Journal of Scientific Computing (2023) 95:23
https://doi.org/10.1007/s10915-023-02142-4

Towards a Machine Learning Pipeline in Reduced Order Modelling for Inverse Problems: Neural Networks for Boundary Parametrization, Dimensionality Reduction and Solution Manifold Approximation

Anna Ivagnes[1] · **Nicola Demo**[1] · **Gianluigi Rozza**[1]

Received: 3 March 2022 / Revised: 26 October 2022 / Accepted: 26 January 2023 /
Published online: 24 February 2023

Abstract
In this work, we propose a model order reduction framework to deal with inverse problems in a non-intrusive setting. Inverse problems, especially in a partial differential equation context, require a huge computational load due to the iterative optimization process. To accelerate such a procedure, we apply a numerical pipeline that involves artificial neural networks to parametrize the boundary conditions of the problem in hand, compress the dimensionality of the (full-order) snapshots, and approximate the parametric solution manifold. It derives a general framework capable to provide an ad-hoc parametrization of the inlet boundary and quickly converges to the optimal solution thanks to model order reduction. We present in this contribution the results obtained by applying such methods to two different CFD test cases.

Keywords Model order reduction · Inverse problem · Machine learning · Artificial neural network

Abbreviations

CFD	Computational fluid dynamics
ROM	Reduced order model
PDE	Partial differential equation
POD	Proper orthogonal decomposition
ANN	Artificial neural network
AE	AutoEncoder

Anna Ivagnes and Nicola Demo have contributed equally.

✉ Gianluigi Rozza
gianluigi.rozza@sissa.it

Anna Ivagnes
anna.ivagnes@sissa.it

Nicola Demo
nicola.demo@sissa.it

[1] SISSA Mathematics Area, mathLab, Via Bonomea, 265, 34136 Trieste, Italy

RBF	Radial basis function

Symbols

σ	Activation function in ANN
M	Number of snapshots in ROM
p	Number of parameters used in ROM
P	Number of degrees of freedom for the velocity field
L	Latent dimension
$(\boldsymbol{\varphi})_{i=1}^{M}$	Velocity POD modes
$\|\cdot\|$	Norm in L^2

1 Introduction

Inverse problems is a wide family of problems that crosses many different sciences and engineering fields. Inverse problems aim to compute from the given observations the cause that has produced them, as also explained in [1, 2]. Formally, we can consider an input I and an output O, and suppose that there exists a map

$$\mathcal{M} : i \to o$$

that models a mathematical or physical law. The computation of the output as $o = \mathcal{M}(i)$ is called *direct problem*, whereas the problem of finding the input given the output is called *inverse problem*. Given a certain output o_t, the inverse problem consists of inverting the map $\mathcal{M}$ and finding the input i_t which produces the output o_t, i.e., which satisfies $\mathcal{M}(i_t) = o_t$. Inverse problems may be of interest for a lot of mathematical fields, from heat transfer problems to fluid dynamics. The case study which is here analysed is a Navier–Stokes flow in a circular cylinder, and the aim is to find the proper boundary condition in order to obtain the expected distribution within the domain. Such an application tries to solve a typical naval problem, numerically looking for the inlet setting which provides the right working condition during the optimization of the propulsion system. Propeller optimization is indeed very sensitive to modifications in the velocity distribution at the propeller plane: to obtain an optimized artifact it becomes very important to set up the numeric simulation such that the velocity distribution is as close as possible to the target distribution, usually collected by experimental tests. The problem is the identification of the inlet boundary condition, given the velocity distribution at the so-called *wake*, which is the plane (or a portion of it) orthogonal to the cylinder axis where the propeller operates. To achieve that, the inlet distribution is searched by parametrizing the target wake—by exploiting a neural network, as we will describe in the next paragraphs—and optimizing these parameters such that in the simulation the velocity is close to the target wake. It must be said that to produce meaningful results, we assume here the flow has a main direction that is perpendicular to the inlet and wake planes: in such a way, the distributions at these planes are similar to each other, allowing us to search for the optimal inlet among the parametrized wake distributions. Even if in this case the numerical experiments are conducted in a Computational Fluid Dynamics (CFD) setting, the methodology is in principle easily transferable to different contexts.

Typically, such an operation is performed within an optimization procedure, in which the direct problem is iteratively solved by varying the input until the desired output is reached. This, of course, implies the necessity to numerically parametrize the input in a proper way, possibly allowing a large variety of admissible inputs and at the same time a limited number of parameters. Moreover, the necessity to solve the direct problem for many different instances

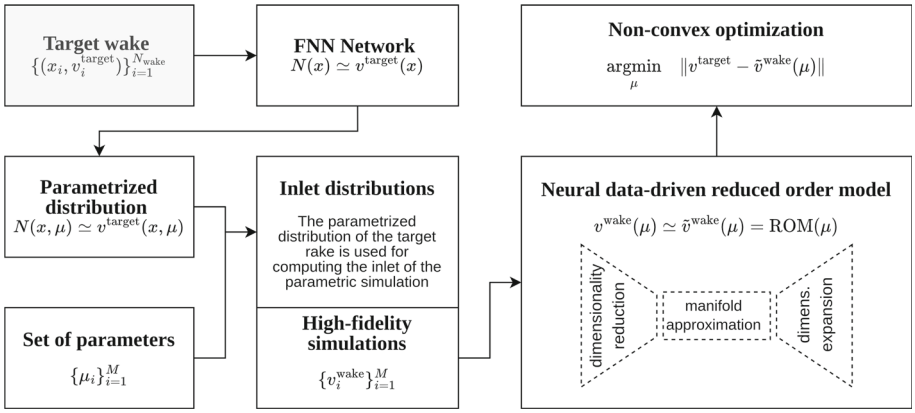


Fig. 1 Flow diagram for the data-driven pipeline followed in the paper

makes the entire process computationally expensive, especially dealing with the numerical solution of Partial Differential Equations (PDEs). A possible solution to overcome such computational burden is offered by the Reduced Order Modelling (ROM) techniques.
ROM constitutes a constantly growing approach for model simplification, allowing for a real-time approximation of the numerical solution of the problem at hand. Among the methods already introduced in the literature, the Proper Orthogonal Decomposition (POD) has become in recent developments an important tool for dealing with PDEs, especially in parametric settings [3–6]. Such a framework aims to efficiently combine the numerical solutions for different configurations of the problem, typically pre-computed using consolidated methods—e.g. finite volume, finite element — such that at any model inference all this information is combined for providing a fast approximation. Within iterative and many-query processes, like inverse problems, this methodology allows a huge computational gain. The many repetitions of the direct problem, needed to find the target input, can be performed at the reduced level, requiring then a finite and fixed set of numerical solutions only for building the ROM. The coupling between ROM and the inverse problem has been already investigated in literature for heat flux estimation in a data assimilation context [7], in aerodynamic application [8], in haemodynamic problems [9]. An alternative way to efficiently deal with this kind of problem has been explored in a Bayesian framework [10]. Moreover, among all the contributions in literature we cite [11, 12] as an example of inverse problem with pointwise observations and inverse problem in a boundary element method context, respectively.

This contribution introduces an entire and novel machine learning pipeline to deal with the inverse problems in a ROM setting. In specific, we combine three different uses of Artificial Neural Network (ANN), that are: *i*) parametrization of the boundary condition given a certain target distribution or pointwise observations, *ii*) dimensionality compression of the discrete space of the original—the so-called full-order—model and *iii*) approximation of the parametric solution manifold. It derives a data-driven pipeline (graphically represented in Fig. 1) able to provide a parametrization of the original problem, which is successively exploited for the optimization in the reduced space. Finally, the optimization is carried out by involving a Genetic Algorithm (GA), but in principle can be substituted by any other optimization algorithm.

The contribution presents in Sect. 2 an algorithmic overview of the employed methods, whereas Sect. 3 illustrates the results of the numerical investigation pursued to the above-

mentioned test case. In particular, we present details for all the intermediate outcomes, comparing them to the results obtained by employing state-of-the-art techniques. Finally, Sect. 4 is dedicated to summarizing the entire content of the contribution, drawing future perspectives and highlighting the criticisms highlighted during the development of this contribution.

2 Methodology

We dedicate this section to providing an algorithmic overview of the numerical tools composing the computational pipeline.

2.1 Boundary Parametrization Using ANN

Neural networks are a class of regression techniques and the general category of *Feed-forward* neural networks has been the subject of considerable research in several fields in recent years. The capability of ANN to approximate any function [13] and the even greater computational availability allowed indeed the massive employment of such an approach to overcome many limitations. In the field of PDEs, we cite [14–21] as some of main impacting frameworks recently explored. A not yet investigated usage of ANN, to the best of authors' knowledge, is instead the parametrization of a given (scattered) function. We suppose that we have a target distribution $\boldsymbol{v}^{\text{target}} = (\boldsymbol{v}_i^{\text{target}})_{i=1}^{P}$, corresponding to the wake velocity distribution in our case, which has P degrees of freedom. We want to reproduce this distribution by exploiting a neural network technique.

