Available online at www.sciencedirect.com
ScienceDirect

Comput. Methods Appl. Mech. Engrg. 415 (2023) 116232

Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Artificial neural network based correction for reduced order models in computational fluid mechanics

Zulkeefal Dar[a,b], Joan Baiges[a,b], Ramon Codina[a,b,*]

[a] Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE), Edifici C1, Campus Nord UPC, Gran Capità S/N, 08034 Barcelona, Spain
[b] Universitat Politècnica de Catalunya, Barcelona Tech, Jordi Girona 1-3, Edifici C1, 08034 Barcelona, Spain

Received 11 November 2022; received in revised form 20 June 2023; accepted 2 July 2023
Available online 24 July 2023

Abstract

In this paper we propose a reduced order model (ROM) for incompressible flows based on proper orthogonal decomposition (POD) and having a finite element approximation as full order model (FOM). The main idea is to start from a purely POD-based ROM, projecting the equations onto the ROM space, and then add a nonlinear correction that depends on the ROM unknowns to enhance the final ROM model. This correction is based on the fact that we do have some available high fidelity data, namely, the snapshots. Thus, the correcting term is built as an artificial neural network (ANN) constructed with the snapshots as the training set. This correction is then introduced in the fully discrete ROM system to achieve more accurate solutions. A further feature of our approach is that we construct both the ROM and the FOM using the variational multi-scale (VMS) concept, and this allows us to understand the correction term as an approximation to the scales that are lost when passing from the FOM to the ROM. The resulting corrected ROM has a significant higher accuracy than the original one.

Keywords: Reduced order models; Proper orthogonal decomposition; Variational multi-scale; Artificial Neural Networks; Correction models

1. Introduction

Many important dynamical processes occurring in nature can be modeled by Partial Differential Equations (PDEs). But the simulation of such processes can incur a high computational cost. This is particularly true for control and optimization problems as they require a large number of simulations to be performed. A feasible option to overcome the high computational cost related to high fidelity simulations is the use of Reduced Order Models (ROMs). ROMs can reduce this computational cost by approximating the large-scale systems by much smaller ones. A summary of popular techniques and approaches used for reduced order modeling can be found in [1]. One of the most commonly used ROMs are projection based ROMs based on Proper Orthogonal Decomposition (POD). POD-ROMs rely on projecting a high-dimensional Full Order Model (FOM) onto a low order space of solutions. POD-ROMs have been widely used in solid mechanics [2] and fluid mechanics [3–11], as well as in shape optimization [12–15] and flow control [16–19] problems, among other applications.

* Corresponding author at: Universitat Politècnica de Catalunya, Barcelona Tech, Jordi Girona 1-3, Edifici C1, 08034 Barcelona, Spain.
E-mail addresses: zulkeefal.dar@upc.edu (Z. Dar), joan.baiges@upc.edu (J. Baiges), ramon.codina@upc.edu (R. Codina).

https://doi.org/10.1016/j.cma.2023.116232
0045-7825/© 2023 Elsevier B.V. All rights reserved.

The approach proposed in this paper is applied to flow problems governed by the incompressible Navier–Stokes equations, and the FOM is obtained from their finite element (FE) approximation. Nevertheless, the methodology we propose here can be extended to any flow problem discretized by standard numerical techniques.

The idea of POD-ROMs applied to the incompressible Navier–Stokes equations is simple. From a collection of snapshots in time, including velocities and pressures, one may construct a basis of a low-dimensional space by keeping R modes of the singular value decomposition (SVD) of the matrix of snapshots. The unknown (velocity and pressure) is then expressed in this basis, plugged into the variational form of the problem and projected onto the space spanned by the basis. In order to make the method effective, only a few high energy modes are kept, and the remaining low energy modes are discarded. However, this can lead to instability and inaccuracy in the ROM solution. This can be attributed to the inherent nonlinearity of the system, resulting in an interaction of the discarded modes with the resolved modes, and also to intrinsic instabilities of the approximation of the incompressible Navier–Stokes equations (velocity–pressure compatibility, convection-dominated flows). So, *correction models* taking into account the effect of unresolved modes on resolved modes are required for the accurate prediction of the dynamics. We will refer to discarded modes as the sub-grid scales (SGSs) in this work. We propose here a two-step approach for the closure modeling of the SGSs, namely, an a priori approach focused on providing the required stabilization followed by an a posteriori approach to achieve the desired accuracy in highly truncated ROMs.

The a priori closure modeling approach used is based on the Variational Multi-Scale (VMS) method, similar to the one used by Reyes and Codina [20] (see also [21]), inspired by the application of VMS for FE methods. The issue of closure modeling is not unique to projection based ROMs, rather it is commonly found in other projection-based methods like Galerkin FE methods, where stabilized formulations have been devised to deal with the instabilities of the Galerkin projection method. Elaborate models have been developed to improve the stability and accuracy of FEs using VMS. These include time dependent dynamic SGSs [22], orthogonal SGSs [23] and element boundary SGSs [24–26]. VMS methods have been applied successfully to a variety of FE problems beyond the incompressible Navier–Stokes equations [27–32]. Extending this to ROMs, Reyes and Codina [20] developed the VMS-ROM using orthogonal SGSs. The method has been applied to thermally coupled low Mach flows [33] and to fluid–structure interaction [34,35]. Owing to the successful application of the VMS-ROM in a variety of cases, we use it as the a prior approach in this work. Note that the VMS-ROM presented in these references improves the accuracy of the solution alongside providing stability. However, the provided accuracy can be insufficient for highly truncated ROMs.

Thus, as the second step we introduce an a posteriori SGS approach based on Machine Learning (ML) using Artificial Neural Networks (ANNs). The use of ANNs for ROMs can be grouped into two categories: intrusive and non-intrusive approaches. Non-intrusive approaches [36–39] do not rely on the governing equations, rather they make predictions based purely on data. As a consequence, non-intrusive approaches require a significant amount of data to work. The typical sparsity of data generally available motivates an intrusive or hybrid approach. In the hybrid approach, instead of by-passing the Galerkin projection step, ANNs are used to build closure models augmenting the physics-based ROMs. Such an approach was used in [40] to model modal eddy viscosity as a function of the mode number, the right-hand side (RHS) of the Galerkin ROM (GROM), and the modal amplitudes using extreme learning machine (ELM) of single layer. ANNs can be used to directly predict the numerical value of the closure term, without assuming the functional form associated to the closure model. Wan et al. [41] used Recurrent Neural Networks to learn the error in the RHS of GROM as a function of the time-history of values of the resolved state to model extreme events. The Mori–Zwanzig formalism [42,43] states that the closure term can be modeled as a memory term based on the time history of the resolved scales. Gupta and Lermusiaux [44] proposed ‘neural closure models’ using neural delay differential equations to model the memory integral terms; they concluded that discrete delays perform better than the distributed delays for closure modeling. Wang et el. [45] utilized a conditioned long short-term memory with an implicit–explicit Runge–Kutta time integration scheme to approximate the memory term. San and Maulik [46,47] utilized a single layer feed forward neural network using ELM [48] to learn the value of the closure term as a function of the GROM RHS and applied to thermal fluid problems. Mou et el. [49] developed a data-driven closure model based on the VMS approach using least squares.

The works mentioned above suggest a closure model for the semi-discretized system, i.e., discretized in thespace

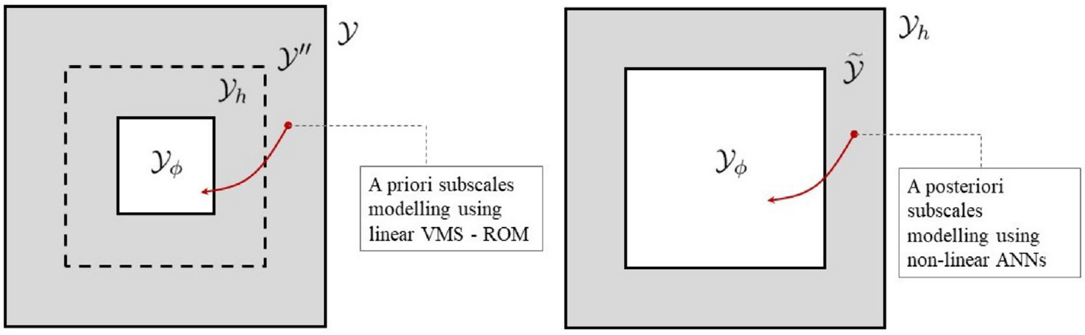
$\mathcal{Y}$
$\mathcal{Y}''$
$\mathcal{Y}_h$
$\mathcal{Y}_\phi$
A priori subscales modelling using linear VMS - ROM
$\mathcal{Y}_h$
$\widetilde{\mathcal{Y}}$
$\mathcal{Y}_\phi$
A posteriori subscales modelling using non-linear ANNs

Fig. 1. Two-step approach for the SGS modeling explained using the associated functional spaces. SGSs spaces are highlighted in gray. Left: a priori step. Right: a posteriori step.

only. However, in this article we present a correction model for the fully discretized system of the ROM. In this way, we account for the errors in the time discretization as well. A correction term was also added to the fully discrete system in [50]; however, it used a linear least square approach to model the correction, whereas in this work non-linear ANNs are used without assuming any ansatz. The suitable nonlinear function that maps the chosen inputs to SGSs is established by the ANN itself during the training phase. The key point is how the correction term can be constructed. To this end, we *reuse the snapshots used to build the ROM basis*, as they are high-fidelity solutions. The ANN is then trained by imposing that the results of the corrected ROM be as close as possible to the projection of the snapshots onto the ROM space at the time instants in which they are available. Note that our approach is motivated by the splitting of scales and we correct the ROM system, and not the solution itself as might be suggested by the loss function of the ANN. Our correction being equal to the difference in the projected FOM solution and the uncorrected ROM solution, Eq. (26), is the consequence of our choice of using Eq. (21).

ANNs are generally regarded as black boxes, which receive inputs and provide outputs. In this work, a dedicated effort has been made to use the simplest ANNs that could produce the desired results to keep the ambiguity associated with ANNs to a minimum. Furthermore, being the method proposed hybrid, the suggested approach requires a very small amount of data and simple architectures to capture the SGSs, thus leading to a very fast training phase. Overall, the use of ANNs leads to improved results at an insignificant computational cost and using the same degrees of freedom as the original ROM.

The two steps of the closure model account for the SGSs belonging to different subspaces. Let us denote the space of the continuous problem as $\mathcal{Y}$, the FOM space as $\mathcal{Y}_h$ and the ROM space as $\mathcal{Y}_\phi$. Furthermore, we denote the different SGS spaces as $\mathcal{Y}'$, $\mathcal{Y}''$, $\widetilde{\mathcal{Y}}$, defined such that $\mathcal{Y} = \mathcal{Y}_h \oplus \mathcal{Y}' = \mathcal{Y}_\phi \oplus \mathcal{Y}''$ and $\mathcal{Y}_h = \mathcal{Y}_\phi \oplus \widetilde{\mathcal{Y}}$. Now, the a priori SGSs are modeled in the $\mathcal{Y}''$ subspace. Since the a posteriori approach is data-driven, it can only model the reduced order SGSs present in the FOM solution space, $\mathcal{Y}_h$ in this case. So, the second step models the SGSs present in $\widetilde{\mathcal{Y}}$. Fig. 1 explains the two steps of this approach with SGSs spaces highlighted in gray.

The paper is organized as follows. In Section 2, we describe the variational problem along with the a priori approach, i.e., the VMS-ROM, employed in this work. In Section 3, we describe the a posteriori approach of how we correct the fully discrete system. In Section 4, we introduce the ANNs and how we design and train them. In Section 5, we present numerical examples for the flow over a cylinder, a backward facing step and three cylinders in a triangular arrangement. The examples are carried out within and outside the training regime. In Section 6, we provide the concluding remarks.

2. Problem statement and a priori sub-grid scale modeling

2.1. Variational problem

Let us consider the incompressible Navier–Stokes equations as the target problem. Let $\Omega \subset \mathbb{R}^d$, d denoting the dimensions of the problem, be the domain in which the fluid flows. The boundary Γ of the domain is divided into Dirichlet and Neumann boundaries denoted by Γ_D and Γ_N, respectively, such that $\Gamma = \Gamma_D \cup \Gamma_N$ and $\emptyset = \Gamma_D \cap \Gamma_N$. Let $]0, t_\mathrm{f}[$ be the time interval for which the flow needs to be analyzed. Calling $\boldsymbol{u}^0$ the initial velocity, the incompressible Navier–Stokes problem reads: find a velocity–pressure field $[\boldsymbol{u}, p] : \Omega\times]0, t_\mathrm{f}[\to \mathbb{R}^d \times \mathbb{R}$ such that

$$\begin{aligned}
\partial_t \boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \qquad & \text{in } \Omega, \quad t\in\,]0, t_\mathrm{f}[,\\
\nabla\cdot\boldsymbol{u} = 0 \qquad & \text{in } \Omega, \quad t\in\,]0, t_\mathrm{f}[,\\
\boldsymbol{u} = \boldsymbol{0} \qquad & \text{on } \Gamma_D, \quad t\in\,]0, t_\mathrm{f}[,\\
\boldsymbol{n}\cdot(\nu\nabla\boldsymbol{u} - p\boldsymbol{I}_{(d)}) = \boldsymbol{0} \qquad & \text{on } \Gamma_N, \quad t\in\,]0, t_\mathrm{f}[,\\
\boldsymbol{u} = \boldsymbol{u}^0 \qquad & \text{in } \Omega, \quad t = 0,
\end{aligned}$$

where ν is the kinematic viscosity of the fluid, $\boldsymbol{n}$ is the outwards unit normal to $\partial\Omega$, $\boldsymbol{f} : \Omega\times]0, t_\mathrm{f}[\to \mathbb{R}^d$ is the forcing vector, and $\boldsymbol{I}_{(d)}$ is the $d\times d$ identity matrix. To simplify the exposition, homogeneous Dirichlet and Neumann boundary conditions have been considered.

We present now the equivalent weak form. Let $\langle\cdot,\cdot\rangle_\omega$ be the integral over a domain $\omega \subseteq \Omega$ of the product of two functions, with the subscript omitted when $\omega = \Omega$. We take $\boldsymbol{u}$ for each time t and the corresponding test function $\boldsymbol{v}$ in the functional space $\mathcal{V}$, consisting of functions in $H^1(\Omega)^d$ vanishing on Γ_D, and p for each time t and the corresponding test function q in the functional space $\mathcal{Q} = L^2(\Omega)$ (or $\mathcal{Q} = L^2(\Omega)/\mathbb{R}$ if $\Gamma_D = \emptyset$). The weak form of the Navier–Stokes problem reads: find $[\boldsymbol{u}, p] :]0, t_\mathrm{f}[\to \mathcal{V}\times\mathcal{Q}$ such that

$$\begin{aligned}
\langle\partial_t\boldsymbol{u}, \boldsymbol{v}\rangle + \nu\langle\nabla\boldsymbol{u}, \nabla\boldsymbol{v}\rangle + \langle\boldsymbol{u}\cdot\nabla\boldsymbol{u}, \boldsymbol{v}\rangle - \langle p, \nabla\cdot\boldsymbol{v}\rangle = \langle\boldsymbol{f}, \boldsymbol{v}\rangle, \qquad & \forall\boldsymbol{v}\in\mathcal{V}, \quad t\in\,]0, t_\mathrm{f}[,\\
\langle q, \nabla\cdot\boldsymbol{u}\rangle = 0, \qquad & \forall q\in\mathcal{Q}, \quad t\in\,]0, t_\mathrm{f}[,\\
\langle\boldsymbol{u}, \boldsymbol{v}\rangle = \langle\boldsymbol{u}^0, \boldsymbol{v}\rangle, \qquad & \forall\,\boldsymbol{v}\in L^2(\Omega)^d, \quad \text{at } t = 0.
\end{aligned} \tag{1}$$

For conciseness, let us define the functional space $\mathcal{Y} = \mathcal{V}\times\mathcal{Q}$, the unknown $\boldsymbol{y} = [\boldsymbol{u}, p] \in \mathcal{Y}$ and the test function $\boldsymbol{z} = [\boldsymbol{v}, q] \in \mathcal{Y}$. We can write problem (1) as: find $\boldsymbol{y} :]0, t_\mathrm{f}[\to \mathcal{Y}$ such that

$$\langle\partial_t\boldsymbol{u}, \boldsymbol{v}\rangle + B(\boldsymbol{u}; \boldsymbol{y}, \boldsymbol{z}) = L(\boldsymbol{z}), \quad \forall\,\boldsymbol{z}\in\mathcal{Y}, \quad t\in\,]0, t_\mathrm{f}[, \tag{2}$$

where

$$\begin{aligned}
B(\boldsymbol{w}; \boldsymbol{y}, \boldsymbol{z}) = B(\boldsymbol{w}; [\boldsymbol{u}, p], [\boldsymbol{v}, q]) &:= \nu\langle\nabla\boldsymbol{u}, \nabla\boldsymbol{v}\rangle + \langle\boldsymbol{w}\cdot\nabla\boldsymbol{u}, \boldsymbol{v}\rangle - \langle p, \nabla\cdot\boldsymbol{v}\rangle + \langle q, \nabla\cdot\boldsymbol{u}\rangle,\\
L(\boldsymbol{z}) = L([\boldsymbol{v}, q]) &:= \langle\boldsymbol{f}, \boldsymbol{v}\rangle.
\end{aligned}$$

We have not stated the initial conditions for problem (2) explicitly. From here onwards it is understood that the problem needs to be provided with the appropriate initial conditions.

2.2. VMS-FOM approximation

Let us consider the FE space $\mathcal{Y}_h \subset \mathcal{Y}$ for the FOM. This space is assumed to be built from a FE partition $\{K\}$ of diameter $h = \max_K\{h_K\}$ of Ω, with $h_K = \mathrm{diam}(K)$. The collection of interior edges (faces, if $d = 3$) is denoted as $\{E\}$. As usual, subscript h will be introduced to refer to FE functions and spaces, the latter considered conforming.

The core of the VMS approach, originally proposed in [51], lies in splitting $\mathcal{Y} = \mathcal{Y}_h \oplus \mathcal{Y}'$, where $\mathcal{Y}'$ is any space that completes $\mathcal{Y}_h$ in $\mathcal{Y}$. Thus, any function $\boldsymbol{g}$ can be written as $\boldsymbol{g} = \boldsymbol{g}_h + \boldsymbol{g}'$, with subscript h denoting a FE function $\in \mathcal{Y}_h$ and superscript $'$ denoting a SGS function $\in \mathcal{Y}'$. Using this splitting, we can replace problem (2) by

$$\langle\partial_t\boldsymbol{u}_h + \partial_t\boldsymbol{u}', \boldsymbol{v}_h\rangle + B(\boldsymbol{u}; \boldsymbol{y}_h + \boldsymbol{y}', \boldsymbol{z}_h) = L(\boldsymbol{z}_h), \quad \forall\,\boldsymbol{z}_h\in\mathcal{Y}_h, \quad t\in\,]0, t_\mathrm{f}[, \tag{3}$$

$$\langle\partial_t\boldsymbol{u}_h + \partial_t\boldsymbol{u}', \boldsymbol{v}'_h\rangle + B(\boldsymbol{u}; \boldsymbol{y}_h + \boldsymbol{y}', \boldsymbol{z}') = L(\boldsymbol{z}'), \quad \forall\,\boldsymbol{z}'\in\mathcal{Y}', \quad t\in\,]0, t_\mathrm{f}[. \tag{4}$$

Note that Eqs. (3)–(4) are exact and we have not introduced any assumptions yet. We want to determine the SGSsfrom Eq. (4) and insert them in Eq. (3) to achieve a stable and more accurate solution in the FOM space. We use the VMS approach developed in [24,52] (see [53] for a review) for this purpose. After some approximations, Eq. (3) can then be transformed to: find $\boldsymbol{y}_h = [\boldsymbol{u}_h, p_h] :]0, t_\mathrm{f}[\to \mathcal{Y}_h$ such that

$$\begin{aligned} &\langle \partial_t \boldsymbol{u}_h, \boldsymbol{v}_h \rangle + B(\boldsymbol{u}^*; \boldsymbol{y}_h, \boldsymbol{z}_h) + \sum_K \langle \boldsymbol{u}'_K, -\nu \Delta \boldsymbol{v}_h - \boldsymbol{u}^* \cdot \nabla \boldsymbol{v}_h - \nabla q_h \rangle_K \\ &\quad - \sum_K \langle p'_K, \nabla \cdot \boldsymbol{v}_h \rangle_K + \sum_E \langle \boldsymbol{u}'_E, [\![\boldsymbol{n} \cdot (\nu \nabla \boldsymbol{v}_h + q_h \boldsymbol{I}_{(d)})]\!] \rangle_E = L(\boldsymbol{z}_h), \quad \forall \boldsymbol{z}_h \in \mathcal{Y}_h, \quad t \in]0, t_\mathrm{f}[, \end{aligned} \tag{5}$$

where $\boldsymbol{u}'_K$, p'_K and $\boldsymbol{u}'_E$ are the solutions of

$$\begin{aligned} \partial_t \boldsymbol{u}'_K + \tau_{1,K}^{-1} \boldsymbol{u}'_K &= \Pi^\perp(\boldsymbol{f} + \nu \Delta \boldsymbol{u}_h - \boldsymbol{u}^* \cdot \nabla \boldsymbol{u}_h - \nabla p_h)|_K, \\ \tau_{2,K}^{-1} p'_K &= -\Pi^\perp(\nabla \cdot \boldsymbol{u}_h)|_K, \\ \tau_E^{-1} \boldsymbol{u}'_E &= -[\![\boldsymbol{n} \cdot (\nu \nabla \boldsymbol{u}_h - p_h \boldsymbol{I}_{(d)})]\!]_E, \end{aligned}$$

where $\tau_{1,K}^{-1}, \tau_{2,K}^{-1}$ and τ_E^{-1} are the stabilization parameters and $[\![\cdot]\!]$ is the jump operator. The choices for the stabilization parameters and the definition of the jump operator can be found in [24,52]. $\Pi^\perp$ is the L^2 orthogonal projection to the corresponding FE space, i.e., orthogonal SGSs will be considered throughout in this work. Concerning the advection velocity $\boldsymbol{u}^*$, one can take it as $\boldsymbol{u}^* = \boldsymbol{u}_h + \boldsymbol{u}'$, yielding what we call nonlinear SGSs, but for simplicity we will approximate $\boldsymbol{u}^* = \boldsymbol{u}_h$ (see [54] for a discussion).

