaction problems. The hyper-reduction methods ensure that the essential dynamics of the flow are captured with reduced computational effort, making it feasible to solve large-scale problems in practical engineering applications.

4.2 Unsteady Laminar Case

This result section, in full, is a reprint of: *A reduced-order model for segregated fluid-structure interaction solvers based on an ALE approach.* **In revision : submitted** 2023.

4.2.1 Description of the configuration and boundary conditions

The benchmark considered is an elastically mounted cylinder restrained to move transversely as shown in Figure 4.1.

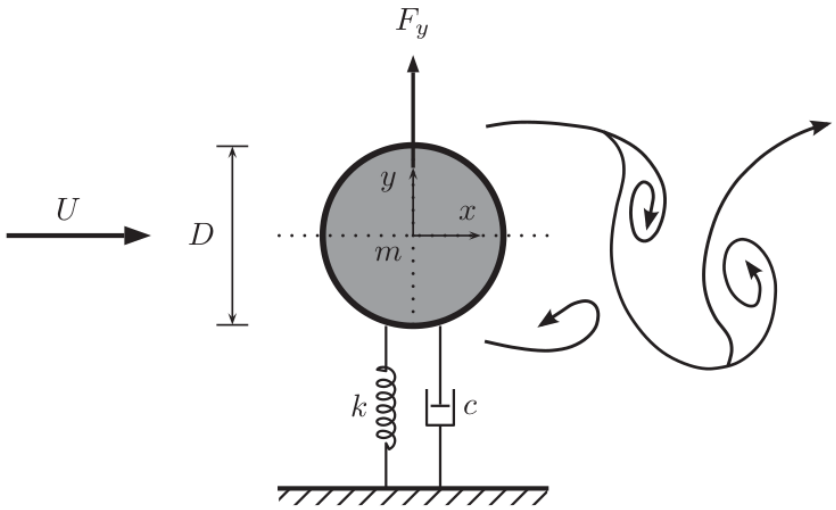


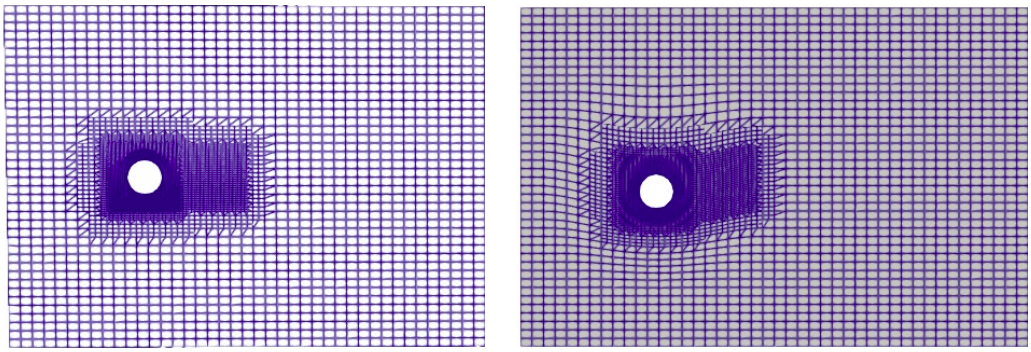
Figure 4.1: Cross flow vortex-induced vibrations

4.2.2 Description of the configuration and boundary conditions

The computational domain has a length of 34 D and a width of $10D$, where $D = 1.0\,\mathrm{m}$ is the cylinder diameter. The cylinder is located at a $5D$ distance from the inlet. Figure 4.2 presents a view of the two-dimensional computational grid,

	Inlet	Sides	Outlet	Cylinder
$\boldsymbol{u}$	$\boldsymbol{u} = (1, 0)$	$\boldsymbol{u} \cdot \boldsymbol{n} = 0$	$\nabla \boldsymbol{u} \cdot \boldsymbol{n} = 0$	$\boldsymbol{u} = (0, \dot{y}^C)^*$
p	$\nabla p \cdot \boldsymbol{n} = 0$	$\nabla p \cdot \boldsymbol{n} = 0$	p = 0	p = 0
$\boldsymbol{d}^g$	$\boldsymbol{d}^g = \boldsymbol{0}$	$\boldsymbol{d}^g = \boldsymbol{0}$	$\boldsymbol{d}^g = \boldsymbol{0}$	$\boldsymbol{d}^g = (0, y^C)^*$

Table 4.1: A summary of the boundary conditions imposed in the ALE fluid dynamic problem. Note that the * subscript indicates quantities that are computed by the rigid body structural solver.



(a) (b)

Figure 4.2: The mesh used in the simulations: (a) the initial mesh, (b) the deformed mesh in correspondence with a cylinder displacement of approximately 40% of the diameter.

both in its reference/initial configuration, and in a deformed state caused by the $0.4D$ vertical displacement of the cylinder. The grid features 11 644 cells (control volumes) and 24 440 points. The flow velocity at the inlet is $\boldsymbol{U}_\infty = (U_{in}, 0)$ with $U_{in} = 1.0\,\mathrm{m\,s^{-1}}$, and the physical viscosity $\nu = 0.005\,\mathrm{kg/ms}$. This corresponds to a Reynolds number of 200. Although the transition to turbulent flow occurs between 150 and 200, so we use a laminar solver anyway. As summarized in Table 4.1, at the inlet boundary non-homogeneous Dirichlet and zero gradient conditions are prescribed for the velocity and the pressure fields respectively. At the outlet boundary, zero gradient and homogeneous Dirichlet conditions are prescribed for velocity, and pressure respectively. On the sides (top and bottom) zero gradient conditions are prescribed for both velocity and pressure respectively. On the cylinder, we apply the structural solver interface coupling conditions described in Section 2.6.

4.2.2.1 Linear solvers for the fluid

The simulations are carried out using the PIMPLE Algorithm 2. The PIMPLE algorithm has the capacity to adapt the time steps in a way that assures the maximum Courant–Friedrichs–Lewy (CFL) does not exceed a prescribed value of 0.5 in this simulation. The implicit Euler scheme is used for the computation of the time derivative of the velocity field. For the spatial gradients, a Gauss linear scheme has been employed. The convective term has been approximated with the Upwind scheme. Gauss linear scheme is used to approximate the diffusive term. The values of the relaxation factors α_u, and α_p have been fixed at 0.7 and 0.3, respectively. One non-orthogonal corrector iteration is used to deal with the mesh's non-orthogonality. In addition, one pressure corrector and two momentum correctors are used in the

Re	f_n [Hz]	c [kg/s]	k[N/m]	m [kg]
200	0.185	0.01	6.76e-2	0.05

Table 4.2: Simulation parameters

simulations. As for the linear solvers, a smoother Gauss-Seidel has been used for solving the momentum equation, and GAMG (geometric-algebraic multi-grid) for solving the pressure equation.

4.2.2.2 Structural solver

As mentioned, the structural model is represented by the second-order differential Eq. (2.56) for rigid-body motion, here solved using the Symplectic 2nd-order explicit time-integrator. The mass of the cylinder considered in the numerical tests is $m = 0.05\,\mathrm{kg}$, the spring stiffness is $m = 6.76 \times 10^{-2}\,\mathrm{N\,m^{-1}}$, which results in the natural frequency $f_n = 0.185\,\mathrm{Hz}$. The cylinder to ground connection damping coefficient is $c = 0.01\,\mathrm{kg\,s^{-1}}$. The flow and structure parameters are summarized in Table 4.2.

4.2.3 Results and discussion

The main objective of the present numerical test is that of evaluating the reduced-order model (ROM) ability to predict the flow fields corresponding to the final periodic regime solution. The full-order model solver used in this simulation campaign to collect snapshots is the Finite Volume Method (FVM) C++ open source library OpenFOAM [76]. At the reduced-order level, modal reduction, as well as the assembling and resolution of the reduced-order systems are carried out using the C++-based open source library ITHACA-FV (In real Time Highly Advanced Computational Applications for Finite Volumes) [155, 157]. ITHACA-FV has been developed to be interfaced to the Finite Volume solvers featured in OpenFOAM. The latter FVM library is in fact widely used in industrial applications. For such a reason, interfacing the present ROM implementation with OpenFOAM data structures makes the methods developed readily applicable for real world problems. Finally, we point out that the C++ library SPLINTER [61] has been used in this work to build the RBF networks.

In the framework of the current cross flow cylinder test case, the full-order model (FOM) simulation was run for enough time to reach a periodic regime solution. After this, it was relaunched for 30 additional seconds with a constant simulation time step of 0.001 s exporting the solution fields every 0.1 s.

4.2.3.1 Computational cost

Table 4.6 reports a comparison analysis of the full-order and reduced-order models execution times as the number of modes for the prediction of velocity, pressure, and grid nodes displacement fields are varied. This allows for evaluating the effect of the number of modes variation on the computational cost of the online phase.

The offline stage comprises four steps: the computation of the snapshot (computed by a numerical approximation of the original high-dimensional system), computation of the POD basis, projection of the dynamics on the low-rank subspace, and the radial basis networks evaluation. But only the computational cost of the first

Stages	*# of modes*	Time [s]
Offline PDE solution	-	4.0567e+03
Online PDE solution	$N_u = N_p = 30,\ N_{pD} = 1$	4.211267616e+03
	$N_u = N_p = 20, N_{pD} = 1$	3.39621e+03
	$N_u = 15, N_p = 10,\ N_{pD} = 1$	2.90248e+03
	$N_u = 20, N_p = 10,\ N_{pD} = 1$	3.28614e+03

Table 4.3: Offline and Online times comparison varying the number of modes

step is reported in Table 4.3 as it is the most expensive one. The online cost is the computational time needed to compute the solutions of the surrogate model. The computational times in Table 4.3, suggest that the ROM solution only allows for a modest speed-up with respect to the FOM solver. Moreover, the speed-up obtained by the online solution of the reduced system is not proportional to the reduction of the unknowns obtained at the reduced-order level. This is because, in the presence of a deforming domain such as the one characterizing our FSI simulations, the entries of the matrices of the ROM system must be computed at each time step through integrals on the updated full-order grid. This at the moment represents a major bottleneck towards a ROM that grants significant computational cost reduction with respect to its FOM counterpart, and work is being carried out — implementing hyper-reduction techniques — towards lowering the computational cost associated with the reduced model assembling. Nonetheless, the main goal of the present work is that of assessing the accuracy of the ROM approach taken. In particular, it is important to establish whether the interaction between the physics-based reduction of the fluid dynamic balance equations, and the data-driven reduction of the fluid dynamic fields and grid displacement motion, results in an accurate solver.

4.2.3.2 Reconstruction error

Fig. 4.14 shows both the decay of the cumulative eigenvalues and the Relative Information Content (RIC) corresponding to the three correlation matrices of the fields of interest — $\boldsymbol{u}, p, \boldsymbol{d}^g$. The RIC is a simple quantitative metric to understand the Kolmogorov width of a given system [3]. The Kolmogorov width provides a measure of the system's reducibility. In the POD context, it can be considered a measure of how well a linear superposition of POD modes might represent the underlying dynamics. The following RIC formula by Eq. (4.1) is used to compute the percentage's modal energy:

$$RIC(M) = \left(\sum_{i=1}^{M} \lambda_i / \sum_{i=1}^{N_s} \lambda_i \right) \times 100, \tag{4.1}$$

where M is the number of POD modes used, and N_s is the total number of modes computed. RIC can then be seen as the amount of the overall system energy retained by the first M POD modes.

Fig. 4.3 displays an extremely fast decay of the grid node displacement ($\boldsymbol{d}^g$) eigenvalues. This fast decay shows that for $\boldsymbol{d}^g$ most of the energy is concentrated

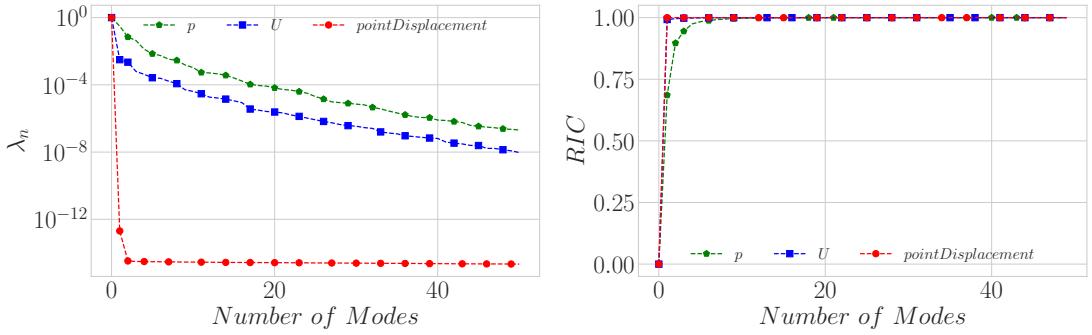


Figure 4.3: From left to right, the eigenvalues decay and cumulative eigenvalues of the POD modes. Blue lines indicate velocity eigenvalues, green lines indicate pressure eigenvalues, and red lines indicate point displacement eigenvalues

in the first POD mode. This observation implies that the ALE field can be reduced with just $M = 1$ POD mode. Thus, for such a variable the original FOM field, which featured 24 440 grid nodes, is approximated with only one degree of freedom. A similar observation was also reported in [63]. One possible reason for this very favourable DOF ratio between ROM and FOM is that making use of Slerp, most of the node's displacement $\boldsymbol{d}^g$ is occurring in a region concentrated around the moving interface, and the node's motion propagates linearly towards the far-field boundaries. Of course, this situation is quite suitable for a linear approximation such as the one provided by POD. Conversely, the eigenvalues of pressure and velocity show a significantly slower decay, with respect to the one observed for $\boldsymbol{d}^g$. This slow decay phenomenon implies that more spatial modes have to be used at the reduced order level to capture the system dynamics of the original system. As pointed out in [5], such a slow eigenvalue decay is likely due to the presence of grid deformation in the problem considered. In the same paper, the authors suggest that steeper eigenvalues decay can be obtained equipping the POD modal matrix with a domain filter. Alternatively, it is possible to treat grid deformation using Hadamard formulation for domain deformation, as suggested in [21], to carry out all simulations in a reference domain.

4.2.3.3 ROM solution error

Once the reconstruction error has been characterized, we aim to analyse the quality of the online problem solution. Thus, to evaluate how close the predicted ROM solutions are with respect to the FOM ones, Figure 4.4 illustrates a qualitative comparison between the solution fields contour plots corresponding to time $t = 20$ s obtained with both the FOM and ROM solvers. The plots confirm that, to the eyeball test, the ROM solutions obtained using the mixed POD-Galerkin projection (for the fluid dynamic variables) and POD-RBF (for the grid displacement field) appear similar to the high-fidelity ones. It is also worth pointing out that the top plots in Fig. 4.4 confirm the ROM is able to reproduce the 2S mode of the classical Von Kármán vortex street as observed in the FOM solution.

