Hijazi *et al. Advanced Modeling and Simulation in Engineering Sciences(2023)10:5*
https://doi.org/10.1186/s40323-023-00242-2

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

POD-Galerkin reduced order models and physics-informed neural networks for solving inverse problems for the Navier–Stokes equations

Saddam Hijazi[1*], Melina Freitag[1] and Niels Landwehr[2]

*Correspondence:
saddam.hijazi@uni-potsdam.de
[1]Institute of Mathematics, University of Potsdam, Karl-Liebknecht-Str. 24-25, 14476 Potsdam, Germany
[2]Institute of Computer Science, University of Hildesheim, Universitätsplatz 1, 31141 Hildesheim, Germany

Abstract

We present a Reduced Order Model (ROM) which exploits recent developments in Physics Informed Neural Networks (PINNs) for solving inverse problems for the Navier–Stokes equations (NSE). In the proposed approach, the presence of simulated data for the fluid dynamics fields is assumed. A POD-Galerkin ROM is then constructed by applying POD on the snapshots matrices of the fluid fields and performing a Galerkin projection of the NSE (or the modified equations in case of turbulence modeling) onto the POD reduced basis. A *POD-Galerkin PINN ROM* is then derived by introducing deep neural networks which approximate the reduced outputs with the input being time and/or parameters of the model. The neural networks incorporate the physical equations (the POD-Galerkin reduced equations) into their structure as part of the loss function. Using this approach, the reduced model is able to approximate unknown parameters such as physical constants or the boundary conditions. A demonstration of the applicability of the proposed ROM is illustrated by three cases which are the steady flow around a backward step, the flow around a circular cylinder and the unsteady turbulent flow around a surface mounted cubic obstacle.

Keywords: Proper orthogonal decomposition, Inverse problems, Physics-based machine learning, Navier–Stokes equations

Introduction and literature overview

In recent decades, research into numerical methods for solving systems of Partial Differential Equations (PDEs) has been growing rapidly. Popular methods include the finite difference (FDM), the finite element (FEM), the finite volume (FVM), and the spectral element method (SEM). However, running computational simulations using those numerical methods can be very expensive, especially in high dimensions. The situation becomes worse when simulations have to be run several times with several different input configurations (as in repetitive computational environment). These common settings can be observed in various fields such as Uncertainty Quantification (UQ), sensitivity analysis,

real-time control problems, optimization, prediction and parameter estimation/inference. In such circumstances, running simulations using the classical numerical methods for each different input value could be deemed prohibitive. Therefore, numerical techniques which could bring a reduction in the computational cost are needed. Reduced Order Methods (ROMs) represent a suitable tool for achieving the goal of having computational speed-up and providing accurate solutions to the problems of interest. These methods have been applied to a variety of mathematical problems, for greater details on ROMs we refer the reader to [1–5]. In this article we focus on ROMs for fluid dynamics problems in the context of the reduction of the Navier–Stokes Equations (NSE).

In reduced order modeling, projection-based ROMs [6,7] represent a popular technique for the construction of surrogate reduced models, these ROMs have been applied in several fields such as civil engineering, aerospace engineering and nuclear engineering. The reduction in the projection-based ROMs is achieved by finding the reduced solution which lies in a subspace of a much smaller dimension N_r, $N_r \ll N_h$, where N_h is the dimension of the original space constructed by the Full Order Model (FOM). The dimension of the FOM (i.e. N_h) represents the number of unknowns or degrees of freedom in the discretized problem. In a projection-based ROM, there are two main ingredients: (i) low dimensional spaces called the reduced spaces which are often generated using a set of snapshots (which are FOM solutions obtained for different values of time and/or parameter) and (ii) a Galerkin or a Petrov–Galerkin projection for the construction of a low dimensional $N_r \times N_r$ problem whose solution is the ROM one. In the context of Parameterized PDEs (PPDEs), projection-based ROMs have been exploited for achieving a solution-space reduction by relying on greedy algorithms [8,9] or Proper Orthogonal Decomposition (POD) [10–14] to generate the reduced space. The application of the POD method together with a Galerkin projection technique results in the so-called POD-Galerkin ROM. This type of ROMs has been used extensively for the reduction of PPDEs, for more details on POD-Galerkin ROMs we refer the reader to [11–13,15–17].

There are several challenges for the construction of efficient POD-Galerkin ROMs for the Navier–Stokes equations. The treatment of the turbulence phenomenon at both the FOM and the ROM levels is one of them. In this work, turbulence is tackled at the full order level through modeling strategies. In other words, turbulent flows are not solved using the Direct Numerical Simulations (DNS) approach because of the enormous computational resources needed to simulate these flows for the problems of interest. In particular, turbulence modeling is done with the help of the Reynolds Averaging Navier–Stokes (RANS) [18] equations and the Large Eddy Simulations (LES) [19,20] approaches. The RANS approach is based on solving the NSE for the time-averaged part of the fluid fields, where it basically assumes that time fluctuations are of no significant interest. On the other hand, the LES approach is based on filtering the Navier–Stokes equations to some scale, then the large scales are simulated while the small scales are modeled.

Beside the issue of turbulence, the treatment at the reduced order level of the nonlinear convective term in the NSE is important and might affect the efficiency of the ROM. In several contributions, the ROM formulation approximates the nonlinear term using a third order tensor [21–24], this tensor has a dimension of $N_u \times N_u \times N_u$, where N_u is the number of reduced velocity unknowns. However, this approach of dealing with the nonlinear term could raise the computational burden when the number of reduced unknowns increases. Hyper-reduction techniques could be used for the approximation of

the nonlinear term such as the EIM or DEIM methods [25,26]. In order to implement EIM or DEIM, an expensive pre-processing is required to obtain a version of parameterized operators [27]. The Gappy-POD approach also can be employed [28].

Reduced order models have been also constructed using data-driven techniques as in [29–35]. In these ROMs, the identification of the reduced solutions is done using data-driven approaches such as regression-based methods, interpolation techniques or Neural Networks (NNs). In addition, hybrid reduced order models which merge projection-based ROMs and data-driven ROMs have been proposed [24,36–41]. The latter ROMs include the use of calibration methods, the introduction of correction terms which can be approximated by the snapshots data, and the employment of data-driven techniques for the approximation of only the turbulent/eddy viscosity in the case of turbulent flows. Deep learning approaches have also been used in ROMs in order to perform nonlinear dimensionality reduction [42–44].

In recent years, several contributions have aimed at using machine learning techniques for solving PDEs arising from physical problems. This field of scientific computing is often termed as physics-based machine learning. We give a brief overview of related works in this field. Early work in [45] presents neural minimization algorithms for solving differential equations. The work in [46] presents an approach for solving ODEs and PDEs using feedforward neural networks which is based on approximating the solution function by a trial function that has two parts. The first part which has no tunable parameters satisfies the initial/boundary conditions, while the second part contains all the adjustable parameters which are determined by training the feedforward neural network. The construction of the second term is made in a way that guarantees no contribution to the initial/boundary conditions.

Physics-Informed Neural Networks (PINNs) have been proposed in [47] for solving general nonlinear PDEs as well as inverse problems which involve PDEs. The approach in [47] consists of approximating the solution function of the general PDEs by deep neural networks, the trainable parameters of these NNs are then learned by minimizing a loss function that takes into consideration initial/boundary data and at the same time penalizes the departure from the equations which model the physical problem. The PINNs are based on two different approaches, namely continuous time and discrete time models. The continuous model PINNs allow to infer the solution of the PDE across all time and space. On the other hand, PINNs with the discrete time model approach employ implicit Runge–Kutta time stepping schemes with unlimited number of stages for the prediction of the solution at large time steps without compromising the accuracy of the approximation.

The PINNs presented in [47] were extended to coupled multi-physics problems in [48], where the latter work presents another application of PINNs for solving inverse problems for a Fluid Structure Interaction (FSI) problem. In [49], the authors present a Deep Galerkin Method (DGM) for the approximation of high-dimensional PDEs with a deep neural network, where they solve high-dimensional free boundary PDEs in 200 dimensions. PINNs for solving the Reynolds-averaged Navier–Stokes equations for incompressible turbulent flows are proposed in [50].

A recent work [51] proposes a reduced basis method based on the use of PINNs for solving PPDEs. This work shows that training the PINNs by only minimizing the loss function that corresponds to the reduced equations does not give as accurate results as the ones obtained by the projection of FOM solution onto the reduced space. The authors

indicate that for complex nonlinear problems, the PINNs trained only on the reduced equations are not accurate in approximating the original high fidelity solution. This is justified by the fact that the reduced equations do not take into account the impact of the truncated modes on the resolved ones. On the other hand, the authors demonstrate that the PINNs trained on both the output data labels and the reduced equations are more accurate.

The work in this paper aims at employing reduced order modeling coupled with PINNs for solving inverse problems. By solving an inverse problem, we attempt to infer unknown inputs or parameters from a given set of observations of the output (output data). For a review on inverse problems, we refer the reader to [52,53]. Research into inverse problems in a Bayesian setting has been conducted extensively, for example see [54–58]. Reduced order methods and dimension reduction techniques have been used previously for the estimation of unknown parameters in inverse problems. The work [59] presents a Bayesian approach for solving nonlinear inverse problems with the help of a Galerkin projection. In [60], stochastic reduced order models were proposed for solving inverse problems. Active subspaces were utilized for the reduction of the parameter space in a UQ problem for turbulent combustion simulations [61]. In [62], a hybrid data-driven/projection-based reduced order model is proposed for the Bayesian solution of inverse problems. The work in [63] proposes a nonlinear reduced order model for large-scale inverse problems in a Bayesian inference setting. Another approach [64] combines the nonlinear Landweber method with adaptive online reduced basis updates for solving the inverse problem related to the construction of the conductivity in the steady heat equation. The authors in [65] present a reduction approach of a parameterized forward model, which is utilized for obtaining a surrogate model in a Bayesian inverse problem setting, the inversion is done during the online stage by using the surrogate model constructed via the projection of the forward model onto the reduced spaces.

Here, we present a model for solving inverse problems in a reduced order setting. The approach is based on integrating the structure of the POD-Galerkin ROMs into physics-informed neural networks (PINNs). In particular, we propose to incorporate the POD-Galerkin reduced order equations into the loss function of the PINNs. Consequently, the task of inferring any parameter or physical unknown which is present in the FOM equations will become feasible, thanks to the possibility of introducing additional parameters in the neural networks and making them trainable at low computational cost. The unknown parameters could be physical constants such as the physical viscosity or boundary/initial conditions or the velocity at the inlet in inlet/outlet fluid problems. The approach developed in this work is termed *POD-Galerkin PINN ROM* and is based on assimilating reduced simulated data into the physical model represented by the POD-Galerkin differential algebraic system. The latter reduced simulated data is obtained from the Galerkin projection of the available FOM data onto the POD reduced spaces.

This new methodology introduces a significant reduction of the computational cost associated with solving inverse problems for the NSE. In fact, the use of the PINNs directly at the full order level for inferring unknown parameters in the mathematical fluid problem could be of significant computational cost. This is due to the number of degrees of freedom in the available full order data, in turbulent $3D$ problems this number is of the order 10^6 or higher. On the other hand, the proposed approach deals with the inference problem by introducing two levels of approximation. The first level is represented by the dimen-

sionality reduction performed by the POD and the Galerkin projection which results in a POD-Galerkin ROM, while in the second approximation level neural networks for the approximation of the reduced fluid variables are utilized. By doing so, one may leverage the power of the PINNs in inferring unknown parameters by solving the optimization problem (which has reduced number of unknowns and hence low computational cost) in which the goal is to minimize the error committed in approximating the (reduced) data and the error caused by the violation of the physics (the reduced POD-Galerkin equations). It is worth mentioning that after the training of the PINNs in the offline stage, the POD-Galerkin PINN ROM will still be able to do fast online forward computations without the need to re-train the neural networks.

This article is organized as follows: "The problem setup: parameterized Navier–Stokes equations" section introduces the full order model and addresses the incompressible Navier–Stokes equations. In "The reduced order model (ROM)" section, the reduced order model structure and methodology is presented. Firstly, the proper orthogonal decomposition method is recalled, then we present the non-intrusive reduced order model developed in this work to treat inverse problems in a reduced setting in the context of the NSE. "Numerical results" section gives three numerical examples that illustrate the results of the parameter identification using the reduction approach proposed in this work. The first example is the steady case of the flow past a backward step in a laminar setting, while the second one is the turbulent case of the flow around a circular cylinder and the last example deals with a more complex $3D$ turbulent case which is the flow around a surface mounted cubic box.

The problem setup: parameterized Navier–Stokes equations

The NSE are ubiquitous in science and engineering where they describe the physics of many phenomena such as modeling the air flow around an airfoil, the flow in boat wakes and the motion of bluff bodies inserted in fluid flows. In this work, the focus is on the parameterized unsteady NSE, the mathematical formulation of the problem reads as follows: Given the fluid spatial domain $\Omega \in \mathbb{R}^d$, with $d = 2$ or 3 and the time window $[0, T]$ under consideration, find the vectorial velocity field $\boldsymbol{u} : \Omega \times [0, T] \mapsto \mathbb{R}^d$ and the scalar pressure field $p : \Omega \times [0, T] \mapsto \mathbb{R}$ such that:

$$\begin{cases} \frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) - \nabla \cdot \nu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T \right) = -\nabla p & \text{in } \Omega \times [0, T], \\ \nabla \cdot \boldsymbol{u} = \boldsymbol{0} & \text{in } \Omega \times [0, T], \\ \boldsymbol{u}(t, \boldsymbol{x}; \boldsymbol{\mu}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\mu}) & \text{on } \Gamma_{\text{inlet}} \times [0, T], \\ \boldsymbol{u}(t, \boldsymbol{x}; \boldsymbol{\mu}) = \boldsymbol{0} & \text{on } \Gamma_0 \times [0, T], \\ (\nu \nabla \boldsymbol{u} - p\boldsymbol{I})\boldsymbol{n} = \boldsymbol{0} & \text{on } \Gamma_{\text{outlet}} \times [0, T], \\ \boldsymbol{u}(0, \boldsymbol{x}) = \boldsymbol{R}(\boldsymbol{x}) & \text{in } (\Omega, 0), \end{cases} \tag{1}$$

where t is the time, $\boldsymbol{x}$ is the spatial variable vector and $\Gamma = \Gamma_{\text{inlet}} \cup \Gamma_0 \cup \Gamma_{\text{outlet}}$ is the boundary of the fluid domain Ω. The three parts that form the boundary are called Γ_{inlet}, Γ_{outlet} and Γ_0, they correspond to the inlet boundary, the outlet boundary and the physical walls, respectively. The fluid kinematic viscosity is denoted by ν and is constant across the spatial domain. The function $\boldsymbol{f}$ includes the boundary conditions for the non-homogeneous boundary. The initial velocity field is given by the function $\boldsymbol{R}(x)$. The normal unit vector is denoted by $\boldsymbol{n}$. We remark that the velocity and the pressure fields depend on time,

space and the parameter $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^q$, where $\mathcal{P}$ is a q-dimensional parameter space, the dependencies are dropped for making the notation concise.

The governing equations of (1) are discretized using the FVM [66]. In this work, the numerical solver used for solving the NSE is the finite volume C++ library OpenFOAM® (OF) [67]. For more details on the finite volume discretization and the techniques used by OpenFOAM, we refer the reader to [68].

The fluid dynamics problems which this work aim at tackling include turbulent problems or problems with moderate to high Reynolds number $Re = \frac{U_\infty L}{\nu}$, where L and U_∞ are the characteristic length and velocity of the particular fluid problem, respectively. Flows with low values of Reynolds number are called laminar flows in which fluid moves smoothly or in regular paths, laminar flows are also characterized by having high momentum diffusion and low convection. In contrast to laminar flows, turbulent flows are chaotic where sudden changes in the velocity and the pressure fields are more common. Turbulent flows can be frequently observed in real life applications, examples include external flows over airplanes or ships and oceanic and atmospheric currents. Therefore, it is important to mention how turbulence is treated at both the FOM and ROM levels.

At the FOM level, turbulence is not solved directly using the so-called DNS approach, instead it is modeled using modeling strategies, namely the Reynolds Averaged Navier–Stokes equations (RANS) and the Large Eddy Simulation (LES). In both cases, one resorts to closure models which introduce an additional viscosity term known as the eddy/turbulent viscosity (denoted by ν_t) which has the same unit as the physical kinematic viscosity ν [69]. The estimation of the eddy viscosity requires the use of the so-called closure turbulence models. These models approximate ν_t as a function of other turbulence variables such as the turbulent kinetic energy k, where they resolve one or more transport-diffusion PDE for the additional turbulence variables. Examples of such closure models under the RANS approach include the one equation Spalart–Allmaras (S–A) turbulence model [70] and the two equations $k - \epsilon$ [71] and SST $k - \omega$ turbulence models [72], where ϵ and ω stand for the turbulent dissipation and the specific turbulent dissipation rate, respectively. As for the LES turbulence models, the Smagorinsky model [73] is a well known LES model, other models are the dynamic eddy viscosity model proposed in [74] and the one equation model named "dynamicKEqn" [75] which has been utilized in this work. Closure turbulence models are also often termed as Eddy Viscosity Models (EVMs). For a comprehensive review on the issue of turbulence modeling, we refer the reader to [18, 19].

We report here as an example the modified equations after employing the RANS approach complemented by the $k - \omega$ turbulence model:

$$\begin{cases} \frac{\partial \bar{\boldsymbol{u}}}{\partial t} + \boldsymbol{\nabla} \cdot (\bar{\boldsymbol{u}} \otimes \bar{\boldsymbol{u}}) = \boldsymbol{\nabla} \cdot \left[-\bar{p}\mathbf{I} + (\nu + \nu_t) \left(\boldsymbol{\nabla} \bar{\boldsymbol{u}} + (\boldsymbol{\nabla} \bar{\boldsymbol{u}})^T \right) \right] & \text{in } \Omega \times [0, T], \\ \boldsymbol{\nabla} \cdot \bar{\boldsymbol{u}} = 0 & \text{in } \Omega \times [0, T], \\ \bar{\boldsymbol{u}}(t, \boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\mu}) & \text{on } \Gamma_{In} \times [0, T], \\ \bar{\boldsymbol{u}}(t, \boldsymbol{x}) = \mathbf{0} & \text{on } \Gamma_0 \times [0, T], \\ (\nu \boldsymbol{\nabla} \bar{\boldsymbol{u}} - \bar{p}\boldsymbol{I})\boldsymbol{n} = \mathbf{0} & \text{on } \Gamma_{Out} \times [0, T], \\ \bar{\boldsymbol{u}}(0, \boldsymbol{x}) = \boldsymbol{R}(\boldsymbol{x}) & \text{in } (\Omega, 0), \\ \nu_t = F(k, \omega), & \text{in } \Omega, \\ \text{Transport-Diffusion equation for } k, & \\ \text{Transport-Diffusion equation for } \omega, & \end{cases} \tag{2}$$

where F is an algebraic function that relates ν_t with the turbulence variables k and ω. LES closure models result in a similar modified momentum equation for the NSE. We remark that in any of the turbulence modeling strategies mentioned above, there exists an additional vector field term in the momentum equation which is $\boldsymbol{\nabla} \cdot \left[\nu_t \left(\boldsymbol{\nabla} \bar{\boldsymbol{u}} + (\boldsymbol{\nabla} \bar{\boldsymbol{u}})^T \right) \right]$. The latter term will be referred to as the turbulent term in this work.

