



## Article

# Real-Time Simulation of Parameter-Dependent Fluid Flows through Deep Learning-Based Reduced Order Models

Stefania Fresca  and Andrea Manzoni \* 

MOX—Dipartimento di Matematica, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy; stefania.fresca@polimi.it

\* Correspondence: andrea1.manzoni@polimi.it

**Abstract:** Simulating fluid flows in different virtual scenarios is of key importance in engineering applications. However, high-fidelity, full-order models relying, e.g., on the finite element method, are unaffordable whenever fluid flows must be simulated in almost real-time. Reduced order models (ROMs) relying, e.g., on proper orthogonal decomposition (POD) provide reliable approximations to parameter-dependent fluid dynamics problems in rapid times. However, they might require expensive hyper-reduction strategies for handling parameterized nonlinear terms, and enriched reduced spaces (or Petrov–Galerkin projections) if a mixed velocity–pressure formulation is considered, possibly hampering the evaluation of reliable solutions in real-time. Dealing with fluid–structure interactions entails even greater difficulties. The proposed deep learning (DL)-based ROMs overcome all these limitations by learning, in a nonintrusive way, both the nonlinear trial manifold and the reduced dynamics. To do so, they rely on deep neural networks, after performing a former dimensionality reduction through POD, enhancing their training times substantially. The resulting POD-DL-ROMs are shown to provide accurate results in almost real-time for the flow around a cylinder benchmark, the fluid–structure interaction between an elastic beam attached to a fixed, rigid block and a laminar incompressible flow, and the blood flow in a cerebral aneurysm.

**Keywords:** fluid dynamics; deep learning; reduced order modeling; proper orthogonal decomposition; Navier–Stokes equations; fluid–structure interaction



**Citation:** Fresca, S.; Manzoni, A. Real-Time Simulation of Parameter-Dependent Fluid Flows through Deep Learning-Based Reduced Order Models. *Fluids* **2021**, *6*, 259. <https://doi.org/10.3390/fluids6070259>

Academic Editor: Toni Lassila

Received: 31 May 2021

Accepted: 5 July 2021

Published: 18 July 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Computational fluid dynamics nowadays provide rigorous and reliable tools for the numerical approximation of fluid flows equations that are exploited in several fields, from life sciences to aeronautical engineering. High-fidelity techniques such as, e.g., finite elements, finite volumes as well as spectral methods have been extensively applied in the past decades to the simulation of challenging problems in fluid dynamics, providing quantitative indication about the physical behavior of the system in view of its better understanding, control, and forecasting. Solving these problems entails the numerical approximation of unsteady Navier–Stokes (NS) equations in three-dimensional domains, possibly accounting for fluid–structure interaction (FSI) effects, requiring fine computational meshes, in case one aims at simulating complex flow patterns, and ultimately yielding large-scale nonlinear systems of equations to be solved.

Simulating fluid flows in complex configurations through high-fidelity, full-order models (FOMs) is computationally infeasible if one aims at solving the problem multiple times for different virtual scenarios or in a very small amount of time—at the limit, in real-time. This is the case, for instance, of blood flow simulations, for which outputs of clinical interest shall be evaluated for different flow conditions and in different geometrical configurations [1]. In this respect, if quantitative outputs are meant to support clinicians' decisions, each new numerical simulation should be carried out very rapidly on deployed platforms, rather than exploiting huge parallel hardware architectures, and thus requiring limited data storage and memory capacity.

In the case virtual scenarios can be described in terms of—e.g., physical and/or geometrical—input parameters, reduced order models (ROMs) built, e.g., through the reduced basis (RB) method [2], can be exploited to reduce the computational complexity and costs entailed by the repeated solution of parameterized fluid flow problems, enabling dramatic reduction of the dimension of the discrete problems arising from numerical approximation of millions to hundreds, or thousands at most, of variables. Several works have addressed the construction of rapid and reliable ROMs for Navier–Stokes equations, mainly exploiting either proper orthogonal decomposition (POD) [3–8] or greedy algorithms [9–12] for the construction of reduced order spaces. However, despite the general principles behind projection-based reduction techniques—such as, e.g., the use of a (Petrov–)Galerkin projection onto a low-dimensional subspace, and the use of a set of FOM snapshots computed for different input parameter values at different times to train the ROM—that provide a rigorous framework to set up ROMs for fluid dynamics equations, some distinguishing properties of Navier–Stokes equations for incompressible flow simulations ultimately make their effective realization quite involved [13]. Among them, we mention the need for (i) efficiently treating nonlinearities and parameter dependencies [14], (ii) approximating both velocity and pressure [15], (iii) ensuring the ROM stability (with respect to both the violation of the inf–sup condition and dominating convection) [16,17], and (iv) keeping error propagation in time under control. The presence of FSI, coupling the fluid model with a model describing the structural displacement of the non-rigid domain where the fluid flows, makes the problem even more involved. Several strategies have been proposed to address these issues: for instance, hyper-reduction techniques have been devised in a purely algebraic way to treat the nonaffine and nonlinear convective terms appearing in the NS equations [8]; suitable enrichment of the velocity space can be considered to ensure the inf–sup stability of the ROM [7,18] as well as alternative, more effective, stabilization techniques for the ROM [19]; mesh-moving techniques have been exploited to efficiently parameterize domain shapes to address geometric variability in fluid flow simulations [8], and either monolithic or segregated strategies have been considered as first attempts to handle fluid–structure interactions in the RB method for parameterized fluid flows [20,21].

On the other hand, machine learning techniques—in particular, artificial neural networks (NNs)—in computational fluid dynamics have witnessed a dramatic blooming in the past ten years [22,23]. Deep neural networks (DNNs) have been exploited to address several issues; a nonexhaustive list includes, for instance:

1. The extraction of relevant flow features, such as recirculation regions or boundary layers through convolutional neural networks (CNNs) [24].
2. The construction of inexpensive, nonintrusive approximations for output quantities of interest for fluid flows [25], or to velocity and pressure field, obtained through Reynolds-averaged Navier–Stokes (RANS) equations [26–28].
3. Data-driven turbulence models in RANS equations through a physics-informed machine learning approach [29], or data-driven eddy viscosity closure models in large eddy simulations (LES) [30].
4. The setting of closure models to stabilize a POD-Galerkin ROM [31] by using, e.g., recurrent neural networks (RNNs) to predict the impact of the unresolved scales on the resolved scales [32], or correction models to adapt a ROM to describe scenarios quite far from the ones seen during the training stage [33].
5. The reconstruction of a high-resolution flow field from limited flow information [34] as well as the assimilation of flow measurements and computational flow dynamics models derived from first physical principles. This task can be cast in the framework of the so-called physics-informed neural networks [35,36], where NNs are trained to solve supervised learning tasks while respecting the fluid dynamics equations, or tackled by means of Bayesian neural networks [37].
6. The nonintrusive estimation of POD coefficients through, e.g., feedforward NNs [38–40] or probabilistic NNs [41].

In this paper, we apply the POD-DL-ROM technique that we recently proposed [42] to fluid flow problems in order to build nonintrusive and extremely efficient ROMs for parameter-dependent unsteady problems in computational fluid dynamics by exploiting (i) deep neural networks as main building block, (ii) a set of FOM snapshots, and (iii) dimensionality reduction of FOM snapshots through (randomized) POD. Even though a preliminary example of its application to a benchmark in fluid dynamics has already been considered in [42] to assess the capability of POD-DL-ROMs to handle vector nonlinear problems such as the unsteady Navier–Stokes equations, in order to compute the fluid velocity field only, in this paper we deepen our analysis by considering: (a) the computation of both velocity and pressure fields in the case of unsteady Navier–Stokes equations, (b) the extension to an FSI problem, and (c) the application to a real-life application of interest, namely the simulation of blood flows through a cerebral aneurysm.

Compared to other works that have recently appeared in the literature, our focus is on parameter-dependent fluid dynamics problems, either involving complex three-dimensional geometries or FSI effects, and on the use of deep learning (DL)-based ROMs for the sake of real-time simulation of fluid flows, thus relying on nonlinear reduction techniques. Motivated by similar goals, nonintrusive ROMs for fluid dynamics equations have been proposed, e.g., in [43–45], where POD has been considered to generate low-dimensional (linear) subspaces, also in the case of FSI problems, and POD coefficients at each time step are either computed through a radial basis function multi-dimensional interpolation, or extrapolated from the POD coefficients at earlier time steps.

Applications of DL algorithms in conjunction with POD have already been proposed for the sake of long-term predictions in time, however without addressing parameter-dependent problems. For instance, a long short-term memory (LSTM) network was used to learn the underlying physical dynamics in [46], generating a nonintrusive ROM through the solution snapshots acquired over time. Deep feedforward neural networks (DFNNs) have been used for a similar task in [47] and compared with the sparse identification of nonlinear dynamics (SINDy) algorithm [48]. This latter defines a sparse representation through a linear combination of selected functions, and has been used for data-driven forecasting in fluid dynamics [49]. RNNs have been considered in [50,51] to evolve low-dimensional states of unsteady flows, exploiting either POD or a convolutional recurrent autoencoder to extract low-dimensional features from snapshots. DL algorithms have also been used in [52] to describe the reduced trial manifold where the approximation is sought, then relying on a minimum residual formulation to derive the ROM—hence, still requiring the assembling and the solution of a ROM as in traditional POD-Galerkin ROMs.

The structure of the paper is as follows. In Section 2, we sketch the basic features of projection-based ROMs for fluid flows and recall the main components of the POD-DL-ROM technique. In Section 3, we show some numerical results obtained for the flow around a cylinder benchmark, the fluid–structure interaction between an elastic beam attached to a fixed, rigid block and a laminar incompressible flow, and the blood flow in a cerebral aneurysm. Finally, a brief discussion of our results and a few comments about future research directions are reported in Section 4.

## 2. Methods

In this section, we briefly recall the main components of the POD-enhanced DL-based ROMs (briefly, POD-DL-ROMs) that we adapt, in the following, to handle problems in computational fluid dynamics. In particular, we aim at simulating parameter-dependent unsteady fluid flows, relying on a velocity–pressure formulation in domains that have either (i) rigid walls or (ii) elastic deformable walls.

In the case of rigid walls, for any input parameter vector  $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu}$ , we aim at solving the nonlinear unsteady Navier–Stokes equations in a given, fixed domain  $\Omega^F \subset \mathbb{R}^d$ ,  $d = 2, 3$  (see Section 3.1)

$$P_F(\mathbf{v}_h, p_h; t, \boldsymbol{\mu}) = 0 \quad \text{in } \Omega^F \times (0, T), \quad (1)$$

in the time interval  $(0, T)$ , provided that suitable initial (at time  $t = 0$ ) and boundary conditions (on  $\partial\Omega^F$ , for each  $t \in (0, T)$ ) are assigned. Here,  $t \in (0, T)$  is the time variable,  $\mathbf{v}_h = \mathbf{v}_h(t; \boldsymbol{\mu})$  the velocity field,  $p_h = p_h(t; \boldsymbol{\mu})$  the pressure field; these two latter quantities are usually obtained through a FOM built, e.g., through the finite element method. Here,  $h > 0$  denotes a discretization parameter, usually related to the mesh size.

In the case of elastic walls, the fluid domain is unknown, and its deformation introduces a further geometric nonlinearity; the structure displacement  $\mathbf{d}_h^S = \mathbf{d}_h^S(t; \boldsymbol{\mu})$  might also be nonlinear, and must match the one of the fluid domain  $\mathbf{d}_h^G = \mathbf{d}_h^G(t; \boldsymbol{\mu})$  at the fluid–structure (FS) interface  $\Sigma(t)$ . Here, we employ the so-called arbitrary Lagrangian–Eulerian (ALE) approach, in which an extra problem for the fluid domain displacement (usually a harmonic extension of the FS interface datum) is solved, thus providing an updated fluid domain, while the fluid problem is reformulated on a frame of reference that moves with the fluid domain. Thus, for any input parameter vector  $\boldsymbol{\mu} \in \mathcal{P}$ , we consider a fluid–structure interaction (FSI) model, which consists of a two-field problem, coupling the incompressible Navier–Stokes equations written in the ALE form with the (non)linear elastodynamics equation modeling of solid deformation [53]. In particular, we aim at solving the unsteady Navier–Stokes equations in a varying domain  $\Omega^F(t) \subset \mathbb{R}^d$ , the elastodynamics equations in the structural domain  $\Omega^S \subset \mathbb{R}^d$ , and a geometric problem in the fixed fluid domain  $\Omega^F \subset \mathbb{R}^d$  (see Section 3.2),

$$\begin{cases} P_F(\mathbf{v}_h, p_h; t, \boldsymbol{\mu}) = 0 & \text{in } \Omega^F(t) \times (0, T), \\ P_S(\mathbf{d}_h^S; t, \boldsymbol{\mu}) = 0 & \text{in } \Omega^S \times (0, T), \\ P_G(\mathbf{d}_h^G; \boldsymbol{\mu}) = 0 & \text{in } \Omega^F \times (0, T), \\ \text{coupling conditions} & \text{on } \Sigma(t) \times (0, T), \end{cases} \quad (2)$$

for a time interval  $(0, T)$ , provided that suitable initial (at time  $t = 0$ ) and boundary conditions (on  $\partial\Omega^F(t) \setminus \Sigma(t)$  for the fluid subproblem, on  $\partial\Omega^S \setminus \Sigma$  for the structural subproblem, on  $\partial\Omega^F$  for the geometric subproblem, for each  $t \in (0, T)$ ), are assigned.