A neural network is defined as a concatenation of an input layer, multiple hidden layers, and a final layer of output neurons. An output of the i-th neuron in the l-th layer of the network is generally defined as:

$$h_i^l = \sigma\left(\sum_{j=1}^{N_{l-1}} W_{ij}^l h_j^{l-1} + b_i^l\right),\ i = 1, \dots, N_l\,, l = 1, \dots, H. \tag{1}$$

where σ is the activation function (which provides non-linearity), b_i^l the bias and W refers to the so-called weights of the synapses of the network, N_l is the number of neurons of the l-th hidden layer, H is the number of layers of the network.

The bias and the weights are the hyperparameters of the network, that are tuned during the training procedure to converge to the optimal values for the approximation in hand. We can think of these hyperparameters as the degree of freedom of our approximation, allowing us to manipulate such distribution by perturbating them. We define the optimal hyperparameters (computed with a generic optimization procedure [22, 23]) as b^* and W^*; the network exploiting these hyperparameters reproduces as output an approximation of our target wake distribution:

$$N(\mathbf{x}) \simeq \boldsymbol{v}^{\text{target}}(\mathbf{x}).$$

The input $\mathbf{x}$ to the whole neural network in this paper corresponds to the polar coordinates of the points of the wake, so we have a two-dimensional input. We can then rearrange Eq. (1) to express the parametrized output of a single hidden layer as follows:

$$h_i^l(\boldsymbol{\mu}) = \sigma\left(\sum_{j=1}^{N_{l-1}} W_{ij}^{*l} h_j^{l-1} + (b_i^{*l} + \mu_i^l)\right),\ i = 1, \ldots, N_l\,, l = 1, \ldots, H. \tag{2}$$

where $\boldsymbol{\mu}^l$ is the vector of parameters in layer l, which applies only to the bias of the layers. We finally obtain the parametrized output $N(\mathbf{x}, \boldsymbol{\mu})$.

The main advantage of this approach is the capability to parametrize any initial distribution, since the weights are initially learnt during the training and then manipulated to obtain its parametric version. On the other hand, the dimension of the weights is typically very large, especially in networks with more than one layer. In networks composed of a large number of layers, a high number of hyperparameters should be tuned. A possible solution to overcome such an issue could be to manipulate just a subset of all the hyperparameters. Indeed, in this paper, only the bias parameters of two hidden layers are perturbed.

Such a posteriori parametrization of any generic distribution is employed in this work to deal with the inverse problem. The main idea is to compute different inlet velocity distributions corresponding to different weights of the ANN. The weights perturbations are used as parameters to build the reduced order model, providing an *ad-hoc* parametrization of the boundary condition based on the target output. It must be said that such parametrization may lead to good results only if some correlation between the boundaries and the target output exists.

2.2 Model Order Reduction

ROMs are a class of techniques aimed to reduce the computational complexity of mathematical models.

The technique used in this paper is *data-driven* or *non-intrusive* ROM, which allows us to build a reduced model without requiring the knowledge of the governing equations of the observed phenomenon. For this reason, this technique is suitable to deal with experimental data and it has been widely used in numerical simulations for industrial, naval, biomedical, and environmental applications [24–30].

Non-intrusive ROMs are based on an offline-online procedure. Each stage of the procedure can be approached in different ways, which are analyzed in the following Sections. All the techniques presented in the next lines have been implemented in the Python package called EZyRB [31].

2.2.1 Dimensionality Reduction

In the dimensionality reduction step, a given set of high-dimensional snapshots is represented by a few reduced coordinates, in order to fight the curse of dimensionality and make the pipeline more efficient.

We consider the following matrix of M snapshots, collecting our data:

$$\mathbf{Y} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{v}_1^{\text{wake}} & \boldsymbol{v}_2^{\text{wake}} & \boldsymbol{v}_3^{\text{wake}} & \ldots & \boldsymbol{v}_M^{\text{wake}} \\ | & | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{v}^{\text{wake}}(\mu_1) & \boldsymbol{v}^{\text{wake}}(\mu_2) & \boldsymbol{v}^{\text{wake}}(\mu_3) & \ldots & \boldsymbol{v}^{\text{wake}}(\mu_M) \\ | & | & | & & | \end{bmatrix},$$

where $\boldsymbol{v}_i^{\text{wake}} \in \mathbb{R}^P$, $i = 1, \ldots, M$ are the velocity distributions in our case, each one corresponding to a different set of parameters $\mu_i \in \mathbb{R}^p$ for $i = 1, \ldots, M$. The goal is to calculate the *reduced* snapshots $\{\hat{\boldsymbol{v}}_i^{\text{wake}} \in \mathbb{R}^L\}_{i=1}^M$ such that

$$\tilde{\boldsymbol{v}}_i^{\text{wake}} \simeq \Psi^{-1}(\Psi(\boldsymbol{v}_i^{\text{wake}})), \quad \hat{\boldsymbol{v}}_i^{\text{wake}} = \Psi(\boldsymbol{v}_i^{\text{wake}}), \tag{3}$$

where $\Psi : \mathbb{R}^P \to \mathbb{R}^L$. We specify that the latent dimension $L \ll P$ has to be selected a priori.

Such phase can be approached by making use of linear or non-linear techniques, such as the POD and the usage of an Autoencoder (AE), respectively.

Proper Orthogonal Decomposition In the first case, the offline part consists of the computation of a reduced basis, composed of a reduced number of vectors named *modes*. The main assumption on which it is based is that each snapshot can be approximated by a linear combination of modes:

$$\boldsymbol{v}_i^{\text{wake}} \simeq \sum_{k=1}^{L} a_i^k \boldsymbol{\varphi}_k, \qquad L \ll P,\ i = 1, \ldots, M,$$

with $\boldsymbol{\varphi}_i \in \mathbb{R}^P$ are the modes and a_i^k are the related modal coefficients.

The computation of modes in the POD procedure can be done in different ways, such as via Singular Value Decomposition (SVD) or the velocity correlation matrix. In the first case, the matrix $\mathbf{Y}$ can be decomposed via singular value decomposition in the following way:

$$\mathbf{Y} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T,$$

where the matrix $\mathbf{U} \in \mathbb{R}^{P \times L}$ stores the POD modes in its columns and matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{L \times L}$ is a diagonal matrix including the singular values in descending order. The matrix of modal coefficients — so, the reduced coordinates — in this case can be computed by:

$$\hat{\mathbf{Y}} = \mathbf{U}^T \mathbf{Y},$$

where the $\hat{\mathbf{Y}} \in \mathbb{R}^{L \times M}$ columns are the reduced snapshots. In the second case, the POD space $\mathbb{V}_{POD} = \text{span}\{[\boldsymbol{\phi}_i]_{i=1}^L\}$ is found solving the following minimization problem:

$$\mathbb{V}_{POD} = \arg\min_{n=1,\ldots,M} \frac{1}{M} \sum_{n=1}^{M} \|\mathbf{Y}^n - \sum_{i=1}^{L} a_n^i \boldsymbol{\varphi}_i\|^2 \, \forall n = 1, \ldots, M,$$

where $\mathbf{Y}^n$ is the n-th column of the matrix $\mathbf{Y}$. This problem is equivalent to computing the eigenvectors and the eigenvalues of the velocity correlation matrix:

$$(\mathbf{C})_{ij} = (\mathbf{Y}^i, \mathbf{Y}^u)_{L^2(\Omega)},$$

where Ω is the domain on which the velocity is defined (the propeller wake plane in this case). The POD modes are expressed as:

$$\boldsymbol{\varphi}_i = \frac{1}{M\lambda_i^u} \sum_{j=1}^{M} \mathbf{Y}^j V_{ij}^y,$$

where V^y stores the eigenvectors of the correlation matrix in its columns and λ_i^y are its eigenvalues.

Autoencoder AE refers to a family of neural networks that, for its architectural features, has become a mathematical tool for dimensionality reduction [17]. In general, an AE is a neural network that is composed of two main parts:

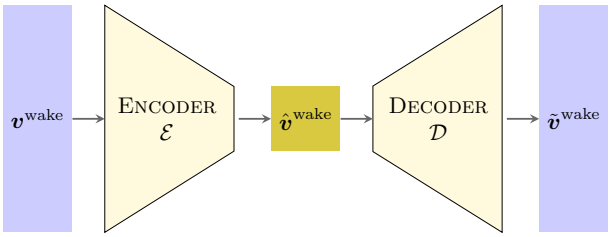


Fig. 2 Schematic structure of an autoencoder

- the encoder: a set of layers that takes as input the high-dimensionality vector(s) and returns the reduced vector(s).
- the decoder, on the other hand, computes the opposite operation, returning a high-dimensional vector by passing as input the low-dimensional one.

The layers composing the AE could be in principle of any type — e.g. convolutional [32], dense—but in this work both the encoder and the decoder are feed-forward neural networks. For sake of simplicity, we assume here that both the encoder and the decoder are built with only one hidden layer and we denote by $\mathcal{D}$ the decoder and with $\mathcal{E}$ the encoder. If $\boldsymbol{v}^{\text{wake}}$ is the input of the AE, we denote by $\tilde{\boldsymbol{v}}^{\text{wake}} = (\mathcal{D} \circ \mathcal{E})(\boldsymbol{v}^{\text{wake}}) = \text{AE}(\boldsymbol{v}^{\text{wake}})$ the output of the encoder, where formally:

$$\mathcal{E}(\boldsymbol{v}^{\text{wake}}) = \sigma(W\boldsymbol{v}^{\text{wake}} + b) = \hat{\boldsymbol{v}}^{\text{wake}},$$
$$\mathcal{D}(\hat{\boldsymbol{v}}^{\text{wake}}) = \sigma(W'\hat{\boldsymbol{v}}^{\text{wake}} + b') = \tilde{\boldsymbol{v}}^{\text{wake}}.$$

A generic structure of an autoencoder is schematized in Fig. 2.