The reduced order model (ROM)

This section presents the reduced order model (ROM) constructed for the reduction of the NSE addressed in the previous section. An effective ROM is sought for the approximation of the solutions of the parameterized NSE (1) and (2) for both laminar and turbulent flows. Therefore, the ROM will take into consideration the features of the full order model (FOM) including turbulence treatment when applicable. The ROM will then be used in parameter estimation/inference tasks.

The main assumption in reduced order modeling is that the dynamics of the FOM are governed by few dominant modes, and therefore, an accurate reproduction of the full order solution is possible when one combines appropriately those dominant modes. This assumption represents a cornerstone in the construction of ROMs and mathematically it implies that the FOM solution fields of the velocity and pressure can be approximated as sum of spatial modes multiplied by temporal coefficients, i.e.:

$$\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{i=1}^{N_u} a_i(t; \boldsymbol{\mu}) \phi_i(\boldsymbol{x}), \quad p(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{i=1}^{N_p} b_i(t; \boldsymbol{\mu}) \chi_i(\boldsymbol{x}), \tag{3}$$

where the reduced velocity and reduced pressure modes are denoted by $\phi_i(\boldsymbol{x})$ and $\chi_i(\boldsymbol{x})$, respectively. The reduced modes of both variables depend only on the spatial variables. The coefficients $a_i(t; \boldsymbol{\mu})$ and $b_i(t; \boldsymbol{\mu})$ represent the i-th reduced solution for velocity and pressure, respectively, they depend on both time t and the parameter $\boldsymbol{\mu}$. Several methods and approaches can be applied for the generation of the reduced order spaces of the velocity and pressure defined by $\mathbb{V}_{rb} = \text{span}\{\phi_i\}_{i=1}^{N_u}$ and $\mathbb{Q}_{rb} = \text{span}\{\chi_i\}_{i=1}^{N_p}$, respectively. The method chosen in this work for the generation of the reduced space is POD [21,22] applied directly on the set of all realizations of the solution fields which might correspond to different values of the parameters and/or time.

Efficient reduced order models rely on the notion of having two decoupled phases termed as the offline and the online phases. In the offline phase, the training procedure of the ROM is carried out. This includes sampling the parameter space and then simulating the FOM in order to generate the snapshots which are used later for the generation

of the reduced order space (here the POD space). Hence, the offline stage consists of computing the POD modes and all reduced quantities, which form the reduced system and are dependent on the POD modes. The offline phase is known to have a significant computational cost due to the fact that the offline computations depend on the FOM dimension. However, the offline phase must be carried out just once for a given choice of the ROM dimension. The final result of the offline stage is the reduced order system of equations.

The online stage utilizes the ROM and hence results in fast computations which are dependent only on the dimension of the ROM. Ideally, the online stage should not depend on any aspect of the full order computational model such as accessing the original finite volume mesh. During the online stage, the solution of the reduced order problem is found by solving the low dimensional system produced during the offline stage.

In this work, POD is used for the generation of the reduced order spaces of both the velocity and pressure. After sampling the parameter space, the FOM described in is solved for each value of the parameter $\boldsymbol{\mu} \in \mathcal{P}_M = \{\boldsymbol{\mu}_1, ...\boldsymbol{\mu}_M\}$ and solutions are acquired at the desired time instants $\{t_1, t_2, \ldots, t_{N_T}\} \subset [0, T]$. This yields a total of $N_s = M * N_T$ snapshots which form the following snapshots matrices for velocity and pressure:

$$\boldsymbol{\mathcal{S}_u} = \{\boldsymbol{u}(\boldsymbol{x}, t_1; \boldsymbol{\mu}_1), \ldots, \boldsymbol{u}(\boldsymbol{x}, t_{N_T}; \boldsymbol{\mu}_M)\} \in \mathbb{R}^{N_u^h \times N_s}, \tag{4}$$

$$\boldsymbol{\mathcal{S}_p} = \{p(\boldsymbol{x}, t_1; \boldsymbol{\mu}_1), \ldots, p(\boldsymbol{x}, t_{N_T}; \boldsymbol{\mu}_M)\} \in \mathbb{R}^{N_p^h \times N_s}. \tag{5}$$

The POD velocity and pressure modes are then computed using the method of snapshots [76]. As for the identification of the reduced coefficients of the velocity and pressure in (3), we utilize feedforward neural networks to achieve this task in the online stage.

In more details, we use Physics Informed Neural Networks (PINNs) to solve the reduced problem. Firstly, the reduced equations are obtained by performing a Galerkin projection of the FOM equations onto the POD spaces of the velocity and pressure. Then, one may encode these reduced equations as a part of the loss function which has to be minimized by the neural network optimizer. The resulted non-intrusive reduced order model merges aspects of POD-Galerkin ROMs with Physics-Informed Neural Networks (PINNs), therefore, it is termed here as *POD-Galerkin PINN ROM*. This reduced order model is designed to solve inverse problems for the Navier–Stokes equations. The goal is to identify/infer unknown parameters or inputs in a mathematical models by comparing the predictions of these models with real or simulated measurements or outputs. The inference task is carried out by leveraging the features of neural networks which allow for the introduction of additional trainable weights. These new weights are present in the loss function through the reduced equations which makes it possible to compute gradients of the loss with respect to these weights and consequently to optimize their value. In rest of this section we describe the construction of the proposed ROM.

The construction of a POD-Galerkin ROM for the Navier–Stokes equations starts by projecting the momentum equation onto the reduced space spanned by the velocity POD modes $[\phi_i]_{i=1}^{N_u}$, i.e.:

$$\left(\phi_i, \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) - \boldsymbol{\nabla} \cdot \nu \left(\boldsymbol{\nabla u} + (\boldsymbol{\nabla u})^T\right) + \boldsymbol{\nabla} p\right)_{L^2(\Omega)} = 0. \tag{6}$$

After inserting the reduced approximation of the velocity and pressure, we obtain the following ODEs which represent the reduced momentum equation:

$$\dot{\boldsymbol{a}} = \nu(\boldsymbol{B} + \boldsymbol{B_T})\boldsymbol{a} - \boldsymbol{a}^T \boldsymbol{C}\boldsymbol{a} - \boldsymbol{H}\boldsymbol{b}, \tag{7}$$

where each of $\boldsymbol{B}, \boldsymbol{B_T}, \boldsymbol{C}$ and $\boldsymbol{H}$ is either a reduced order matrix or tensor. These terms are computed as follows:

$$(\boldsymbol{B})_{ij} = \left(\phi_i, \nabla \cdot \nabla \boldsymbol{\phi}_j\right)_{L^2(\Omega)}, \tag{8}$$

$$(\boldsymbol{B_T})_{ij} = \left(\phi_i, \nabla \cdot (\nabla \boldsymbol{\phi}_j^T)\right)_{L^2(\Omega)}, \tag{9}$$

$$(\boldsymbol{C})_{ijk} = \left(\phi_i, \nabla \cdot (\phi_j \otimes \boldsymbol{\phi}_k)\right)_{L^2(\Omega)}, \tag{10}$$

$$(\boldsymbol{H})_{ij} = \left(\phi_i, \nabla \chi_j\right)_{L^2(\Omega)}. \tag{11}$$

We note that the treatment of non-convective term in the reduced momentum equation above is done by the use of the third-order tensor $\boldsymbol{C}$. This approach might lead to a substantial increase in the computational cost of solving the reduced problem when the number of the reduced velocity unknowns N_u grows. This approach approximates the projection of the non-linear term $\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u})$ onto the velocity POD mode $\boldsymbol{\phi}_i$ as follows:

$$(\boldsymbol{\phi}_i, \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}))_{L^2(\Omega)} \approx \boldsymbol{a}^T \boldsymbol{C_{i\bullet\bullet}} \boldsymbol{a}. \tag{12}$$

However, we propose to utilize a different approach in which we add a variable for the approximation of the convective term in the reduced momentum equation named $\boldsymbol{c}$, where:

$$(\boldsymbol{\phi}_i, \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}))_{L^2(\Omega)} = \boldsymbol{c}_i, \tag{13}$$

the additional variable $\boldsymbol{c}$ represents the projection of the non-linear vector field $\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u})$ (which can be retrieved/isolated from any velocity snapshot) onto the velocity POD modes $[\boldsymbol{\phi}_i]_{i=1}^{N_u}$. The final form of the reduced momentum equation is then given by

$$\dot{\boldsymbol{a}} = \nu(\boldsymbol{B} + \boldsymbol{B_T})\boldsymbol{a} - \boldsymbol{c} - \boldsymbol{H}\boldsymbol{b}. \tag{14}$$

An additional set of reduced equations can be obtained by the employment of either the supremizer enrichment approach [77,78] or by considering a reduced version of the Poisson Equation for Pressure (PPE) [22,79,80]. The supremizer approach computes artificial velocity-like modes which are termed the supremizers and then it enriches the original velocity POD modes with the newly computed supremizers in a way that ensures the fulfillment of a reduced version of the inf-sup condition. The additional velocity-like fields or the supremizers are computed by solving the following problems:

$$\begin{cases} \Delta \boldsymbol{s}_i = -\nabla \chi_i & \text{in } \Omega, \ \forall \chi_i \in \mathbb{V}^p_{POD}, \\ \boldsymbol{s}_i = \boldsymbol{0} & \text{on } \partial\Omega. \end{cases} \tag{15}$$

After that the velocity POD space is enriched with the supremizer modes:

$$\tilde{\mathbb{V}}^u_{POD} = [\phi_1, \ldots, \phi_{N_u}] \oplus [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{N_S}] \in \mathbb{R}^{N^h_u \times (N_u + N_S)}. \tag{16}$$

The original velocity POD modes are divergence free by construction since they are just a linear combination of the velocity snapshots. This implies that the projection of the continuity equation onto the pressure modes before enriching the velocity POD space would have added no reduced equations. The newly added supremizer modes are not divergence free, therefore, the continuity equation could be utilized for obtaining an additional set of scalar reduced algebraic equations, as follows:

$$(\chi_i, \nabla \cdot \boldsymbol{u})_{L^2(\Omega)} = 0. \tag{17}$$

The final POD-Galerkin ROM with the supremizer enrichment approach is given by

$$\boldsymbol{M}\dot{\boldsymbol{a}} = \nu(\boldsymbol{B} + \boldsymbol{B_T})\boldsymbol{a} - \boldsymbol{c} - \boldsymbol{H}\boldsymbol{b}, \tag{18a}$$

$$\boldsymbol{P}\boldsymbol{a} = \boldsymbol{0}, \tag{18b}$$

with two additional reduced matrices $\boldsymbol{M}$ and $\boldsymbol{P}$. The first matrix $\boldsymbol{M}$ is the mass matrix, which is not unitary anymore as a result of the additional supremizer modes. The matrix $\boldsymbol{P}$ is called the divergence reduced matrix. The entries of the additional matrices are given by:

$$(\boldsymbol{M})_{ij} = \left(\boldsymbol{\phi}_i, \phi_j\right)_{L^2(\Omega)}, \tag{19}$$

$$(\boldsymbol{P})_{ij} = \left(\chi_i, \boldsymbol{\nabla} \cdot \phi_j\right)_{L^2(\Omega)}. \tag{20}$$

In the turbulent case, an additional term in the reduced momentum equation will appear. This term corresponds to the projection of the added turbulence modeling term in the momentum equation (in the RANS or the LES formulation at the FOM level) onto the velocity POD modes. The turbulent POD-Galerkin ROM with the employment of the supremizer enrichment approach is given by

$$\boldsymbol{M}\dot{\boldsymbol{a}} = \nu(\boldsymbol{B} + \boldsymbol{B_T})\boldsymbol{a} - \boldsymbol{c} - \boldsymbol{H}\boldsymbol{b} + \boldsymbol{h}, \tag{21a}$$

$$\boldsymbol{P}\boldsymbol{a} = \boldsymbol{0}, \tag{21b}$$

where $\boldsymbol{h}$ is the turbulent reduced variable.

At this point, we describe the structure of the PINNs which are used to approximate the solution of the POD-Galerkin ROMs. The PINNs have as input the time and the parameter. The outputs are the reduced velocity, pressure, convective and turbulent terms denoted by $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ and $\boldsymbol{h}$, respectively. The dimension of each of these output terms is N_u except for the reduced pressure which is of N_p dimension.

The starting point of the PINN training is the computation of output label data. This data consist of the L^2 projection coefficients for each of the FOM variables fields onto the velocity or pressure POD basis. The velocity L^2 projection coefficients are computed as follows:

$$\mathbb{R} \ni a^j_{i,L^2} = \left(\boldsymbol{\mathcal{S}}^{\boldsymbol{i}}_{\boldsymbol{u}}, \phi_j\right)_{L^2(\Omega)}, \quad \text{for} \quad i = 1, 2, \ldots, N_s, \quad j = 1, 2, \ldots, N_u, \tag{22}$$

similarly the pressure coefficients are given by:

$$\mathbb{R} \ni b^j_{i,L^2} = \left(\boldsymbol{\mathcal{S}}^{\boldsymbol{i}}_{\boldsymbol{p}}, \chi_j\right)_{L^2(\Omega)}, \quad \text{for} \quad i = 1, 2, \ldots, N_s, \quad j = 1, 2, \ldots, N_p. \tag{23}$$

Then, the FOM vectorial fields of the convective and turbulent terms are retrieved from the original snapshots of the velocity, pressure and the turbulent eddy viscosity. Then, one may compute the projection of these vectorial fields onto the velocity POD modes $[\boldsymbol{\phi}_i]_{i=1}^{N_u}$. This yields the output data for the vectors $\boldsymbol{c}$ and $\boldsymbol{h}$ which are also needed for the training of the PINN. The additional coefficients are given by:

$$\mathbb{R} \ni c^j_{i,L^2} = \left(\boldsymbol{\nabla} \cdot (\boldsymbol{\mathcal{S}}^{\boldsymbol{i}}_{\boldsymbol{u}} \otimes \boldsymbol{\mathcal{S}}^{\boldsymbol{i}}_{\boldsymbol{u}}), \phi_j\right)_{L^2(\Omega)}, \quad \text{for} \quad i = 1, 2, \ldots, N_s, \quad j = 1, 2, \ldots, N_u, \tag{24}$$

$$\mathbb{R} \ni h^j_{i,L^2} = \left(\boldsymbol{\mathcal{S}}^{\boldsymbol{i}}_{\boldsymbol{t}}, \boldsymbol{\phi}_j\right)_{L^2(\Omega)}, \quad \text{for} \quad i = 1, 2, \ldots, N_s, \quad j = 1, 2, \ldots, N_u, \tag{25}$$

where $\boldsymbol{\mathcal{S}}_t^i$ is the i-th snapshot of the turbulent additional term in the FOM formulation of the RANS or the LES turbulence modeling approach.

