POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Full order and reduced order models for the Navier-Stokes equations in stream function-vorticity formulation

Tesi di Laurea Magistrale in
Ingegneria Matematica

Author: **Gaia Buccino**

Student ID: 969290
Advisor: Prof. Nicola Parolini, Prof. Gialuigi Rozza
Co-advisors: Prof. Michele Girfoglio
Academic Year: 2021-22

Abstract

This work aims to perform an analysis based on the Reduced Order Methods (ROMs) constructed through a data-driven approach in the context of geophysical applications. This study has been developed starting from the derivation of the Full Order Model (FOM) of the Navier-Stokes Equations (NSE) in the alternative formulation based on the vorticity and stream function variables, whose equivalence with the original model described through velocity and pressure is verified in the vortex merger benchmark. Once the model has been validated, the FOM is simulated in order to generate the dataset required to perform the ROM analysis. The reduced coefficients associated with the two variables (vorticity and stream function) have been estimated using different variants of the method based on the combination between the Proper Orthogonal Decomposition (POD) and different data-driven techniques: (i) Radial Basis Functions (RBF) interpolation, (ii) Gaussian Process Regression (GPR) and (iii) Artificial Neural Networks (ANNs). We compared our results with the ones obtained in previous studies in terms of accuracy and efficiency. We found that all methods produce acceptable accuracy, with drastic speed-ups with respect to the FOM. However, we observed poorer estimations of the vorticity with respect to the stream function. Among the three ROM approaches, the POD-GPR and POD-RBF resulted in higher speed-ups, while the time required to train the POD-ANN model can make it unsuitable for real applications. Given the promising prediction results on the sample dataset, we extended the analysis to larger time horizons to test the extrapolation accuracy of the ROM prediction. This analysis revealed the high potential of data-driven methods for geophysical applications and could make significantly faster and more efficient large-scale predictions.

Keywords: Navier-Stokes equations, vorticity and stream functions, data-driven reduced order models, neural networks

Abstract in lingua italiana

Questo lavoro si propone di sviluppare un'analisi dei modelli ridotti (ROM) costruiti con un approccio data-driven nell'ambito di applicazioni geofisiche. Tale studio viene condotto partendo dalla derivazione di un modello full order (FOM) delle equazioni di Navier-Stokes nella formulazione alternativa basata sulle variabili vorticità e funzioni di corrente, la cui equivalenza con il modello originale di velocità e pressione viene dimostrata nel caso test del vortex merger. Una volta validato, il metodo introdotto viene utilizzato per produrre il dataset necessario all'analisi con modelli ridotti, in cui i coefficienti delle basi ridotte relativi alle due variabili (vorticità e funzione di corrente) vengono stimati attraverso la combinazione del metodo di decomposizione ortogonale (POD) e diversi metodi data-driven: (i) interpolazione con *radial basis functions* (RBF), (ii) regressione con processi gaussiani (GPR) e (iii) reti neurali artificiali (ANN). Abbiamo comparato i risultati ottenuti con quelli di studi precedenti in termini di accuratezza ed efficienza, scoprendo che tutti i metodi analizzati producono stime delle soluzioni con errori accettabili in un tempo notevolmente ridotto rispetto al modello full-order. Inoltre, abbiamo osservato che in termini di predizione, le stime ottenute per la vorticità sono peggiori di quelle ottenute per le funzioni di corrente. Confrontando i metodi utilizzati, abbiamo verificato infine che POD-GPR e POD-RBF producono buoni risultati in un tempo ragionevole, mentre il tempo richiesto per allenare la rete neurale la rende inadatta ad applicazioni reali. Dati comunque i risultati promettenti nella predizione del dataset analizzato, abbiamo esteso l'analisi ad un orizzonte temporale più ampio per verificare l'accuratezza nell'estrapolazione del comportamento della soluzione in un tempo futuro. Questa analisi ha rivelato il notevole potenziale dei metodi data-driven nelle applicazioni geofisiche e può migliorare le predizioni su larga scala rendendole piú rapide ed efficienti.

Parole chiave: equazioni di Navier-Stokes, vorticità e funzioni di corrente, modelli di ordine ridotto data-driven, reti neurali

Contents

Introduction

An open problem in the field of Computational Fluid Dynamics (CFD) is the exploration of alternative numerical methods to find the approximated solution to the Navier-Stokes Equations (NSE) [43]. In the latest years, many researchers address their efforts to find innovative techniques to reduce the computational cost of high-fidelity simulations while maintaining an acceptable level of accuracy. One of the possibilities to make computations feasible is represented by Reduced Order Methods (ROMs) [11, 12, 17, 18, 21, 25, 26, 32, 33, 38, 40, 46] that nowadays constitute a consolidated technique for the approximated resolution of parameterized Partial Differential Equations (PDEs). In particular, this research focuses on *non-intrusive* ROMs, i.e., a class of data-driven ROMs solely built starting from a dataset of parameters and high-fidelity solutions. This work, carried out in collaboration with *Scuola Internazionale Superiore di Studi Avanzati* (*SISSA*) of Trieste, explores the potentialities of non-intrusive ROMs to simulate geophysical and environmental phenomena where the goal is to describe the flow of either the air in the atmosphere or the water in the sea. In particular, starting from this general context, we analyze the problem using an alternative formulation, where the original variables, velocity and pressure, are replaced by the vorticity and the stream function[12]. This different approach is used to build an alternative Full Order Model (FOM), and is applied to compute the high-fidelity solutions of geophysical and atmospheric problems.

First of all, we derive the vorticity and stream function model from a theoretical point of view and discretize it in time and space. Next, we proceed with the implementation of a solver to address the problem in the new configuration and demonstrate that it is able to produce comparable results with respect to the original formulation. The vortex merger test case [36] is used as a benchmark to compare the performances of the above-mentioned solvers. Once the interchangeability of the two solvers has been verified, we integrate the alternative FOM in *ITHACA-FV* [38, 40], a finite-volume based library integrated with OpenFOAM. In particular, we follow a standard *offline-online* procedure. In the offline stage, we run the FOM simulations to collect the high-fidelity solutions, the so-called *snapshots*. Then, the Proper Orthogonal Decomposition (POD) technique is exploited to find an optimal orthonormal basis that captures the most relevant features, namely the

modes. The snapshots are then projected onto the space spanned by a reduced number of modes, which is *a priori* chosen. The estimation of the reduced coefficients associated with the reduced basis is done by making use of either interpolation or regression techniques. In particular, we adopt three different techniques for the coefficients' prediction: Radial Basis Functions (RBF) interpolation [22, 48], Gaussian Process Regressor (GPR) [8, 34, 35, 41] and Neural Networks (NNs) [7, 14, 17, 19, 37]. Then, we build the corresponding ROM in its variants POD-RBF, POD-GPR and POD-NN, and make a comparison between the performances of the three in terms of accuracy and CPU time. In the online stage, we exploit the above-mentioned models to compute the approximated solution corresponding to unknown values of the parameters under analysis and compare them to the corresponding ones directly simulated with the high-fidelity solver. In particular, we analyze the solutions predicted by the ROM in a time interval outside the one used for the simulations to see how far the model is able to extrapolate the evolution of the variables advancing in time. We finally quantify the accuracy of these predictions by validating them through the FOM and computing the trend of error in time.

The analysis carried out in this work is a comparison between the performance of the data-driven approach and the intrusive approach carried out in [12]. As a further development, we plan to expand this work to explore the behavior of the method when we insert in the model a stochastic variable describing the angle of an external wind. This additional parameter allows to include possible perturbations of the motion of the fluid.

The work presented is organized as follows:

- Chapter 1: Full Order Model
  After a brief introduction to the FOM, we describe the derivation and the discretization of the model and discuss its advantages and disadvantages.
- Chapter 2: Reduced Order Method
  In this chapter, we introduce different approximation methods for the non-intrusive ROM, in particular, we focus on the description of the POD-RBF, POD-GPR, POD-NN algorithms.
- Chapter 3: Numerical results
  This chapter is dedicated to the presentation of the obtained results.
- Chapter 4: Conclusions and future developments
  Here we present the conclusions of this work, its future developments and how we plan to proceed in the analysis.

Regarding the tools applied, we use *OpenFOAM*® [1], a widely used open-source Finite

Volume C++ library, to develop the code of the full order part, both for the implementation of the alternative solver and for its validation. To compute the full-order solutions, we use *ITHACA-FV* [38, 40], making the solver compatible with the structure of the library. For what concerns the ROM, we perform the computations using *EZyRB* [9], a Python package developed and maintained by the mathLab group at *SISSA*.

1 | Chapter 1: Full Order Model

This chapter is dedicated to the description and derivation of the Full Order Model (FOM) that we used to tackle the problem. In particular, starting from the Navier-Stokes equations in primitive variables [30], i.e., velocity and pressure (whose derivation is fully described in Appendix A), we investigated an alternative formulation [24, 27] described by the variables vorticity and stream function and analyzed the advantages and disadvantages that this approach introduces.

1.1. Derivation of the model

We started by considering the motion of a two-dimensional, incompressible, viscous fluid in a fixed domain $\Omega \subset \mathbb{R}^2$ over a time interval of interest $(t_0, T]$ described by the Navier-Stokes Equations:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \Delta \mathbf{u} + \nabla p = \mathbf{f}, & \text{(1.1a)} \\ \nabla \cdot \mathbf{u} = 0, & \text{(1.1b)} \end{cases}$$

where $\mathbf{u}$ and p are the unknowns of the model and represent the velocity and the pressure of the fluid (rescaled by the density), respectively, while ν is its kinematic viscosity.

We derived the alternative formulation, depending on vorticity and stream function, through the application of the operator $\nabla\times$ to the momentum equation (1.1a):

$$\partial_t \omega + \nabla \cdot ((\nabla \times \boldsymbol{\Psi})\omega) - \frac{1}{Re}\Delta\omega = \mathbf{F}, \tag{1.2}$$

where

- $\omega(x, y, t) = \nabla \times \boldsymbol{u} = (0, 0, \omega) = (0, 0, \partial_x v - \partial_y u)$,
- $\boldsymbol{\Psi}(\mathbf{x}, \mathbf{y}, \mathbf{t}) = (0, 0, \Psi)$,
- $\mathbf{u} = \nabla \times \boldsymbol{\Psi} = (\partial_y \Psi, -\partial_x \Psi)$.

Considering the definitions introduced above, and substituting component-wise the expression of $\boldsymbol{u}$ in (1.1b), we could express the link between the vorticity and the stream-function as a Poisson equation:

$$-\Delta \boldsymbol{\Psi} = \omega. \tag{1.3}$$

We highlight that the divergence-free constraint on the velocity, $\nabla \cdot \mathbf{u} = 0$, is automatically satisfied, and, therefore, there is no need to impose it as an additional constraint.
Combining together the two equations just derived, we obtained the following alternative system:

$$\begin{cases} \partial_t \omega + \nabla \cdot ((\nabla \times \boldsymbol{\Psi})\omega) - \frac{1}{Re}\Delta\omega = F & \text{in} \quad \Omega \times (t_0, T], \\ -\Delta \boldsymbol{\Psi} = \omega & \text{in} \quad \Omega \times (t_0, T], \end{cases} \tag{1.4}$$

Finally, to close the model, we had to impose the boundary and initial conditions. The first issue has been solved by endowing the system with homogeneous Dirichlet boundary conditions, both for ω and for $\boldsymbol{\Psi}$:

$$\begin{cases} \omega = 0 & \text{on} \quad \partial\Omega \times (t_0, T], \\ \boldsymbol{\Psi} = 0 & \text{on} \quad \partial\Omega \times (t_0, T], \end{cases} \tag{1.5}$$

the latter, by setting $\omega(x, y, t) = \omega_0(x, y)$, whose expression will be defined in the Section 3.1.1.

Once we outlined the configuration of the model, we could exploit the structure of the introduced variables.
Since we have defined $\omega(x, y, t) = \nabla \times \boldsymbol{u} = (0, 0, \omega) = (0, 0, \partial_x v - \partial_y u)$ and $\boldsymbol{\Psi}(\mathbf{x}, \mathbf{y}, \mathbf{t}) = (0, 0, \Psi)$ we could simplify our problem by considering two different scalar variables as unknowns, namely ω and Ψ, and restore their vectorial characterization only after the computations. This permitted us to solve the following system:

$$\begin{cases} \partial_t \omega + \nabla \cdot ((\nabla \times \Psi)\omega) - \frac{1}{Re}\Delta\omega = F & \text{in} \quad \Omega \times (t_0, T], \\ -\Delta \Psi = \omega & \text{in} \quad \Omega \times (t_0, T], \end{cases} \tag{1.6}$$

that is characterized by a remarkably reduced number of degrees of freedom, allowing us to significantly reduce computational costs.

1.1.1. Advantages and disadvantages of the alternative formulation

The alternative approach used to solve the Navier-Stokes Equations introduces some novelties in the resolution of the problem. First of all, it allowed us to treat the main unknowns as scalars. This appreciably reduces the computational cost, since it scales with the number of degrees of freedom. In particular, the CPU time required for the simulations is of 805 s for the original solver against 60 s for the alternative one. Another advantage is that the incompressibility constraint is automatically satisfied thanks to the definition of the stream function. On the other hand, it introduces some limitations on the adaptability of the model to complex geometrical configurations. In fact, in more complicated domains, the usage of the vorticity and stream function configuration can introduce some issues in the treatment of boundary conditions. This is the case of non-simply connected domains, where the presence of holes dramatically complicates the imposition of boundary conditions based on $\omega = \nabla \times \mathbf{u}$ and on $\mathbf{u} = \nabla \times \boldsymbol{\Psi}$. To overcome this issue, in the current work, we have chosen a very simple test case that will be described in detail in Chapter 3. For the sake of simplicity, we have imposed on it homogeneous Dirichlet boundary conditions, but the same simulations can be performed imposing non-homogeneous boundary conditions as well and re-adapting the lifting procedure to the two variables. Moreover, we have to specify that, even if this formulation is limited to the models expressed in 2D, it results particularly suitable to the description of meteorological and geophysical phenomena, since it simplifies dealing with vortical structures that abound in the atmosphere and in the ocean motion.

1.2. Discretization of the model

Once obtained the starting model, we have to address the discretization of the domain in which our problem has to be solved. The strategies chosen for the discretization are the finite differences approach [31] for the time and the finite volume discretization [20, 25, 39, 40] for the space.

1.2.1. Time discretization: Finite differences

We started with the discretization of the time derivative. We divided the time interval $(t_0, T]$ into $N_T \in \mathbb{N}$ smaller intervals of length $\Delta t = \frac{T-t_0}{N_T}$ and defined a generic time $t^n = t_0 + n\Delta t$ with $n = 0, ..., N_T$. Given $\omega = \omega^0$ and $\Psi = \Psi^0$, we look for $(\omega^{n+1}, \Psi^{n+1})$, i.e, variables computed at t^{n+1} that represents the solution of the problem (1.6) discretized

in time using the Backward Euler differentiation formula of order 1 (BDF1):

$$\begin{cases} \frac{1}{\Delta t}\omega^{n+1} + \nabla \cdot ((\nabla \times \Psi^{n+1})\omega^{n+1}) - \frac{1}{Re}\Delta\omega^{n+1} = b^{n+1} & \text{in} \quad \Omega \times (t_0, T], \\ -\Delta\Psi^{n+1} = \omega^{n+1} & \text{in} \quad \Omega \times (t_0, T], \end{cases} \tag{1.7}$$

where $b^{n+1} = F^{n+1} + \frac{\omega^n}{\Delta t}$.

This model is very complex from a computational point of view because of the presence of the non-linear term that encodes the coupling of the two variables. To overcome this issue, we decided to apply the segregated algorithm, namely to substitute the value of the stream function at time t^{n+1} in the first equation of the system with its extrapolation based on BDF1. This allowed us to get rid of the non-linear term in (1.7) and solve the equations one after the other as follows:

- Find ω^{n+1} such that

$$\frac{1}{\Delta t}\omega^{n+1} + \nabla \cdot ((\nabla \times \Psi^{n})\omega^{n+1}) - \frac{1}{Re}\Delta\omega^{n+1} = b^{n+1} \qquad in \quad \Omega \times (t_0, T]. \tag{1.8}$$

- Then, solve the second one for Ψ^{n+1}:

$$-\Delta\Psi^{n+1} = \omega^{n+1} \qquad in \quad \Omega \times (t_0, T]. \tag{1.9}$$

1.2.2. Space discretization: Finite Volume

To address the space discretization we decided to use the Finite Volume (FV) approach. It is related to the discretization of a 3D domain based on its partition into non-overlapping finite volumes, called control volumes (see an example in Figure 1.1). In each of these control volumes, we addressed the solution of our discretized system of partial differential equations. This is performed by integrating them over each considered volume and manipulating them in order to obtain an algebraic expression. For our problem, we considered a structured and orthogonal set of cells, whose advantages will be introduced below.