The important feature of the formulation presented is that it allows one to use arbitrary interpolations of velocity and pressure and to deal with convection-dominated flows. In particular, we will consider equal continuous interpolation for velocity and pressure.

Let us discretize the time interval using a partition of size δt, uniform for simplicity. The temporal derivative ∂_t is replaced by δ_t, denoting a backward difference increment of the required order, involving the unknowns at the current and previous time steps. Suitable initializations are required for schemes of order higher than one. The projection of the initial condition onto the FE space is denoted by $\boldsymbol{u}_h^0$. Using a superscript as a counter for the time step and $n+1$ denoting the current time level, the discrete problem reads: for $n = 0, \ldots, nt-1$ (nt is total number of time steps), find $\boldsymbol{y}_h^{n+1} \in \mathcal{Y}_h$ such that

$$\begin{aligned} &\langle \delta_t \boldsymbol{u}_h^{n+1}, \boldsymbol{v}_h \rangle + B(\boldsymbol{u}_h^{n+1}; \boldsymbol{y}_h^{n+1}, \boldsymbol{z}_h) + \sum_K \langle \boldsymbol{u}'^{n+1}_K, -\nu \Delta \boldsymbol{v}_h - \boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{v}_h - \nabla q_h \rangle_K \\ &\quad - \sum_K \langle p'^{n+1}_K, \nabla \cdot \boldsymbol{v}_h \rangle_K + \sum_E \langle \boldsymbol{u}'^{n+1}_E, [\![\boldsymbol{n} \cdot (\nu \nabla \boldsymbol{v}_h + q_h \boldsymbol{I}_{(d)})]\!] \rangle_E = L(\boldsymbol{z}_h), \quad \forall \boldsymbol{z}_h \in \mathcal{Y}_h, \end{aligned} \tag{6}$$

with

$$\begin{aligned} \delta_t \boldsymbol{u}'^{n+1}_K + \tau_{1,K}^{-1} \boldsymbol{u}'^{n+1}_K &= \Pi^\perp(\boldsymbol{f} + \nu \Delta \boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{u}_h^{n+1} - \nabla p_h^{n+1})|_K, \\ \tau_{2,K}^{-1} p'^{n+1}_K &= -\Pi^\perp(\nabla \cdot \boldsymbol{u}_h^{n+1})|_K, \\ \tau_E^{-1} \boldsymbol{u}'^{n+1}_E &= -[\![\boldsymbol{n} \cdot (\nu \nabla \boldsymbol{u}_h^{n+1} - p_h^{n+1} \boldsymbol{I}_{(d)})]\!]_E. \end{aligned}$$

The superscript of the time level is omitted in the following, understanding that the unknown to be computed is that of time level $n+1$.

2.3. A priori sub-grid scales using VMS-ROM

We want to construct a ROM space $\mathcal{Y}_\phi$ such that $\mathcal{Y}_\phi \subset \mathcal{Y}_h \subset \mathcal{Y}$. There are several ways to develop the ROM space, but we use POD, as indicated earlier. We can build a reduced order basis such that the basis functions are L^2 orthogonal to each other. In Section 2.2 we have also introduced Orthogonal SGSs, which are also orthogonal in the L^2 functional sense. It will be shown later in this section that using L^2 orthogonal basis functions allows us to relate ROM SGSs with FOM SGSs explicitly.

Let us describe the difference between what we will call *algebraic orthogonality* (with respect to the identity matrix) and L^2 *functional orthogonality*, and how we achieve the L^2 functional orthogonality for the ROM basis functions.

We start off by collecting FOM results at different time instants, i.e., the snapshots. Let the collection of s discrete snapshots be

$$\boldsymbol{S}_h = \{\boldsymbol{S}_h^{n_1}, \dots, \boldsymbol{S}_h^{n_s}\}, \quad \boldsymbol{S}_h^{n_i} \in \mathbb{R}^{np},\ i = 1, \dots, s,$$

with the vector $\boldsymbol{S}_h^{n_i} = \boldsymbol{Y}_h^{n_i} - \overline{\boldsymbol{Y}}_h$, with $\boldsymbol{Y}_h^{n_i} \in \mathbb{R}^{np}$ being the solution of the FOM at the snapshot instant n_i and $\overline{\boldsymbol{Y}}_h \in \mathbb{R}^{np}$ being the mean value over all snapshot instances. Here, np denotes the total number of degrees of freedom. To simplify the writing, we will take $np = M \times m$, with M being the number of nodes of the FE mesh and m being the degrees of freedom associated with each node of the mesh assuming equal order interpolation for all the unknowns (in our case, $m = d + 1$); nevertheless, the degrees of freedom associated to prescribed velocities need to be excluded. Note that the snapshots need not to be captured for every time step and, hence, we will denote the total number of time steps with nt and the total number of snapshots with s. The same symbol $\boldsymbol{S}_h$ will be used for the matrix in $\mathbb{R}^{np\times s}$ containing the snapshots in columns.

Any array of $\boldsymbol{G} \in \mathbb{R}^{np}$ can be uniquely identified with a vector function of position $\boldsymbol{x} \in \Omega$, $\boldsymbol{g}(\boldsymbol{x}) \in \mathcal{Y}_h$, simply by considering that the components of the array are the nodal values of the FE function. From now on, a lowercase latin subscript in an array (resp. function) will denote a member of a family of arrays (resp. functions), an uppercase subscript a nodal value and a greek subscript a component of a nodal array. Thus, $\boldsymbol{G} \in \mathbb{R}^{np}$ is made of M arrays $\boldsymbol{G}_I$, $I = 1, \dots, M$, each with components $G_{I,\alpha}$, $\alpha = 1, \dots, m$. We shall identify the global index $i = (I - 1) \times m + \alpha$ with I, α, i.e., $G_{I,\alpha} \equiv G_i$, and likewise for matrix components. If $N_I(\boldsymbol{x})$ is the shape function of node I of the FE mesh, recalling that we consider equal interpolation for all variables, we may write

$$g_\alpha(\boldsymbol{x}) = \sum_{I=1}^{M} N_I(\boldsymbol{x}) G_{I,\alpha}, \quad \alpha = 1, \dots m, \iff \boldsymbol{g}(\boldsymbol{x}) = \sum_{I=1}^{M} N_I(\boldsymbol{x}) \boldsymbol{G}_I.$$

Let the truncated SVD of $\boldsymbol{S}_h$ be

$$\boldsymbol{S}_h \approx \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}, \quad \boldsymbol{U} \in \mathbb{R}^{np\times R}, \boldsymbol{\Lambda} \in \mathbb{R}^{R\times R}, \boldsymbol{V} \in \mathbb{R}^{R\times ns}, \tag{7}$$

where R is the dimension of the truncated reduced order space and $\boldsymbol{U}$ is the matrix containing basis vectors, $\boldsymbol{U} = [\boldsymbol{U}_1, \dots, \boldsymbol{U}_R]$, $\boldsymbol{U}_i \in \mathbb{R}^{np}$, with the following property:

$$\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_{(R)}, \tag{8}$$

where $\boldsymbol{I}_{(R)}$ is the $R \times R$ identity matrix, i.e., the basis vectors are orthogonal in *algebraic sense* with respect to the $R \times R$ identity matrix.

We want to construct a basis $\{\boldsymbol{\phi}_1(\boldsymbol{x}), \dots, \boldsymbol{\phi}_R(\boldsymbol{x})\}$ for $\mathcal{Y}_\phi$ such that the basis functions are L^2 orthogonal, that is to say,

$$\int_\Omega \boldsymbol{\phi}_i(\boldsymbol{x}) \cdot \boldsymbol{\phi}_j(\boldsymbol{x}) = \delta_{ij} \quad i, j = 1, \dots, R. \tag{9}$$

Note that $\boldsymbol{\phi}_i(\boldsymbol{x})$ is a function of m components, $i = 1, \dots, R$. Let its array of nodal values be $\boldsymbol{\Phi}_i \in \mathbb{R}^{np}$, and consider the matrix $\boldsymbol{\Phi} = [\boldsymbol{\Phi}_1, \dots, \boldsymbol{\Phi}_R]$, that we wish to construct instead of $\boldsymbol{U}$. The orthogonality condition (9) translates into

$$\boldsymbol{\Phi}^T\boldsymbol{M}\boldsymbol{\Phi} = \boldsymbol{I}_{(R)}, \tag{10}$$

where $\boldsymbol{M}$ is the $np \times np$ mass matrix, constructed as

$$M_{I,\alpha,J,\beta} = \left(\int_\Omega N_I N_J\right)\delta_{\alpha\beta}, \quad I, J = 1, \dots, M,\ \alpha, \beta = 1, \dots, m.$$

Thus, the orthogonality of the basis vectors $\boldsymbol{\Phi}_i, i = 1, \dots, R$, should hold with respect to the mass matrix as inner product. We need to modify the minimization problem associated with the SVD in a way that we get the basis fulfilling (10) instead of (8).

Decomposition (7) is the result of minimizing

$$J(\boldsymbol{U}_1, \dots, \boldsymbol{U}_R) = \sum_{i=1}^{s} \left\| \boldsymbol{S}_h^{n_i} - \sum_{j=1}^{R} ({\boldsymbol{S}_h^{n_i}}^T \boldsymbol{U}_j)\boldsymbol{U}_j \right\|_{\mathbb{R}^{np}}^2 \quad \text{subject to} \quad \boldsymbol{U}_i^T\boldsymbol{U}_j = \delta_{ij}. \tag{11}$$

If instead of the basis satisfying $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_{(R)}$ we want a basis satisfying $\boldsymbol{\Phi}^T\boldsymbol{M}\boldsymbol{\Phi} = \boldsymbol{I}_{(R)}$, we need to replace problem (11) by its functional counterpart of minimizing

$$j(\boldsymbol{\phi}_1(\boldsymbol{x}), \dots, \boldsymbol{\phi}_R(\boldsymbol{x})) = \sum_{i=1}^{s} \left\| \boldsymbol{s}^{n_i}(\boldsymbol{x}) - \sum_{j=1}^{R} \left(\int_\Omega \boldsymbol{s}^{n_i}(\boldsymbol{x}) \cdot \boldsymbol{\phi}_j(\boldsymbol{x}) \right) \boldsymbol{\phi}_j(\boldsymbol{x}) \right\|^2_{L^2(\Omega)} \quad \text{subject to} \quad \int_\Omega \boldsymbol{\phi}_i(\boldsymbol{x}) \cdot \boldsymbol{\phi}_j(\boldsymbol{x}) = \delta_{ij}, \tag{12}$$

where $\boldsymbol{s}^{n_i}(\boldsymbol{x})$ is the function associated to array $\boldsymbol{S}_h^{n_i}, i = 1, \dots, s$. It can be shown that (12) is equivalent to minimizing its algebraic counterpart

$$J_M(\boldsymbol{\Phi}_1, \dots, \boldsymbol{\Phi}_R) = \sum_{i=1}^{s} \left\| \boldsymbol{M}^{1/2}\boldsymbol{S}_h^{n_i} - \sum_{j=1}^{R} \left((\boldsymbol{M}^{1/2}\boldsymbol{S}_h^{n_i})^T \boldsymbol{M}^{1/2}\boldsymbol{\Phi}_j \right) \boldsymbol{M}^{1/2}\boldsymbol{\Phi}_j \right\|^2_{\mathbb{R}^{np}}$$

$$\text{subject to} \quad \boldsymbol{\Phi}^T\boldsymbol{M}\boldsymbol{\Phi} = \boldsymbol{I}_{(R)}.$$

Therefore, if the standard SVD decomposition of $\hat{\boldsymbol{S}} = \boldsymbol{M}^{1/2}\boldsymbol{S}_h$ is

$$\hat{\boldsymbol{S}} = \hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{V}},$$

the basis we need to take for the ROM space is

$$\boldsymbol{\Phi} = \boldsymbol{M}^{-1/2}\hat{\boldsymbol{\Phi}} \quad \text{satisfying} \quad \boldsymbol{\Phi}^T\boldsymbol{M}\boldsymbol{\Phi} = \boldsymbol{I}_{(R)}.$$

Using the above described POD, we can now construct the ROM space:

$$\mathcal{Y}_\phi = \operatorname{span}\{\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_R\},$$

with $R = \dim(\mathcal{Y}_\phi) \ll \dim(\mathcal{Y}_h)$.

The VMS approach for ROMs is exactly the same as the one described for the FOM with only one key difference: in the case of a VMS-ROM, instead of approximating the unknowns in $\mathcal{Y}_h$, they are approximated in $\mathcal{Y}_\phi$. Thus, we use a FOM-ROM consistent formulation. Consistent formulations have been shown to perform better in [55,56].

In the case of ROMs, using VMS we have the following splitting:

$$\mathcal{Y} = \mathcal{Y}_h \oplus \mathcal{Y}' = \mathcal{Y}_\phi \oplus \mathcal{Y}'', \tag{13}$$

where $\mathcal{Y}''$ represents the SGS space of the ROM space. Assume there exists a POD basis of $\mathcal{Y}_h$, with $H = \dim(\mathcal{Y}_h)$. As indicated earlier, $H = np$ minus the number of velocity degrees of freedom that have prescribed Dirichlet boundary conditions. Obviously, this is possible if there are at least H linearly independent snapshots. In this case, we may write

$$\mathcal{Y}_h = \operatorname{span}\{\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_H\}.$$

Since the POD employed provides L^2 functional orthogonal basis, this in combination with the use of orthogonal SGSs allows us to write

$$\mathcal{Y}'' = \operatorname{span}\{\boldsymbol{\phi}_{R+1}, \dots, \boldsymbol{\phi}_H\} \oplus \mathcal{Y}' = \widetilde{\mathcal{Y}} \oplus \mathcal{Y}'.$$

With the above description, the final problem to be solved reads: for $n = 0, \dots, nt-1$, find $\boldsymbol{y}_\phi^{n+1} = [\boldsymbol{u}_\phi^{n+1}, p_\phi^{n+1}] \in \mathcal{Y}_\phi$ such that

$$\begin{aligned} &\langle \delta_t \boldsymbol{u}_\phi, \boldsymbol{v}_\phi \rangle + B(\boldsymbol{u}_\phi; \boldsymbol{y}_\phi, \boldsymbol{z}_\phi) + \sum_K \langle \boldsymbol{u}''_K, -\nu\Delta\boldsymbol{v}_\phi - \boldsymbol{u}_\phi \cdot \nabla\boldsymbol{v}_\phi - \nabla q_\phi \rangle_K \\ &\quad - \sum_K \langle p''_K, \nabla\cdot\boldsymbol{v}_\phi \rangle_K + \sum_E \langle \boldsymbol{u}''_E, [\![\boldsymbol{n} \cdot (\nu\nabla\boldsymbol{v}_\phi + q_\phi \boldsymbol{I}_{(d)})]\!] \rangle_E = L(\boldsymbol{z}_\phi) \quad \forall \boldsymbol{z}_\phi \in \mathcal{Y}_\phi, \end{aligned} \tag{14}$$

where $\boldsymbol{u}''_K$, p''_K and $\boldsymbol{u}''_E$ are the solutions of

$$\delta_t \boldsymbol{u}''_K + \tau_{1,K}^{-1}\boldsymbol{u}''_K = \Pi^\perp(\boldsymbol{f} + \nu\Delta\boldsymbol{u}_\phi - \boldsymbol{u}_\phi \cdot \nabla\boldsymbol{u}_\phi - \nabla p_\phi)|_K,$$

$$\tau_{2,K}^{-1} p_K'' = -\Pi^{\perp}(\nabla \cdot \boldsymbol{u}_\phi)|_K,$$
$$\tau_E^{-1} \boldsymbol{u}_E'' = -[\![\boldsymbol{n} \cdot (\nu \nabla \boldsymbol{u}_\phi - p_\phi \boldsymbol{I}_{(d)})]\!]_E.$$

The time level superscript has been omitted to lighten the notation.

It is important to note that we are relying on the fact that the ROM problem is also based on a FE mesh and the ROM basis can be written as piece-wise polynomial functions based on FE interpolation functions. This also justifies using the same stabilization parameters of the FOM for the ROM as well [20].

3. A posteriori sub-grid scale modeling

3.1. Fully discrete system

The spatial and temporal discretization of the FOM problem given by Eq. (6) results, for each time step, in a matrix system of the form:

$$\boldsymbol{A}_h(\boldsymbol{Y}_h)\boldsymbol{Y}_h = \boldsymbol{R}_h, \tag{15}$$

where $\boldsymbol{A}_h \in \mathbb{R}^{np \times np}$ is the system matrix, $\boldsymbol{Y}_h \in \mathbb{R}^{np}$ is the array of unknowns and $\boldsymbol{R}_h \in \mathbb{R}^{np}$ is the RHS, which takes into account the contributions of the previous values of $\boldsymbol{Y}_h$. For simplicity, the dependence of $\boldsymbol{A}_h$ on $\boldsymbol{Y}_h$ will be omitted from the notation from here on, except when it is needed for clarity. We assume that Dirichlet conditions are also incorporated in matrix $\boldsymbol{A}_h$; otherwise the dimension np of the system should be reduced to H.

Before moving into the fully discrete ROM system, let us first provide a few definitions and notations related to POD-ROM used in this work. As described in Section 2.3, we decompose the unknown into mean $\overline{\boldsymbol{Y}}_h$ and fluctuating part and then represent the fluctuating part only using the reduced order basis. Thus, we can approximate vectors $\boldsymbol{Y}_h$ as:

$$\boldsymbol{Y}_h \approx \boldsymbol{\Phi}\boldsymbol{Y}_\phi + \overline{\boldsymbol{Y}}_h, \tag{16}$$

where $\boldsymbol{Y}_\phi \in \mathbb{R}^R$, called *ROM coefficients*, are the components of the approximation of $\boldsymbol{Y}_h - \overline{\boldsymbol{Y}}_h$ in $\mathcal{V}_\phi$. Now, let us define some operators for convenience. Firstly, we define the mapping $\mathcal{P}_{h\phi} : \mathbb{R}^{np} \to \mathbb{R}^R$ as

$$\mathcal{P}_{h\phi}(\boldsymbol{Y}_h) := \boldsymbol{\Phi}^T \boldsymbol{M}(\boldsymbol{Y}_h - \overline{\boldsymbol{Y}}_h),$$

which is the algebraic version of the L^2 projection from $\mathcal{V}_h$ (subtracting the mean) onto $\mathcal{V}_\phi$. Secondly, we define the operator $\mathcal{P}_{\phi h} : \mathbb{R}^R \to \mathbb{R}^{np}$ as:

$$\mathcal{P}_{\phi h}(\boldsymbol{Y}_\phi) := \boldsymbol{\Phi}\boldsymbol{Y}_\phi + \overline{\boldsymbol{Y}}_h.$$

Finally, we define the third operator $\mathcal{P}_{hh} : \mathbb{R}^{np} \to \mathbb{R}^{np}$ as

$$\mathcal{P}_{hh}(\boldsymbol{Y}_h) := \mathcal{P}_{\phi h}(\mathcal{P}_{h\phi}(\boldsymbol{Y}_h)).$$

Note that, in general, $\boldsymbol{Y}_h \neq \mathcal{P}_{hh}(\boldsymbol{Y}_h)$, because we lose information when moving form the high-dimensional FOM space to the low-dimensional ROM space using $\mathcal{P}_{h\phi}$ and this information is not recovered when we come back to the FOM space using $\mathcal{P}_{\phi h}$. By construction, $\mathcal{P}_{hh}$ is a projection ($\mathcal{P}_{hh}^2 = \mathcal{P}_{hh}$), and we shall also refer to $\mathcal{P}_{h\phi}$ as a projection, whereas we will call $\mathcal{P}_{\phi h}$ an extension (embedding) operator.