Weights and activation functions can be different for encoder and decoder, and of course the case with more than one hidden layer for the encoder and the decoder can be easily derived from this formulation. The AE is trained by minimizing the following loss function:

$$\|\boldsymbol{v}^{\text{wake}} - \tilde{\boldsymbol{v}}^{\text{wake}}\|^2 = \|\boldsymbol{v}^{\text{wake}} - \sigma'(W'(\sigma(W\boldsymbol{v}^{\text{wake}} + b)) + b')\|^2,$$

where $\boldsymbol{v}^{\text{wake}}$ is the original (high-dimensional) vector to reduce, $\hat{\boldsymbol{v}}^{\text{wake}}$ represents the reduced coordinates and $\tilde{\boldsymbol{v}}^{\text{wake}}$ is the predicted vector. In this way, the network weights are optimized such that the entire AE produces the approximation of the original vector, but compressing it onto an a priori reduced dimension. The learning step aims so to discover the latent dimension of the problem at hand.

For what concerns the test cases here considered, two different types of autoencoders are taken into account:

(i) a *linear* autoencoder, i.e. without an activation function between the hidden layers, with a single hidden layer composed of a number of neurons equal to the reduced dimension: it should exactly reproduce the behavior of the POD;
(ii) a *non-linear* autoencoder, i.e. with an activation function, and with multiple hidden layers, whose performance is compared to that of the POD.

2.2.2 Approximation Techniques

The problem in the online part is to predict the (unknown) latent dimension $\tilde{\boldsymbol{v}}^{\text{wake}}$ given a new parameter $\boldsymbol{\mu}$:

$$\tilde{\boldsymbol{v}}^{\text{wake}} = \pi(\boldsymbol{\mu}) \quad \text{s.t.} \quad \tilde{\boldsymbol{v}}_i^{\text{wake}} = \pi(\boldsymbol{\mu}_i) \text{ for } i = 1, \ldots, M,$$

where $\pi : \mathbb{R}^p \to \mathbb{R}^L$ is the mapping from parameter space to reduced space. We can approximate such mapping by means of interpolation techniques, such as Radial Basis Functions (RBF), or regression techniques, such as ANN, in order to predict the latent dimension for any new parameter. Finally, the approximation of the solution in the original (high-dimensional) space requires the expansion of the reduced coordinates, which relies on the inverse compression method computed during the dimensionality reduction.

Remark 1 Many approximation techniques can be employed to reconstruct the solution in reduced order models. For instance, two spread techniques are the RBF and the Gaussian Process Regression (GPR). However, we remark that many other approximants can be adopted, such as the Moving Least Squares approach (MLS), which is described in detail in [33, 34]. The reason for choosing the RBF approach is that it allows us to tune different parameters, such as the radius, and the kernel of the radial basis functions, in order to adapt our approximation to different settings of the training dataset.

Radial Basis Functions The RBF is an approximation technique that reconstructs the original field in the following way:

$$\tilde{\boldsymbol{v}}^{\text{wake}} \equiv \tilde{\boldsymbol{v}}^{\text{wake}}(\boldsymbol{\mu}) = \sum_{i=1}^{M} \omega_i \, \boldsymbol{\phi}(\|\boldsymbol{\mu} - \boldsymbol{\mu}_i\|),$$

where $(\boldsymbol{\phi}(\|\boldsymbol{\mu}-\boldsymbol{\mu}_i\|))_{i=1}^{M}$ are called *radial basis functions*, each one associated with a different center $\boldsymbol{\mu}_i$ and weighted with a weight ω_i. The radial functions can have different expressions, in our case we consider the multiquadric functions $\boldsymbol{\phi}(r) = \sqrt{1 + (\varepsilon r)^2}$, where $r = \|\boldsymbol{\mu}-\boldsymbol{\mu}_i\|$.

Artificial Neural Networks The other technique here investigated to approximate the parametric solution manifold is ANN. The basic structure of the method is already explained in Sect. 2.1,

We consider a neural network composed of a unique hidden layer. Its structure is:

$$\text{ANN}(\boldsymbol{\mu}) = \sigma(W\boldsymbol{\mu} + b)$$

The weight matrix and bias of the ANN are found by training the neural network with the set of parameters and snapshots $(\boldsymbol{\mu}_i, \boldsymbol{v}_i^{\text{wake}})_{i=1}^{M}$. Then, the approximated solution $\tilde{\boldsymbol{v}}^{\text{wake}}$ is computed from the related vector of parameters $\boldsymbol{\mu}$ as $\tilde{\boldsymbol{v}}^{\text{wake}} = \text{ANN}(\boldsymbol{\mu})$.

The reduced order techniques presented in this Section both for dimensionality reduction and approximation are applied in this paper to two different inverse problems. In particular, the following cases are considered:

1. POD-RBF;
2. POD-ANN;
3. AE-RBF;
4. AE-ANN; item[5.] non-linear AE-RBF;
6. non-linear AE-ANN.

2.3 Wake Optimization

The construction of the reduced order model is followed by the research of the vector of parameters which better reconstructs the velocity distribution we want to reproduce. This investigation is addressed by solving an optimization problem, in which the aim is to minimize the difference between the approximated wake distribution predicted by the ROM and the real

wake distribution. The optimization problem can be addressed either by using a *search-based*, such as the genetic algorithm (GA), initially proposed in [35], or a *gradient-based* algorithm. In Sects. 3.2.3 and 3.3.2 we compare the results obtained employing both approaches.

Remark 2 However, it is worth remarking that the genetic algorithm allows us to reach the global theoretical minimum without getting stuck into a local minimum. Gradient-based methods, instead, require derivable objective functions and get trapped in local minima in non-convex optimization. The genetic algorithm requires a high number of evaluations, which is not a real issue since the employment of the reduced model, but it is able to converge to the global minimum. In a data-driven ROM framework, as the one proposed in this manuscript, the solution manifold is approximated with regression techniques, without any warranties on convexity. Thus, genetic methods offer a robust approach in this context, as demonstrated by its employment in similar frameworks [25, 36].

We dedicated this section to providing a basic introduction to the genetic method, retaining a full discussion out of the topic of the present work. For a deeper focus on genetic optimization, we refer the reader to the original contribution. The first step of the algorithm is the definition of a population composed of N_{pop} individuals, in our case vectors of parameters $(\boldsymbol{\mu}_j)_{j=1}^{N_{pop}}$, composed of p genes, $\boldsymbol{\mu}_j \in \mathbb{R}^p$. Then, we proceed by defining the objective function which should be minimized. We indicate by $\tilde{\boldsymbol{v}}_j^{\text{wake}} \equiv \tilde{\boldsymbol{v}}^{\text{wake}}(\boldsymbol{\mu}_j)$ the approximation of the wake distribution computed with the reduced order model that we are taking into account. We call $\boldsymbol{v}^{\text{target}}(\boldsymbol{\mu})$ the real wake distribution.

The objective function defined for each individual in the population is:

$$\mathcal{F}(\boldsymbol{\mu}_j) = \|\tilde{\boldsymbol{v}}^{\text{wake}}(\boldsymbol{\mu}_j) - \boldsymbol{v}^{\text{target}}(\boldsymbol{\mu}_j)\|. \tag{4}$$

The GA consists in an iterative process composed of three main steps: *selection*, in which the best individuals are chosen; *mate* or *crossover*, where the genes of the best individuals are combined according to a certain mate probability; *mutation*, changing some of the genes of the individuals. This process is iterated a number of times which is named *number of generations*. Regarding the technical side, the GA has been performed using the open-source package DEAP [37].

3 Numerical Results

In the present Section, the methods presented in Sect. 2 are applied to the test case of the flow in a circular cylinder.

3.1 The Inverse Problem

The computational domain is a circular cylinder with height 4.67 m, diameter 2.36 m and it is schematized in Fig. 4, where the inlet is indicated as Γ_i, the outlet as Γ_o and the lateral surface of the cylinder as Γ_{side}. The aim is to reconstruct the inlet velocity distribution given the velocity distribution at the so-called wake, which is a plane placed at a distance of 2.97 meters from the inlet plane, as showed in 4. This type of problem is known as *inverse problem*, which is in this case applied in a CFD setting. The physical problem at hand is modelled by the Navier-Stokes Equations (NSE) for incompressible flows. We call the fluid domain $\Omega \in \mathbb{R}^3$, Γ its boundary; $t \in [0, T]$ is the time, $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the flow velocity vector

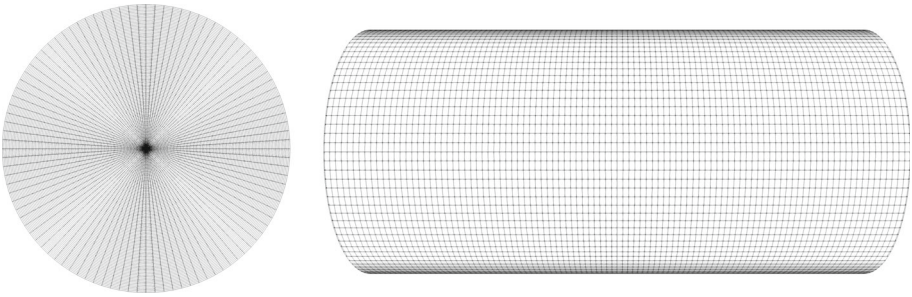


Fig. 3 Representation of the mesh on a slice orthogonal to the cylinder axis (left), and on the walls (right)

field and $p = p(\mathbf{x}, t)$ is the normalized pressure scalar field divided by the fluid density, $\nu = 1.124 \times 10^{-6} m^2/s$ is the fluid kinematic viscosity. The strong form of the NSE is the following.