Let N_c be the total number of control volumes. For each control volume V_j, $j = 1, ...N_c$, we performed the following computations:

$$\frac{1}{\Delta t}\int_{V_j} \omega^{n+1} d\Omega + \int_{V_j} \nabla \cdot ((\nabla \times \Psi^{n})\omega^{n+1}) d\Omega - \frac{1}{Re}\int_{V_j} \Delta\omega^{n+1} d\Omega = \int_{V_j} b^{n+1} d\Omega. \tag{1.10}$$

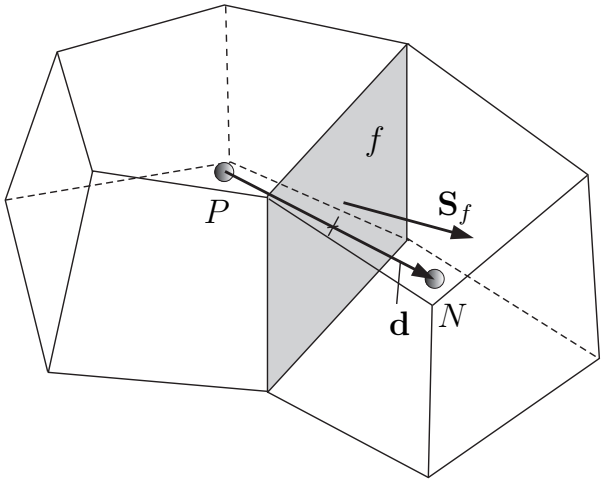


Figure 1.1: Example of Finite Volume discretization of a two-cells domain (image source [15]).

$$-\int_{V_j} \Delta\Psi^{n+1} d\Omega = \int_{V_j} \omega^{n+1} d\Omega. \tag{1.11}$$

Exploiting the Gauss divergence theorem, we could manipulate expressions (1.10) and (1.11) and obtain

$$\begin{cases} \frac{1}{\Delta t}\int_{V_j} \omega^{n+1} d\Omega + \int_{\partial V_j}((\nabla \times \Psi^n)\omega^{n+1}) \cdot d\mathbf{A_f} - \frac{1}{Re}\int_{\partial V_j} \nabla\omega^{n+1} \cdot d\mathbf{A_f} = \int_{V_j} b^{n+1} d\Omega, \\ -\int_{\partial V_j} \nabla\Psi^{n+1} \cdot d\mathbf{A_f} = \int_{V_j} \omega^{n+1} d\Omega, \end{cases}$$

where $\mathbf{A_f}$ represents the surface vector of each face i of the control volume under analysis.

After that, we could separately linearize each term of the equations as follows:

- *Mass term:* it remains defined as $\frac{1}{\Delta t}\int_{V_j} \omega^{n+1} d\Omega$.
- *Convective term:*

$$\int_{\partial V_j} ((\nabla \times \Psi^n)\omega^{n+1}) \cdot d\mathbf{A_f} \simeq \sum_f ((\nabla \times \Psi_f^n)\omega_f^{n+1}) \cdot \mathbf{A}_f = \sum_f (\varphi_f^n \omega_f^{n+1}), \tag{1.12}$$

  where $\varphi_f^n = \nabla \times \Psi_f^n \cdot \mathbf{A_f}$. This variable represents the convective flux of $\nabla \times \Psi^n$ through each face f of the control volume that can be computed as a linear interpolation using different schemes. In our case, we addressed this problem using the Central Differencing (CD) scheme that, for a generic variable Φ_f, corresponds to $\Phi_f = fx\Phi_P + (1 - fx)\Phi_N$ with $fx = \frac{fN}{PN}$, where fN corresponds to the distance between the face f and the centroid N, and PN is the distance between the two

centroids P and N, with reference to Figure 1.1.

- *Diffusive term:*

$$\int_{\partial V_j} \nabla\omega^{n+1} \cdot d\mathbf{A_j} \simeq \sum_f (\nabla\omega^{n+1})_f \cdot \mathbf{A}_f. \tag{1.13}$$

To deal with the approximation of the gradient normal to the face f, the hypothesis of structured and orthogonal mesh introduced before comes into play. In this context, it is possible to approximate $(\nabla\omega^{n+1})_f = \frac{\omega_N^{n+1} - \omega_P^{n+1}}{|\mathbf{d}|}$ where N and P refer to Figure 1.1 and $\mathbf{d}$ is the distance between P and N.

The same procedure has been used to discretize (1.11) obtaining:

$$\int_{\partial V_j} \nabla\Psi_{\cdot}^{n+1} d\mathbf{A_j} \simeq \sum_f (\nabla\Psi^{n+1})_f \cdot \mathbf{A}_f. \tag{1.14}$$

1.2.3. Fully discretized problem

Before considering the fully-discretized problem, we have to introduce some new notations. In particular, we denoted as ω_j^{n+1} the average value of the vorticity and B_j^{n+1} the average value of the source term in the control volume V_j. We introduced also $\omega_{j,f}^{n+1}$ to represent the vorticity associated with the centroid of the face f divided by the volume V_j. By doing so, we could write the fully discretized model as:

$$\begin{cases} \dfrac{1}{\Delta t}\omega_j^{n+1} + \displaystyle\sum_f \varphi_f^n \omega_{j,f}^{n+1} - \frac{1}{Re}\sum_f (\nabla\omega_f^{n+1})_j \cdot \mathbf{A}_f = b_j^{n+1}, & \text{(1.15a)} \\ -\displaystyle\sum_f (\nabla\omega_f^{n+1})_j \cdot \mathbf{A}_f = \omega_j^{n+1}. & \text{(1.15b)} \end{cases}$$

The solution of this model is addressed through the usage of two different linear solvers: the symmetric Gauss-Seidel method to solve the problem with respect to the vorticity and the Geometric Agglomerated Algebraic Multigrid (GAMG) solver for the stream function.

2 | Chapter 2: Reduced order methods

The full order model presented simulates a time-dependent multi-query problem, that in most cases, has to be repeated for a huge number of configurations depending on a generic parameter $\mu \in \mathbb{P}$. Usually, dealing with such problems using the FOM is unfeasible, because the full description of the solutions makes the computations dramatically involving, both in terms of required CPU time and memory demand, due to the large number of degrees of freedom. In this framework, the Reduced Order Method (ROM) [3–5] represents a valuable solution: starting from the original model, it builds a lower-dimensional one that is faster and cheaper to evaluate and, at the same time, provides accurate solutions. The procedure of the model reduction is divided into two phases, described as follows:

- **Offline phase:** During this phase, the high-fidelity simulations of the problem are performed for different values of the parameters under investigation. This step produces the matrix of snapshots (one column for each parameter). Once this first part is carried out using the FOM, the computation of the modes can be performed through the POD. This step consists of finding the set of the most informative basis to span a lower-dimensional space and projecting the matrix of snapshots onto it. It can be easily understood that the offline procedure is very expensive, but it has to be performed only once.
- **Online phase:** In this second phase, the reduced matrix resulting from the offline phase is used to compute the solutions corresponding to new values of the parameter with less computational effort.

Both the offline and the online phases can be addressed using different methods that are divided into two main categories: intrusive and non-intrusive. The first class is based on the manipulation of the FOM to implement the ROM in an analytical way, while the latter is based on a data-driven approach that, given the matrix of snapshots, aims at learning the reduced coefficients corresponding to each snapshot. In this work, the ROM is analyzed using the non-intrusive approach in its different variants: POD-RBF, POD-GPR,

and POD-NN. This means that we performed the reduced basis computation through the Proper Orthogonal Decomposition (POD) and approximated the reduced coefficients through three different methods: the Radial Basis Functions (RBF) interpolation, the Gaussian Process (GP) and the usage of Neural Networks (NN). In particular, the analysis proceeded with the evaluation of the predictive power of these models in view of future developments.

2.1. Reduced basis approximation

Let us start by analyzing the offline phase of the method. One of the main ingredients to understand the reduced basis approximation, which is the main issue addressed in the offline phase, is the concept of *solution manifold* that is the set of all solutions of the parameterized problem under the variation of the parameter considered. The offline phase is typically computed using the FOM.
As already specified, we addressed a parameterized problem, where the parameter μ is represented by the time t. The exact solution to (1.6) can be described by the generic expression $\mathbf{u} = \mathbf{u}(\mu)$, but in most cases, it is not available in an analytical way, so we had to discretize the model and compute $\mathbf{u_h}(\mu)$, we recall that we obtained it through finite volume approximation. From now on, we will refer to it as the *true solution*. Using the notations just introduced, we can define the solution manifold described above as

$$\mathbb{M} = \{\mathbf{u}(\mu), \mu \in \mathbb{P}\}, \tag{2.1}$$

in the case of the exact solution and as

$$\mathbb{M}_h = \{\mathbf{u_h}(\mu), \mu \in \mathbb{P}\}. \tag{2.2}$$

if we consider the true solution. We took on the approximation of the solution manifold of the discretized problem with the FOM based on the alternative formulation of the NSE explained in Chapter 1, obtaining the matrix of the snapshots. This matrix has dimension $N_d \times N_s$, where N_d represents the number of degrees of freedom involved in the computation, while N_s is the number of simulations (snapshots). Once we collect the matrix of snapshots, we can exploit it to compute the reduced space. This can be done with two different methods: the POD and the greedy algorithm. In the current work, we focused on the former, but we referred the interested reader to [18] for a detailed explanation of the latter.

2.1.1. Proper Orthogonal Decomposition (POD)

The POD method [2, 10, 16, 47, 49] is used to compute the reduced basis space as follows. First of all, we introduce a discrete and finite-dimensional point-set $\mathbb{P}_h = \{\mu_1, \dots, \mu_N\} \subset \mathbb{P}$ belonging to the space of the parameters. In our case, this space reduces to scalar values that represent the time. Depending on the chosen set of parameters, it is possible to construct a set of N_s snapshots $\{\mathbf{u}_h(\mu_1), \dots, \mathbf{u}_h(\mu_{N_s})\}$ and obtain the space generated by an approximation of the solution manifold related to the true solutions:

$$\mathbb{M}_{\{\mu_1,\dots,\mu_{N_s}\}} = \text{span}\{\mathbf{u}_h(\mu_1), \dots, \mathbf{u}_h(\mu_{N_s})\}. \tag{2.3}$$

The larger the number of snapshots, the more accurate is this approximation. If we consider the value of the true solutions corresponding to each value of the parameters considered, we can construct the matrix of snapshots:

$$S = \{\mathbf{u}_h(\mu_1)|\dots|\mathbf{u}_h(\mu_{N_s})\} \quad N_d \times N_s. \tag{2.4}$$

This matrix is typically rectangular, therefore, in order to factorize it, the Singular Valued Decomposition (SVD) is required. Its factorization is

$$S = W\Sigma V^T, \tag{2.5}$$

where $W = \{\mathbf{w}_1|\dots|\mathbf{w}_{N_d}\} \in N_d \times N_d$ and $V = \{\mathbf{v}_1|\dots|\mathbf{v}_{N_s}\} \in N_s \times N_s$ are two orthogonal matrices whose columns are respectively the left singular vectors and right singular vectors, while $\Sigma \in N_d \times N_s$ is a diagonal matrix with r non-zero ordered singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ where r represents the rank of the matrix S. The aim of the POD is to approximate the columns of S with a number of orthonormal vectors, called modes, much smaller than r. The Schmidt-Eckart-Young theorem [11] states that the reduced basis of dimension $L \leq r$ is composed by the first L left singular vector of the matrix S. So we can truncate the W matrix by taking into account only the first L columns and obtain the reduced basis

$$W_{rb} = \{\mathbf{w}_1|\dots|\mathbf{w}_L\}, \tag{2.6}$$

that generates the reduced space

$$\mathbb{W}_{rb} = \text{span}\{\mathbf{w}_1|\dots|\mathbf{w}_L\} \subset \mathbb{R}^L, \tag{2.7}$$

.

The chosen number of modes depends on the tolerance required to describe our problem since the error introduced in model reduction is quantified by the sum of the squares of the neglected singular values. Therefore, from a theoretical point of view, we can approximate the matrix of snapshots with arbitrary accuracy neglecting a different number of singular values. It is useful to observe that, if we take all singular values, then we recover the original matrix S.
Many problems show an exponential decay of the singular values allowing us to approximate the matrix of snapshots through the ROM with adequate accuracy.

Since the analysis carried out in [12] deeply investigated the intrusive approach based on the POD-Galerkin, we decided to study a different procedure based on a non-intrusive approach to compute the reduced basis and analyze the differences between FOM and ROM, both in terms of performance and CPU time required for the simulations.

2.1.2. Non-intrusive approach

By taking into account the analysis addressed in [12], we decided to explore the data-driven approach to compare the obtained results and analyze the advantages and disadvantages of the two different strategies. There are several methods to perform the ROM through the data-driven approach, in particular, we have tested the RBF interpolation, GPR and ANN.

Radial basis functions (RBF) interpolation

Radial basis function (RBF) interpolation [22, 48] is an effective method used to solve Navier–Stokes Equations in both two and three dimensions, since many studies have shown that this method can effectively capture the non-linearities of PDEs and provide a spectral convergence rate. A generic radial function $f : \mathbb{R}^P \longrightarrow \mathbb{R}$ is a multivariate function depending only on the norm of its arguments, i.e., $f = f(\|\mu\|)$. Since the main requirement of the chosen distance is that, defined a certain point μ_C, the function $f = f(\|\mu - \mu_C\|)$ is radially symmetric, the most common options of distance employed for this method are:

- *Gaussian*: $f(\mu) = e^{-\frac{\|\mu-\mu_C\|)}{\sigma^2}}$.
- *Linear spline*: $f(\mu) = \|\mu - \mu_C\|$.
- *Multiquadric*: $f(\mu) = \sqrt{\|\mu - \mu_C\|^2 + \sigma^2}$.
- *Inverse multiquadric*: $f(\mu) = \frac{1}{\sqrt{\|\mu-\mu_C\|^2+\sigma^2}}$.

- *Cubic spline*: $f(\mu) = \|\mu - \mu_C\|^3$.
- *Thin plate spline*: $f(\mu) = \|\mu - \mu_C\|^2 \log(\|\mu - \mu_C\|)$.

For the interpolation task, multiple radial basis functions are used: $f_m = f(\|\mu - \mu_m\|)$, with $1 \leq m \leq M$, where M is the number of different basis functions considered by shifting the center within the discretized parameter space $\mathbb{P}_h$. The idea behind the application of this method to the ROM is to approximate the vector of reduced coefficients component-wise using the radial functions as basis:

$$u_{r,k}(\mu) = \bar{\pi}_k(\mu; w_k) = \sum_{m=1}^{M} (w_k)_m f(\|\mu - \mu_m\|). \tag{2.8}$$

The estimated coefficients of the expansion could be computed by imposing the exact interpolation of the training data:

$$\bar{\pi}_k(\mu; w_k) = u_{r,k}(\mu_m) \qquad 1 \leq m \leq M. \tag{2.9}$$

By substituting the last expression in equation (2.8) we obtained:

$$Aw_k = u_{r,k}, \quad \text{where} \quad \begin{cases} A_{(i,j)} = f(\|\mu_i - \mu_j\|) & 1 \leq i, j \leq M, \\ (u_{r,k})_j = u_{r,k}(\mu_j) & 1 \leq j \leq M, \end{cases} \tag{2.10}$$

Finally, we had to solve n systems (one for each component of the reduced solution u_r), but, as shown in (2.10), the matrix $A \in \mathbb{R}^{M \times M}$ and its decomposition must be computed only once, making the resolution computationally efficient.

