Data-driven dynamical modeling using machine learning and data assimilation

Nishant Kumar, Franck Kerhervé, Laurent Cordier

Data-driven dynamical modeling using machine learning and data assimilation

Nishant Kumar [1] and Franck Kerhervé [2] and Laurent Cordier [3]

Abstract

In fluid flow problems, complex dynamics, manifested in terms of large and small scale spatio-temporal features, is commonly encountered. Resolution of such dynamics results in high-dimensional discretized numerical models with associated high computational cost. This renders the full-scale models intractable in applications where repeated realizations are required, such as flow control. Reduced-order model (ROM) offers a way to mitigate this issue by offering a low-dimensional system which is computationally efficient and accurate. An automated data-driven inference of ROMs circumvents the requirement of an *a priori* knowledge of the governing equations by relying on the time-series data obtained from simulation or experiments for the modeling. In this paper, a non-intrusive approach based on artificial neural network (NN-ROM) is considered for the reduced-order modeling. NN-ROM serves as a map approximating the reduced coefficients of a high-fidelity solution in low-dimensional space. A novel multistep, residual-based neural network framework is proposed. The proposed approach is used to recover the dynamical states in numerical and experimental fluid flow problems. The framework provides sufficiently accurate initial estimate. Deviations in the long-term prediction are mitigated by augmenting the framework with data assimilation (DA),

1. Introduction

A wide range of systems in the field of fluid mechanics exhibit *complex* dynamics. The complexity pertains to spatial and temporal features originating from instabilities, nonlinearities, or turbulence which span a range from small to large scales, especially for flows involving high Reynolds number, compressibility, or combustion. In these flow cases, very fine spatial and temporal discretizations are required for the adequate resolution and propagation of the flow states. The numerical analysis performed to understand the complex underlying physical phenomena involves solving the associated full-scale models governing the flow dynamics. A detailed analysis of the flow features demands the availability of highly resolved spatio-temporal data. This requires discretized systems with high degrees of freedom, *e.g.* $Re^{9/4}$ for direct numerical simulations. With the advancements in numerical simulations and experimental measurement tools, handling large data representing the high-dimensional nonlinear dynamics poses the challenge of high computational cost. These high-fidelity systems are therefore not suited for applications such as real-time control [13], multidisciplinary optimization (MDO) [48], or uncertainty quantification (UQ) [51] where repeated realizations of the system are required. To alleviate this computational burden, it is of key interest to develop and study approaches that seek to reduce the size and cost of computational models while providing robust prediction of the state of a fluid flow system and forecast its evolution. This motivates the formulation of reduced, low-dimensional, efficient models representing the flow dynamics. *Reduced-order modeling* offers a viable approach to facilitate the real-time turn-around of computational results without sacrificing accuracy [3]. In this regard, a large variety of reduced-order modeling methodologies have been proposed over the years. One of the ways to obtain reduced-order models (ROMs) is offered by physics-based approaches, such as, projection-based models. Alternatively, there has

[1]Institut PPRIME CNRS UPR 3346, Université de Poitiers, ENSMA (France)
[2]Institut PPRIME CNRS UPR 3346, Université de Poitiers, ENSMA (France)
[3]Institut PPRIME CNRS UPR 3346, Université de Poitiers, ENSMA (France)

been growing interest towards non-intrusive approaches recently, such as, subspace identification and neural networks, in which empirical ROMs are derived from the input-output data.

The dimensionality reduction of fluid flow systems is made apparent by the observation that similar spatial flow features – like the von Kármán vortex street and Kelvin-Helmholtz instability – emerge across a wide range of flow geometries and parameters. The occurrence of these similar prominent features indicates the existence of common underlying phenomena that capture the essence of the flow [75]. These features are extracted using *modal decomposition* techniques based on some energetic or dynamical criteria, providing *modes* as the spatial features of the flow, and *modal coefficients* as the associated characteristic values representing the energy content levels or growth rates and frequencies. The modes allow the realization of a low-dimensional *latent space* coordinate system (reduced basis) from a high-dimensional *physical space* and the formulation of a low-order approximation of the flow dynamics in terms of its dominant components. Different methods have been employed in the literature which are unique in terms of the modal structures that are extracted, highlighting different aspects of the flow field. Based on the inputs used for calculating the modes, the modal decomposition methods are classified into two main groups [75] – *data-based* methods which rely on flow-field data from numerical simulations or experiments , and *operator-based* methods which use the operator describing the state dynamics to obtain the modes. In this paper, the data-based modal decomposition method of proper orthogonal decomposition (POD) [40] is considered to perform reduced-order modeling in the latent space provided by POD modes. Other alternatives are provided by the data-based dynamic mode decomposition (DMD) [71] or the operator-based methods such as Koopman analysis [54], global linear stability analysis [76], and resolvent analysis [72].