Now, let us introduce decomposition (16) in Eq. (15) and take the FOM test function in the ROM subspace; we get

$$\boldsymbol{\Phi}^T \boldsymbol{A}_h \boldsymbol{\Phi}\boldsymbol{Y}_\phi = \boldsymbol{\Phi}^T \boldsymbol{R}_h - \boldsymbol{\Phi}^T \boldsymbol{A}_h \overline{\boldsymbol{Y}}_h. \tag{17}$$

Using the notations

$$\boldsymbol{A}_\phi := \boldsymbol{\Phi}^T \boldsymbol{A}_h \boldsymbol{\Phi} \quad \in \mathbb{R}^{R \times R},$$
$$\boldsymbol{R}_\phi := \boldsymbol{\Phi}^T (\boldsymbol{R}_h - \boldsymbol{A}_h \overline{\boldsymbol{Y}}_h) \quad \in \mathbb{R}^R,$$

Eq. (17) can be written as:

$$\boldsymbol{A}_\phi \boldsymbol{Y}_\phi = \boldsymbol{R}_\phi. \tag{18}$$

As we have used the same variational formulation for both, the FOM and the ROM, Eq. (18) is the matrix system corresponding to problem (14). Since, $R \ll np$, system (18) can be solved at a fraction of computational cost as compared to the FOM system (15). The computational cost of building the system matrix (18) for ROM is still of the order of that of the FOM, because of the dependence of this matrix on the unknown. Hyper-reduction techniques can be used to reduce this cost [57–59]. However, no hyper-reduction techniques were pursued in this work, as the ideas we wish to present can be applied independently of the hyper-reduction technique employed.

Eq. (18) represents the Galerkin projection of the FOM system onto the ROM space. In the case of non-linear problems, Petrov–Galerkin (PG) projection has proved to be more robust in terms of obtaining convergence of non-linear iterations [60]. In the case of PG, the final system is

$$\boldsymbol{A}_{\phi,PG}\boldsymbol{Y}_{\phi} = \boldsymbol{R}_{\phi,PG}, \tag{19}$$

where now

$$\begin{aligned}\boldsymbol{A}_{\phi,PG} &:= \boldsymbol{\Phi}^T \boldsymbol{A}_h^T \boldsymbol{A}_h \boldsymbol{\Phi} \quad \in \mathbb{R}^{R\times R},\\ \boldsymbol{R}_{\phi,PG} &:= \boldsymbol{\Phi}^T \boldsymbol{A}_h^T (\boldsymbol{R}_h - \boldsymbol{A}_h \overline{\boldsymbol{Y}}_h) \quad \in \mathbb{R}^{R}.\end{aligned}$$

The techniques developed in this work are valid for both, Galerkin (18) and PG (19) systems. However, for simplicity, we will explain the methodology for the Galerkin system (18).

3.2. ROM sub-grid scales in the FOM space

The main contribution of this paper consists in modifying system (18) so as to get a higher accuracy for the ROM solution. This will be done by introducing the concept of ROM-SGSs in the FOM space. These SGSs will be the difference between the ROM solution and an optimal representation of a FOM solution in the ROM space. Therefore, we have to start by defining what do we consider this optimal representation:

Assumption 1. The projection $\mathcal{P}_{h\phi}(\boldsymbol{Y}_h)$ is the best possible representation of the FOM solution $\boldsymbol{Y}_h$ in the ROM space, and $\mathcal{P}_{hh}(\boldsymbol{Y}_h)$ its best extension back to the FOM space. □

Using this assumption, we can write the unknown $\boldsymbol{Y}_h$ as

$$\boldsymbol{Y}_h = \mathcal{P}_{hh}(\boldsymbol{Y}_h) + \widetilde{\boldsymbol{Y}}_h, \tag{20}$$

where $\mathcal{P}_{hh}(\boldsymbol{Y}_h)$ is the part of $\boldsymbol{Y}_h$ that the ROM space can capture and $\widetilde{\boldsymbol{Y}}_h \in \widetilde{\mathcal{Y}}$ are the SGSs, which the ROM space cannot capture. Since we have L^2 orthogonal bases, the SGS space $\widetilde{\mathcal{Y}}$ is the L^2 orthogonal complement $\mathcal{Y}_{\phi}^{\perp}$ of $\mathcal{Y}_{\phi}$ in $\mathcal{Y}_h$, i.e.

$$\mathcal{Y}_h = \mathcal{Y}_{\phi} \oplus \widetilde{\mathcal{Y}} = \mathcal{Y}_{\phi} \oplus \mathcal{Y}_{\phi}^{\perp}.$$

Note that now in order to model the SGSs using the a posteriori approach, we are looking to complete $\mathcal{Y}_{\phi}$ in $\mathcal{Y}_h$, and not in $\mathcal{Y}$.

Of course we do not have available the FOM solution $\boldsymbol{Y}_h$ for all time steps, but *we do have it for the time steps corresponding to the snapshots*. The key idea of our approach is *to construct a model to approximate the ROM-SGSs in the FOM space at these time steps and then use this model for all time steps.*

Inserting the exact expression for the unknown (20) in Eq. (15) and projecting it onto the ROM space, we get

$$\boldsymbol{A}_{\phi}\mathcal{P}_{h\phi}(\boldsymbol{Y}_h) + \boldsymbol{\Phi}^T \boldsymbol{A}_h \widetilde{\boldsymbol{Y}}_h = \boldsymbol{R}_{\phi}.$$

The term $\boldsymbol{\Phi}^T \boldsymbol{A}_h \widetilde{\boldsymbol{Y}}_h$ represents the (non-linear) interaction of the resolved scales with the unresolved scales. We represent it as

$$\boldsymbol{A}_{\phi}\boldsymbol{Z}_{\phi} = \boldsymbol{\Phi}^T \boldsymbol{A}_h \widetilde{\boldsymbol{Y}}_h, \tag{21}$$

and refer to $\boldsymbol{Z}_{\phi}$ as the *a posteriori SGSs*. Note that this is just a matter of representation, as we could have opted for representing directly $\boldsymbol{D}_{\phi} := \boldsymbol{\Phi}^T \boldsymbol{A}\widetilde{\boldsymbol{Y}}_h$, which was the approach followed in [61] in the context of mesh or time discretization coarsening. It is stressed again that we use SGSs to correct the ROM system and not the solution.

From here onwards, the abbreviation *SGSs* is used exclusively for the a posteriori SGSs $\boldsymbol{Z}_{\phi}$. Fig. 2 shows the FOM solution, its projection onto the ROM space $\mathcal{P}_{h\phi}(\boldsymbol{Y}_h)$, the ROM solution without taking into account the

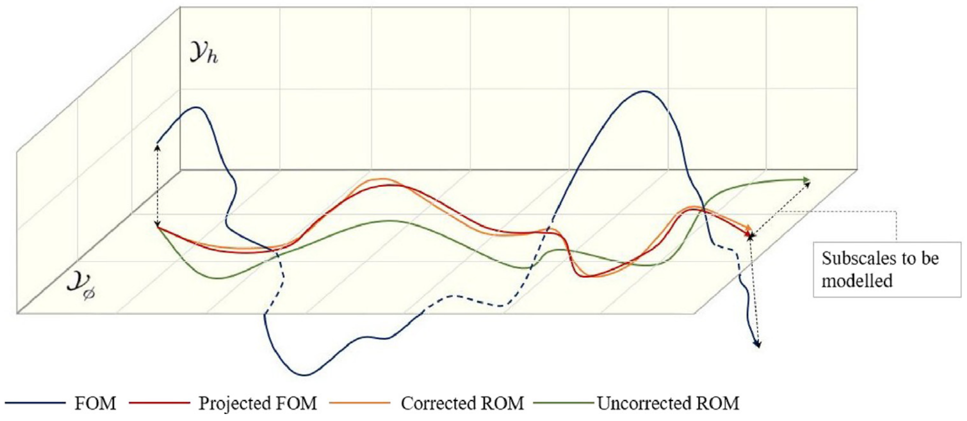

Fig. 2. A representation of the evolution of the FOM, the projection of the FOM on the ROM subspace, the corrected ROM and the uncorrected ROM solutions. The SGSs (error) to be modeled in the a posteriori approach are also shown.

SGSs, called *uncorrected ROM*, and the ROM solution taking into account the SGSs, called *corrected ROM*. The time evolution of these variables is represented as a continuous curve, but we can evaluate them all only at the time instants in which the snapshots have been computed.

The final reduced-order system for time step $n+1$, making now explicit this time step, is

$$\boldsymbol{A}_\phi(\boldsymbol{Y}_\phi^{n+1} + \boldsymbol{Z}_\phi^{n+1}) = \boldsymbol{R}_\phi. \tag{22}$$

At the time steps $n+1$ in which the snapshots are known, $\boldsymbol{Z}_\phi^{n+1}$ could be computed exactly from Eqs. (20) and (21) and the solution to system (22) would be $\boldsymbol{Y}_\phi^{n+1} = \mathcal{P}_{h\phi}(\boldsymbol{Y}_h^{n+1})$, assumed to be optimal. However, this is not possible for all time steps. What we propose is *to construct a model for* $\boldsymbol{Z}_\phi^{n+1}$ *from an ANN, of the form*

$$\boldsymbol{Z}_\phi^{n+1} \approx \boldsymbol{Z}_{\text{ann}}^{n+1} = \mathcal{F}_{\text{ann}}^{n+1}(\boldsymbol{Y}_\phi), \tag{23}$$

where the ANN $\mathcal{F}_{\text{ann}}$ will be trained by imposing that, at the time steps $n+1$ corresponding to the snapshots, $\boldsymbol{Z}_{\text{ann}}^{n+1}$ be as close as possible to the exact SGSs $\boldsymbol{Z}_\phi^{n+1}$. Note that the ANN depends only on the ROM solution. This dependence will be further elaborated later on.

An important remark concerning the possibility to construct the exact SGSs $\boldsymbol{Z}_\phi^{n+1}$ is in order. First, as we have said, $n+1 \in \{n_1, \ldots, n_s\}$. But, since system (22) involves also an approximation to the time derivative, this needs to be computed also from previous projected snapshots. For example, using first order backward differences (BDF1), if $n+1 = n_{k+1}$ we need to approximate the time derivative in the FOM space using also the snapshot at n_k, or to store directly not only the snapshots, but also their approximate time derivatives. The latter is precisely the strategy we have adopted in our implementation when the time derivative of the unknown is required. For the former, the time step size to be used is $\delta t_k = (n_{k+1} - n_k)\delta t$. Note that we may consider $n_0 = 0$ and take the initial condition as a known snapshot. The extension of this procedure to higher order multi-step schemes is straightforward, although for the sake of conciseness we shall restrict the exposition to BDF1.

4. Modeling a posteriori sub-grid scales with artificial neural networks

The objective of this work is to present a correction for ROMs and to demonstrate that ANNs can be used to model the correction. The choices made regarding the numerical methodology and ANNs do not affect the main idea of this work. We were able to achieve desired results using particular implementations but we do not claim them to be optimal. Wherever possible, we discuss our choices regarding ANNs, however, we do not want to emphasize them too much so that it obscures the main message of this work.

In this work, we use Feedforward Neural Networks (FNNs). These are among the simplest ANNs and thus used here to keep the element of ambiguity associated to ANNs to a minimum. Furthermore, an effort is made to keep the architecture and inputs of FNNs to be as simple as possible as well, increasing the complexity only when required. We use these ANN mostly as black boxes.

In this work, all of the implementation related to ANNs is done using Keras API [62], with TensorFlow [63] as the backend.

4.1. Data gathering for ANN training

The data to construct the ANN we propose are the exact snapshots. As it has been said, they can be computed from Eqs. (20) and (21). Let us explain now in detail this calculation, making explicit the dependence of the system matrix and RHS on the snapshots.

Let n be such that $n+1 = n_{k+1} \in \{n_1, \dots, n_s\}$, i.e., we have at our disposal the snapshot $\boldsymbol{S}_h^{n+1}$. As stated in the main assumption, the target of the corrected ROM is $\boldsymbol{Y}_\phi^{n+1} \approx \mathcal{P}_{h\phi}(\boldsymbol{S}_h^{n+1}) =: \boldsymbol{S}_\phi^{n+1}$, i.e, $\boldsymbol{Z}_\phi^{n+1}$ has to be such that the solution to

$$\boldsymbol{A}_{\phi,S}(\boldsymbol{Y}_\phi^{n+1} + \boldsymbol{Z}_\phi^{n+1}) = \boldsymbol{R}_{\phi,S} \tag{24}$$

is $\boldsymbol{Y}_\phi^{n+1} = \boldsymbol{S}_\phi^{n+1}$, where

$$\begin{aligned}\boldsymbol{A}_{\phi,S} &:= \boldsymbol{\Phi}^T \boldsymbol{A}(\mathcal{P}_{hh}(\boldsymbol{S}_h^{n+1}))\boldsymbol{\Phi},\\ \boldsymbol{R}_{\phi,S} &:= \boldsymbol{\Phi}^T[\boldsymbol{R}(\mathcal{P}_{hh}(\boldsymbol{S}_h^{n_k})) - \boldsymbol{A}(\mathcal{P}_{hh}(\boldsymbol{S}_h^{n+1}))\overline{\boldsymbol{Y}}_h].\end{aligned}$$

These are the matrix and RHS computed with the best possible representation of the snapshots in the ROM space transferred to the FOM space (we have built $\boldsymbol{A}$ using $\mathcal{P}_{hh}(\boldsymbol{Y}_h^{n+1})$ instead of $\boldsymbol{Y}_h^{n+1}$), and taking as initial condition $\mathcal{P}_{hh}(\boldsymbol{S}_h^{n_k})$. The time step for the calculation of the time derivative is δt_k.

Let us see a particular implementation of how to compute $\boldsymbol{Z}_\phi^{n+1}$ so that the solution to Eq. (24) is $\boldsymbol{Y}_\phi^{n+1} = \boldsymbol{S}_\phi^{n+1}$. For that, let us consider the system

$$\boldsymbol{A}_{\phi,S}\boldsymbol{S}_{\phi,\text{pred}}^{n+1} = \boldsymbol{R}_{\phi,S}. \tag{25}$$

From Eq. (24), imposing that $\boldsymbol{Y}_\phi^{n+1} = \boldsymbol{S}_\phi^{n+1}$, we have:

$$\begin{aligned}\boldsymbol{Z}_\phi^{n+1} &= \boldsymbol{A}_{\phi,S}^{-1}(\boldsymbol{R}_{\phi,S} - \boldsymbol{A}_{\phi,S}\boldsymbol{S}_\phi^{n+1})\\ &= \boldsymbol{S}_{\phi,\text{pred}}^{n+1} - \boldsymbol{S}_\phi^{n+1}\\ &= \boldsymbol{S}_{\phi,\text{pred}}^{n+1} - \mathcal{P}_{h\phi}(\boldsymbol{S}_h^{n+1}).\end{aligned} \tag{26}$$

Thus, we need to gather $\mathcal{P}_{h\phi}(\boldsymbol{S}_h^{n+1})$ by projecting onto the ROM space the snapshots solution of (15) for different time steps, and we need to compute $\boldsymbol{S}_{\phi,\text{pred}}^{n+1}$ by solving (25).

Let us denote the projected snapshot matrix as $\boldsymbol{S}_\phi = [\mathcal{P}_{h\phi}(\boldsymbol{S}_h^{n_1}), \dots, \mathcal{P}_{h\phi}(\boldsymbol{S}_h^{n_s})] \in \mathbb{R}^{R\times n_s}$, the uncorrected ROM solution matrix as $\boldsymbol{S}_{\phi,\text{pred}} = [\boldsymbol{S}_{\phi,\text{pred}}^{n_1}, \dots, \boldsymbol{S}_{\phi,\text{pred}}^{n_s}] \in \mathbb{R}^{R\times n_s}$ and the target SGS matrix as $\boldsymbol{Z}_\phi = [\boldsymbol{Z}_\phi^{n_1}, \dots, \boldsymbol{Z}_\phi^{n_s}] \in \mathbb{R}^{R\times n_s}$. Then, Eq. (26) can be written as

$$\boldsymbol{Z}_\phi = \boldsymbol{S}_{\phi,\text{pred}} - \boldsymbol{S}_\phi. \tag{27}$$

We have therefore built a mapping:

$$\boldsymbol{S}_h^{n+1} \mapsto \boldsymbol{S}_\phi^{n+1} \mapsto \boldsymbol{Z}_\phi^{n+1}, \quad \forall n+1 = n_{k+1} \in \{n_1, \dots, n_s\},$$

i.e., for each snapshot we obtain the target (exact) SGS. Equivalently, we have the mapping

$$\boldsymbol{S}_h \mapsto \boldsymbol{S}_\phi \mapsto \boldsymbol{Z}_\phi, \quad \text{with } \boldsymbol{S}_h \in \mathbb{R}^{np\times n_s},\ \boldsymbol{S}_\phi, \boldsymbol{Z}_\phi \in \mathbb{R}^{R\times n_s}.$$

This will serve to construct the approximation indicated in (23) using an ANN, that is to say, $\{\boldsymbol{S}_\phi, \boldsymbol{Z}_\phi\}$ will be taken as the *training set* of the ANN.

Remark 1. In this work we correct the system/model and not the solution. Since our correction comes out to be equal to the difference between two solutions, it might seem valid to find the uncorrected solution by solving the uncorrected system (18) and then subtracting the subscales from it to get the correct solution as per Eq. (26) during the execution phase, i.e.,

1. Solve (18) to get $\boldsymbol{Y}_{\phi,\mathrm{pred}}^{n+1}$
2. Find $\boldsymbol{Y}_{\phi}^{n+1} = \boldsymbol{Y}_{\phi,\mathrm{pred}}^{n+1} - \boldsymbol{Z}_{\phi}^{n+1}$ as per Eq. (26).

However, the solution to the uncorrected system (18) is not $\boldsymbol{Y}_{\phi,\mathrm{pred}}^{n+1} = \boldsymbol{S}_{\phi,\mathrm{pred}}^{n+1}$. During training, we obtain $\boldsymbol{S}_{\phi,\mathrm{pred}}^{n+1}$ from Eq. (25) which uses the *correct nonlinear operator* $\boldsymbol{A}_{\phi,S}(\mathcal{P}_{hh}(\boldsymbol{S}_h^{n+1}))$ instead of $\boldsymbol{A}_{\phi}(\boldsymbol{Y}_{\phi,\mathrm{pred}}^{n+1})$. Hence, $\boldsymbol{S}_{\phi,\mathrm{pred}}^{n+1}$ is not the solution to the *uncorrected* ROM system (18). Consequently, in the execution phase, the correction must be included in every iteration, thus updating the nonlinear operators using the corrected solution so that we arrive at system (24) near convergence. Then the solution is $\boldsymbol{Y}_{\phi}^{n+1} \approx \boldsymbol{S}_{\phi}^{n+1}$, as desired.

4.2. Functional dependence of the ANN

According to (23), $\boldsymbol{Z}_{\mathrm{ann}}^{n+1} = \mathcal{F}_{\mathrm{ann}}^{n+1}(\boldsymbol{Y}_{\phi})$ depends generically of $\boldsymbol{Y}_{\phi}$; on purpose, we have not specified the time level of the argument. In fact, we have observed that when designing $\boldsymbol{Z}_{\mathrm{ann}}^{n+1}$ it is very important that $\mathcal{F}_{\mathrm{ann}}$ *can accept the values of* $\boldsymbol{Y}_{\phi}$ *at different times steps and the values of time derivatives of* $\boldsymbol{Y}_{\phi}$ *of different order as input*. In particular, we have found this necessary when considering parametric cases (see Section 5). In our case, in which we have a first order derivative in time and using the BDF1 scheme, we will construct the ANN at time step $n+1$ as a function of $\boldsymbol{Y}_{\phi}^{n+1}$ and, if needed, of the approximation of the derivative of this variable at $n+1$, which we shall denote as $\dot{\boldsymbol{Y}}_{\phi}^{n+1}$. Thus,

$$\boldsymbol{Z}_{\mathrm{ann}}^{n+1} = \mathcal{F}_{\mathrm{ann}}^{n+1}(\boldsymbol{Y}_{\phi}) = \mathcal{F}_{\mathrm{ann}}(\boldsymbol{Y}_{\phi}^{n+1}, \dot{\boldsymbol{Y}}_{\phi}^{n+1}). \tag{28}$$

An additional decision that needs to be made here is whether to predict the SGSs components $Z_{\mathrm{ann},i}$, $i = 1, \dots, R$, associated to each $Y_{\phi,i}$, $i = 1, \dots, R$, collectively using a single ANN $\mathcal{F}_{\mathrm{ann}}$, or having a separate ANN $\mathcal{F}_{\mathrm{ann},i}$ for each SGS component. The actual choices made in this regard will be discussed in Section 5, as this depends on the problem at hand. The goal is to use the simplest ANN that can learn the mapping: inputs $\mapsto$ outputs.

The implicit dependence given by Eq. (28) introduces the need for some sort of linearization when the SGSs are introduced in the ROM equations. This can be treated using a fixed point strategy, as adopted in this work, or a Newton-like linearization, as it is suggested in [61].

Regarding the dependence of the SGS on the time derivative of the ROM solution, it can be considered a more general dependence than just taking the ROM solution. However, even if we do not have an evolution equation for the ROM scales and the SGSs, should we have started from such equations, we would have obtained such a dependence, using for example Mori–Zwanzig formalism (see, e.g., [64,65] for applications of this formalism in the FE and ROM contexts, respectively). In fact, the exact dependence would be on the whole history of ROM scales, but the most relevant terms are known to be those of the previous time steps, contained in their temporal derivate. This provides a heuristic justification for the functional dependence indicated in Eq. (28). The relevant number of previous timesteps, or the derivative order, is problem-dependent. For the cases considered in this work, including just the first derivative proved to be sufficient.