We can then collect all the weights in a unique matrix $W = [\mathbf{w}_1 | \dots | \mathbf{w}_n] \quad \in \mathbb{R}^{M \times n}$ and write the map of the reduced coefficients as:

$$\pi(\mu) = \mathbf{W}^T \mathbf{F}(\mu), \quad \text{where} \quad (\mathbf{F}(\mu))_m = f(\|\mu - \mu_m\|) \qquad 1 \leq m \leq M. \tag{2.11}$$

Gaussian Process Regression (GPR)

A Gaussian Process (GP) is a widely used technique to address regression tasks based on the Bayesian approach. The main idea behind the GP regression [8, 34, 35, 41] is the attempt to learn the data generating distribution starting from the punctual values of the underlying (unknown) function at any point of its domain. This strategy leads to a flexible model that allows to capture any function that interpolates the available dataset. In the case of ROMs, considered a fixed point $\mu* \in \mathbb{P}$, our goal is to estimate the k-th

coefficient $u_{r,k}(\mu*)$ of the reduced solution $u_r(\mu*)$ through the GPR technique. To do this, we can consider the regression function $f : \mathbb{P} \longrightarrow \mathbb{R}$ as a scalar function whose distribution is Gaussian: $f(\mu^*) \sim \mathcal{N}(m(\mu^*), \sigma^2)$.
Collecting a finite number of inputs into a vector $\mathbf{x} = [\mu_1, \dots, \mu_M]$ we can obtain the resulting output expressed by M different Gaussian distributions. However, since all the outputs are represented by the values of the same function evaluated at different points, these distributions must necessarily be correlated to each other. Therefore, if we consider a value in the parameter space for which the function assumes a certain value with relative uncertainty, we expect that small perturbations of the input will correspond to a limited change in both the output and the uncertainty value. This means that the input $\mathbf{x}$ is associated with a multivariate distribution for the output $\mathbf{y} = [f(\mu_1), \dots, f(\mu_M)]$ that is

$$\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}), \quad \text{where} \quad \begin{cases} (\mathbf{m})_{\mathbf{i}} = \mathbb{E}[\mathbf{y_i}], \\ \mathbf{K}_{(\mathbf{i},\mathbf{j})} = \mathbb{E}[(y_i - m_i)(y_j - m_j)], \end{cases} \tag{2.12}$$

where $\mathbf{m}$ is the mean vector and $\mathbf{K}$ represents the covariance matrix. All these considerations lead to define the regression function as a GP:

$$f(\mu) \sim \mathcal{GP}(m(\mu), k(\mu, \mu')) \qquad \forall \mu, \mu' \in \mathbb{P}. \tag{2.13}$$

More precisely, a GP is a stochastic process $f(\mu)$ in which any finite number of random variables taken from the random process has a joint Gaussian probability distribution. Equation (2.13) represents a distribution over functions for which, as for any regular Gaussian distribution, we specify a mean and a covariance. However, since we are dealing with infinite dimensions both of the vector and of the covariance matrix, $\mathbf{m}$ is replaced by the mean function $m(\cdot)$ and $\mathbf{K}$ is replaced by the two-dimensional kernel covariance function $k(\cdot, \cdot)$ such that:

$$\begin{cases} m(\mu) = \mathbb{E}[f(\mu)], \\ k(\mu, \mu') = \mathbb{E}[(f(\mu) - m(\mu))(f(\mu') - m(\mu'))]. \end{cases} \tag{2.14}$$

In particular, considering (2.12), we can observe that the parameters involved are specific realizations of a GP, over a finite subset of inputs. Namely, if we consider a vector $\mathbf{x_{tr}} = [\mu_1, \dots, \mu_M]$ containing all elements of the *training* set $\mathbb{P}_h$, we obtain:

$$\begin{cases} \mathbf{m} = m(\mathbf{x_{tr}}) := [m(\mu_i)]_{1 \le i \le M} \in \mathbb{R}^M, \\ \mathbf{K} = k(\mathbf{x_{tr}}, \mathbf{x_{tr}}) := [k(\mu_i, \mu_j)]_{1 \le i,j \le N} \in \mathbb{R}^{M \times M}. \end{cases} \tag{2.15}$$

In order to tackle the regression task, we have to choose a prior distribution for both $m(\cdot)$ and $k(\cdot,\cdot)$. The former is usually taken equal to zero, $m(\mu) = 0$, while for the latter there are different options. Two examples of covariance functions widely used are:

- *squared exponential* $k(\mu,\mu') = \sigma_f^2 \exp\{-\frac{1}{2\ell^2}\|\mu_i - \mu_i'\|^2_{\mathbb{R}^P}\}$,
- *squared exponential kernel* $k(\mu,\mu') = \sigma_f^2 \exp\{-\sum_{i=1}^{P} \frac{(\mu_i-\mu_i')^2}{2\ell^2}\}$,

where two different hyper-parameters have been introduced: σ_f is the standard deviation that controls the uncertainty in the vertical direction and ℓ represents the correlation length scale, namely how far we need to move (along a particular axis) in the input space for the function values to become uncorrelated. The choice of these hyper-parameters strongly influences the performance of the model and has to be tuned depending on the covariance kernel function used. We refer to [42], [23] for further details.
To carry out the regression, namely to adapt the probability distribution given by the GP to new observations, we have to compute the probability of this function conditioned by new observations

$$\mathbb{P}(f(\cdot)|\mathbf{x_{tr}}, \mathbf{y_{tr}}) = \mathcal{GP}(\bar{f}(\cdot), k'(\cdot,\cdot)), \tag{2.16}$$

where $\bar{f}(\cdot)$ and $k'(\cdot,\cdot)$ are the corrected mean and covariance respectively given by:

$$\begin{cases} \bar{f}(\cdot) = k(\cdot, \mathbf{x_{tr}})\mathbf{K}^{-1}\mathbf{y_{tr}}, \\ k'(\cdot,\cdot) = k(\cdot,\cdot) - k(\cdot, \mathbf{x_{tr}})\mathbf{K}^{-1}k(\mathbf{x_{tr}}, \cdot). \end{cases} \tag{2.17}$$

Since the main aim is to use this method for prediction purposes, we can collect a set of *testing* parameters in the vector $\mathbf{x_{pr}}$ and compute the expected value of the corresponding output we want to predict, $\mathbf{y_{pr}}$, namely the value of $m(\mathbf{x_{pr}})$. Combining (2.17) with the properties of the $\mathcal{GP}s$, we can evaluate the mean and covariance of the posterior distribution $\mathbf{y_{pr}}|\mathbf{x_{pr}}, \mathbf{x_{tr}}, \mathbf{y_{tr}} \sim \mathcal{N}(\mathbf{m_{pr}}, \mathbf{K'_{pr}})$ as:

- $\mathbf{m_{pr}} = \bar{f}(\mathbf{x_{pr}}) = k(\mathbf{x_{pr}}, \mathbf{x_{tr}})\mathbf{K}^{-1}\mathbf{y_{tr}}$,
- $\mathbf{K'_{pr}} = k(\mathbf{x_{pr}}, \mathbf{x_{pr}}) - k(\mathbf{x_{pr}}, \mathbf{x_{tr}})\mathbf{K}^{-1}k(\mathbf{x_{tr}}, \mathbf{x_{pr}})$.

The procedure just described, allowed us to construct component-wise the regression map $\pi_{GP} : \mathbb{P} \longrightarrow \mathbb{R}^n$, where the k-th entry is given considering $\mathbf{x}_{tr} = [\mu]_{\mu\in\mathbb{P}_h}$ and $\mathbf{y}_{tr} = [u_{r,k}(\mu)_{\mu\in\mathbb{P}_h}]$ as training data. Finally, using the mean of the M independent distributions obtained with the procedure just described, we are able to predict the reduced coefficients at new parameter values.

Artificial Neural Networks (ANN)

The Artificial Neural Network (ANN) [7, 14, 17, 19, 37], or simply Neural Network (NN), is a model that learns information from observational data, offering an alternative to the algorithmic programming paradigm. It consists of artificial neurons connected by a set of directed weighted synapses that can be represented by an oriented graph, with neurons as nodes and "synapses" as oriented edges, whose weights are tuned by means of a training process to specialize the network for a specific application. A representation of a neuron, that is the fundamental unit of a neural network, is shown in Figure 2.1.

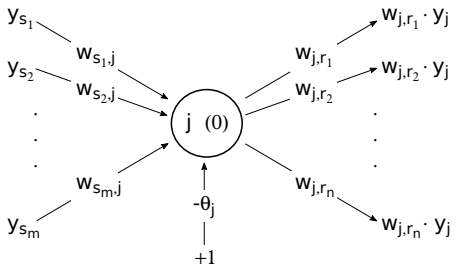


Figure 2.1: Scheme of a neuron: the fundamental unit of the NN (image source [17]).

As deeply explained in [17], each neuron j in the network is connected with m sending neurons $\{s_1, \dots, s_m\}$ from which receives different inputs. These latter must be elaborated by the neuron through the combination of three different functions, the *propagation function*, the *activation function* and the *output function* to produce an output. The role of each of the three functions, that fully characterize the action of each neuron, is clarified below:

Propagation function: converts the input received by the sending neurons $[y_1, \dots, y_{s_m}]$ to a scalar $u_j = f_{prop}(w_{s_k,j}, y_{s_k})$. The most common choice for f_{prop} is the weighted sum, namely $u_j = \sum_{k=1}^{m} w_{s_k,j} y_{s_k}$.

Activation function: at each time step, it prescribes to which degree the neuron j is active. This is specified by the activation state $a_j = f_{act}(u_j; \theta_j)$, where θ_j represents a threshold parameter of the network chosen in the training phase. To properly treat θ_j, it is common to introduce a bias neuron in the network, with constant output $y_b = 1$, which is directly connected with neuron j, assigning the bias weight $w_{b,j} = -\theta_j$ to the connection. So, θ_j is treated as a weight during the training phase of the network obtaining new expressions for $u_j = \sum_{k=1}^{m} w_{s_k,j} y_{s_k} - \theta_j$ and $a_j = f_{act}(u_j)$. There are different functions that can be used as activation functions, typically non-linear. Some examples are *Tanh*, *Sigmoid*, *ReLU*, *Linear*, *SoftMax*.

Output function: it evaluates the scalar output y_j based on the activation state a_j of the neuron, $y_j = f_{out}(a_j)$. Usually, it is the identity function, so that activation and output of a neuron coincide.

After selecting the characterization of neurons, it remains to define the topology of the network, namely how the neurons are interconnected. Among all the possible architectures in the literature, the feed-forward neural network is the one preferred to address regression tasks. Its peculiarity is that neurons are arranged into layers: the input layer made of M^I neurons, a variable number of hidden layers H and an output layer made of M^O neurons. In this kind of network, the neurons belonging to a particular layer can only communicate to neurons belonging to the next layer and so on towards the output layer. Therefore, basically, the neural network constructs a non-linear map π between the input space and the output space (that encodes the role of hidden layers)

$$\pi : \mathbb{M}^I \mapsto \mathbb{M}^O. \tag{2.18}$$

This map-like behavior makes the feed-forward neural networks particularly suitable for continuous function approximation. Depending on the number of hidden layers, we can distinguish two different categories of feed-forward neural networks: *Single-Layer Perceptrons (SLPs)* and *Multi-Layers Perceptrons (MLPs)*. Since the former are suitable to handle problems based on linearly separable data, we focused on networks belonging to the second category. Another important parameter that has to be tuned is the number of neurons belonging to each layer. There are no clues about how to choose the optimal number of neurons so we tackled this issue by testing different combinations and making decisions based on errors. An example of MLP with three hidden layers, with five neurons

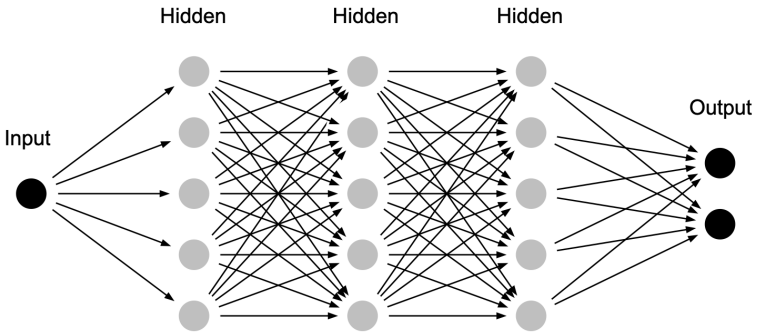


Figure 2.2: Example of a MLPs neural network with three hidden layers (image source [28]).

each, is shown in Figure 2.2. The role of the network we have just described is to compute the reduced coefficients, minimizing the loss function on the training set, computed

through the Mean Squared Error (MSE) between labels (real value of the training point) and outputs. In practice, since with the reduced basis we can describe each vector in the reduced space $\mathbf{V}_{rb}$ as

$$\mathbf{v}_{rb} = V\alpha = \sum_{j=1}^{L} \mathbf{v}_{rb}^{j}\mathbf{w}_j = \sum_{j=1}^{L} \mathbf{v}_{rb}^{j} \sum_{i=1}^{N_h} V_{i,j}\phi_i = \sum_{i=1}^{N_h} (V\mathbf{v}_{rb})_i\phi_i, \tag{2.19}$$

where $\{\phi_1, \ldots, \phi_{N_h}\}$ is a basis of $\mathbb{V}_h$, we want to use the network to approximate α, the vector of reduced coefficients.

We recall that the projection of a vector $\mathbf{u_h} \in \mathbb{V}_h$ on the space $\mathbb{V}_{rb}$ corresponds to the projection of the same vector onto $col(V)$, where V is the matrix that has as columns a basis of $\mathbb{V}_{rb}$. Since VV^T is the projection matrix and can be written as $VV^T = \sum_i \langle \mathbf{w}_i, \cdot \rangle \mathbf{w}_i$, the reduced solution $\mathbf{u}_{rb}$ can be expressed as:

$$\mathbf{u}_h \simeq \mathbf{u}_{rb} = VV^T\mathbf{u}_h = \sum_{i=1}^{N_h} (VV^T\mathbf{u}_h)_i\phi_i = \sum_{j=1}^{L} (V^T\mathbf{u}_h)_j\mathbf{w}_j. \tag{2.20}$$

Moreover, considering that $\mathbf{u_h} = \mathbf{u_h}(\mu)$, the aim becomes to approximate the function:

$$\pi : \mu \in \mathbb{P} \longrightarrow V^T\mathbf{u}_h \in \mathbb{R}^L, \tag{2.21}$$

with a neural network.
This allows the computation of new instances during the online phase as $V\bar{\pi}(\mu)$ where the $\bar{\pi}$ represents the estimated function.
The network we needed is characterized by an input layer of the same dimension on the parameter vector (in our case 1 since the time is a scalar), h hidden layers whose value has to be decided and an output layer of dimension given by the function $\bar{\pi}(\mu)$ that in our case is the number of degrees of freedom of the FOM solution.

2.1.3. Intrusive approach: hints

To complete the discussion about the ROM, we briefly introduce also the analytical non-intrusive approach.
In a context in which we know the model generating the snapshots, it is natural to address the ROM using an intrusive approach, as done in [12]. This method consists of implementing the projection of the algebraic model onto the reduced space, previously computed using one of the two methods already explained, POD or greedy algorithm. In particular, having the algebraic expression of the governing equations, the reduced model

can be analytically computed through the Galerkin projection. This method consists of solving at t^{n+1} the following system for β^{n+1} and γ^{n+1}:

$$M_r(\frac{\beta^{n+1}-\beta^n}{\Delta t}) + (\gamma^n)^T G_r \beta^{n+1} - \frac{1}{Re} A_r \beta^{n+1} = 0, \tag{2.22}$$

$$B_r \gamma^{n+1} + \tilde{M}_r \beta^{n+1} = 0, \tag{2.23}$$

where β^n and γ^n are two vectors containing the coefficients of the solutions with respect to the reduced basis function φ and ξ and the matrices involved encode the differential operators as follows:

- Mass matrix for ω: $M^W_{r_{ij}} = (\varphi_i, \varphi_j)_{L^2}$,
- Diffusion matrix for ω: $A_{r_{ij}} = (\varphi_i, \Delta\varphi_j)_{L^2}$,
- Mass matrix for Ψ: $M^\Psi_{r_{ij}} = (\xi_i, \xi_j)_{L^2}$,
- Diffusion matrix for Ψ: $B_{r_{ij}} = (\xi_i, \Delta\xi_j)_{L^2}$,
- Non-linear term: $G_{r_{ijk}} = (\varphi_i, \nabla \cdot ((\nabla \times \xi_j)\varphi_k)_{L^2})$.