The input and output data matrices $\tilde{\boldsymbol{A}} \in \mathbb{R}^{N_s \times (q+1)}$ and $\tilde{\boldsymbol{G}} \in \mathbb{R}^{N_s \times (3N_u + N_p)}$ are given by

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{\mu}_1 & t_1 \\ \boldsymbol{\mu}_1 & t_2 \\ \vdots & \vdots \\ \boldsymbol{\mu}_1 & t_{N_T} \\ \boldsymbol{\mu}_2 & t_1 \\ \boldsymbol{\mu}_2 & t_2 \\ \vdots & \vdots \\ \boldsymbol{\mu}_2 & t_{N_T} \\ \vdots & \vdots \\ \boldsymbol{\mu}_M & t_1 \\ \boldsymbol{\mu}_M & t_2 \\ \vdots & \vdots \\ \boldsymbol{\mu}_M & t_{N_T} \end{bmatrix}, \tag{26}$$

$$\tilde{\boldsymbol{G}} = \begin{bmatrix} a^1_{1,L^2} & \dots & a^{N_u}_{1,L^2} & b^1_{1,L^2} & \dots & b^{N_p}_{1,L^2} & h^1_{1,L^2} & \dots & h^{N_u}_{1,L^2} & c^1_{1,L^2} & \dots & c^{N_u}_{1,L^2} \\ a^1_{2,L^2} & \dots & a^{N_u}_{2,L^2} & b^1_{2,L^2} & \dots & b^{N_p}_{2,L^2} & h^1_{2,L^2} & \dots & h^{N_u}_{2,L^2} & c^1_{2,L^2} & \dots & c^{N_u}_{2,L^2} \\ \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots \\ a^1_{N_s,L^2} & \dots & a^{N_u}_{N_s,L^2} & b^1_{N_s,L^2} & \dots & b^{N_p}_{N_s,L^2} & h^1_{N_s,L^2} & \dots & h^{N_u}_{N_s,L^2} & c^1_{N_s,L^2} & \dots & c^{N_u}_{N_s,L^2} \end{bmatrix}. \tag{27}$$

The number of PINN outputs is more than the number of outputs in the POD-NN case [33] because of the introduction of the additional terms which appear in the reduced momentum equation as part of the PINN output. The current setting ensures that the dimension of the reduced problem scales linearly with the number of reduced variables of both velocity and pressure. The POD-Galerkin PINN ROM is then constructed by training deep neural networks whose input is time and parameter and whose output is the reduced velocity, pressure, convective and turbulent terms. The loss function which has to be minimized will be a weighted loss that takes into consideration the available data and the POD-Galerkin formulation imposed by the algebraic differential system in (21). The training procedure utilizes the training set $\{\boldsymbol{l}^n, \boldsymbol{r}^n = \mathcal{F}(\boldsymbol{l}^n)\}_{n=1}^{N_s}$, where $\{\boldsymbol{l}^n\}_{n=1}^{N_s}$ is the set of input vectors, $\{\boldsymbol{r}^n\}_{n=1}^{N_s}$ is the set of output or target vectors and $\mathcal{F}$ is the function that relates the input to the output in the neural network. Each row of the matrices defined above $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{G}}$ represent a sample (recall that the number of samples was $N_s = M * N_T$). The input and output vectors are defined as $\boldsymbol{l}^n = \tilde{\boldsymbol{A}}(n, :)$ and $\boldsymbol{r}^n = \tilde{\boldsymbol{G}}(n, :)$, respectively. The overall loss function can be written as follows

$$E(\boldsymbol{w}) = E_{\text{data}}(\boldsymbol{w}) + \alpha_1 E_1(\boldsymbol{w}) + \alpha_2 E_2(\boldsymbol{w}), \tag{28}$$

where

$$E_{\text{data}}(\boldsymbol{w}) = \sum_{n=1}^{N_s} \frac{1}{3N_u + N_p} \sum_{k=1}^{3N_u+N_p} \{y_k(\boldsymbol{l}^n, \boldsymbol{w}) - r_k^n\}^2, \tag{29}$$

$$E_1(\boldsymbol{w}) = \sum_{n=1}^{N_s} \frac{1}{N_u} \sum_{k=1}^{N_u} \{\boldsymbol{R}_k^a(\boldsymbol{l}^n, \boldsymbol{y}, \boldsymbol{w})\}^2, \tag{30}$$

$$E_2(\boldsymbol{w}) = \sum_{n=1}^{N_s} \frac{1}{N_p} \sum_{k=1}^{N_p} \{\boldsymbol{R}_k^b(\boldsymbol{l}^n, \boldsymbol{y}, \boldsymbol{w})\}^2, \tag{31}$$

and

$$\boldsymbol{R}^a = -\boldsymbol{M}\dot{\boldsymbol{a}}_{\text{PINN}} + \nu(\boldsymbol{B} + \boldsymbol{B_T})\boldsymbol{a}_{\text{PINN}} - \boldsymbol{c}_{\text{PINN}} - \boldsymbol{H}\boldsymbol{b}_{\text{PINN}} + \boldsymbol{h}_{\text{PINN}} \in \mathbb{R}^{N_u}, \tag{32}$$

$$\boldsymbol{R}^b = \boldsymbol{P}\boldsymbol{a}_{\text{PINN}} \in \mathbb{R}^{N_p}, \tag{33}$$

$$\boldsymbol{y} = [\boldsymbol{a}_{\text{PINN}}, \boldsymbol{b}_{\text{PINN}}, \boldsymbol{h}_{\text{PINN}}, \boldsymbol{c}_{\text{PINN}}] \in \mathbb{R}^{3N_u+N_p}. \tag{34}$$

The two loss functions E_1 and E_2 enforce the reduced equations given by the POD-Galerkin model. The two weighting coefficients α_1 and α_2 are tuned heuristically depending on the problem but are also in general trainable. The above formulation gives the PINN the ability to estimate unknown parameters which are present in the POD-Galerkin formulation, such parameters might include for example the physical viscosity ν. The approximation of the time derivative of the reduced velocity $\dot{\boldsymbol{a}}_{\text{PINN}}$ which appears in $\boldsymbol{R}^a$ is done with the help of automatic differentiation [81]. Automatic differentiation represents a crucial tool in PINNs, where it is capable of differentiating the neural networks with respect to their input coordinates and model parameters, the latter model parameters do not include only the weights and biases stacked in the vector $\boldsymbol{w}$ but also any other unknown physical quantity in the model.

It is worth mentioning that the POD-Galerkin PINN ROM could incorporate physical constraints related to the velocity at the boundary. In fact, it is common to have inhomogeneous Dirichlet boundary conditions for the velocity field at specific parts of the boundary. This is typical in inlet–outlet problems such as the flow around a circular cylinder or the flow past a backward step. In these circumstances, an additional effort has to be made for the treatment of the inhomogeneous velocity boundary conditions at the ROM level. The common strategies for tackling this issue are the lifting function method [82–84] and the penalty method [85–89]. A brief description of the two methods will be given and then the strategy of incorporating them inside the PINN formulation will be addressed.

The lifting function method treats the non-homogeneous Dirichlet boundary condition through the introduction of a lifting function (or several lifting functions). In this method the inhomogeneity is transferred to the lifting function and a new set of velocity snapshots is created. The POD procedure is then performed on the newly created set of velocity snapshots (which have homogeneous Dirichlet boundary conditions). This results in velocity POD modes which have also homogeneous boundary conditions at the Dirichlet boundary. We remark that the lifting function must satisfy certain conditions such as being divergence free. This function is also added to the velocity POD basis. Unlike the lifting method, the penalty method does not involve any modification on the velocity snapshots. The penalty method enforces the inhomogeneous boundary condition by the introduction of an additional term in the reduced momentum equation, this term

corresponds to the projection of a function (which has zero value everywhere except on the Dirichlet boundary) onto the velocity POD modes. The penalty method modifies the POD-Galerkin ROM as follows:

$$\boldsymbol{M}\dot{\boldsymbol{a}} = \nu(\boldsymbol{B} + \boldsymbol{B_T})\boldsymbol{a} - \boldsymbol{c} - \boldsymbol{H}\boldsymbol{b} + \boldsymbol{h} + \tau\,(U_{BC}\boldsymbol{D} - \boldsymbol{E}\boldsymbol{a}), \tag{35a}$$

$$\boldsymbol{P}\boldsymbol{a} = \boldsymbol{0}, \tag{35b}$$

where U_{BC} is the velocity value at the Dirichlet boundary Γ_D, and τ is the penalization factor whose value is tuned heuristically. Higher values of τ generally tend to enforce the boundary conditions in a stronger fashion. The additional reduced operators $\boldsymbol{D}$ and $\boldsymbol{E}$ are defined as follows:

$$(\boldsymbol{D})_i = (\phi_i)_{L^2(\Gamma_D)}, \tag{36}$$

$$(\boldsymbol{E})_{ij} = \left(\phi_i, \phi_j\right)_{L^2(\Gamma_D)}. \tag{37}$$

In (35), we assumed to have only one inhomogeneous boundary condition at the Dirichlet boundary, however, a generalization for more than one condition can be done [90]. The POD-Galerkin PINN model can be adopted to the penalty method by including the additional constraint as a separate loss function denoted by $E_3(\boldsymbol{w})$ defined as follows:

$$E_3(\boldsymbol{w}) = \sum_{n=1}^{N_s} \frac{1}{N_u} \sum_{k=1}^{N_u} \{\boldsymbol{R}^c_k(\boldsymbol{l}^n, \boldsymbol{y}, \boldsymbol{w})\}^2, \tag{38}$$

with

$$\boldsymbol{R}^c = U_{BC}\boldsymbol{D} - \boldsymbol{E}\boldsymbol{a}_{\text{PINN}} \in \mathbb{R}^{N_u}. \tag{39}$$

As for the case of the lifting function method, the homogenization procedure leads to the transfer of the inhomogeneous Dirichlet boundary conditions from the velocity snapshots to the lifting functions. Therefore, the lifting velocity mode will have a normalized version of the velocity at the Dirichlet boundary at the inlet.[1]

The reduced approximation of the velocity field is modified to the following one:

$$\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx a_l\phi_l(\boldsymbol{x}) + \sum_{i=1}^{N_u} a_i^u(t; \boldsymbol{\mu})\phi_i(\boldsymbol{x}) + \sum_{i=1}^{N_S} a_i^S(t; \boldsymbol{\mu})\boldsymbol{s}_i(\boldsymbol{x}), \tag{40}$$

where $\boldsymbol{a} = [a_l, a_1^u, \ldots, a_{N_u}^u, a_1^S, \ldots, a_{N_S}^S] \in \mathbb{R}^{1+N_u+N_S}$, here the upper-subscript in a_i^u and a_i^S refer to the reduced coefficients corresponding to the original velocity POD modes and the supremizer added ones, respectively. The coefficient a_l is the one that corresponds to the lifting mode and represents as mentioned a normalized version of the velocity at Γ_D. In this case, the reduced operators $\boldsymbol{R}^a$ and $\boldsymbol{R}^b$ from (32) and (33) in the PINN formulation become:

$$\boldsymbol{R}^a = -\boldsymbol{M}[a_l, \boldsymbol{a}_{\text{PINN}}] + \nu(\boldsymbol{B} + \boldsymbol{B_T})[a_l, \boldsymbol{a}_{\text{PINN}}] - \boldsymbol{c}_{\text{PINN}} - \boldsymbol{H}\boldsymbol{b}_{\text{PINN}} + \boldsymbol{h}_{\text{PINN}} \in \mathbb{R}^{N_u}, \tag{41}$$

$$\boldsymbol{R}^b = \boldsymbol{P}[a_l, \boldsymbol{a}_{\text{PINN}}] \in \mathbb{R}^{N_p}, \tag{42}$$

[1] The velocity POD modes including the lifting function are normalized and this causes a_l to be a normalized version of the velocity at the inlet.

Hence, we observe that the PINNs are able to incorporate the velocity at the boundary in their formulation using both the lifting function and penalty methods, making it possible to learn these physical parameters through the training procedure and the optimization of the total loss function.

Numerical results

This section presents the application of the POD-Galerkin PINN reduced order models on three problems with unknown inputs or parameters. The first problem is the benchmark case of the flow past a backward step. In the first problem the POD-Galerkin PINN model is used for solving inverse and forward problems in the parameterized setting. The second one is the flow around a circular cylinder in a turbulent setting modeled by the RANS approach, in this problem we consider a situation of incomplete data, where the ROM will be used to infer an unknown input value and its corresponding missing output data. The last problem is the $3D$ flow around a surface mounted cube, this problem is a turbulent one with a large number of degrees of freedom, where turbulence is modeled using the LES turbulence approach. In the latter problem, the POD-Galerkin PINN ROM is used for the identification of the physical viscosity which is assumed to be unknown. This is done by assuming the presence of simulated data for the velocity, pressure and the eddy viscosity fields. The full order solver utilized is OpenFOAM® (OF) [67] which is an open-source C++-based library for solving fluid problems with the finite volume method. At the reduced order level, the POD modes and the L^2 projection coefficients needed for the training of the PINNs are computed using the library ITHACA-FV [91] which is also based on C++, while the training of the neural network is done using the Python library TensorFlow V2 [92].

Steady case: The Flow past a backward step

We consider the application of the POD-Galerkin PINN based reduced order model to the benchmark case of the flow past a backward step. The problem is studied here in laminar steady setting with physical parameterization. In Fig. 1, the computational domain is depicted, one may see the lengths of the different parts of the domain expressed in terms of the characteristic length $L = 1\,\text{m}$. The boundary conditions for both the velocity and pressure fields are reported in the latter figure. The objective is to utilize the POD-Galerkin PINN model for solving inverse problems, these inverse problems involve the estimation of certain physical parameters such as the velocity at the inlet or the physical viscosity. In the current setting, the physical viscosity ν is parameterized, where it is varied in the range of $[0.02, 7.1]\,\text{m}^2/\text{s}$. The velocity at the inlet is fixed at $1\,\text{m/s}$, this gives a parameterized Reynolds number which lies in the range of $[0.1408, 50]$. As for the numerical schemes used at the FOM level, a 1-st order bounded Gauss upwind scheme is used to approximate the convective term. The gradients are approximated using a Gauss linear scheme, while a Gauss linear scheme with non-orthogonal correction has been utilized for the approximation of the Laplacian terms.

The offline stage starts by collecting snapshots for different values of the parameter ν, in particular 150 samples were drawn from the range $[0.02, 7.1]$ and for each viscosity sample, the FOM was run using the SIMPLE solver. The snapshots of velocity, pressure and flux fields were stored for the computation of the POD modes and the reduced

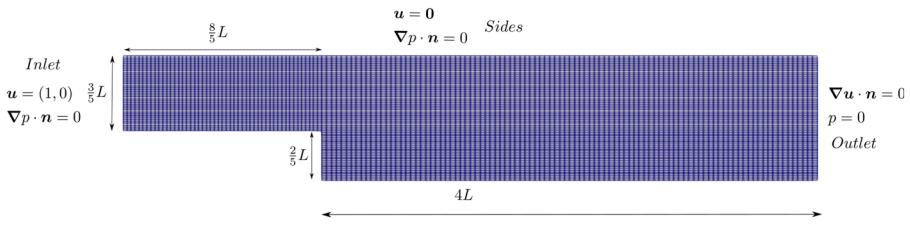

Fig. 1 The computational domain used in the numerical simulations of the flow past a backward step, all lengths are described in terms of the characteristic length L that is equal to 1 meter

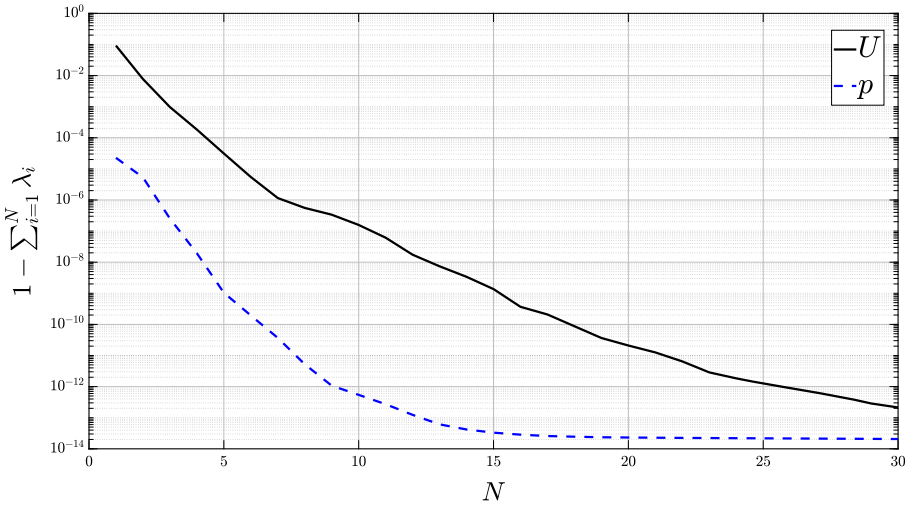

Fig. 2 The cumulative ignored eigenvalues decay for the first numerical case of the flow past a backward step. In this figure, the solid black line refers to the velocity eigenvalues and the dashed blue line corresponds to the pressure eigenvalues

operators involved in the formulation of the POD-Galerkin PINN model. Firstly, the non-homogeneous velocity boundary condition at the inlet is treated with the help of the lifting function method. In particular, the average of the velocity snapshots computed for different values of the parameter is computed and then used as the lifting function. The lifting function is then subtracted from the original snapshots which lead to the creation of a new set of homogenized velocity snapshots (velocity snapshots which have homogeneous Dirichlet boundary condition at the inlet). At this stage one can apply the POD procedure on the snapshots matrices of both the homogenized velocity and the pressure. The cumulative eigenvalues decay can be seen in Fig. 2, where the cumulative eigenvalue decay for the pressure is observed to be slower. The second step involves the computation of the supremizer modes which can be done by solving the supremizer problems corresponding to each pressure POD mode. The velocity POD space is then enriched by the supremizer modes. The convective part of each snapshots is then retrieved for later use during the training stage of the neural network.

At this point, we describe the numerical tests which will be done in this subsection. The first test has two objectives which are:

- To estimate a physical unknown which is the velocity at the boundary, where we assume for this test that this value is unknown. The identification of this physical unknown is carried out by the POD-Galerkin PINN model.

- To approximate the velocity and pressure fields for test values of the physical viscosity ν which were not seen during the offline stage.

The two objectives are achieved simultaneously by training only once the PINN informed by the POD-Galerkin system using the training data and then performing a prediction task for the test data.

In the second test, the input test data of the physical viscosity mentioned above will be assumed to be unknown and then the POD-Galerkin PINN model will be used to solve the inverse problems associated with the latter test data while assuming that the velocity at the inlet is known.

It is important to mention that the velocity at the boundary is embedded in the POD-Galerkin PINN formulation. In fact, the first coefficient of the reduced velocity vector $\boldsymbol{a}$, namely a_l corresponds to a normalized version of the velocity at the inlet. The reduced approximation of the velocity in this example read as in (40).

The next step is to compute the reduced operators which appear in the following POD-Galerkin system that defines the reduced equations:

$$\nu(\boldsymbol{B}+\boldsymbol{B_T})\boldsymbol{a}-\boldsymbol{c}-\boldsymbol{H}\boldsymbol{b}=\boldsymbol{0}, \tag{43a}$$

$$\boldsymbol{P}\boldsymbol{a}=\boldsymbol{0}. \tag{43b}$$

The input of the PINN is the physical viscosity ν and its output is formed by the reduced velocity (except the lifting coefficient a_l), reduced pressure and the reduced convective term. The number of the PINN output variables is $N_o = 3N_u + 3N_S + N_p + 1$.