### 2.1. Projection-Based ROMs: Main Features

The spatial discretization of problem (1) or (2) through finite elements yields a nonlinear dynamical system of dimension  $N_h$  to be solved for each input parameter value; then, a fully discretized problem is obtained relying, e.g., on either semi-implicit or implicit methods introducing a partition of the interval  $[0, T]$  in  $N_t$  subintervals of equal size  $\Delta t = T/N_t$ , such that  $t^k = k\Delta t$ . This results in a sequence of either linear or nonlinear algebraic systems to be solved at each time step  $t^k, k = 1, \dots, N_t$ —which we refer to as the high-fidelity FOM. Note that the dimension  $N_h$  accounts for the degrees of freedom of either the fluid problem (involving velocity and pressure) or the FSI problem (also including the structural and the geometrical subproblem). Building a projection-based ROM through, e.g., the RB method then requires performing this calculation for  $n_s$  selected parameter values  $\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^{n_s}$ , and to perform POD on the solution snapshots (obtained for each  $\boldsymbol{\mu}^j, j = 1, \dots, n_s$ , and for each time step  $t^k, k = 1, \dots, N_t$ ). Focusing, for the sake of simplicity, on the fluid problem (1), the RB approximation of velocity and pressure fields at time  $t^k$  is expressed as a linear combination of the RB basis functions,

$$\mathbf{v}_h(t^k; \boldsymbol{\mu}) \approx \mathbf{V}^v \mathbf{v}_N(t^k; \boldsymbol{\mu}), \quad p_h(t^k; \boldsymbol{\mu}) \approx \mathbf{V}^p \mathbf{p}_N(t^k; \boldsymbol{\mu})$$

where  $\mathbf{V}^v \in \mathbb{R}^{N_h \times N_v}$  and  $\mathbf{V}^p \in \mathbb{R}^{N_h \times N_p}$  denote the matrices whose columns form the basis for the velocity and the pressure RB spaces, respectively, and are selected as the first left singular vectors of the (velocity and pressure) snapshots matrices. Note that in this case, the RB approximation is sought in a linear trial manifold. A similar approximation also holds for the additional variables appearing in the FSI problem (2). The reduced dynamics are then obtained by solving a low-dimensional dynamical system, obtained by performing

a Galerkin projecting of the FOM onto the spaces spanned by the RB spaces; alternatively, a Petrov–Galerkin projection could also be used.

Projection-based ROMs for parameterized PDEs thus rely on a suitable offline–online computational splitting: computationally expensive tasks required to build the low-dimensional subspaces and to assemble the ROM arrays, are performed once for all in the so-called offline (or ROM training) stage. This latter allows us to compute—ideally—in an extremely efficient way the ROM approximation for any new parameter value during the so-called online (or ROM testing) stage. This splitting, however, might be compromised if (i) the dimension of the linear trial subspace becomes very large (compared to the intrinsic dimension of the solution manifold being approximated), such as in the case of problems featuring coherent structures that propagate over time such as transport, wave, or convection-dominated phenomena, or (ii) hyper-reduction techniques, required to approximate  $\mu$ -dependent nonlinear terms, require linear subspaces whose dimension is also very large. Even more importantly, two additional issues make the construction of ROMs quite critical in the case of fluid dynamics problems, especially in the following cases.

1. A Galerkin projection onto the RB space built through the POD procedure above does not ensure the stability of the resulting ROM (in the sense of the fulfillment of an inf–sup condition at the reduced level). Several strategies can be employed to overcome this issue such as, e.g., (a) the augmentation of the velocity space by means of a set of enriching basis functions computed through the so-called pressure supremizing operator, which depends on the divergence term; (b) the use of a Petrov–Galerkin (e.g., least squares, (LS)) RB method, or (c) the use of a stabilized FOM (such as, e.g., a P1-P1 streamline upwind Petrov–Galerkin (SUPG) finite element method); (d) an independent treatment of the pressure, to be reconstructed from the velocity by solving a Poisson equation, in the case divergence-free velocity basis functions, are used—an assumption that might be hard to fulfill.
2. The need for dealing with both a mixed formulation and a coupled FSI problem requires the construction of a reduced space for each variable, no matter if one is interested in the evaluation of output quantities of interest only involving a single variable. For instance, even if one is interested in the evaluation of fluid velocity in the FSI case, a projection-based ROM must account for all the variables appearing as unknowns in the coupled FSI problem. The same consideration also holds in the case of a fluid problem, where the pressure must be treated as an unknown of the ROM problem even if one is not interested in its evaluation.

## 2.2. POD-Enhanced DL-ROMs (POD-DL-ROMs)

POD-DL-ROMs are nonintrusive ROMs which aim at approximating the map  $(t, \mu) \rightarrow \mathbf{u}_h(t, \mu)$ , for any field variable of interest  $\mathbf{u}_h(t, \mu)$  by describing both the trial manifold and the reduced dynamics through deep neural networks. These latter are trained on a set of FOM snapshots

$$\mathbf{S}_u = [\mathbf{u}_h(t^1; \mu_1) \mid \dots \mid \mathbf{u}_h(t^{N_i}; \mu_1) \mid \dots \mid \dots \mid \mathbf{u}_h(t^1; \mu_{N_{train}}) \mid \dots \mid \mathbf{u}_h(t^{N_i}; \mu_{N_{train}})], \quad (3)$$

computed for different parameter values  $\mu_1, \dots, \mu_{N_{train}} \in \mathcal{P}$ , suitably sampled over the parameter space at different time instants  $\{t^1, \dots, t^{N_i}\} \subset [0, T]$ . Avoiding the *projection* stage, POD-DL-ROMs can be cheaply evaluated once trained, only involving those variables one is interested in. In case multiple variables are involved (e.g., both velocity and pressure), the procedure below can be performed simultaneously on each of them.

To reduce the dimensionality of the snapshots and avoid feeding training data of very large dimension  $N_h$ , we first apply POD—realized through randomized SVD (rSVD)—to the snapshot set  $\mathbf{S}_u$ ; then, a DL-ROM is built to approximate the map between  $(t, \mu)$  and the POD generalized coordinates. Using rSVD, we build  $N$ -dimensional subspace  $\text{Col}(\mathbf{V}_N)$  spanned by the  $N \leq N_h$  columns of  $\mathbf{V}_N \in \mathbb{R}^{N_h \times N}$ , the matrix of the first  $N$  singular vectors of the snapshot matrix  $\mathbf{S}_u$ . Here,  $N$  denotes the dimension of the linear manifold, which

can be taken (much) larger than the one of the reduced linear trial manifold used in a POD-Galerkin ROM.

Hence, the POD-DL-ROM approximation of the FOM solution  $\mathbf{u}_h(t; \boldsymbol{\mu})$  is

$$\tilde{\mathbf{u}}_h(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D) = \mathbf{V}_N \tilde{\mathbf{u}}_N(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D) \approx \mathbf{u}_h(t; \boldsymbol{\mu}),$$

that is, it is sought in a linear trial manifold of (potentially large) dimension  $N$ ,

$$\tilde{\mathcal{S}}_h^N = \{ \mathbf{V}_N \tilde{\mathbf{u}}_N(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D) \mid \tilde{\mathbf{u}}_N(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D) \in \mathbb{R}^N, t \in [0, T], \boldsymbol{\mu} \in \mathcal{P} \} \subset \mathbb{R}^{N_h}, \quad (4)$$

by applying the DL-ROM strategy [54] to approximate  $\mathbf{V}_N^T \mathbf{u}_h(t; \boldsymbol{\mu})$ —rather than directly  $\mathbf{u}_h(t; \boldsymbol{\mu})$ . The DL-ROM approximation  $\tilde{\mathbf{u}}_N(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D) \approx \mathbf{V}_N^T \mathbf{u}_h(t; \boldsymbol{\mu})$  takes the form

$$\tilde{\mathbf{u}}_N(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D) = \mathbf{f}_N^D(\boldsymbol{\phi}_n^{DF}(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}); \boldsymbol{\theta}_D), \quad (5)$$

and is sought in a reduced nonlinear trial manifold  $\tilde{\mathcal{S}}_N^n$  of very small dimension  $n \ll N$ ; usually,  $n \approx n_\mu + 1$ —here time is considered as an additional parameter. As for DL-ROMs (see, e.g., [54]), both the reduced dynamics and the reduced nonlinear manifold (or trial manifold) where the ROM solution is sought must be learnt. In particular:

- *Reduced dynamics learning.* To describe the system dynamics on the nonlinear trial manifold  $\tilde{\mathcal{S}}_N^n$ , the intrinsic coordinates of the approximation  $\tilde{\mathbf{u}}_N$  are defined as

$$\mathbf{u}_n(t; \boldsymbol{\mu}) = \boldsymbol{\phi}_n^{DF}(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}),$$

where  $\boldsymbol{\phi}_n(\cdot; \cdot, \boldsymbol{\theta}_{DF}) : [0, T] \times \mathbb{R}^{n_\mu+1} \rightarrow \mathbb{R}^n$  is a DFNN consisting of the repeated composition of a nonlinear activation function, applied to a linear transformation of the input multiple times. Here,  $\boldsymbol{\theta}_{DF}$  denotes the DFNN parameters vector, collecting the weights and biases of each of its layers;

- *Nonlinear trial manifold learning.* To model the reduced nonlinear trial manifold  $\tilde{\mathcal{S}}_N^n$ , we employ the decoder function of a convolutional autoencoder (CAE), that is,

$$\tilde{\mathcal{S}}_N^n = \{ \tilde{\mathbf{u}}_N(t; \boldsymbol{\mu}) = \mathbf{f}_N^D(\boldsymbol{\phi}_n^{DF}(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}); \boldsymbol{\theta}_D) \mid \mathbf{u}_n(t; \boldsymbol{\mu}, \boldsymbol{\theta}_{DF}) \in \mathbb{R}^n, t \in [0, T], \boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_\mu} \} \subset \mathbb{R}^N, \quad (6)$$

where  $\mathbf{f}_N^D(\cdot; \boldsymbol{\theta}_D) : \mathbb{R}^n \rightarrow \mathbb{R}^N$  denotes the decoder function of a CAE obtained as the composition of several layers (some of which are convolutional), depending upon a vector  $\boldsymbol{\theta}_D$  collecting all the corresponding weights and biases.

Finally, the encoder function  $\mathbf{f}_n^E(\cdot; \boldsymbol{\theta}_E) : \mathbb{R}^N \rightarrow \mathbb{R}^n$ —depending upon a vector  $\boldsymbol{\theta}_E$  of parameters—of the CAE can be used to map the intrinsic coordinates  $\mathbf{V}_N^T \mathbf{u}_h(t, \boldsymbol{\mu})$  associated to  $(t, \boldsymbol{\mu})$  onto a low-dimensional representation

$$\tilde{\mathbf{u}}_n(t; \boldsymbol{\mu}, \boldsymbol{\theta}_E) = \mathbf{f}_n^E(\mathbf{V}_N^T \mathbf{u}_h(t; \boldsymbol{\mu}); \boldsymbol{\theta}_E).$$

Hence, training a POD-DL-ROM requires to solve the optimization problem

$$\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{N_{train} N_t} \sum_{i=1}^{N_{train}} \sum_{k=1}^{N_t} \mathcal{L}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}), \quad (7)$$

where the *per-example* loss function  $\mathcal{L}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta})$  is given by the sum of two terms,

$$\mathcal{L}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}) = \frac{\omega_h}{2} \mathcal{L}_{rec}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}) + \frac{1 - \omega_h}{2} \mathcal{L}_{int}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}); \quad (8)$$

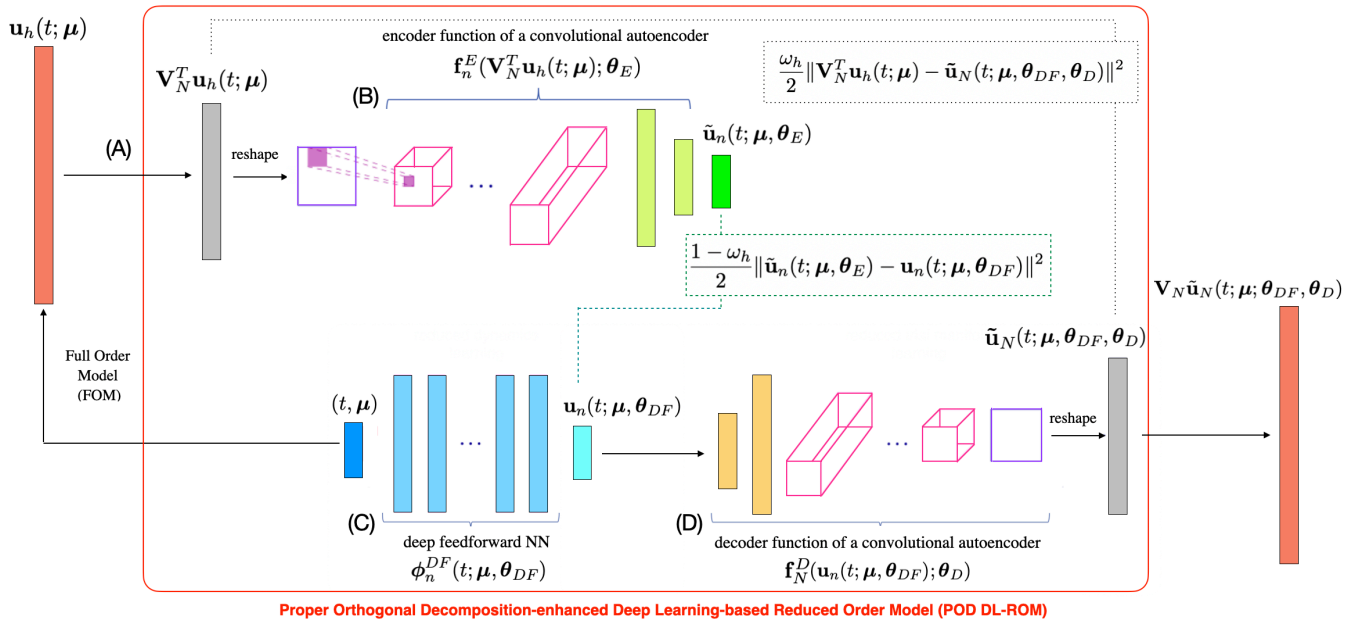
the former is the reconstruction error between the FOM and the POD-DL-ROM solutions,

$$\mathcal{L}_{rec}(t^k, \boldsymbol{\mu}_i; \boldsymbol{\theta}) = \|\mathbf{V}_N^T \mathbf{u}_h(t^k; \boldsymbol{\mu}_i) - \tilde{\mathbf{u}}_N(t^k; \boldsymbol{\mu}_i, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D)\|^2;$$

the latter is the misfit between the intrinsic coordinates and the output of the encoder,

$$\mathcal{L}_{int}(t^k, \mu_i; \theta) = \|\tilde{\mathbf{u}}_n(t^k; \mu_i; \theta_E) - \mathbf{u}_n(t^k; \mu_i; \theta_{DF})\|^2.$$

Finally,  $\omega_h \in [0, 1]$  is a prescribed weighting parameter. The architecture of the POD-DL-ROM neural network is shown in Figure 1.