A more quantitative assessment of the ROM accuracy is presented in Figure 4.5, and, Figure 4.6 which depict the time evolution of the ROM's L^2 absolute error of the velocity and pressure fields, respectively. In the diagrams, each curve is

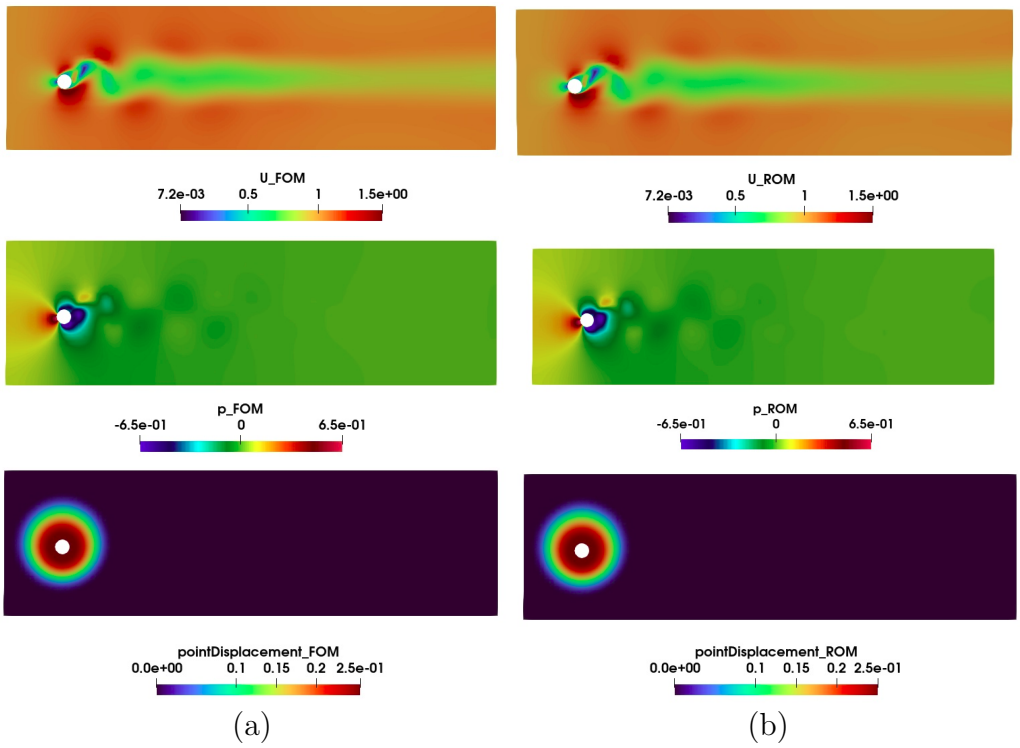


Figure 4.4: FOM and ROM solutions comparison at $t = 20$ s from left to right: column (a) FOM solutions and column (b) predicted solutions. The first row represents the velocity fields, second-row pressure fields, and third-row grid nodes displacement fields of both FOM and ROM. The reduced solution used POD 20 modes for both velocity and pressure, and 1 mode for grid nodes displacement.

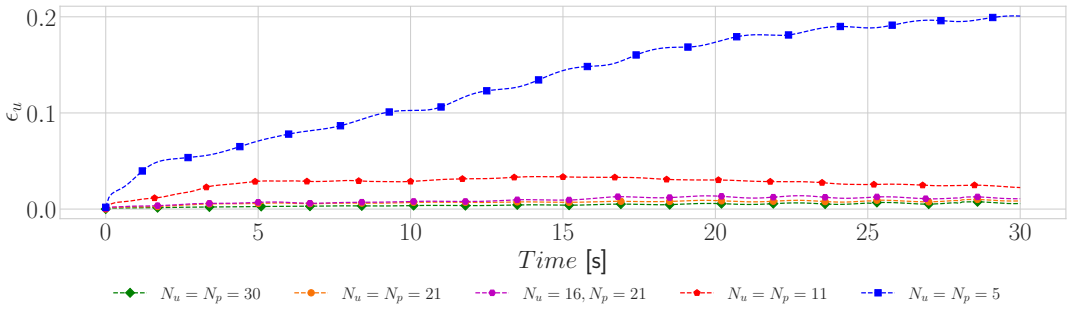


Figure 4.5: Velocity field Absolute ROM error in the L^2 norm as a function of time.

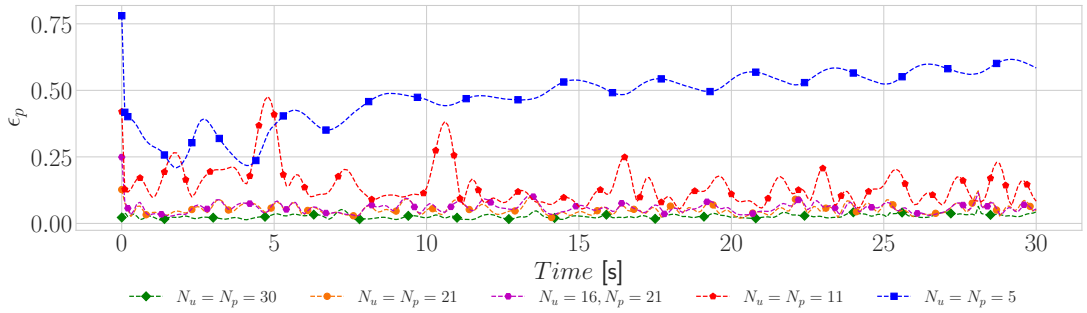


Figure 4.6: Pressure field Absolute ROM error in the L^2 norm as a function of time.

obtained with different combination of modal truncation orders. The L^2 absolute error reported in the plots is computed, for a given quantity q, as

$$\epsilon_q = ||q_{FOM} - q_{ROM}||_{L^2(\Omega)}. \tag{4.2}$$

The plots in Figure 4.5, and Figure 4.6 indicate that selecting different numbers of modes has the expected impact on the ROM prediction, as the error values drop as the number of modes used in the online stage is increased. The absolute value of the ROM solution error for the velocity field shown in Figure 4.5 can be related to the average velocity error dividing it ϵ_u by the overall domain area $A_\Omega = 339.21\,\mathrm{m}^2$. In our case, even using as low as $N_u = 5, N_p = 5$, the average velocity error in the domain is approximately $5.9 \times 10^{-4}\,\mathrm{m\,s^{-1}}$. The corresponding average error for the pressure field obtained with $N_u = 5, N_p = 5$ is $1.8 \times 10^{-3}\,\mathrm{Pa}$. Not only both values appear quite acceptable, compared to the peak velocity and pressure values — shown for instance in Figure 4.4 — but significantly lower values are obtained making use of more modes. This confirms that the methodology proposed for the online resolution of the ROM system is able to accurately approximate the FOM solution.

However, a low overall or average error in the pressure and velocity fields might still be in principle associated with high local error in small regions, for instance surrounding the cylinder. One of the main goals for researchers and engineers studying fluid dynamic problems such as the cross-flow cylinder here considered is often the evaluation of the forces acting on a body or a boundary surface in general. Such forces depend on the local values of the pressure and velocity fields around the body of interest. The global error evaluators shown so far in Figures 4.5 and 4.6 might not be good indicators if the aim is the assessment of how well the ROM solvers are able to predict the fluid dynamic forces acting on a body. The plots in Figures 4.5 and 4.6 provide in fact little information on the local distribution of such errors, which might have a relevant impact on the body forces of our FSI simulations. In such a case, both the fluid dynamic forces and the cylinder displacement might be computed with low accuracy. So, a further step in the ROM results analysis is represented by the evaluation of the fluid dynamic forces and cylinder displacement accuracy.

Fig. 4.7 depicts the time history of the lift force exerted by the fluid on the cylinder.

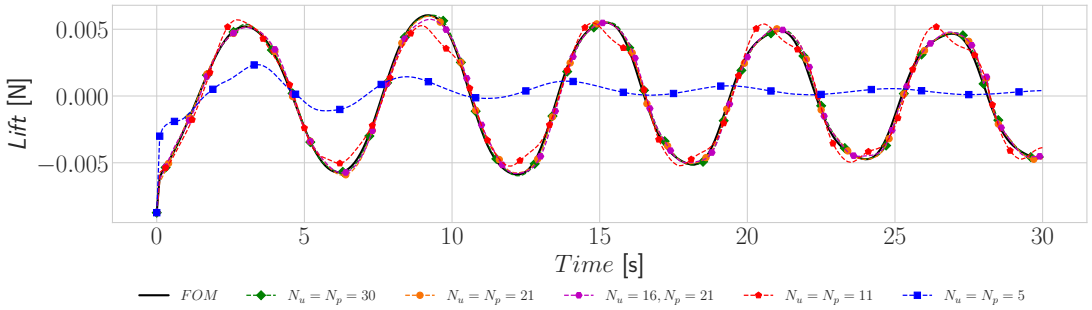


Figure 4.7: Time series comparison between the reference curve of the lift force acting on the cylinder in Newton unit with predicted curves .

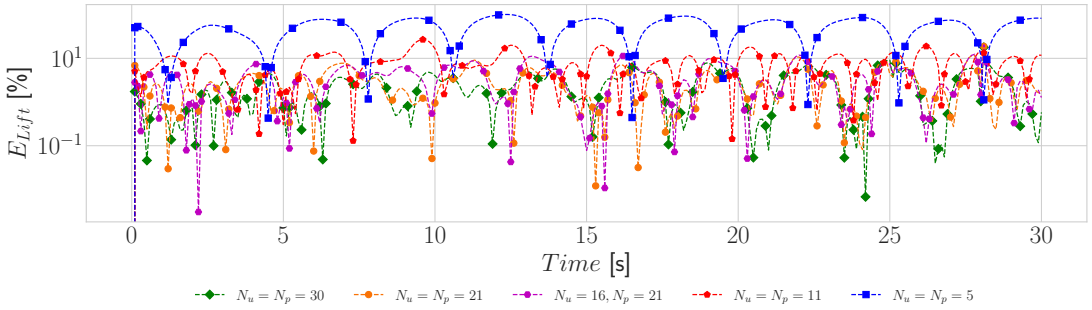


Figure 4.8: Time evolution of the absolute errors of the pressure reduced approximation. The error values in both graphs are in percentages.

In the plot, the FOM solution is compared to the ones obtained with ROMs making use of different number of pressure and velocity modes. The plot clearly shows that the ROM lift force values converge to the FOM ones as the number of modes used in the online stage is increased. The plot also suggests that the ROM methodology proposed can obtain a qualitatively good approximation of the lift force throughout the time integration window considered when as many as 21 modes are used for both velocity and pressure fields. Further confirmation of this is given by the corresponding absolute error plots presented in Figure 4.8, in which it is possible to observe that in the combinations of pressure and velocity modes using more than 20 modes each, the error is around the 1% value.

Similar plots relative to the drag are presented in Figure 4.9 and Figure 4.10. Also in this case, the diagram presents a comparison between the FOM drag curve and the corresponding curves obtained with ROM models making use of different modal truncation orders. These plots suggest that the qualitative behaviour of the cylinder resistance is well captured across all time steps of the flow simulation with a higher number of modes, and that higher absolute error appears when the number of modes decreases. Thus, it can be said that the accuracy shown by these plots is quite satisfactory when a number of both pressure and velocity modes higher than 20 is used. Additional confirmation to complement ROM accuracy is shown by comparing the power spectral density curves as depicted in Figure 6.2. The time histories of the cylinder displacement are also interesting data because the motion is not known

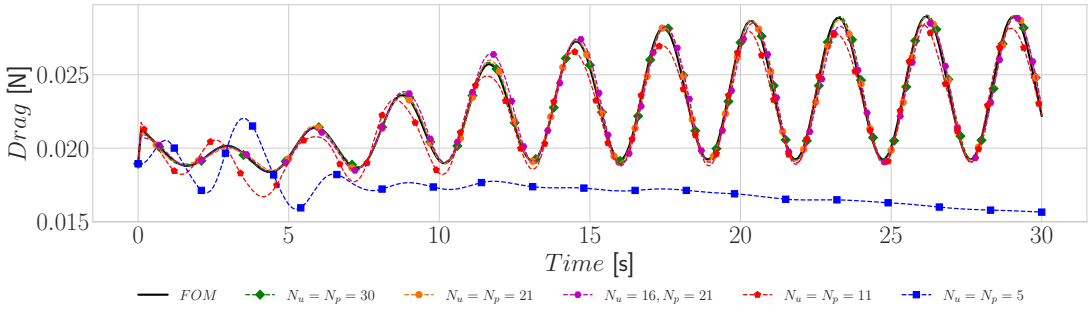


Figure 4.9: Time series comparison between the reference curve of the drag force acting on the cylinder in Newton unit with predicted curves.

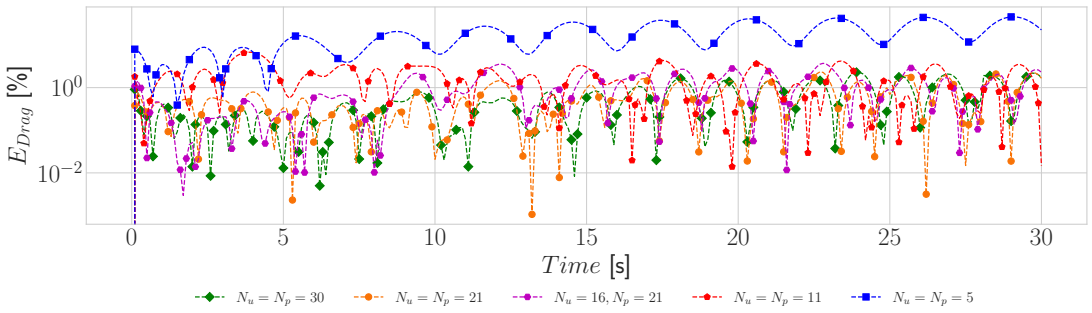


Figure 4.10: Time series of the absolute error analysis of the drag force (original and predicted signals) from Figure 4.9

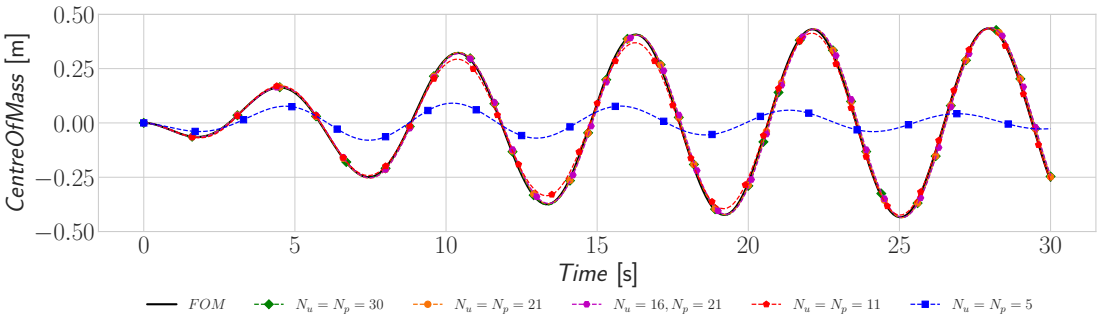


Figure 4.11: Time series evolution of the centre of mass.

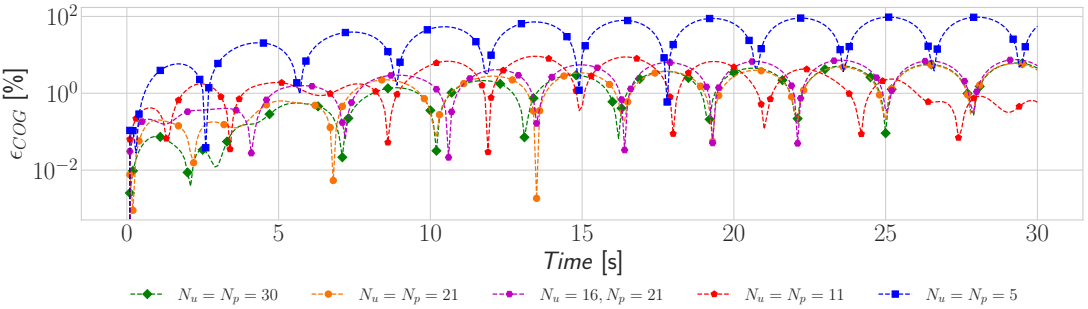


Figure 4.12: Time series evolution of the centre of mass absolute error.

as a priori, as is the case with forced vibrations. So, as a final confirmation of the proposed ROM results quality, it is also important to consider the time history of the displacement of centre of the cylinder computed during the simulations. Fig. 4.11 presents the comparison of the original curve of the displacement of the centre of the mass and the corresponding curves obtained with ROM models making use of different modal truncation orders. Also in this case, the accurately of the ROM to reproduce the time history of the cylinder motion depends on the number of modes used. This is because the displacement of the centre of the mass depend on the lift force as one can see in Eq. (2.56) and the lift force are computed using velocity and pressure fields. The ROM solution obtained making use of $N_u = 5$ velocity modes and $N_p = 5$ pressure modes (blue line) visually appears less accurate with respect to the full-order one, and to the other ROM solutions. All the curves corresponding to other modal truncation orders appear considerably more accurate, as confirmed by the error plots in Figure 4.12, in which the cylinder centre of gravity error for such ROMs fall below the 2% threshold throughout the entire time series.