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} = -\nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \nabla \cdot \nu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) - \nabla p & \text{in } \Omega \times [0, T], \quad (5a) \\ \nabla \cdot \mathbf{u} = \mathbf{0} & \text{in } \Omega \times [0, T], \quad (5b) \\ + \text{ boundary conditions} & \text{on } \Gamma \times [0, T], \quad (5c) \\ + \text{ initial conditions} & \text{in } (\Omega, 0). \quad (5d) \end{cases}$$

The boundary and initial conditions appearing in (5a) are the following:

- at the inlet Γ_i: $\begin{cases} \mathbf{u} = \mathbf{u_0}, \\ \nabla p \cdot \mathbf{n} = 0, \end{cases}$ where $\mathbf{u_0} = \mathbf{u_0}(\mathbf{x}) = (0, f(\mathbf{x}), 0)$ and $f(\mathbf{x})$ is the distribution set at the inlet, specified in (6) and (7).
- at the outlet Γ_o: $\begin{cases} \nabla \mathbf{u} \cdot \mathbf{n} = \mathbf{0}, \\ p = 0. \end{cases}$
- on the walls Γ_{side}: $\begin{cases} \mathbf{u} \cdot \mathbf{n} = 0, \\ \nabla p \cdot \mathbf{n} = 0. \end{cases}$
- moreover, $\mathbf{u}(\mathbf{x}, t) = \mathbf{0}$ and $p(\mathbf{x}, t) = 0, \forall \mathbf{x} \in \Omega$ at initial time $t = 0$.

In the computation of the full order solutions of the NSE in (5a), the finite volume discretization is employed, by means of the open-source software OpenFOAM [38]. The finite volume method [39] is a mathematical technique that converts the partial differential equations (the NSE in our case) defined on differential volumes in algebraic equations defined on finite volumes. The computational mesh considered in this test case has been generated by *blockMesh* and *snappyHexMesh* and it is composed by 1×10^6 cells. The mesh is a regular radial mesh, which presents 100 refinements both in the radial, the tangential and the axial direction, as can be seen from Fig. 3.

The turbulence treatment at the full order level is characterized making use of the RANS (Reynolds Averaged Navier–Stokes) approach. This approach is based on the Reynolds decomposition, which was proposed for the first time by Osborne Reynolds [40], in which each flow field is expressed as the sum of its mean and its fluctuating part. The RANS equations are obtained by taking the time average of the NSE and adding a closure model for the well-known Reynolds stress tensor. The closure model considered in the full order model in

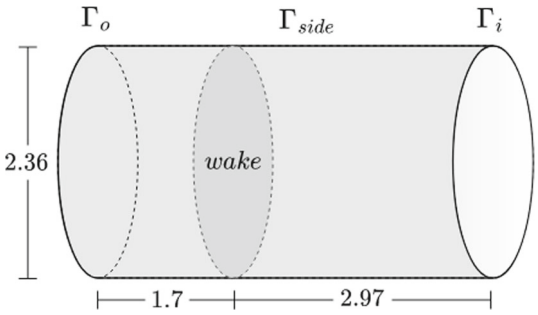


Fig. 4 The computational domain

OpenFOAM is the $\kappa - \omega$ model, proposed in its standard form in [41], and in the SST form in [42]. This model is based on the Boussinesq hypothesis, which models the Reynolds stress tensor as proportional to the so-called *eddy viscosity* and it is based on the resolution of two additional transport equation for the kinetic energy κ and for the specific turbulent dissipation rate ω. In the full order model, we consider the SST $\kappa - \omega$ model. The CFD simulation is run making use of the PIMPLE algorithm until convergence to a steady state (Fig. 4).

In this paper, the inverse problem presented is applied for two different wake distributions. The first one is a smooth function $f : \mathcal{W} \to \mathbb{R}$ defined in all the wake plane $\mathcal{W}$, written as a function of the radial coordinate:

$$f(r) = \sin(4r)^2 - 10. \tag{6}$$

The second distribution is given as a set of pointwise observations only in a selected region of the wake plane. For the sake of simplicity only the distribution along the axis of the cylinder, which is the main direction of the flow, is taken into account. The contributions along the secondary directions are ignored. Thus, we denote with *wake distribution* or *velocity distribution* only the component of the velocity along the main direction of the flow. The observations are computed using the function f defined as

$$f(x, y) = \frac{\sin(\pi x)\sin((y + 0.2)\pi)}{1.2^{e^{x+y}}}, \tag{7}$$

where x and y are the cartesian coordinates in the wake plane. In total we collect 36 observations by sampling with equispaced cartesian grid the domain $[-0.5, 0.5]^2$ in the wake plane.

3.2 First Test Case: A Smooth Distribution

In this Section of the results, the problem considered is the reconstruction of the inlet velocity distribution when the wake velocity has the smooth distribution in (6).

3.2.1 Parametrization Using ANN

The first step in the resolution of the inverse problem is the parametrization of the real velocity distribution at the wake through a fully-connected ANN. The ANN is represented in Fig. 5 and it is composed by 3 hidden layers, with 10, 5 and 3 neurons, respectively. The inputs are the polar coordinates of the wake points and the output is the wake velocity distribution; the degree of freedom of the distribution is $P = 10001$, which corresponds to the number of points at the inlet.

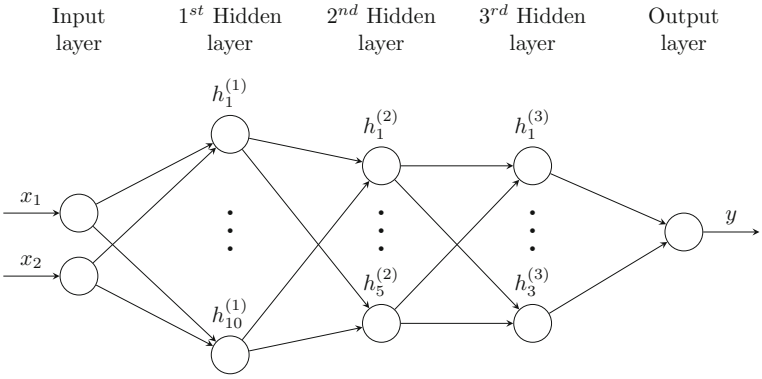


Fig. 5 Structure of the ANN used for parametrization

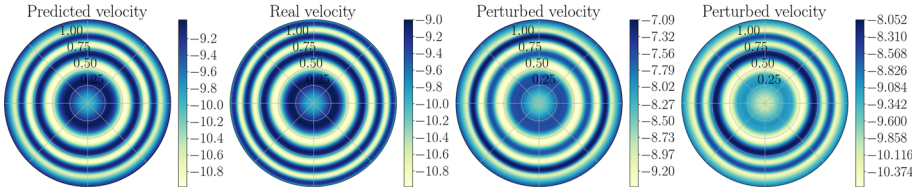


Fig. 6 *First test case*: parametrization of the given target sinusoidal function using a ANN: from left to right are sketched the distribution after the training phase, the original distribution, and two perturbed configurations

As explained in Sect. 2.1, the parameters coincide with a subset of weights of the ANN. In particular, our choice is to consider 4 parameters, given by the biases of the last two layers.

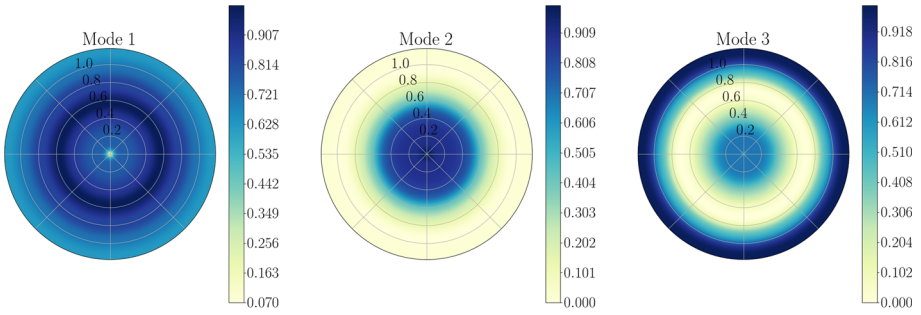
The main idea is to generate M different distributions from the ANN by randomly perturbating the weights considered as parameters. Those distributions are set as inlet distributions in the full order model. The perturbated parameters and the wake distributions obtained from the full order computation are considered as parameters and snapshots, respectively, for the formulation of the ROM.