**Figure 1.** POD-DL-ROM architecture. Starting from the FOM solution  $\mathbf{u}_h(t; \mu)$ , the intrinsic coordinates  $\mathbf{V}_N^T \mathbf{u}_h(t; \mu)$  are computed by means of rSVD; their approximation  $\tilde{\mathbf{u}}_N(t; \mu)$  is provided by the neural network as output so that the reconstructed solution  $\tilde{\mathbf{u}}_h(t; \mu) = \mathbf{V}_N \tilde{\mathbf{u}}_N(t; \mu)$  is recovered through the rPOD basis matrix. In particular, the intrinsic coordinates  $\mathbf{V}_N^T \mathbf{u}_h(t; \mu)$  are provided as input to block (B) which outputs  $\tilde{\mathbf{u}}_n(t; \mu)$ . The same parameter instance associated to the FOM, i.e.,  $(t; \mu)$ , enters block (C) which provides as output  $\mathbf{u}_n(t; \mu)$ , and the error between the low-dimensional vectors is accumulated. The minimal coordinates  $\mathbf{u}_n(t; \mu)$  are given as input to block (D) returning the approximated intrinsic coordinates  $\tilde{\mathbf{u}}_N(t; \mu)$ . Then, the reconstruction error is computed.

Computing the POD-DL-ROM approximation  $\tilde{\mathbf{u}}_h(t; \mu_{test})$  of  $\mathbf{u}_h(t; \mu_{test})$ , for any  $t \in (0, T)$  and  $\mu_{test} \in \mathcal{P}$ , corresponds to the testing stage of the DFNN and of the decoder function of the CAE, and does not require the evaluation of the encoder function. Finally, the POD-DL-ROM approximation of the FOM solution is recovered as

$$\tilde{\mathbf{u}}_h(t; \mu, \theta_{DF}, \theta_D) = \mathbf{V}_N \tilde{\mathbf{u}}_N(t; \mu, \theta_{DF}, \theta_D).$$

In the formerly proposed DL-ROM methodology [54,55], we employed a convolutional AE due to the fact that, thanks to the shared parameters and local connectivity properties [56], convolutional layers are better suited than dense layers to handle high-dimensional spatially correlated data. Regarding instead the description of the reduced dynamics, we introduced a DFNN since no particular data structure must be exploited in the learning task, which is indeed simpler than the nonlinear trial manifold learning.

Let us remark that the former construction of a POD-DL-ROM can be extended to the case of  $p > 1$  (either scalar or vector) field variables of interest in a straightforward way. In this case, provided a snapshot set  $\mathbf{S}_i$  and a corresponding basis  $\mathbf{V}_{N,i} \in \mathbb{R}^{N_{h,i} \times N}$ ,  $i = 1, \dots, p$ , for each of the variables  $\mathbf{u}_{h,1}, \dots, \mathbf{u}_{h,p}$ , the POD-DL-ROM approximation of the field variable  $\mathbf{u}_{h,i}(t; \mu) \in \mathbb{R}^{N_{h,i}}$  is given by

$$\tilde{\mathbf{u}}_{h,i}(t; \mu, \theta_{DF}, \theta_D) = \mathbf{V}_{N,i} \tilde{\mathbf{u}}_{N,i}(t; \mu, \theta_{DF}, \theta_D) \approx \mathbf{u}_{h,i}(t; \mu),$$

where a DFNN and a CAE are trained by considering simultaneously all the  $p$  field variables. Due to its *data-driven* nature, each variable can be approximated in an independent way—in other words, there are no physical constraints appearing in the loss function, thus making the  $p$  approximated field variables uncoupled, despite they might be originally coupled. For instance,  $p = 2$  field variables are considered if we aim at approximating both the velocity and the pressure fields in the case of fluid flows.

Another noteworthy aspect deals with the way snapshots are handled when considering convolutional layers in the NNs, in presence of either vector and/or coupled problems. Exploiting the analogy with red–green–black images in image processing, each snapshot computed for a variable of interest is reshaped in a square matrix of dimension  $(\sqrt{N}, \sqrt{N})$ , where  $N = 2^{(2m)}$  with  $m \in \mathbb{N}$  (if  $N \neq 2^{(2m)}$  the input is zero-padded), and stacked together forming a tensor with  $k \leq 3$  channels. The latter tensor is then provided as input to the POD-DL-ROM neural network architectures when dealing with vector and/or coupled problems; as a result, the output of the network, for each sample  $(t, \mu)$ , takes a form similar to (5), collecting the approximation of all the field variables,

$$\tilde{\mathbf{u}}_N(t; \mu, \theta_{DF}, \theta_D) = [\tilde{\mathbf{u}}_{N,1}(t; \mu, \theta_{DF}, \theta_D) \mid \dots \mid \tilde{\mathbf{u}}_{N,p}(t; \mu, \theta_{DF}, \theta_D)] \in \mathbb{R}^{N \times p}.$$

In the case of vector field variables, such as the fluid velocity or the structure displacement, different variable components are usually grouped together and treated in the same channel. In the case of  $p$  field variables, each one with at most  $d$  components, many of them can be grouped together and assigned to the same channel; in the case of  $p \leq 3$  scalar field variables, or when dealing with vector field variables involving at most 3 components—as in the case of the velocity and the pressure fields in dimension  $d = 2$ , or in the case of the velocity field in dimension  $d \leq 3$ —different components/field variables can be assigned to different channels.

We remark that considering vector and/or coupled problems does not entail main changes in the architecture of the POD-DL-ROM as well as in the total number of parameters of the neural network. Indeed, only the first layer of the encoder function and the last one of the decoder function are responsible for the handling of different channels of the input/output. This implies that training the neural network by providing data with  $k$  channels is remarkably less computationally expensive than training several independent POD-DL-ROMs, each of them responsible for a single component/field variable of the solution.

### 3. Results

In this section, we show several numerical results obtained with the POD-DL-ROM technique. In particular, we focus on the solution of three problems: (i) the unsteady Navier–Stokes equations for a two-dimensional flow around a cylinder, (ii) an FSI problem for a two-dimensional flow past an elastic beam attached to a fixed, rigid block, and (iii) the unsteady Navier–Stokes equations for blood flow in a cerebral aneurysm. To evaluate the performance of POD-DL-ROM, we rely on the loss function (8) and on:

- the error indicator  $\epsilon_{rel} \in \mathbb{R}$  given by

$$\epsilon_{rel} = \epsilon_{rel}(\mathbf{u}_h, \tilde{\mathbf{u}}_h) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left( \frac{\sqrt{\sum_{k=1}^{N_t} \|\mathbf{u}_h^k(\mu_{test,i}) - \tilde{\mathbf{u}}_h^k(\mu_{test,i})\|^2}}{\sqrt{\sum_{k=1}^{N_t} \|\mathbf{u}_h^k(\mu_{test,i})\|^2}} \right); \quad (9)$$

- the relative error  $\epsilon_k \in \mathbb{R}^{\sum_{i=1}^d N_{h,i}}$ , for  $k = 1, \dots, N_t$ , defined as

$$\epsilon_k = \epsilon_k(\mathbf{u}_h, \tilde{\mathbf{u}}_h) = \frac{|\mathbf{u}_h^k(\mu_{test}) - \tilde{\mathbf{u}}_h^k(\mu_{test})|}{\sqrt{\frac{1}{N_t} \sum_{k=1}^{N_t} \|\mathbf{u}_h^k(\mu_{test})\|^2}}. \quad (10)$$

Note that (9) is a scalar indicator, while (10) provides a spatially distributed error field.



The configuration of the POD DL-ROM neural network used in our test cases is the one given below. We choose a 12-layer DFNN equipped with 50 neurons per hidden layer and  $n$  neurons in the output layer, where  $n$  represents the dimension of the (nonlinear) reduced trial manifold. The architectures of the encoder and decoder functions are instead reported in Tables 1 and 2. No activation function is applied at the last convolutional layer of the decoder neural network, as usually done in AEs.

**Table 1.** Attributes of convolutional and dense layers in the encoder  $\mathbf{f}_n^E$ .

Layer	Input Dimension	Output Dimension	Kernel Size	#of Filters	Stride	Padding
1	$[N, N, d]$	$[N, N, 8]$	$[5, 5]$	8	1	SAME
2	$[N, N, 8]$	$[N/2, N/2, 16]$	$[5, 5]$	16	2	SAME
3	$[N/2, N/2, 16]$	$[N/4, N/4, 32]$	$[5, 5]$	32	2	SAME
4	$[N/4, N/4, 32]$	$[N/8, N/8, 64]$	$[5, 5]$	64	2	SAME
5	$N$	64				
6	64	$n$				

**Table 2.** Attributes of dense and transposed convolutional layers in the decoder  $\mathbf{f}_N^D$ .

Layer	Input Dimension	Output Dimension	Kernel Size	#of Filters	Stride	Padding
1	$n$	256				
2	256	$N_h$				
3	$[N/8, N/8, 64]$	$[N/4, N/4, 64]$	$[5, 5]$	64	2	SAME
4	$[N/4, N/4, 64]$	$[N/2, N/2, 32]$	$[5, 5]$	32	2	SAME
5	$[N/2, N/2, 32]$	$[N, N, 16]$	$[5, 5]$	16	2	SAME
6	$[N, N, 16]$	$[N, N, d]$	$[5, 5]$	$d$	1	SAME

To solve the optimization problem (7) and (8), we use the ADAM algorithm [57], which is a stochastic gradient descent method computing an adaptive approximation of the first and second momentum of the gradients of the loss function. In particular, it computes exponentially weighted moving averages of the gradients and of the squared gradients. We set the starting learning rate to  $\eta = 10^{-4}$ , and perform cross-validation in order to tune the hyperparameters of the POD-DL-ROM by splitting the data in training and validation sets with a proportion 8:2. Moreover, we implement an early-stopping regularization technique to reduce overfitting [56], stopping the training if the loss does not decrease over a certain amount of epochs. As nonlinear activation function, we employ the ELU function [58]. The parameters, weights, and biases are initialized through He uniform initialization [59]. The rPOD dimension  $N$  is selected, in all test cases, in order to fulfill the condition  $\epsilon_{rel}(\mathbf{u}_h, \mathbf{V}_N \mathbf{V}_N^T \mathbf{u}_h) \approx 10^{-3}$ . The interested reader can refer to [42] for a detailed version of the algorithms used for the training/testing phases. These latter have been carried out on either a GTX 1070 8 GB or a Tesla V100 32 GB GPU by means of the Tensorflow DL framework [60] for the cases described in the following subsections. The Matlab library redbKIT [2,61] has been employed to carry out all the FOM simulations.

### 3.1. Test Case 1: Flow around a Cylinder

In this first test case, we deal with the unsteady Navier–Stokes equations for incompressible flows in primitive variables (fluid velocity  $\mathbf{v}$  and pressure  $p$ ). We consider the flow around a cylinder test case, a well-known benchmark problem for the evaluation of numerical algorithms for incompressible Navier–Stokes equations in the laminar case [62]. The problem reads as follows:

$$\begin{cases} \rho \frac{\partial \mathbf{v}}{\partial t} + \rho \mathbf{v} \cdot \nabla \mathbf{v} - \nabla \cdot \boldsymbol{\sigma}(\mathbf{v}, p) = \mathbf{0} & (\mathbf{x}, t) \in \Omega^F \times (0, T), \\ \nabla \cdot \mathbf{v} = 0 & (\mathbf{x}, t) \in \Omega^F \times (0, T), \\ \mathbf{v} = \mathbf{0} & (\mathbf{x}, t) \in \Gamma_{D_1} \times (0, T), \\ \mathbf{v} = \mathbf{h} & (\mathbf{x}, t) \in \Gamma_{D_2} \times (0, T), \\ \boldsymbol{\sigma}(\mathbf{v}, p) \mathbf{n} = \mathbf{0} & (\mathbf{x}, t) \in \Gamma_N \times (0, T), \\ \mathbf{v}(0) = \mathbf{0} & \mathbf{x} \in \Omega^F, t = 0. \end{cases} \quad (11)$$

The domain consists of a two-dimensional pipe with a circular obstacle, i.e.,  $\Omega^F = (0, 2.2) \times (0, 0.41) \setminus \bar{B}_{0.05}(0.2, 0.2)$ —here  $B_r(\mathbf{x}_c)$  denotes a ball of radius  $r > 0$  centered at  $\mathbf{x}_c$ , see Figure 2 for a sketch of the geometry. The boundary is given by  $\partial\Omega^F = \Gamma_{D_1} \cup \Gamma_{D_2} \cup \Gamma_N$ , where  $\Gamma_{D_1} = \{x_1 \in [0, 2.2], x_2 = 0\} \cup \{x_1 \in [0, 2.2], x_2 = 0.41\} \cup \partial B_{0.05}((0.2, 0.2))$ ,  $\Gamma_{D_2} = \{x_1 = 0, x_2 \in [0, 0.41]\}$ , and  $\Gamma_N = \{x_1 = 2.2, x_2 \in [0, 0.41]\}$ , while  $\mathbf{n}$  denotes the (outward directed) normal unit vector to  $\partial\Omega^F$ . We denote by  $\rho$  the fluid density, and by  $\boldsymbol{\sigma}$  the stress tensor,

$$\boldsymbol{\sigma}(\mathbf{v}, p) = -p\mathbf{I} + 2\nu\boldsymbol{\epsilon}(\mathbf{v}); \quad (12)$$

here,  $\nu$  denotes the dynamic viscosity of the fluid, while  $\boldsymbol{\epsilon}(\mathbf{v})$  is the strain tensor,

$$\boldsymbol{\epsilon}(\mathbf{v}) = \frac{1}{2}(\nabla \mathbf{v} + \nabla \mathbf{v}^T).$$

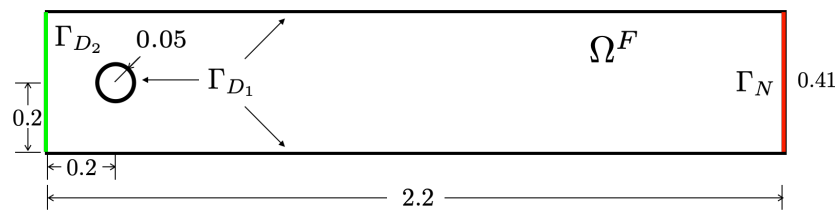


Figure 2. Test case 1: geometrical configuration, domain and boundaries [m].