4.3. Normalization of the data

The inputs and outputs of the training phase will be normalized to improve the training performance. Two commonly used normalization techniques are z-score based and min–max based. Let $\boldsymbol{X}_{\phi} \in \mathbb{R}^{ni \times s}$, be the collection of training inputs, with ni being the number of input components to the ANN. In the case given by Eq. (28) we would have that $\boldsymbol{X}_{\phi}^{n_j} = [\boldsymbol{S}_{\phi}^{n_j}, \dot{\boldsymbol{S}}_{\phi}^{n_j}]$, $j = 1, \dots, s$, and $ni = 2R$.

A min–max normalization is used to get the values in the range of $0 - 1$. The use of min–max normalization in our case can be justified by the fact that we will be using the ReLU activation function, which offers quicker training for regression problems. Since it zeros all the negative values, the normalization used should represent the inputs in terms of positive numbers to prevent any loss of information, thus leading to our use of min–max normalization in the range of $0 - 1$. To do this, first the minimum and maximum values for each input and output are found as

$$X_{\min,i} = \min_{j=1,\dots,s} X_{\phi,i}^{n_j}, \quad X_{\max,i} = \max_{j=1,\dots,s} X_{\phi,i}^{n_j}, \quad i = 1, \dots, ni,$$

$$Z_{\min,i} = \min_{j=1,\dots,s} Z_{\phi,i}^{n_j}, \quad Z_{\max,i} = \max_{j=1,\dots,s} Z_{\phi,i}^{n_j}, \quad i = 1, \dots, R,$$

and then the values are normalized by applying the transformations

$$X_{\phi,i}^{n_j} \mapsto \frac{X_{\phi,i}^{n_j} - X_{\min,i}}{X_{\max,i} - X_{\min,i}}, \tag{29}$$

$$Z_{\phi,i}^{n_j} \mapsto \frac{Z_{\phi,i}^{n_j} - Z_{\min,i}}{Z_{\max,i} - Z_{\min,i}}. \tag{30}$$

Transformations (29) are applied also to the validation and testing inputs but using the min–max value found for the training set. To denormalize the ANN outputs, we use the inverse transformation of (30). We assume henceforth that data and outputs are normalized.

4.4. Loss function

The ANN given by Eq. (28) has to be constructed with the goal that it approximates as closely as possible the exact SGSs given by Eq. (26) (or its matrix version given by Eq. (27)) when the inputs are the snapshots. Therefore, the ANN is trained by minimizing the loss function which is a sum of mean squared error (MSE) in the prediction of SGSs and the L^2 regularization term, i.e., the following problem is solved:

$$\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \left(\frac{1}{s} \sum_{i=1}^{s} \| \boldsymbol{Z}_{\text{ann}}^{n_i} - \boldsymbol{Z}_{\phi}^{n_i} \|^2 + \lambda \sum_{j=1}^{nw} W_j^2 \right) \tag{31}$$

where $\boldsymbol{\theta}$ consists of the weights $\boldsymbol{W}$ and biases of the ANN, λ is the regularization parameter of the L^2 regularization and nw is the total number of weights. The regularization helps in preventing the ANN from over-fitting the training data. Adam stochastic optimizer [66] is used to solve (31) to obtain the optimal weights and biases. We use mini-batches of size $1 < nb < s$ equal to 2^n, where n is an integer, to speedup the training process. The training is stopped when the convergence with respect to specified tolerance is achieved, i.e., $\mathcal{J} <$ `tol-train`, or when the training has lasted the prescribed number of steps `max-epochs`. The training speed is controlled by a constant learning rate η. In the numerical examples, the hyper-parameters are tuned using a trial and error method. The training on a given data is performed multiple times to minimize the effect of random weight initialization. The trained model which gives the best performance for the validation set is used.

4.5. Testing (execution) phase

The sequence of operations for a time step of corrected ROM in the testing phase is as follows:

$$\boldsymbol{Y}_{\phi}^{n} \xrightarrow[\text{FOM space}]{\text{extend to}} \mathcal{P}_{\phi h}(\boldsymbol{Y}_{\phi}^{n}) \xrightarrow[\text{system}]{\text{build matrix}} \boldsymbol{A}_{\phi}(\boldsymbol{Y}_{\phi}^{n+1} + \boldsymbol{Z}_{\text{ann}}^{n+1}) = \boldsymbol{R}_{\phi} \xrightarrow{\text{solve}} \boldsymbol{Y}_{\phi}^{n+1}$$

where

$$\begin{aligned}
\boldsymbol{A}_{\phi} &= \boldsymbol{\Phi}^T \boldsymbol{A}(\mathcal{P}_{\phi h}(\boldsymbol{Y}_{\phi}^{n+1}))\boldsymbol{\Phi}, \\
\boldsymbol{R}_{\phi} &= \boldsymbol{\Phi}^T [\boldsymbol{R}(\mathcal{P}_{\phi h}(\boldsymbol{Y}_{\phi}^{n})) - \boldsymbol{A}(\mathcal{P}_{\phi h}(\boldsymbol{Y}_{\phi}^{n+1}))\overline{\boldsymbol{Y}}_h], \\
\boldsymbol{Z}_{\text{ann}}^{n+1} &= \mathcal{F}_{\text{ann}}(\boldsymbol{Y}_{\phi}^{n+1}, \dot{\boldsymbol{Y}}_{\phi}^{n+1}).
\end{aligned}$$

Note that the problem is nonlinear, both because of the dependence of matrix $\boldsymbol{A}$ on the unknown and because of the nonlinearity of the ANN.

The training and testing processes are outlined in Algorithm 1.

5. Numerical examples

The numerical results are performed for non-parametric and parametric cases. The non-parametric cases involve the flow over a cylinder, over a backward facing step and over three cylinders in a triangular arrangement. The parametric analysis is carried out for the flow over a cylinder only, with the Reynolds number Re serving as the parameter. It will be assumed that the snapshots are known for consecutive time steps $1, 2, \dots, s$.

The performance of ANNs is evaluated in the following scenarios:

Algorithm 1 Training and testing algorithm

```
Training
 1: Run the FOM for $nt$ time steps and gather $s$ solutions (snapshots), storing them in $\boldsymbol{S}_h$
 2: Use the POD approach explained in section Section 2.3 to find the basis matrix $\boldsymbol{\Phi}$
 3: Project the FOM snapshot matrix $\boldsymbol{S}_h$ using $\mathcal{P}_{h\phi}(\boldsymbol{S}_h)$ and store it in the projected snapshot matrix $\boldsymbol{S}_\phi$
 4: Solve the uncorrected system (25) and store the results in matrix $\boldsymbol{S}_{\phi,\text{pred}}$
 5: Find the SGSs matrix $\boldsymbol{Z}_\phi$ using Eq. (27)
 6: Normalize the training data $\{\boldsymbol{S}_\phi, \dot{\boldsymbol{S}}_\phi, \boldsymbol{Z}_\phi\}$ using transformations (29)-(30)
 7: $j = 0$
 8: while $\mathcal{J} \geq$ tol-train and $j \leq$ max-epochs do
 9:     Solve the optimization problem (31) to find the weight and bias matrix $\boldsymbol{\theta}$
10:     $j = j + 1$
11: end while
12: Save the trained ANNs and the normalization data
Testing
 1: Load the saved ANNs and the normalization data
 2: for $n = 0; n < nt; n + 1$ do                                        ▷ time loop
 3:     for $k = 0; k < k_{\max}; k + 1$ do                             ▷ nonlinear iteration loop
 4:         Compute the normalized $\boldsymbol{Z}_{\text{ann}}$ based on the inputs, normalized using transformations (29)
 5:         Denormalize $\boldsymbol{Z}_{\text{ann}}$ using the inverse of transformation (30)
 6:         Solve a non-linear iteration of ROM system (22) using $\boldsymbol{Z}_{\text{ann}}$
 7:     end for
 8: end for
```

- **Reconstruction evaluation:** The ANN is trained on the data obtained on the time interval $[0, T_1]$ and the parameter value Re, and then used for prediction in the same time interval $[0, T_1]$ and for the same parameter value Re. Note that even for the reconstruction phase, the input values to the ANN for the training and testing phases do vary. During the training phase, the FOM projection of the snapshots $\mathcal{P}_{h\phi}(\boldsymbol{S}_h)$ (and time derivatives if needed) is used as the input, whereas during the testing phase the ROM solution $\boldsymbol{Y}_\phi$ (and time derivatives if needed), close enough but not equal to $\mathcal{P}_{h\phi}(\boldsymbol{S}_h)$, is provided as the input. Hence, the reconstruction phase is not equivalent to evaluating the performance of a trained ANN on the training data itself.
- **Temporal evaluation:** The ANN is trained on the data obtained on the time interval $[0, T_1]$ and the parameter value Re, and then used for prediction in the extended time interval $[0, T_2]$, where $T_2 > T_1$. The parameter value is still the same Re. Both periodic and non-periodic cases are considered. For periodic cases, periodicity is the inherent characteristic of the desired solution i.e. we can only have a solution if the state of periodicity is reached. Though the original flow is periodic, it will be seen later that the uncorrected ROM is unable to maintain or achieve the periodicity, whereas, the corrected ROM maintains it in most of the cases. Hence, in general, the uncorrected ROM would be unable to achieve a state that could be regarded as the solution. This motivates using temporal evaluation criteria for periodic solutions as well.
- **Parametric evaluation:** The ANN is trained on the data obtained on the time interval $[0, T_1]$ and the parameter values $\text{Re} = \{\text{Re}_1, \ldots, \text{Re}_{nv}\}$, with Re_1 being the minimum and Re_{nv} being the maximum value, and then used for prediction in the same time interval $[0, T_1]$ but for a different parameter value Re. Both, interpolation and extrapolation cases are analyzed. For the interpolation case, $\text{Re} \in [\text{Re}_1, \text{Re}_{nv}]$, whereas for the extrapolation case $\text{Re} \notin [\text{Re}_1, \text{Re}_{nv}]$.
- **Temporal and Parametric evaluation:** The ANN is trained on the data obtained on the time interval $[0, T_1]$ and the parameter values $\text{Re} = \{\text{Re}_1, \ldots, \text{Re}_{nv}\}$, and then used for prediction in the extended time interval $[0, T_2]$, where $T_2 > T_1$, and for a different parameter value Re. Again, this is done for both the parametric interpolation and the extrapolation phases.

Note that in the interpolation and extrapolation of parametric cases, only ANN based SGSs are interpolated and extrapolated, respectively. The basis functions are based on the value of the testing Re, as this work focuses on evaluating the performance of the ANN and not in the way bases are interpolated. Doing so, other sources of errors are eliminated. For all the examples, we use the BDF scheme of first order for the time integration and Picard's scheme for linearization. We use Petrov–Galerkin projection to have an improved convergence of non-linearities. The data used for constructing the ROM basis and training the ANNs vary from case to case. The variation is introduced to show that the ANNs can work effectively under a variety of scenarios.

Performance evaluation criteria

Accuracy: For each case, we use all or some of the following accuracy evaluation criteria. The qualitative comparison is done by comparing the plots of the time evolution of the problem unknowns, e.g., pressure p and velocity magnitude u. Additionally, we perform a quantitative analysis using a root-mean-square deviation (RMSD) of the FOM and ROM solutions at a point to get an error measure. If Y_{FOM}^{n+1} is a scalar quantity obtained by solving the FOM at time $n+1, n=0,\dots,nt-1$, and Y_{ROM}^{n+1} is the one obtained with the ROM, we have

$$Y_{\text{RMSD}} = \sqrt{\frac{1}{nt}\sum_{n=0}^{nt-1}(Y_{\text{FOM}}^{n+1} - Y_{\text{ROM}}^{n+1})^2}.$$

In addition to the error measurement, we analyze the frequency spectrum of scalar outputs using a Discrete Fourier Transform (DFT).

Computational cost: In addition to accuracy comparison, wall clock times are provided and compared for non-parametric and parametric cases. Computational cost associated with all the representative numerical examples is provided and discussed in Section 5.3.

5.1. Non parametric analysis

ANN. For the non-parametric study, it was found sufficient to approximate the SGSs using $\boldsymbol{Z}_{\text{ann}} = \mathcal{F}_{\text{ann}}(\boldsymbol{Y}_\phi)$, i.e., the mapping could be learned without the time derivatives. For the testing phase, we write system (22) for the $k+1$-th non-linear iteration of a time step as

$$\boldsymbol{A}_\phi(\boldsymbol{Y}_\phi^{(k)})\boldsymbol{Y}_\phi^{(k+1)} = \boldsymbol{R}_\phi - \boldsymbol{A}_\phi(\boldsymbol{Y}_\phi^{(k)})\mathcal{F}_{\text{ann}}(\boldsymbol{Y}_\phi^{(k)}),$$

where (k) now is the iteration counter for our nonlinear problem and the time step counter has been omitted. In the following, if only the time step superscript is used (without parenthesis) it is understood that it corresponds to converged values, whereas if only the iteration superscript is used (with parenthesis) it is understood that it corresponds to the current time step.

It is observed that we use a fixed point linearization scheme and that the correcting term is evaluated in the previous nonlinear iteration. Other options could be explored, but in the numerical examples we have found this one effective.

In the case of transient problems, like the ones considered in this work, for the first iteration of a time step $n+1$ we take $\boldsymbol{Y}_\phi^{n+1,(0)} = \boldsymbol{Y}_\phi^n$, i.e., the converged value of the unknown at the previous time step.

A single ANN is used, which takes all the ROM coefficients as inputs and gives all the SGSs as output:

$$\boldsymbol{Y}_\phi \rightarrow \mathcal{F}_{\text{ann}} \rightarrow \boldsymbol{Z}_{\text{ann}}.$$

For the non-parametric study, this approach worked. However, this is not the case in general, as we will see for the parametric case. As a matter of fact, there is not any theoretically correct approach when it comes to ANNs. Different combinations need to be experimented, and the one that works best is selected. The parameters and hyper-parameters chosen for the ANN in this section are shown in Table 1. The small value of λ in this case corresponds to almost no need of regularization, as the ANN was not over-fitting the training data.

5.1.1. Flow over a cylinder

Computational setting

Consider the two dimensional channel flow over a cylinder. The computational domain is $\Omega =]-4, 12[\times]-4, 4[$ minus the cylinder, with the center of the cylinder located at $(0, 0)$. We take its diameter as $D = 1$. At

Table 1
List of hyper-parameters used in training the ANNs for non-parametric examples.

Variables	Flow over a cylinder	Backward facing step
Number of hidden layers	1	3
Number of neurons in each hidden layer	20	20
Batch size	8	8
`tol-train`	10^{-4}	10^{-4}
`max-epochs`	200	200
Validation data set %	20	20
Regularization parameter λ	10^{-10}	10^{-10}
Learning rate	10^{-3}	10^{-3}

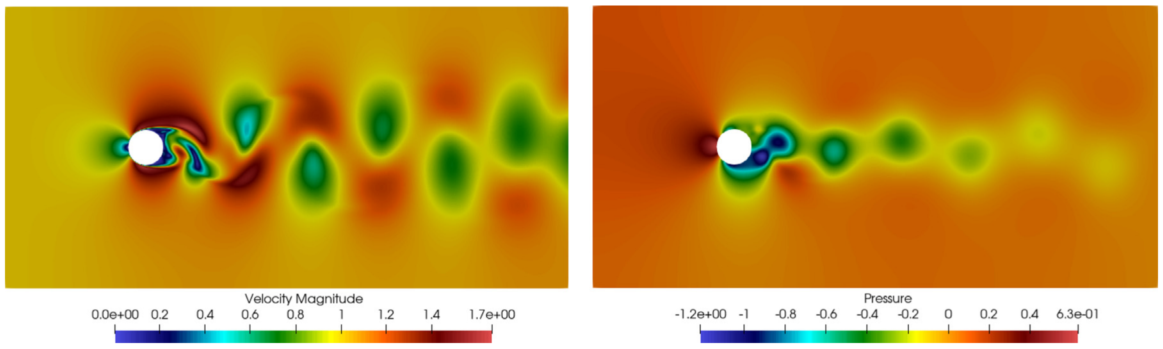

Fig. 3. Contours of the FOM for the flow over a cylinder at Re = 250.

the inlet $x = -4$, we take $u_x = 1$ and $u_y = 0$, whereas the top edge $y = -4$ and the bottom edge $y = 4$ have $u_y = 0$ and u_x is left free. Both velocity components, u_x and u_y, are left free at the outlet $x = 12$. The viscosity ν is selected to achieve the required Re. For the FOM, a symmetric mesh of 39360 bilinear quadrilateral elements and 39760 nodal points is used to discretize the domain. The mesh is refined near the cylinder to better capture the vortex generation. $\delta t = 0.05$ is used as the time step for both the FOM and the ROM. A preliminary run of 2000 time steps is performed to obtain a fully developed flow. We refer to this developed flow as the initial condition for the examples, i.e., $t = 0$ corresponds to the developed flow. Fig. 3 shows velocity and pressure contours of the fully developed flow for visualization. The control point of coordinates $(1, 0.5)$ located immediately behind the cylinder in the vortex generation region is used to compare the values of p and u.

Reconstruction phase
Data gathering for ROM basis and ANN. The FOM solutions for $t \in [0, 15]$, i.e., 300 time steps, are used to calculate the ROM basis. The ROM is also run for 300 time steps to gather data, which, in addition to the data from the FOM, are used to train the ANN. The number of basis to be used is chosen to be $R = 3$ to replicate a highly truncated ROM, such that the ROM space can represent the FOM space with reasonable accuracy but it has enough error so that the improvement introduced by the ANN could be seen.

Results. The performance of the ANN for the reconstruction phase is evaluated for Re = {100, 150, 200, 250, 300, 500, 750, 1000} using the same Re for the training and testing phases. Fig. 4 shows a comparison of p and u at the control point (1,0.5) at Re = 250 for the FOM, the uncorrected ROM and the corrected ROM. It can be seen that the results of the corrected ROM are almost indistinguishable from the FOM, whereas the uncorrected ROM diverges from the FOM results. Fig. 5 shows a comparison of the DFT of p and u for the FOM and the ROMs. Here again, the corrected ROM perfectly captures all the frequencies. As a result, the corrected ROM correctly captures the shape of the time evolution of u in Fig. 4, whereas the uncorrected ROM fails to do so. Table 2 shows that the corrected ROM has *an order of magnitude* less RMSD than the uncorrected ROM.

The RMSD for p and u at different Re are plotted in Fig. 6. An interesting observation is that while the error increases monotonically with Re (more complex flows) for the uncorrected ROM, the same is not true for the

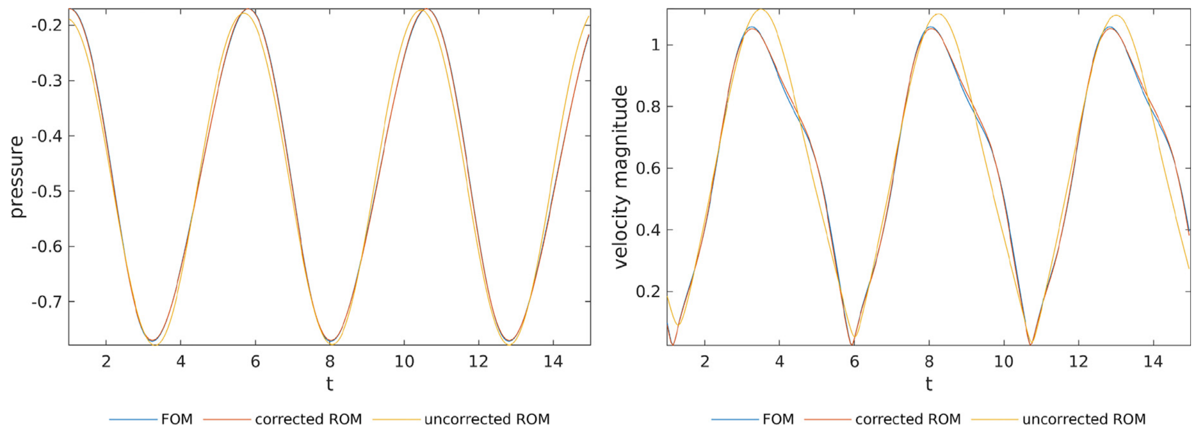

Fig. 4. Comparison between the FOM and the ROMs for the reconstruction phase at Re = 250. Flow over a cylinder.

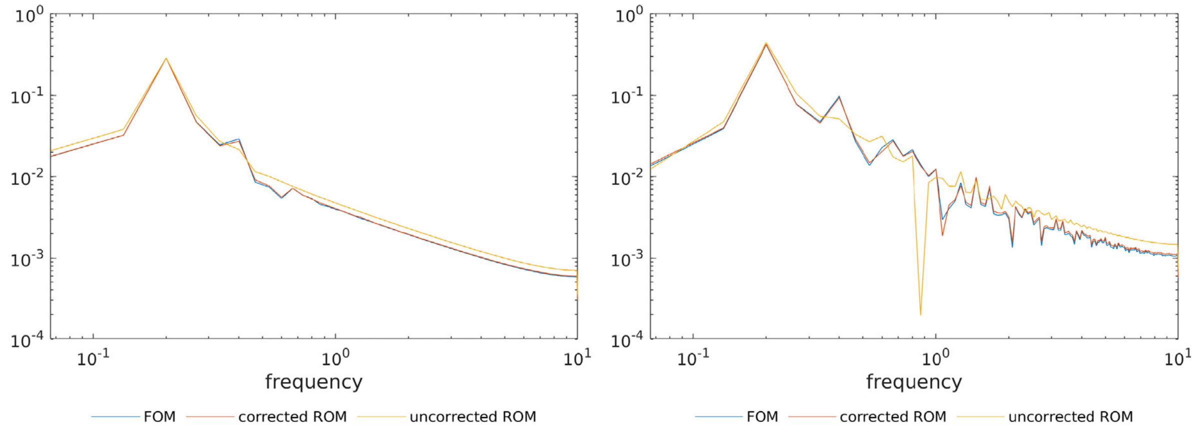

Fig. 5. Comparison of DFT between the FOM and the ROMs for the reconstruction phase at Re = 250. Left: p. Right: u. Flow over a cylinder.