In Fig. 6, the following distributions are represented from the left to the right: the distribution predicted from the ANN considering the same parameters as in the training stage; the real target wake distribution; two among the M distributions obtained by perturbating the parameters.

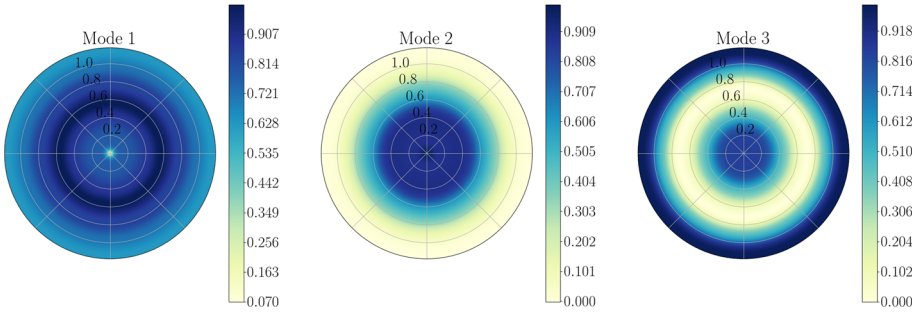
3.2.2 Model Order Reduction

The distributions obtained by the parametrization described in Sect. 3.2.1 are then used as inlet initial conditions to run a set of M offline simulations, following the setting described in Sect. 3.1. The perturbed parameters of the ANN and the wake distributions found from high-fidelity simulations are then used to perform a model reduction. In particular, in this Section different types of techniques for the reduction and approximation stages of the ROM are evaluated and compared and the models considered are here listed:

1. POD-RBF;
2. POD-ANN;
3. AE-RBF;
4. AE-ANN.



(a) POD modes



(b) AE modes without activation function

Fig. 7 The POD and AE modes. Modes in (b) are obtained considering a linear autoencoder

5. Non linear AE-RBF;
6. Non linear AE-ANN.

The details related to the structure of the neural networks and the hyperparameters used for training, i.e. the learning rate, the stopping criteria, the number of hydden layers and of neurons for each layer, are defined in the discussion Sect. 3.4.

First of all, we consider a comparison regarding the dimensionality reduction step. The reduced dimension considered in this test case is 3 and we consider a linear AE, i.e. without any activation function, composed by a single hidden layer of 3 neurons for both the encoder and the decoder parts. The idea is indeed to reproduce the behaviour of the POD. A preliminary study is the graphical comparison between the POD and the AE modes. In particular, the AE modes are computed by taking as input to the decoder part of the network the identity matrix of size 3. Since the AE does not include any non linear computation, the operation is linear and ideally identical to the computation of the first 3 POD modes. From Fig. 7, the shapes of the first 3 POD and AE modes appear similar.

In Fig. 8 we provide a representation of the error in the L^2 norm for: (a) a fixed test dataset; (b) the training dataset used to build each ROM. In this sensitivity analysis, the test dataset is fixed and composed by 10 snapshots and the number of snapshots used to train the reduced models varies between 10 and 90.

Considering a dataset composed by n element, we first define the relative error for each i-th element, say:

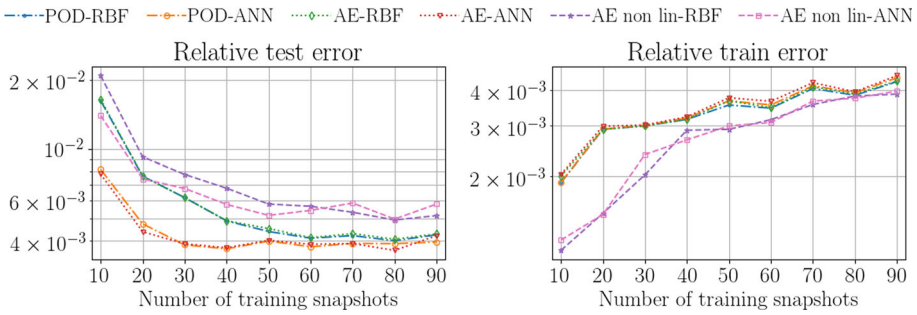


Fig. 8 *First test case*: sensitivity analysis for all the ROMs with the reduced dimension $L = 3$. Error is shown for test (left) and training (right) datasets

$$\varepsilon_i = \frac{\|\tilde{\boldsymbol{v}}_i^{\text{wake}} - \boldsymbol{v}_i^{\text{wake}}\|}{\|\boldsymbol{v}_i^{\text{wake}}\|},$$

where $\tilde{\boldsymbol{v}}_i^{\text{wake}}$ is the velocity distribution predicted by the reduced model starting from the test parameter, and $\boldsymbol{v}_i^{\text{wake}}$ is the full-order snapshot in the test dataset. Then, the mean over all the elements of the test dataset is computed: $(\sum_{i=1}^{n} \varepsilon_i)/n$. This definition applies to both the test and the training datasets and it is used to compute the errors represented in Fig. 8. The neural networks appearing in ROMs are trained 3 times and Fig. 8 provides the mean test and train errors among the 3 different runs, to ensure a more reliable analysis of the ROMs accuracy.

As can be seen from Fig. 8, the results obtained for the test error of POD-RBF and AE-RBF are similar to each other and, as intuitively expected, the general trend is decreasing as the number of training snapshots increases. The reduced methods which include a regression technique for the approximation, i.e. the ANN, produce better results in terms of test error with respect to the interpolating technique. In particular, the regression technique outperforms the classical RBF if we consider a small number of training snapshots; however, the results obtained by the two approximating techniques are similar when a larger training dataset is used to train the ROM.

Remark 3 The behavior of the neural networks (both the AE used for reduction and the ANN used as approximant) strongly depends on the hyperparameters of the networks. This instability occurs also in the case of *nonlinear* AE. Indeed, in this case the training error outperforms the results obtained with the POD, but the test error remains close to the one obtained with a standard POD approach, leading to the *overfitting* phenomenon. A wider discussion on the stability issues related to neural networks will be undertaken in Sect. 3.4.

3.2.3 Wake Optimization

In this Section, two different optimization methods are exploited to find the combination of parameters that provides the highest accuracy in the reconstruction of the original wake. In particular, the results obtained with a genetic algorithm are compared to those obtained with a gradient-based method. Both optimization processes exploit the ROM potentiality to predict the velocity distribution corresponding to each evaluation point in the parameter space. The reduced model taken into account is POD-ANN, which provided the highest precision in the sensitivity analysis in 3.2.2 when the number of snapshots used for training is 90.

The GA, composed by the selection, crossover and mutation steps, is evolved for 20 generations, considering an initial population of 200 individuals, where each individual is

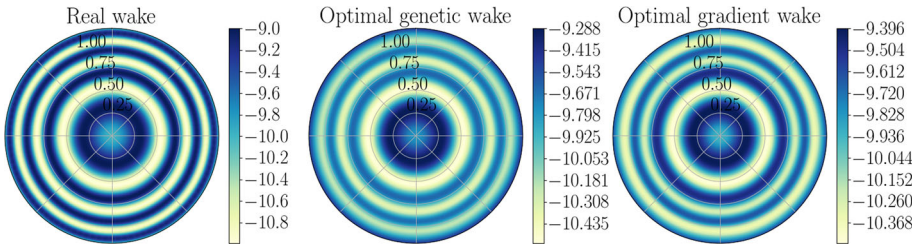


Fig. 9 *First test case*: graphical representations of the exact benchmark wake (left), the optimal wakes obtained with a genetic (center) and gradient (right) optimization

represented by a different combination of the four parameters used for the wake parametrization in Sect. 3.2.1. All the attributes of each individual are in the interval $[-0.5, 0.5]$. The fitness of each individual of the population, i.e. the objective function minimized in the GA, is the relative error of the wake obtained from the ROM with respect to the reference wake we want to reproduce. We specify here more details about the implementation of the genetic evolution:

- in the *mate* stage, a blend crossover is performed, that modifies in-place the input individuals;
- in the *mutation* step, a Gaussian mutation of mean $\mu = 0$ and standard deviation $\sigma = 1$ is applied to the input individual. The independent probability for each attribute to be mutated is set to 0.5;
- the *evolutionary algorithm* employed in the genetic process is the $(\mu + \lambda)$ algorithm, which selects the $\mu = 50$ best individuals for the next generation and produces $\lambda = 100$ children at each generation. Moreover, the probabilities that an offspring is produced by crossover and by mutation are set to 0.4 and 0.6, respectively.

On the other hand, for the gradient optimization we considered the *BFGS* (*Broyden-Fletcher-Goldfarb-Shanno*) algorithm. The starting point chosen in the function evaluation is $[0.5, -0.5, -0.5, 0.5]$. The algorithm evaluated the derivative of the objective function via forward differences, setting the absolute step size to 1×10^{-17}.

The optimal individual obtained from the genetic optimization is $\left[-0.157\ -0.179\ -0.157\ 0.0851\right]$, whereas the optimal individual predicted by the gradient-based algorithm is $\left[0.00611\ -0.117\ -0.0095\ 0.0143\right]$. The final value of the objective function for the optimal individual is $\sim 3.89\,\%$ for the genetic algorithm, and $\sim 3.94\,\%$ for the gradient algorithm. The graphical representations of the optimal and of the real distributions are displayed in Fig. 9.