Here, we take  $\rho = 1 \text{ kg/m}^3$  as fluid density, and assign no-slip boundary conditions on  $\Gamma_1$ , a parabolic inflow profile

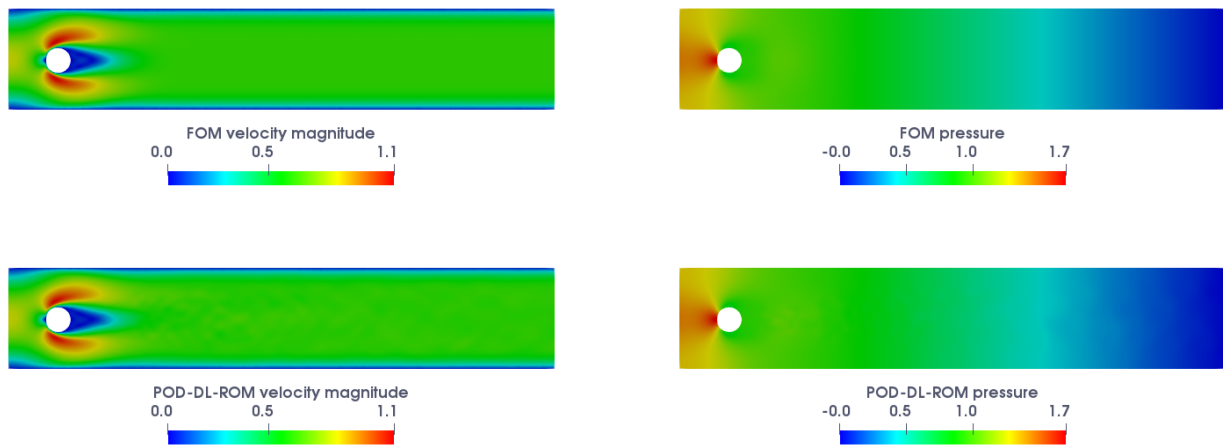
$$\mathbf{h}(\mathbf{x}, t; \mu) = \left( \frac{4U(t, \mu)x_2(0.41 - x_2)}{0.41^2}, 0 \right), \quad \text{where } U(t; \mu) = \mu \sin(\pi t/8), \quad (13)$$

on the inlet  $\Gamma_{D_2}$ , and zero-stress Neumann conditions on the outlet  $\Gamma_N$ . We consider as parameter ( $n_\mu = 1$ )  $\mu \in \mathcal{P} = [1, 2] \text{ m/s}$ , which reflects the Reynolds number varying in the range [66, 133]. Equations (11) have been discretized in space by means of linear–quadratic ( $\mathbb{P}_2 - \mathbb{P}_1$ ), inf–sup stable, finite elements, and in time through a backward differentiation formula (BDF) of order 2 with semi-implicit treatment of the convective term (see, e.g., [63]) over the time interval  $(0, T)$ , with  $T = 8 \text{ s}$ , and a time step  $\Delta t = 2 \times 10^{-3} \text{ s}$ . This strategy allows us to mitigate the computational cost associated with the use of a fully implicit BDF scheme by linearizing the nonlinear convective terms; this latter task is realized by extrapolating the convective velocity through an extrapolation formula of the same order of the BDF introduced.

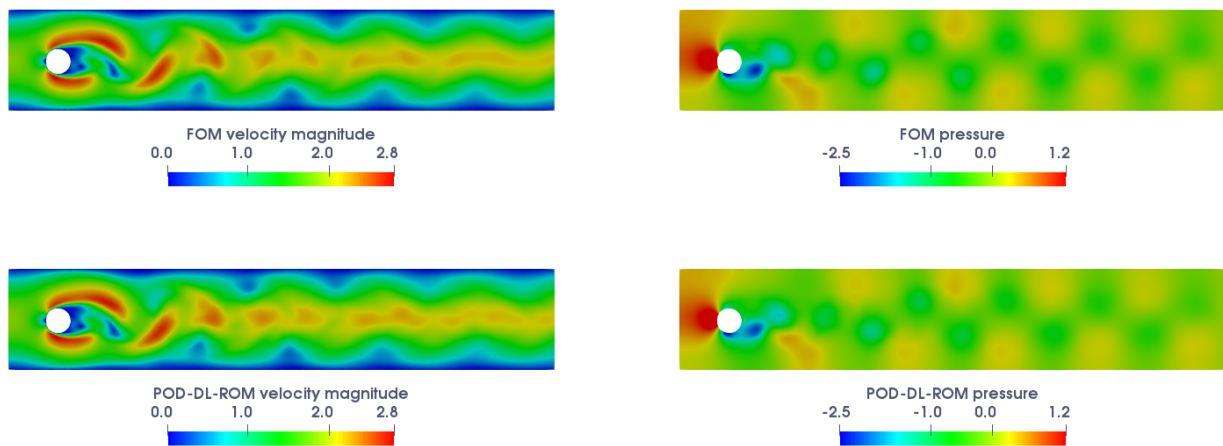
We already analyzed this test case in [42], where we were interested in approximating only the velocity field. Here, we aim at assessing the performance of POD-DL-ROM neural network in approximating both the velocity and the pressure fields. In particular, we provide, to the network, data under the form of a tensor with 3 channels—that is, we set the dimension equal to  $k = 3$ . The FOM dimension is equal to  $N_h = [32, 446, 32, 446, 8239]$  (for the two velocity components and the pressure, respectively) and we select  $N = 256$  as dimension of the POD basis for each component of the solution. We choose  $n = 5$  as dimension of the nonlinear trial manifold  $\tilde{\mathcal{S}}_n$ . We uniformly sample  $N_t = 400$  time

instances and consider  $N_{train} = 21$  and  $N_{test} = 20$  training- and testing-parameter instances uniformly distributed over  $\mathcal{P}$ . The total number of parameters (i.e., weights and biases) of the neural network is equal to 295,337.

In Figures 3 and 4, we compare the FOM and POD-DL-ROM pressure and velocity fields, these latter obtained with  $n = 5$ , together with the relative error  $\epsilon_k$  in Figure 5, for the testing-parameter instance  $\mu_{test} = 1.975$  m/s ( $Re = 131$ ) at  $t = 1.062$  s and  $t = 4.842$  s. We highlight the ability of the POD-DL-ROM approximation to accurately capture the variability of the solution: indeed, in the case  $t = 1.062$  s (Figure 3), we do not assist any vortex shedding; this latter is instead present in the case  $t = 4.842$  s (Figure 4), and is correctly reproduced.



**Figure 3.** Test case 1: FOM and POD-DL-ROM solutions for the testing-parameter instance  $\mu_{test} = 1.975$  m/s at  $t = 1.062$  s, with  $N = 256$  and  $n = 5$ . **Left:** velocity field magnitude; **right:** pressure field.



**Figure 4.** Test case 1: FOM and POD-DL-ROM solutions for the testing-parameter instance  $\mu_{test} = 1.975$  m/s at  $t = 4.842$  s, with  $N = 256$  and  $n = 5$ . **Left:** velocity field magnitude; **right:** pressure field.

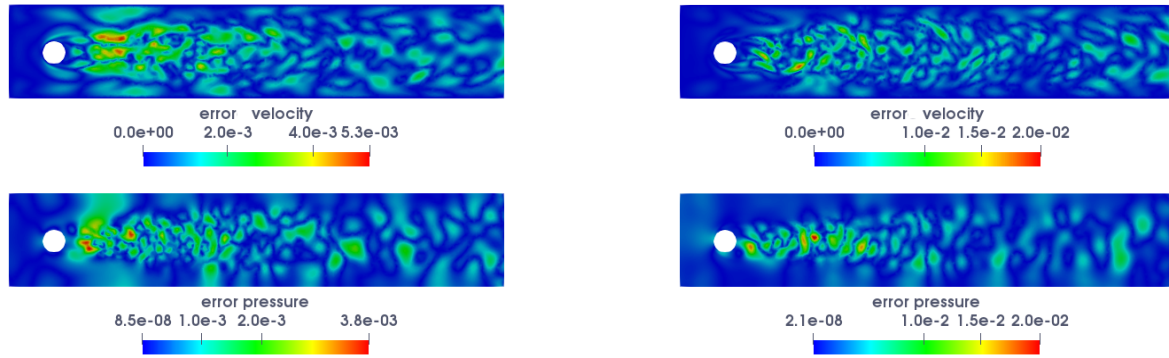
To assess the ability of the POD-DL-ROM to provide accurate evaluations of output quantities of interest, we evaluate the drag and lift coefficients related to the drag and lift forces around the circular obstacle; these are defined, in our case, as

$$F_D = \int_{\partial B_r} \left( v \frac{\partial u_2}{\partial \eta} \eta_2 - p \eta_1 \right) d\sigma, \quad \text{and} \quad F_L = \int_{\partial B_r} \left( v \frac{\partial u_1}{\partial \eta} \eta_1 - p \eta_2 \right) d\sigma \quad (14)$$

where  $\boldsymbol{\eta} = (\eta_1, \eta_2)^T$  denotes the (outward directed) normal unit vector to  $\partial\Omega$ . From (14), the dimensionless drag and lift coefficient can be obtained as

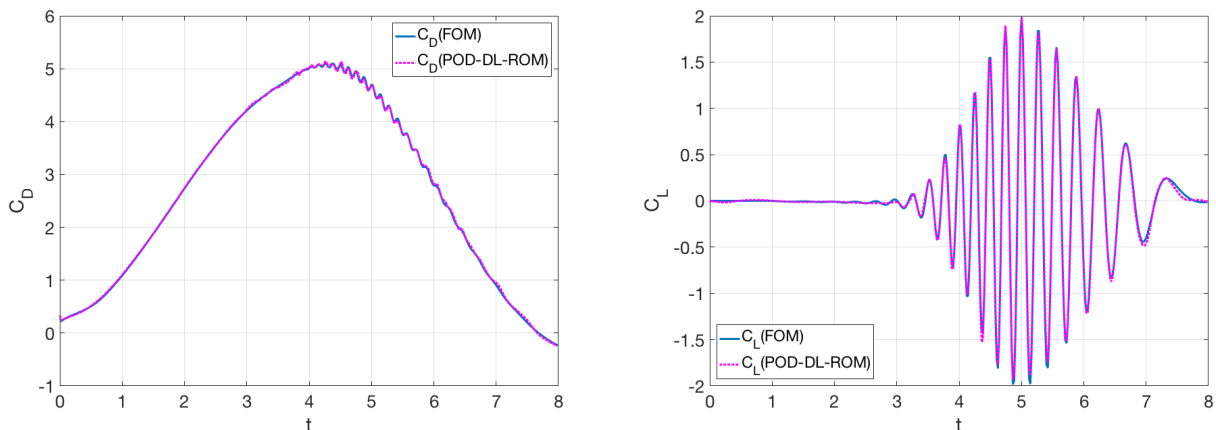
$$C_D = \frac{2}{U_{mean}^2 L} F_D, \quad \text{and} \quad C_L = \frac{2}{U_{mean}^2 L} F_L,$$

where  $U_{mean}$  is the parabolic input profile mean velocity.



**Figure 5.** Test case 1: Relative errors  $\epsilon_k$  for the testing-parameter instance  $\mu_{test} = 1.975$  m/s, with  $N = 256$  and  $n = 5$ . Relative errors at  $t = 1.062$  s (left) and  $t = 4.842$  s (right) for velocity (top) and pressure (bottom).

The drag and lift coefficients computed over time by the FOM and the POD-DL-ROM, for the testing-parameter instances  $\mu_{test} = 1.975$  m/s, are reported in Figure 6. The POD-DL-ROM technique is also able to accurately capture the evolution of  $C_D$  and  $C_L$ , related to the prescribed  $\mu$ -dependent input profile in (13), in both cases. Indeed, we remark that the oscillatory behavior observed over time, due to vortex shedding, is fully reconstructed by the POD-DL-ROM and is consistent with the results obtained at low  $Re$  numbers reported in literature (see, e.g., [64]).



**Figure 6.** Test case 1: FOM and POD-DL-ROM drag (left) and lift (right) coefficients for the testing-parameter instance  $\mu_{test} = 1.975$ .

Furthermore, the testing computational time, i.e., the time needed to compute  $N_t$  time instances for an unseen testing-parameter instance, of the POD-DL-ROM on a GTX 1070 8 GB GPU is given by 0.2 s, thus implying a speed-up equal to  $1.25 \times 10^5$  with respect to the time needed for the solution of the FOM. For test case 1, the FOM simulations have been performed on 20 cores of 1.7 TB node (192 Intel® Xeon Platinum® 8160 2.1 GHz cores) of the HPC cluster available at MOX, Politecnico di Milano.

We also highlight that the application of POD-DL-ROMs to fluid problems showing a dominant transport behavior, such as reacting fluid flows, would require only slight modifications. For instance, a larger rPOD dimension, a more careful selection of the

training-parameter instances, and an increase of the number of parameters of the neural network might be required, thus yielding longer training computational times, though without dramatically impacting the overall accuracy and efficiency of the methodology. In this respect, preliminary results obtained with POD-DL-ROMs in the case of a dominant transport problem can be found, e.g., in [42].

### 3.2. Test Case 2: Fluid–Structure Interaction

We now focus on the case of a two-dimensional flow past an elastic beam attached to a fixed, rigid block [65–67] (see Figure 7 for a sketch of the geometry). The FSI model that we consider consists of a two-field problem, where the incompressible Navier–Stokes equations written in the arbitrary Lagrangian–Eulerian (ALE) form for the fluid are coupled with the nonlinear elastodynamics equation modeling the solid deformation [53]. Because of the ALE approach we employ, a third non-physical geometry (or mesh motion) problem is introduced, which accounts for the fluid domain deformation and in turn defines the so-called ALE map.

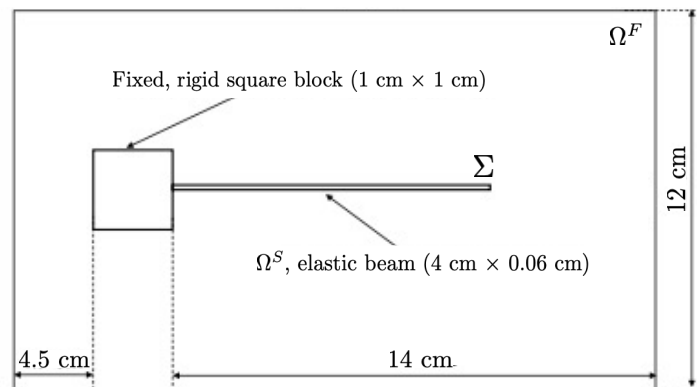


Figure 7. Test case 2: Geometrical configuration and domains.