Table 2
RMSD values for the reconstruction phase for p and u at Re = 250.

Re	Pressure		Velocity magnitude	
	Corrected	Uncorrected	Corrected	Uncorrected
250	2.015×10^{-3}	2.449×10^{-2}	8.646×10^{-3}	6.495×10^{-2}

corrected ROM. Although the same hyper-parameters and multiple weight initializations have been used to train the ANN for each Re, the trained ANN has different prediction accuracy in the testing phase for the various tested Re. This explains the non-monotonic error behavior.

We also analyze the behavior of the ROM coefficients and SGSs. Fig. 7 shows the time evolution of the projected FOM coefficients $\mathcal{P}_{h\phi}(\boldsymbol{Y}_h)$, the corrected ROM coefficients $\boldsymbol{Y}_\phi$ and the uncorrected ROM coefficients at Re = 250. It also shows the associated exact SGSs $\boldsymbol{Z}_\phi$ and the ones modeled using the ANN, $\boldsymbol{Z}_{\text{ann}}$. We note that the highest mode $Y_{\phi,3}$ has the highest SGS amplitudes, thus leading to a significant difference in its behavior for the corrected and the uncorrected ROM. In terms of frequencies, $Z_{\phi,3}$ appears to consist of one or two dominant frequencies, whereas

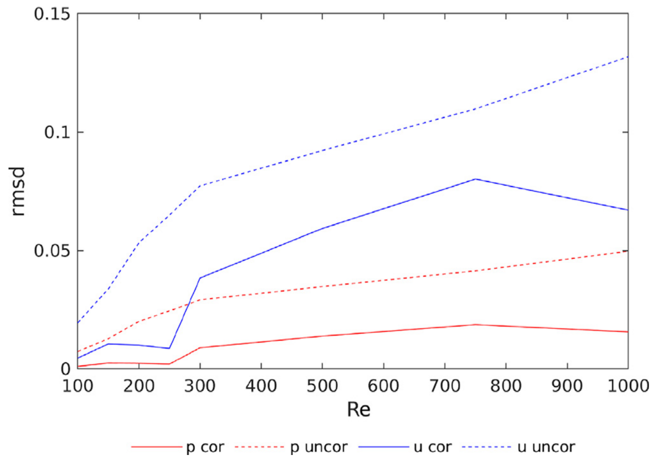

Fig. 6. Variation of RMSD with Re for the corrected and the uncorrected ROM. Flow over a cylinder.

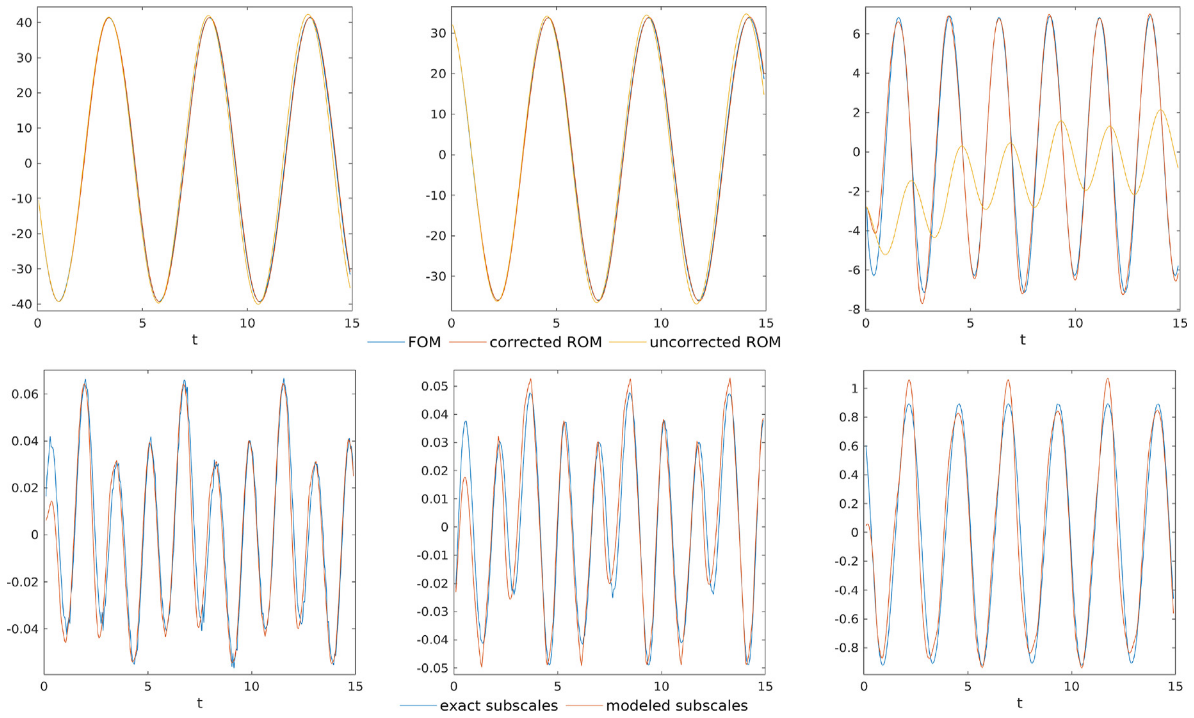

Fig. 7. Comparison of projected FOM and ROM coefficients (top) and SGSs (bottom). From left to right: Coefficients and SGSs associated to modes 1, 2 and 3. Flow over a cylinder.

$Z_{\phi,1}$ and $Z_{\phi,2}$ are a more complex functions of frequencies. The ANN can take into account all the complexities and predict $\mathbf{Z}_{\text{ann}}$ very close to $\mathbf{Z}_{\phi}$.

Temporal extrapolation phase

Data gathering for ROM basis and ANN. We use a different number of snapshots and basis functions to show that the correction works for a variety of scenarios. For this example, the FOM solutions for $t \in [0, 5]$, i.e., 100 time steps, are used to calculate the ROM basis. The ROM is also run for 100 time steps to gather data, which,

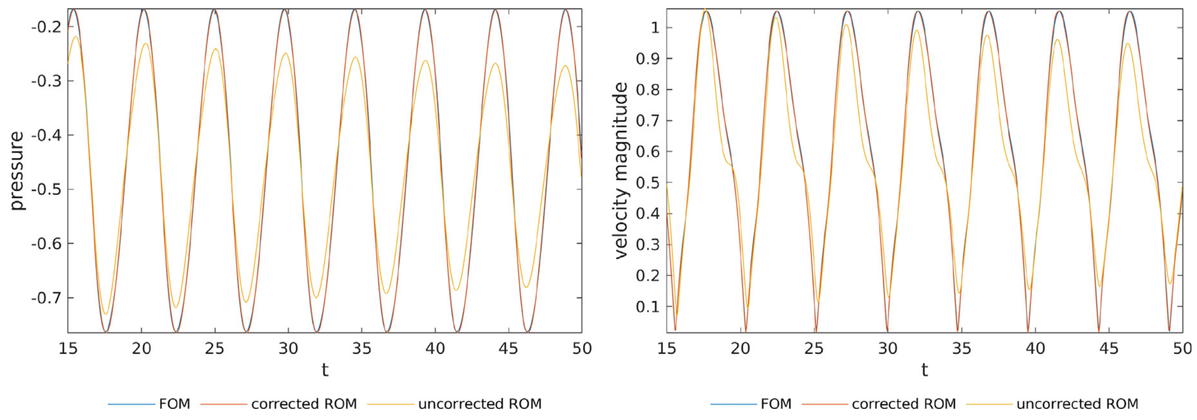

Fig. 8. Comparison between the FOM and the ROMs for the temporal extrapolation phase at Re = 250. Flow over a cylinder.

Table 3
RMSD values for the temporal extrapolation phase for p and u at Re = 250, 750.

Re	Pressure		Velocity magnitude	
	Corrected	Uncorrected	Corrected	Uncorrected
250	4.905×10^{-3}	5.829×10^{-2}	7.019×10^{-3}	8.900×10^{-2}
750	9.420×10^{-2}	6.323×10^{-1}	1.678×10^{-1}	9.263×10^{-1}

in addition to the data from the FOM, are used to train the ANN. The number of basis to be used is chosen to be $R = 6$ in this case.

Results. The performance of the ANN for the temporal extrapolation is evaluated for Re = {250, 750}. The training is done for $t \in [0, 5]$, whereas the testing is done for $t \in [0, 50]$, i.e., 10 times the training period. A comparison of p and u for the FOM, the uncorrected ROM and the corrected ROM at Re = 250 is shown in Fig. 8. The corrected ROM performs excellently for this temporal extrapolation, producing a solution nearly identical to that of the FOM. However, the uncorrected ROM quickly losses the amplitude with time, having a large discrepancy in amplitude at $t = 50$ with the FOM solution. Moreover, the uncorrected ROM has not yet converged to an amplitude value. Based on the current trend, the uncorrected ROM would either converge to a much lower amplitude or might even result in a steady flow without oscillations. Both solutions would be significantly different from the actual flow. In terms of quantitative error, Table 3 shows that the corrected ROM has *up to an order of magnitude* less error than the uncorrected ROM.

5.1.2. Flow over a backward facing step

Computational setting

The computational domain consists of a rectangle $]0, 44[\times]0, 9[$ minus a step of unit height placed at (4, 0). At the inlet $x = 0$, we take $u_x = 1$ and $u_y = 0$, whereas the top and bottom boundaries are prescribed a classical wall law. We take the wall distance defining the wall model to be $\delta = 0.001$. The outlet $x = 44$ has both, u_x and u_y, left free. The viscosity is prescribed to be $\nu = 2.5 \times 10^{-5}$, resulting in Re = 40000 calculated using the inlet velocity and the step size. A mesh of 31000 bilinear quadrilateral elements and 31371 nodes is used to discretize the domain for the FOM. The mesh is refined near the step to better capture the vortex generation. The BDF scheme of first order with $\delta t = 0.05$ is used for time integration for both the FOM and the ROM. A preliminary run of 2000 time steps is performed to obtain a fully developed flow. We consider this developed flow as the initial condition for the examples, i.e., $t = 0$ correspond to the developed flow. Also, to stabilize the solution at Re = 40000, dynamic SGSs are used [22]. Fig. 9 shows velocity and pressure contours of the FOM for the flow visualization. The control

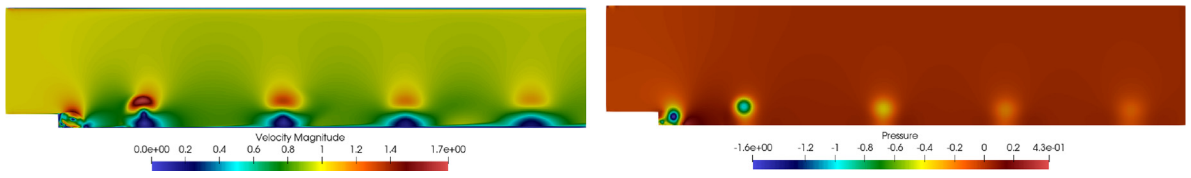

Fig. 9. Contours of the FOM for the flow over a backward facing step at Re = 40000.

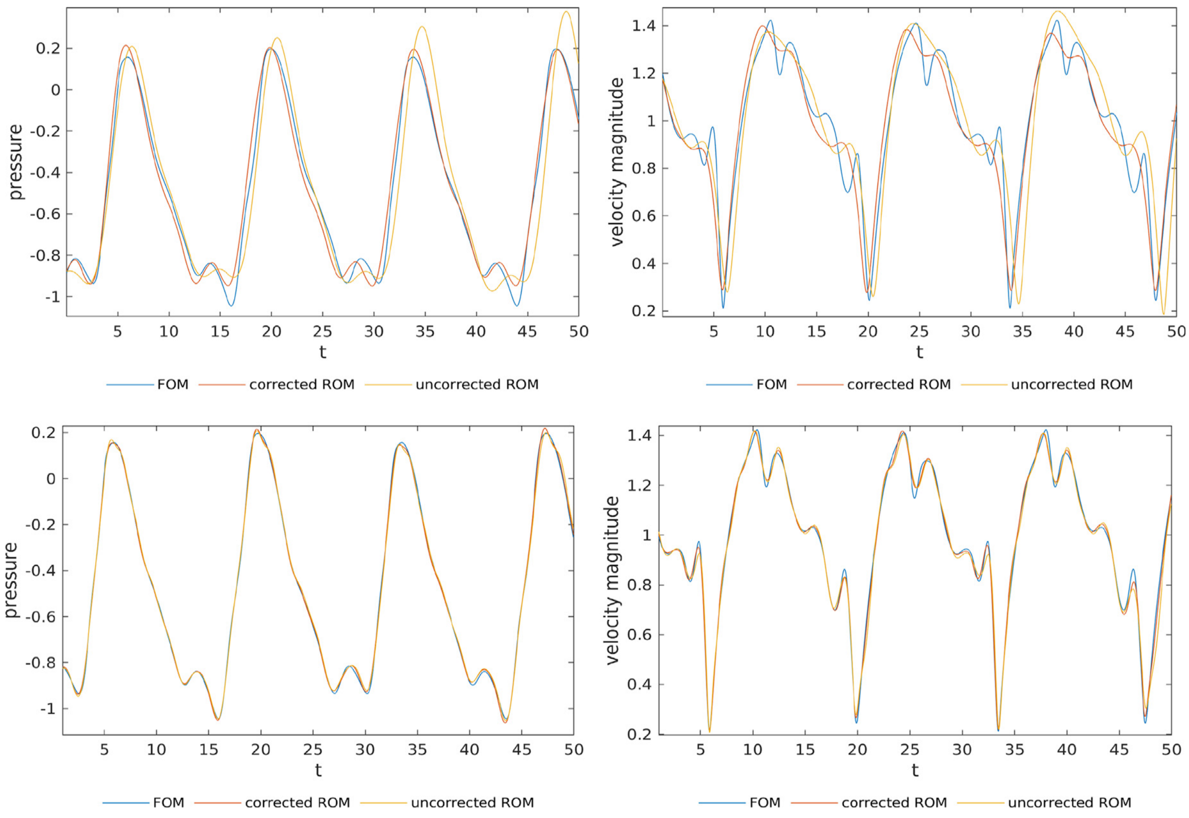

Fig. 10. Comparison between the FOM and the ROMs for the reconstruction phase. Top: $\eta = 0.6$ and $R = 7$, bottom: $\eta = 0.9$ and $R = 26$. Flow over a backward facing step.

point (5, 1), located immediately behind the step in the vortex generation region, is used to compare the values of p and u.

Reconstruction phase

Data gathering for ROM basis and ANN:. The FOM solutions for $t \in [0, 50]$ with snapshots at every third time step, i.e., a total of 333 snapshots, are used to calculate the ROM basis. The ROM is also run for 1000 time steps, with data gathered for every third time step, which, in addition to the data from the FOM, are used to train the ANN. We use three different numbers of POD modes, $R = \{7, 26, 77\}$, corresponding to the retained energy $\eta = \{0.6, 0.9, 0.95\}$, to analyze the behavior of the corrected ROM.

Results:. Fig. 10 shows the temporal evolution of p and u for $R = \{7, 26\}$. Also, the DFTs of the variables p and u for $R = \{7, 26\}$ are shown in Fig. 11. Finally, the RMSD for $R = \{7, 26, 77\}$ for p and u is shown in Table 4. From the time evolution behavior, the DFT and the RMSD it can be seen that the corrected ROM outperforms the uncorrected ROM irrespective of the variable being analyzed or the number of basis vectors being used. There might be some slight discrepancies, e.g., the DFT of velocity magnitude for $R = 26$, in which the uncorrected ROM

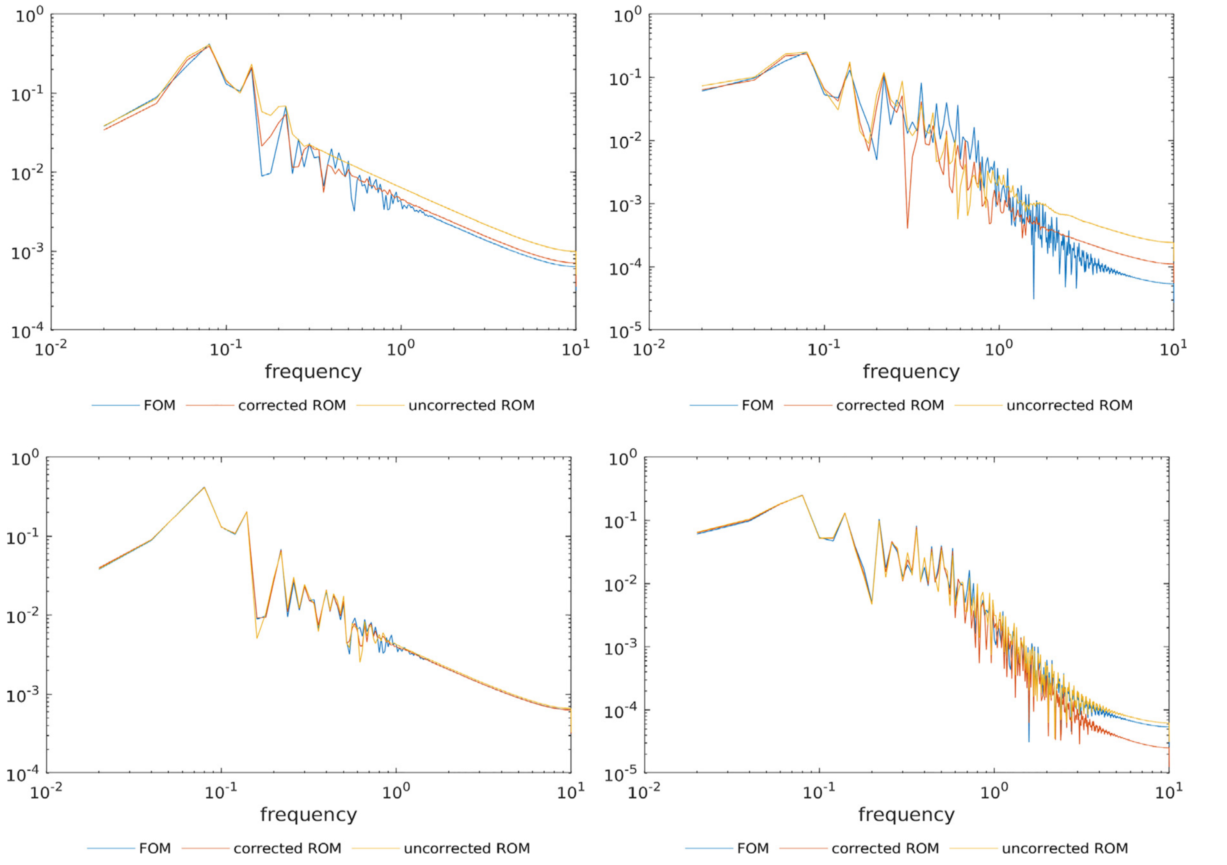

Fig. 11. Comparison of DFT between the FOM and the ROMs for the reconstruction phase. Top: $R = 7$, bottom: $R = 26$, left: p, right: u. Flow over a backward facing step.

Table 4
RMSD values for the reconstruction phase for p and u. Flow over a backward facing step.

No of basis	Pressure		Velocity magnitude	
	Corrected	Uncorrected	Corrected	Uncorrected
7	6.177×10^{-2}	1.808×10^{-1}	9.113×10^{-2}	1.819×10^{-1}
26	1.036×10^{-2}	1.566×10^{-2}	1.871×10^{-2}	3.127×10^{-2}
77	5.871×10^{-4}	9.374×10^{-4}	9.240×10^{-4}	1.096×10^{-3}

appears to capture slightly better the high frequencies as compared to the corrected ROM. However, we can see from the RMSD values that the corrected ROM still performs better, even for this case. Fig. 12 shows the RMSD for p and u as a function of R. In terms of convergence, both ROMs, corrected and uncorrected, approach the FOM solution as the number of basis is increased from 7 to 77. We also note that the difference between the corrected and the uncorrected ROM solutions is maximum for fewer modes, i.e., $R = 7$, with the corrected ROM being *more than twice* more accurate than the uncorrected ROM. For $R = 77$, both ROM solutions approach the FOM solution, with the corrected ROM being consistently more accurate.