Lastly, in implementing the POD for low-dimensional modelling, we project the infinite dimensional evolution equation such as the Navier-Stokes equations, onto a finite-dimensional empirical subspace, of possibly quite low dimension. One natural question that arises is how well do the truncation and projection approximate the attractor present in the original dynamical system [69]. Additional plots, not reported here, give an answer to this question where the ROM accurately reproduces all the limit cycles present in the original system.

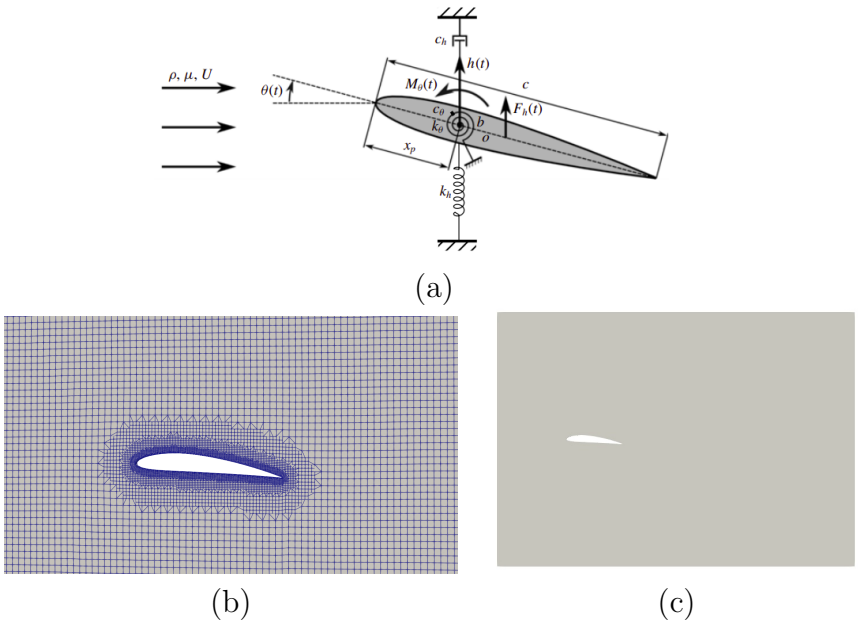
4.3 Unsteady Turbulent Case

This result section, in full, is a reprint of: *A hybrid reduced-order model for segregated fluid-structure interaction solvers in an ALE approach at high Reynolds number.* **Submitted**, 2024.

4.3.1 Definition of test case, simulation results, and discussion

This section shows the results obtained for our reference test case which represents two-dimensional turbulent flow past a plunging and pitching airfoil. The simulation is carried out for a total time of 500 flow through times (FTTs). In the present case, such time unit is defined as the time $\text{FTT} = \dfrac{L}{||\boldsymbol{U}_\infty||} = 0.01$ required by a fluid particle to travel a distance equivalent to the airfoil chord L at the speed of the undisturbed stream $||\boldsymbol{U}_\infty||$ [79].

The 500 FTTs duration choice allows for the flow to fully develop also in the wake region. The test case of the study is the analysis of a two-degree freedom flutter as shown in Figure 4.13.



(a)

(b) (c)

Figure 4.13: (a) Schematic of the fluid-structure system considered: a foil allowed to undergo 2 degrees of freedom fully passive plunging and pitching motion with spring constraints [171], (b) a picture of the zoomed mesh with 12 556 cells (control volumes) and 26 316 node points near an airfoil of chord length 1.0 m, (c) picture showing the position of the foil in the computational domain

4.3.1.1 Definition of the test case

Fig. 4.13 (c) shows the 2D computational domain used in this work. The grid features 12 556 cells (control volumes) and 26 316 node points. It also features an extrusion layer around the airfoil to better capture the physical boundary layer.

The molecular viscosity $\nu = 10^{-5}\,\mathrm{m}^2/\mathrm{s} \approx \nu_{air}$. The boundary conditions prescribed for the velocity field at the inflow boundary are non-homogeneous Dirichlet. The velocity value imposed is that of a uniform and constant horizontal velocity $\boldsymbol{U}_\infty = (U_{in}, 0)$ with $U_{in} = 10^2\,\mathrm{m\,s}^{-1}$. Given the airfoil's chord length $L = 1.0\,\mathrm{m}$, the resulting Reynolds number is 10^7. On the — moving — airfoil boundary a Dirichlet boundary condition is applied, imposing that the fluid's velocity is equal to the airfoil surface one. On the top and bottom boundaries, we made use of symmetry boundary conditions, while, zero pressure value and zero normal velocity gradient are prescribed at the outflow boundary. The full-order simulations are carried out using the PIMPLE algorithm, as described in Algorithm 2. The PIMPLE algorithm can adapt the time step in a way that assures the maximum Courant number does not exceed a prescribed value which in this case, has been set to 0.5. As Reynolds average Navier-Stokes (RANS) is used in this work, the time step Δt is chosen based on the following *rule of thumb* [79]: $\Delta t \approx 100\Delta t_{DNS}$.

$$\text{with} \quad \Delta t_{DNS} \approx \frac{Co \times \eta}{U_{in}} \quad \text{and} \quad \eta \equiv L \times Re^{-\frac{3}{4}}, \tag{4.3}$$

where L is the size of the largest eddies (in this work, $L = 1m$), η is the Kolmogorov length scale and it is the smallest hydrodynamic scale in turbulent flows. The turbulence model used in this test is the $k-\omega$ SST model, which in several works (see for instance [130]) proved capable of simulating turbulent flows associated with vortex induced vibrations.

The Implicit Euler scheme is used for time discretization, and for the spatial gradients, a Gauss linear scheme is employed. The convective and diffusive terms have been approximated with the first-order Upwind scheme [153] for more stability. The reason is that, in the transport-dominated turbulent regime here under study, the local Peclet number can reach peak values greater than 2. The values of the relaxation factors α_u, and α_p are fixed at 0.7 and 0.3 respectively. One non-orthogonal correction at each PIMPLE iteration is used to deal with the mesh's non-orthogonality. In addition, one pressure correction (inner correctors) and two momentum corrections (outer correctors) are used in the simulations. The linear solver selected combines a smoother Gauss-Seidel solver used for the pressure equation, and a symmetric Gauss-Seidel solver for the momentum equation. The structural motion is computed by means of the `sixDoFRigidBodyMotionSolver` OpenFOAM solver. Given the external fluid dynamic forces acting on a rigid body, such a solver is able to compute the linear and angular displacements in three dimensions. However, the airfoil here considered is only free to translate along the vertical direction (plunge displacement) and rotate along the axis perpendicular to the planar domain Ω (pitch displacement). The resulting system of two second-order differential Eqs. (2.57) and (2.58) is solved using the Symplectic second-order explicit time-integrator for solid-body motion [48]. The Arbitrary Lagrangian-Eulerian (ALE) method deals with the motion of the grid nodes resulting from the fluid-structure coupling. In particular, the plunging displacement h, and the pitching displacement θ of the airfoil are used to deform the mesh (in the transverse and rotational directions), as described in SubSection 3.3.4. Table 4.4 reports a comprehensive summary of all the modelling and numerical parameter values used for the simulation setup.

4.3.1.2 Simulation results

This section presents the results obtained with the reduced model developed on the airfoil test case described earlier. As mentioned, the ROM is based on the POD-Galerkin approach for the momentum and continuity equations (velocity and pressure fields), on POD-LSTM or POD-NNs for the online eddy viscosity computation, and POD-RBF for the mesh displacement update.

Table 4.4: Summary of simulation settings of the flow passing pitch-plunge airfoil

Flow settings		**Structure settings**	
Re	10^7	$f_{sh} = f_h = f_\theta$	20 Hz
Time scheme	Implicit Euler	Time scheme	Sympletic
Gradient scheme	cellLimited Gauss linear 1	m	22.9 g
Convective scheme	Gauss upwind	g_z	$-9.81\,\mathrm{m/s^2}$
Laplacian scheme	Gauss linear limited 0.5	L_z	-2
$St = \frac{f_{sh} c \sin\alpha}{U_\infty}$	0.2	k_h	$3.6262\times10^5\,\mathrm{N\,m^{-1}}$
U_{in}	$10^2\,\mathrm{m\,s^{-1}}$	k_θ	$3.25\times10^4\,\mathrm{N\,m^{-1}}$
Co	0.5	c_h	$2\,\mathrm{N\,m^{-1}}$
Δt_{DNS}	$\frac{C_0\times Re^{-\frac{3}{4}}}{U_{in}}$	c_θ	$5\times10^{-1}\,\mathrm{N\,m^{-1}}$
Turbulence model	$k-\omega$ SST	I_{zz}	2.057121362

In Table 4.4, St is the Strouhal number, f_{sh} the frequency of the vortex shedding. L_z is the angular momentum in Z-direction, and g_z the gravity in the Z-direction. The following quantities: m, f_h, f_θ, k_h, c_θ, k_θ, c_h, and I_{zz} are defined in Section 2.5.2 Note that the FVM C++ library *OpenFOAM* version 2106 [76] has been used for data collection at the full-order model level. Such a numerical solver, widely used in industrial applications [75, 147] exploits the fact that FVM locally respects the balance of momentum and mass. At the reduced level, the reduction and resolution of the reduced system are carried out using the C++-based library ITHACA-FV (In real Time Highly Advanced Computational Applications for Finite Volumes) [155, 157]. ITHACA-FV is designed to carry out Galerkin projection of PDE problems that, at the full-order level, are solved making use of FV discretization based on OpenFOAM. The interpolation using RBF in this work has been carried out using the C++ library SPLINTER [61]. The radial basis used for interpolation is the *thin plate spline* with the radial basis's radius set to 1. Finally, the RNNs/ NNs are built using the PyTorch library [123]. The next Subsection assesses the qualitative prediction of the reduced model.

4.3.1.3 Prediction quality

The point of this Subsection is that of establishing the accuracy of the modal decomposition upon which the reduced model is based. Table 4.5 presents the eigenvalues associated with the first five dominant modes of all the fields of interest. The values reported also suggest that for the grid nodes motion (pointDisplacement field), one

single mode could be enough to predict the mesh motion with acceptable accuracy. Hence, in this study, 1 mode will be used to predict the airfoil motion. A more comprehensive view of the modes eigenvalues magnitude is presented in Figure 4.14.

# Modes	Eigenvalues U	Eigenvalues p	Eigenvalues ν_t	Eigenvalues pointDisplacement
1	1	1	1	1
2	0.001480623	0.07835354	0.007290981	0.027573111
3	0.001133095	0.008968419	0.001695786	0.00201184
4	0.000992664	0.001758634	0.000879206	0.000000351
5	0.000356768	0.000941899	0.000580963	0.00000000018

Table 4.5: Normalized eigenvalues of the POD modes of the fields of interest

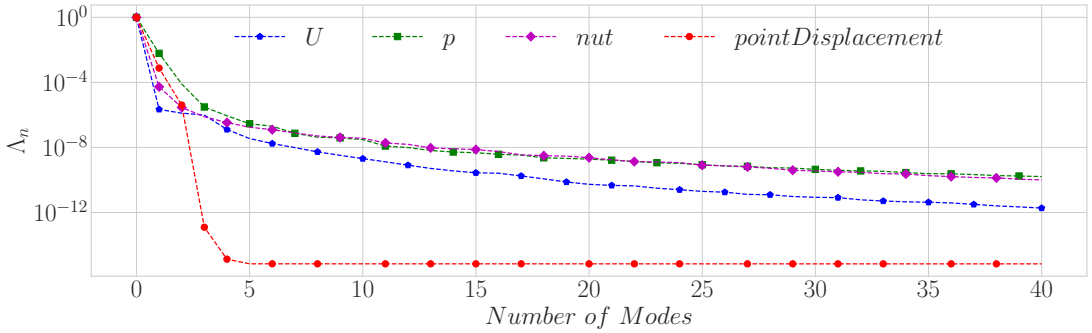


Figure 4.14: The decay of the POD modes eigenvalues for velocity, pressure, pointDisplacement, and Eddy viscosity fields. Color code: blue – velocity, green –pressure, red – pointDisplacement, magenta – eddy viscosity

4.3.1.4 Machine learning of the temporal eddy viscosity coefficients

For all the training runs in this work, a variant of the stochastic gradient descent algorithm called adaptive moment estimation ADAM [78] is used as an optimizer. ADAM has an adaptive learning rate method which is commonly used to train deep networks. The optimization is based on a scaled version of the modal coefficients, given by

$$\hat{\boldsymbol{a}}_j(t) = \frac{\boldsymbol{a}_j(t) - \langle \boldsymbol{a}_j(t) \rangle}{\sigma[\boldsymbol{a}_j(t)]}, \tag{4.4}$$

where $\langle \boldsymbol{a}_j(t) \rangle$ is the mean value of the modal coefficient time series considered and $\sigma[\boldsymbol{a}_j(t)]$ is the corresponding variance. The dataset scaling is necessary to avoid that the gradients that enter the computations of the cost function are too small. In such a case, it would be impossible to generate significant updates of the parameters of the network [20]. The model parameters (weights and bias) are trained with the PyTorch library [124] and later imported in the C++ solver to generate the transient predicted solution for the eddy viscosity during the online computations. The full

details on training, validation, and testing are reported in 6.1. The accuracy of the resulting feed-forward NN and LSTM-RNNs model results are illustrated in Fig. 6.1. The four diagrams represent the time series of the modal coefficients corresponding to the four energetically dominant modes of the full-order model eddy viscosity, as well as their data driven approximations. In the diagrams, the two dashed vertical lines divide the time axis into the training window, on the left, the validation window, located between the red and blue dashed lines, and the testing window, on the right. The picture confirms that both models are able to capture the overall trend of the reference FOM solution, not only in the training time window, but also in the and validation and testing ones. By a quantitative perspective, both data driven models appear accurate in the reproduction of frequency, amplitude and phase of the first three modal coefficients of the eddy viscosity field. The plot corresponding to the fourth modal coefficient shows instead a drop of the accuracy of the feed-forward NN prediction, while, on the other hand, the LSTM-RNN prediction remains as accurate as for the previous modes. Thus, this preliminary analysis suggests that both feed-forward NN and LSTM-RNN data driven algorithms used are in principle capable of approximating with good accuracy the eddy viscosity modal coefficients. This is of course crucial for a correct closure of the turbulent problem at the reduced level. The next sections will then assess if the quality of the eddy viscosity approximation — which appears high except for higher order modal coefficients obtained with feed-forward NN — will translate into accurate ROM results.

4.3.1.5 ROM online resolution time

The data were generated by transient simulations running for one second and saving snapshots of the flow field every 0.0005 seconds for a total of 2001 snapshots. All simulations were run on an HP Pavilion laptop with AMD Ryzen 7 5700u with Radeon graphics ×16, 16GB RAM, AMD Renoir graphics card, and Ubuntu 20.04 operating system. Table 4.6 reports a comparison analysis of the full-order and reduced-order models execution times as the number of modes for the prediction of velocity, pressure, pointDisplacement, and eddy viscosity is varied. This allows for evaluating the effect of the number of modes variation on the computational cost of the online phase.