Let  $\Omega^F$  and  $\Omega^S$  be the domains occupied by the fluid and the solid, respectively, in their reference configuration. We denote, by  $\Sigma = \partial\Omega^F \cap \partial\Omega^S$ , the fluid–structure interface on the reference configuration. At any time  $t$ , the domain occupied by the fluid  $\Omega^F(t)$  can be retrieved from  $\Omega^F$  by the ALE mapping

$$\mathcal{A}_t : \Omega^F \rightarrow \Omega^F(t), \quad \mathbf{X} \mapsto \mathcal{A}_t(\mathbf{X}) = \mathbf{X} + \mathbf{d}^G(\mathbf{X}),$$

where  $\mathbf{d}^G(\mathbf{X})$  represents the displacement of the fluid domain. The coupled FSI problem thus consists of the following set of equations.

- Navier–Stokes in ALE form governing the fluid problem:

$$\begin{cases} \rho^F \frac{\partial \mathbf{v}^F}{\partial t} \Big|_{\mathbf{X}} + (\mathbf{v}^F - \mathbf{w}^G) \cdot \nabla \mathbf{v}^F - \nabla \sigma^F(\mathbf{v}^F, p^F) = 0 & \text{in } \Omega^F(t), \\ \nabla \cdot \mathbf{v}^F = 0 & \text{in } \Omega^F(t); \end{cases} \quad (15)$$

- nonlinear elastodynamics equation governing the solid dynamics:

$$\rho^S \frac{\partial \mathbf{d}^S}{\partial t^2} - \nabla \cdot \mathbf{P}(\mathbf{d}^S) = 0 \quad \text{in } \Omega^S; \quad (16)$$

- coupling at the FS interface  $\Sigma$ :

$$\begin{cases} \mathbf{v}^F = \frac{\partial \mathbf{d}^S}{\partial t}, \\ \sigma^F(\mathbf{v}^F, p^F) \mathbf{n}^F + \sigma^S(\mathbf{d}^S) \mathbf{n}^S = \mathbf{0}; \end{cases} \quad (17)$$

- linear elasticity equations modeling the mesh motion problem:

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma}^G(\mathbf{d}^G) = 0 & \text{in } \Omega^F, \\ \mathbf{d}^G = \mathbf{d}^S & \text{on } \Sigma, \end{cases} \quad (18)$$

where  $\boldsymbol{\sigma}^F(\mathbf{v}^F, p^F) = -p^F \mathbf{I} + 2\mu^F \boldsymbol{\epsilon}(\mathbf{v}^F)$  is the fluid Cauchy stress tensor,  $\boldsymbol{\sigma}^S(\mathbf{d}^S) = J^{-1} \mathbf{P} \mathbf{F}^T$  is the solid Cauchy stress tensor,  $\mathbf{F} = \mathbf{I} + \nabla \mathbf{d}^S$  is the deformation tensor, and  $\mathbf{w}^G = \frac{\partial \mathbf{d}^G}{\partial t} \Big|_X$  is the fluid mesh velocity. See, e.g., [53] for further details.

Both fluid and solid equations are complemented by appropriate initial and boundary conditions. In particular, the lateral boundaries are assigned zero normal velocity and zero tangential stress. Zero-traction boundary condition is applied at the outflow. The flow is driven by a uniform inflow velocity of 51.3 cm/s. Zero-initial conditions are assigned both for the fluid and the solid equations. The fluid density and viscosity are  $1.18 \times 10^{-3}$  g/cm<sup>3</sup> and  $1.82 \times 10^{-4}$  g/(cm·s) respectively, resulting in a Reynolds number of 100 based on the edge length of the block. The beam is modeled as a solid made of the St. Venant–Kirchhoff material and the density of the beam is 0.1 g/cm<sup>3</sup>.

The field equations are discretized in space and time using:

- matching spatial discretizations between fluid and structure at the interface;
- for the fluid subproblem, SUPG stabilized linear finite elements ( $\mathbb{P}_1 - \mathbb{P}_1$ ) and a BDF of order 2 in time;
- for the structural subproblem, the same finite element space as for the fluid velocity and the Newmark scheme in time;
- for the fluid displacement, the same finite element space as for the fluid velocity.

The resulting nonlinear problem is solved through a monolithic geometry-convective explicit (GCE) scheme, obtained by linearizing the fluid convective term (with a BDF extrapolation) and treating the geometry problem explicitly [68,69]. Here,  $n_\mu = 2$  parameters are considered, the Young modulus  $\mu_1$  and the Poisson ratio  $\mu_2$ , varying in the parameter space  $\mathcal{P} = 10^6 \cdot [2.3, 2.7] \text{ g}/(\text{cm} \cdot \text{s}^2) \times [0.3, 0.4]$ . We build a FOM considering  $N_h = [16, 452, 8226, 1974]$  DOFs for the velocity, pressure, and displacement fields, respectively, and a time step  $\Delta t = 1.65 \times 10^{-3}$  over  $(0, T)$  with  $T = 3$  s.

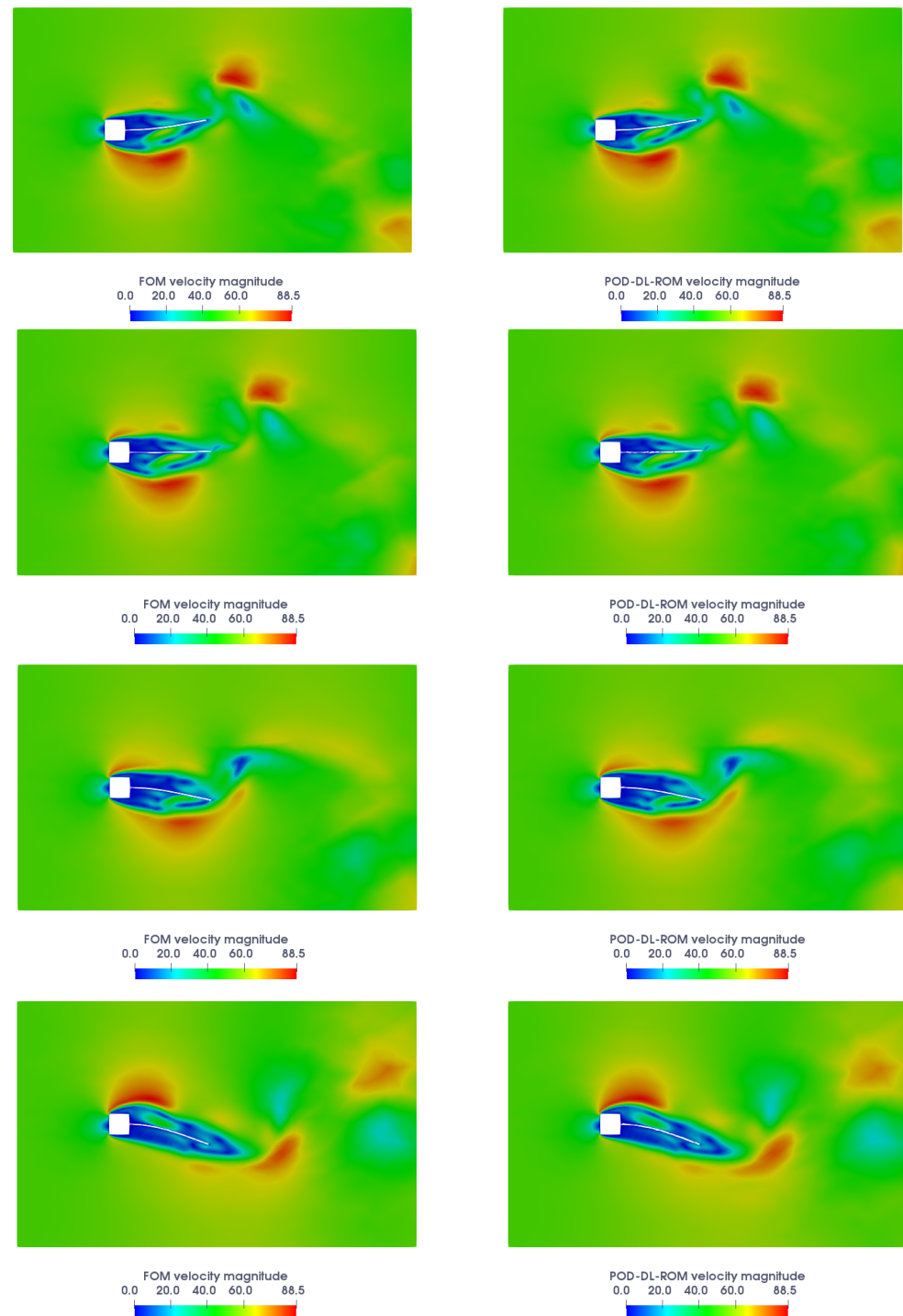
Regarding the construction of the proposed POD-DL-ROM, for the training of the neural networks, we uniformly sample  $N_t = 606$  time instances and  $N_{train} = 5 \times 3 = 15$  training-parameter instances, uniformly distributed in each parametric direction. At testing phase,  $N_{test} = 4 \times 2 = 8$  testing-parameter instances, different from the training ones, have been considered. The maximum number of epochs is set equal to  $N_{epochs} = 20,000$ , the batch size is  $N_b = 40$  and, regarding the early-stopping criterion, we stop the training if the loss function does not decrease within 500 epochs.

We are interested in reconstructing the velocity and the displacement fields, so we set the number of channels to  $k = 2$  and we recall the ability of the POD-DL-ROM neural network to handle different FOM dimensions,  $N_{h,i}$ , for  $i = 1, \dots, p$ ; that is, only the POD dimension must be equal among the different fields considered. Moreover, we set  $N = 256$  as the dimension of the POD basis, and  $n = 5$  as the dimension of the reduced nonlinear trial manifold.

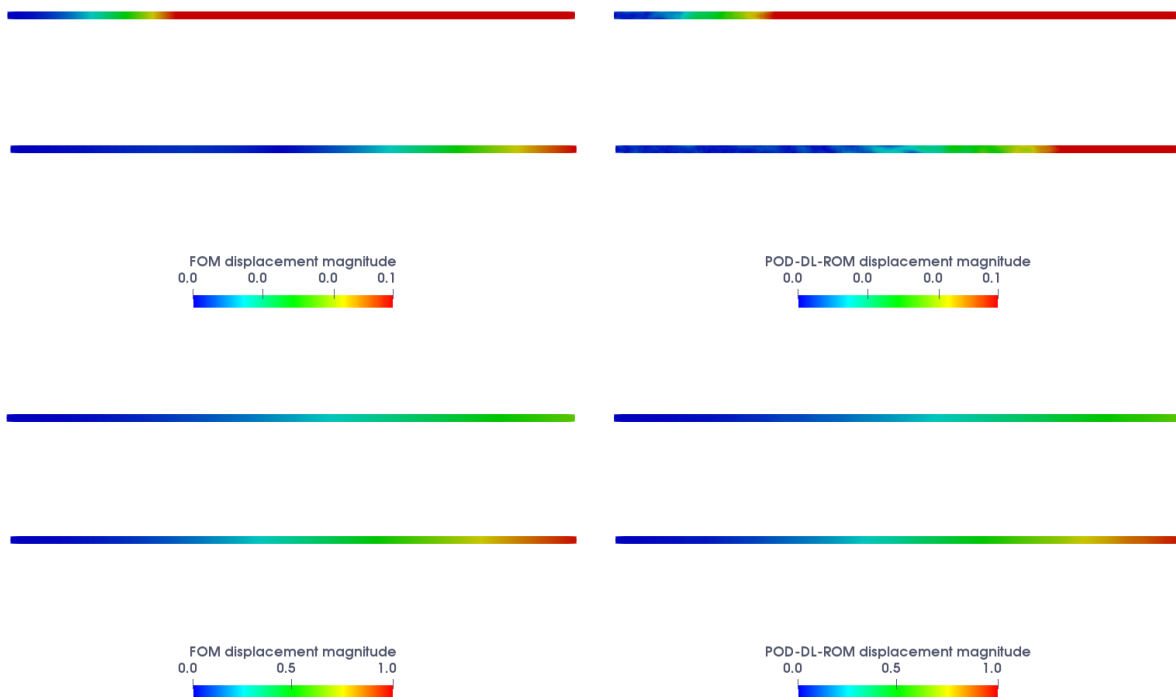
The training and testing phases of the POD-DL-ROM neural network have been performed on a Tesla V100 32 GB GPU. The total number of parameters of the POD-DL-ROM neural network is equal to 294,185.

In Figure 8, we report the FOM and the POD-DL-ROM velocity magnitudes, the latter with  $N = 256$  and  $n = 5$ , for two testing-parameter instances— $\boldsymbol{\mu}_{test} = [2.3 \times 10^6 \text{ g}/(\text{cm} \cdot \text{s}^2), 0.325]$  and  $\boldsymbol{\mu}_{test} = [2.7 \times 10^6 \text{ g}/(\text{cm} \cdot \text{s}^2), 0.375]$ —at  $t = 2.3084$  s and  $t = 2.64$  s. In particular, we can observe that vortices, which are being shed from the square block, are impinging on the bar, eventually forcing it to have an oscillating motion, during which it undergoes large deformations. We point out that the dependence of the displacement field on the parameters reflects on the velocity field by producing a strong variability over the

parameter space, which is accurately captured by the POD-DL-ROM solutions. Indeed, at the same time instants, depending on the value of the parameters  $\mu$ , the solution exhibits remarkably different patterns. The FOM and POD-DL-ROM displacement magnitudes for the testing-parameter instances  $\mu_{test} = [2.3 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.325]$  and  $\mu_{test} = [2.7 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.375]$  at  $t = 2.3084 \text{ s}$  and  $t = 2.64 \text{ s}$  over the domain, are shown in Figure 9.