3 | Chapter 3: Numerical results

Before discussing the results obtained, we briefly recall the leitmotif of our analysis. We derived an alternative formulation of the Navier-Stokes equations based on the variables ω and $\mathbf{\Psi}$ (1.4) to tackle the description of the motion of a fluid in a 2D, discretized it and validated it by comparing its computed solutions with those obtained through the original implemented solver *icoFoam*. Then, we integrated this model into the ITHACA-FV library [38, 40] to perform the offline phase of the reduced order method that made us construct the matrix of snapshots required to the proper application of the ROM. Actually, once we obtained the dataset of the full order simulations, we used it to train a neural network to perform the POD and reconstruct the original solutions starting from the reduced ones. Moreover, we explored the capability of this model to predict instances related to new values of the parameter and compare them with the corresponding one obtained through the FOM. In the following, we reported all results obtained.

3.1. Validation of the vorticity - stream function solver

After the derivation of the $\omega - \mathbf{\Psi}$ model and its discretization through the finite difference scheme BDF1 in time and the finite volume method in space, we addressed the computations on the widely used vortex merger benchmark, where we also performed the same simulations using the $\mathbf{u} - p$ solver. By doing so, we could compare the obtained solutions and demonstrate the equivalence between the two formulations.

3.1.1. Test case: vortex merger

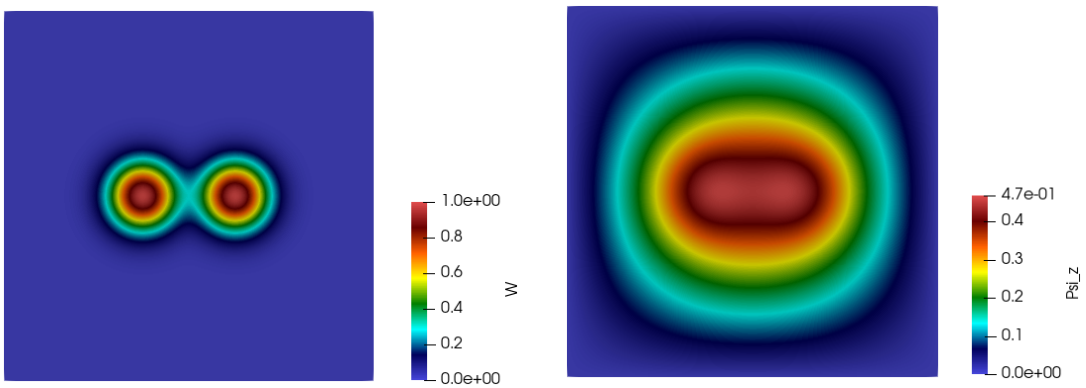
The vortex merger benchmark on which we tested the models consists of two co-rotating vortices of the same sign at a certain distance that evolve into a single one [36]. It represents a typical initial condition for astrophysical, meteorological, and geophysical phenomena.

Geometry and computational settings

The geometry of the case under consideration is very simple. It consists of a $2\pi \times 2\pi \times 0.2$ volume discretized with a grid made of $256 \times 256 \times 1$ elements. This is a 3D domain that can be considered as a 2D domain since we neglect the variation of the solution in the z-direction.
On this mesh, we set as boundary conditions the already prescribed Dirichlet homogeneous boundary conditions (1.5), we enforced $F = 0$ for sake of simplicity and, as initial conditions, we imposed $\omega = \omega_0$ with

$$\omega(x, y, 0) = \omega_0(x, y) = \sum_{i=1}^{2} e^{-\pi[(x-x_i)^2+(y-y_i)^2]}. \tag{3.1}$$

In the expression above, (x_i, y_i) represents the initial location of each vortex i: $(x_1, y_1) = (\frac{3\pi}{4}, \pi)$ and $(x_2, y_2) = (\frac{5\pi}{4}, \pi)$. In Figure 3.1 we report the representation of the initial condition $\omega = \omega_0$ and of the condition for Ψ_0, while Figure 3.2 shows the computed $\mathbf{u}$ at t = 0.



(a) ω at t = 0 (b) Ψ at t = 0

Figure 3.1: Initial conditions of the variables ω and Ψ.

3.1.2. Results and comments

Once the initial condition has been imposed, we simulated the problem setting the starting time to 0 s, the time step to 0.01 s and the end time to 20 s. Letting the system evolve, we obtained the computed solutions for the variables under analysis: ω, Ψ, u. In order to verify if the solver just implemented is equivalent to *icoFoam*, the one already implemented in OpenFOAM, that addresses the solution of the Navier-Stokes equation in the original

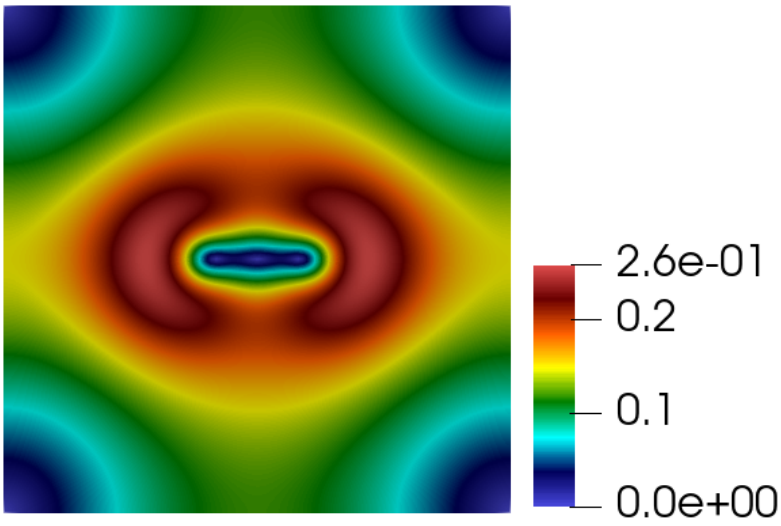


Figure 3.2: Initial value of the magnitude of $\mathbf{u}$ computed as $\nabla \times \boldsymbol{\Psi}$.

$\mathbf{u} - p$ formulation, we run the same simulation using the original variables. The main issue we had to face when performing the second computation was the setting of the initial condition: we had to describe the vortex merger introduced in the first case in terms of the velocity. To do so, since the initial value for Ψ is set to zero and this prevents us to use it to reconstruct the vortex merger initial condition depending on $\mathbf{u}$, we decided to perform a simulation with the solver written in the alternative formulation in a very short time interval, namely $(10^{-6}, 2 \cdot 10^{-6}]$. We obtained the values of the solution $\mathbf{u}$ computed as $\mathbf{u} = \nabla \times \boldsymbol{\Psi}$ at time $t = 2 \cdot 10^{-6}$ and we set it as the initial condition of the solver in the original $(\mathbf{u}, p)$ formulation. This strategy allowed us to approximate the initial condition for the velocity in a much simpler way than describing the vortex merger using directly the variable $\mathbf{u}$. For what concerns p, instead, we imposed homogeneous Neumann condition.
In Figure 3.3, Figure 3.4 and Figure 3.5 the motion of the variables $\mathbf{u}, \omega, \Psi$ computed in the two different formulations of the problem are compared.

As it can be easily observed, the simulations of each variable considered in the two different formulations are basically equivalent. If we focus on the representation of $\mathbf{u}$, we have to take into account that the solution computed using the $\omega - \boldsymbol{\Psi}$ formulation is the result of a post-processing of $\boldsymbol{\Psi}$ while, using *icoFoam*, it is directly computed by the solver. This could make us appreciate the possibility to reconstruct the motion of $\mathbf{u}$ using the alternative solver. This observation can be repeated to analyze the performance of the computation of the two alternative variables starting from the $\mathbf{u} - p$ formulation, but in this case we have to consider the opposite behavior: $\mathbf{u}$ is directly computed by the *icoFoam* solver and post-processed to obtain the evolution of ω and $\boldsymbol{\Psi}$ as $\omega = \nabla \times \mathbf{u}$

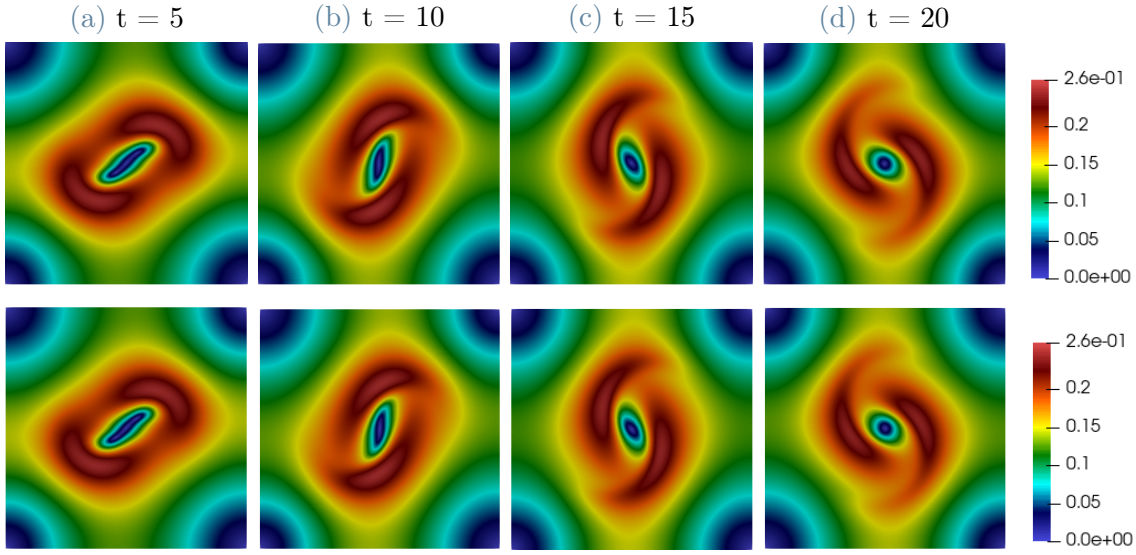


Figure 3.3: Evolution in time of $\mathbf{u}$ computed in the $\omega - \mathbf{\Psi}$ formulation (top) and in $\mathbf{u} - p$ formulation (bottom) at the different time steps of the simulation.

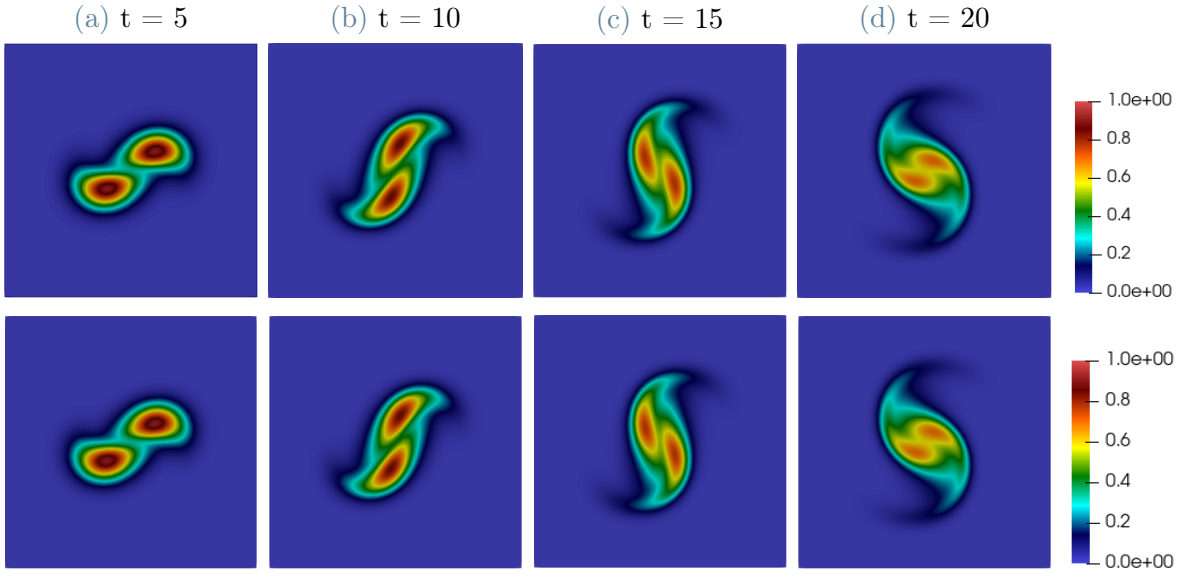


Figure 3.4: Evolution in time of ω computed in the $\omega - \mathbf{\Psi}$ formulation (top) and in $\mathbf{u} - p$ formulation (bottom) at the different time steps of the simulation.

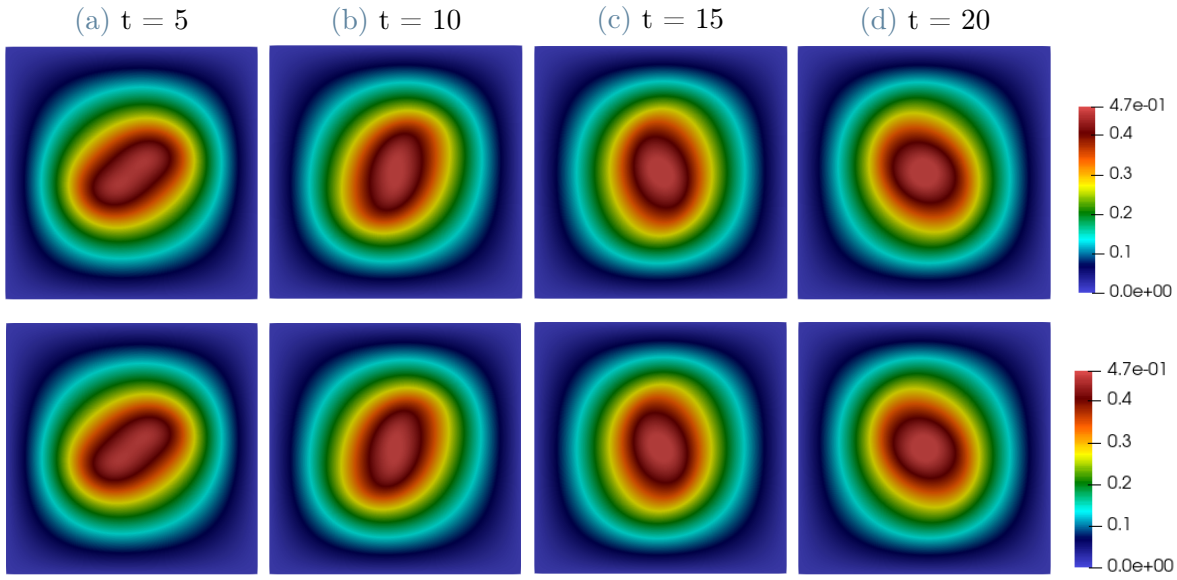


Figure 3.5: Evolution in time of Ψ computed in the $\omega - \boldsymbol{\Psi}$ formulation (top) and in $\mathbf{u} - p$ formulation (bottom) at the different time steps of the simulation.

and $\Psi = \Delta\omega$ respectively. We point out that, since the main focus of this procedure is the validation of the model in the alternative formulation, we discard the representation of the pressure, even if it can be reconstructed by solving a pressure Poisson equation at each time step once $\boldsymbol{\Psi}$ is computed, as prescribed by the Chorin-Temam projection method [43].

3.2. POD-non intrusive model order reduction

Once shown the equivalence between the original and the alternative model, we used the full order model in the alternative formulation to obtain the matrix of the snapshots. In particular, we recreated the framework described in [12] in which the time interval considered, (0,20] s, is divided into sub-intervals of length 0.08 s to obtain 250 snapshots arranged in a matrix. This represented the starting point for the analysis we made to investigate the predictive power of the different data-driven methods explained in Section 2.1.2. To do this we divided our dataset into train and test. We decided to use 50% of the dataset for the training and the remaining 50% for the testing, distributing the snapshots alternatively, one in the training set and one in the test set, in order to have a uniform distribution both of the computed information and of the one to predict. The result is a training set of 126 snapshots and a test set of 124. At this point, by means of *EZyRB* [9], a Python package developed and maintained by the mathLab group

at SISSA to deal with data-driven ROM, we applied the reduced order method in three different variants, *POD-RBF, POD-GPR, POD-NN* and evaluated their performances. In order to perform the ROM, we considered separated computations for the two variables ω and Ψ since the construction of the reduced models of different variables can be tackled independently. In particular, by doing so, we could set a different number of modes for the two variables to perform the POD, as suggested by the analysis of the decay of the eigenvalues taken on in [12]. Figure 3.6 highlights that, considering a threshold for the error of 10^{-5}, the right combination to consider is 14 modes for the vorticity and 6 for the stream function.