Remark 4 Although the two methods reach similar precision, the genetic algorithm is preferred since the gradient method is significantly influenced by the choice of the initial guess and by the step size used to evaluate the approximated derivative. We will show the numerical evidence for this argumentation in Sect. 3.4.

3.3 Second Test Case: A Set of Pointwise Observations

In practical applications, the available data is typically provided not in the form of a smooth distribution, but in a small number of pointwise observations, which are usually localized on a reduced part of the computational domain. In this section, the known data at the wake is

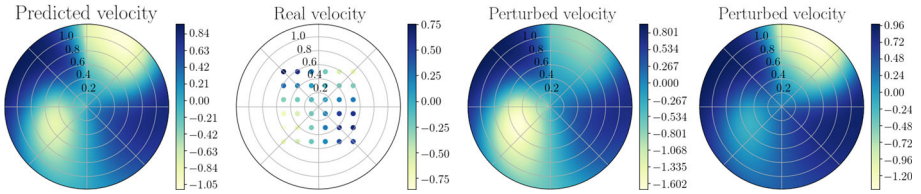


Fig. 10 *Second test case*: parametrization of the given target using a ANN: from left to right are sketched the distribution after the training phase, the original observations, and two perturbed configurations

composed of a set of 36 measurements placed inside a square of side 1 m centered on the wake plane. The reference distribution considered is expressed in (7).

The logical passages followed in Sect. 3.2 are reproduced in the present section with a different set of data.

3.3.1 Parametrization Using ANN

The ANN used to parametrize the wake in this test case is composed of 3 hidden layers, the first one with 8 neurons and the other two with 2 neurons. The number of parameters is 6 in this application and coincides with the perturbations of the weight matrix and the bias of the last hidden layer. The ANN is trained for 50000 epochs with a learning rate of 3×10^{-3}. Since in this numerical experiment there is no radial symmetry (contrary to the previous test case) we add an additional contribution to the loss term to force the continuity between 0 and 2π angles. Formally, the loss to minimize during the train is

$$L(\tilde{\boldsymbol{v}}^{\text{wake}}, \boldsymbol{v}^{\text{target}}) = MSE(\tilde{\boldsymbol{v}}^{\text{wake}}, \boldsymbol{v}^{\text{target}}) + \lambda \|\text{ANN}(r_i, 0) - \text{ANN}(r_i, 2\pi)\|, \tag{8}$$

where $MSE(\cdot, \cdot)$ is the mean square error, λ is a weight of the additional contribution (here $\lambda = 1$) and r_i for $i = 1, \ldots, 100$ are the equispaced sample points we spanned along the radius at the angles 0 and 2π. Figure 10 shows indeed a continuous surface that keeps such a feature also after the perturbation of the weights.

As already done in Sect. 3.2.1, M sets of parameters are generated as random perturbations in the interval $[-1, 1]$, which will be used as parameters of the ROM. The ANN perturbed with the generated parameters is used to obtain M new velocity distributions, which are set as inlet distributions for the full-order model. Then, from the full-order simulations, the wake distributions corresponding to each set of parameters are obtained and considered as snapshots for the ROM (Fig. 11).

3.3.2 Model Order Reduction and Optimization

From the sensitivity analysis in Sect. 3.2.2 the combination of the AE and the ANN proved to have the highest precision in the reconstruction of the wake. For this reason, the AE-ANN is the ROM which is chosen in this test case to train the optimization algorithm and to provide the best (approximated) wake.

The GA, composed by the selection, mate, and mutation, is here iterated for 20 generations. The number of individuals considered is 200 for the first generation and 100 for all the other generations. The fitness is the relative error between the real wake we want to reproduce and the wake reconstructed by the ROM, where only the points in data are considered. In this experiment, the fitness decreases as the population evolves, until a percentage error of

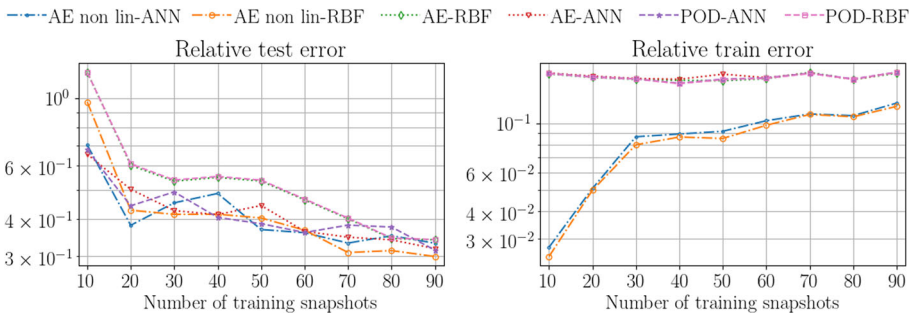


Fig. 11 *Second test case*: sensitivity analysis for all the ROMs with the reduced dimension $L = 4$. Error is shown for the testing (left) and training (right) datasets

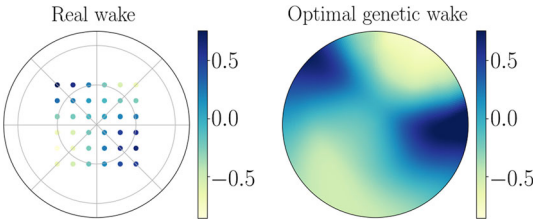


Fig. 12 *Second test case*: the target output at wake plane (left) and the optimum representation, obtained with a genetic algorithm (right)

$\simeq 49\%$ is reached. Also in this case, the *BFGS* technique is not able to reach that fitness and gets blocked instead in some local minima at 61% of error with respect to the target wake.

Figure 12 shows the graphical representation of the optimum wake obtained from the GA. From a graphical point of view, the optimization is able to capture the trend exhibited by the pointwise measurements. However, the error remains high after the optimization since the data in this test case is not given by a target smooth distribution, but by a few amount of points. Another possible issue is that the points are given only in a limited region of the wake, leading to a more difficult reconstruction of the distribution at the inlet, especially in the external region.

3.4 Discussion

This Section is dedicated to a wider discussion of the numerical results obtained in Sects. 3.2 and 3.3.

Table 1 provides a summary of the results obtained for the two test cases. The first thing that emerges from the table is that none of the methods performs better than the others for each different value of the training snapshots. In general, for the first test case, the methods that provide the highest accuracy on the test snapshots are the POD-ANN and the linear AE-ANN, as can be seen also from Fig. 8. On the other hand, in the second test case, the reduced order methods employing the nonlinear autoencoder provide the best results in most of the M values, as can be seen in Fig. 11. For what concerns the training error, from both Figs. 8 and 11 we can deduce that the introduction of non-linearity in the reduction stage leads to a better reconstruction of the training snapshots, especially when M is small, i.e. with a limited amount of training data. Therefore, especially in the first test case, the autoencoder reaches a high accuracy on the train snapshots, but it is not able to generalize the information learned during training, leading to high errors on the test database. Indeed, the worst results

Table 1 Summary of results

		First test case	Second test case
Number of parameters p		4	6
Reduced dimension L		3	4
10 train snapshots	Best method	Linear AE-ANN	Linear AE-ANN
	Mean test error	0.781%	65.32%
50 train snapshots	Best method	POD-ANN	Non linear AE-ANN
	Mean test error	0.399%	36.82%
90 train snapshots	Best method	POD-ANN	Non linear AE-RBF
	Mean test error	0.395%	29.95%
Optimal fitness	Genetic	3.89%	49.83%
	Gradient	3.94%	61.72%

in the first case are obtained with the nonlinear AE and are highlighted in red in Table 2. This behavior describes the *overfitting* phenomenon, which is here mitigated by the addition of a regularization term in the loss function, say:

$$L(\tilde{\boldsymbol{v}}^{\text{wake}}, \boldsymbol{v}^{\text{target}}) = MSE(\tilde{\boldsymbol{v}}^{\text{wake}}, \boldsymbol{v}^{\text{target}}) + \frac{\alpha}{2}\|\mathbf{W}\|^2,$$

where $\|\mathbf{W}\|$ is the L^2 norm of the weight matrix, i.e. the sum over all the squared weight values, and α is named *weight decay*.

In addition, we provide here different arguments to explain the stability issues concerning the neural networks' performance:

- The networks are highly influenced by the choice of the hyper-parameters, i.e. the learning rate, the structure of the hidden layers, the stopping criteria, and the weight decay. In order to have a good result for each value of M, the value of the hyperparameters should be tuned *ad-hoc* considering each case individually. In our analysis, instead, we consider the parameters fixed for each method for a fair comparison. These values are reported in Appendix 1.
- The different performance of the autoencoder in the two test cases depends on the input data. We obtained better results in the second test case, where the POD has poor accuracy and is not able to capture the system's behavior. To support this argumentation, we provide the results obtained in the first test case when $L = 1$. From Fig. 13 we can notice that the autoencoder improves the results obtained with the POD when M is low (Table 3).

An additional point that can be discussed is the type of optimization algorithm used. As pointed out in Sect. 3.2.3, the genetic algorithm is preferred to a gradient-based algorithm, because the gradient method can get stuck into local minima. The evidence of this statement is provided in Table 4, where we show the results obtained with the *BFGS* method starting from different initial guesses. In the first three cases analyzed, the method converges to three different local minima; in the last two cases, it gets stuck in the first evaluation point.