To solve the inverse problem involved and to approximate the relation between the physical viscosity and the reduced velocity and pressure, we put forward a physics informed neural network which has 10 layers with each layer containing 100 neurons with tangent hyperbolic activation. This PINN takes as output the reduced variables mentioned above and uses the data available from the offline snapshots together with the knowledge given by the system in (43) to approximate the unknown coefficient a_l (or the velocity at the inlet). This is done by training the PINN with a loss function that takes into consideration the data given by the coefficients of the L^2 projections and also by constraining the output to follow the POD-Galerkin reduced equations. The output data is obtained by performing the projections in equations (22), (23) and (24). We would like to remark that both input and output values have been standardized to range of [0, 1] in order to make the neural networks learning task easier. The PINN loss function is written as

$$E(\boldsymbol{w}) = E_{\text{data}}(\boldsymbol{w}) + \alpha_1 E_1(\boldsymbol{w}) + \alpha_2 E_2(\boldsymbol{w}), \tag{44}$$

where

$$E_{\text{data}}(\boldsymbol{w}) = \sum_{n=1}^{N_s} \frac{1}{N_o} \sum_{k=1}^{N_o} \{y_k(\boldsymbol{l}^n, \boldsymbol{w}) - r_k^n\}^2, \tag{45}$$

$$E_1(\boldsymbol{w}) = \sum_{n=1}^{N_s} \frac{1}{N_u} \sum_{k=1}^{N_u} \{\boldsymbol{R}_k^a(\boldsymbol{l}^n, \boldsymbol{y}, \boldsymbol{w})\}^2, \tag{46}$$

$$E_2(\boldsymbol{w}) = \sum_{n=1}^{N_s} \frac{1}{N_p} \sum_{k=1}^{N_p} \{\boldsymbol{R}_k^b(\boldsymbol{l}^n, \boldsymbol{y}, \boldsymbol{w})\}^2, \tag{47}$$

Table 1 The PINNs results for the data mean squared error and the mean squared errors corresponding to the residual functions defining the POD-Galerkin ROM DAE

Initial a_l	$E_{\text{data}}(\boldsymbol{w})$	$E1(\boldsymbol{w})$	$E2(\boldsymbol{w})$	PINN a_l
0	$9.152303 * 10^{-6}$	0.00045994046	$3.0014705 * 10^{-6}$	2.7291148
5	$9.2302425 * 10^{-6}$	0.0009832102	$3.0044891 * 10^{-6}$	2.7288582
10	0.00020264629	0.00036979085	$2.9998373 * 10^{-6}$	2.7270014
20	$1.4284992 * 10^{-5}$	0.0016940477	$3.0059625 * 10^{-6}$	2.7295387

The values of the errors are reported for different initial values of the added weight a_l, the PINN identified value of a_l is reported in the last column. The PINNs are run for 30000 epochs with a learning rate of $1 * 10^{-3}$

and

$$\boldsymbol{R}^a = \nu(\boldsymbol{B} + \boldsymbol{B_T})[a_l, \boldsymbol{a}_{\text{PINN}}] - \boldsymbol{c}_{\text{PINN}} - \boldsymbol{H}\boldsymbol{b}_{\text{PINN}} \in \mathbb{R}^{N_u}, \quad \boldsymbol{R}^b = \boldsymbol{P}[a_l, \boldsymbol{a}_{\text{PINN}}] \in \mathbb{R}^{N_p},$$
$$\boldsymbol{y} = [\boldsymbol{a}_{\text{PINN}}, \boldsymbol{b}_{\text{PINN}}, \boldsymbol{c}_{\text{PINN}}] \in \mathbb{R}^{N_o}.$$

In the above formulation, the weights and biases vector $\boldsymbol{w}$ contains all trainable parameters which include the scalar coefficient a_l which is then learned during the training procedure of the PINN. As for the coefficients of the equations losses α_1 and α_2, they are determined in a heuristic fashion or they can also be trained like the other weights of the network.

The total number of trainable parameters in the PINN for the first test is 83627. The PINN is run for 30, 000 epochs with a learning rate of $1 * 10^{-3}$ and with one batch per epoch (batch size is equal to the number of data points N_s). The two weighting coefficients α_1 and α_2 are set to 0.01. The physical parameter a_l is initialized with zero value.

The first results are shown for the following number of modes $N_u = N_p = N_S = 5$. This number of modes gives a total number of PINN outputs of $N_o = 26$. The true value of a_l is 2.7302, while the PINN has identified the value of a_l to be 2.7291. The relative error in approximating a_l is about 0.0409 %. As mentioned above a_l has been initialized with zero value, however, we show also the results for different initial values of a_l in Table 1, one can see that the PINN is not sensitive to the initial values of the unknown parameter a_l. As for the forward task in the first test, we have generated 300 samples for ν which are equidistant samples in the range [0.05, 7]. Then another simulation campaign is launched for these viscosity samples in order to validate the PINN model. After the training of the PINN for the identification of a_l, the PINN is used for approximating the output for the newly created set of input ν. Figure 3 show the results of the forward task, where one can see the validation results for the first, second and third components of the reduced variables of the velocity, pressure and convective terms versus the value of the viscosity. We recall that the values of the physical viscosity for which the latter figure is depicted were not used in the training procedure of the POD-Galerkin ROM or the PINN. The error committed in approximating the reduced variables in the mean squared sense is about $9.8921 * 10^{-6}$, we remark that the latter error is computed on the standardized variables. As for the operators errors for the test data we have $\tilde{E}_1(\boldsymbol{w}) = 0.001027$ and $\tilde{E}_2(\boldsymbol{w}) = 2.6525 * 10^{-6}$. The last quantitative values of the errors show that the PINN was able to generalize for unseen values of the parameter and at the same time constraining the results to satisfy the algebraic system. As for the training time of the PINNs, it ranges from 6 to 7.402 min using "Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz".

As a final result for the first case, we report an assessment of the approximation accuracy of the POD-Galerkin PINN ROM. Namely, we compute the relative L^2 error for velocity

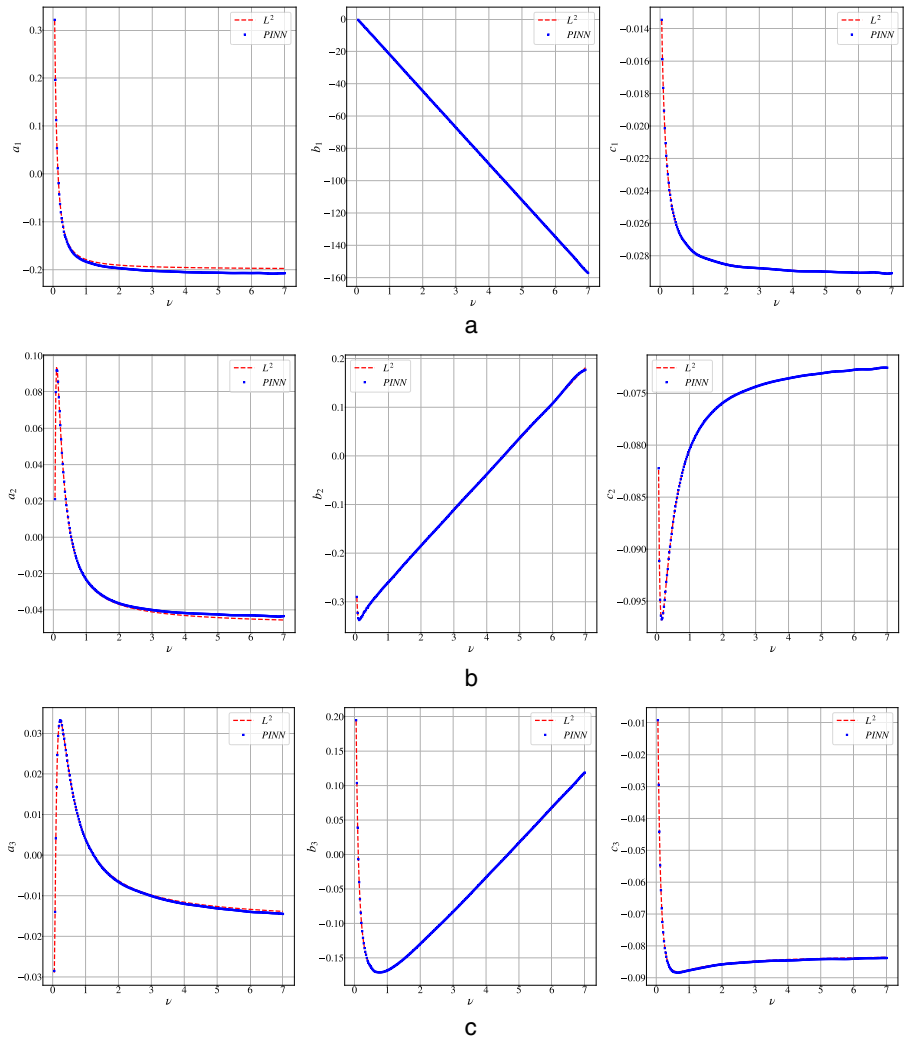

Fig. 3 The results of the POD-Galerkin PINN predictions for the 1st, 2nd and 3rd components of the velocity, pressure and convective reduced coefficients for the first numerical test. The plots compare the reduced coefficients with the L^2 projection coefficients of the test velocity and pressure fields onto the corresponding POD modes. The red-dashed lines refers to the L^2 projection coefficients, while the blue-dots correspond to the reduced coefficients obtained by the PINNs. The coefficients are plotted versus the physical viscosity values at which the test data was generated. (a) The first reduced coefficients for velocity, pressure and convective terms compared to the ones obtained by the L^2 projection. (b) The second reduced coefficients for velocity, pressure and convective terms compared to the ones obtained by the L^2 projection. (c) The third reduced coefficients for velocity, pressure and convective terms compared to the ones obtained by the L^2 projection

and pressure which, respectively, read as:

$$\epsilon_u = \frac{\|\boldsymbol{u} - \boldsymbol{u}_r\|_{L^2(\Omega)}}{\|\boldsymbol{u}\|_{L^2(\Omega)}} \times 100\%, \qquad \epsilon_p = \frac{\|p - p_r\|_{L^2(\Omega)}}{\|p\|_{L^2(\Omega)}} \times 100\%, \tag{48}$$

where $\boldsymbol{u}_r$ and p_r are the reduced order velocity and pressure fields, respectively. The values of the relative errors for the velocity and the pressure are computed for the 300 test parameter values. The mean value of the ϵ_u reduced velocity error is 0.3081 %, while the one of the pressure is 0.3459 %.

In the second test, we assume that the input values for the test data in the latter test are unknown. We aim at solving the inverse problems involved with the test data. To this end, we put forward a PINN based on the POD-Galerkin ROM with the following loss

function:

$$E(\boldsymbol{w}) = E_{\text{data}}(\boldsymbol{w}) + \alpha_1 E_1(\boldsymbol{w}) + \alpha_2 E_2(\boldsymbol{w}) + \alpha_3 \tilde{E}_1(\boldsymbol{w}), \tag{49}$$

where

$$\tilde{E}_1 = \sum_{n=1}^{300} \frac{1}{N_u} \sum_{k=1}^{N_u} \{\tilde{\boldsymbol{R}}_k^a(\tilde{l}^n, \boldsymbol{y}, \boldsymbol{w})\}^2, \tag{50}$$

and

$$\tilde{\boldsymbol{R}}^a = \tilde{l}^n(\boldsymbol{B} + \boldsymbol{B_T})[a_l, \tilde{\boldsymbol{a}}] - \tilde{\boldsymbol{c}} - \boldsymbol{H}\tilde{\boldsymbol{b}} \in \mathbb{R}^{N_u}, \tag{51}$$

where $\tilde{l}^n$ is the n-th value of the unknown input physical viscosity. The PINN used in this second test has the same structure as the one utilized in the first test. The training parameters are also the same, where the PINN is run for 30000 epochs with learning rate of $1 * 10^{-3}$. The vector of unknown viscosity values $\tilde{\boldsymbol{l}}$ is considered as additional trainable weight of the PINN and is embedded into $\boldsymbol{w}$. The identified values of the physical viscosity match to high degree of accuracy the true values. The mean squared error of the difference of the true input vector and the PINN-identified one is 0.00055.

In a last numerical example, we would like to examine the advantage of having the reduced residual terms in the PINN formulation compared to the POD-NN approach which relies only on the reduced data. We consider the current test case with parameterized physical viscosity. Snapshots are obtained for parameter samples in the range [0.01, 2.1], we assume the presence of 100 snapshots for the velocity and the pressure for equally distributed values of ν in the aforementioned range. We assume the availability of test data for the fluid fields in the range [0.025, 2] which is contained in the training snapshots window. We construct the POD-NN and POD-Galerkin PINN models for the same test case. After the computation of the reduced velocity and pressure using both models, we reconstruct the reduced approximation of the full fields using the stored POD modes. Finally we have computed the L^2 relative error committed by the reduced approximation of both the POD-NN and POD-Galerkin PINN. The errors are computed as function of the parameter ν. The errors in (48) are computed for 320 test samples equally distributed in the range [0.025, 2]. We note that the effects of the parametric variation on the velocity and pressure fields could be noticed more apparently for lower values of ν in the considered sampling range. Therefore, we expect that the results of ROMs generalization for lower values of the parameter to be less accurate than for higher values given the uniform sampling. The PINN is trained by minimizing the data loss given by the reduced data obtained from the 100 snapshots and the reduced equations loss which is computed at random points in the parameter range. We consider the reduced setting of $N_u = N_S = 8$, Fig. 4 shows the results of this test for both the velocity and the pressure fields. The results as expected shows that both ROMs (the POD-NN ROM and the POD-Galerkin PINN ROM) have given less accurate results for small values of ν. However, it could be clearly seen that the POD-Galerkin PINN ROM has contained the error values especially for the pressure field in the range of $\nu < 0.3$. Here, we mention that the maximum value of ϵ_p for the POD-Galerkin PINN ROM is about 6.1709 % while the corresponding value obtained by the POD-NN ROM is 32.4325 %. The POD-NN ROM has in total 8 test samples for which the reduced pressure error exceeded 10 %.

The reason for having more accurate results by the PINN-based ROM could be attributed to the regularization effect that is brought to the ROM by taking into consideration the physics at the reduced level. The POD-NN has given better results only in

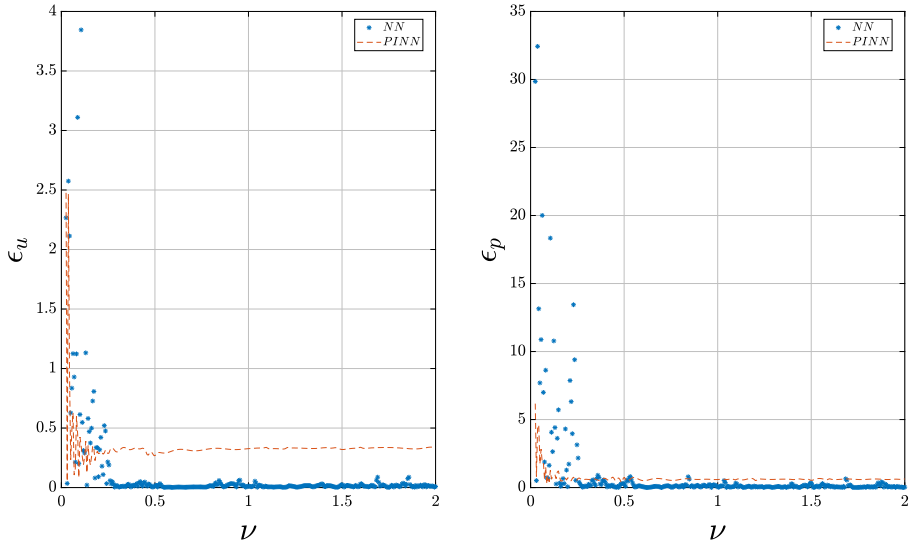

Fig. 4 The L^2 relative error for the POD-NN and POD-Galerkin PINN ROMs for both the velocity field (on the left) and the pressure field (on the right). The curves are already in percentages, the values of the errors are depicted versus the test values of the physical viscosity

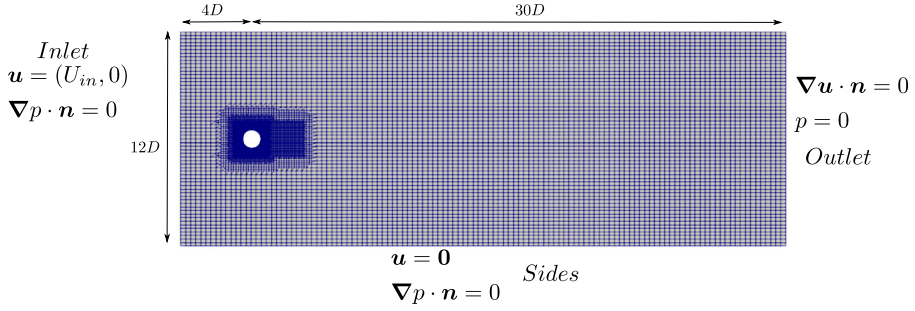

Fig. 5 The computational grid of the problem of the flow around a circular cylinder

the region where the density of the samples was high enough and in which the parametric variation is least observed. We conclude that in general the POD-Galerkin PINN ROM is more useful than the POD-NN in case of limited data since the physical equations provide additional information.

The flow around a circular cylinder

In this subsection we address the case of having incomplete data for different input configurations/settings. The computational problem considered is the one of the flow around a circular cylinder. The problem is 2D turbulent one, where turbulence is modeled using the RANS approach. The domain of the problem is $\Omega := [-4D, 30D] \times [-6D, 6D] \backslash B_D(0, 0)$, where $D = 1$ m is the diameter of the cylinder. fig. 5 shows the grid used for simulating the problem using OpenFOAM, it also reports the boundary conditions for the velocity and pressure fields. The grid has around 18000 cells, the physical viscosity is 2.5×10^{-4} m^2/s.

We assume in this test that we are presented with an incomplete set of data for the fluid dynamics fields. This set of data contains full information about the fluid dynamics fields for some parameter values and contains a set of partial output data for an unknown

Table 2 Offline parameter samples and the corresponding vortex shedding frequency and snapshots time window

Parameter sample : U_{in} in m/s	Time period T_p in s	Snapshot time window in s
1	4.255	[0, 11]
1.5	2.830	[0, 7.5]
2	2.1127	[0, 5.5]

input/parameter. In particular, we consider the parameter in this test to be the velocity at the inlet U_{in}. The fluid dynamics fields for velocity, pressure and the eddy viscosity are available for three different known values of the parameter U_{in} which are $\{1, 1.5, 2\}$ m/s. Another set of fluid data for unknown parameter value is presented, the latter set contains partial information as it lacks the pressure fields. The inference tasks in this example are (i) to approximate the unknown velocity at the inlet $U^{\star}_{in}$, (ii) to recover the missing pressure field history and finally (iii) to compute the lift and drag forces acting on the surface of the cylinder which are dependent on both $U^{\star}_{in}$ and the missing pressure data.

The snapshots coming from the fluid simulations are covering at least 2 periodic cycles of the developed regime. For each parameter value, the regime frequency known as the vortex shedding frequency is different, resulting in various snapshots time windows for the three parameter data samples. Table 2 shows the snapshots time windows for each parameter sample and the corresponding time period. The number of snapshots per parameter sample is 201 giving a total of 603 snapshots. The POD is done on the velocity and pressure snapshots which yields the velocity and pressure POD basis, then an enrichment procedure is carried out by solving the supremizer problems. The ROM is then obtained via a Galerkin projection, in this example the penalty method is used for the enforcement of the inlet velocity at the reduced level.