Remark 2. The capability of the uncorrected ROM, equipped with the a priori closure model only, to capture the FOM solution correctly is in contrast to what we observed in [50]. This can be attributed to two differences in the approach used. First, in [50] the a priori SGSs were assumed to belong to $\widetilde{\mathcal{Y}}$, whereas in this work they are assumed to belong to $\mathcal{Y}'' = \widetilde{\mathcal{Y}} \oplus \mathcal{Y}'$. Second, the ROM basis used in [50] was orthogonal in the algebraic sense,

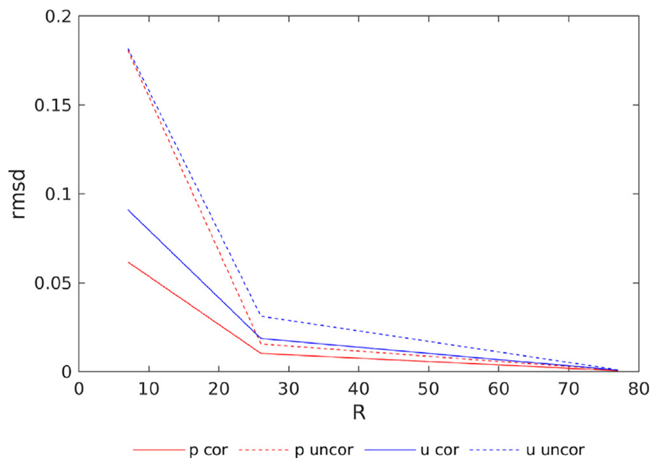

Fig. 12. RMSD as a function of R for the corrected and the uncorrected ROM for the reconstruction phase. Flow over a backward facing step.

Table 5
RMSD values for the temporal extrapolation phase for p and u. Flow over a backward facing step.

No of basis	Pressure		Velocity magnitude	
	Corrected	Uncorrected	Corrected	Uncorrected
7	1.956×10^{-1}	3.702×10^{-1}	2.071×10^{-1}	2.489×10^{-1}
26	3.405×10^{-2}	1.043×10^{-1}	3.806×10^{-2}	1.043×10^{-1}
77	5.156×10^{-3}	9.065×10^{-2}	5.974×10^{-3}	1.062×10^{-2}

i.e., $\boldsymbol{\Phi}^T \boldsymbol{\Phi} = \boldsymbol{I}_{(R)}$, whereas in this work the ROM basis is chosen to be orthogonal in the L^2 functional sense, i.e., $\boldsymbol{\Phi}^T \boldsymbol{M} \boldsymbol{\Phi} = \boldsymbol{I}_{(R)}$.

Temporal extrapolation phase
Data gathering for ROM basis and ANN. The data used for the basis construction and ANN training are the same as those used in the reconstruction phase, i.e., the FOM and ROM solutions for $t \in [0, 50]$.

Results. The corrected and uncorrected ROMs are run for $t \in [0, 150]$, i.e., 3000 time steps. Again, results are analyzed for three different number of POD modes, $R = \{7, 26, 77\}$. Fig. 13 shows the temporal evolution of p and u for $R = 7$. The RMSD values for different R for p and u are shown in Table 5. The time evolution behavior and RMSD values show that the trend of the corrected ROM performing consistently better is also replicated in the temporal extrapolation phase. Fig. 13 shows that for the highly truncated scenario, $R = 7$, the uncorrected ROM starts to differ drastically in amplitude at $t = 100$, whereas the corrected ROM still maintains the amplitude with slight error in frequency. Thus the more the ROM basis is truncated, the higher is the need of the ANN based correction model to get accurate results. It can also be noticed that the pressure amplitude of the uncorrected ROM is constantly increasing and could result in quite a large value if the simulation is continued for a longer time. For the velocity magnitude, even the mean value seems to be increasing with time. For the corrected ROM, a slight mismatch in frequency can be observed. However, the amplitude remains approximately constant throughout. Hence, reasonable conclusions can still be drawn regarding the dynamical behavior of the system using the corrected ROM. Fig. 14 shows the RMSD for p and u as a function of R. A convergence behavior similar to the reconstruction phase can be observed here as well.

5.1.3. Flow over three cylinders in a triangular arrangement

Computational Setting

The purpose of this example is to show the capability of the corrected ROM to capture the complexity of transient non-periodic flows. Consider the two dimensional channel flow over three circular cylinders in a triangular arrangement. The computational domain is $\Omega =]-4, 16[\times]-5, 5[$ minus the three cylinders, with the centers of

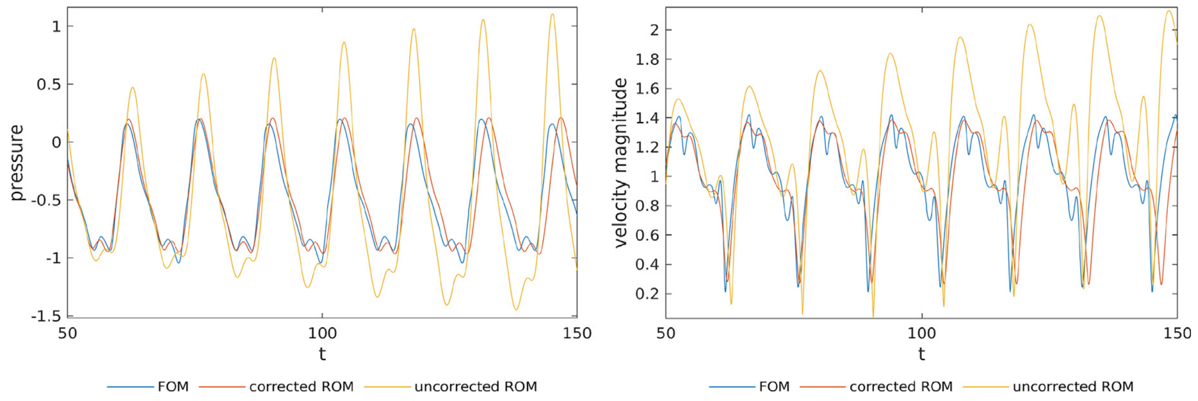

Fig. 13. Comparison between the FOM and the ROMs with $\eta = 0.6$ and $R = 7$ for the temporal extrapolation phase. Flow over a backward facing step.

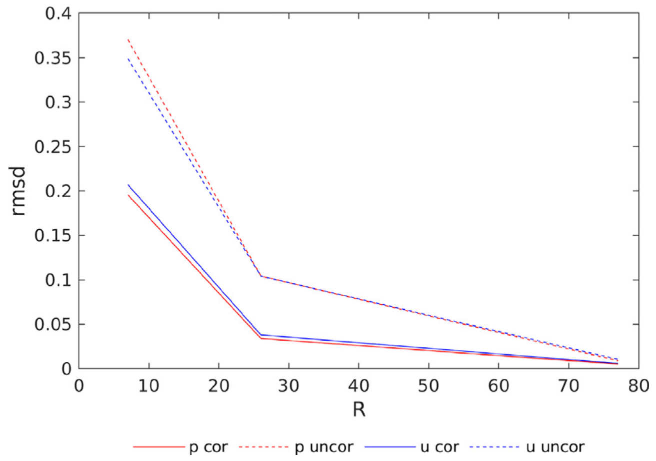

Fig. 14. RMSD as a function of R for the corrected and the uncorrected ROM in the temporal extrapolation phase. Flow over a backward facing step.

these located at $(0, 0)$, $(2, 1)$ and $(2, -1)$. The diameter of the all the cylinders is set to $D = 1$. Boundary conditions similar to the flow over a cylinder are used. The viscosity ν is selected to achieve $\mathrm{Re} = 1000$. An unsymmetrical mesh of 12289 bilinear triangular elements and 6349 nodal points is used to discretize the domain for the FOM. $\delta t = 0.05$ is used as the time step for both the FOM and the ROM. A preliminary run of 3000 time steps is performed to allow the flow to develop. We refer to this flow as the initial condition for the examples, i.e., $t = 0$ corresponds to this flow. Fig. 15 shows velocity and pressure contours for flow visualization. It can be seen that the flow has more complex features than the previous examples. The control point located at $(4, 0.5)$ is used to compare the values of p and u.

Reconstruction phase

Data gathering for ROM basis and ANN. The FOM solutions for $t \in [0, 15]$, i.e., 300 time steps, are used to calculate the ROM basis. The ROM is also run for 300 time steps to gather data, which, in addition to the data from the FOM, are used to train the ANN. The corrected ROM is run using $R = 20$, whereas the uncorrected ROM is run using $R = 80$.

Results. Due to the lack of periodicity, the uncorrected ROM is expected to produce poor results irrespective of the number of basis modes used. Fig. 16 shows that even with $R = 80$ and $\eta = 0.999$, the uncorrected ROM gives poor results for p and u, whereas the corrected ROM with only $R = 20$ can produce results very similar to the

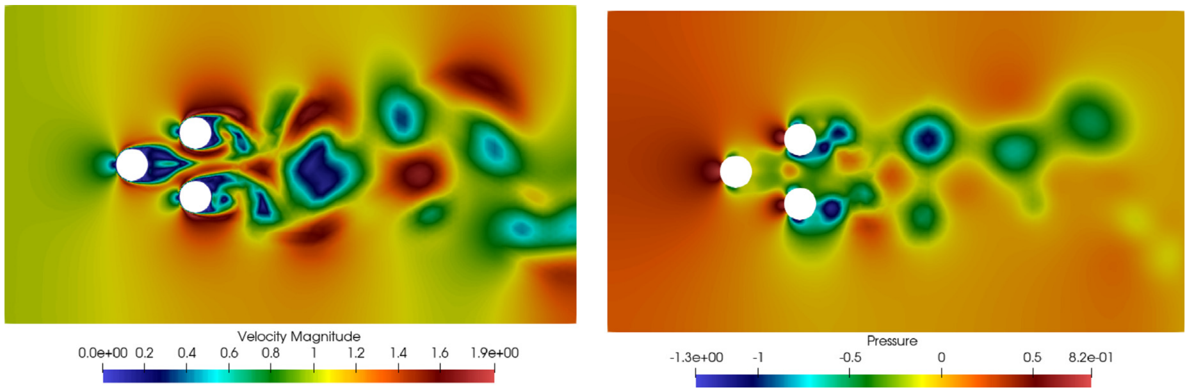
Velocity Magnitude
0.0e+00 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.9e+00
Pressure
-1.3e+00 -1 -0.5 0 0.5 8.2e-01

Fig. 15. Contours of the FOM for the flow over three cylinders at Re = 1000.

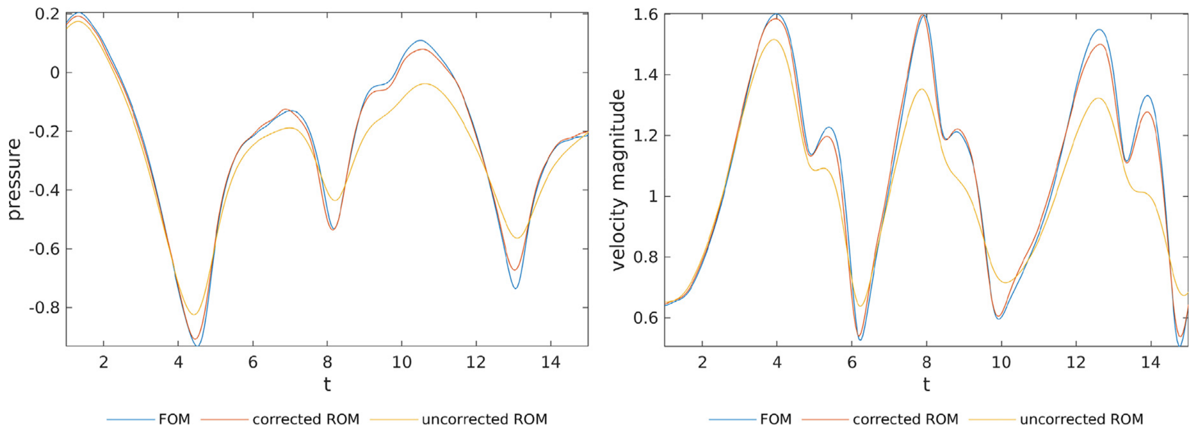

Fig. 16. Comparison between the FOM and the ROMs for the reconstruction phase at Re = 1000. Flow over three cylinders.

Table 6
RMSD values for the reconstruction phase for p and u at Re = 1000.

Re	Pressure		Velocity magnitude	
	Corrected	Uncorrected	Corrected	Uncorrected
1000	1.813×10^{-2}	7.259×10^{-2}	2.801×10^{-2}	1.206×10^{-1}

FOM. Fig. 17 shows that the corrected ROM captures the frequency spectrum better as well. Table 6 shows that the corrected ROM has *up to four times* less error than the uncorrected ROM. Thus, the corrected ROM cannot only produce more accurate results than the uncorrected ROM for highly truncated cases, but it can also capture a complex non-periodic flow evolution that the uncorrected ROM cannot capture even with $\eta \approx 1$.

Temporal extrapolation phase

Data gathering for ROM basis and ANN. Due to the chaotic nature of the flow, more snapshots are gathered for basis construction, as well as, for the ANN training. Also, the mesh used was found to be not fine enough to resolve the flow properly even for the FOM. Thus, the Smagorinsky turbulence model is used with a Smagorinsky constant value of 0.1. The FOM solutions for $t \in [0, 100]$, i.e., 2000 time steps, are used to calculate the ROM basis. The ROM is also run for the same number of steps to gather data, which, in addition to the data from the FOM, are used to train the ANN. The number of basis to be used is chosen to be $R = 16$ in this case, so that the basis is able to capture the prominent features of the system.

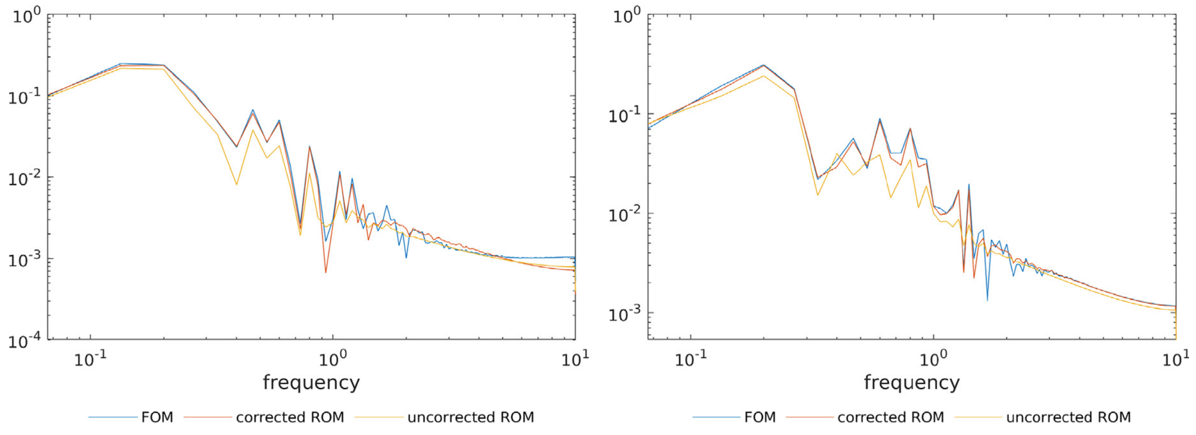

Fig. 17. Comparison of DFT between the FOM and the ROMs for the reconstruction phase at Re = 1000. Left: p, right: u. Flow over three cylinders.

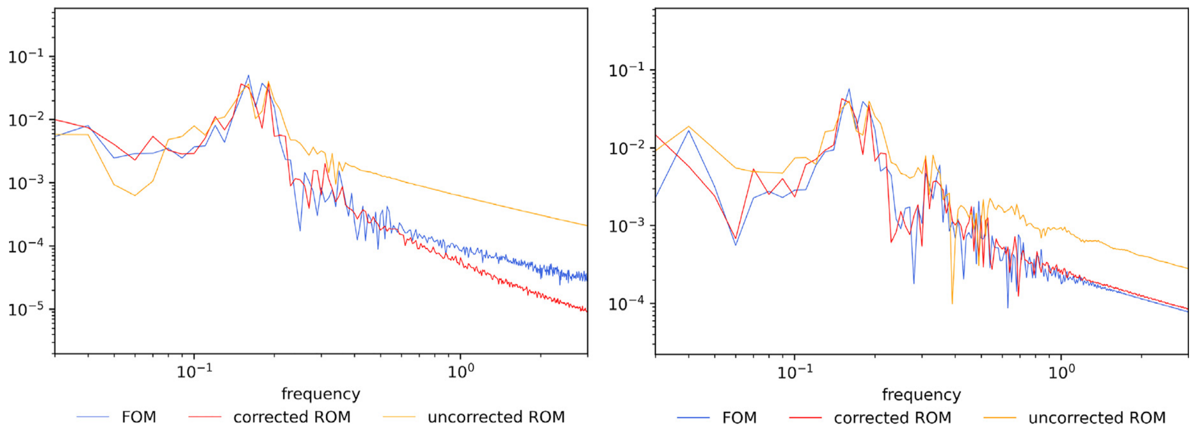

Fig. 18. Comparison of DFT between the FOM and the ROMs for the temporal extrapolation phase at Re = 1000. Left: p, right: u. Flow over three cylinders.

Results. The performance of the ANN for the temporal extrapolation is evaluated for $t \in [100, 200]$. Owing to the chaotic nature of the flow, the spectra of flow variables, p and u, provide more relevant information than their time evolution, as this time evolution may differ completely at a given point and a given time for slight differences in the numerical approximation. So, we compare the DFTs of flow variables to draw conclusions regarding the performance of ANN correction for the flow over three cylinders. A comparison of DFTs of p and u for the FOM, the uncorrected ROM, and the corrected ROM is shown in Fig. 18. The maximum frequency along the x-axis is chosen to be 3 Hz, so that with $\Delta t = 0.05$, we have enough samples, around 7 in this case, to determine the system response at the highest frequency. From the figure, it can be seen that the spectra of corrected ROM follow the FOM spectra much more closely than the uncorrected ROM. In general, the corrected ROM spectra capture the trend and peaks more accurately for low frequencies. However, the difference in the behavior is more pronounced for the higher frequencies. Particularly, for the velocity magnitude, it can be observed that the uncorrected ROM incurs about an order of magnitude of error at high frequencies.

5.2. Parametric analysis

The parametric examples are performed for the flow over a cylinder by interpolating and extrapolating the Reynolds number Re. This is analyzed with and without extrapolation in time.

The computational setting is the same as for the flow over a cylinder used in the non-parametric analysis.

Table 7
Training and testing iterative schemes for the parametric analysis.

Scheme No.	Training inputs	Testing inputs			
		1st iteration		2nd, … iterations	
		$\boldsymbol{Y}_\phi^{(l)}$	$\dot{\boldsymbol{Y}}_\phi^{(l)}$	$\boldsymbol{Y}_\phi^{(l)}$	$\dot{\boldsymbol{Y}}_\phi^{(l)}$
1	$\boldsymbol{S}_\phi^{n+1}, \dot{\boldsymbol{S}}_\phi^{n+1}$	$\boldsymbol{Y}_\phi^n$	$\dot{\boldsymbol{Y}}_\phi^n$	$\boldsymbol{Y}_\phi^{(k)}$	$\dot{\boldsymbol{Y}}_\phi^{(k)}$
2	$\boldsymbol{S}_\phi^{n+1}, \dot{\boldsymbol{S}}_\phi^{n+1}$	$\boldsymbol{Y}_\phi^n + \dot{\boldsymbol{Y}}_\phi^{(k)}\delta t$	$\dot{\boldsymbol{Y}}_\phi^n + \ddot{\boldsymbol{Y}}_\phi^n\delta t$	$\boldsymbol{Y}_\phi^{(k)}$	$\dot{\boldsymbol{Y}}_\phi^{(k)}$
3	$\boldsymbol{S}_\phi^{n+1}, \dot{\boldsymbol{S}}_\phi^{n+1}$	$\boldsymbol{Y}_\phi^n + \dot{\boldsymbol{Y}}_\phi^n\delta t$	$\frac{1}{\delta t}(\boldsymbol{Y}_\phi^{(k)} - \boldsymbol{Y}_\phi^n)$	$\boldsymbol{Y}_\phi^{(k)}$	$\dot{\boldsymbol{Y}}_\phi^{(k)}$
4	$\boldsymbol{S}_\phi^{n+1}, \dot{\boldsymbol{S}}_\phi^{n+1}$	$\boldsymbol{Y}_\phi^n + \dot{\boldsymbol{Y}}_\phi^n\delta t$	$\dot{\boldsymbol{Y}}_\phi^n + \ddot{\boldsymbol{Y}}_\phi^n\delta t$	$\boldsymbol{Y}_\phi^{(k)}$	$\dot{\boldsymbol{Y}}_\phi^{(k)}$
5	$\boldsymbol{S}_\phi^n, \dot{\boldsymbol{S}}_\phi^n$	$\boldsymbol{Y}_\phi^n$	$\dot{\boldsymbol{Y}}_\phi^n$	$\boldsymbol{Y}_\phi^n$	$\dot{\boldsymbol{Y}}_\phi^n$
6	$\boldsymbol{S}_\phi^{n+1}, \dot{\boldsymbol{S}}_\phi^n$	$\boldsymbol{Y}_\phi^n$	$\dot{\boldsymbol{Y}}_\phi^n$	$\boldsymbol{Y}_\phi^{(k)}$	$\dot{\boldsymbol{Y}}_\phi^n$

Table 8
List of the hyper-parameters utilized to train the ANNs for the parametric examples..