The offline stage comprises four steps: the computation of the snapshot (computed by a numerical approximation of the original high-dimensional system), computation of the POD basis, projection of the dynamics on the low-rank subspace, and the Machine Learning training of the neural networks (including for the radial basis networks). But only the computational coast of the first step is reported in Table 4.6 as it is the most expensive one. The online coast is the computational time needed to compute the solutions of the surrogate model. In Table 4.6, one can observe a small speed-up, as this work does not employ hyper-reduction technique.

However, the results in Table 4.6 suggest that the speed-up obtained by the online solution of the reduced system is not proportional to the reduction of the unknowns obtained at the reduced-order level. This is due to the fact that in the presence of a deforming domain such as the one characterizing our FSI simulations, the entries of the matrices of the ROM system must be computed at each time step through integrals on the updated full-order grid. Clearly, this is at the moment representing a major bottleneck towards a ROM which grants significant computational cost reduction with respect to its FOM counterpart, and work is being carried

Stages	*# of modes*	Time [s]
Offline	-	3.441516e+4
POD-NNs	$N_u = N_p = 5,\ N_{nut} = 3,\ N_{pD} = 1$	1.770725259e+4
	$N_u = N_p = 10, N_{nut} = 5, N_{pD} = 1$	1.957406359e+4
	$N_u = 15, N_p = 5,\ N_{nut} = 2,\ N_{pD} = 3$	2.159144848e+4
	$N_u = 10, N_p = 5, N_{nut} = 3,\ and\ N_{pD} = 1$	1.942751287e+4
POD-LSTM	$N_u = N_p = N_{nut} = 5\ and\ N_{pD} = 3$	1.766056714e+4
	$N_u = 15, N_p = 5,\ N_{nut} = 3\ and\ N_{pD} = 3$	2.136608204e+4

Table 4.6: Offline and Online times comparison varying the number of modes

out towards lowering the computational cost associated with the reduced model assembling. Nonetheless, the main goal of the present work is that of assessing the accuracy of the ROM approach taken. In particular, it is important establishing whether the interaction between the physics based reduction of the fluid dynamic balance equations, and the data driven reduction of the turbulence and grid displacement equations, results in an accurate solver.

4.3.1.6 ROM online resolution quality

The present Subsection aims to analyse how close the predicted ROM solutions are to the FOM ones. To this end, Figures 4.15 and 4.16 shows a qualitative comparison between the solution fields contour plots corresponding to time $t = 0.1$ s obtained with both the FOM and ROM solvers. The plots in the figure confirm that, to the eyeball test, the ROM solutions obtained using both POD-NNs and POD-LSTM appear by all means similar to the high-fidelity ones.

More quantitative considerations can be driven from Figure 4.17, Figure 4.18, and Figure 4.19 which depict the time evolution of the ROM L^2 error of the velocity, pressure, and eddy viscosity obtained with different combinations of modal truncation orders for the velocity field. Note that the L^2 relative error for a given quantity q is computed as follows:

$$\epsilon_q = \frac{||q_{FOM} - q_{ROM}||_{L^2(\Omega(t))}}{||q_{FOM}||_{L^2(\Omega(t))}} \times 100\%. \tag{4.5}$$

The results in Figure 4.17 confirm the qualitative impression of accuracy given by Figures 4.15 and 4.16, as the velocity field errors plotted remain well below the 1% threshold for the entire simulation. The plot also clearly indicates that the velocity field accuracy obtained making use of POD-NN is higher than that obtained with the POD-LSTM approach. This is further confirmed by Figure 4.18, in which the error associated with POD-LSTM is approximately as high as 1%, while the POD-NN approach results in appreciably lower error levels. The satisfactory results shown in these plots depend on the NN/ LSTM algorithm effectiveness in the calculation of

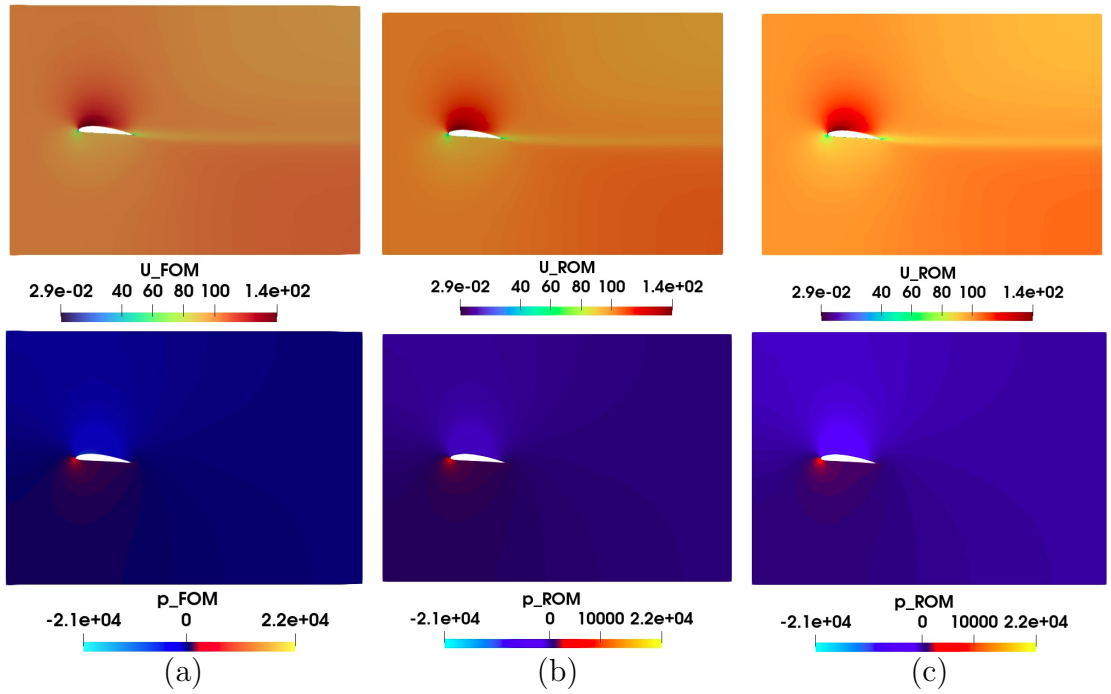


Figure 4.15: Comparison of the velocity and pressure fields. First row velocities comparison and second row pressure comparison. Column (a) FOM fields, column (b) reduced solution with POD-NNs and column (c) reduced solution with POD-LSTM. The snapshots are captured in the second period i.e. $t = T = 0.1$ s

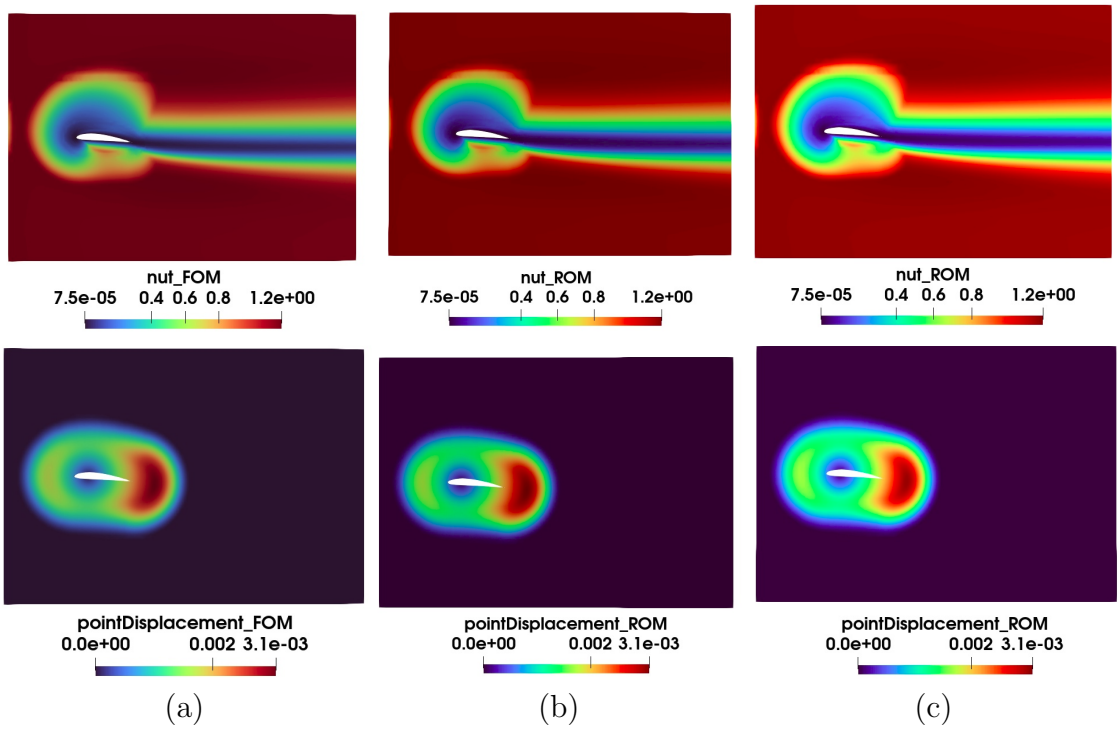


Figure 4.16: Comparison of the eddy viscosity and grid node displacement fields. First row eddy viscosity comparison and second row grid node displacement comparison. Column (a) FOM fields, column (b) reduced solution with POD-NNs and column (c) reduced solution with POD-LSTM. The snapshots are captured in the second period i.e. $t = T = 0.1$ s.

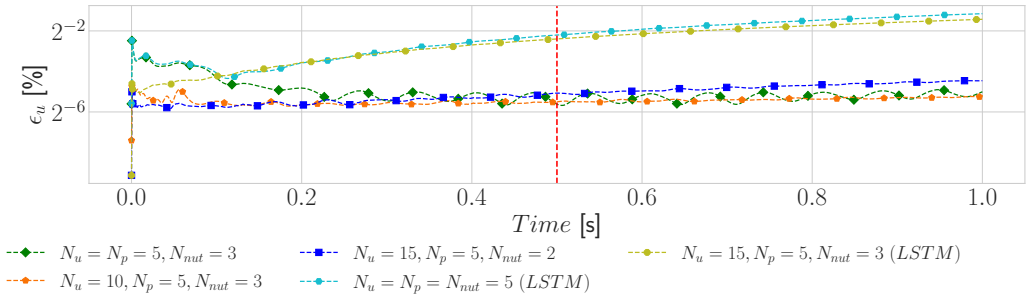


Figure 4.17: Sensitivity study of the error (log-scale) in the L^2-norm versus the time evolution of the velocity field. The red line shows the results obtained inside and outside the time window

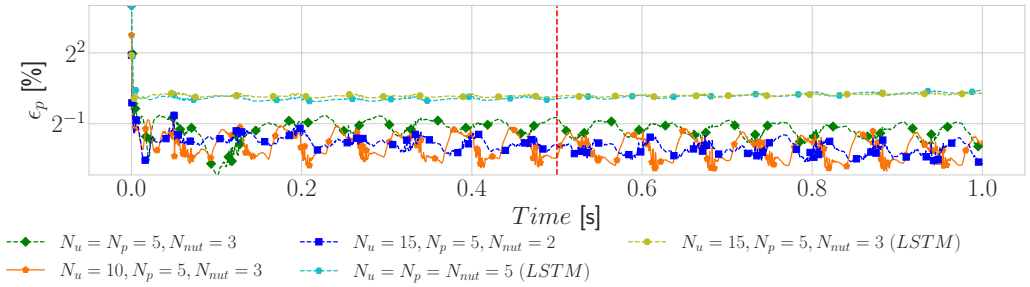


Figure 4.18: Sensitivity study of the error (log-scale) in the L^2-norm versus the time evolution of the pressure field. The red line shows the results obtained inside and outside the time window

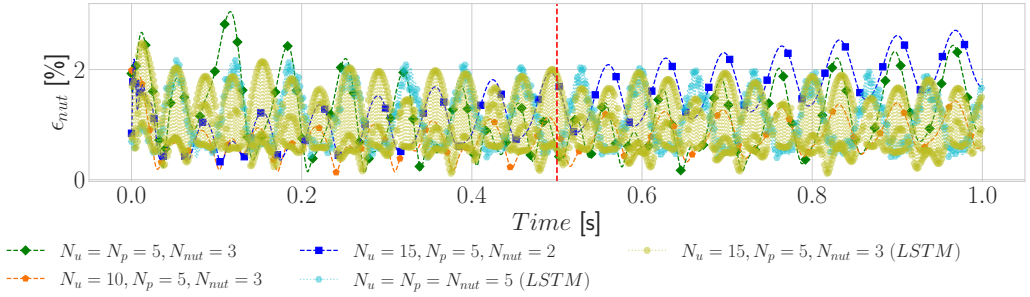


Figure 4.19: Sensitivity study of the error in the L^2 norm in log-scale versus the time evolution of the eddy viscosity field. The red line shows the results obtained inside and outside the time window

the eddy viscosity field POD coefficients time evolution. In this regard, it is worth pointing out that the Figure 4.19 error magnitude suggests that a 1-2% error level for the eddy viscosity field approximation, leads to consistently lower error values on the velocity and pressure fields. These observations seem to further validate the data driven approach taken for the eddy viscosity field, as it appears to lead to small errors of pressure and velocity fields, which are the main fields of interest for our simulations. Fig. 4.19 also shows that the error associated with the POD-LSTM approach presents a pattern characterized high frequency oscillations, which might be the culprit for the higher error levels observed in the velocity and pressure fields. This is consistent with what other researchers have observed in [20, 52, 154] in the prediction of the velocity field in a channel flow. In their study, they reported that the phenomenon was due to an insufficient number of snapshots of the training dataset or when the number of cells in the hidden layer was not enough. In addition to that, the lifetime of a transient turbulent state is highly sensitive to the initial conditions and if only one component of the initial conditions differs by 10^{-12}, the resulting trajectory will diverge from the truth one. It is also possible that the memory in the sequence can affect the LSTM accuracy as reported in [109] because signals originating from chaotic dynamic systems are known to have quite short correlated events and memory does not typically persist over long periods. In this case, the Hurst exponent is a prominent solution [70] for further investigations. However, the mentioned suggestions were not the emphasis of this paper. Instead, the emphasis was concentrated on the design of the overall solver to generate a reduced model for segregated FSI solvers for turbulent regime in the FV context.

The plots in Figure 4.17, Figure 4.18, and Figure 4.19 also visualize the effect of the number of velocity modes on the accuracy of the ROM solutions. In general, increasing the number of velocity modes considered at the online level increases the accuracy, especially in the initial transient part of the time integration. However, an increase to values higher than 10 modes does not appear to result in significant gains. It is finally important to point out that the snapshots for the POD have been collected only in a training window corresponding to the first half of the time history plotted — the one on the left of the red dashed vertical line in each plot. So, the plots also indicate that, in the case of a periodic problem as the one analyzed, the ROM errors are not significantly growing if time extrapolation is carried out.

A further step in the ROM results analysis is represented by the evaluation of the fluid dynamic forces and airfoil displacement accuracy. In fact, the L^2 errors discussed in the previous plots provide information on the average discrepancy of the most relevant fluid dynamic quantities in the fluid domain. However, the previous plots provide little information on the local distribution of such errors, which might have relevant impact on our FSI simulations. In fact, a low overall error in the pressure and velocity fields might still be in principle associated with high local error in small regions, for instance surrounding the airfoil. In such a case, both the fluid dynamic forces and the airfoil displacement might be computed with low accuracy.