At this stage, one may compute the input and output data matrices which will be used to train the PINN. The input of the PINN is the combination of time and the parameter. However, a non-dimensionalization conversion for time is needed in order to give meaningful information for the PINN. To this end, the non-dimensionalized time $t^{\star}$ is defined as $t^{\star} = \frac{tU_{in}}{D} = tU_{in}$. As for the output data, it consist in the following reduced coefficients:

- The L^2 projection coefficients of the velocity snapshots onto the velocity POD modes, see (22).
- The L^2 projection coefficients of the pressure snapshots onto the pressure POD modes, see (23).
- The L^2 projection coefficients of the convective terms snapshots onto the velocity POD modes, see (24).
- The L^2 projection coefficients of the turbulent terms snapshots onto the velocity POD modes, see (25).

The output of the PINN is the stacked vector of $[\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{h}, \boldsymbol{c}]$. The POD-Galerkin DAE that models this problem is the one reported in (35). The matrices and vectors which appear in the latter DAE are computed during the offline stage. The partially known output fields for the unknown parameter are also projected onto the velocity POD space and the resulting L^2 coefficients are also used for training the PINN.

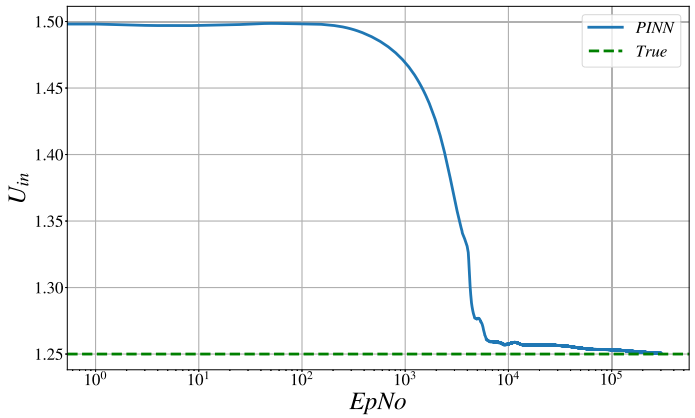

Fig. 6 The PINN approximation of the $U_{in}^{\star}$

The set of original weights and biases of the PINN denoted by $\boldsymbol{w}$ is then enlarged. This is done by introducing an additional set of weights denoted by $\boldsymbol{w}^{\star}$ which contains trainable weights that correspond to the unknown quantities to be approximated. In more details, $\boldsymbol{w}^{\star}$ contains the scalar weight $w_{U_{\text{in}}^{\star}}$ which is introduced for the approximation of the unknown velocity at the inlet. Also $\boldsymbol{w}^{\star}$ encapsulates a matrix of weights denoted by $\boldsymbol{w}^{b^{\star}}$ whose i-th column $\boldsymbol{w}_i^{b^{\star}}$ is the reduced pressure vector for the unknown pressure output data corresponding to $U_{in}^{\star}$ at a fixed time instant. Hence, the number of additional weights is $201 * N_p + 1$ (we assume that the number of data points for the unknown input is $N_T^{\star} = 201$).

The PINN loss function $E(\boldsymbol{w}, \boldsymbol{w}^{\star})$ is defined as follows:

$$E(\boldsymbol{w}, \boldsymbol{w}^{\star}) = E_{\text{data}}(\boldsymbol{w}, \boldsymbol{w}^{\star}) + \sum_{i=1}^{3} E_i(\boldsymbol{w}) + \sum_{i=1}^{3} E_i^{\star}(\boldsymbol{w}, \boldsymbol{w}^{\star}) + E_b^{\star}(\boldsymbol{w}, \boldsymbol{w}^{\star}), \tag{52}$$

as one can notice, the loss function incorporates four different types of loss, the first one $E_{\text{data}}(\boldsymbol{w}, \boldsymbol{w}^{\star})$ is the fitting data loss for both the fully known input–output data and the unknown input and partially known output data. The second loss $\sum_{i=1}^{3} E_i(\boldsymbol{w})$ corresponds to the reduced equation losses evaluated only at the known input samples. The third loss is the same as the second one but computed at the unknown input data points. The final loss is defined as $E_b^{\star}(\boldsymbol{w}, \boldsymbol{w}^{\star}) = \frac{1}{N_p} \sum_{i=1}^{N_T^{\star}} \| \boldsymbol{w}_i^{b^{\star}} - \boldsymbol{b}_{\text{PINN}}(t_i^{\star}, w_{U_{\text{in}}^{\star}}) \|_{\mathbb{R}^{N_p}}^2$, which penalizes the difference between the reduced pressure output of the PINN computed at the unknown input data points and the weights $\boldsymbol{w}^{b^{\star}}$. The last loss component will ensure that the reduced missing pressure will be recovered through the additional weights. It is worth mentioning that initial values of $w_{U_{in}^{\star}}$ is set to be the media of U_{in} of the known data, while a full zero matrix is used as a starting point for $\boldsymbol{w}^{b^{\star}}$.

The PINN used in this numerical test has 5 layers and 64 neurons per layer with mixed activations (tanh and SIREN). The Adam optimizer is used for solving the optimization problem. The PINN is run for 3×10^5 epochs, we show in Fig. 6 the evolution history of the inlet velocity weight during training. One can see that the PINN approximated inlet velocity has converged to a value which is close to the true value of 1.25 m/s. In fact the weight $w_{U_{\text{in}}^{\star}}$ at the end of the training was 1.25029 which implies that the approximated Reynolds number is 5001.178 while the real one is 5000. The reduced order settings for the last result are $N_u = N_p = N_S = 15$.

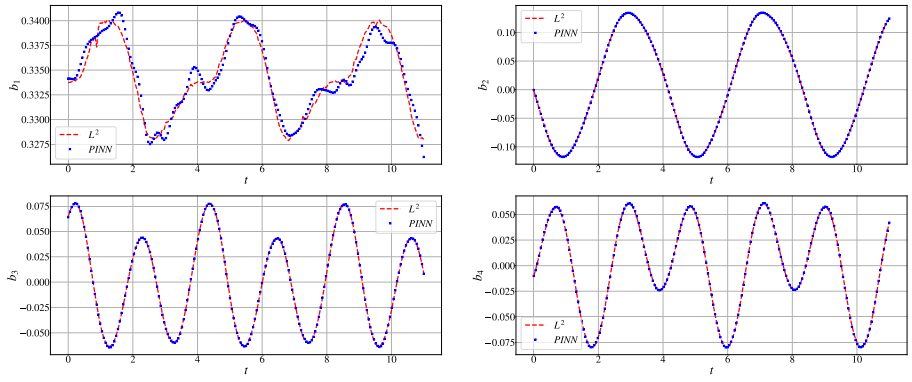

Fig. 7 The first four pressure reduced coefficients for the unknown input data

As for the reconstruction of the pressure fields, the additional matrix of weights $\boldsymbol{w}^{b^\star}$ which correspond to the reduced pressure of the unknown input data has been optimized in the PINN training process. Figure 7 depicts the time history of the first four reduced coefficients in the ROM approximation of the missing pressure fields and it shows also the corresponding four coefficients obtained by the L^2 projection of the pressure data onto the pressure POD modes.

To assess the accuracy of the approximation of the missing pressure fields, we compute the error ϵ_p in (48) between the inferred PINN pressure fields and the FOM ones for the 201 different snapshots. For the reduced model with $N_u = N_p = N_S = 15$, the values of the maximum and mean relative pressure error are 3.1396 % and 1.4457 %, respectively.

Besides solving the inverse problem, we are interested in having an accurate reconstruction of the time-history of the cylinder drag and lift coefficients. These coefficients come from the fluid dynamics forces $\boldsymbol{F}$ acting on the surface of the cylinder which depend locally on the pressure and velocity fields as follows:

$$\boldsymbol{F} = \int_{\partial\Gamma_{cy}} (2\mu\nabla\boldsymbol{u} - p\boldsymbol{I})\boldsymbol{n}ds. \tag{53}$$

If F_l and F_d are the forces components acting on the surface of the cylinder in the lift and drag direction (the lift direction is the one perpendicular to the flow, while the drag direction is horizontal to the flow), respectively, then the non-dimensionalized drag and lift forces coefficients denoted by C_d and C_l, respectively, are given by:

$$C_d = \frac{F_d}{\frac{1}{2}\rho U_{in}^2 A_{ref}}, \quad C_l = \frac{F_l}{\frac{1}{2}\rho U_{in}^2 A_{ref}}, \tag{54}$$

where ρ is the fluid density and A_{ref} is the reference area. The evaluation of the forces at the reduced level is done in a way that respects the full decoupling of the offline and the online stages, for more details we refer the reader to section 2.8 in [90]. The PINN is used to recover the lift and drag coefficients by performing forward test for all time values at which we had recorded the forces during the FOM simulation. Then the PINN forces approximation is used together with the PINN-inferred inlet velocity to obtain the reduced lift and drag coefficients signals. The results of this test are shown in Fig. 8.

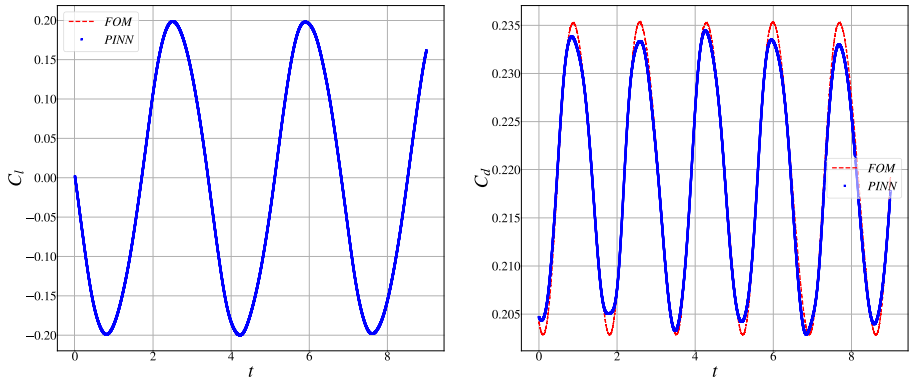

Fig. 8 The lift and drag coefficients curve for the unknown input data ($N_u = N_p = N_S = 15$)

In order to have a quantitative evaluation of the accuracy of the lift and drag coefficients approximation, we compute the following L^2 relative errors

$$\epsilon_{C_l} = \frac{\left\| C_l(t) - {C_l}^r(t) \right\|_{L_2(T_1,T_2)}}{\left\| C_l(t) \right\|_{L_2(T_1,T_2)}} \times 100\%, \quad \epsilon_{C_d} = \frac{\left\| C_d(t) - {C_d}^r(t) \right\|_{L_2(T_1,T_2)}}{\left\| C_d(t) \right\|_{L_2(T_1,T_2)}} \times 100\%, \tag{55}$$

where $C_l(t)$ and $C_d(t)$ are the signal functions corresponding to the FOM lift and drag coefficients, respectively. As for ${C_l}^r(t)$ and ${C_d}^r(t)$ they are the corresponding ROM signals, and $[T_1, T_2]$ is the time interval in which the error is sought. The error values for the reduced model with $N_u = N_p = N_S = 15$ are 0.8551 % and 0.5699 % for lift and drag, respectively. This shows that the POD-Galerkin PINN ROM has been able to reconstruct important CFD performance indicators such as the lift and drag coefficients despite the uncertainty in presence.

In the last results, we have considered the presence of incomplete output data, nevertheless the set of output data was known on the whole internal domain. Now we assume that we have only local data points given by the vector forces acting on the cylinder in (53). This makes the inference problem more difficult to solve because of the locality of the data presented. In spite of that, the POD-Galerkin PINN ROM could be used to solve the inference problem using the data points of the forces. This can be carried out thanks to the physical (reduced) equations which correspond to (53) and which are then incorporated in the PINN loss function. We show the evolution of the PINN approximation of the U_{in} in Fig. 9 for this ultimate test.

The flow around a surface mounted cube

The case considered in this subsection is the one of a flow past a cubic obstacle. The problem is considered in a turbulent setting in three dimensions. Turbulence is modeled using the LES strategy, in particular, the SGS turbulence model used is the one-equation eddy viscosity model named "dynamicKEqn" in OpenFOAM. This model is proposed in [75] as a continuation of the SGS model presented in [74]. The fluid domain is $\Omega := [0, 14.5\,L] \times [0, 9\,L] \times [0, 2\,L]$, where $L = 1$ m is the length of the cube. The boundary of the domain Γ is formed by the three parts which are the inlet $\Gamma_{\text{inlet}} = \{0\} \times [0, 9\,L] \times [0, 2\,L]$, the walls (the sides and the cube) Γ_0, and the outlet $\Gamma_{\text{outlet}} := \{14.5\,L\} \times [0, 9\,L] \times [0, 2\,L]$. The computational grid used in the simulations is depicted in Fig. 10, where one can see

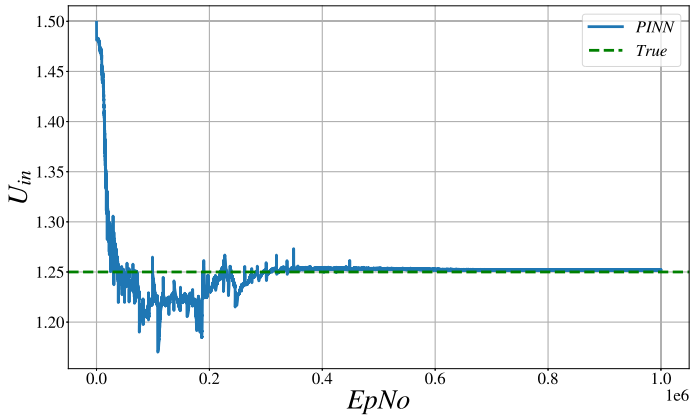

Fig. 9 The PINN approximation of the $U_{in}^{\star}$ in the case of forces data

in image 10a a cross-sectional view of the domain at $x_3 = L$. The cube is mounted on the ground surface with a distance of $3.5L$ from the inlet, where the velocity of the flow is horizontal with magnitude U_{in}. In Fig. 10b, a similar image is shown for the cross-sectional view at $x_2 = 4.5L$ and a zoomed picture of the cube is viewed in Fig. 10c. The boundary conditions for the velocity and pressure at each part of the boundary are reported in the latter figures. The finite volume mesh features around 1.2 millions cells. The physical viscosity ν is equal to $2.5 \times 10^{-5}\,\mathrm{m}^2/\mathrm{s}$. The velocity at the inlet U_{in} is 1 m/s. This gives a Reynolds number (based on the cube length) of 40, 000.

In this numerical test, the physical viscosity ν is assumed to be unknown, the goal is to identify its value with a high degree of accuracy and to approximate the non-dimensionalized forces coefficients coming from the lift and drag forces acting on the surface of the cubic obstacle.

The fluid problem is simulated for a timespan which is long enough to observe stable values of the time-average of certain output quantities. These quantities include the mean and the Root Mean Squared (RMS) values of the non-dimensionalized forces coefficients coming from the lift and drag forces acting on the surface of the cubic obstacle [see (53) and (54].

This problem has been studied for the same value of the Reynolds number mentioned above in [93–95]. In the latter studies, RANS and LES simulation were carried of for the approximation of the values of the lift and drag coefficients of the cubic box. The nature of this problem is characterized by having chaotic turbulent response for the velocity and pressure field profiles. Thus, in order to have an accurate approximation of the mean drag and lift coefficients, at least 100 non-dimensionalized time units $t^{\star}$ were simulated, where $t^{\star} = \frac{tU_{in}}{L}$. Hence, the NSE are simulated for 100 s. The resulted graph for the drag coefficient time-history in the build-up phase is shown in Fig. 11a. The mean value of the drag coefficient across the build-up phase is 1.4829. The RMS values of the mean subtracted drag coefficients signal is 0.0648.

After completing the build-up phase, snapshots are taken for the construction of the reduced order model, where the simulation is resumed for another 50 s. Snapshots are acquired each 0.25 s which results in a total of 201 snapshots. The time-history of the drag coefficient during the offline snapshots time-window is depicted in Fig. 11b.

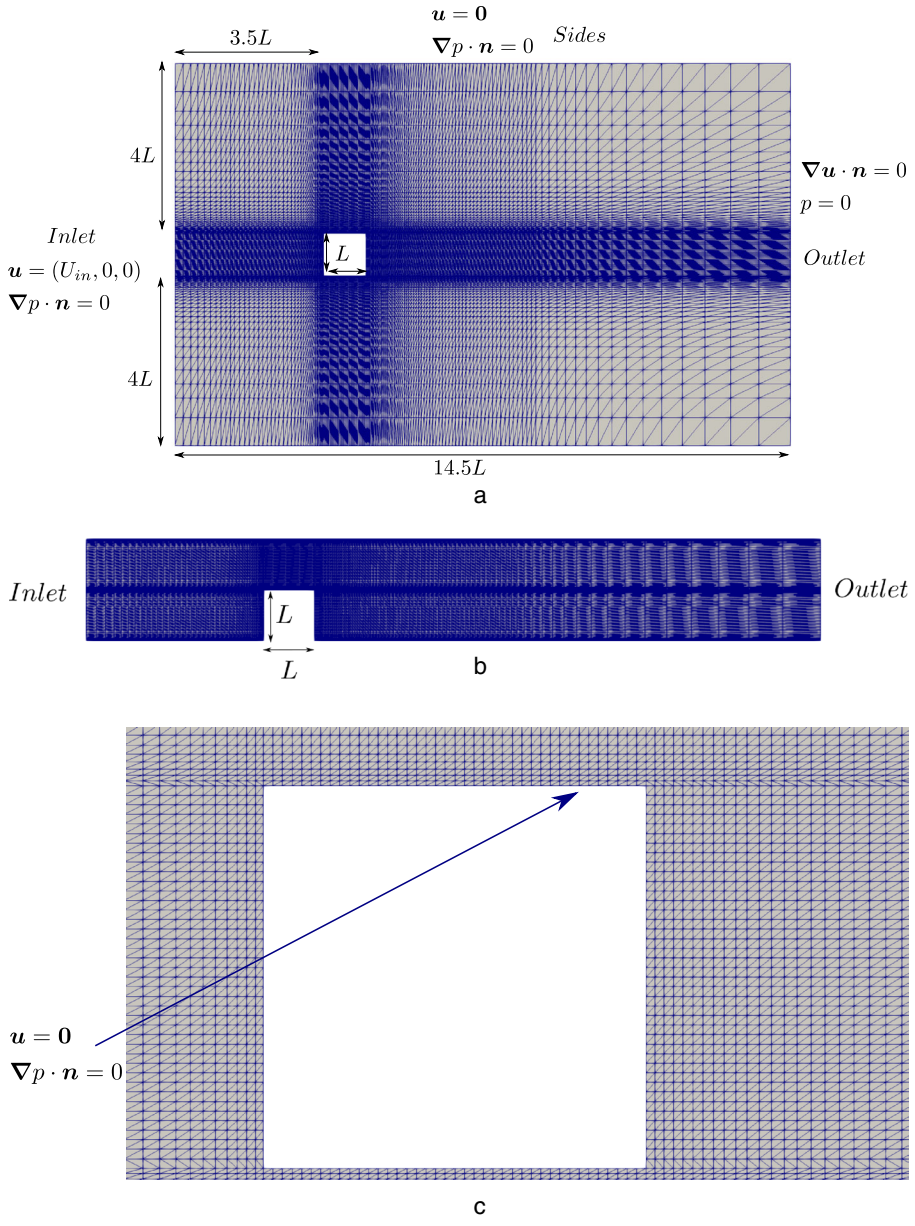

Fig. 10 The OpenFOAM mesh used in the simulations for the case of the flow around a surface mounted cube. (a) A cross-sectional view of the grid at $x^3 = L$, the boundary conditions for the velocity and pressure are reported for the inlet, outlet and the sides. (b) A cross-sectional view of the grid at $x^3 = 4.5L$. (c) A zoomed picture near the box with boundary conditions imposed on it

The POD method is then applied on the snapshots matrices of velocity and pressure. Figure 12 shows the first two POD modes of the velocity and the pressure. After the computation of the POD modes of the velocity and pressure, one may solve the supremizer problems in order to obtain the supremizer modes which are then added to the original velocity POD basis.