The POD analysis was first introduced in the context of fluid dynamics by Lumley [50] as a way to extract and analyze coherent structures in experimental turbulent flows, and was later used to analyze numerical simulations of turbulent flows [62, 79]. A variant known as bi-orthogonal decomposition (BOD) to obtain temporal structures of the modes was investigated by Aubry [5], and a *snapshot POD* variant to compute both the spatial and temporal components of the modes was proposed by Sirovich [73]. Subsequently, a link between POD and singular value decomposition (SVD) enabled the formulation of a discrete dataset in terms of a simple matrix factorization, as demonstrated by Kunisch and Volkwein [45]. Owing to the simple matrix factorization framework, POD is a popular tool for analyzing experimental and numerical fluid mechanics. Typical applications of POD include the identification of coherent structures from experimental data [26], flow control [17, 13], reduced-order modeling [24], and data-driven identification of nonlinear systems [18, 49].

The ROMs are fundamentally characterized by the reduced basis that defines them. Here, the POD modal decomposition has been used to obtain low-dimensional and computationally tractable ROMs, providing approximations of the high-fidelity dynamics with low computational times [53, 56]. Other commonly used projection methods for constructing ROMs are DMD method [77], reduced basis method [61, 36, 42], cross Gramian method [10, 38], piecewise tangential interpolation method [9], matrix interpolation method [58, 25], approximate balancing method [21], and balanced truncation method [35, 47, 52]. The choice of the projection method to represent the dynamics has direct implications on the success of the subsequent modeling and flow analysis. The methods based on POD and its variants have been applied successfully to numerous research fields including fluid flow control [2, 7, 8, 14, 22, 39, 46] and data assimilation [19, 27, 74, 64]. The complexity of the models used in practical applications means that numerical packages for creating and computing these models are not readily available. Several strategies are used to circumvent this issue regarding model complexity and based on their dependency on governing equations, the ROMs are divided into two categories [30] – physics-based *intrusive* ROM and purely data-driven *non-intrusive* ROM.

The classical POD-Galerkin projection approach has been found to provide an efficient and accurate way to generate ROMs [6, 12] along with successful implementation closure models that have been suggested in order to model the effects of discarded modes [57]. However, it has been found that the dynamics corresponding to certain flow phenomena is not captured accurately by the few dominant spatial modes [67]. The truncation also introduces instability in the projection-based ROM [70]. Inclusion of more modes to counter this limitation and recover the embedded structures leads to an increase in the computational expense which may render the ROM computationally

inefficient. The construction of a reduced-order state space is also crucial in control as each degree of freedom can potentially amplify noise [28]. Also, the low-dimensional deterministic ROM fails to resolve the frequency range of the high-dimensional Navier-Stokes solutions [6]. Moreover, the development of intrusive ROMs is hindered if the source code describing the physical model is not available or if it is too complicated, which is often the case for legacy codes.