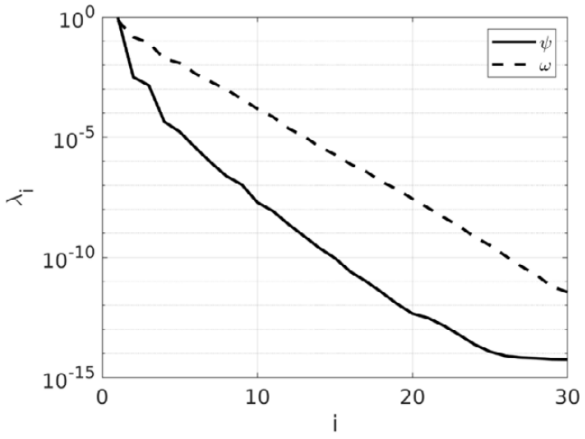


Figure 3.6: Eigenvalues decay for the vorticity (left) and stream function (right).

In the following, we present and compare the results obtained in this framework using the three different methods.

3.2.1. Radial Basis Functions (RBF) interpolation

The first method we applied is POD-RBF. In this framework, we used the RBF interpolation in the default *EZyRB* configuration, namely, we considered the Euclidean distance $d(\mu, \mu_C) = \|\mu - \mu_C\|$ and the thin plate spline as radial functions, $f(\mu) = \|\mu - \mu_C\|^2 \log(\|\mu - \mu_C\|)$. Once set these parameters, we computed the relative L^2 error in percentage as

$$E_\Phi(t) = 100 \cdot \frac{\|\Phi_h(t) - \Phi_r(t)\|_{L^2}}{\|\Phi_h(t)\|_{L^2}}, \tag{3.2}$$

where Φ_h represents the FOM solution while Φ_r the ROM one, see Figure 3.7. Over the entire time interval considered, the error remains below 0.175% for the vorticity and below 0.15% for the stream functions. For what concerns ω, the resulting error is significantly better than the ones obtained with the intrusive approach, while those computed for Ψ

are very similar. In terms of CPU time, we verified that the POD-RBF requires 0.24 s to compute the solution for ω and 0.23 s for Ψ, so summing the two values, 0.47 s are needed to reconstruct the entire solution, that is the same amount of time required using the intrusive approach. So the total speed-up is 10^2 also for the POD-RBF method.

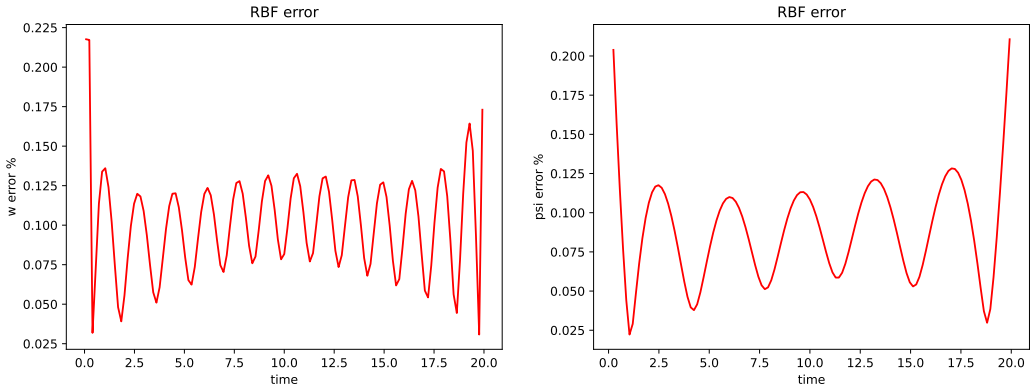


Figure 3.7: Relative percentage error between the full order and the one reconstructed through POD-**RBF** ROM: ω (left), Ψ (right).

In Figure 3.8 and Figure 3.9, we show the qualitative reconstruction of the POD-RBF solutions at three different times and compare them with FOM.

3.2.2. Gaussian Process Regression (GPR)

The second method we applied to our dataset is the POD-GPR in the default *EZyRB* configuration [29]. We computed the error between the exact solution and the one reconstructed by the ROM over the entire time horizon considered, see Figure 3.10. From these representations, we can observe that the error computed for the vorticity remains below the 0.175 % for the entire time interval and for the stream function its maximum value is 0.125 %. For both the variables, the resulting error computed through GPR is very similar to the one computed with RBF even if for Ψ it is slightly lower. Also in this case, the results obtained for the vorticity are better than the one computed with the intrusive approach, while, for the stream function they are comparable. Analyzing the speed-ups, we can assess that POD-GPR requires 0.31 s to compute the solution for ω and 0.25 s for Ψ, so finally 0.56 s are required to reconstruct the entire solution, that is slightly more than the amount of time required using the intrusive approach. The order of magnitude of the speed-up of POD-GPR is 10^2.

In Figure 3.11 and Figure 3.12 we reported the visualization of the comparison between the FOM and POD-GPR solutions at the same instants considered for the previous method.

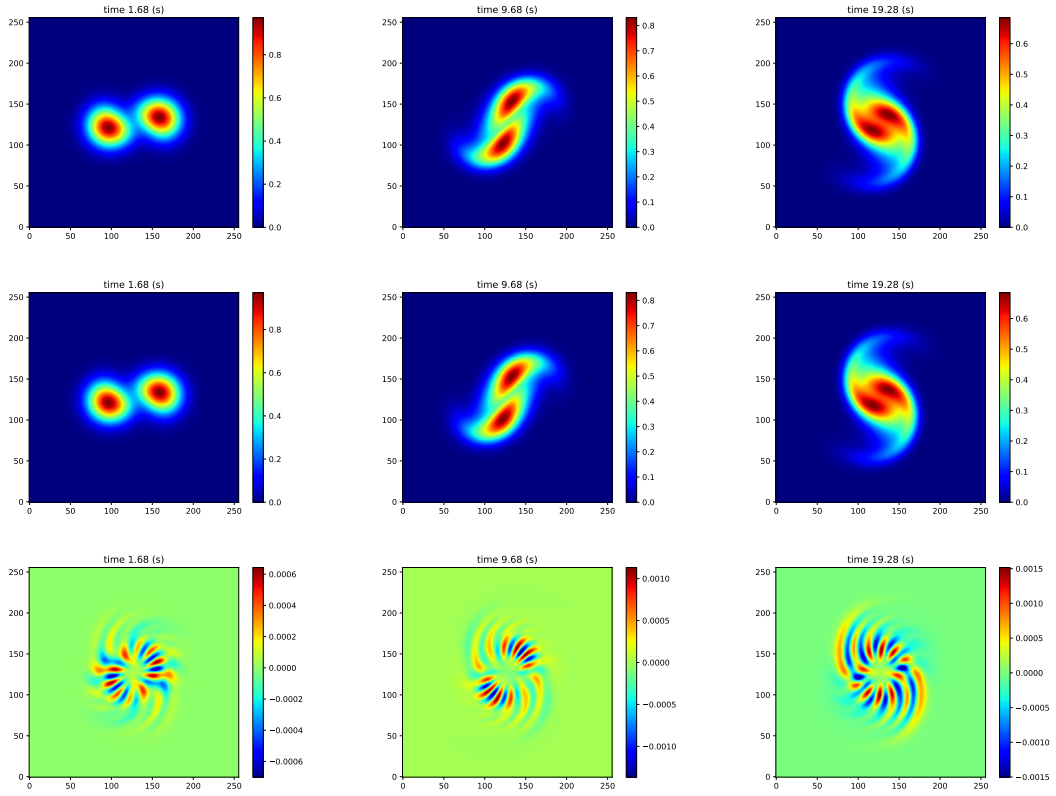


Figure 3.8: Representation of the solution ω at three different times computed through the FOM (top), reconstructed with the POD-**RBF** ROM (middle) and the error between the two (bottom).

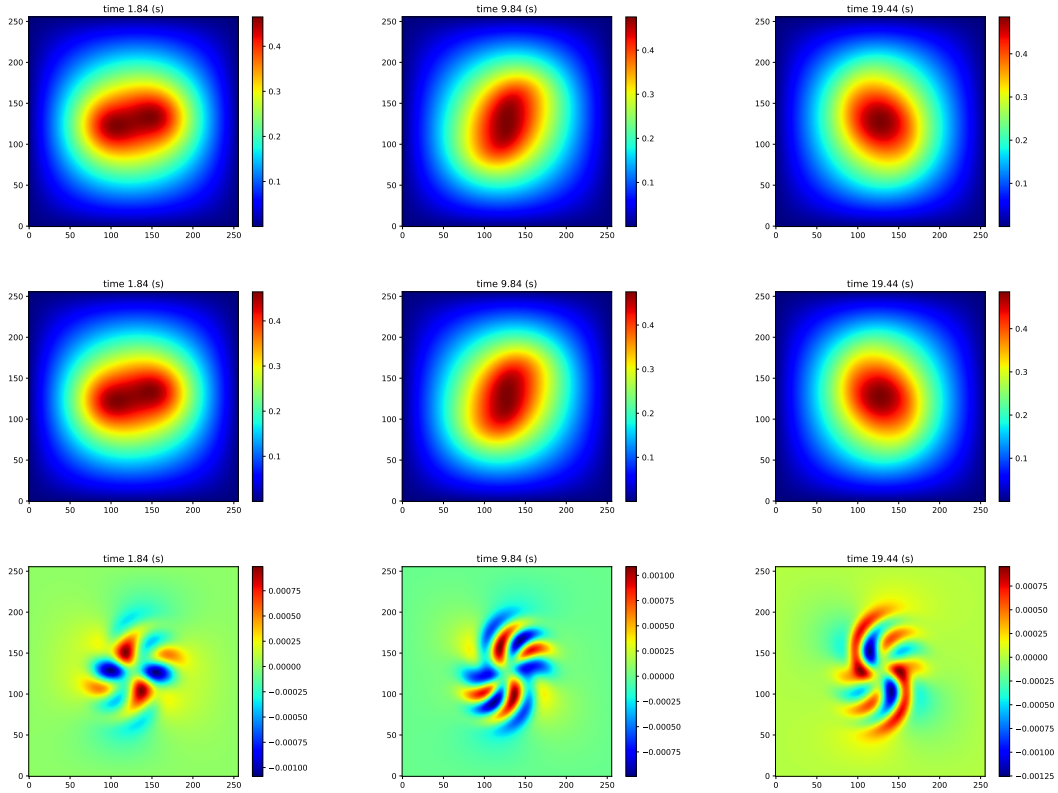


Figure 3.9: Representation of the solution Ψ at three different times computed through the FOM (top), reconstructed with the POD-**RBF** ROM (middle) and the error between the two (bottom).

3.2.3. Artificial neural network (ANN)

Finally, we addressed the problem with the POD-NN configuration. To do so, we trained a Multi-Layers Perceptron (MLP) made of three hidden layers of respectively 16, 64 and 64 neurons. This choice has been made based on the results given by different simulations. Once we decided the topology of the network and decided to use the Mean Squared Error (MSE) as loss function, namely $\sum_{i=1}(y_i - \hat{y}_i)^2$, (where $\hat{y}_i$ is the approximated value), we trained the NN on the dataset and obtained the evolution of the loss function over the epochs depicted in Figure 3.13.

Then, we computed the relative L^2-error in percentage between the solution computed with the FOM and with the POD-NN ROM obtaining the results shown in Figure 3.14. It can be easily observed that the error in percentage achieved very low values for both the two variables, namely it reaches the maximum value of 0.25% for ω and 0.15% for Ψ. These results are slightly higher than the previous approach for ω, while they are quite similar for Ψ. The disadvantage of the ANN lies in the speed-up. Requiring 28

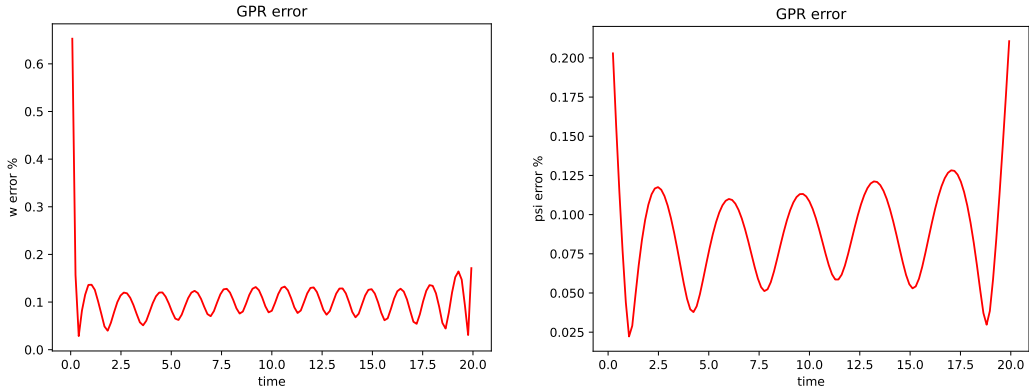


Figure 3.10: Relative percentage error between the full order and the one reconstructed through POD-**GPR** ROM: ω (left), Ψ (right).

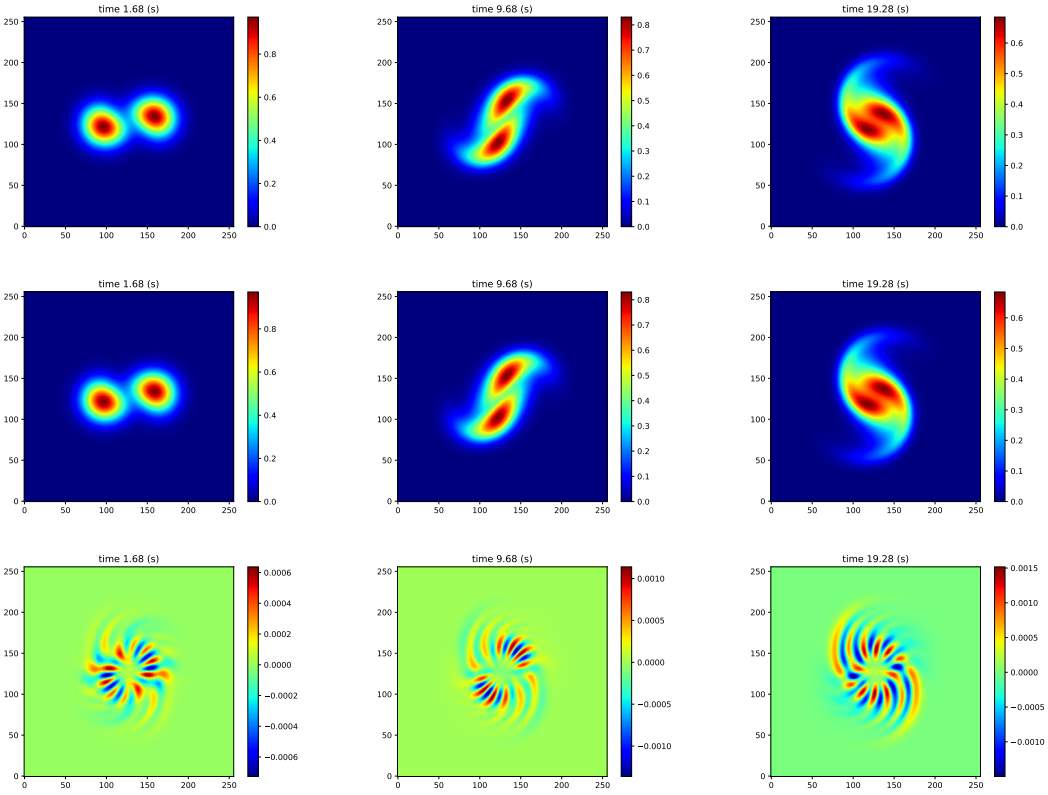


Figure 3.11: Representation of the solution ω in three different time computed through the FOM (top), reconstructed with the POD-**GPR** ROM (middle) and the error between the two (bottom).

s to be trained for the vorticity and 34 s for the stream function, the speed-up for the neural network is almost 1. This result makes this method in this baseline configuration unsuitable for real applications. In Figure 3.15 and Figure 3.16 we report the plot of the full-order and reduced-order solutions to make a comparison.