Fig. 4.20 depicts the time history of the lift force exerted by the fluid on the airfoil. In the plot, the FOM solution is compared to the ones obtained with ROMs making use of different number of pressure and velocity modes. The plot suggests that both the ROM methodologies proposed are able to obtain qualitatively good approximation of the lift force throughout the time integration window considered,

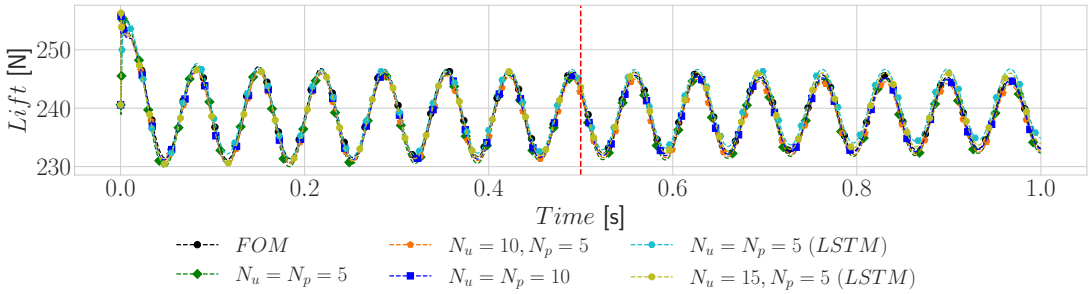


Figure 4.20: Time series comparison between the reference signal of the lift force acting on the foil in Newton unit with predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation).

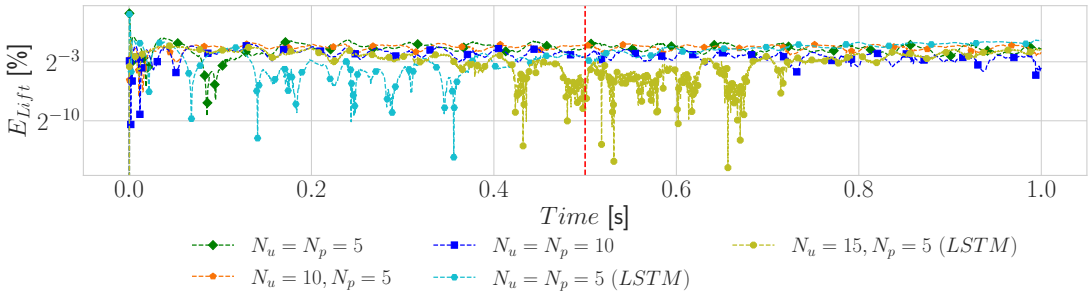


Figure 4.21: Time series of the error analysis of the lift force (original and predicted signals) from Figure 4.20. The vertical line in red divides the error plots in two. Left: interpolation error and right: extrapolation error.

even including the initial transient. Also in this case, the plot indicates with a red dashed vertical line the separation between the training window, on the left, and the time window, on the right, in which the solution is extrapolating over the time variable. An inspection of the lift curves suggests that, when POD-NN is considered, no significant error increase is associated to time extrapolation. As for POD-LSTM, a slight degradation of the prediction quality is observed in the extrapolation region. Further confirmation of this is given by the corresponding error plots presented in Figure 4.21, in which it is possible to observe that for all the combination of pressure and velocity modes considered, the error remains below the 1% threshold, except for the initial transient, in which the $N_u = N_p = 5$ solution for both POD-NN and POD-LSTM presents a slightly higher error.

Similar plots relative to the airfoil drag are presented in Figure 4.22 and Figure 4.23. Also in this case, the figure presents a comparison between the FOM drag curve and the corresponding curves obtained with ROM models making use of different modal truncation orders. The plot suggests that the qualitative behaviour of the airfoil resistance is well captured across all time steps of the flow simulation, including the extrapolation time window. The value of the drag error obtained with POD-NN is again lower than 1% for the most part of the overall time history,

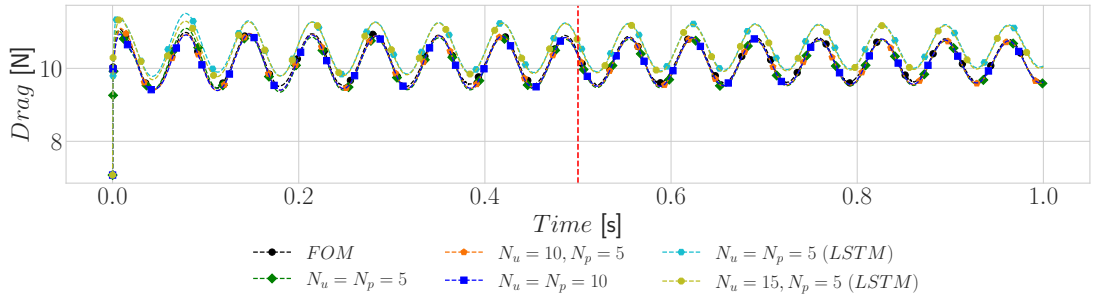


Figure 4.22: Time series comparison between the reference signal of the drag force acting on the foil in Newton unit with predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation).

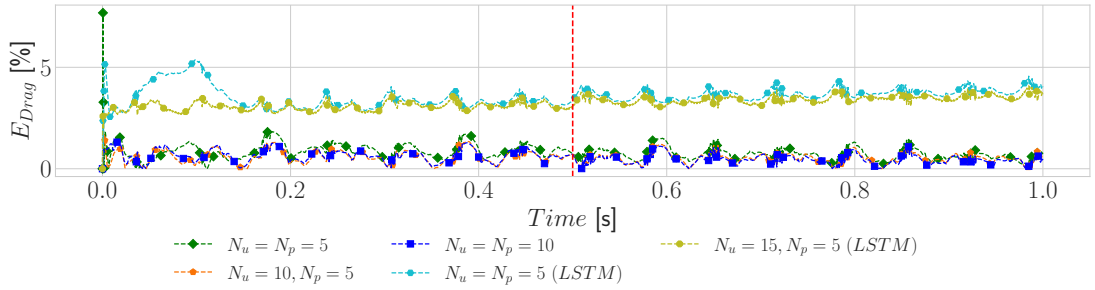


Figure 4.23: Time series of the error analysis of the drag force (original and predicted signals) from Figure 4.22. The vertical line in red divides the error plots in two. Left: interpolation error and right: extrapolation error.

although non-negligible portions of the error curve pass such a threshold. Such a higher relative error is just a product of the lower absolute value of the drag force with respect to the lift values. Thus, it can be said that the accuracy shown by the plots is quite remarkable, and is not degrading in the time extrapolation window. On the other hand, the error obtained with POD-LSTM appears to settle for higher values, although it remains below 5% for the time window analysed, including the extrapolation region. Also, in this case, the increased error scale with respect to the lift should be associated with the smaller magnitude of the drag force absolute value and amplitude.

As a final confirmation of the proposed ROM results quality, it is important to also consider the time history of the airfoil displacements computed during the simulations. Fig. 4.24 depicts the airfoil centre of gravity position, as computed at each time step by the FOM and by the ROM models proposed. The plot clearly indicates that all the reduced-order model solutions closely track the full-order one. For the most part of the time window — which as usual is divided into a training part and an extrapolation part by the red dashed line in the plot — the plunge coordinates curves obtained with all the POD-NN models considered appear in fact overlapped to the FOM one. The POD-LSTM results obtained making use of

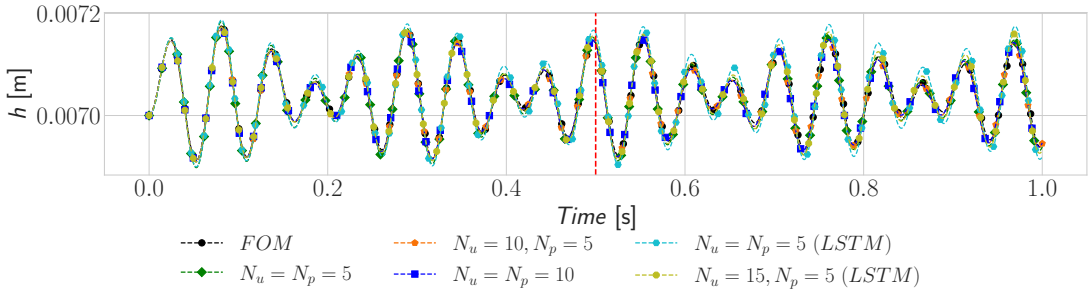


Figure 4.24: Time series comparison between the reference signal of the plunge with different predicted signals. The vertical line divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation).

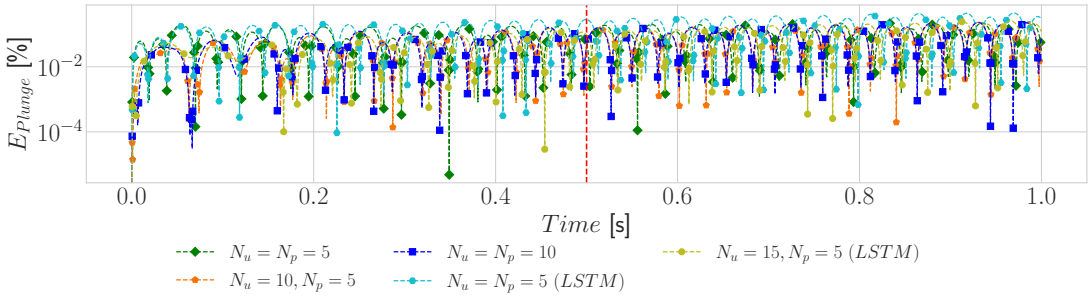


Figure 4.25: Plunge's error analysis versus time.

$N_u = 15$ velocity modes and $N_p = 5$ pressure modes have accuracy comparable with the POD-NN ones. Instead, the POD-LSTM curve associated with $N_u = N_p = 5$ visually appears less accurate, especially in the extrapolation region. The corresponding error plot is presented in Figure 4.25. The values reported in the diagram substantially confirm that the POD-NN error obtained with all the modal truncation combination considered, is in average as low as 0.1%, and always below the 1% threshold. Again, it has to be remarked that this satisfactory result does not appear to be negatively affected by time extrapolation, as the values in the second half of the plot remain generally low.

Fig. 4.26 shows the time history of the airfoil pitch angle, as simulated with the FOM solver, and with ROM solvers making use of different eddy viscosity approximation strategies and different modal truncation orders. Here, all the ROM solvers tested — even the ones with lower truncation orders — lead to airfoil pitch curve approximation that are barely distinguishable from the FOM one. This is suggesting that not only the aerodynamic force, but also its point of application are reproduced with accuracy at the reduced order level. As a consequence, the pitch angle percentage error plot, presented in Figure 4.27 displays errors that are consistently below the 0.1% value throughout the entire simulation, for all the ROM models considered.

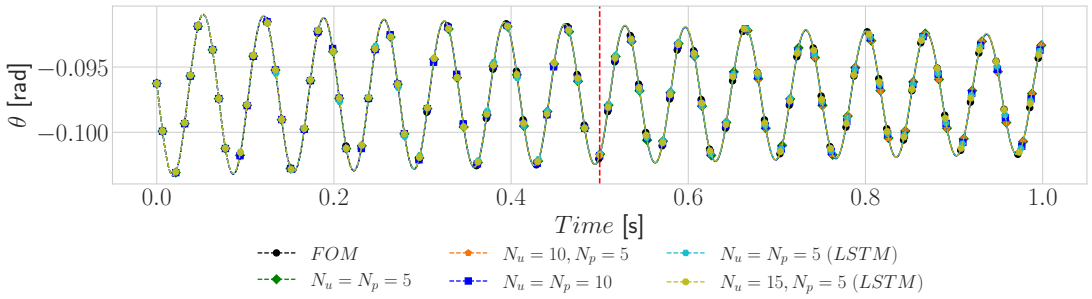


Figure 4.26: Time series comparison between the reference signal of the pitch (angle of attack) with different predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation).

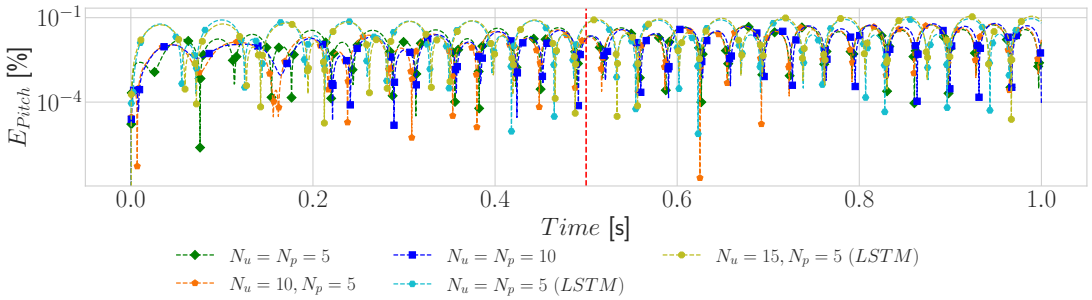


Figure 4.27: Pitch's error analysis versus time.

4.4 Empirical Lagrangian Interpolation

In this paragraph, as a preliminary result on hyper-reduction, we consider the Burgers' equation, the simplest well-known non-linear time-dependent partial differential equation. The equation has found applications in fields as diverse as number theory, gas and fluid dynamic, heat conduction, elasticity, viscous fluid, free turbulence, and continuous stochastic processes. Besides, this equation is often used to test and compare numerical techniques [148]. Burgers' equation, being a non-linear PDE, represents various physical problems arising in engineering, which are inherently difficult to solve [149].

The Burgers' equation is similar to the Navier-Stokes Equations (NSEs) without the pressure term. In this application, we consider the following 2D Burgers' equation,

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{u} = 0, \quad (\boldsymbol{x}, t) \in \Omega \times [0, T] \tag{4.6}$$

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \tag{4.7}$$

$$\boldsymbol{u}(\boldsymbol{x}, t)|_{\partial\Omega} = 0 \quad t \in [0, T]. \tag{4.8}$$

Where $\Omega \subset \mathbb{R}^2$ is a bounded domain with boundary $\partial\Omega$. $\boldsymbol{u}$ being the velocity, ν the viscosity. In this application, the viscosity $\nu = 0.0001$. The time step is set to $\Delta t = 0.005s$, but a snapshot is collected every 20 time steps. We solve the 2D non-linear conservation law with OpenFOAM [118]. Thus, we employ the FVM in a structured orthogonal grid of 12 225 cells. At the inlet boundary, non-homogeneous Dirichlet and zero gradient are prescribed for the velocity field. At the outlet boundary, zero gradient is prescribed for the velocity. On the upper and lower walls (top and bottom) no slip boundary conditions are prescribed for the velocity. In the full order simulations, Gauss linear scheme was selected for the approximation of the gradients and Gauss linear scheme with non-orthogonal correction was selected to approximate the Laplacian term. A Gauss upwind scheme was instead used for the approximation of the convective term. The FOM numerical scheme is the SIMPLE algorithm. The time discretization is performed with the semi-implicit Euler method. The linear solver used for the velocity equation is based on the GAMG with Gauss Seidel smoother until a tolerance of 1e-08 on the FVM residual is reached.

Figure 4.28: The 50 Lagrangian points are represented in red, the stencils of the cells and associated degrees of freedom needed for the evaluation of the discrete differential operators are in light-green. The discarded nodes in the evolution of the dynamics are in blue. The stencil is made by 1 layer of cells.

In Figure 4.28, the first 50 empirical Lagrangian interpolation points are displayed. In this case, the estimation problem consists of reconstructing the remain-

ing $N_h - M$ components of $\boldsymbol{u}$ from the available M measurements, where M is the number of Lagrangian points, and $N_h = 12225$ is the number of cells in the domain.