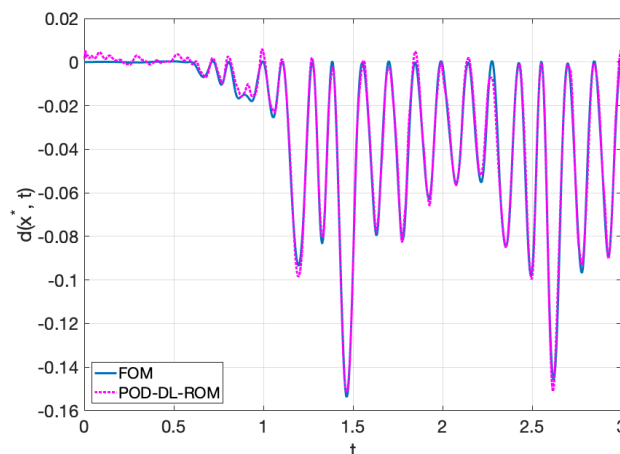


**Figure 8.** Test case 2: FOM (left) and POD-DL-ROM (right) fluid velocity magnitudes for the testing-parameter instances  $\mu_{test} = [2.3 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.325]$  (rows 1–2) and  $\mu_{test} = [2.7 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.375]$  (rows 3–4), at  $t = 2.3084 \text{ s}$  (top) and  $t = 2.64 \text{ s}$  (bottom).



**Figure 9.** Test case 2: FOM (left) and POD-DL-ROM (right) structure displacement magnitudes for the testing-parameter instances  $\mu_{test} = [2.3 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.325]$  at  $t = 2.3084 \text{ s}$  and  $t = 2.64 \text{ s}$  (rows 1–2) and  $\mu_{test} = [2.7 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.375]$  at  $t = 2.3084 \text{ s}$  and  $t = 2.64 \text{ s}$  (rows 3–4).

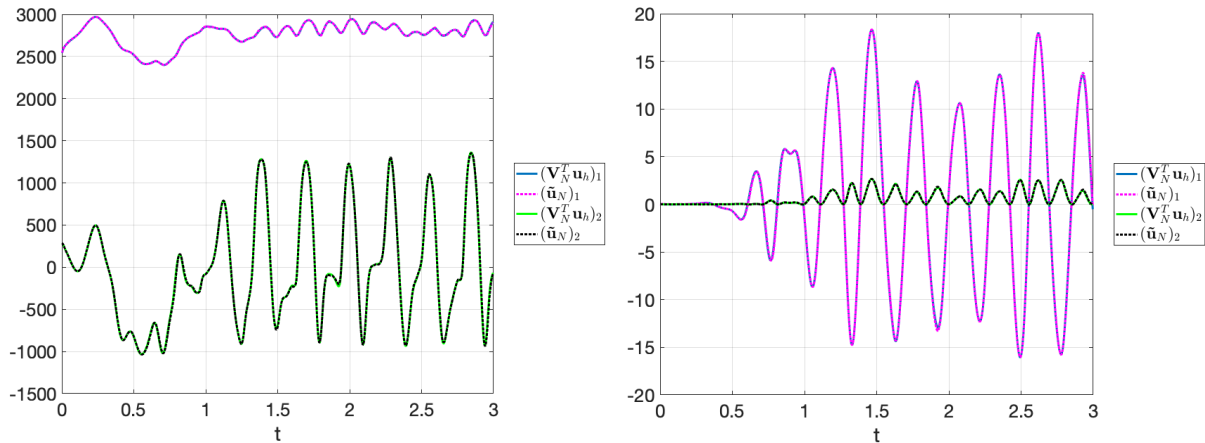
We point out that the disagreement between the FOM and POD-DL-ROM displacement magnitudes is larger for the testing-parameter instance  $\mu_{test} = [2.3 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.325]$  at  $t = 2.64 \text{ s}$  with respect to the other cases reported in Figure 9. This is related to the fact that the POD-DL-ROM neural network is biased towards larger values of displacement over the parameter space and generates higher errors when the bar displacement is very small. The comparison between the FOM and POD-DL-ROM displacement magnitudes, for the testing-parameter instance  $\mu_{test} = [2.7 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.375]$  at  $\mathbf{x}^* = (5.50, 6.07) \text{ cm}$  over time, is reported in Figure 10, from which it is clearly visible that the POD-DL-ROM is also able to capture the main features of the oscillating elastic beam dynamics.



**Figure 10.** Test case 2: FOM and POD-DL-ROM displacement at  $\mathbf{x}^* = (5.50, 6.07) \text{ cm}$  for the testing-parameter instance  $\mu_{test} = [2.7 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.375]$ .



The full accordance between the (first) components of the intrinsic coordinates vector  $\mathbf{V}_N^T \mathbf{u}_h(t; \boldsymbol{\mu}_{test})$  and their POD-DL-ROM approximation  $\tilde{\mathbf{u}}_N(t^k; \boldsymbol{\mu}_{test}, \boldsymbol{\theta}_{DF}, \boldsymbol{\theta}_D)$ , both for the velocity and the displacement fields, for the testing parameter instance  $\boldsymbol{\mu}_{test} = [2.7 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.375]$ , is shown in Figure 11. We remark that, as expected, the first components are the ones retaining most of the energy of the system, thus being the ones showing higher magnitude.



**Figure 11.** Test case 2: Comparison between the intrinsic coordinates  $\mathbf{V}_N^T \mathbf{u}_h$  components and the POD-DL-ROM approximation  $\tilde{\mathbf{u}}_N$  for the velocity (left) and the displacement (right) fields for the testing-parameter instance  $\boldsymbol{\mu}_{test} = [2.7 \times 10^6 \text{ g}/(\text{cm}\cdot\text{s}^2), 0.375]$ .

Finally, in Table 3, we report the POD-DL-ROM GPU total (training and validation) time, the testing time, i.e., the time needed to compute  $N_t$  time instances for a testing-parameter instance, and the time required to compute one time instance at testing time. Indeed, we recall that the DL-ROM solution can be queried at a given time without requiring any solution of a dynamical system to recover the former time instances. We also show the speed-up gained by POD-DL-ROM with respect to the computational time needed to solve the FOM. For test case 2, the FOM simulations have been carried out on a MacBook Pro Intel Core i7 6-core with 16 GB RAM CPU.

**Table 3.** Test case 2: POD-DL-ROM GPU computational times.

Total Time [h]	Testing Time [s]	1-Sample Testing Time [s]	Speed-Up
7	$4 \times 10^{-2}$	$5 \times 10^{-3}$	$1.77 \times 10^5$ ( $1.41 \times 10^6$ )

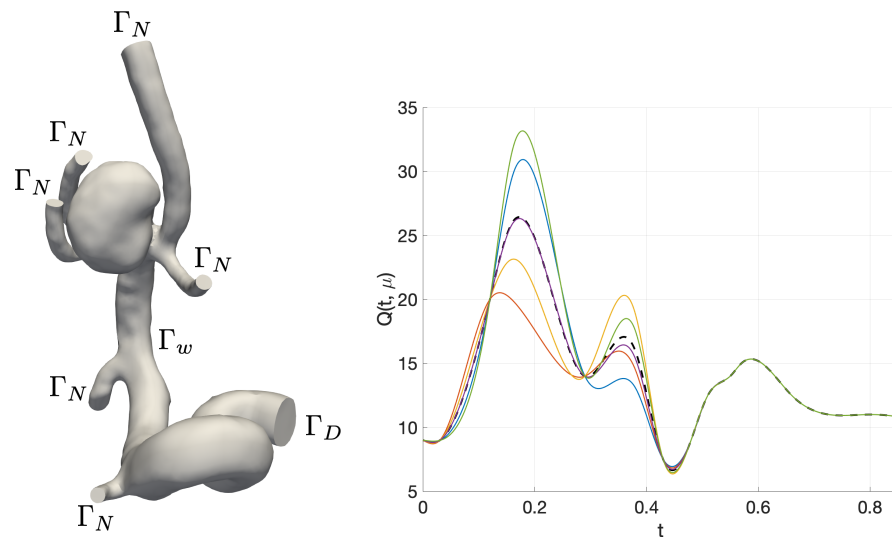
### 3.3. Test Case 3: Blood Flow in a Cerebral Aneurysm

In this last test case, we consider the fast simulation of blood flows in a cerebral (or intracranial) aneurysm; that is, a localized dilation or ballooning of a blood vessel in the brain, often occurring in the circle of Willis, the vessel network at the base of the brain. Blood velocity and pressure, wall shear stress (WSS), blood flow impingement, and particle residence time all play a key role in the growth and rupture of cerebral aneurysms—see e.g., [70–72]—which might ultimately yield potentially severe brain damage. For these reasons, computational hemodynamics inside aneurysm models can provide output quantities of interest useful for planning their surgical treatment.

We consider the artery aneurysm shown in Figure 12 (left), whose geometry has been supplied by the Aneurisk project [73–75]. We consider blood as a Newtonian fluid, with constant viscosity, and a rigid arterial wall, so that blood flow dynamics can be described by the following Navier–Stokes equations:

$$\begin{cases} \rho \frac{\partial \mathbf{v}}{\partial t} + \rho \mathbf{v} \cdot \nabla \mathbf{v} - \nabla \cdot \boldsymbol{\sigma}(\mathbf{v}, p) = \mathbf{0} & (\mathbf{x}, t) \in \Omega^F \times (0, T), \\ \nabla \cdot \mathbf{v} = 0 & (\mathbf{x}, t) \in \Omega^F \times (0, T), \\ \mathbf{v} = \mathbf{0} & (\mathbf{x}, t) \in \Gamma_w \times (0, T), \\ \mathbf{v} = k \mathbf{v}_{in} Q(t) & (\mathbf{x}, t) \in \Gamma_D \times (0, T), \\ \boldsymbol{\sigma}(\mathbf{v}, p) \mathbf{n} = \mathbf{0} & (\mathbf{x}, t) \in \Gamma_N \times (0, T), \\ \mathbf{v}(0) = \mathbf{0} & \mathbf{x} \in \Omega, t = 0, \end{cases} \quad (19)$$

where the stress tensor is defined as in (12). On the arterial wall  $\Gamma_w$ , a no-slip condition on the fluid velocity is imposed, flow resistance at the outlet boundaries  $\Gamma_N$  is neglected, while a parabolic profile  $\mathbf{v}_{in}$  is specified at the lumen inlet, where the parameterization of the inlet flow rate profile  $Q(t; \boldsymbol{\mu})$  has been obtained by interpolating with radial basis functions a base profile  $Q(t)$  taken from [76], and then treating some of the interpolated values as parameters (see Figure 12, right), see [77] for further details.



**Figure 12.** Test case 3. Aneurysm geometry (left) and inlet flow rate  $Q(t; \boldsymbol{\mu})$  during the heart cycle for different parameter values (right); the black dashed curve corresponds to the base profile  $Q(t)$ .

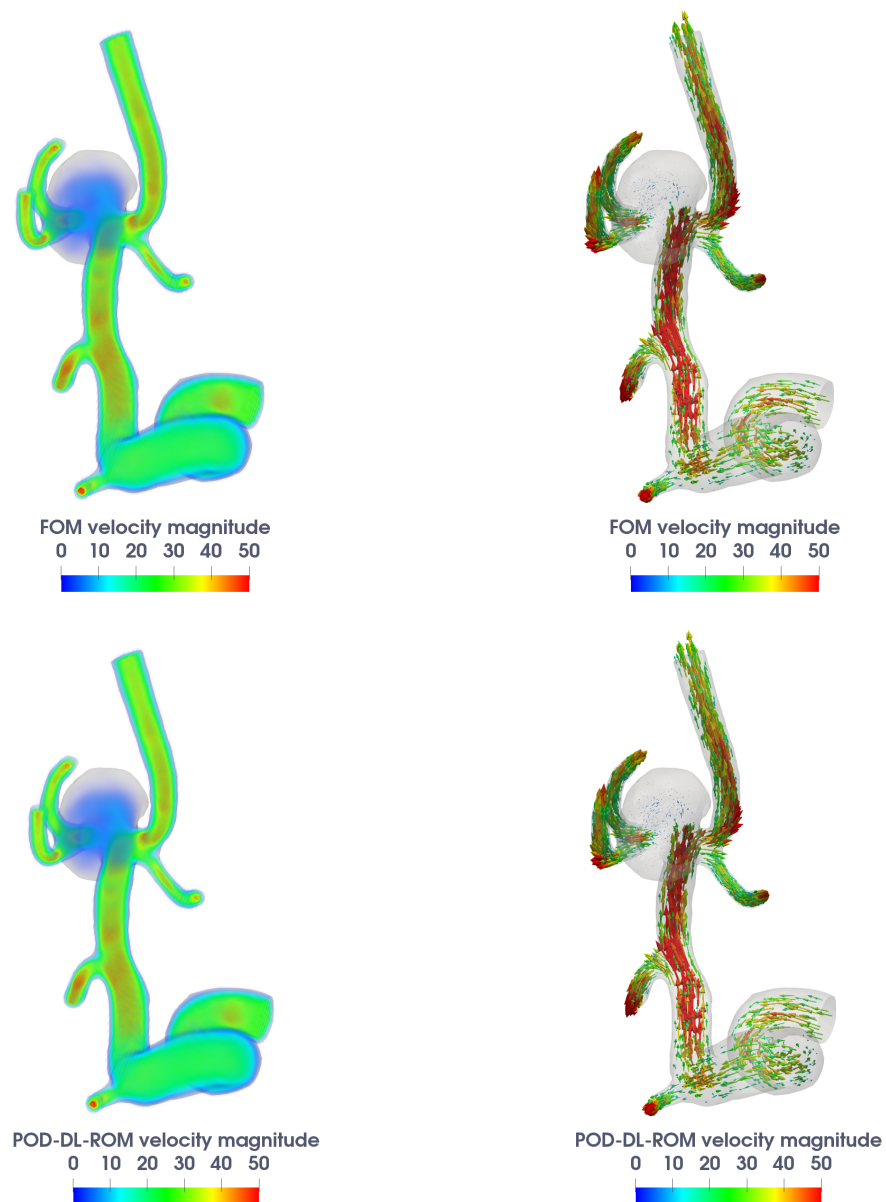
In particular, we consider  $n_\mu = 2$  parameters  $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^2$  such that the flow rate at  $t = 0.16$  s and  $t = 0.38$  s admits variations up to 15% of the reference value. A comparison between some flow rate profiles corresponding to different parameter values is shown in Figure 12, right. The scaling factor  $k$  in (19) is such that

$$\int_{\Gamma_D} k \mathbf{v}_{in} \cdot \mathbf{n} d\sigma = 1.$$

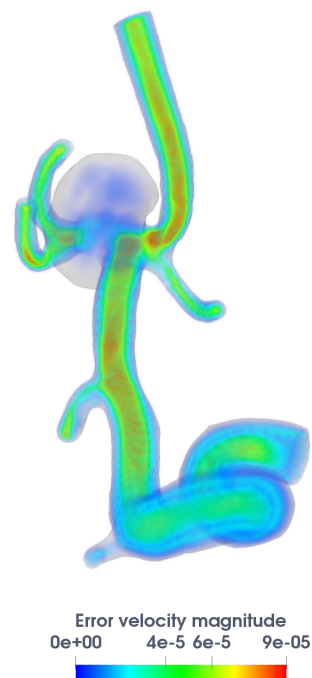
Blood dynamic viscosity  $\nu = 0.035$  P and density  $\rho = 1$  g/cm<sup>3</sup> are set.