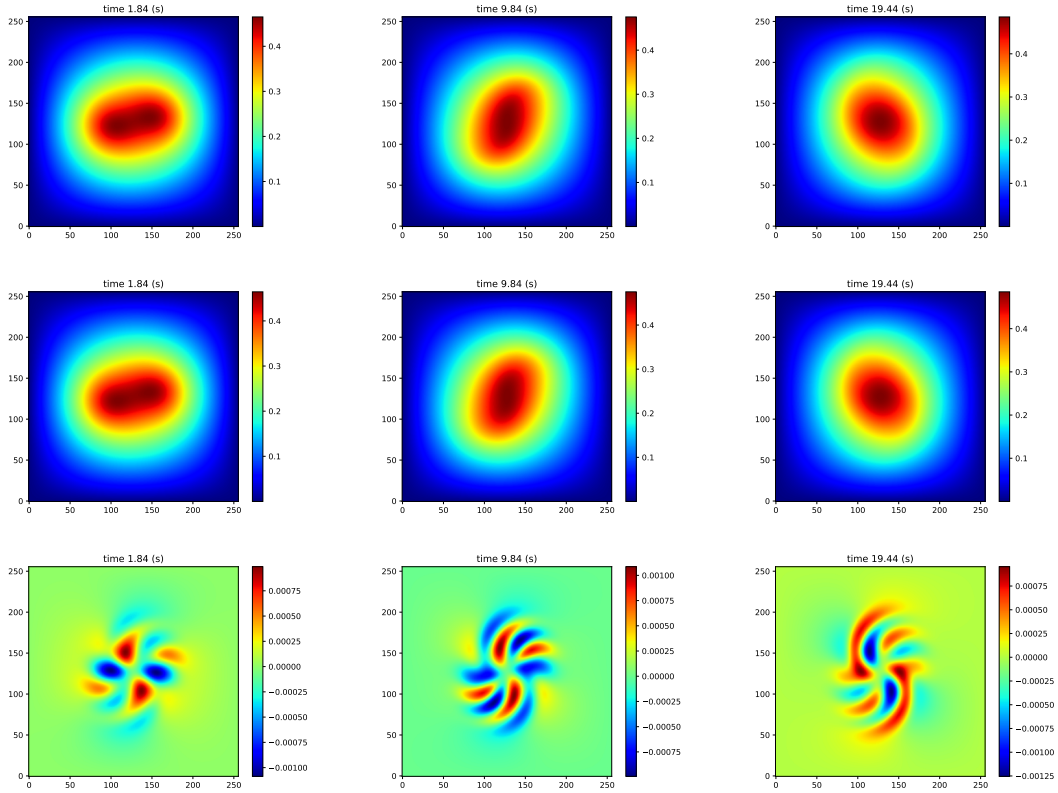


Figure 3.12: Representation of the solution Ψ in three different times computed through the FOM (top), reconstructed with the POD-**GPR** ROM (middle) and the error between the two (bottom).

3.2.4. Extrapolation

Given the promising prediction results on the sample dataset, we wondered if the three data-driven methods investigated in the previous sections could be exploited to make predictions on a wider time horizon. With this in mind, we decided to answer this question by testing the performance of each method in extrapolating the solution advancing in time, given the model trained as explained in Section 3.2. In particular, we addressed the predictions in the time interval (20 s, 24 s] with the usual time step $\Delta t = 0.08$ s and compared the computed solutions with the different methods at three different instants, one very close to the training set, 20.48 s, one in the middle of the extrapolation interval, 22 s, and one at the end 24 s.

In the following, we briefly discuss the performances of each method.

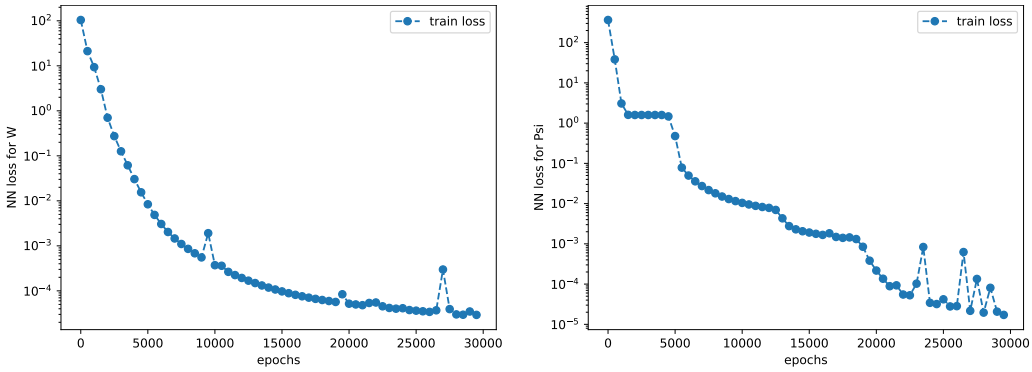


Figure 3.13: Loss function of the NN for ω (left) and Ψ (right) with respect to the training-set.

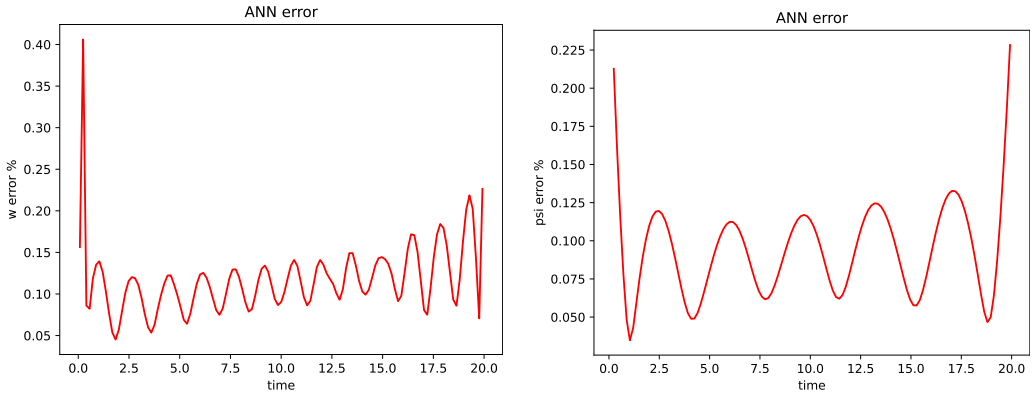


Figure 3.14: Relative percentage error for ω (left) and of Ψ (right) computed through POD-**NN** method.

RBF extrapolation

With the same configuration of the method described in Section 3.2, we took on the prediction of the following interval, obtaining the evolution of the error shown in Figure 3.17. From these plots, we could observe that the evolution of Ψ can be captured advancing in time with a small error, while for ω the error introduced is quite significant.

In Figure 3.18 and Figure 3.19 we showed the comparison of the solutions.

GPR extrapolation

Addressing the same extrapolation task with the GPR method we obtained the errors shown in Figure 3.20.

In Figure 3.21 and Figure 3.22 we reported the simulated solutions in the time under analysis.

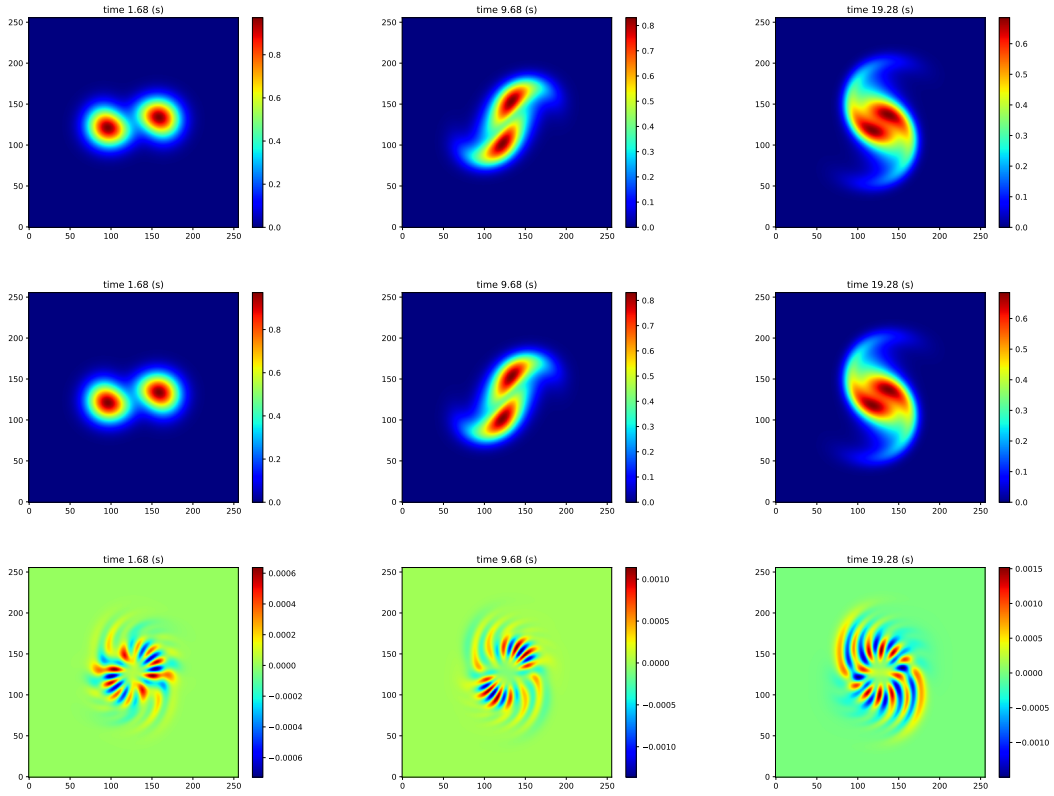


Figure 3.15: Representation of the solution ω in three different time computed through the FOM (top), reconstructed with the POD-**NN** ROM (middle) and the error between the two (bottom).

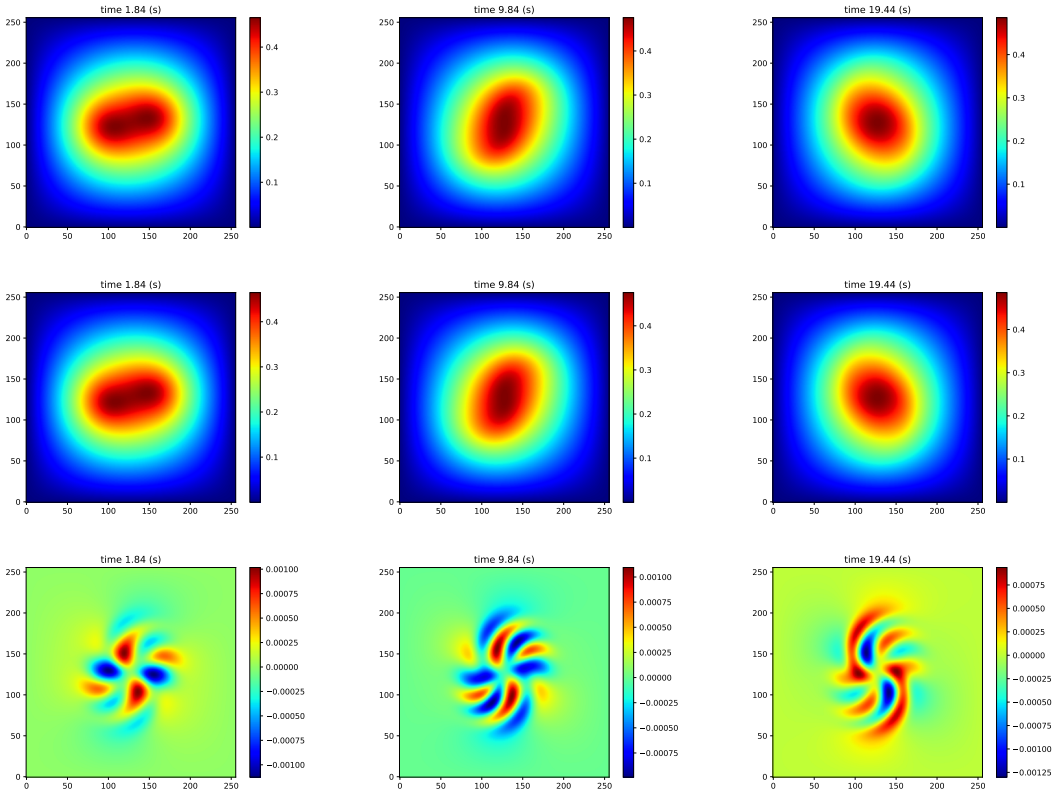


Figure 3.16: Representation of the solution Ψ in three different times computed through the FOM (top), reconstructed with the POD-**NN** ROM (middle) and the error between the two (bottom).

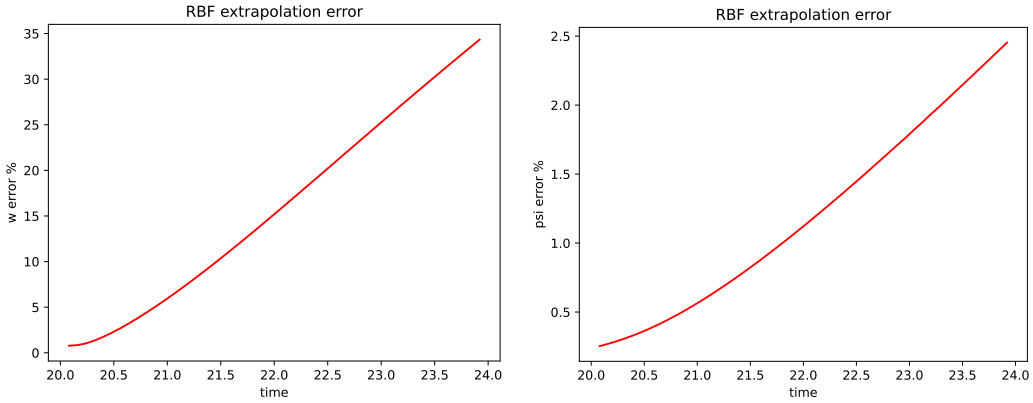


Figure 3.17: Relative percentage error for ω (left) and of Ψ (right) computed through POD-**RBF** method in the extrapolation of the solutions.

ANN extrapolation

Finally, we performed the analysis through the neural network, obtaining the errors depicted in Figure 3.23.

The reconstructed solutions computed with the ANN can be visualized in Figure 3.24 and Figure 3.25.

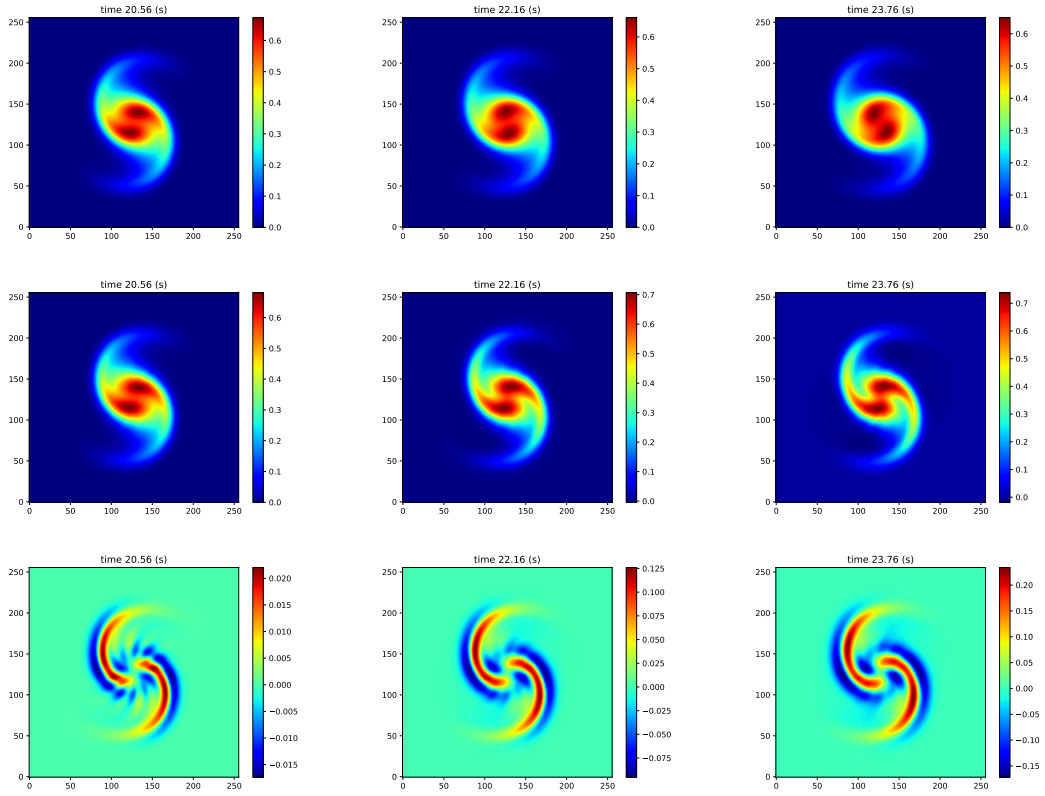


Figure 3.18: Representation of the **extrapolated** solution ω in three different time computed through the FOM (top), reconstructed with the POD-**RBF** ROM (middle) and the error between the two (bottom).