Due to these limitations, there is a recent interest in non-intrusive reduced-order modeling as these models do not require the knowledge of the physical system while attempting to accurately approximate the truncated basis. Several NIROMs following different approaches have been proposed in the literature. An approach for reduced-order modeling based on *neural networks* was introduced by Noack et al. [56]. A neural network is a form of machine learning method which is capable of approximating an arbitrary function using observed data. The neural network based reduced-order models has been applied in several fluid dynamics applications [60, 20, 37, 59, 78]. A neural network derived from simulation of a cylinder wake flow was used by Gillies [31] for controlling a self-excited cylinder wake oscillations. This empirical model of the modal response of the wake to external forcing was then used to design a closed-loop control algorithm. A non-intrusive POD-ROM for aerodynamic shape optimization was developed by Iuliano and Quagliarella [43]. A NIROM framework based on POD and radial basis function (RBF) and the discrete empirical interpolation method (DEIM) algorithm using artificial neural network (ANN) was proposed by Winter and Breitsamter [81] and Guenot et al. [34]. A kernel method based on both support vector machines (SVMs) and a vectorial kernel greedy algorithm was proposed by Wirtz and Haasdonk [82]. A NIROM based on constrained POD (CPOD) and Kriging interpolation method was proposed by Xiao et al. [83]. A non-intrusive method for the polynomial chaos representation to extract a set of optimal basis functions from a coarser mesh and use it to perform finer mesh analysis was presented by Raisee et al. [65].

In this paper, a novel artificial neural network (ANN) framework – named NN-ROM for neural network reduced-order model – is presented as a NIROM method for the time series prediction of transient dynamics of the POD modes. This strategy is based on the multistep configuration of the deep neural network (DNN) presented by Pawar et al. [59] where the values of the states at previous time steps are used to obtain the time evolution of the POD projection coefficients. Additionally, as this NIROM framework provides sequential estimates of the dynamics, it can be considered as a forward model in the data assimilation paradigm of ensemble Kalman filter (EnKF) while assimilating the estimates and observations.

The workflow of the dynamical modeling and reconstruction approach involves a sequence of three main operations. First, data-based modal decomposition is performed using the high-fidelity snapshots in the physical space in order to extract the reduced basis in the latent space. Next, the reduced-order model in terms of the latent space variables is identified using the non-intrusive approach. Lastly, the identified model, augmented with data assimilation, is used to obtain future estimates of the latent space variables which, in turn, can be used to provide high-fidelity reconstructions in the physical space. In Sec. 2.1, the data-based modal decomposition technique, namely POD, used to extract the reduced bases from high-dimensional nonlinear flow data is described. In Sec. 2.2, the non-intrusive ROM framework based on deep neural network, which is capable of providing iterative predictions of the dynamical component of POD, is presented. The EnKF data assimilation method used to assimilate the model estimates with observations in order to provide long-term predictions is also described. In Sec. 3, the neural network based non-intrusive ROM (NN-ROM) is applied to a toy model and two fluid flow problems and the results are discussed. The consideration of memory effect by the NN-ROM is demonstrated for Lorenz-63 system. Also, the sequential NN-ROM is combined with the EnKF data assimilation algorithm to enable long-term predictions of the dynamics of a cylinder wake flow at high Reynolds number. Finally, Sec. 4 summarizes the most important results and proposes some perspectives for future work.

2. Reduced-order modeling and dynamical prediction

The modal decomposition technique used to provide a reduced basis for the representation of high-dimensional nonlinear flow systems and the non-intrusive approach to obtain the reduced-order model, augmented with data assimilation for long-term prediction, are discussed in this section.

2.1. POD modal decomposition

Proper orthogonal decomposition (POD) is a method to determine energetically optimal set of modes to represent the data which was introduced by Lumley [50] as a mean to deterministically identify the large-scale structures present in turbulent flows. As POD is a data-based method, it only requires the information of the scalar (*e.g.* pressure, temperature) or vectorial (*e.g.* velocity, vorticity) elements associated with the linear or nonlinear dynamics. Let us consider a field $\boldsymbol{q}(\boldsymbol{\chi}, t)$, where $\boldsymbol{\chi}$ is the vector representing the relevant spatial coordinates and t is a scalar time. Let $\boldsymbol{q}'(\boldsymbol{\chi}, t)$ be the fluctuating component which is obtained by subtracting the temporal mean $\overline{\boldsymbol{q}}(\boldsymbol{\chi})$ from the field $\boldsymbol{q}$,

$$\boldsymbol{q}'(\boldsymbol{\chi}, t) = \boldsymbol{q}(\boldsymbol{\chi}, t) - \overline{\boldsymbol{q}}(\boldsymbol{\chi}). \tag{1}$$