Concerning the FOM discretization, we employ a SUPG-BDF semi-implicit time scheme of order 2 with linear finite elements for both velocity and pressure variables. We employ a time step  $\Delta t = 10^{-3}$  over the interval  $(0, T)$  with  $T = 0.85$  s. We simulate the blood flow starting from an initial condition obtained by solving the steady Stokes problem. We are interested in reconstructing the blood velocity field, so we set the FOM dimensions to  $N_h = [41,985, 41,985, 41,985]$ , and  $k = 3$ . The POD dimension is equal to  $N = 64$  for each component of the solution, and the dimension of the reduced nonlinear trial manifold is chosen to be equal to  $n = 5$ , very close to the intrinsic dimension of the problem  $n_\mu + 1 = 3$ . We consider  $N_t = 850$  time instances,  $N_{train} = 6$ , and  $N_{test} = 3$  training- and testing-parameter instances, sampled over  $\mathcal{P}$  by means of the latin hypercube sampling strategy. The total number of parameters (i.e., weights and biases) of the neural network is equal to 269,417.

In Figure 13, we compare the FOM and POD-DL-ROM velocity field magnitudes, the latter obtained with  $N = 64$  and  $n = 5$  for the testing-parameter instance  $\mu_{test} = (5.9102, 3.1179)$  at the systolic peak  $t = 0.18$  s, along with the relative error  $\epsilon_k$  reported in Figure 14. By looking at the pattern and the magnitude of the vector velocity field in Figure 13, it is evident that the abnormal bulge and the inlet are the portions of the domain where the blood flow velocity is smaller, and we remark the ability of the POD-DL-ROM technique in capturing such dynamics in an extremely detailed manner.

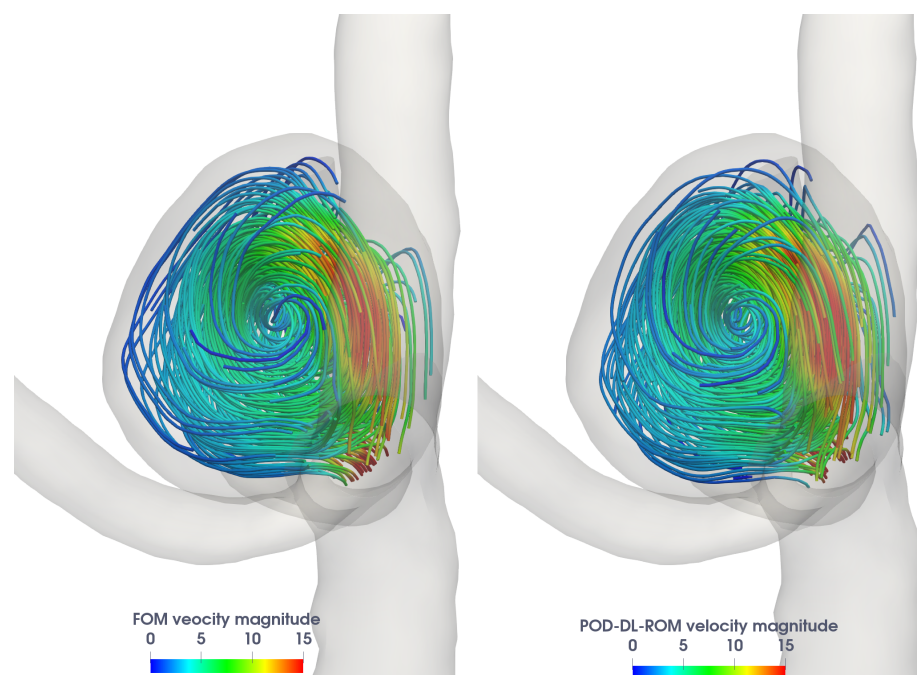


**Figure 13.** Test case 3: FOM (top) and POD-DL-ROM (bottom) velocity fields for the testing-parameter instance  $\mu_{test} = (5.9102, 3.1179)$  at the systolic peak  $t = 0.18$  s.

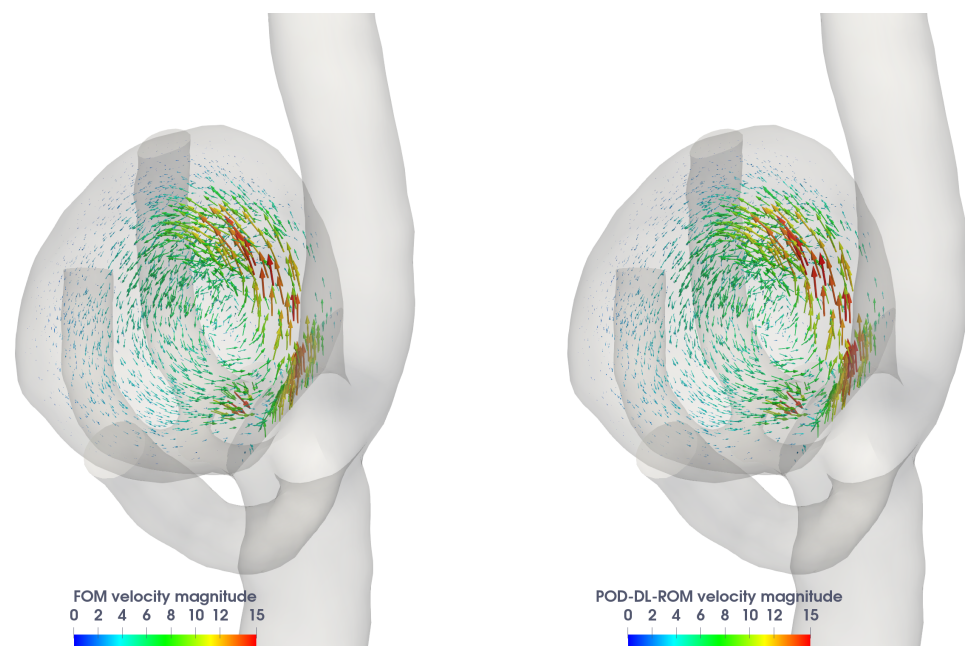


**Figure 14.** Test case 3: Relative error in the velocity magnitude for the testing-parameter instance  $\mu_{test} = (5.9102, 3.1179)$  at the systolic peak  $t = 0.18$  s.

In Figure 15, we report the streamlines of the blood velocity field, obtained with the FOM and the POD-DL-ROM, for the testing-parameter instance  $\mu_{test} = (5.9102, 3.1179)$  at  $t = 0.5$  s. In Figure 16, we report instead a detailed view of the pattern of the fluid velocity field obtained for the testing-parameter instance  $\mu_{test} = (5.9102, 3.1179)$  at  $t = 0.18$  s, highlighting the recirculation of the flow in the bulge and the blood stasis in this region.



**Figure 15.** Test case 3: FOM (left) and POD-DL-ROM (right) velocity magnitude streamlines for the testing-parameter instance  $\mu_{test} = (5.9102, 3.1179)$  at the systolic peak  $t = 0.5$  s.



**Figure 16.** Test case 3: FOM (left) and POD-DL-ROM (right) velocity field magnitude for the testing-parameter instance  $\mu_{test} = (5.9102, 3.1179)$  at  $t = 0.18$  s.

Finally, the testing computational time, i.e., the time needed to compute  $N_t$  time instances for an unseen testing-parameter instance, of the POD-DL-ROM on a Tesla V100 32 GB GPU is given by 0.28 s, thus implying a speed-up equal to  $3.98 \times 10^5$  with respect to the time needed for the solution of the FOM, and the possibility to obtain a fully detailed simulation of a complex blood flow in real-time. For test case 3, the FOM simulations have been carried out on 20 cores of 1.7 TB node (192 Intel® Xeon Platinum® 8160 2.1 GHz cores) of the HPC cluster available at MOX, Politecnico di Milano.

We point out that a similar test case is analyzed in [77], dealing with blood flows through a cerebral aneurysm, however employing a classical POD-Galerkin ROM. In that specific case, even by looking for a lower accuracy compared to our results and relying on a similar amount of snapshots data, the speed-up introduced is only about  $10^2$ , due to the relatively large dimension of the ROM caused by the linear superimposition of (global) basis functions.

#### 4. Discussion

In this work, we have taken advantage of a recently proposed technique [42] to build nonintrusive low-dimensional ROMs by exploiting DL algorithms to handle fluid dynamics problems. This strategy allows us to overcome some drawbacks of classical projection-based ROM techniques arising when they are applied to incompressible flow simulations.

In particular, POD-DL-ROMs overcome the need of:

- treating efficiently nonlinearities and (nonaffine) parameter dependencies, thus avoiding expensive and intrusive hyper-reduction techniques;
- approximating both velocity and pressure fields, in those cases where one might be interested only in the visualization of a single field;
- imposing physical constraints that couple different submodels, as in the case of fluid–structure interaction (the different field variables are indeed treated as independent by the neural network);
- ensuring the ROM stability by enriching the reduced basis spaces;
- solving a dynamical system at the reduced level to model the fluid dynamics, though keeping the error propagation in time under control.

We assessed the performance of the POD DL-ROM technique on three test cases, dealing with the flow around a cylinder benchmark, the fluid–structure interaction between an elastic beam attached to a fixed, rigid block and a laminar incompressible flow, and the blood flow in a cerebral aneurysm, by showing its ability in providing accurate and efficient (even on moderately large-scale problems) ROMs, which multi-query and real-time applications may ultimately rely on. In particular, the prior dimensionality reduction performed through POD on the snapshot matrices also enhances the overall efficiency of the technique during the offline training stage.

Despite that their construction is mainly data-driven, being informed by the physics only through the snapshots, the POD-DL-ROMs provide results that are consistent with the underlying physical model. Indeed, the residual of the POD-Galerkin ROM evaluated on the POD-DL-ROM solution, computed during the training phase, decreases over the epochs even if this term is not included in the loss function. A possible pitfall of the methodology is represented by the amount/quality of training data: if too few (or low-quality) snapshots are considered, further operations such as (i) increasing the number of parameters of the network, or (ii) training the network for a larger number of epochs, or (iii) generating more data by means of data augmentation techniques can be required. Finally, a relevant issue is related to the generalization properties of the network *outside* the parameter range and/or the time interval where snapshots have been sampled. At the moment, ensuring good approximation properties when handling long-time scenarios, even in presence of almost periodic regimes, without more specific network architectures, is an open issue in which our efforts are focused—however, this represents a general aspect related with several machine/deep learning algorithms.

Therefore, we can conclude that POD-DL-ROMs provide a nonintrusive and general-purpose tool enabling us to perform real-time numerical simulations of fluid flows. Since they return a (FOM-like detailed) computation of the field variables, rather than approximating selected output quantities of interest as in the case of traditional emulators or surrogate models, POD-DL-ROMs are a viable tool for detailed flow analysis, without any requirement in terms of computational resources during the online testing stage. Last, but not least, we highlight that, in contrast to FOM simulations, both the training and testing phases of the POD-DL-ROM neural networks can be carried out on a fairly inexpensive GPU—such as those ones that nowadays can be found in a mid-tier personal computer, bolstering the case that this approach could ultimately be exploited without the need of high-performance computing resources.

**Author Contributions:** Conceptualization: S.F. and A.M.; Formal analysis: S.F.; Funding acquisition: A.M.; Investigation: S.F.; Methodology: S.F. and A.M.; Software: S.F.; Supervision: A.M.; Validation: S.F.; Visualization: S.F.; Writing: S.F. and A.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work has been supported by Fondazione Cariplo (grant agreement no. 2019-4608, P.I. A. Manzoni).