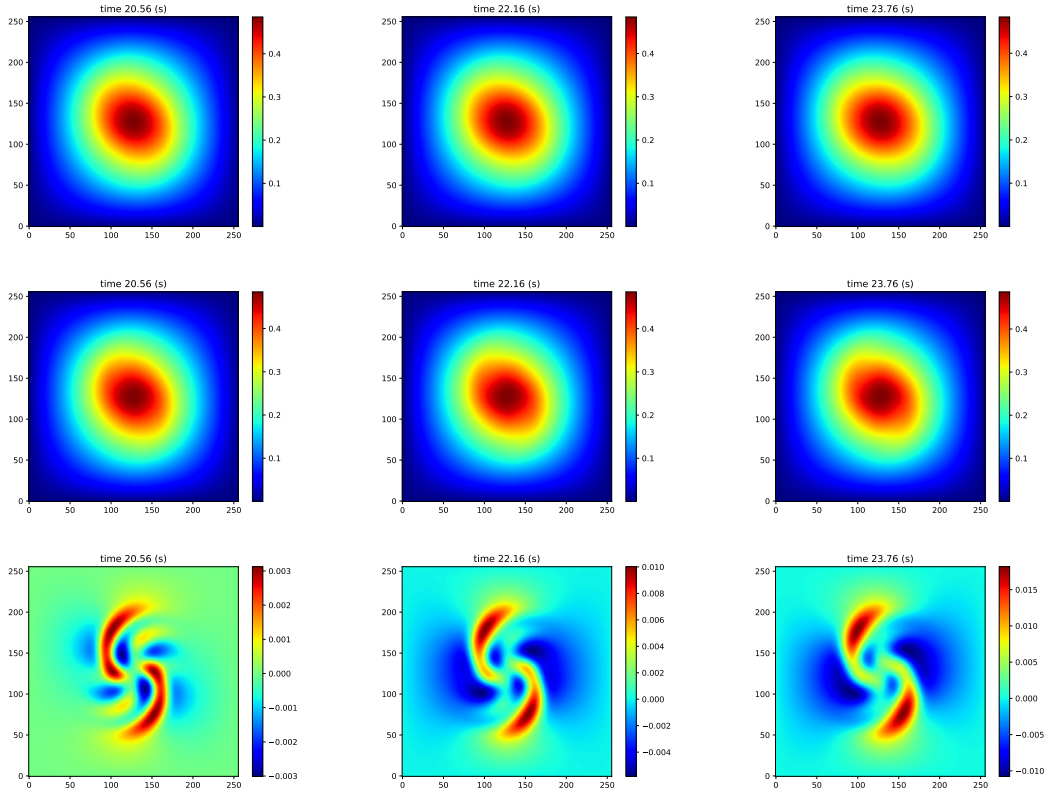


Figure 3.19: Representation of the **extrapolated** solution Ψ in three different times computed through the FOM (top), reconstructed with the POD-**RBF** ROM (middle) and the error between the two (bottom).

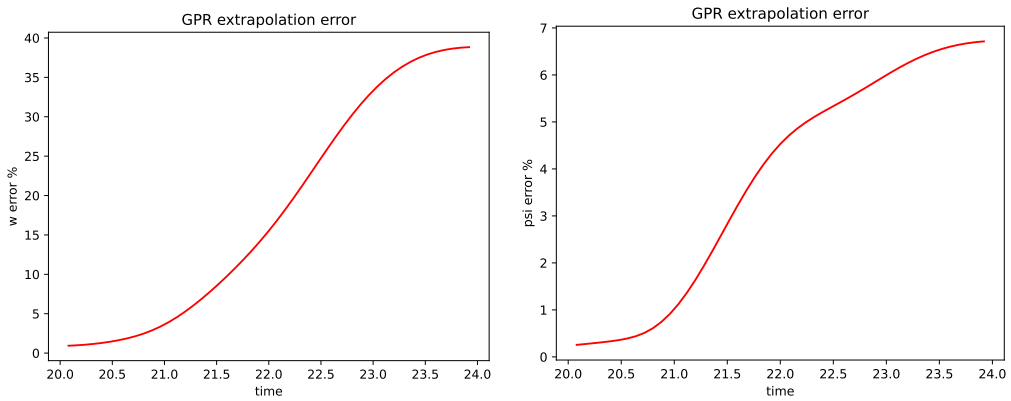


Figure 3.20: Relative percentage error for ω (left) and of Ψ (right) computed through POD-**GPR** method in the extrapolation of the solutions.

3.2.5. Comments

To summarize, the methods analyzed to address the non-intrusive ROM give promising results in predicting the values of the solutions that encourage to further investigation. We

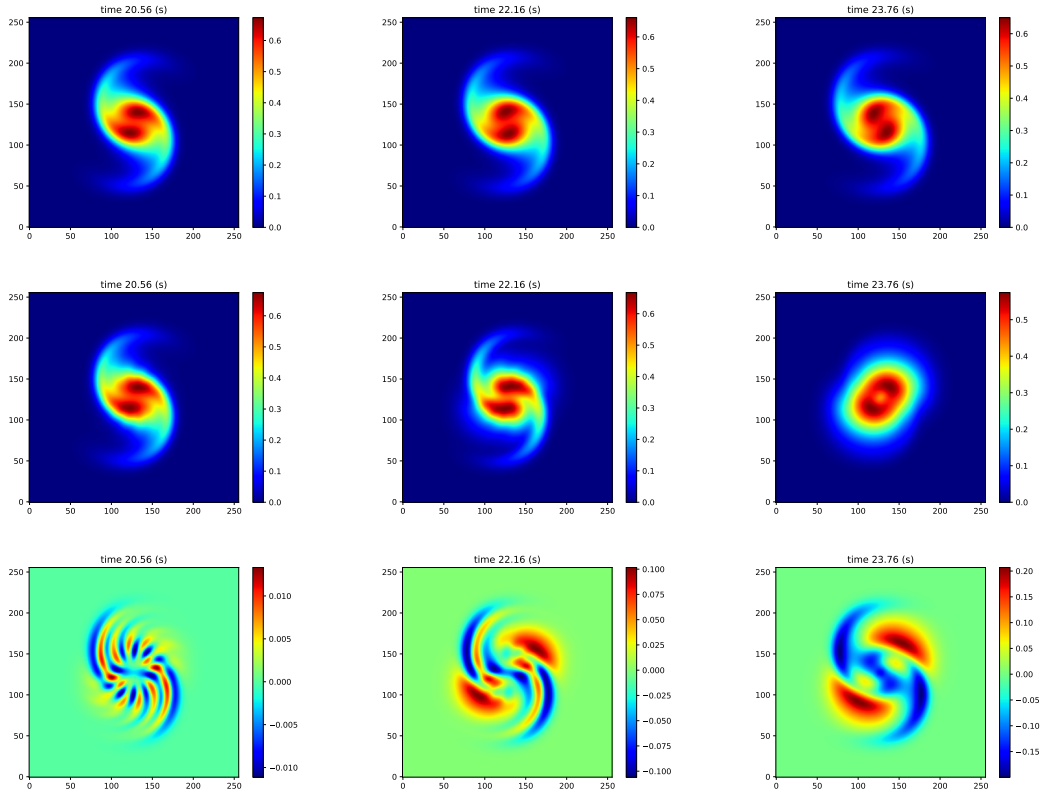


Figure 3.21: Representation of the **extrapolated** solution ω in three different time computed through the FOM (top), reconstructed with the POD-**GPR** ROM (middle) and the error between the two (bottom).

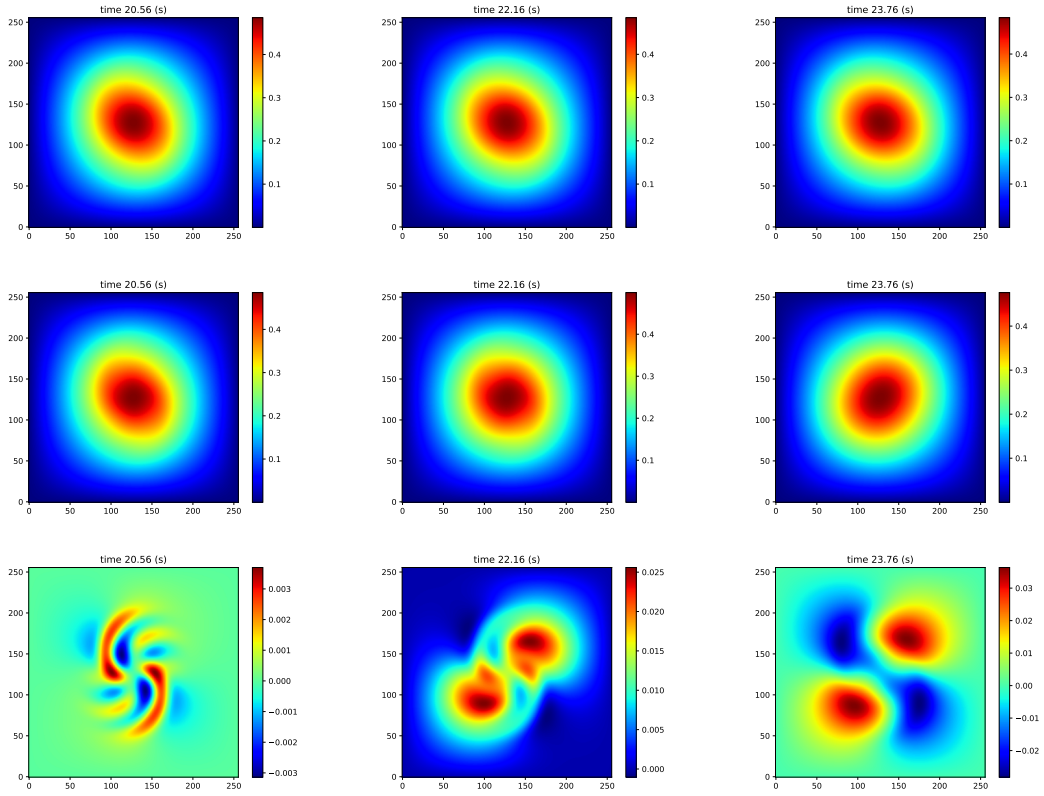


Figure 3.22: Representation of the **extrapolated** solution Ψ in three different times computed through the FOM (top), reconstructed with the POD-**GPR** ROM (middle) and the error between the two (bottom).

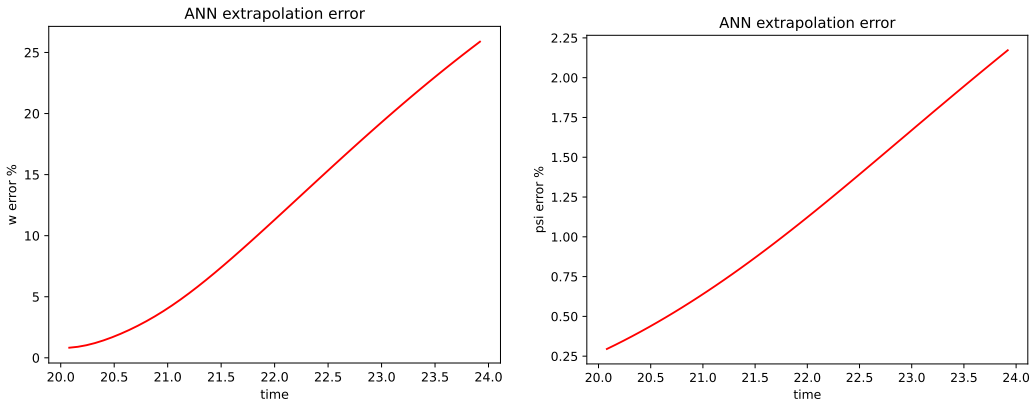


Figure 3.23: Relative percentage error in the extrapolation of the solutions for ω (left) and of Ψ (right) computed through POD-**NN** method.

can observe that each technique reveals better performance for the variable Ψ while the computations of ω produce worse predictions. This can be due to the higher complexity of the variable, as stated in [12]. Considering the percentage of the error committed on the test set and the speed-ups for the vorticity, the best method is the RBF, while for the prediction of the stream function, the performances of the RBF and GPR are comparable. We can also assess that the worse method to tackle the predictive task is the NN since the error computed is similar to the others, but it loses in the speed-up. While, for what concerns the extrapolation issue, the NN achieves the lowest error value for both variables among the three methods. In this analysis, the worst is the GPR which reaches a maximum error value of 40% for the vorticity and of 7% for the stream function.

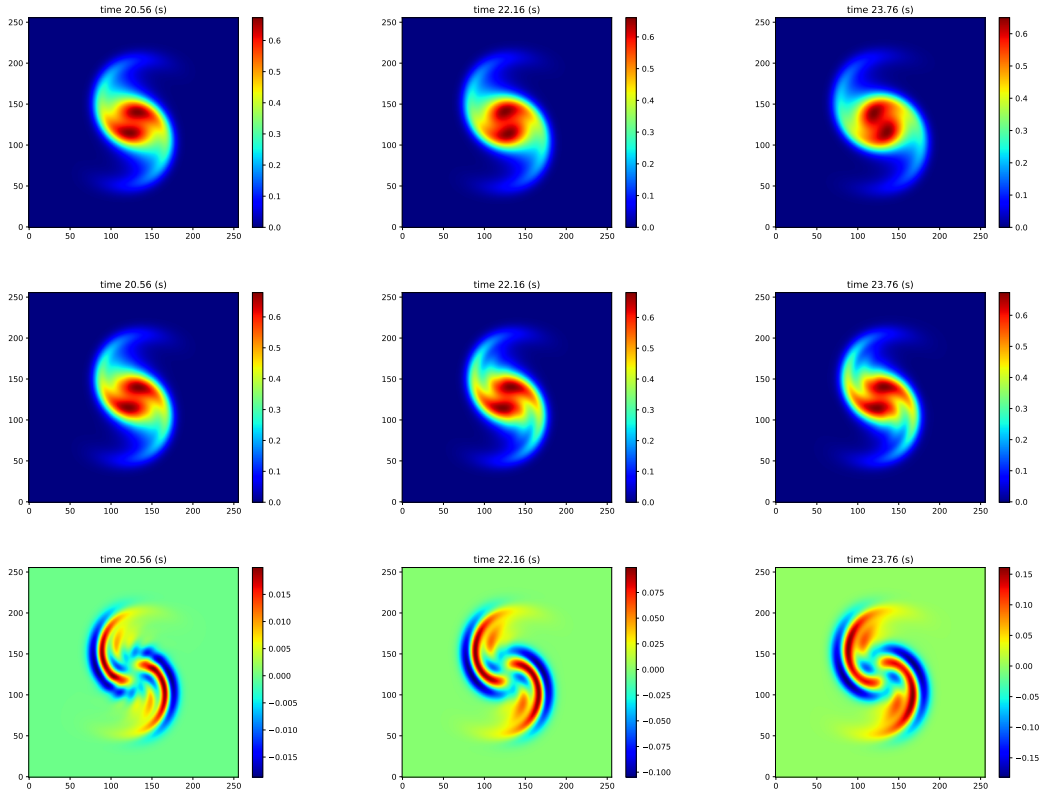


Figure 3.24: Representation of the **extrapolated** solution ω in three instants under analysis computed through the FOM (top), reconstructed with the POD-**ANN** ROM (middle) and the error between the two (bottom).