Variables	Values
Number of hidden layers	1
Number of neurons in each hidden layer	20
Batch size	32
`tol-train`	10^{-4}
`max-epocs`	200
Validation data set %	20
Regularization parameter λ	10^{-3}
Learning rate	10^{-3}

ANN. For the parametric study, the parameter Re needs to be provided as an additional input for training the ANN. Furthermore, the approximation $\boldsymbol{Z}_\phi^n \approx \boldsymbol{Z}_{\text{ann}}^n = \mathcal{F}_{\text{ann}}(\boldsymbol{Y}_\phi^n, \text{Re})$ for each time step n was found to be insufficient in this case. The ANN was unable to predict the solution for a Re for which it was not trained, irrespectively of the tuning of the hyper-parameters, the number of neurons and layers, and the amount of training data. To overcome this bottleneck, for the parametric study it is assumed that $\boldsymbol{Z}_{\text{ann}}^n = \mathcal{F}_{\text{ann}}(\boldsymbol{Y}_\phi^n, \dot{\boldsymbol{Y}}_\phi^n, \text{Re})$, i.e., SGSs are assumed to depend on the first time derivatives of the ROM coefficients as well.

Once we have the trained model, the linearized system at a given time step while testing is given by

$$\boldsymbol{A}_\phi \boldsymbol{Y}_\phi^{(k+1)} = \boldsymbol{R}_\phi - \boldsymbol{A}_\phi \mathcal{F}_{\text{ann}}(\boldsymbol{Y}_\phi^{(l)}, \dot{\boldsymbol{Y}}_\phi^{(l)}, \text{Re}), \tag{32}$$

with $\boldsymbol{A}_\phi$ evaluated with $\boldsymbol{Y}_\phi^{(k)}$. In the case of the parametric study, setting $l = k$ with $\boldsymbol{Y}_\phi^{n+1,(0)} = \boldsymbol{Y}_\phi^n$, i.e., the converged value of $\boldsymbol{Y}_\phi$ at the previous time step, did not work as it worked for the non-parametric case. This can be attributed to the large difference in the values of the derivative for the first and subsequent iterations. In order to find the iteration scheme that works best, several schemes given in Table 7 were tested. In this table, k denotes the values of the previous iteration of the current time step and n denotes the converged values of the previous time step. BDF1 is used approximate time derivatives. Thus:

$$\dot{\boldsymbol{Y}}_\phi^n = \frac{\boldsymbol{Y}_\phi^n - \boldsymbol{Y}_\phi^{n-1}}{\delta t}, \quad \dot{\boldsymbol{Y}}_\phi^{n+1,(k)} = \frac{\boldsymbol{Y}_\phi^{n+1,(k)} - \boldsymbol{Y}_\phi^n}{\delta t}, \quad \ddot{\boldsymbol{Y}}_\phi^n = \frac{\dot{\boldsymbol{Y}}_\phi^n - \dot{\boldsymbol{Y}}_\phi^{n-1}}{\delta t}.$$

Note that schemes 1–4 converge to $\mathcal{F}_{\text{ann}}(\boldsymbol{Y}_\phi^{n+1}, \dot{\boldsymbol{Y}}_\phi^{n+1}, \text{Re})$ (they only differ in the initial guess of the iterative scheme), scheme 5 converges to $\mathcal{F}_{\text{ann}}(\boldsymbol{Y}_\phi^n, \dot{\boldsymbol{Y}}_\phi^n, \text{Re})$ and scheme 6 to $\mathcal{F}_{\text{ann}}(\boldsymbol{Y}_\phi^{n+1}, \dot{\boldsymbol{Y}}_\phi^n, \text{Re})$. For simplicity, Re is not shown as a training input in Table 7.

The parameters and the hyper-parameters chosen for the ANN for the parametric study are shown in Table 8. In this case, some regularization is required to avoid over-fitting, as shown by the much higher value of λ as compared to the non-parametric case. We also use a larger batch size to improve the training speeds, as more training data are used for the parametric analysis.

Moreover, we use R different ANNs, with each ANN modeling the correction term for a single mode only. Each ANN takes the ROM coefficient and its time derivative as the inputs and provides the corresponding SGS as the

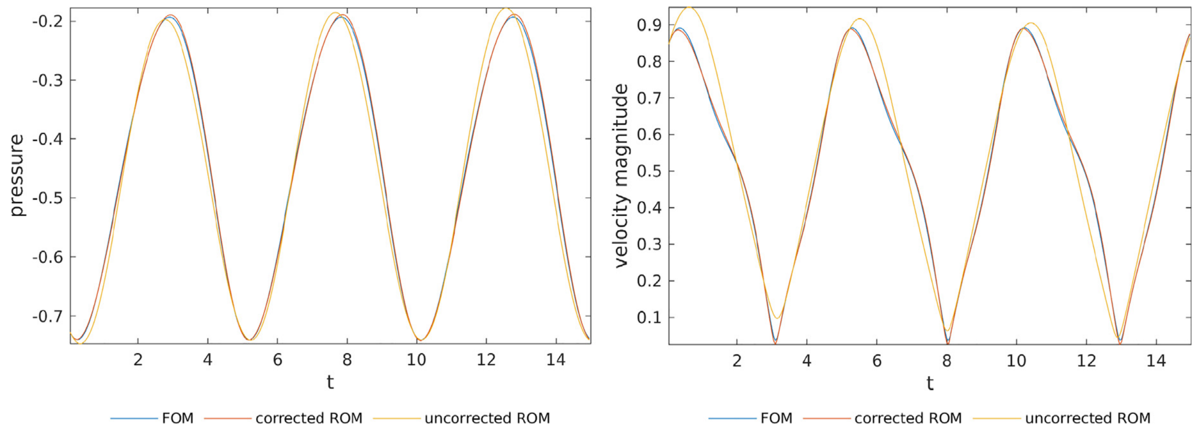

Fig. 19. Comparison between the FOM and the ROMs at Re = 200 for parametric interpolation. Flow over a cylinder.

output:

$$Y_{\phi,i}, \dot{Y}_{\phi,i}, \mathrm{Re} \to \mathcal{F}_{\mathrm{ann},i} \to Z_{\mathrm{ann},i}, \quad i = 1, \ldots, R.$$

The final setting used for parametric cases was tested for a few non-parametric cases and was found to work correctly. This is understandable as the non-parametric cases are less challenging. However, since we already had a *simpler* setting working for the non-parametric cases, it was deemed unnecessary to use a relatively more complex setting for them.

5.2.1. Interpolation

Data gathering for ROM basis and ANN. The data are gathered for Re = $\{100, 150, 250, 300\}$ to be used for the interpolation to Re = 200. The FOM solutions for $t \in [0, 15]$ are used to calculate the ROM basis. The ROM is also run for the same time interval to gather data, which, in addition to the data from the FOM, are used to train the ANN. The number of basis to be used is chosen to be $R = 3$. For the ANN training, data for all the Re was normalized together and then used for training. Schemes 1–4 use the same trained ANNs, whereas schemes 5 and 6 use separately trained ANNs for each of them. All ANNs were trained such that the validation error is approximately the same to ensure that the difference in the results is primarily due to different iterative schemes.

Results for different iterative schemes. First, the results of using different iterative schemes provided in Table 7 are analyzed to decide the most appropriate scheme. The time evolution and RMSD of the ROM coefficients $\boldsymbol{Y}_\phi$ were used to decide the best scheme. It turned out that the variable that was most difficult to capture is the third component $Y_{\phi,3}$. For schemes 1–4, the RMSD after three iterations was approximately the same, and significantly high. This indicates that trying to make the SGSs converge to the value of the present iteration is difficult, regardless of the initial guess of the iterative procedure. This is particularly so because of the difficulty to converge of the derivative. The situation improved significantly with scheme 5, which is nothing but a explicit treatment of the SGSs. Scheme 6 did not introduce any relevant improvement, and therefore the results to be presented next correspond to scheme 5.

Results for interpolation. Now the results for the interpolation case are analyzed. A comparison of p and u for the FOM, the uncorrected ROM and the corrected ROM is shown in Fig. 19. It can be seen that the results of the corrected ROM are almost indistinguishable from the FOM results. However, results of the uncorrected ROM incur error both in amplitude and in frequency. Fig. 20 also shows a comparison of the DFT of p and u for the FOM and the ROMs. Here again the corrected ROM performs significantly better than the uncorrected ROM by capturing the FOM behavior closely.

The RMSD values in Table 9 show almost *an order of magnitude* less error achieved by the corrected ROM.

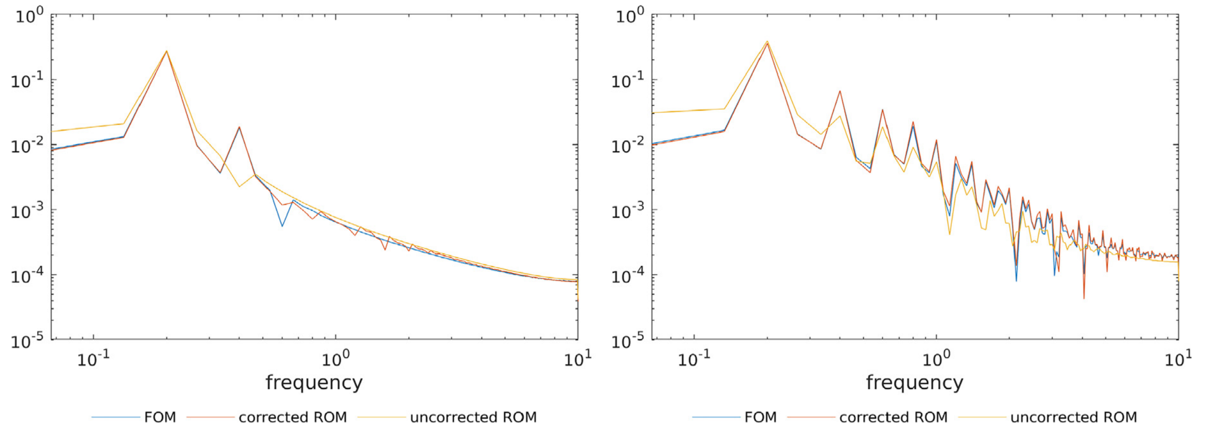

Fig. 20. Comparison of DFT between the FOM and the ROMs at Re = 200 for parametric interpolation. Left: p, right: u. Flow over a cylinder.

Table 9
RMSD values for Re = 200 trained using Re = {100, 150, 250, 300}.

Re	Pressure		Velocity magnitude	
	Corrected	Uncorrected	Corrected	Uncorrected
200	4.170×10^{-3}	2.401×10^{-2}	5.788×10^{-3}	5.814×10^{-2}

Table 10
RMSD values for Re = {300, 400, 500, 600} trained using Re = {100, 150, 200, 250}.

Re	Pressure		Velocity magnitude	
	Corrected	Uncorrected	Corrected	Uncorrected
300	7.575×10^{-3}	2.838×10^{-2}	2.315×10^{-2}	7.874×10^{-2}
400	1.010×10^{-2}	3.434×10^{-2}	2.542×10^{-2}	8.898×10^{-2}
500	1.324×10^{-2}	3.576×10^{-2}	3.837×10^{-2}	9.055×10^{-2}
600	3.645×10^{-2}	3.718×10^{-2}	8.691×10^{-2}	9.213×10^{-2}

5.2.2. *Extrapolation*

Data gathering for ROM basis and ANN. The data are gathered for Re = {100, 150, 200, 250} to be used for the extrapolation to Re = {300, 400, 500, 600}. The FOM solutions are collected for $t \in [0, 15]$, i.e., 300 time steps are used to calculate the ROM basis. The ROM is also run for 300 time steps to gather data, which, in addition to the data from the FOM, are used to train the ANN. The number of basis to be used is chosen to be the same as before, $R = 3$. For the ANN training, data for all Re are normalized together and then used for training.

Results. Extrapolation cases are perhaps the most tricky cases for ANNs. A comparison of p and u for the FOM, the uncorrected ROM and the corrected ROM for Re = 300 is shown in Fig. 21. It can be seen that, even for the extrapolation case, the results of the corrected ROM are very similar to the FOM results. However, the results of the uncorrected ROM diverge from those of the FOM as time advances. Fig. 22 also shows a comparison of the spectra of u and p for the FOM and the ROMs. Here again the corrected ROM performs significantly better than the uncorrected ROM by capturing the FOM behavior closely.

The RMSD values in Table 10 show almost *three times* less error achieved by the corrected ROM at Re = 300. The improvement in the performance of corrected ROM decreases as we move away from the training interval by increasing the Re. The corrected ROM gives excellent results for Re = 400 and still better results for Re = 500. For Re = 600 the error in corrected and uncorrected ROMs are almost the same. Thus, the correction has good extrapolation capabilities as it can provide good results up to Re = 500 which is 1.7 times outside the training interval.

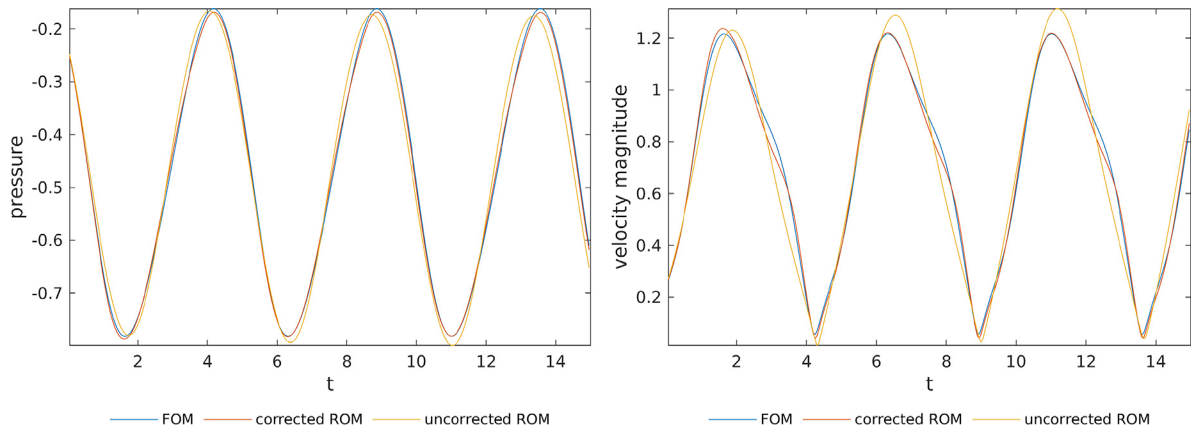

Fig. 21. Comparison between the FOM and the ROMs at Re = 300 for parametric extrapolation. Flow over a cylinder.

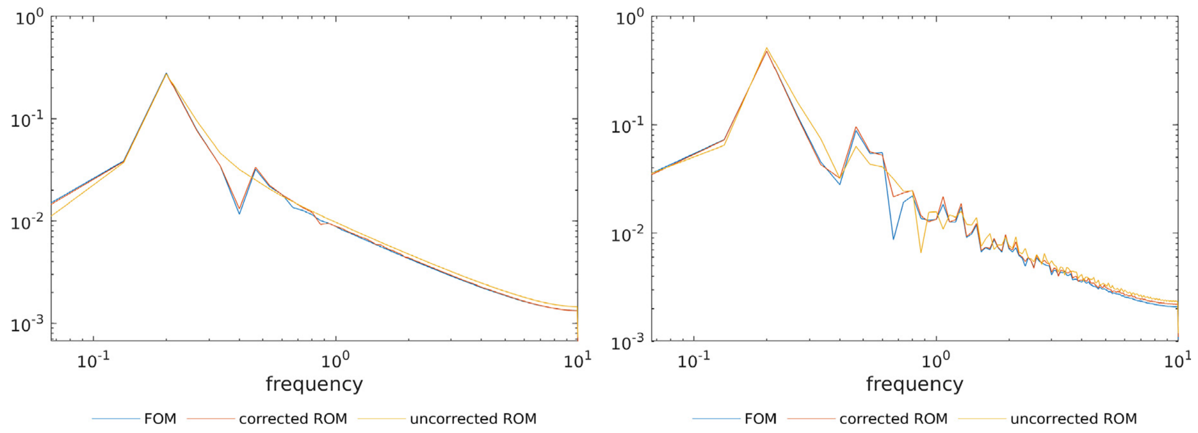

Fig. 22. Comparison of DFT between the FOM and the ROMs at Re = 300 for parametric extrapolation. Left: p, right: u. Flow over a cylinder.

5.2.3. *Parametric interpolation/extrapolation combined with temporal extrapolation phase*

Data gathering for ROM basis and ANN. Data from the FOM and the ROM are gathered the same way as it was done for the parametric interpolation and extrapolation cases, i.e., for $t \in [0, 15]$ (300 time steps) and for different Re. The number of basis functions to be used is again $R = 3$.

Results. The performance of the ANN for the temporal extrapolation is evaluated for Re = 200 and Re = 300 trained using Re = $\{100, 150, 250, 300\}$ and Re = $\{100, 150, 200, 250\}$ as interpolation and extrapolation cases, respectively. Furthermore, training is done for $t \in [0, 15]$, whereas testing is done for $t \in [0, 50]$, i.e., more than three times the training period. A comparison of p and u for Re = $\{200, 300\}$ for the FOM, the uncorrected ROM and the corrected ROM is shown in Figs. 23–24. It is interesting to note that after the initial deviation from the FOM periodic solution, the uncorrected ROM solution seems to be approaching a new periodic state as the variation in its amplitude is diminishing over time. But the amplitude of the new uncorrected ROM solution differs reasonably from the FOM solution, with minor differences in frequency as well. Whereas, the corrected ROM very closely retains the amplitude and frequency of the FOM solution even when extrapolated in time for an unseen Re.

The RMSD values are shown in Table 11, with the corrected ROM offering about *three to four times* less error than the uncorrected ROM for the interpolation and extrapolation cases .

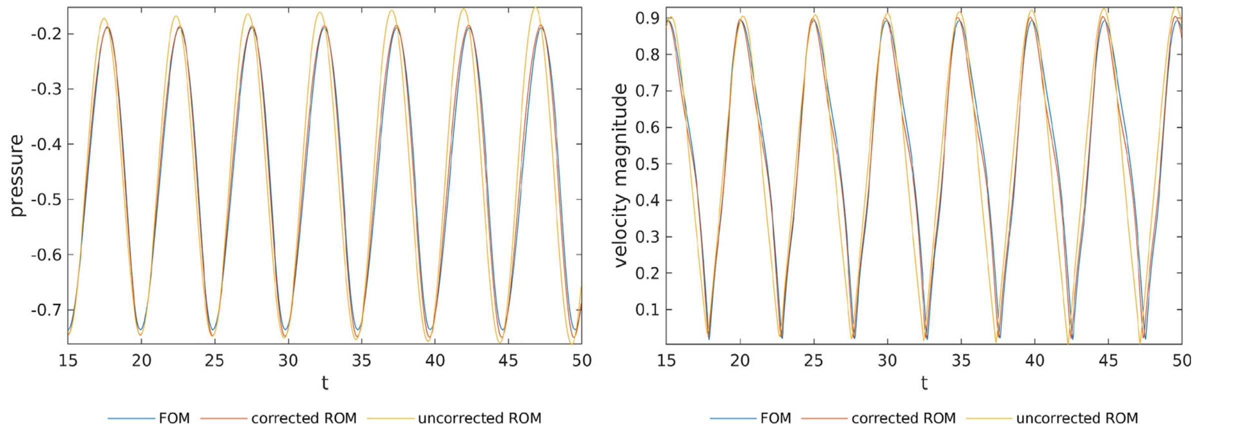

Fig. 23. Comparison between the FOM and the ROMs at Re = 200 for parametric interpolation combined with the temporal extrapolation phase. Flow over a cylinder.

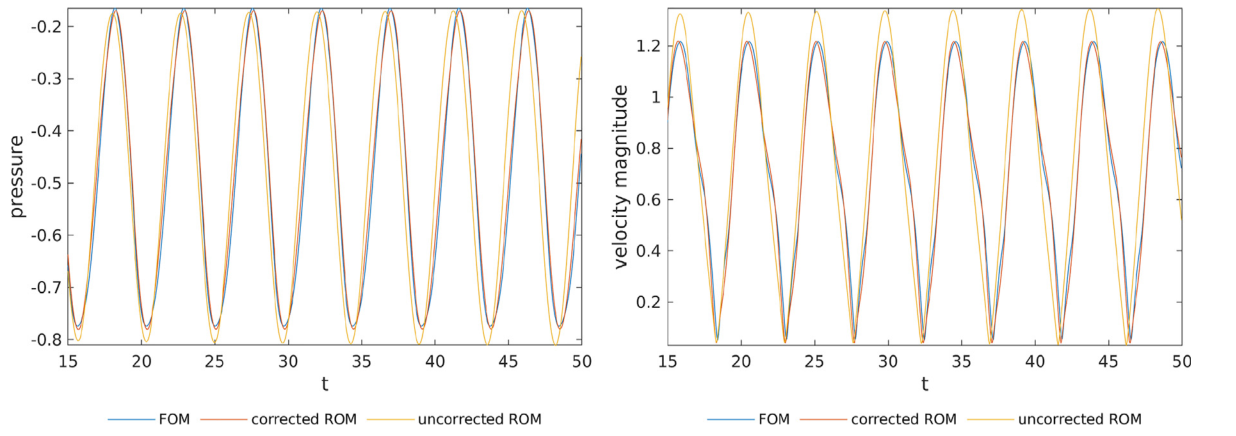

Fig. 24. Comparison between the FOM and the ROMs at Re = 300 for parametric extrapolation combined with the temporal extrapolation phase. Flow over a cylinder.