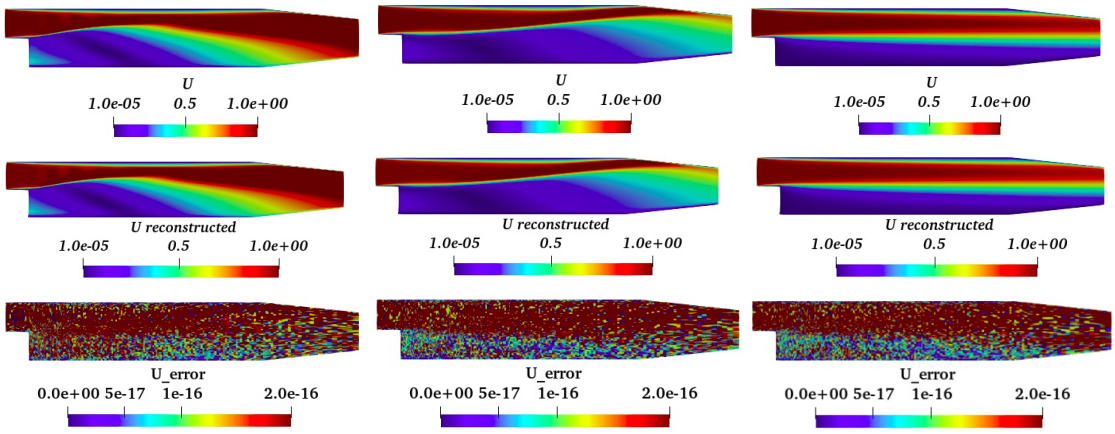


Figure 4.29: Comparison of the velocity field. First row original solutions, second row reconstructed solutions using 50 Lagrangian points, and third row the reconstructed errors. Snapshots are given at $t = 0.1$ s, $t = 0.22$ s, and $t = 3$ s.

A close inspection to Figure 4.28, and Figure 4.29 one can notice that the Lagrangian points are located where the solution has the highest value according to the colour bar indicator.

4.5 Summary

In this chapter, we have presented the results of the application of various reduced-order models (ROMs) on different CFD problems. These ROMs have been developed in this thesis for the goal of reducing problems discretized by the FVM. In addition, a number of these ROMs were specifically constructed for the reduction of FSI problems in both laminar and turbulent regime for incompressible flows. The results shown in this chapter had been presented in the following works [113, 115]. Eventually, the main goal of this chapter was to evaluate the performance of the efficiency of the reduced models developed in this dissertation. In Section 4.2, we first tested the reduced model on the flow passing a translation cylinder in the laminar regime. Next, in Section 4.3, we considered a different test case on a flow passing a translation and rotational airfoil at high Reynolds number and the last application in Section 4.4 was applied on the backward step case on a Burgers's equation. The last test case was presented as a preliminary results on the hyper-reduction technique to circumvent the expensive cost of the Galerkin projection.

Chapter 5

Conclusions and Outlooks

This chapter firstly presents the conclusions which are drawn from the results of the work done in this thesis. Secondly, it also gives an idea of possible future extensions which could enhance or complete the work presented in this dissertation.

5.1 Conclusions

This thesis makes a substantial contribution to the field of reduced-order model for fluid-structure interaction for segregated solvers in FV settings by developing enhanced models, novel techniques, integrated framework, and improved computational methods, all validated through real-world applications. These advancements might not only push the boundaries of academic knowledge but also offer practical solutions for engineering challenges, paving the way for future research and innovation in this dynamic field. The main goals presented in this thesis were the following

- The development of ROMs techniques specifically tailored for finite volumes discretization schemes for segregated solvers for moving boundaries problems such as FSI. This is particularly relevant since the FVM is mostly use in industries to simulate real-life problems.
- The implementation of ROMs designed for the reduction of both laminar and turbulent flows problems. As these types of flows are ubiquitous in real-world applications, there is in fact an increasing demand to simulate them efficiently, accurately, and inexpensively.

The following items summarize the methodologies used in this dissertation

- The Chapter 2 first presented the strong form of the governing equations of the problem of interest, which is the incompressible Navier-Stokes Equations (NSEs). Next, it addressed as well the discretization technique employed at the full-order level, which is the Finite Volume Method. After that, the numerical algorithms used by the full-order solver are also explained.
- In Chapter 3, the attention shifted to the reduction methodologies used in this thesis. At first, the POD as a method for generating reduced-order spaces is addressed. After that, both projection-based method and hybrid techniques are presented. The first hybrid technique combines the POD and interpolation using RBF to approximate the grid mesh motion, and the second mixes the

POD and machine learning algorithms to approximate the eddy viscosity at the online level. The chapter also dealt with a mathematical derivation for an efficient projection technique and hyper-reduction to circumvent the expensive projection-based method. In nutshell, the main objective of this chapter was to provide a general reduction methodology for FSI problems with moving boundaries for laminar, and turbulent flows when the Full-order Models (FOM) is based on the Reynolds Averaged Navier-Stokes (RANS) equations. This goal has been accomplished thanks to incorporating two data-driven techniques in the ROM formulation.

After having summarized the methodologies followed in this thesis, we proceed to the conclusions which can be drawn from the numerical tests conducted in Chapter 4. The summary of the numerical tests and the conclusions are reported in the following points

- The first one proposed in Section 4.2 is on the well-known flow passing an oscillating cylinder. The results confirm that the ROM is able to reproduce all response characteristics of the original system with acceptable accuracy, even with high cylinder displacement.

- In Section 4.3, we consider a flow passing a translation and rotational airfoil at a Reynolds number ($Re = 10^7$). The results confirm that the ROM is able to reproduce the wake dynamics of the flow passing a moving body and all the response characteristics of the system such as the lift and drag forces, amplitude/ frequency of the displacement, and the limit cycle of the non-linear dynamical system. The results show good convergence properties without any necessity of additional stabilization for what concerns pressure solutions. This is due to the fact that we used a segregated scheme, which tries to circumvent the saddle-point stability issue.

- In Section 4.4, a first step on the construction of hyper-reduction technique for segregated FSI problems addressed in this dissertation in order to hyper-reduce the expensive projection strategy firstly introduced. Also, results show that by selecting some optimal locations in the discretized domain, one can reconstruct the field of interest with good accuracy. In this case, for what regards the construction of the basis functions for the solution manifold covering, a greedy algorithm has been selected in place of the POD we selected for the two first test cases.

5.2 Outlooks and Perspectives

Finally, we suggest some possible future extensions of the work carried out in this thesis.

- The first perspective would be to implement the Algorithm 5 of the surrogate model with hyper-reduction technique.

- The second perspective would be to consider compressible flows with shocks and see if the reduced-order models (ROMs) will also capture the shocks present in the high fidelity problem;

- Another natural direction for future work will be that of adding an attention layer in the LSTM encoder-decoder model, to keep track of coefficient relationships within the inputs and output coefficients.

- Also an additional extension of this work will be the concept to divide and conquer by applying the reduced algorithm on the splitting domains as shown in Figure 6.5. This idea was already introduced in [6];

- The intersection of geometrical parametrization and fluid-structure interactions is a critical focus in computational engineering and design. By incorporating parametric models, we can achieve a more precise definition, analysis, and optimization of structures subjected to fluid forces. This approach not only enhances the accuracy of simulations but also enables the fine-tuning of designs for maximum efficiency and robustness. Such advancements are vital across various fields, from aerospace to biomedical engineering, where the ability to accurately simulate and optimize fluid-structure interactions leads to more innovative, effective, and resilient designs. One could consider this as a direction of research.

Chapter 6

Appendices

6.1 Machine learning of the temporal eddy viscosity coefficients

The main building blocks of the feed-forward NNs contain four layers: an input layer which is the *temporal vector coefficients of the velocity*, two hidden layers of 10 units each followed by an ELU (exponential linear unit) activation function, and the output layer which is the *temporal vector coefficients of the eddy viscosity.* The first 50% of the data in the time series is used for training (30%) and validation (20%) and the remaining for testing. The model is trained over 500 epochs.

For LSTM-RNNs, an LSTM Encoder-Decoder model is used for training, validation, and test on the data set. The architecture of such a LSTM-Encoder-Decoder model has one LSTM layer in the encoder part and another LSTM layer — in addition to one hidden layer of 5 units — in the decoder part. The reason for the choice is that this model has been used extensively for sequence-to-sequence prediction in the literature for NLP (natural language processing). The setup parameters of the LSTM are reported in Table 6.1.

6.2 Cylinder additional results

In this section, we provide additional results for the flow around the moving cylinder and some ideas for extension.

6.2.1 Synchronization analysis

The periodic state reached is characterized by the oscillation of the drag coefficient at twice ($f_{drag} \approx 2f_{sh}$) the lifting frequency [132] as one can see in the right plot

Parameters	Hidden size	learning rate	optimizer	LSTM layers	Batch size	Epochs	input dim	output dim
Value	1	1e-4	ADAM	1	5	500	5	5

Table 6.1: LSTM hyper-parameters

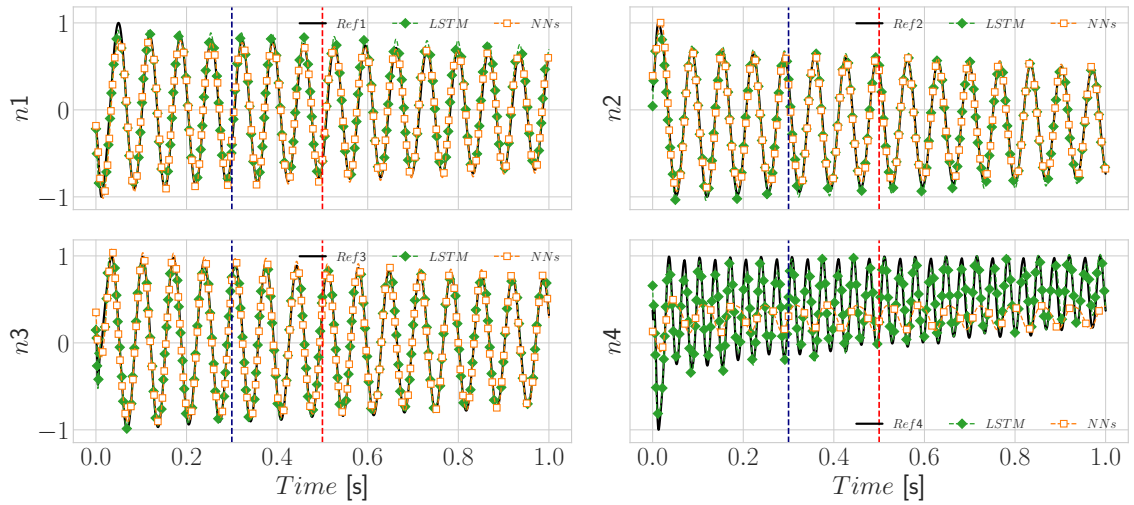


Figure 6.1: Training, validation, and test of the first four POD temporal coefficients of the eddy viscosity at the offline level using LSTM and NNs. The first half of the datasets (scaled between -1 and 1) is divided in two parts (30% for training and 20% for validation) for training and validation on both models. The remaining half is used for testing on both architectures.

of Figure 6.2. One of the most exciting characteristics of the fluid body interaction is that of synchronization, or "lock-in," between the vortex shedding and the cylinder vibration frequencies. When the wake is synchronized, the vortex-shedding frequency diverges from that corresponding to a fixed cylinder. It becomes equal to the frequency of the cylinder oscillation which is 0.185, as shown in Figure 6.2. So, our surrogate model is capable of reproducing the lock-in phenomenon.

6.2.2 Temporal coefficients analysis

In this subsection, we provide additional analysis of the efficiency of the predicted temporal coefficients. Figure 6.3 and Figure 6.4 provide a comparison between the predicted and reference coefficients.

6.2.3 Domain Filtering

In building ROMs or POD, the most accurate solution is typically generated by the most independent modes. For a large region of nearly constant flow where the perturbation solution tends to zero, large numbers of independent modes can generate numerical errors when trying to create a nearly freestream condition. An approach to dealing with this numerical error is to subdivide the computational domain. For the regions of highest interest (near field), more modes will be retained as depicted in Figure 6.5. Fewer modes will be required for intermediate field areas, which are of lower interest. In area of little change from the freestream (far field),

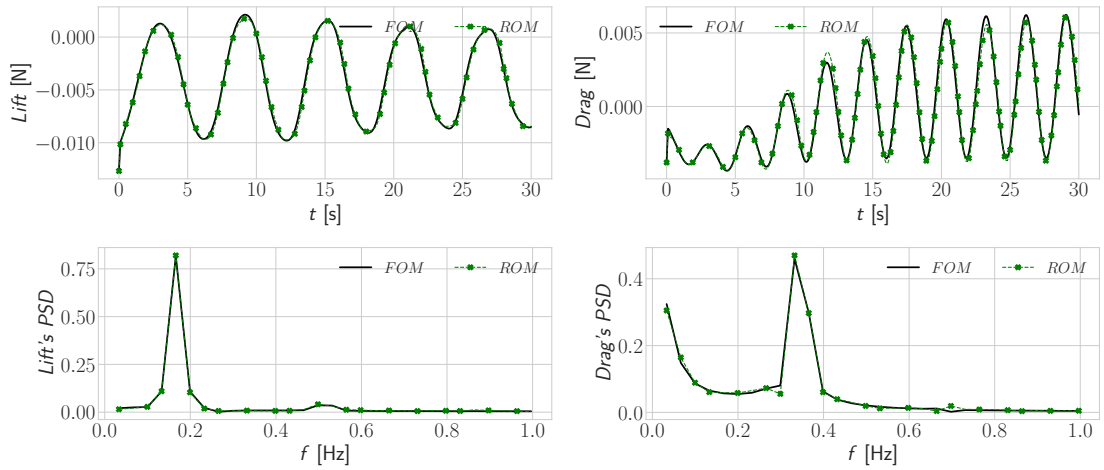


Figure 6.2: First row, from left to right: the time histories of the lift and drag forces. The solid black lines are the FOM curves, and the dashed green lines are the ROM curves obtained with 16 modes for the velocity and 21 modes for the pressure. Second row, from left to right: Power spectra density comparison of the lift and drag coefficients.

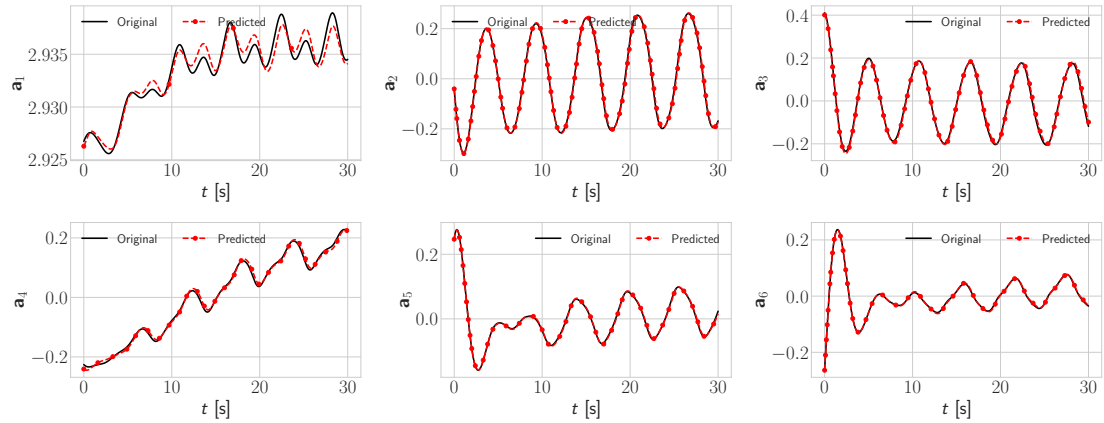


Figure 6.3: First six temporal coefficients of the velocity: black original signals and red predicted signals.