At this point, the computation of the output data of the PINNs is carried out. The penalty method is used for the enforcement of the inhomogeneous Dirichlet boundary condition at the inlet, therefore, the POD-Galerkin DAE that models this problem is

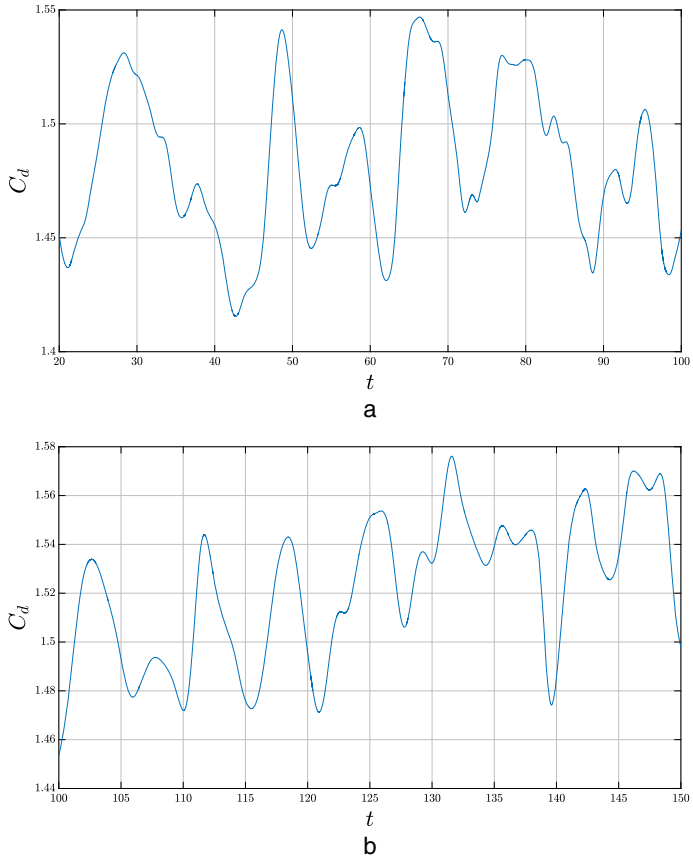

Fig. 11 The drag coefficient of the cubic obstacle for the build-up phase and the offline snapshots time-window. (a) The time-history of the drag coefficient C_d during the build-up phase in the time interval [20, 100]. (b) The time-history of the drag coefficient C_d in the snapshots time-window [100, 150]

the one reported in (35). The matrices and vectors which appear in the latter DAE are computed during the offline stage.

In order to approximate the value of the unknown physical viscosity, we put forth deep neural networks which have time as the input and the stacked vector of $[\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{h}, \boldsymbol{c}]$ as their output. The neural networks used in this test have 7 layers with each layer containing 200 neurons. The activation function used at each neuron is the tangent hyperbolic function. The Adam optimizer is used for training the neural networks with a decaying learning rate of initial value of 1×10^{-4}, the optimization is run for 5×10^5 training epochs. The loss function which has to be minimized during the training procedure of the PINNs is the following:

$$E(\boldsymbol{w}) = E_{\text{data}}(\boldsymbol{w}) + E_1(\boldsymbol{w}) + E_2(\boldsymbol{w}) + E_3(\boldsymbol{w}), \tag{56}$$

where $E_{\text{data}}(\boldsymbol{w})$, $E_1(\boldsymbol{w})$, $E_2(\boldsymbol{w})$ and $E_3(\boldsymbol{w})$ are defined as in (29), (30), (31) and (38), respectively. An additional trainable variable which corresponds to the physical viscosity and denoted by ν_{PINN} is added to the set of the PINNs tunable weights $\boldsymbol{w}$. This additional trainable parameter of the neural network is present in the loss function through $E_1(\boldsymbol{w})$. Therefore, the Adam optimizer will be able to compute the gradient of the total loss function $E(\boldsymbol{w})$ with respect to ν_{PINN} and as a result optimize its value. The initial value of ν_{PINN} is assumed to be 10^{-4}.

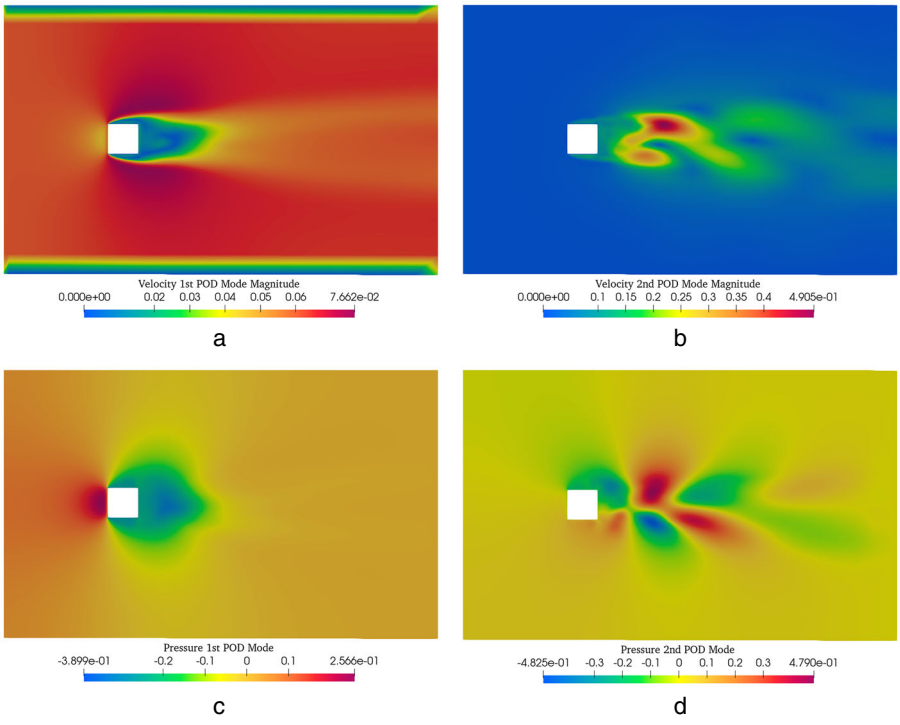

Fig. 12 The first two POD modes of the velocity and pressure, a $2D$ cross-sectional view of the domain at $x_3 = L$ is shown. (a) The first velocity POD mode. (b) The second velocity POD mode. (c) The first pressure POD mode. (d) The second pressure POD mode

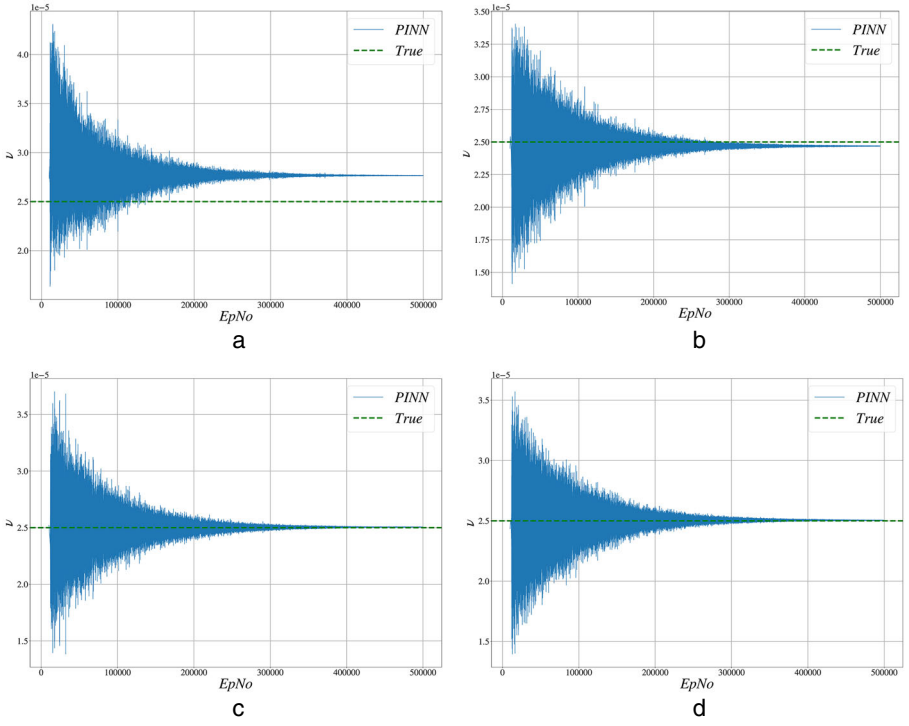

Fig. 13 The approximation of the physical viscosity by the PINN at each training epoch for different reduced number of modes. (a) Reduced setting : $N_u = N_p = N_S = 10$. (b) Reduced setting : $N_u = N_p = N_S = 30$. (c) Reduced setting : $N_u = N_p = N_S = 40$. (d) Reduced setting : $N_u = N_p = N_S = 80$

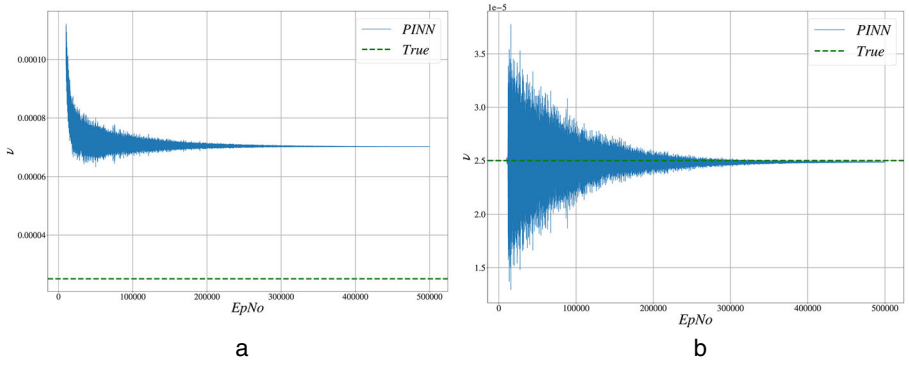

Fig. 14 The approximation of the physical viscosity by the PINN at each training epoch for the reduced setting $N_u = N_p = N_S = 60$, Fig. 14a refers to the PINN based on the quadratic approximation of the convective term using a third order tensor and Fig. 14b corresponds to the PINN with the convective term being approximated as an output of the neural network (the approach adopted in this work). (a) The PINN with the convective term approximated as a quadratic product at the reduced level. (b) The PINN with the convective term approximated as an additional reduced output of the neural network

The PINN proposed in this work relies on approximating the nonlinear convective term as an additional auxiliary variable in the neural network structure. Another approach is based on transforming the nonlinear term into a quadratic form as done in [51]. The last approach implies that one needs to compute a third order tensor $\boldsymbol{C}$ [see (10)] whose dimension is $N_u + N_S$. The number of terms which has to be computed for each reduced setting will be $(N_u + N_S)^3$. Consequently, the cost of computing this tensor for large number of reduced modes is considered significant even if such cost is an offline one. For example, for the case of $N_u = N_S = 60$, the computation time is around 7.97 hours when using 24 CPUs and it increases even to 37.27 hours in the case of $N_u = N_S = 100$. We would like to compare the accuracy of the PINN based on the latter approach in approximating the physical viscosity in the current test case with the one proposed here (the PINN based on incorporating the reduced convective term as part of the neural network outputs).

The results of the approximation conducted by the PINN proposed in this work for different number of reduced modes are shown in Fig. 13. Figure 14 presents the comparison of the results obtained by the two PINNs which differ in the way the nonlinear term is approximated. The last figure shows that the PINN based on the approximation of the nonlinear term as an additional reduced output has obtained accurate approximation when 60 modes were used in the construction of the ROM for each reduced variable, where the relative error in approximating ν is as low as 1 %. On the other hand the PINN based on the quadratic approximation assumption of the nonlinear term has not yielded an accurate approximation where it converged to a value of 7.0213×10^{-5}. The inaccuracy of the last result could be attributed to the effect of linear-based reduction given by the POD on the approximation of nonlinear quantities at the reduced level. In the presented approach such an effect is excluded since the reduced nonlinear term is considered an additional output of the neural network with labeled data available for the training.

The POD-Galerkin PINN ROM is also used for the approximation of the lift and drag coefficients of the cubic obstacle. The reduced forces coefficients are then compared to the FOM ones which are recorded during the full order simulation. In order to obtain the reduced forces, we perform a set of forward computations using the trained PINNs

for the time values at which the FOM has recorded the forces. The number of time steps performed by the FOM solver during the acquisition of the offline snapshots is 28370. The results of the PINNs forward computations are the reduced velocity and reduced pressure vectors at the aforementioned 28370 time instants. We remark that these computations are carried out in a computational time which is significantly low, yielding high speed-up factors. The results of the forward application of the PINNs are shown in Fig. 15, where the first and second coefficients of each reduced variable are plotted, the figure depicts both the L^2 projection coefficients and the ROM coefficients. The last figure shows that the PINNs forward predictions are matching the original L^2 projection coefficients curves to a high degree despite the presence of uncertain parameter in the ROM formulation. The reduced velocity and pressure outputs will be used together with the reduced forces matrices which were computed during the offline stage to yield the reduced three dimensional forces acting on the surface of the box. Then, the reduced lift and drag coefficients are computed.

The results of this test for different values of the reduced modes are shown in Figs. 16 and 17. The latter figures depict the time history of the FOM and the ROM forces coefficients for different reduced spaces dimensions. It is clear from the previous results that the ROM results are not matching their FOM counterparts when only $10-20$ modes for each of N_u, N_p and N_S are used for the ROM construction. However, Figs. 16d and 17d demonstrate that the level of qualitative reproduction of the FOM C_d and C_l curves by the surrogate model becomes substantially better when at least 60 modes are used for each reduced variable.

In order to have a quantitative evaluation of the accuracy of the lift and drag coefficients approximation, we compute the L^2 relative errors [see (55)]. The errors are computed for different number of reduced modes, Fig. 18 plots the error values versus the number of modes used for the ROM construction. The last plot shows that the error in approximating the drag coefficients reaches 0.3531 % when $N_u = N_p = N_S = 100$ modes, while the lift coefficients error is 0.1523 % for the same setting.