Table 11
RMSD values for parametric interpolation/extrapolation combined with the temporal extrapolation phase.

Re	Pressure		Velocity magnitude	
	Corrected	Uncorrected	Corrected	Uncorrected
200	1.823×10^{-2}	6.735×10^{-2}	3.039×10^{-2}	1.034×10^{-1}
300	1.970×10^{-2}	8.881×10^{-2}	3.717×10^{-2}	1.504×10^{-1}

5.3. Computational cost

For comparison of computational costs, some representative examples are selected for each type of evaluation case i.e. reconstruction phase, temporal extrapolation phase, and parametric phase. All the numerical tests are carried out on Dell Precision 3551 laptop having Intel(R) Core(TM) i7-10750H CPU @ 2.60 GHz, 12 cores, and 16.4 GB of memory. All FOM and ROM numerical simulations are carried out using a parallel FORTRAN-based numerical code.

The wall clock time for the online and the offline phases are shown in Table 12. Let us first discuss the more important computational cost i.e. the cost of the online phase. The ROMs provide a speed-up factor of 6 to 14

Table 12
Computational costs associated with different steps for representative numerical examples..

		Reconstruction			Temporal Extrapolation		Parametric
	Description	Cylinder (Re=250)	Backward facing step (R=7)	Three cylinders (Re=1000)	Cylinder (Re=250)	Backward facing step (R=7)	Cylinder (Re=300)
1	Full Order Model	2509.5	3553.9	1723.3	9129.3	8583.5	2610.2
	Offline costs						
2	Snapshots generation	2524.8	3578.7	1735.4	678.1	3578.7	15639.1
3	POD	56.2	155.8	27.2	49.6	155.8	206.0
4	Uncorrected ROM results collection	181.9	614.7	134.2	181.9	614.7	816.3
5	Training data collection (2+3+4)	2762.9	4349.3	1896.8	909.6	4349.3	16661.4
6	Hyper-parameter tuning	26.9	31.4	27.8	9.6	31.4	30.2
7	Training (multi-restart)	4.5	5.2	4.9	1.8	5.2	4.6
8	Total offline cost (5+6+7)	2794.3	4385.9	1929.5	921.0	4385.9	16696.2
	Online costs						
9	VMS-ROM uncorrected	176.0	590.0	180.2	638.5	1016.4	187.1
10	VMS-ROM corrected	179.7	603.2	133.5	645.3	1059.4	192.5
	Costs comparison						
11	FOM-uncorrected ROM speed-up factor	14.3	6.0	9.6	14.3	8.4	14.0
12	FOM-corrected ROM speed-up factor	14.0	5.9	12.9	14.1	8.1	13.6
13	Additional online cost of corrected ROM	2.1%	2.2%	-25.9%	1.1%	4.2%	2.9%

times when compared with the FOM. It is to be noted that no hyper-reduction is used and the speed-up experienced by ROMs is due to the reduction in system solving times only. Also, the additional cost associated with corrected ROMs in comparison to the uncorrected ROM is less than 4.7%. For triple cylinders, a speed-up can be seen in the case of corrected ROM as it uses fewer POD modes than the uncorrected one. During the execution of the corrected ROM, all ANN-related tasks are performed using Python using a single processor due to implementation requirements. The use of Python and serial processing incurs a computational cost to corrected ROM in addition to the one that corresponds to the evaluation of the correction term. However, even with this additional cost, the corrected ROM has an insignificant increase in the computational expense which can further be reduced by a more efficient implementation.

Now, let us briefly discuss the offline costs. All the steps of the offline phase are performed in the same order as they are listed in Table 12. Each step is parallelized to use the laptop resources optimally. During the hyper-tuning of ANNs, the use of simple ANNs allowed the launching of many ANN training sessions for different hyper-parameter values simultaneously. The same is true for multi-restart based training. The offline cost is dominated by snapshot generation, followed by collecting uncorrected ROM data for subscale calculation. Whereas, the cost associated with tuning and training the ANNs is pretty insignificant.

To better visualize some of the conclusions drawn using Table 12, a Pareto plot for the flow over a cylinder at Re $=$ 250 for the reconstruction phase is also shown in Fig. 25. The bars are arranged in decreasing order of the associated computational cost. It provides two insights. First, it shows how strongly the snapshot generation dominates the offline cost. Second, it also visually represents the online speed-up achieved. The cost of snapshots is roughly the same as the FOM cost and the uncorrected ROM cost is approximately the same as the online costs of both the ROMs. Thus the difference between the first and second bar represents the online speed-up achieved.

6. Conclusions

ROMs have the capability to drastically reduce the computational cost of numerical simulations, especially in the case of optimization and control problems, which require performing a large number of simulations. The real potential of ROMs is unlocked if only a few high energy modes are kept and the remaining low energy modes are discarded. However, this can lead to instability and inaccuracy in the solution.

In this paper, we have presented a two step closure model for ROMs to account for the interaction of the discarded scales with the resolved scales for ROMs. The aim of the first step is to provide the required stabilization. We use the already established VMS approach for this purpose. This approach uses physical arguments to develop an a priori linear SGS model. The VMS method also improves the accuracy, alongside providing the necessary stabilization. However, the accuracy provided is not sufficient in the case of highly truncated ROMs or for complex non-periodic flows. Thus, as a second step we introduce an a posteriori approach for SGS modeling using machine learning. We use ANNs to develop a non-linear correction for the SGSs and apply this correction to a fully discrete system, i.e., discretized in both space and time. To minimize the ANN training cost, we use the FOM data used in formulating reduced order basis, i.e., the snapshots, for ANN training as well; the ANNs is in fact designed so that the ROM solution is as close as possible to the projection of the snapshots onto the ROM space at the time levels

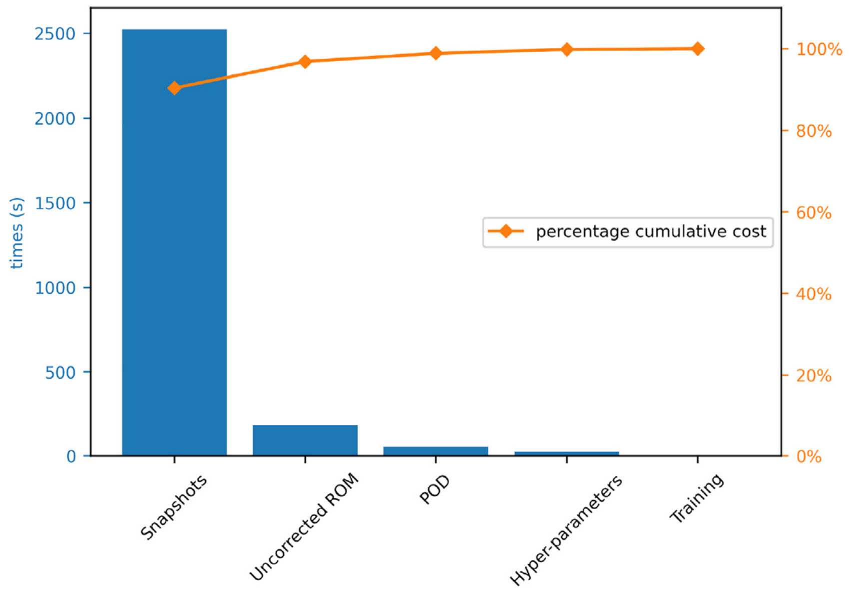

Fig. 25. Pareto plot showing the individual wall-clock times of different offline steps and their contribution towards the cumulative cost.

in which the snapshots are available. Furthermore, to keep the uncertainty associated to ANNs to a minimum, we use as simple as possible ANNs, increasing their complexity only when deemed necessary.

We have compared the corrected ROM based on the two step closure model with the uncorrected ROM based on only the a priori SGSs. The comparison has been carried out for a variety of cases including a reconstruction phase, a time extrapolation phase, parametric interpolation and extrapolation phases, and a combined parametric and time extrapolation phase. The parameter selected in this case has been the Reynolds number. The non-parametric analysis has been carried out for the flow over a cylinder, over a backward facing step and over three cylinders in a triangular arrangement, while the parametric analysis has been carried out for the flow over a cylinder only. In all cases, the corrected ROM has consistently outperformed the uncorrected ROM, providing solutions much closer to those of the FOM. The ANN based correction has been able to account for errors in amplitude as well as in frequency. Thus, ANNs can successfully account for the interaction of discarded scales with resolved scales for highly truncated ROMs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

R. Codina gratefully acknowledges the support received through the ICREA Acadèmia Research Program of the Catalan Government. CIMNE is a recipient of a “Severo Ochoa Programme for Centers of Excellence in R&D” grant (CEX2018-000797-S) by the Spanish Ministry of Economy and Competitiveness.

References

[1] E.W. Sachs, S. Volkwein, POD-Galerkin approximations in PDE-constrained optimization, GAMM-Mitt. 33 (2) (2010) 194–208.
[2] J. Yvonnet, Q.-C. He, The reduced model multiscale method (R3M) for the non-linear homogenization of hyperelastic media at finite strains, J. Comput. Phys. 223 (1) (2007) 341–368.

[3] J. Baiges, R. Codina, S. Idelsohn, Explicit reduced-order models for the stabilized finite element approximation of the incompressible Navier–Stokes equations, Internat. J. Numer. Methods Fluids 72 (12) (2013) 1219–1243.

[4] J. Baiges, R. Codina, S. Idelsohn, A domain decomposition strategy for reduced order models. Application to the incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 267 (2013) 23–42.

[5] J. Burkardt, M. Gunzburger, H.-C. Lee, POD and CVT-based reduced-order modeling of Navier–Stokes flows, Comput. Methods Appl. Mech. Engrg. 196 (1–3) (2006) 337–355.

[6] B. Galletti, C. Bruneau, L. Zannetti, A. Iollo, Low-order modelling of laminar flow regimes past a confined square cylinder, J. Fluid Mech. 503 (2004) 161–170.

[7] B. Glaz, L. Liu, P.P. Friedmann, Reduced-order nonlinear unsteady aerodynamic modeling using a surrogate-based recurrence framework, AIAA J. 48 (10) (2010) 2418–2429.

[8] D.J. Lucia, P.S. Beran, Projection methods for reduced order models of compressible flows, J. Comput. Phys. 188 (1) (2003) 252–280.

[9] I. Kalashnikova, M. Barone, Stable and efficientGalerkin reduced order models for non-linear fluid flow, in: 6th AIAA Theoretical Fluid Mechanics Conference, 2011, p. 3110.

[10] K. Veroy, A.T. Patera, Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: Rigorous reduced-basis a posteriori error bounds, Internat. J. Numer. Methods Fluids 47 (8–9) (2005) 773–788.

[11] Z. Wang, I. Akhtar, J. Borggaard, T. Iliescu, Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison, Comput. Methods Appl. Mech. Engrg. 237 (2012) 10–26.

[12] I. Akhtar, J. Borggaard, A. Hay, Shape sensitivity analysis in flow models using a finite-difference approach, Math. Probl. Eng. 2010 (2010).

[13] M. Bergmann, L. Cordier, J.-P. Brancher, Drag minimization of the cylinder wake by trust-region proper orthogonal decomposition, in: Active Flow Control, Springer, 2007, pp. 309–324.

[14] T. Lassila, G. Rozza, Parametric free-form shape design with PDE models and reduced basis method, Comput. Methods Appl. Mech. Engrg. 199 (23–24) (2010) 1583–1592.

[15] G. Rozza, T. Lassila, A. Manzoni, Reduced basis approximation for shape optimization in thermal flows with a parametrized polynomial geometric map, in: Spectral and High Order Methods for Partial Differential Equations, Springer, 2011, pp. 307–315.

[16] E. Arian, M. Fahl, E.W. Sachs, Trust-Region Proper Orthogonal Decomposition for Flow Control, Technical Report, Institute for Computer Applications in Science and Engineering, Hampton VA, 2000.

[17] B.R. Noack, M. Morzynski, G. Tadmor, Reduced-Order Modelling for Flow Control, Vol. 528, Springer Science & Business Media, 2011.

[18] W. Graham, J. Peraire, K. Tang, Optimal control of vortex shedding using low-order models. Part I — Open loop model development, Internat. J. Numer. Methods Engrg. 44 (7) (1999) 945–972.

[19] A. Deane, I. Kevrekidis, G.E. Karniadakis, S. Orszag, Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders, Phys. Fluids A: Fluid Dyn. 3 (10) (1991) 2337–2354.

[20] R. Reyes, R. Codina, Projection-based reduced order models for flow problems: A variational multiscale approach, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112844.

[21] R. Reyes, Stabilized Reduced Order Models for Low Speed Flows (Ph.D. thesis), Universitat Politècnica de Catalunya, 2020.

[22] R. Codina, J. Principe, O. Guasch, S. Badia, Time dependent subscales in the stabilized finite element approximation of incompressible flow problems, Comput. Methods Appl. Mech. Engrg. 196 (21–24) (2007) 2413–2430.

[23] R. Codina, Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods, Comput. Methods Appl. Mech. Engrg. 190 (13–14) (2000) 1579–1599.

[24] R. Codina, J. Principe, J. Baiges, Subscales on the element boundaries in the variational two-scale finite element method, Comput. Methods Appl. Mech. Engrg. 198 (5–8) (2009) 838–852.

[25] R. Codina, J. Baiges, Finite element approximation of transmission conditions in fluids and solids introducing boundary subgrid scales, Internat. J. Numer. Methods Engrg. 87 (1–5) (2011) 386–411.

[26] J. Baiges, R. Codina, A variational multiscale method with subscales on the element boundaries for the Helmholtz equation, Internat. J. Numer. Methods Engrg. 93 (6) (2013) 664–684.

[27] S. Parada, R. Codina, J. Baiges, A VMS–based fractional step technique for the compressible Navier–Stokes equations using conservative variables, J. Comput. Phys. 459 (2022) 111137.

[28] L. Moreno, R. Codina, J. Baiges, Solution of transient viscoelastic flow problems approximated by a term-by-term VMS stabilized finite element formulation using time-dependent subgrid-scales, Comput. Methods Appl. Mech. Engrg. 367 (2020) 113074.

[29] I. Castañar, J. Baiges, R. Codina, H. Venghaus, Topological derivative-based topology optimization of incompressible structures using mixed formulations, Comput. Methods Appl. Mech. Engrg. 390 (2022) 114438.

[30] L. Moreno, R. Codina, J. Baiges, Numerical simulation of non-isothermal viscoelastic fluid flows using a VMS stabilized finite element formulation, J. Non-Newton. Fluid Mech. 296 (2021) 104640.

[31] S. Parada, R. Codina, J. Baiges, Development of an algebraic fractional step scheme for the primitive formulation of the compressible Navier-Stokes equations, J. Comput. Phys. 433 (2021) 110017.

[32] I. Castañar, J. Baiges, R. Codina, A stabilized mixed finite element approximation for incompressible finite strain solid dynamics using a total Lagrangian formulation, Comput. Methods Appl. Mech. Engrg. 368 (2020) 113164.

[33] R. Reyes, R. Codina, J. Baiges, S. Idelsohn, Reduced order models for thermally coupled low Mach flows, Adv. Model. Simul. Eng. Sci. 5 (1) (2018) 1–20.

[34] A. Tello, R. Codina, J. Baiges, Fluid structure interaction by means of variational multiscale reduced order models, Internat. J. Numer. Methods Engrg. 121 (12) (2020) 2601–2625.

[35] A. Tello, R. Codina, Field-to-field coupled fluid structure interaction: A reduced order model study, Internat. J. Numer. Methods Engrg. 122 (1) (2021) 53–81.

[36] M. Guo, J.S. Hesthaven, Reduced order modeling for nonlinear structural analysis using Gaussian process regression, Comput. Methods Appl. Mech. Engrg. 341 (2018) 807–826.

[37] Q. Wang, J.S. Hesthaven, D. Ray, Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem, J. Comput. Phys. 384 (2019) 289–307.

[38] J.N. Kani, A.H. Elsheikh, DR-RNN: A deep residual recurrent neural network for model reduction, 2017, http://dx.doi.org/10.48550/ARXIV.1709.00939, URL https://arxiv.org/abs/1709.00939.

[39] A. Berzins, J. Helmig, F. Key, S. Elgeti, Standardized non-intrusive reduced order modeling using different regression models with application to complex flow problems, 2020, arXiv preprint arXiv:2006.13706.

[40] O. San, R. Maulik, Extreme learning Machine for reduced order modeling of turbulent geophysical flows, Phys. Rev. E 97 (4) (2018) 042322.

[41] Z.Y. Wan, P. Vlachas, P. Koumoutsakos, T. Sapsis, Data-assisted reduced-order modeling of extreme events in complex dynamical systems, PLoS One 13 (5) (2018) e0197704.

[42] A.J. Chorin, O.H. Hald, Stochastic Tools in Mathematics and Science, vol. 1, Springer, 2009.

[43] A.J. Chorin, O.H. Hald, R. Kupferman, Optimal prediction with memory, Physica D 166 (3–4) (2002) 239–257.

[44] A. Gupta, P.F. Lermusiaux, Neural closure models for dynamical systems, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 477 (2252) (2021) 20201004.

[45] Q. Wang, N. Ripamonti, J.S. Hesthaven, Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism, J. Comput. Phys. 410 (2020) 109402.

[46] O. San, R. Maulik, Neural network closures for nonlinear model order reduction, Adv. Comput. Math. 44 (6) (2018) 1717–1750.

[47] O. San, R. Maulik, Machine learning closures for model order reduction of thermal fluids, Appl. Math. Model. 60 (2018) 681–710.

[48] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning Machine: Theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.

[49] C. Mou, B. Koc, O. San, L.G. Rebholz, T. Iliescu, Data-driven variational multiscale reduced order models, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113470.

[50] J. Baiges, R. Codina, S. Idelsohn, Reduced-order subscales for POD models, Comput. Methods Appl. Mech. Engrg. 291 (2015) 173–196.

[51] T.J. Hughes, G.R. Feijóo, L. Mazzei, J.-B. Quincy, The variational multiscale method — A paradigm for computational mechanics, Comput. Methods Appl. Mech. Engrg. 166 (1–2) (1998) 3–24.

[52] R. Codina, Stabilized finite element approximation of transient incompressible flows using orthogonal subscales, Comput. Methods Appl. Mech. Engrg. 191 (39–40) (2002) 4295–4321.

[53] R. Codina, S. Badia, J. Baiges, J. Principe, Variational multiscale methods in computational fluid dynamics, in: Encyclopedia of Computational Mechanics, Wiley Online Library, 2018, pp. 1–28.

[54] O. Colomés, S. Badia, R. Codina, J. Principe, Assessment of variational multiscale models for the large eddy simulation of turbulent incompressible flows, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 285 (2015) 32–63, http://dx.doi.org/10.1016/j.cma.2014.10.041, URL https://www.sciencedirect.com/science/article/pii/S0045782514004113.

[55] G. Stabile, F. Ballarin, G. Zuccarino, G. Rozza, A reduced order variational multiscale approach for turbulent flows, Adv. Comput. Math. 45 (2019) 2349–2368.

[56] M. Strazzullo, M. Girfoglio, F. Ballarin, T. Iliescu, G. Rozza, Consistency of the full and reduced order models for evolve-filter-relax regularization of convection-dominated, marginally-resolved flows, Internat. J. Numer. Methods Engrg. 123 (14) (2022) 3148–3178.

[57] M. Barrault, Y. Maday, N.C. Nguyen, A.T. Patera, An ‘empirical interpolation’method: Application to efficient reduced-basis discretization of partial differential equations, C. R. Math. 339 (9) (2004) 667–672.

[58] N.C. Nguyen, A.T. Patera, J. Peraire, A ‘best points’ interpolation method for efficient approximation of parametrized functions, Internat. J. Numer. Methods Engrg. 73 (4) (2008) 521–543.

[59] J.A. Hernandez, M.A. Caicedo, A. Ferrer, Dimensional hyper-reduction of nonlinear finite element models via empirical cubature, Comput. Methods Appl. Mech. Engrg. 313 (2017) 687–722.

[60] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient non-linear model reduction via a least-squaresPetrov –Galerkin projection and compressive tensor approximations, Internat. J. Numer. Methods Engrg. 86 (2) (2011) 155–181.

[61] A. Fabra, J. Baiges, R. Codina, Finite element approximation of wave problems with correcting terms based on training artificial neural networks with fine solutions, Comput. Methods Appl. Mech. Engrg. 399 (2022) 115280.

[62] F. Chollet, et al., Keras, 2015, https://github.com/fchollet/keras.

[63] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale Machine learning on heterogeneous distributed systems, 2016, arXiv preprint arXiv:1603.04467.

[64] A. Pradhan, K. Duraisamy, Variational multiscale closures for finite element discretizations using the Mori–Zwanzig approach, Comput. Methods Appl. Mech. Engrg. 368 (2020) 113152.

[65] A. Gouasmi, E.J. Parish, K. Duraisamy, A priori estimation of memory effects in reduced-order models of nonlinear systems using the Mori–Zwanzig formalism, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 473 (2017) 20170385.

[66] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.