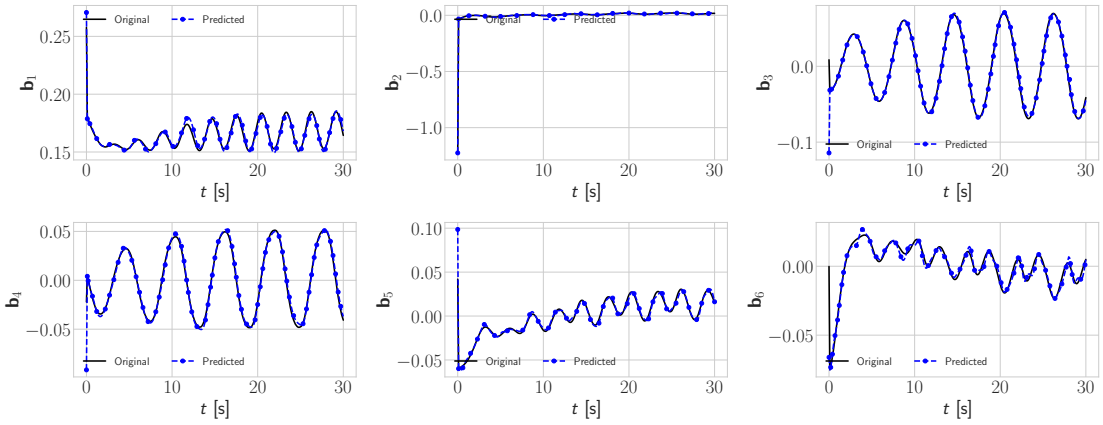


Figure 6.4: First six temporal coefficients of the pressure: black original signals and blue predicted signals

only two or three modes will be necessary. To implement this, a domain-based filtering of the POD modal matrix, $\mathbf{\Phi}$ has to be performed. For the grid-point locations to be filtered, the value of $\mathbf{\Phi}$ corresponding to the appropriate modes was set to zero. Thus, the number of modes used in various regions was truncated. In the near field, the number of modes varied depending on the complexity of the fluid flow. For both the intermediate and far fields, an adequate number of modes can be determined through trial and error. This domain filtering technique differs from the domain decomposition technique. The interested reader should refer to [6] for more insights.

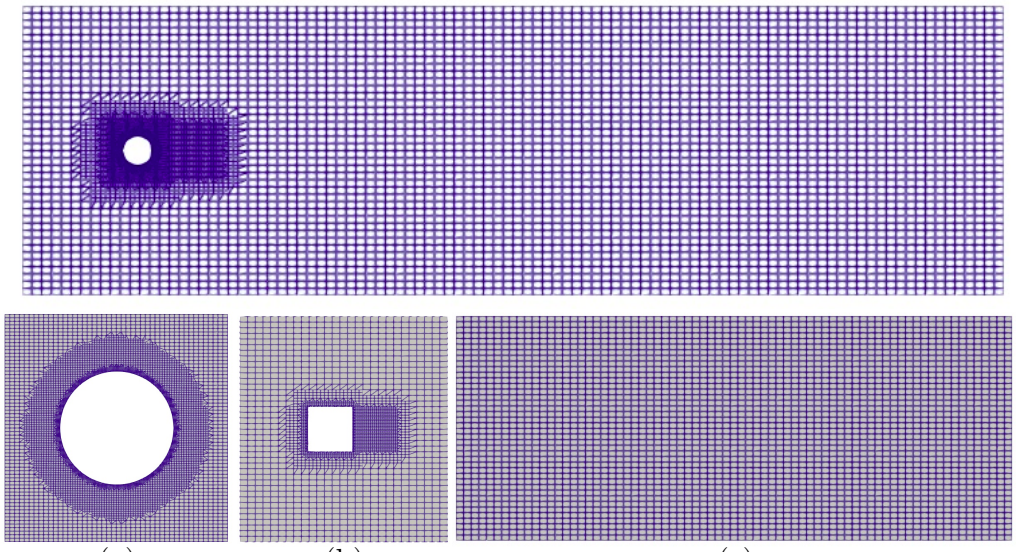


Figure 6.5: Domain filtering: first row initial mesh, second row: (a) region of highest interest (near mesh), (b) region of lower interest (intermediate and (c) region not affected by the mesh motion (far field).

Bibliography

1. T. Ø. Aanonsen. Empirical interpolation with application to reduced basis approximations. Master's thesis, Institutt for matematiske fag, 2009.

2. K. Academy. K21 academy - cloud computing and database management training, 2024. Accessed: 2024-07-24.

3. S. E. Ahmed and O. San. Breaking the kolmogorov barrier in model reduction of fluid flows. *Fluids*, 5(1):26, Feb. 2020.

4. D. Amsallem and C. Farhat. Stabilization of projection-based reduced-order models. *International Journal for Numerical Methods in Engineering*, 91(4):358–377, 2012.

5. J. Anttonen, P. King, and P. Beran. Pod-based reduced-order models with deforming grids. *Mathematical and Computer Modelling*, 38(1–2):41–62, July 2003.

6. J. S. Anttonen, P. I. King, and P. S. Beran. Applications of multi-POD to a pitching and plunging airfoil. *Mathematical and Computer Modelling*, 42(3-4):245–259, 2005.

7. J. S. R. Anttonen. *Techniques for reduced order modeling of aeroelastic structures with deforming grids*. Air Force Institute of Technology, 2001.

8. M. Balajewicz and E. H. Dowell. Stabilization of projection-based reduced order models of the navier–stokes. *Nonlinear Dynamics*, 70:1619–1632, 2012.

9. F. Ballarin, A. Manzoni, A. Quarteroni, and G. Rozza. Supremizer stabilization of pod–galerkin approximation of parametrized steady incompressible navier–stokes equations. *International Journal for Numerical Methods in Engineering*, 102(5):1136–1161, 2015.

10. F. Ballarin and G. Rozza. POD–Galerkin monolithic reduced order models for parametrized fluid-structure interaction problems. *International Journal for Numerical Methods in Fluids*, 82(12):1010–1034, 2016.

11. M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus. Mathématique*, 339(9):667–672, 2004.

12. Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

13. P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

14. P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

15. G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

16. S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64:525–545, 2019.

17. K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz. Deep model predictive control with online learning for complex physical systems. *arXiv preprint arXiv:1905.10094*, 2019.

18. D. Boffi, F. Brezzi, M. Fortin, et al. *Mixed finite element methods and applications*, volume 44. Springer, 2013.

19. B. Bonev, L. Prantl, and N. Thuerey. Pre-computed liquid spaces with generative neural networks and optical flow. *arXiv preprint arXiv:1704.07854*, 3, 2017.

20. G. Borrelli, L. Guastoni, H. Eivazi, P. Schlatter, and R. Vinuesa. Predicting the temporal dynamics of turbulent channels through deep learning, 2022.

21. R. Bourguet, M. Braza, and A. Dervieux. Reduced-order modeling of transonic flows around an airfoil submitted to small deformations. *Journal of Computational Physics*, 230(1):159–184, 2011.

22. S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52(1):477–508, 2020.

23. S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

24. M. Budišić, R. Mohr, and I. Mezić. Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4), 2012.

25. S. R. Bukka. *Data-driven computing for the stability analysis and prediction of fluid-structure interaction.* PhD thesis, National University of Singapore (Singapore), 2020.

26. S. R. Bukka, A. R. Magee, and R. K. Jaiman. Deep convolutional recurrent autoencoders for flow field prediction, 2020.

27. J. Burkardt, M. Gunzburger, and H.-C. Lee. POD and CVT-based reduced-order modeling of Navier–Stokes flows. *Computer methods in applied mechanics and engineering*, 196(1-3):337–355, 2006.

28. K. Carlberg, Y. Choi, and S. Sargsyan. Conservative model reduction for finite-volume models. *Journal of Computational Physics*, 371:280–314, 2018.

29. K. T. Carlberg, A. Jameson, M. J. Kochenderfer, J. Morton, L. Peng, and F. D. Witherden. Recovering missing cfd data for high-order discretizations using deep neural networks and dynamics learning. *Journal of Computational Physics*, 395:105–124, 2019.

30. K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

31. S. Chaturantabut and D. C. Sorensen. Discrete empirical interpolation for nonlinear model reduction. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 4316–4321. IEEE, 2009.

32. S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

33. R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

34. Y. Chen, S. Gottlieb, L. Ji, and Y. Maday. An EIM-degradation free reduced basis method via over collocation and residual hyper reduction-based error estimation. *Journal of Computational Physics*, 444:110545, 2021.

35. Y. Chen, S. Gottlieb, L. Ji, Y. Maday, and Z. Xu. L1-ROC and R2-ROC: L1- and R2-based Reduced Over-Collocation methods for parametrized nonlinear partial differential equations. *arXiv preprint arXiv:1906.07349*, 2019.

36. F. Chinesta, P. Ladeveze, and E. Cueto. A Short Review on Model Order Reduction Based on Proper Generalized Decomposition. *Archives of Computational Methods in Engineering*, 18(4):395–404, 2011.

37. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

38. P. Y. Chou. On velocity correlations and the solutions of the equations of turbulent fluctuation. *Quarterly of Applied Mathematics*, 3(1):38–54, 1945.

39. P. Choudhury, R. Allen, and M. Endres. Developing theory using machine learning methods. *Available at SSRN 3251077*, 2018.

40. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2022.

41. D. Coscia, N. Demo, and G. Rozza. Generative adversarial reduced order modelling. *Scientific Reports*, 14(1):3826, 2024.

42. M. H. Dao. Projection-based reduced order model for simulations of nonlinear flows with multiple moving objects. *arXiv preprint arXiv:2106.02338*, 2021.

43. M. H. Dao and H. H. Nguyen. Projection-Based Reduced Order Model and Machine Learning Closure for Transient Simulations of High-Re Flows. *arXiv preprint arXiv:2105.10854*, 2021.

44. F. Di Donfrancesco. *Reduced Order Models for the Navier-Stokes equations for aeroelasticity*. PhD thesis, Sorbonne Université, 2019.

45. G. Dimitriu, R. Ştefănescu, and I. M. Navon. Comparative numerical analysis using reduced-order modeling strategies for nonlinear large-scale systems. *Journal of Computational and Applied Mathematics*, 310:32–43, Jan. 2017.

46. M. Dobrowolski. On the lbb condition in the numerical analysis of the stokes equations. *Applied numerical mathematics*, 54(3-4):314–323, 2005.

47. J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran. Arbitrary l agrangian–e ulerian methods. *Encyclopedia of computational mechanics*, 2004.

48. A. Dullweber, B. Leimkuhler, and R. McLachlan. Symplectic splitting methods for rigid body molecular dynamics. *The Journal of chemical physics*, 107(15):5840–5851, 1997.

49. K. Duraisamy, G. Iaccarino, and H. Xiao. Turbulence modeling in the age of data. *Annual review of fluid mechanics*, 51(1):357–377, 2019.

50. K. Duraisamy, Z. J. Zhang, and A. P. Singh. New approaches in turbulence and transition modeling using data-driven techniques. In *53rd AIAA Aerospace sciences meeting*, page 1284, 2015.

51. J. L. Eftang and B. Stamm. Parameter multi-domain 'hp'empirical interpolation. *International Journal for Numerical Methods in Engineering*, 90(4):412–428, 2012.

52. H. Eivazi, L. Guastoni, P. Schlatter, H. Azizpour, and R. Vinuesa. Recurrent neural networks and koopman-based frameworks for temporal predictions in a low-order model of turbulence. *International Journal of Heat and Fluid Flow*, 90:108816, Aug. 2021.

53. H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian. Deep neural networks for nonlinear model order reduction of unsteady flows. *Physics of Fluids*, 32(10), Oct. 2020.

54. N. B. Erichson, M. Muehlebach, and M. W. Mahoney. Physics-informed autoencoders for lyapunov-stable fluid flow prediction. *arXiv preprint arXiv:1905.10866*, 2019.

55. A. Falaize, E. Liberge, and A. Hamdouni. POD-based reduced order model for flows induced by rigid bodies in forced rotation. *Journal of Fluids and Structures*, 91:102593, Nov. 2019.

56. B. A. Freno, N. R. Matula, R. L. Fontenot, and P. G. Cizmas. The use of dynamic basis functions in proper orthogonal decomposition. *Journal of Fluids and Structures*, 54:332–360, Apr. 2015.

57. K. Fukami, K. Fukagata, and K. Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

58. L. Garelli. *Fluid structure interaction using an arbitrary lagrangian eulerian formulation*. PhD thesis, INSTITUTO DE DESARROLLO TECNOLOGICO PARA LA INDUSTRIA QUIMICA, 2011.

59. T. B. Gatski and M. N. Glauser. Proper orthogonal decomposition based turbulence modeling. In *Instability, Transition, and Turbulence*, pages 498–510. Springer, 1992.

60. F. Gonzalez and M. Balajewicz. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. arxiv 2018. *arXiv preprint arXiv:1808.01346*, 1808.

61. B. Grimstad et al. SPLINTER: a library for multivariate function approximation with splines. http://github.com/bgrimstad/splinter, 2015. Accessed: 2015-05-16.

62. G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

63. R. Gupta and R. Jaiman. A hybrid partitioned deep learning methodology for moving interface and fluid-structure interaction. *Computers & Fluids*, 233:105239, 2022.

64. J. Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning. *Genetic Programming and Evolvable Machines*, 19(1-2):305–307, Oct. 2017.

65. J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, volume 590. Springer, 2016.

66. S. Hijazi, G. Stabile, A. Mola, and G. Rozza. Data-driven POD-Galerkin reduced order model for turbulent flows. *Journal of Computational Physics*, 416:109513, 2020.

67. M. Hirsch, F. Pichi, and J. S. Hesthaven. Neural empirical interpolation method for nonlinear model reduction, 2024.

68. D. H. Hodges and G. A. Pierce. *Introduction to structural dynamics and aeroelasticity*, volume 15. cambridge university press, 2011.

69. P. J. Holmes, J. L. Lumley, G. Berkooz, J. C. Mattingly, and R. W. Wittenberg. Low-dimensional models of coherent structures in turbulence. *Physics Reports*, 287(4):337–384, Aug. 1997.

70. H. E. Hurst. Long-term storage capacity of reservoirs. *Transactions of the American society of civil engineers*, 116(1):770–799, 1951.

71. C. Inc. Comsol, 2024.

72. R. I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of computational physics*, 62(1):40–65, 1986.

73. A. Ivagnes, G. Stabile, A. Mola, T. Iliescu, and G. Rozza. Hybrid data-driven closure strategies for reduced order modeling. *Applied Mathematics and Computation*, 448:127920, 2023.

74. H. Jasak. Error analysis and estimation for the finite volume method with applications to fluid flows. *PhD thesis*, 1996.

75. H. Jasak. OpenFOAM: Open source CFD in research and industry. *International Journal of Naval Architecture and Ocean Engineering*, 1(2):89–94, Dec. 2009.

76. H. Jasak, A. Jemcov, Z. Tukovic, et al. OpenFOAM: A C++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pages 1–20. IUC Dubrovnik Croatia, 2007.

77. B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer graphics forum*, volume 38, pages 59–70. Wiley Online Library, 2019.

78. D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2017.

79. E. Komen, A. Shams, L. Camilo, and B. Koren. Quasi-DNS capabilities of OpenFOAM for different mesh types. *Computers & Fluids*, 96:87–104, 2014.

80. M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

81. M. Korda and I. Mezić. On convergence of extended dynamic mode decomposition to the koopman operator. *Journal of Nonlinear Science*, 28:687–710, 2018.