Table 2 *First test case:* stability analysis on the percentage test errors of the ROMs when $L = 3$

M	POD-RBF		POD-ANN		linear AE-RBF		linear AE-ANN		nonlinear AE-RBF		nonlinear AE-ANN	
	min	max	min	max	min	max	min	max	min	max	min	max
10	1.62	1.62	0.78	0.87	1.61	1.69	0.76	0.81	2.04	2.14	1.24	1.65
20	0.76	0.76	0.42	0.57	0.76	0.76	0.43	0.45	0.9	0.94	0.64	0.83
30	0.62	0.62	0.38	0.38	0.62	0.62	0.39	0.39	0.77	0.78	0.66	0.69
40	0.49	0.49	0.37	0.37	0.49	0.49	0.37	0.38	0.63	0.71	0.57	0.58
50	0.44	0.44	0.40	0.40	0.44	0.48	0.40	0.40	0.54	0.61	0.46	0.60
60	0.41	0.41	0.38	0.38	0.41	0.42	0.38	0.41	0.55	0.60	0.49	0.59
70	0.42	0.42	0.36	0.44	0.42	0.43	0.37	0.41	0.51	0.57	0.54	0.63
80	0.40	0.40	0.36	0.41	0.40	0.42	0.36	0.37	0.46	0.53	0.45	0.53
90	0.43	0.43	0.39	0.41	0.43	0.43	0.41	0.45	0.49	0.54	0.53	0.61

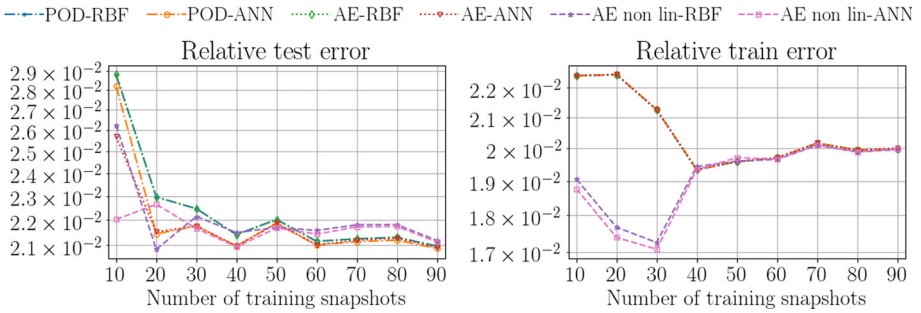


Fig. 13 *First test case*: sensitivity analysis for all the ROMs with the reduced dimension $L = 1$. Error is displayed for test (left) and training (right) datasets

Table 3 *Second test case*: stability analysis on the percentage test errors of the ROMs, when $L = 4$

M	POD-RBF		POD-ANN		linear AE-RBF		linear AE-ANN		nonlinear AE-RBF		nonlinear AE-ANN	
	min	max	min	max	min	max	min	max	min	max	min	max
10	122.08	122.08	56.46	80.45	122.34	122.38	58.16	75.26	92.35	101.82	54.26	100.21
20	60.80	60.80	40.72	47.85	60.13	60.24	42.16	55.72	41.07	44.48	30.32	46.01
30	53.78	53.78	41.54	52.91	53.29	53.30	38.26	48.82	40.52	41.93	37.53	49.45
40	55.25	55.25	37.16	43.01	54.76	55.04	38.42	46.34	38.96	44.21	45.40	52.15
50	53.73	53.73	35.96	41.43	53.30	53.37	40.07	47.69	36.77	42.99	32.15	40.05
60	46.49	46.49	33.44	38.89	46.17	46.22	33.94	38.85	35.91	37.16	31.84	42.95
70	40.20	40.20	37.17	39.85	39.93	40.00	30.06	37.57	28.92	33.38	25.91	39.48
80	34.52	34.52	33.99	43.17	34.35	34.39	33.56	34.85	30.09	32.61	29.09	43.84
90	34.10	34.10	28.99	35.16	34.03	34.04	29.24	35.51	28.60	30.67	27.26	37.48

Table 4 *First test case*: results of gradient optimization for different initial guesses

N. test	Function evaluations	Gradient evaluations	Final fitness (%)	Final point
1	85	17	3.94	$[0.01\ -0.12\ -0.01\ 0.01]$
2	236	47	3.98	$[-0.1\ -0.06\ -0.18\ 0.1]$
3	252	48	4.03	$[-0.33\ 0.03\ -0.1\ 0.1]$
4	5	1	4.25	$[-0.1\ 0.1\ -0.1\ 0.1]$
5	5	1	5.36	$[0.1\ 0.1\ 0.1\ 0.1]$

Table 5 *Second test case*: results of gradient optimization for different initial guesses

N. test	Function evaluations	Gradient evaluations	Final fitness (%)	Final point
1	467	65	110.30	$[-0.047\ -0.045\ -0.23\ -0.27\ 0.108\ 0.55]$
2	536	75	61.72	$[0.27\ -0.54\ 0.107\ 0.44\ 0.35\ -0.53]$
3	512	72	69.42	$[0.25\ 0.062\ -0.39\ 0.17\ -0.42\ 0.33]$
4	719	101	74.8	$[0.03\ -0.11\ -0.097\ 0.25\ 0.49\ -0.27]$
5	382	53	79.38	$[-0.096\ 0.46\ 0.28\ -0.027\ 0.058\ -0.49]$

4 Conclusion

We presented in this paper a data-driven computational pipeline that allows us to efficiently face inverse problems by means of the model order reduction technique. Neural networks are involved in the such framework at three different levels:

- The parametrization of a given (generic) wake distribution by perturbating a subset of the weights of a fully-connected neural network. The parameters are exploited to obtain the inlet distributions of the full-order simulations. The wake distributions obtained from the full-order simulations are considered snapshots associated with the input parameters, obtaining a parametrization of the original problem (Table 5).
- The compression of the dimensionality of the full-order snapshots, which typically belong to high-dimensional spaces, obtained with a properly structured autoencoder;
- The approximation of the solution manifold, allowing for predicting new solutions for any unseen parameters. This last task is obtained with a second fully-connected neural network.

The results obtained by both the test cases in this work are promising. The neural network parametrization shows the possibility to produce several different distributions playing with very few parameters (4 and 6), becoming a valuable tool also for such an objective. Of course, the benefits of such parametrization are strictly related to the problem at hand, since here we are assuming that the target distribution is somehow related to the boundary condition. The investigation of the parametric configuration shows also a large range of admissible distributions. However, we highlight there is no possibility to precisely select the region of interest for the parametrization and the weights of the neural network which have been perturbated have no physical significance linked to the wake velocity distribution. Dimensionality compression and manifold approximation are already investigated uses of neural networks, and also in this contribution, the results confirm their better performances with respect to the more consolidated techniques.

Possible future developments of the presented pipeline may interest the parametrization through neural networks: at the current stage, it results in a sort of *black-box* procedure, since the parameters and their ranges are selected following a trial and error process. Such an issue can be alleviated by looking at the intermediate output—the output of the hidden layers of the network—to propose a meaningful criterion for parameter selection.

Author Contributions Conceptualization: AI, ND; Methodology: AI, ND; Formal analysis and investigation: AI; Writing—original draft preparation: AI; Writing—review and editing: ND, GR; Funding acquisition: GR; Supervision: GR.

Funding Open access funding provided by Scuola Internazionale Superiore di Studi Avanzati - SISSA within the CRUI-CARE Agreement. This work was partially funded by INdAM-GNCS 2020-2021 projects, by European High-Performance Computing Joint Undertaking project Eflows4HPC GA N. 955558, by PRIN "Numerical Analysis for Full and Reduced Order Methods for Partial Differential Equations" (NA-FROM-PDEs) project by European Union Funding for Research and Innovation—Horizon 2020 Program—in the framework of European Research Council Executive Agency: H2020 ERC CoG 2015 AROMA-CFD project 681447 "Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics" P.I. Professor Gianluigi Rozza.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest No potential competing interest was reported by the authors.

Appendix

In this section, we report the specifications of the hyperparameters of the neural networks used to build the ROMs in the first and second test cases (Tables 6 and 7).