We report a study of the computational time needed for the completion of each task or step involved in the POD-Galerkin PINN ROM. This study is conducted in Table 3 for different numbers of the reduced modes. Firstly, we mention the computational time taken by the FOM solver for running the problem on 24 CPUs (CPU model "AMD EPYC 7302 16-Core Processor @ 1498MHz") that is $T_{\text{Off}} = 11,460$ s or approximately 3.1 hours. In the second column of Table 3, we report $T_{\text{proj,DAE}}$ which is the time required for projecting the equations and computing the DAE reduced vectors and matrices. The third column lists the time taken for the computation of the PINNs outputs data, this time is denoted by $T_{\text{proj,data}}$. We remark that the computational cost corresponding to $T_{\text{proj,DAE}}$ is present also in the case of the intrusive POD-Galerkin ROM, while the cost that corresponds to $T_{\text{proj,data}}$ is only present in non-intrusive or hybrid ROMs such as the POD-Galerkin PINN ROM. The most significant cost is the one reported in the fourth column, where one can see the time taken for training the PINNs for 5×10^5 epochs on the Graphics Processing Unit (GPU). Table 3 also details in the fifth column the cost for the forward computations carried out by the PINNs for the approximation of the forces denoted by $T_{\text{PINN,F}}$ (GPU cost). Finally the speed-up SU is recorded in the last column, where this value is calculated as follows $\text{SU} = \frac{T_{\text{Off}}}{T_{\text{PINN,F}} + T_{\text{Onl,Forces}}}$, with $T_{\text{Onl,Forces}}$ being the time required for assembling

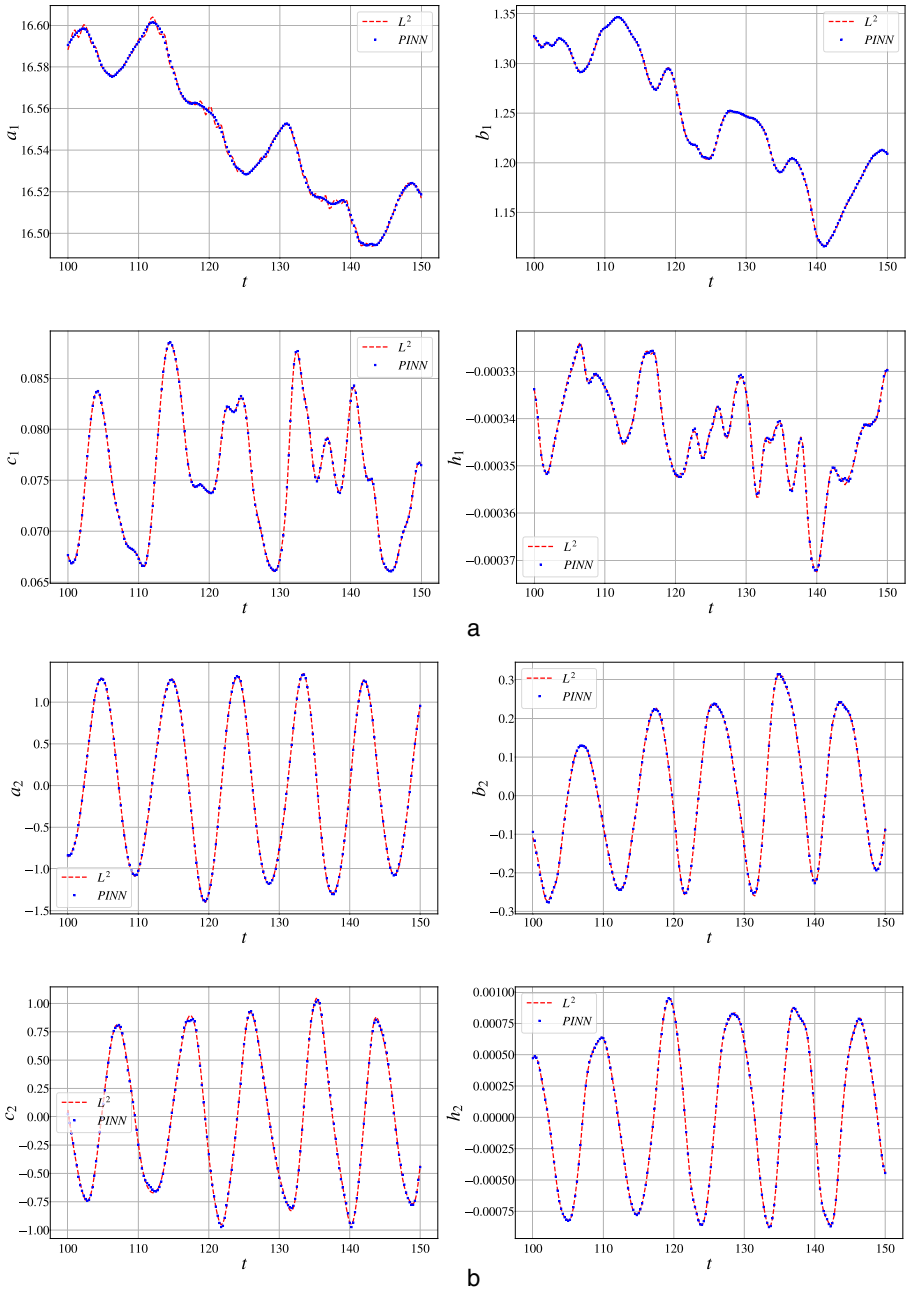

Fig. 15 The results of the PINNs predictions for all reduced variables, the reduced order spaces are constructed with $N_u = N_p = N_S = 60$. The plots compare the reduced coefficients with the L^2 projection coefficients. The red-dashed lines refers to the L^2 projection coefficients, while the blue-dots correspond to the reduced coefficients obtained by the PINNs. (a) The first reduced coefficients for velocity, pressure, turbulent and convective terms compared to the ones obtained by the L^2 projection. (b) The second reduced coefficients for velocity, pressure, turbulent and convective terms compared to the ones obtained by the L^2 projection

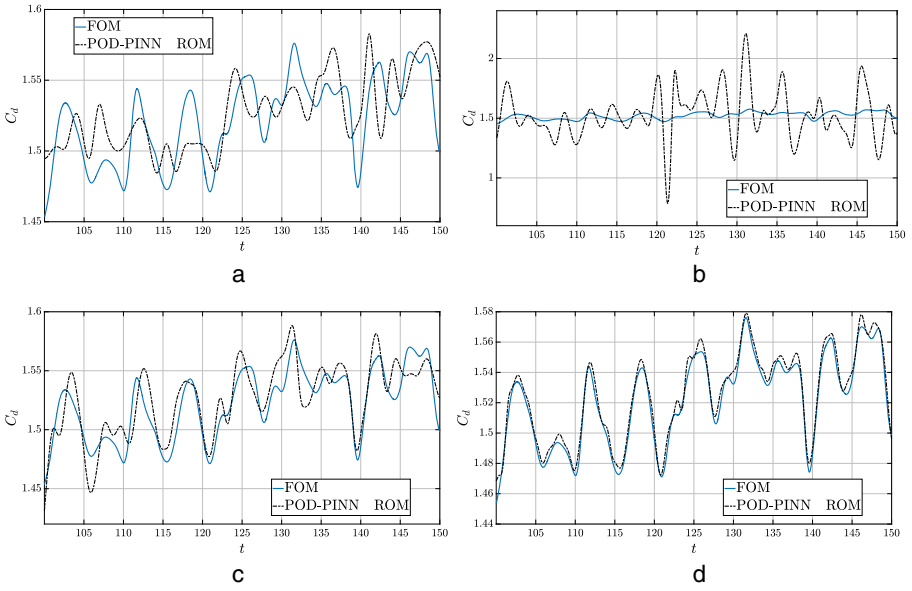

Fig. 16 The drag coefficients curve over the time range [100, 150] s. The figures show the one obtained by the FOM solver and the ones approximated by the ROM for different values of and N_u, N_p and N_S. (a) The drag coefficients curve reconstructed by the ROM using $N_u = N_p = N_S = 10$ compared to the FOM one. (b) The drag coefficients curve reconstructed by the ROM using $N_u = N_p = N_S = 20$ compared to the FOM one. (c) The drag coefficients curve reconstructed by the ROM using $N_u = N_p = N_S = 30$ compared to the FOM one. (d) The drag coefficients curve reconstructed by the ROM using $N_u = N_p = N_S = 60$ compared to the FOM one

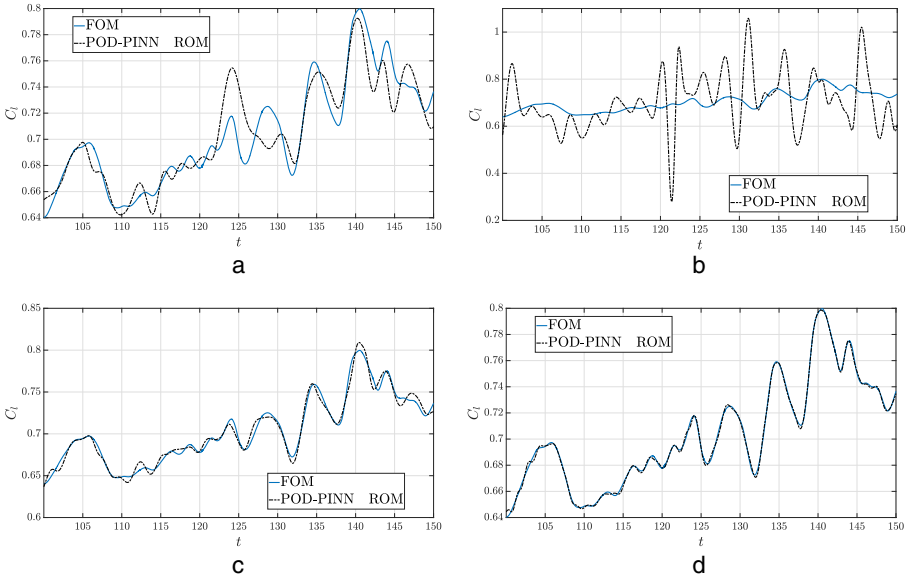

Fig. 17 The Lift coefficients curve over the time range [100, 150] s. The figures show the one obtained by the FOM solver and the ones approximated by the ROM for different values of and N_u, N_p and N_S. (a) The Lift coefficients curve reconstructed by the ROM using $N_u = N_p = N_S = 10$ compared to the FOM one. (b) The Lift coefficients curve reconstructed by the ROM using $N_u = N_p = N_S = 20$ compared to the FOM one. (c) The Lift coefficients curve reconstructed by the ROM using $N_u = N_p = N_S = 30$ compared to the FOM one. (d) The Lift coefficients curve reconstructed by the ROM using $N_u = N_p = N_S = 60$ compared to the FOM one.

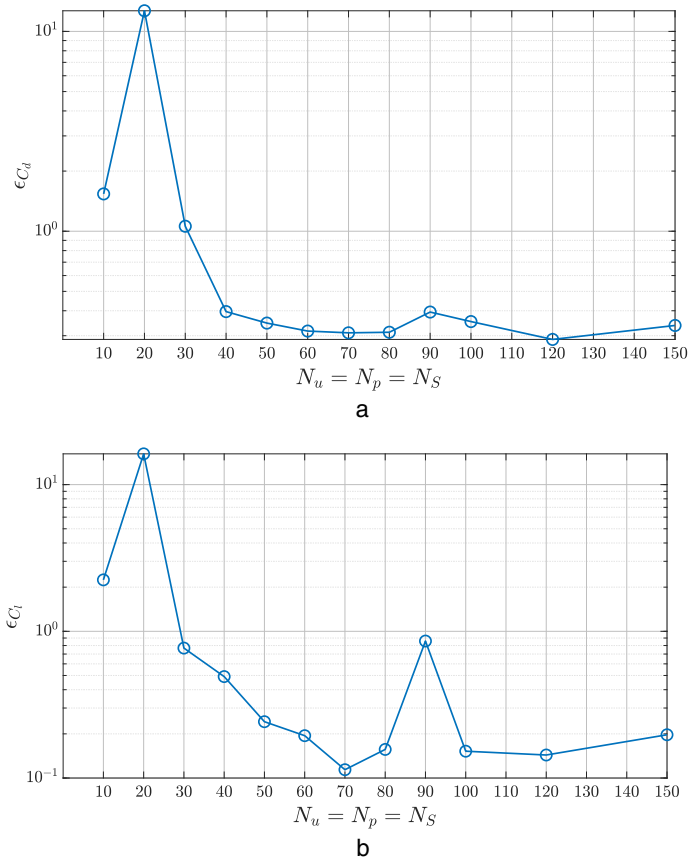

Fig. 18 The L^2 relative errors for the approximation of the drag and lift coefficients in the time span [100, 150] as function of the number of modes. A base-10 logarithmic scale on the *y*-axis is used in both plots, the error values are already in percentages. (a) The L^2 relative error in approximating the drag coefficient C_d over the time span [100,150] as function of the number of modes. (b) The L^2 relative error in approximating the lift coefficient C_l over the time span [100,150] as function of the number of modes

the reduced forces in the online stage and whose value is about $7 - 10 * 10^{-5}$ s. As it can be seen from the latter study, the use of deep neural networks in reduced order modeling results in a substantial increase in the offline cost given by the time needed for the PINNs training in this setting $T_{\text{PINNs,T}}$. However, the nature of the inverse problem at hand requires this training procedure for the approximation of the unknown parameter. In addition, the POD-Galerkin PINN ROM compensates the high offline-cost by giving in return high speed-up (SU) value which reaches as high as 10^6. These speed-up factors are not easily attainable in fully intrusive POD-Galerkin ROMs.

To conclude, it is evident that the POD-Galerkin PINN ROM has provided accurate approximation for parameter estimation problems. At the same time, the presence of unknown parameter has not affected the quality of the approximation of important outputs such as the lift and drag forces acting on the surface of the box. However, the results for the study of the quality of the PINNs approximation for the forward and inverse cases illustrate that the quality of the inverse approximation is not so much damaged when only 10–20 modes are used for each reduced variable. Unlike the inverse case, the forward approximation of the lift and drag coefficients needs substantially larger number of reduced modes.

Table 3 The computational time taken by different tasks needed for the implementation of the POD-Galerkin PINN ROM (i) the second column reports the time taken for performing the projection of the equations (computing the reduced quantities in the reduced DAE system) in parallel setting using 24 processors, (ii) the third column details the CPU time (also using 24 processors in parallel) consumed for the computation of the PINNs output data represented by the L^2 projection coefficients of the different variables

$N_u = N_p = N_S$	$T_{\text{proj,DAE}}$ in s	$T_{\text{proj,data}}$ in s	$T_{\text{PINNs,T}}$ in min	$T_{\text{PINN,F}}$ in s	Speedup
10	8.269	37.6084	58.01	0.004612	$2.4438 * 10^6$
20	25.219	59.5178	59.94	0.004715	$2.3711 * 10^6$
30	55.0379	77.1995	61.78	0.004778	$2.3557 * 10^6$
40	96.2729	100.27687	58.01	0.004873	$2.2907 * 10^6$
50	145.838	126.1021	71.7	0.006929	$1.6261 * 10^6$
60	207.629	159.0223	70.63	0.006481	$1.7393 * 10^6$
70	289.609	187.9712	59.91	0.006036	$1.8584 * 10^6$
80	387.812	233.1214	72.06	0.005258	$2.1340 * 10^6$
90	462.827	270.7364	65.49	0.005462	$2.0480 * 10^6$
100	649.555	322.3108	89.91	0.005543	$2.0314 * 10^6$
120	850.28	428.9114	101.0	0.005463	$2.0368 * 10^6$
150	1289.12	625.9247	136.8	0.006089	$1.8480 * 10^6$

(iii) The fourth column shows the time required for running the PINNs on the Graphics Processing Unit (GPU), (iii) the fifth column indicates the time taken for carrying out the forward online computations of the PINNs for the approximation of the reduced variables. (iv) The last column reports the speedup achieved by the reduced order model. The speedup is calculated as follows Speedup $= \frac{T_{\text{Off}}}{T_{\text{PINN,F}}+T_{\text{Onl,Forces}}}$, where $T_{\text{Off}} = 11,460\,\text{s}$ is the CPU time needed for simulating the FOM for the snapshots time window using 24 processors in parallel, and $T_{\text{Onl,Forces}}$ is the time consumed for assembling the reduced forces in the online stage which has taken around $7 - 10 * 10^{-5}$ s in the current experiments. The table shows the computational costs for different sizes of reduced modes

Conclusions

We have presented a reduced order model which is designed to learn unknown input parameters or physical quantities for fluid problems governed by the Navier–Stokes equations.

The proposed model employs the POD for the generation of the reduced order space and then utilizes Galerkin projection for the construction of the reduced order system. The solution of the reduced order system is obtained by the use of physics-informed neural networks (PINNs). The PINNs have as input time and/or parameters, while their output is the combined vector of the reduced velocity, pressure, turbulent and convective terms. The training procedure of the PINNs is carried out by minimizing a loss function which is a combination of the data loss and the reduced equations losses. Unknown physical parameters which appear in the POD-Galerkin DAE could be approximated using the PINNs by exploiting their feature of introducing additional trainable weights. The PINNs optimizer was then used to compute the gradient of the loss function with respect to the additional trainable weight and consequently optimize its value.

The proposed POD-Galerkin PINN ROM has proven being accurate in solving inverse problems involving unknown physical quantities such as the physical viscosity. At the same time, this ROM is able to reconstruct fluid dynamics outputs with high degree of accuracy despite having input uncertainty. Three test cases have been used for the validation of the proposed model. The first case is the steady flow past a backward step, the second case is the one of a circular cylinder immersed in horizontal flow, while the last one is the unsteady flow past a surface mounted cube. The steady case was considered in laminar setting while the unsteady cases are turbulent ones with Reynolds number up

to $Re = 40,000$. The POD-Galerkin PINN ROM has given very promising results when it comes to the approximation of the unknown parameters and also for the prediction of the fluid dynamics outputs, both for the non-turbulent and the turbulent case.

The approach presented in this work is useful in several different circumstances such as a situation in which parameterized data is presented with the data being incomplete or only partially known. The approach also can be used for the inference of unknown constant quantities such as the PDE identification tasks where physical constants (that define the PDE operator) are not known. The approach may become limited when the data is presented for only one setting/configuration (non-parameterized case) and/or when the unknown variable is a spatial field for which no data is available. We also highlight that the approach requires the computation of turbulent and nonlinear terms from each collection of fluid fields snapshots. We understand that the last requirement could be difficult to meet in certain settings.

Abbreviations

FOM Full Order Model
ROM Reduced Order Model
PINNsPhysics Informed Neural Networks
NSE Navier–Stokes Equations
POD Proper Orthogonal Decomposition
RANS Reynolds Averaging Navier–Stokes
LES Large Eddy Simulations

Author contributions

All authors participated in the development of the methodology. SH conducted the numerical experiments and wrote the draft paper. All authors read and approved the final manuscript.

Funding

Open Access funding enabled and organized by Projekt DEAL. The research of S.H. has been supported by a Jacobi Fellowship at the University of Potsdam. The research of M.F. is partially funded by the Deutsche Forschungsgemeinschaft (DFG)- Project-ID 318763901 - SFB1294.

Declarations

Consent for publication

The authors give their consent for publication.

Competing interests

The authors declare that they have no competing interests.

Received: 18 August 2022 Accepted: 8 February 2023
Published online: 18 March 2023

References

1. Hesthaven JS, Rozza G, Stamm B. Certified reduced basis methods for parametrized partial differential equations. Cham: Springer; 2016. https://doi.org/10.1007/978-3-319-22470-1.
2. Quarteroni A, Manzoni A, Negri F. Reduced basis methods for partial differential equations. Cham: Springer; 2016. https://doi.org/10.1007/978-3-319-15431-2.
3. Benner P, Ohlberger M, Pater A, Rozza G, Urban K. Model reduction of parametrized systems. Cham: Springer; 2017.
4. Benner P, Gugercin S, Willcox K. A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Rev. 2015;57(4):483–531. https://doi.org/10.1137/130932715.
5. Bader E, Kärcher M, Grepl MA, Veroy K. Certified reduced basis methods for parametrized distributed elliptic optimal control problems with control constraints. SIAM J Sci Comput. 2016;38(6):3921–46. https://doi.org/10.1137/16m1059898.
6. Balajewicz M, Dowell EH. Stabilization of projection-based reduced order models of the Navier–Stokes. Nonlinear Dyn. 2012;70(2):1619–32. https://doi.org/10.1007/s11071-012-0561-5.
7. Amsallem D, Farhat C. Stabilization of projection-based reduced-order models. Int J Numer Methods Eng. 2012;91(4):358–77. https://doi.org/10.1002/nme.4274.
8. DeVore R, Petrova G, Wojtaszczyk P. Greedy algorithms for reduced bases in banach spaces. Construct Approxim. 2013;37(3):455–66. https://doi.org/10.1007/s00365-013-9186-2.
9. Binev P, Cohen A, Dahmen W, DeVore R, Petrova G, Wojtaszczyk P. Convergence rates for greedy algorithms in reduced basis methods. SIAM J Math Anal. 2011;43(3):1457–72. https://doi.org/10.1137/100795772.