82. K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical Analysis*, 40(2):492–515, Jan. 2002.

83. J. N. Kutz, X. Fu, and S. L. Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.

84. L. Ladickỳ, S. Jeong, B. Solenthaler, M. Pollefeys, and M. Gross. Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG)*, 34(6):1–9, 2015.

85. S. Lawrence. Turbulence and the dynamics of coherent structures. parts i-iii. *Q Appl Math*, 46(3):561, 1987.

86. D. Lazzaro and L. B. Montefusco. Radial basis functions for the multivariate interpolation of large scattered data sets. *Journal of Computational and Applied Mathematics*, 140(1):521–536, 2002. Int. Congress on Computational and Applied Mathematics 2000.

87. Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

88. K. Lee and K. T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

89. G. Lewin and H. Haj-Hariri. Reduced-order modeling of a heaving airfoil. *AIAA journal*, 43(2):270–283, 2005.

90. G. C. Lewin and H. Haj-Hariri. Reduced-order modeling of a heaving airfoil. *AIAA Journal*, 43(2):270–283, Feb. 2005.

91. E. Liberge, M. Benaouicha, and A. Hamdouni. Proper orthogonal decomposition investigation in fluid structure interaction. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, 16(3-4):401–418, 2007.

92. E. Liberge and A. Hamdouni. Reduced order modelling method via proper orthogonal decomposition (POD) for flow around an oscillating cylinder. *Journal of fluids and structures*, 26(2):292–311, 2010.

93. E. Liberge, M. Pomarede, and A. Hamdouni. Reduced-order modelling by pod-multiphase approach for fluid-structure interaction. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, 19(1-3):41–52, 2010.

94. Z. Liu, S. Kundu, L. Chen, and E. Yeung. Decomposition of nonlinear dynamical systems using koopman gramians. In *2018 Annual American Control Conference (ACC)*, pages 4811–4818. IEEE, 2018.

95. Z. Long, Y. Lu, and B. Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019.

96. E. Longatte, E. Liberge, M. Pomarede, J.-F. Sigrist, and A. Hamdouni. Parametric study of flow-induced vibrations in cylinder arrays under single-phase fluid cross flows using POD-ROM. *Journal of Fluids and Structures*, 78:314–330, Apr. 2018.

97. J. L. Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, pages 166–178, 1967.

98. J. L. Lumley. Stochastic tools in turbulence. volume 12. applied mathematics and mechanics. *Technical Report No. AD071031182: RISO-R-1653*, 1970.

99. B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

100. Y. Maday, N. C. Nguyen, A. T. Patera, and G. S. H. Pau. A general multipurpose interpolation procedure: the magic points. *Communications on Pure and Applied Analysis*, 8:383–404, 2008.

101. R. Maulik, A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, and D. Livescu. Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D: Nonlinear Phenomena*, 405:132368, 2020.

102. R. Maulik, O. San, A. Rasheed, and P. Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, 2019.

103. F. R. Menter, M. Kuntz, R. Langtry, et al. Ten years of industrial experience with the SST turbulence model. *Turbulence, heat and mass transfer*, 4(1):625–632, 2003.

104. I. Mezić. Analysis of fluid flows via spectral properties of the koopman operator. *Annual review of fluid mechanics*, 45(1):357–378, 2013.

105. M. Mifsud. *Reduced-order modelling for high-speed aerial weapon aerodynamics*. PhD thesis, Cranfield University, 2008.

106. T. Miyanawala and R. Jaiman. A low-dimensional learning model via convolutional neural networks for unsteady wake-body interaction. *arXiv preprint arXiv:1807.09591*, page 14, 2018.

107. T. Miyanawala and R. K. Jaiman. A hybrid data-driven deep learning technique for fluid-structure interaction. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 58776, page V002T08A004. American Society of Mechanical Engineers, 2019.

108. T. P. Miyanawala and R. K. Jaiman. Decomposition of wake dynamics in fluid–structure interaction via low-dimensional models. *Journal of Fluid Mechanics*, 867:723–764, 2019.

109. A. T. Mohan and D. V. Gaitonde. A Deep Learning based Approach to Reduced Order Modeling for Turbulent Flow Control using LSTM Neural Networks. *arXiv preprint arXiv:1804.09269*, 2018.

110. F. Moukalled, L. Mangani, and M. Darwish. *The finite volume method in computational fluid dynamics*, volume 113. Springer, 2016.

111. A. G. Mowat, A. G. Malan, L. H. van Zyl, and J. P. Meyer. Hybrid finite-volume reduced-order model method for nonlinear aeroelastic modeling. *Journal of Aircraft*, 51(6):1805–1812, 2014.

112. T. Murata, K. Fukami, and K. Fukagata. Nonlinear mode decomposition for fluid dynamics. *arXiv preprint arXiv:1906.04029*, 2019.

113. V. N. Ngan, G. Stabile, A. Mola, and G. Rozza. A reduced-order model for segregated fluid-structure interaction solvers based on an ale approach, 2023.

114. N.-C. Nguyen, A. T. Patera, and J. Peraire. A 'best points' interpolation method for efficient approximation of parametrized functions. *International journal for numerical methods in engineering*, 73(4):521–543, 2008.

115. V. Nkana Ngan, G. Stabile, A. Mola, and G. Rozza. A hybrid reduced-order model for segregated fluid-structure interaction solvers in an ALE approach at high Reynolds number. *arXiv e-prints*, pages arXiv–2406, 2024.

116. M. Nonino, F. Ballarin, and G. Rozza. A monolithic and a partitioned, reduced basis method for fluid–structure interaction problems. *Fluids*, 6(6):229, June 2021.

117. A. Nouy. A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1603–1626, 2010.

118. OpenFOAM Foundation. Openfoam: The open source cfd toolbox. https://www.openfoam.com, 2021. v2112.

119. S. E. Otto and C. W. Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

120. S. Pan and K. Duraisamy. Data-driven discovery of closure models. *SIAM Journal on Applied Dynamical Systems*, 17(4):2381–2413, 2018.

121. S. Pan and K. Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19(1):480–509, 2020.

122. R. Paoli. *Equations notes ME-518 "Fundamentals of turbulence"*. Mechanical Engineering Department, University of Illinois at Chicago, 2019.

123. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

124. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019.

125. S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In *Numerical prediction of flow, heat transfer, turbulence and combustion*, pages 54–73. Elsevier, 1983.

126. V. Patel, S. Parthasarathy, V. J. Shinde, and D. V. Gaitonde. Machine learning based model reduction for fluid-structure interaction. In *AIAA Scitech 2021 Forum*, page 1747, 2021.

127. V. V. Patel. *Reduced Order Modeling For Fluid-Structure Interaction Using Machine Learning*. PhD thesis, The Ohio State University, 2021.

128. A. Patera and G. Rozza. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations. version 1.0, 2006, copyright mit, to appear in (tentative rubric) mit pappalardo graduate monographs in mechanical engineering. *tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering*, 2006.

129. S. Peitz and S. Klus. Koopman operator-based model reduction for switched-system control of pdes. *Automatica*, 106:184–191, 2019.

130. R. Pigazzini, G. Contento, S. Martini, T. Puzzer, M. Morgut, and A. Mola. Viv analysis of a single elastically-mounted 2d cylinder: Parameter identification of a single-degree-of-freedom multi-frequency model. *Journal of Fluids and Structures*, 78:299–313, 2018.

131. A. Placzek. *Construction de modèles d'ordre réduit non-linéaires basés sur la décomposition orthogonale propre pour l'aéroélasticité*. PhD thesis, Conservatoire national des arts et metiers-CNAM, 2009.

132. A. Placzek, J.-F. Sigrist, and A. Hamdouni. Numerical simulation of an oscillating cylinder in a cross-flow at low Reynolds number: Forced and free oscillations. *Computers & Fluids*, 38(1):80–100, Jan. 2009.

133. J. L. Proctor, S. L. Brunton, and J. N. Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

134. C. Prud'Homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *J. Fluids Eng.*, 124(1):70–80, 2002.

135. M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

136. M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, 2019.

137. S. B. Reddy, A. R. Magee, and R. K. Jaiman. A data-driven approach for the stability analysis of vortex-induced vibration. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 51210, page V002T08A004. American Society of Mechanical Engineers, 2018.

138. O. Reynolds. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical Transactions of the Royal Society of London. (A.)*, 186:123–164, Dec. 1895.

139. C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–1532, Nov. 1983.

140. F. Romor, G. Stabile, and G. Rozza. Explicable hyper-reduced order models on nonlinearly approximated solution manifolds of compressible and incompressible navier-stokes equations, 2023.

141. F. Romor, G. Stabile, and G. Rozza. Non-linear manifold reduced-order models with convolutional autoencoders and reduced over-collocation method. *Journal of Scientific Computing*, 94(3), Feb. 2023.

142. F. Romor, G. Stabile, and G. Rozza. Non-linear manifold ROM with Convolutional Autoencoders and Reduced Over-Collocation method. *Journal of Scientific Computing*, 94(3), 2023.

143. S. Rudy, A. Alla, S. L. Brunton, and J. N. Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, 2019.

144. L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3):352–364, 2020.

145. H. Sak, A. Senior, and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

146. P. J. Schmid. Dynamic Mode Decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, July 2010.

147. M. Shademan, R. Barron, and R. Balachandar. Evaluation of openfoam in academic research and industrial applications. In *21st Conference of the CFD Society of Canada*, page 7, 2013.

148. Y. Sheng and T. Zhang. The finite volume method for two-dimensional burgers' equation. *Personal and Ubiquitous Computing*, 22(5):1133–1139, 2018.

149. Y. Sheng and T. Zhang. The finite volume method for two-dimensional burgers' equation. *Personal and Ubiquitous Computing*, 22(5–6):1133–1139, May 2018.

150. S. Shinde and M. Pandey. Modelling fluid structure interaction using one-way coupling and proper orthogonal decomposition (POD). *WIT Transactions on Engineering Sciences*, 105:27–35, 2016.

151. V. Shinde, E. Longatte, F. Baj, Y. Hoarau, and M. Braza. Galerkin-free model reduction for fluid-structure interaction using proper orthogonal decomposition. *Journal of Computational Physics*, 396:579–595, 2019.

152. H. Song, Y. Wang, and K. Pant. Parametric fluid-structural interaction reduced order models in continuous time domain for aeroelastic analysis of high-speed vehicles. In *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020.

153. D. B. Spalding. A novel finite difference formulation for differential expressions involving both first and second derivatives. *International Journal for Numerical Methods in Engineering*, 4(4):551–559, July 1972.

154. A. Srinivasan and P. Anand. Deep Learning models for turbulent shear flow, 2018.

155. G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, and G. Rozza. POD-Galerkin reduced order methods for CFD using Finite Volume Discretisation: vortex shedding around a circular cylinder. *Communications in Applied and Industrial Mathematics*, 8(1):210–236, 2017.

156. G. Stabile and G. Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier–Stokes equations. *Computers & Fluids*, 173:273–284, 2018.

157. G. Stabile and G. Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier–Stokes equations. *Computers & Fluids*, 173:273–284, 2018.

158. G. Stabile, M. Zancanaro, and G. Rozza. Efficient geometrical parametrization for finite-volume-based reduced order methods. *International Journal for Numerical Methods in Engineering*, 121(12):2655–2682, 2020.

159. W. Stankiewicz, M. Morzyński, R. Roszak, B. Noack, and G. Tadmor. Reduced order modeling of a flow around an airfoil with a changing angle of attack. *Archives of Mechanics*, 60(6):509–526, 2008.

160. W. Stankiewicz, R. Roszak, and M. Morzyński. Arbitrary lagrangian-eulerian approach in reduced order modeling of a flow with a moving boundary. In P. Reijasse, D. Knight, M. Ivanov, and I. Lipatov, editors, *Progress in Flight Physics*. EDP Sciences, June 2013.

161. S. K. Star, G. Stabile, F. Belloni, G. Rozza, and J. Degroote. A novel iterative penalty method to enforce boundary conditions in Finite Volume POD-Galerkin reduced order models for fluid dynamics problems. *Communications in Computational Physics*, 30(1):34–66, 2021.

162. N. Thuerey, K. Weißenow, L. Prantl, and X. Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

163. M. Tiglio and A. Villanueva. On the stability and accuracy of the empirical interpolation method and gravitational wave surrogates. *Classical and Quantum Gravity*, 38(13):135011, 2021.

164. M. Tiglio and A. Villanueva. Reduced order and surrogate models for gravitational waves. *Living Reviews in Relativity*, 25(1):2, 2022.

165. V. Troshin, A. Seifert, D. Sidilkover, and G. Tadmor. Proper orthogonal decomposition of flow-field in non-stationary geometry. *Journal of Computational Physics*, 311:329–337, Apr. 2016.

166. Y.-Y. Tsui, Y.-C. Huang, C.-L. Huang, and S.-W. Lin. A finite-volume-based approach for dynamic fluid-structure interaction. *Numerical Heat Transfer, Part B: Fundamentals*, 64(4):326–349, 2013.

167. J. H. Tu. *Dynamic mode decomposition: Theory and applications*. PhD thesis, Princeton University, 2013.

168. K. Veroy and A. T. Patera. Certified real-time solution of the parametrized steady incompressible navier–stokes equations: rigorous reduced-basis a posteriori error bounds. *International Journal for Numerical Methods in Fluids*, 47(8-9):773–788, 2005.

169. H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

170. J.-X. Wang, J.-L. Wu, and H. Xiao. Physics-informed machine learning for predictive turbulence modeling: using data to improve rans modeled reynolds stresses. *arXiv preprint arXiv:1606.07987*, pages 1041–4347, 2016.

171. Z. Wang, L. Du, J. Zhao, M. C. Thompson, and X. Sun. Flow-induced vibrations of a pitching and plunging airfoil. *Journal of Fluid Mechanics*, 885, Jan. 2020.

172. Z. Wang, D. Xiao, F. Fang, R. Govindan, C. C. Pain, and Y. Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

173. P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

174. M. J. Whisenant and K. Ekici. Galerkin-Free Technique for the Reduced-Order Modeling of Fluid-Structure Interaction via Machine Learning. In *AIAA Scitech 2020 Forum*, page 1637, 2020.

175. S. Wiewel, M. Becher, and N. Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pages 71–82. Wiley Online Library, 2019.

176. D. Wilcox. One-equation and two-equation Models. *Turbulence modeling for CFD*, 1:192–209, 2006.

177. J.-L. Wu, R. Sun, S. Laizet, and H. Xiao. Representation of stress tensor perturbations with application in machine-learning-assisted turbulence modeling. *Computer Methods in Applied Mechanics and Engineering*, 346:707–726, 2019.

178. D. Xiao, P. Yang, F. Fang, J. Xiang, C. C. Pain, and I. M. Navon. Non-intrusive reduced order modelling of fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 303:35–54, 2016.

179. Y. Xie, E. Franz, M. Chu, and N. Thuerey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018.

180. B. Xu, H. Gao, M. Wei, and J. Hrynuk. Global POD-Galerkin ROMs for Fluid Flows with Moving Solid Structures. *AIAA Journal*, pages 1–15, 2021.

181. W. Yao and R. Jaiman. Model reduction and mechanism for the vortex-induced vibrations of bluff bodies. *Journal of Fluid Mechanics*, 827:357–393, 2017.

182. K. Yeo and I. Melnyk. Deep learning algorithm for data-driven simulation of noisy dynamical system. *Journal of Computational Physics*, 376:1212–1231, 2019.

183. M. Zancanaro, M. Mrosek, G. Stabile, C. Othmer, and G. Rozza. Hybrid neural network reduced order modelling for turbulent flows with geometric parameters. *Fluids*, 6(8):296, 2021.

184. L. Zhu, W. Zhang, J. Kou, and Y. Liu. Machine learning methods for turbulence modeling in subsonic flows around airfoils. *Physics of Fluids*, 31(1), 2019.