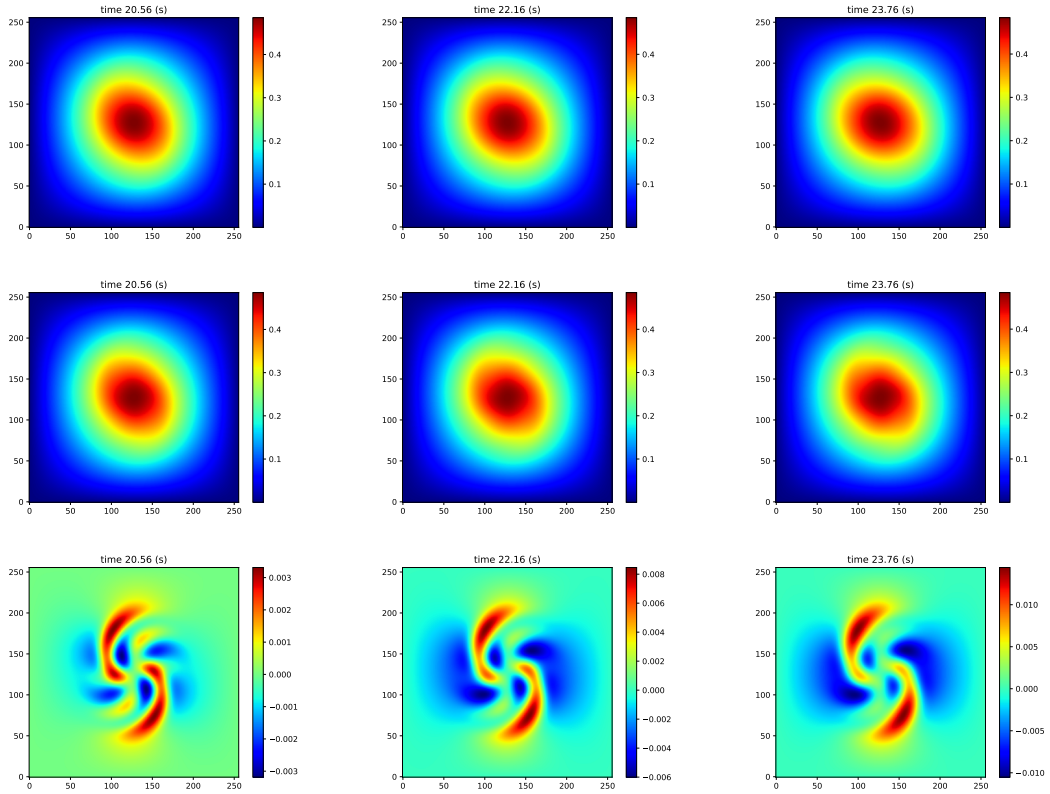


Figure 3.25: Representation of the **extrapolated** solution Ψ in three instants under analysis computed through the FOM (top), reconstructed with the POD-**ANN** ROM (middle) and the error between the two (bottom).

4 | Conclusions and future developments

The work just presented aimed to analyze a different approach to the solution of the Navier-Stokes equations in the context of atmospheric and geophysical problems where the alternative variables under analysis have great use. We constructed and validated the alternative model on a simple test case and used the solutions computed as a starting point for the application of the data-driven ROM in the POD-RBF, POD-GPR and POD-NN configurations, we made a comparison between these data-driven methods and the analytical approach followed in [12]. For what concerns the conclusions we can draw from this study, we can state the equivalence of the alternative $\omega - \mathbf{Psi}$ solver and the original one in the full order approach. We can also assess that the capabilities of the non-intrusive approach in the configurations investigated produced significant results in making predictions, both on the sample dataset and on the advanced time horizon. In future analysis, it would be interesting first to test and compare the extrapolation capabilities of the intrusive approach and to further investigate the data-driven methods, for instance, by changing the parameters involved in the training. In particular, for what concerns the RBF, it would be challenging to try different radial functions and different definitions of the distance, for the GPR the changing of the kernel functions could lead to better results, while for the NN a deeper study can be carried out to find out the optimal network to address the problem. For instance, as suggested in [13], the usage of Long-Short Term Memory neurons could reveal more suitable in the prediction of sequential data. Given the promising results obtained by training the model on the particular configuration of the dataset (one snapshot for training and one for testing alternatively), different patterns for the division of data in train and test can be considered to quantify how the dimension of the training set influences the performances in the prediction. However, we are aware that these advantages are paid in generalization, since, as for any data-driven method, their performance depends on the set used for the training, furthermore, the fact that they work as black-boxes, makes them poorly interpretable with respect to the intrusive approach.

We plan to expand this work in the future making the model more realistic by including a stochastic variable to describe the angle of incidence of the wind. This allows us to take into account a possible external source of perturbations. Since this parameter is, by definition, affected by uncertainty, its introduction requires the application of a different variant of the ROM. In particular, we tackle this issue by applying the *weighted* ROMs for forward uncertainty problems. Weighted ROMs are slight modifications of classical ROM methods that take into account some previous knowledge of the distribution of the parameters and use this information to accelerate the reduced simulations [6], [45]. We point out that the proposed weighted ROMs operates weighing the ROM outcome without changing their form. However, this is not the unique strategy to deal with this kind of problems. Another possibility is to reconstruct the stochastic PDEs operating in the discretization procedure, as done in [44]. For what concerns our analysis, we decided to address the new problem by restoring the original intrusive approach analyzed in [12] and expand the analysis dealing with the issue just introduced. To do this, we will, first of all, perform the weighted POD and then decide which strategy is better to use for the sampling among the most common: *Monte Carlo uniform sampling* to approximate the momenta of the error between the ROM and FOM solution, *tensor product quadrature rule*, *sparse Smolyak quadrature rule*.

Bibliography

[1] OpenFOAM Library. https://openfoam.org/.

[2] J. A. Atwell and B. B. King. Proper orthogonal decomposition for reduced basis feedback controllers for parabolic equations. volume 33, pages 1–19. 2001. doi: 10.1016/S0895-7177(00)00225-9. URL https://doi.org/10.1016/S0895-7177(00)00225-9. Computation and control, VI (Bozeman, MT, 1998).

[3] P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. Miguel Silveira. *Model Order Reduction: Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter, 2020.

[4] P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. Miguel Silveira. *Model order reduction: volume 3 applications*. De Gruyter, 2020.

[5] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira. Model order reduction: Volume i: System-and data-driven methods and algorithms. 2021.

[6] P. Chen, A. Quarteroni, and G. Rozza. A weighted reduced basis method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 51:3163–3185, 01 2013. doi: 10.1137/130905253.

[7] G. Cybenko. Continuous valued neural networks with two hidden layers are sufficient, department of computer science. *Trfts. University*, 31, 1988.

[8] A. G. de G. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[9] N. Demo, M. Tezzele, and G. Rozza. EZyRB: Easy Reduced Basis method. *The Journal of Open Source Software*, 3(24):661, 2018. doi: https://doi.org/10.21105/joss.00661.

[10] A. Dumon, C. Allery, and A. Ammar. Proper generalized decomposition method for

incompressible flows in stream-vorticity formulation. *European Journal of Control - EUR J CONTROL*, 19:591–617, 11 2010. doi: 10.3166/ejcm.19.591-617.

[11] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:221–218, 1936.

[12] M. Girfoglio, A. Quaini, and G. Rozza. A POD-Galerkin reduced order model for the Navier–Stokes equations in stream function-vorticity formulation. *Computers & Fluids*, 2022.

[13] I. C. Gonnella, M. W. Hess, G. Stabile, and G. Rozza. A two stages deep learning architecture for model reduction of parametric time-dependent problems, 2023.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[15] C. J. Greenshields. OpenFOAM programmer's guide. *OpenFOAM Foundation Ltd*, 2015.

[16] M. D. Gunzburger. *Perspectives in flow control and optimization*. SIAM, 2002.

[17] J. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.02.037. URL `https://www.sciencedirect.com/science/article/pii/S0021999118301190`.

[18] J. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer, (2016).

[19] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[20] H. Jasak. Error analysis and estimation for the finite volume method with applications to fluid flows. *Direct*, M, 01 1996.

[21] 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, 2002.

[22] 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(1-2):521–536, 2002.

[23] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes, 2018.

[24] J. Lequeurre and A. Munnier. Vorticity and stream function formulations for the 2d navier–stokes equations in a bounded domain. *Journal of Mathematical Fluid Mechanics*, 22(2):15, 2020.

[25] S. Lorenzi, A. Cammi, L. Luzzi, and G. Rozza. POD-Galerkin method for finite volume approximation of Navier–Stokes and RANS equations. *Computer Methods in Applied Mechanics and Engineering*, 311:151–179, 2016.

[26] D. J. Lucia, P. S. Beran, and W. A. Silva. Reduced-order modeling: new approaches for computational physics. *Progress in Aerospace Sciences*, 40(1):51–117, 2004. ISSN 0376-0421. doi: https://doi.org/10.1016/j.paerosci.2003.12.001. URL `https://www.sciencedirect.com/science/article/pii/S0376042103001131`.

[27] P. Minev and P. N. Vabishchevich. An operator-splitting scheme for the stream function–vorticity formulation of the unsteady navier–stokes equations. *Journal of computational and applied mathematics*, 293:147–163, 2016.

[28] E. Muravleva, I. Oseledets, and D. Koroteev. Application of machine learning to viscoplastic flow modeling. 09 2018.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] A. Quarteroni. *Le equazioni di Navier-Stokes*. Springer, 2012.

[31] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*, volume 37 of *Texts in Applied Mathematics*. Springer-Verlag, Berlin, second edition, 2007. ISBN 978-3-540-34658-6; 3-540-34658-9. doi: 10.1007/b98885.

[32] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015.

[33] S. M. Rahman, O. San, and A. Rasheed. A hybrid approach for model order reduction of barotropic quasi-geostrophic turbulence. *Fluids*, page 86, 2018. doi: 10.3390/fluids3040086.

[34] C. E. Rasmussen. *Gaussian Processes in Machine Learning*, pages 63–71. Springer Berlin Heidelberg, 2004.

[35] C. E. Rasmussen, C. K. Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

[36] J. N. Reinaud and D. G. Dritschel. The critical merger distance between two co-rotating quasi-geostrophic vortices. *Journal of Fluid Mechanics*, 522:357–381, 2005. doi: 10.1017/S0022112004002022.

[37] S. Sharma, S. Sharma, and A. Athaiya. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 04:310–316, 05 2020. doi: 10.33564/IJEAST.2020.v04i12.054.

[38] G. Stabile and G. Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier-Stokes equations. *Computers & Fluids*, 2018. doi: 10.1016/j.compfluid.2018.01.035.

[39] 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.

[40] 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). doi: 10.1515/caim-2017-0011.

[41] W. T. Stephenson, S. Ghosh, T. D. Nguyen, M. Yurochkin, S. Deshpande, and T. Broderick. Measuring the robustness of gaussian processes to kernel choice. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 3308–3331. PMLR, 28–30 Mar 2022.

[42] W. T. Stephenson, S. Ghosh, T. D. Nguyen, M. Yurochkin, S. K. Deshpande, and T. Broderick. Measuring the robustness of gaussian processes to kernel choice, 2022.

[43] R. Temam. *Navier-Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Soc., 2001.

[44] S. Tokareva, A. Zlotnik, and V. Gyrya. Stochastic finite volume method for uncertainty quantification of transient flow in gas pipeline networks, 2022.

[45] L. Venturi, D. Torlo, F. Ballarin, and G. Rozza. *Weighted Reduced Order Methods for Parametrized Partial Differential Equations with Random Inputs*, pages 27–40. 01 2019. ISBN 978-3-030-04869-3. doi: 10.1007/978-3-030-04870-9_2.

[46] R. Vinuesa and S. L. Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022.

[47] S. Volkwein. *Model Reduction using Proper Orthogonal Decomposition*. Lecture notes, University of Konstanz, 2011.

[48] S. Walton, O. Hassan, and K. Morgan. Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions. *Applied Mathematical Modelling*, 37(20-21):8930–8945, 2013.

[49] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA journal*, 40(11):2323–2330, 2002.

A | Appendix: Derivation of Navier-Stokes equation in primitive variables

In order to obtain the Navier-Stokes equation describing the motion of an incompressible fluid, we started considering the mass conservation law and the second Newton's law.

Continuity equation: We consider the fact that in any finite volume, the mass does not change in time. This principle can be formulated as:

$$\frac{d}{dt}\int_{V_t} \rho dV = 0, \tag{A.1}$$

by applying the Reynolds transport theorem for a material element, we obtain that

$$\frac{d}{dt}\int_{V_t} \rho dV = \int_{V_t} (\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}))dV = 0. \tag{A.2}$$

Where $\rho = \rho(x,t)$ is intended as the density of the mass while $\mathbf{u} = \mathbf{u}(x,t)$ is the Eulerian velocity. Thanks to the arbitrariness of the volume under analysis, we can consider the differential form of the equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{A.3}$$

and since we are considering an incompressible fluid, namely with constant density in space and time, this expression leads to the continuity equation:

$$\nabla \cdot \mathbf{u} = 0. \tag{A.4}$$

Momentum equation: The second step is to analyze the second Newton's law stating that the variation of the linear momentum of a system is equal to the resultant forces

acting on it that we consider as the sum of volume and surface.

$$\frac{d}{dt}\int_{V_t}\rho\mathbf{u}dV = \int_{V_t}\rho\mathbf{f}dV + \int_{\partial V_t}\mathbf{T}\cdot\mathbf{n}d\sigma, \tag{A.5}$$

where $\mathbf{f}$ are the volume forces per unit of mass and T is the stress tensor. Applying again the Reynolds theorem and considering the arbitrariness of the volume under analysis, we can focus on the differential form of the expression:

$$\frac{\partial}{\partial t}(\rho\mathbf{u}) + \nabla\cdot(\rho\mathbf{u}\mathbf{u}^{\mathbf{T}}) = \mathbf{f} + \nabla\cdot\mathbf{T}. \tag{A.6}$$

We now introduce the approximations typical of a Newtonian viscous fluid and we obtain the momentum equation:

$$\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p = \mathbf{f}, \tag{A.7}$$

where we divided the entire equation by ρ and obtained a new parameter $\nu = \frac{\mu}{\rho}$ that represents the viscosity of the fluid. Solving the two equations together we obtain the Navier-Stokes equation:

$$\begin{cases} \frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p = \mathbf{f}, \\ \nabla\cdot\mathbf{u} = 0, \end{cases} \tag{A.8}$$

that has to be solved in $\Omega\times(t_0, t^*]$, where Ω is the space domain and $(t_0, t^*]$ represents the interval of time considered.

List of Figures

Acknowledgements

This thesis has been developed in collaboration with the SISSA mathLab group in Trieste. First of all, I would like to thank my thesis advisor Prof. Nicola Parolini of Politecnico di Milano for giving me the opportunity to test myself on this stimulating and challenging project. I want to thank also Prof. Gianluigi Rozza of Scuola Internazionale Superiore di Studi Avanzati (SISSA) for believing in my potential and capabilities (more than myself) and for introducing me to the research environment and to a group where I have found many colleagues that now I can call friends. I feel grateful also to Dr. Michele Girfoglio for his kindness and availability in answering my questions and solving my doubts. Moreover, I would like to really thank Caterina Balzotti and Davide Torlo for their support and for their help at any moment both from a professional and personal point of view. I want also to thank Pierfrancesco, Anna, Francesco, Isabella, Dario, Guglielmo e Nicola for welcoming me and for creating strong relationships and a pleasant environment that will be hard to leave. I want to really thank my family and friends for always supporting and encouraging me not to give up, in particular Antonella, Lucrezia e Tommaso, for celebrating with me each goal and supporting me in each difficult moment and Alessandro for proposing to me this great opportunity. Last but not least I want to thank my rugby team LeBlacks and the Old Blacks for their continuous encouragement and moral support.