10. Volkwein S. Proper orthogonal decomposition: Theory and reduced-order modelling. Lecture Notes, University of Konstanz. 2013;4(4):8.
11. Bergmann M, Bruneau C-H, Iollo A. Enablers for robust POD models. J Comput Phys. 2009;228(2):516–38. https://doi.org/10.1016/j.jcp.2008.09.024.
12. Baiges J, Codina R, Idelsohn SR. Reduced-order modelling strategies for the finite element approximation of the incompressible Navier-Stokes equations. Comput Appl Sci. 2014;33:189–216. https://doi.org/10.1007/978-3-319-06136-8_9.
13. Burkardt J, Gunzburger M, Lee H-C. POD and CVT-based reduced-order modeling of navier-stokes flows. Computer Methods Appl Mech Eng. 2006;196(1–3):337–55. https://doi.org/10.1016/j.cma.2006.04.004.
14. Ballarin F, Rozza G. POD-Galerkin monolithic reduced order models for parametrized fluid-structure interaction problems. Int J Numer Methods Fluids. 2016;82(12):1010–34. https://doi.org/10.1002/fld.4252.
15. Noack BR, Eckelmann H. A low-dimensional galerkin method for the three-dimensional flow around a circular cylinder. Phys Fluids. 1994;6(1):124–43. https://doi.org/10.1063/1.868433.
16. Akhtar I, Nayfeh AH, Ribbens CJ. On the stability and extension of reduced-order Galerkin models in incompressible flows. Theor Comput Fluid Dyn. 2009;23(3):213–37. https://doi.org/10.1007/s00162-009-0112-y.
17. Kunisch K, Volkwein S. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. SIAM J Numer Anal. 2002;40(2):492–515. https://doi.org/10.1137/S0036142900382612.
18. Wilcox DC. Turbulence Modeling for CFD. Turbulence Modeling for CFD, vol. v. 1. DCW Industries, La Canada, California, U.S.A, La Cañada, California; 2006.
19. Berselli LC, Iliescu T, Layton WJ. Mathematics of large eddy simulation of turbulent flows. Berlin: Springer; 2005.
20. Sagaut P. Large Eddy simulation for incompressible flows. Berlin: Springer; 2006. https://doi.org/10.1007/b137536.
21. Lorenzi S, Cammi A, Luzzi L, Rozza G. POD-Galerkin method for finite volume approximation of Navier–Stokes and RANS equations. Computer Methods Appl Mech Eng. 2016;311:151–79. https://doi.org/10.1016/j.cma.2016.08.006.
22. Stabile G, Hijazi S, Mola A, Lorenzi S, Rozza G. POD-Galerkin reduced order methods for CFD using Finite Volume Discretisation: vortex shedding around a circular cylinder. Commun Appl Ind Math. 2017;8(1):210–36. https://doi.org/10.1515/caim-2017-0011.
23. Stabile G, Rozza G. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier-Stokes equations. Computers Fluids. 2018;173:273–84. https://doi.org/10.1016/j.compfluid.2018.01.035.
24. Xie X, Mohebujjaman M, Rebholz LG, Iliescu T. Data-driven filtered reduced order modeling of fluid flows. SIAM J Sci Computing. 2018;40(3):834–57. https://doi.org/10.1137/17m1145136.
25. Xiao D, Fang F, Buchan AG, Pain CC, Navon IM, Du J, Hu G. Non linear model reduction for the Navier Stokes equations using residual DEIM method. J Comput Phys. 2014;263:1–18. https://doi.org/10.1016/j.jcp.2014.01.011.
26. Barrault M, Maday Y, Nguyen NC, Patera AT. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. Comptes Rendus Mathematique. 2004;339(9):667–72. https://doi.org/10.1016/j.crma.2004.08.006.
27. Bonomi D, Manzoni A, Quarteroni A. A matrix DEIM technique for model reduction of nonlinear parametrized problems in cardiac mechanics. Computer Methods Appl Mech Eng. 2017;324:300–26. https://doi.org/10.1016/j.cma.2017.06.011.
28. Carlberg K, Farhat C, Cortial J, Amsallem D. The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. J Comput Phys. 2013;242:623–47. https://doi.org/10.1016/j.jcp.2013.02.028.
29. Ionita AC, Antoulas AC. Data-driven parametrized model reduction in the loewner framework. SIAM J Scic Computing. 2014;36(3):984–1007. https://doi.org/10.1137/130914619.
30. Peherstorfer B, Willcox K. Dynamic data-driven reduced-order models. Computer Methods Appl Mech Eng. 2015;291:21–41. https://doi.org/10.1016/j.cma.2015.03.018.
31. Kaiser E, Noack BR, Cordier L, Spohn A, Segond M, Abel M, Daviller G, Östh J, Krajnović S, Niven RK. Cluster-based reduced-order modelling of a mixing layer. J Fluid Mech. 2014;754:365–414. https://doi.org/10.1017/jfm.2014.355.
32. Guo M, Hesthaven JS. Reduced order modeling for nonlinear structural analysis using gaussian process regression. Computer Methods Appl Mech Eng. 2018;341:807–26. https://doi.org/10.1016/j.cma.2018.07.017.
33. Hesthaven JS, Ubbiali S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. J Comput Phys. 2018;363:55–78. https://doi.org/10.1016/j.jcp.2018.02.037.
34. Noack BR, Afanasiev K, Morzyński M, Tadmor G, Thiele F. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. J Fluid Mech. 2003;497:335–63. https://doi.org/10.1017/s0022112003006694.
35. Guo M, Hesthaven JS. Data-driven reduced order modeling for time-dependent problems. Computer Methods Appl Mech Eng. 2019;345:75–99. https://doi.org/10.1016/j.cma.2018.10.029.
36. Galletti B, Bruneau CH, Zannetti L, Iollo A. Low-order modelling of laminar flow regimes past a confined square cylinder. J Fluid Mech. 2004;503:161–70. https://doi.org/10.1017/s0022112004007906.
37. Couplet M, Basdevant C, Sagaut P. Calibrated reduced-order POD-Galerkin system for fluid flow modelling. J Comput Phys. 2005;207(1):192–220. https://doi.org/10.1016/j.jcp.2005.01.008.
38. Noack BR, Papas P, Monkewitz PA. The need for a pressure-term representation in empirical Galerkin models of incompressible shear flows. J Fluid Mech. 2005;523:339–65. https://doi.org/10.1017/S0022112004002149.
39. Hijazi S, Ali S, Stabile G, Ballarin F, Rozza G. The Effort of Increasing Reynolds Number in Projection-Based Reduced Order Methods: From Laminar to Turbulent Flows. In: Lecture Notes in Computational Science and Engineering, pp. 245–264. Cham: Springer; 2020. https://doi.org/10.1007/978-3-030-30705-9_22
40. Hijazi S, Stabile G, Mola A, Rozza G. Data-driven POD-Galerkin reduced order model for turbulent flows. J Comput Phys. 2020;416: 109513. https://doi.org/10.1016/j.jcp.2020.109513.
41. Mou C, Koc B, San O, Rebholz LG, Iliescu T. Data-driven variational multiscale reduced order models. Computer Methods Appl Mech Eng. 2021;373: 113470. https://doi.org/10.1016/j.cma.2020.113470.
42. Fresca S, Dede' L, Manzoni A. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. J Sci Comput. 2021. https://doi.org/10.1007/s10915-021-01462-7.

43. Fresca S, Manzoni A. POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. Computer Methods Appl Mech Eng. 2022;388: 114181. https://doi.org/10.1016/j.cma.2021.114181.
44. Romor F, Stabile G, Rozza G. Non-linear manifold ROM with Convolutional Autoencoders and Reduced Over-Collocation method. 2022. https://doi.org/10.48550/ARXIV.2203.00360
45. Lee H, Kang IS. Neural algorithm for solving differential equations. J Comput Phys. 1990;91(1):110–31. https://doi.org/10.1016/0021-9991(90)90007-n.
46. Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw. 1998;9(5):987–1000. https://doi.org/10.1109/72.712178.
47. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019;378:686–707. https://doi.org/10.1016/j.jcp.2018.10.045.
48. Raissi M, Wang Z, Triantafyllou MS, Karniadakis GE. Deep learning of vortex-induced vibrations. J Fluid Mech. 2018;861:119–37. https://doi.org/10.1017/jfm.2018.872.
49. Sirignano J, Spiliopoulos K. DGM: A deep learning algorithm for solving partial differential equations. J Comput Phys. 2018;375:1339–64. https://doi.org/10.1016/j.jcp.2018.08.029.
50. Eivazi H, Tahani M, Schlatter P, Vinuesa R. Physics-informed neural networks for solving reynolds-averaged navier-stokes equations. Phys Fluids. 2022;34(7): 075117. https://doi.org/10.1063/5.0095270.
51. Chen W, Wang Q, Hesthaven JS, Zhang C. Physics-informed machine learning for reduced-order modeling of nonlinear problems. J Comput Phys. 2021;446: 110666. https://doi.org/10.1016/j.jcp.2021.110666.
52. Banks HT, Kunisch K. Estimation Techniques for Distributed Parameter Systems. Boston: Birkhäuser; 1989. https://doi.org/10.1007/978-1-4612-3700-6.
53. Kirsch A. An Introduction to the Mathematical Theory of Inverse Problems. New York: Springer; 2011. https://doi.org/10.1007/978-1-4419-8474-6.
54. Stuart AM. Inverse problems: A Bayesian perspective. Acta Numerica. 2010;19:451–559. https://doi.org/10.1017/s0962492910000061.
55. Kaipio J, Somersalo E. Statistical and Computational Inverse Problems. New York: Springer; 2005. https://doi.org/10.1007/b138659.
56. Matthies HG, Zander E, Rosić BV, Litvinenko A. Pajonk O. In: Ibrahimbegovic A, editor. Inverse Problems in a Bayesian Setting. Cham: Springer; 2016. p. 245–86. https://doi.org/10.1007/978-3-319-27996-1_10.
57. Dashti M, Stuart AM. The Bayesian Approach to Inverse Problems, 2017;311–428. https://doi.org/10.1007/978-3-319-12385-1_7
58. Cotter SL, Dashti M, Stuart AM. Approximation of Bayesian inverse problems for PDEs. Med Res. 2010;48(1):322–45. https://doi.org/10.1137/090770734.
59. Marzouk YM, Najm HN. Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems. J Comput Phys. 2009;228(6):1862–902. https://doi.org/10.1016/j.jcp.2008.11.024.
60. Warner JE, Aquino W, Grigoriu MD. Stochastic reduced order models for inverse problems under uncertainty. Computer Methods Appl Mech Eng. 2015;285:488–514. https://doi.org/10.1016/j.cma.2014.11.021.
61. Ji W, Ren Z, Marzouk Y, Law CK. Quantifying kinetic uncertainty in turbulent combustion simulations using active subspaces. Proce Combustion Instit. 2019;37(2):2175–82. https://doi.org/10.1016/j.proci.2018.06.206.
62. Cui T, Marzouk YM, Willcox KE. Data-driven model reduction for the Bayesian solution of inverse problems. Int J Numer Methods Eng. 2014;102:5. https://doi.org/10.1002/nme.4748.
63. Galbally D, Fidkowski K, Willcox K, Ghattas O. Non-linear model reduction for uncertainty quantification in large-scale inverse problems. Int J Numer Methods Eng. 2009. https://doi.org/10.1002/nme.2746.
64. Garmatter D, Haasdonk B, Harrach B. A reduced basis Landweber method for nonlinear inverse problems. Res Method. 2016;32(3): 035001. https://doi.org/10.1088/0266-5611/32/3/035001.
65. Himpe C, Ohlberger M. Data-driven combined state and parameter reduction for inverse problems. Advan Comput Math. 2015;41(5):1343–64. https://doi.org/10.1007/s10444-015-9420-5.
66. Moukalled F, Mangani L, Darwish M. The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab, 1st edn. Cham: Springer; 2015. https://doi.org/10.1007/978-3-319-16874-6
67. Weller HG, Tabor G, Jasak H, Fureby C. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers Phys. 1998;12(6):620–31. https://doi.org/10.1063/1.168744.
68. Jasak H. Error analysis and estimation for the finite volume method with applications to fluid flows. PhD thesis, Imperial College, University of London. 1996.
69. Boussinesq J. Essa sur latheories des eaux courantes. memoires presentes par divers savants a l'academic des sciences de l'institut national de france. Tome XXIII (1), 1877.
70. Spalart P, Allmaras S. A one-equation turbulence model for aerodynamic flows. In: 30th Aerospace Sciences Meeting and Exhibit. American Institute of Aeronautics and Astronautics, Reno,NV,U.S.A. 1992. https://doi.org/10.2514/6.1992-439
71. Hanjalic K, Launder BE. A Reynolds stress model of turbulence and its application to thin shear flows. J Fluid Mech. 1972;52(04):609. https://doi.org/10.1017/s002211207200268x.
72. Menter FR. Two-equation eddy-viscosity turbulence models for engineering applications. AIAA J. 1994;32(8):1598–605. https://doi.org/10.2514/3.12149.
73. Smagorinsky J. General circulation experiments with the primitive equations i. the basic experiment. Monthly Weather Review, 1963;91(3), 99–164. 10.1175/1520-0493(1963)091<0099:gcewtp>2.3.co;2
74. Germano M, Piomelli U, Moin P, Cabot WH. A dynamic subgrid-scale eddy viscosity model. Phys Fluids A: Fluid Dyn. 1991;3(7):1760–5. https://doi.org/10.1063/1.857955.
75. Kim W-W, Menon S. A new dynamic one-equation subgrid-scale model for large eddy simulations. In: 33rd Aerospace Sciences Meeting and Exhibit. American Institute of Aeronautics and Astronautics, Reno,NV,U.S.A. 1995. https://doi.org/10.2514/6.1995-356

76. Sirovich L. Turbulence and the Dynamics of Coherent Structures part I: Coherent Structures. Quart Appl Mathe. 1987;45(3):561–71. https://doi.org/10.1090/qam/910464.
77. Ballarin F, Manzoni A, Quarteroni A, Rozza G. Supremizer stabilization of POD-Galerkin approximation of parametrized steady incompressible Navier-Stokes equations. Int J Numer Methods Eng. 2014;102(5):1136–61. https://doi.org/10.1002/nme.4772.
78. Rozza G, Veroy K. On the stability of the reduced basis method for Stokes equations in parametrized domains. Computer Methods Appl Mech Eng. 2007;196(7):1244–60. https://doi.org/10.1016/j.cma.2006.09.005.
79. Johnston H, Liu J-G. Accurate, stable and efficient Navier-Stokes solvers based on explicit treatment of the pressure term. J Comput Phys. 2004;199(1):221–59. https://doi.org/10.1016/j.jcp.2004.02.009.
80. Liu J-G, Liu J, Pego RL. Stable and accurate pressure approximation for unsteady incompressible viscous flow. J Comput Phys. 2010;229(9):3428–53. https://doi.org/10.1016/j.jcp.2010.01.010.
81. Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: A survey. J Mach Learn Res. 2017;18(1):5595–637.
82. Graham WR, Peraire J, Tang KY. Optimal control of vortex shedding using low-order models. Part I-open-loop model development. Int J Numer Methods Eng. 1999;44(7):945–72.
83. Gunzburger MD, Peterson JS, Shadid JN. Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data. Computer Methods Appl Mech Eng. 2007;196(4–6):1030–47. https://doi.org/10.1016/j.cma.2006.08.004.
84. Hijazi S, Stabile G, Mola A, Rozza G. Non-intrusive Polynomial Chaos Method Applied to Full-Order and Reduced Problems in Computational Fluid Dynamics: A Comparison and Perspectives. Cham: Springer; 2020. p. 217–40. https://doi.org/10.1007/978-3-030-48721-8_10.
85. Bizon K, Continillo G. Reduced order modelling of chemical reactors with recycle by means of POD-penalty method. Computers Chem Eng. 2012;39:22–32. https://doi.org/10.1016/j.compchemeng.2011.10.001.
86. Babuška I. The finite element method with penalty. Math Comput. 1973;27(122):221–221. https://doi.org/10.1090/s0025-5718-1973-0351118-5.
87. Barrett JW, Elliott CM. Finite element approximation of the dirichlet problem using the boundary penalty method. Numerische Mathematik. 1986;49(4):343–66. https://doi.org/10.1007/bf01389536.
88. Kalashnikova I, Barone MF. Efficient non-linear proper orthogonal decomposition/Galerkin reduced order models with stable penalty enforcement of boundary conditions. Int J Numer Methods Eng. 2012;90(11):1337–62. https://doi.org/10.1002/nme.3366.
89. Sirisup S, Karniadakis GE. Stability and accuracy of periodic flow solutions obtained by a POD-penalty method. Physica D: Nonlinear Phenomena. 2005;202(3–4):218–37. https://doi.org/10.1016/j.physd.2005.02.006.
90. Hijazi S. Reduced order methods for laminar and turbulent flows in a finite volume setting: projection-based methods and data-driven techniques. dissertation, SISSA. 2020. http://hdl.handle.net/20.500.11767/114353
91. Stabile G, Rozza G. ITHACA-FV - In real Time Highly Advanced Computational Applications for Finite Volumes. http://www.mathlab.sissa.it/ithaca-fv. Accessed 2018-01-30
92. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray D, Steiner B, Tucker P, Vasudevan V, Warden P, Zhang X. Tensorflow: A system for large-scale machine learning. 2016.
93. Shah KB, Ferziger JH. A fluid mechanicians view of wind engineering: Large eddy simulation of flow past a cubic obstacle. J Wind Eng Ind Aerodyn. 1997;67–68:211–24. https://doi.org/10.1016/s0167-6105(97)00074-3.
94. Breuer M, Lakehal D, Rodi W. Flow around a surface mounted cubical obstacle: Comparison of LES and RANS-results. In: Notes on Numerical Fluid Mechanics (NNFM), pp. 22–30. Vieweg+Teubner Verlag, Cham; 1996. https://doi.org/10.1007/978-3-322-89838-8_4
95. Krajnovic S, Davidson L. Large-eddy simulation of the flow around a bluff body. AIAA J. 2002;40(5):927–36. https://doi.org/10.2514/2.1729.

Publisher's Note