**Data Availability Statement:** The code used in the numerical tests is available at <https://github.com/stefaniafresca/POD-DL-ROM>.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Quarteroni, A.; Manzoni, A.; Vergara, C. The cardiovascular system: Mathematical modelling, numerical algorithms and clinical applications. *Acta Numer.* **2017**, *26*, 365–590. [[CrossRef](#)]
2. Quarteroni, A.; Manzoni, A.; Negri, F. *Reduced Basis Methods for Partial Differential Equations: An Introduction*; Springer: Cham, Switzerland, 2016.
3. Kunisch, K.; Volkwein, S. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM J. Numer. Anal.* **2002**, *40*, 492–515. [[CrossRef](#)]
4. Gunzburger, M.D.; Peterson, J.S.; Shadid, J.N. Reducer-order modeling of time-dependent PDEs with multiple parameters in the boundary data. *Comput. Methods Appl. Mech. Eng.* **2007**, *196*, 1030–1047. [[CrossRef](#)]

5. Bergmann, M.; Bruneau, C.H.; Iollo, A. Enablers for robust POD models. *J. Comput. Phys.* **2009**, *228*, 516–538. [[CrossRef](#)]
6. Weller, J.; Lombardi, E.; Bergmann, M.; Iollo, A. Numerical methods for low-order modeling of fluid flows based on POD. *Int. J. Numer. Methods Fluids* **2010**, *63*, 249–268. [[CrossRef](#)]
7. 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.* **2015**, *102*, 1136–1161. [[CrossRef](#)]
8. Dal Santo, N.; Manzoni, A. Hyper-reduced order models for parametrized unsteady Navier–Stokes equations on domains with variable shape. *Adv. Comput. Math.* **2019**, *45*, 2463–2501. [[CrossRef](#)]
9. Veroy, K.; Patera, A. Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: Rigorous reduced-basis a posteriori error bounds. *Int. J. Numer. Method Fluids* **2005**, *47*, 773–788. [[CrossRef](#)]
10. Deparis, S. Reduced basis error bound computation of parameter-dependent Navier–Stokes equations by the natural norm approach. *SIAM J. Numer. Anal.* **2008**, *46*, 2039–2067. [[CrossRef](#)]
11. Manzoni, A. An efficient computational framework for reduced basis approximation and a posteriori error estimation of parametrized Navier–Stokes flows. *ESAIM Math. Model. Numer. Anal.* **2014**, *48*, 1199–1226. [[CrossRef](#)]
12. Yano, M. A Space-Time Petrov–Galerkin Certified Reduced Basis Method: Application to the Boussinesq Equations. *SIAM J. Sci. Comput.* **2014**, *36*, A232–A266. [[CrossRef](#)]
13. Lassila, T.; Manzoni, A.; Quarteroni, A.; Rozza, G. Model order reduction in fluid dynamics: Challenges and perspectives. In *Reduced Order Methods for Modeling and Computational Reduction*; Quarteroni, A., Rozza, G., Eds.; Springer: Cham, Switzerland, 2014; Volume 9, pp. 235–274.
14. 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*, 623–647. [[CrossRef](#)]
15. Caiazzo, A.; Iliescu, T.; John, V.; Schyschlowa, S. A numerical investigation of velocity–pressure reduced order models for incompressible flows. *J. Comput. Phys.* **2014**, *259*, 598–616. [[CrossRef](#)]
16. Carlberg, K.; Bou-Mosleh, C.; Farhat, C. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *Int. J. Numer. Methods Eng.* **2011**, *86*, 155–181. [[CrossRef](#)]
17. Baiges, J.; Codina, R.; Idelsohn, S. Explicit reduced-order models for the stabilized finite element approximation of the incompressible Navier–Stokes equations. *Int. J. Numer. Meth. Fluids* **2013**, *72*, 1219–1243. [[CrossRef](#)]
18. Rozza, G.; Huynh, D.; Manzoni, A. Reduced basis approximation and error bounds for Stokes flows in parametrized geometries: Roles of the inf–sup stability constants. *Numer. Math.* **2013**, *125*, 115–152. [[CrossRef](#)]
19. Dal Santo, N.; Deparis, S.; Manzoni, A.; Quarteroni, A. An algebraic least squares reduced basis method for the solution of parametrized Stokes equations. *Comput. Meth. Appl. Mech. Eng.* **2019**, *344*, 186–208. [[CrossRef](#)]
20. Lassila, T.; Manzoni, A.; Quarteroni, A.; Rozza, G. A reduced computational and geometrical framework for inverse problems in hemodynamics. *Int. J. Numer. Methods Biomed. Eng.* **2013**, *29*, 741–776. [[CrossRef](#)]
21. Colciago, C.; Deparis, S.; Quarteroni, A. Comparisons between reduced order models and full 3D models for fluid–structure interaction problems in haemodynamics. *J. Comput. Appl. Math.* **2014**, *265*, 120–138. [[CrossRef](#)]
22. Brunton, S.; Noack, B.; Koumoutsakos, P. Machine learning for fluid mechanics. *Ann. Rev. Fluid Mech.* **2020**, *52*, 477–508. [[CrossRef](#)]
23. Kutz, J. Deep learning in fluid dynamics. *J. Fluid Mech.* **2017**, *814*, 1–4. [[CrossRef](#)]
24. Ströfer, C.; Wu, J.L.; Xiao, H.; Paterson, E. Data-driven, physics-based feature extraction from fluid flow fields using convolutional neural networks. *Commun. Comput. Phys.* **2019**, *25*, 625–650. [[CrossRef](#)]
25. Lye, K.; Mishra, S.; Ray, D. Deep learning observables in computational fluid dynamics. *J. Comput. Phys.* **2020**, *410*, 109339. [[CrossRef](#)]
26. Bhatnagar, S.; Afshar, Y.; Pan, S.; Duraisamy, K.; Kaushik, S. Prediction of aerodynamic flow fields using convolutional neural networks. *Comput. Mech.* **2019**, *64*, 525–545. [[CrossRef](#)]
27. Thuerey, N.; Weissenow, K.; Prantl, L.; Hu, X. Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows. *AIAA J.* **2020**, *58*, 25–36. [[CrossRef](#)]
28. Eichinger, M.; Heinlein, A.; Klawonn, A. Stationary flow predictions using convolutional neural networks. In *Numerical Mathematics and Advanced Applications ENUMATH 2019*; Vermolen, F., Vuik, C., Eds.; Lecture Notes in Computational Science and Engineering; Springer: Cham, Switzerland, 2021; Volume 139, pp. 541–549.
29. Wang, J.X.; Huang, J.; Duan, L.; Xiao, H. Prediction of Reynolds stresses in high-Mach-number turbulent boundary layers using physics-informed machine learning. *Theor. Comput. Fluid Dyn.* **2019**, *33*, 1–19. [[CrossRef](#)]
30. Beck, A.; Flad, D.; Munz, C.D. Deep neural networks for data-driven LES closure models. *J. Comput. Phys.* **2019**, *398*, 108910. [[CrossRef](#)]
31. San, O.; Maulik, R. Neural network closures for nonlinear model order reduction. *Adv. Comput. Math.* **2018**, *44*, 1717–1750. [[CrossRef](#)]
32. Wang, Q.; Ripamonti, N.; Hesthaven, J. Recurrent neural network closure of parametric POD–Galerkin reduced-order models based on the Mori–Zwanzig formalism. *J. Comput. Phys.* **2020**, *410*, 109402. [[CrossRef](#)]
33. Baiges, J.; Codina, R.; Castañar, I.; Castillo, E. A finite element reduced-order model based on adaptive mesh refinement and artificial neural networks. *Int. J. Numer. Methods Eng.* **2020**, *121*, 588–601. [[CrossRef](#)]

34. Raissi, M.; Yazdani, A.; Karniadakis, G. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **2020**, *367*, 1026–1030. [CrossRef] [PubMed]
35. Raissi, M.; Perdikaris, P.; Karniadakis, G. 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. [CrossRef]
36. Kissas, G.; Yang, Y.; Hwuang, E.; Witschey, W.; Detre, J.; Perdikaris, P. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **2020**, *358*, 112623. [CrossRef]
37. Sun, L.; Wang, J.X. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theor. Appl. Mech. Lett.* **2020**, *10*, 161–169. [CrossRef]
38. Hesthaven, J.; Ubbiali, S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J. Comput. Phys.* **2018**, *363*, 55–78. [CrossRef]
39. Wang, Q.; Hesthaven, J.; Ray, D. Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *J. Comput. Phys.* **2019**, *384*, 289–307. [CrossRef]
40. San, O.; Maulik, R.; Ahmed, M. An artificial neural network framework for reduced order modeling of transient flows. *Commun. Nonlinear Sci. Numer. Simul.* **2019**, *77*, 271–287. [CrossRef]
41. Fukami, K.; Maulik, R.; Ramachandra, N.; Fukagata, K.; Taira, K. Probabilistic neural network-based reduced-order surrogate for fluid flows. *arXiv* **2020**, arXiv:2012.08719.
42. Fresca, S.; Manzoni, A. POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *arXiv* **2021**, arXiv:2101.11845.
43. Xiao, D.; Fang, F.; Pain, C.; Hu, G. Non-intrusive reduced-order modelling of the Navier–Stokes equations based on RBF interpolation. *Int. J. Numer. Methods Eng.* **2015**, *79*, 580–595. [CrossRef]
44. Xiao, D.; Fang, F.; Buchan, A.; Pain, C.; Navon, I.; Muggeridge, A. Non-intrusive reduced order modelling of the Navier–Stokes equations. *Comput. Methods Appl. Mech. Eng.* **2015**, *293*, 522–541. [CrossRef]
45. Xiao, D.; Yang, P.; Fang, F.; Xiang, J.; Pain, C.; Navon, I. Non-intrusive reduced order modelling of fluid–structure interactions. *Comput. Methods Appl. Mech. Eng.* **2016**, *303*, 35–54. [CrossRef]
46. Wang, Z.; Xiao, D.; Fang, F.; Govindan, R.; Pain, C.; Guo, Y. Model identification of reduced order fluid dynamics systems using deep learning. *Int. J. Numer. Methods Fluids* **2018**, *86*, 255–268. [CrossRef]
47. Lui, H.; Wolf, W. Construction of reduced-order models for fluid flows using deep feedforward neural networks. *J. Fluid Mech.* **2019**, *872*, 963–994. [CrossRef]
48. Brunton, S.; Proctor, J.; Kutz, J. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Nat. Acad. Sci. USA* **2016**, *113*, 3932–3937. [CrossRef] [PubMed]
49. Rudy, S.H.; Kutz, J.N.; Brunton, S.L. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *J. Comput. Phys.* **2019**, *396*, 483–506. [CrossRef]
50. González, F.J.; Balajewicz, M. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. *arXiv* **2018**, arXiv:1808.01346.
51. Bukka, S.; Gupta, R.; Magee, A.; Jaiman, R. Assessment of unsteady flow predictions using hybrid deep learning based reduced-order models. *Phys. Fluids* **2021**, *33*, 013601. [CrossRef]
52. Lee, K.; Carlberg, K.T. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Comput. Phys.* **2020**, *404*, 108973. [CrossRef]
53. Bazilevs, Y.; Takizawa, K.; Tezduyar, T. *Computational Fluid-Structure Interaction: Methods and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
54. Fresca, S.; Dedè, L.; Manzoni, A. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. *J. Sci. Comput.* **2021**, *87*, 1–36. [CrossRef]
55. Fresca, S.; Manzoni, A.; Dedè, L.; Quarteroni, A. Deep learning-based reduced order models in cardiac electrophysiology. *PLoS ONE* **2020**, *15*, e0239416. [CrossRef]
56. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
57. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
58. Clevert, D.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv* **2015**, arXiv:1511.07289.
59. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
60. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
61. Negri, F. redbKIT Version 1.0. 2016. Available online: <http://redbkit.github.io/redbKIT/> (accessed on 1 March 2021).
62. Schäfer, M.; Turek, S. Benchmark computations of laminar flow around a cylinder. In *Flow Simulation with High-Performance Computers II*; Hirschel, E.H., Ed.; Vieweg+ Teubner Verlag: Braunschweig/Wiesbaden, Germany, 1996; pp. 547–566.



63. Forti, D.; Dedè, L. Semi-implicit BDF time discretization of the Navier-Stokes equations with VMS-LES modeling in a high performance computing framework. *Comput. Fluids* **2015**, *117*, 168–182. [[CrossRef](#)]
64. Singh, S.; Mittal, S. Vortex-induced oscillations at low Reynolds numbers: Hysteresis and vortex-shedding modes. *J. Fluids Struct.* **2005**, *20*, 1085–1104. [[CrossRef](#)]
65. Wall, W. Fluid Structure Interaction with Stabilized Finite Elements. Ph.D. Thesis, University of Stuttgart, Stuttgart, Germany, 1999.
66. Wall, W.; Ramm, E. Fluid Structure Interaction Based Upon a Stabilized (ALE) Finite Element Method. In Proceedings of the 4th World Congress on Computational Mechanics: New Trends and Applications, Buenos Aires, Argentina, 29 June–2 July 1998.
67. Bazilevs, Y.; Calo, V.M.; Hughes, T.; Zhang, Y. Isogeometric fluid-structure interaction: Theory, algorithms, and computations. *Comp. Mech.* **2008**, *43*, 3–37. [[CrossRef](#)]
68. Crosetto, P.; Deparis, S.; Fourestey, G.; Quarteroni, A. Parallel algorithms for fluid-structure interaction problems in haemodynamics. *SIAM J. Sci. Comput.* **2011**, *33*, 1598–1622. [[CrossRef](#)]
69. Gee, M.; Küttler, U.; Wall, W. Truly monolithic algebraic multigrid for fluid–structure interaction. *Int. J. Numer. Methods Eng.* **2011**, *85*, 987–1016. [[CrossRef](#)]
70. Bazilevs, Y.; Hsu, M.C.; Zhang, Y.; Wang, W.; Liang, X.; Kvamsdal, T.; Brekken, R.; Isaksen, J. A fully-coupled fluid-structure interaction simulation of cerebral aneurysms. *Comput. Mech.* **2010**, *46*, 3–16. [[CrossRef](#)]
71. Cebal, J.; Mut, F.; Sforza, D.; Löhner, R.; Scrivano, E.; Lylyk, P.; Putman, C. Clinical application of image-based CFD for cerebral aneurysms. *Int. J. Numer. Methods Biomed. Eng.* **2011**, *27*, 977–992. [[CrossRef](#)]
72. Valencia, A.; Morales, H.; Rivera, R.; Bravo, E.; Galvez, M. Blood flow dynamics in patient-specific cerebral aneurysm models: The relationship between wall shear stress and aneurysm area index. *Med. Eng. Phys.* **2008**, *30*, 329–340. [[CrossRef](#)]
73. AneuriskWeb. The Aneurisk Dataset Repository. Emory University & Orobix Srl, 2012–2013. Available online: <http://ecm2.mathcs.emory.edu/aneuriskweb> (accessed on 1 May 2021).
74. Aneurisk Project. MOX, Mathematics Department, Politecnico di Milano. Available online: <https://statistics.mox.polimi.it/aneurisk/> (accessed on 1 May 2021).
75. Piccinelli, M.; Veneziani, A.; Steinman, D.; Remuzzi, A.; Antiga, L. A framework for geometric analysis of vascular structures: Application to cerebral aneurysms. *IEEE Trans. Med. Imag.* **2009**, *28*, 1141–1155. [[CrossRef](#)] [[PubMed](#)]
76. Blanco, P.J.; Watanabe, S.M.; Passos, M.A.R.F.; Lemos, P.A.; Feijóo, R.A. An Anatomically Detailed Arterial Network Model for One-Dimensional Computational Hemodynamics. *IEEE Trans. Biomed. Eng.* **2015**, *62*, 736–753. [[CrossRef](#)] [[PubMed](#)]
77. Negri, F. Efficient Reduction Techniques for the Simulation and Optimization of Parametrized Systems: Analysis and Applications. Ph.D. Thesis, EPFL Lausanne, Lausanne, Switzerland, 2015.