Table 6 *First test case:* neural networks setting in ROMs

NN	Structure	Non-linearity	Learning rate	Stop criteria		Weight decay
				Epochs	Final loss	
ANN (POD-ANN)	[4, 4]	Softplus	5×10^{-3}	200,000	1×10^{-3}	0
ANN (lin AE-ANN)	[4, 4]	Softplus	5×10^{-3}	100,000	1×10^{-4}	0
ANN (non lin AE-ANN)	[40, 40]	Softplus	5×10^{-3}	100,000	1×10^{-5}	0
AE (linear)	[3, 3]	None	1×10^{-3}	1000		0
AE (non linear)	[200, 3, 3, 200]	Leaky ReLU	5×10^{-4}	5000	5×10^{-4}	5×10^{-4}

Table 7 *Second test case:* neural networks setting in ROMs

NN	Structure	Non-linearity	Learning rate	Stop criteria		Weight decay
				Epochs	Final loss	
ANN	[40, 20, 10]	Softplus	2×10^{-3}		0.1	0
AE (linear)	[4, 4]	None	3×10^{-3}	5000		0
AE (non linear)	[200, 4, 4, 200]	Leaky ReLU	3×10^{-4}	2000		5×10^{-4}

References

1. Cetrangolo, A.: Reduced order methods for inverse problem in CFD. Master Thesis, Politecnico di Torino (2021)
2. Richter, M.: Inverse Problems: Basics, Theory and Applications in Geophysics. Springer, Switzerland (2020). https://doi.org/10.1007/978-3-319-48384-9
3. Salmoiraghi, F., Scardigli, A., Telib, H., Rozza, G.: Free-form deformation, mesh morphing and reduced-order methods: enablers for efficient aerodynamic shape optimisation. Int. J. Comput. Fluid Dyn. **32**(4–5), 233–247 (2018). https://doi.org/10.1080/10618562.2018.1514115
4. Quarteroni, A., Manzoni, A., Negri, F.: Reduced Basis Methods for Partial Differential Equations: An Introduction, 1st edn., p. 296. Springer, Switzerland (2015). https://doi.org/10.1007/978-3-319-22470-1
5. Hesthaven, J.S., Rozza, G., Stamm, B.: Certified Reduced Basis Methods for Parametrized Partial Differential Equations, 1st edn., p. 135. Springer, Switzerland (2015)
6. Rozza, G., Hess, M., Stabile, G., Tezzele, M., Ballarin, F.: Basic ideas and tools for projection-based model reduction of parametric partial differential equations. In: Benner, P., Grivet-Talocia, S., Quarteroni, A., Rozza, G., Schilders, W.H.A., Silveira, L.M. (eds.) Model Order Reduction, vol. 2, pp. 1–47. De Gruyter, Berlin (2020). https://doi.org/10.1515/9783110671490-001
7. Morelli, U.E., Barral, P., Quintela, P., Rozza, G., Stabile, G.: A numerical approach for heat flux estimation in thin slabs continuous casting molds using data assimilation. Int. J. Numer. Meth. Eng. **122**(17), 4541–4574 (2021). https://doi.org/10.1002/nme.6713
8. Bui-Thanh, T., Damodaran, M., Willcox, K.: Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. AIAA J. **42**(8), 1505–1516 (2004). https://doi.org/10.2514/1.2159

9. Lassila, T., Manzoni, A., Quarteroni, A., Rozza, G.: A reduced computational and geometrical framework for inverse problems in hemodynamics. Int. J. Numer. Methods Biomed. Eng. **29**(7), 741–776 (2013). https://doi.org/10.1002/cnm.2559
10. Li, J., Marzouk, Y.M.: Adaptive construction of surrogates for the Bayesian solution of inverse problems. SIAM J. Sci. Comput. **36**(3), 1163–1186 (2014). https://doi.org/10.1137/130938189
11. Vitale, G., Preziosi, L., Ambrosi, D.: Force traction microscopy: an inverse problem with pointwise observations. J. Math. Anal. Appl. **395**(2), 788–801 (2012). https://doi.org/10.1016/j.jmaa.2012.05.074
12. Huang, C.-H., Chen, C.-W.: A boundary element-based inverse-problem in estimating transient boundary conditions with conjugate gradient method. Int. J. Numer. Methods Eng. **42**(5), 943–965 (1998). https://doi.org/10.1002/(SICI)1097-0207(19980715)42:5<943::AID-NME395>3.0.CO;2-V
13. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989). https://doi.org/10.1016/0893-6080(89)90020-8
14. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019). https://doi.org/10.1016/j.jcp.2018.10.045
15. Chen, W., Wang, Q., Hesthaven, J.S., Zhang, C.: Physics-informed machine learning for reduced-order modeling of nonlinear problems. J. Comput. Phys. **446**, 110666 (2021). https://doi.org/10.1016/j.jcp.2021.110666
16. Hesthaven, J.S., Ubbiali, S.: Non-intrusive reduced order modeling of nonlinear problems using neural networks. J. Comput. Phys. **363**, 55–78 (2018). https://doi.org/10.1016/j.jcp.2018.02.037
17. Lee, K., Carlberg, K.T.: Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. J. Comput. Phys. **404**, 108973 (2020). https://doi.org/10.1016/j.jcp.2019.108973
18. Pichi, F., Ballarin, F., Rozza, G., Hesthaven, J.S.: Artificial neural network for bifurcating phenomena modelled by nonlinear parametrized PDEs. PAMM **20**(S1), 202000350 (2021). https://doi.org/10.1002/pamm.202000350
19. Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat. Mach. Intell. **3**(3), 218–229 (2021). https://doi.org/10.1038/s42256-021-00302-5
20. Wang, S., Wang, H., Perdikaris, P.: Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets. Sci. Adv. (2021). https://doi.org/10.1126/sciadv.abi8605
21. Papapicco, D., Demo, N., Girfoglio, M., Stabile, G., Rozza, G.: The neural network shifted-proper orthogonal decomposition: a machine learning approach for non-linear reduction of hyperbolic equations. Comput. Methods Appl. Mech. Eng. **392**, 114687 (2022). https://doi.org/10.1016/j.cma.2022.114687
22. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
23. Rojas, R.: The backpropagation algorithm. In: Neural Networks, pp. 149–182. Springer, Berlin (1996)
24. Tezzele, M., Demo, N., Mola, A., Rozza, G.: An integrated data-driven computational pipeline with model order reduction for industrial and applied mathematics. Mathematics in Industry. In: Günther, M., Schilders, W. (eds.) Novel Mathematics Inspired by Industrial Challenges. Springer, Switzerland (2022)
25. Demo, N., Ortali, G., Gustin, G., Rozza, G., Lavini, G.: An efficient computational framework for naval shape design and optimization problems by means of data-driven reduced order modeling techniques. Bollettino dell'Unione Matematica Italiana (2020). https://doi.org/10.1007/s40574-020-00263-4
26. Tezzele, M., Demo, N., Stabile, G., Mola, A., Rozza, G.: Enhancing CFD predictions in shape design problems by model and parameter space reduction. Adv. Model. Simul. Eng. Sci. (2020). https://doi.org/10.1186/s40323-020-00177-y
27. Georgaka, S., Stabile, G., Star, K., Rozza, G., Bluck, M.J.: A hybrid reduced order method for modelling turbulent heat transfer problems. Comput. Fluids **208**, 104615 (2020). https://doi.org/10.1016/j.compfluid.2020.104615
28. Dutta, S., Farthing, M.W., Perracchione, E., Savant, G., Putti, M.: A greedy non-intrusive reduced order model for shallow water equations. J. Comput. Phys. **439**, 110378 (2021)
29. Zancanaro, M., Mrosek, M., Stabile, G., Othmer, C., Rozza, G.: Hybrid neural network reduced order modelling for turbulent flows with geometric parameters. Fluids **6**(8), 296 (2021). https://doi.org/10.3390/fluids6080296
30. Girfoglio, M., Scandurra, L., Ballarin, F., Infantino, G., Nicolo, F., Montalto, A., Rozza, G., Scrofani, R., Comisso, M., Musumeci, F.: Non-intrusive data-driven ROM framework for hemodynamics problems. Acta. Mech. Sin. **37**(7), 1183–1191 (2021). https://doi.org/10.1007/s10409-021-01090-2
31. Demo, N., Tezzele, M., Rozza, G.: EZyRB: easy reduced basis method. J. Open Source Softw. **3**(24), 661 (2018). https://doi.org/10.21105/joss.00661
32. Romor, F., Stabile, G., Rozza, G.: Non-linear manifold rom with convolutional autoencoders and reduced over-collocation method. arXiv preprint arXiv:2203.00360 (2022)

33. Levin, D.: The approximation power of moving least-squares. Math. Comput. **67**(224), 1517–1531 (1998)
34. Lancaster, P., Salkauskas, K.: Surfaces generated by moving least squares methods. Math. Comput. **37**(155), 141–158 (1981)
35. Holland, J.H.: Genetic algorithms and the optimal allocation of trials. SIAM J. Comput. **2**(2), 88–105 (1973). https://doi.org/10.1137/0202009
36. Demo, N., Tezzele, M., Mola, A., Rozza, G.: Hull shape design optimization with parameter space and model reductions, and self-learning mesh morphing. J. Mar. Sci. Eng. (2021). https://doi.org/10.3390/jmse9020185
37. Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., Gagné, C.: DEAP: evolutionary algorithms made easy. J. Mach. Learn. Res. **13**, 2171–2175 (2012)
38. OpenFOAM website. https://openfoam.org/
39. Moukalled, F., Mangani, L., Darwish, M.: The Finite Volume Method in Computational Fluid Dynamics, vol. 113. Springer, Switzerland (2016). https://doi.org/10.1007/978-3-319-16874-6
40. Reynolds, O.: IV. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. Philos. Trans. R. Soc. Lond. **186**, 123–164 (1895). https://doi.org/10.1098/rsta.1895.0004
41. Kolmogorov, A.N.: Equations of turbulent motion in an incompressible fluid. In: Dokl. Akad. Nauk SSSR, vol. 30, pp. 299–303 (1941)
42. Menter, F.R.: Two-equation eddy-viscosity turbulence models for engineering applications. AIAA J. **32**(8), 1598–1605 (1994). https://doi.org/10.2514/3.12149