The goal is to decompose the fluctuating field into a set of deterministic spatial functions modulated by time coefficients, *i.e.*

$$\boldsymbol{q}'(\boldsymbol{\chi}, t) = \sum_{i=1}^{+\infty} a_i(t)\boldsymbol{\Phi}_i(\boldsymbol{\chi}), \tag{2}$$

where, $\boldsymbol{\Phi}_i$ and a_i represent the POD (spatial) *modes* and the *modal coefficients*, respectively. In the POD framework, we seek the "proper" or optimal basis functions $\boldsymbol{\Phi}_i$ such that the projection of the field $\boldsymbol{q}'$ on the first K spatial function $\boldsymbol{\Phi}_i$ is maximized on average, irrespective of the value of K. We assume that the field $\boldsymbol{q}'$ is defined at discrete times t_k, where $k = 1, \ldots, N_t$, with N_t being the total number of snapshots of the field. The corresponding POD maximization problem is given as[4]

$$\max_{\{\boldsymbol{\Phi}_i\}_{i=1}^{K}} \sum_{j=1}^{N_t} \|\Pi_{\mathrm{POD}}\boldsymbol{q}'(\boldsymbol{\chi}, t_j)\|_{\Omega}^2 , \tag{3a}$$

subject to

$$\|\boldsymbol{\Phi}_k\|_{\Omega}^2 = 1 \quad k = 1, \cdots, K, \tag{3b}$$

where Π_{POD} is the orthogonal projector on the space spanned by the first K functions $\boldsymbol{\Phi}_i$, *i.e.*

$$\Pi_{\mathrm{POD}}\boldsymbol{q}'(\boldsymbol{\chi}, t_j) = \sum_{k=1}^{K} \langle \boldsymbol{q}'(\boldsymbol{\chi}, t_j), \boldsymbol{\Phi}_k(\boldsymbol{\chi}) \rangle_{\Omega} \boldsymbol{\Phi}_k(\boldsymbol{\chi}). \tag{4}$$

Since Π_{POD} is an orthogonal projector, the maximization problem (3) is equivalent to the minimization problem given by

$$\min_{\{\boldsymbol{\Phi}_i\}_{i=1}^{K}} \sum_{j=1}^{N_t} \|\boldsymbol{q}'(\boldsymbol{\chi}, t_j) - \Pi_{\mathrm{POD}}\boldsymbol{q}'(\boldsymbol{\chi}, t_j)\|_{\Omega}^2 . \tag{5}$$

This means that for any subspace of size K, POD optimizes the averaged residual with respect to the norm (4) [see 23, for instance]. The POD method allows the representation of a signal using a minimum number of modes, which suits reduced-order modeling. Hereafter, we introduce the solution of the constrained maximization problem (3) in terms of correlation matrix.

The data obtained from experiments or numerical simulations are always finite-dimensional. Consequently, the field $\boldsymbol{q}'(\boldsymbol{\chi}, t_j)$ is considered to be defined at discrete spatial points $\boldsymbol{\chi}_j$ $(j = 1, \ldots, N_\chi)$ where N_χ is the number of spatial grid points. The fluctuating component of the field in (1) can be

[4]The definitions of the inner product and its induced norm is introduced. Let $\boldsymbol{q}^{\mathrm{I}}$ and $\boldsymbol{q}^{\mathrm{II}}$ be two given fields defined in a spatial domain, the spatial inner product is defined as

$$\left\langle \boldsymbol{q}^{\mathrm{I}}(\boldsymbol{\chi}, t), \boldsymbol{q}^{\mathrm{II}}(\boldsymbol{\chi}, t) \right\rangle_{\Omega} \coloneqq \int_{\Omega} \boldsymbol{q}^{\mathrm{I}}(\boldsymbol{\chi}, t) \cdot \boldsymbol{q}^{\mathrm{II}}(\boldsymbol{\chi}, t)\, \mathrm{d}\boldsymbol{\chi},$$

where the dot represents the Euclidean inner product. The induced norm is $\|\boldsymbol{q}^{\mathrm{I}}\|_{\Omega} = \sqrt{\langle \boldsymbol{q}^{\mathrm{I}}, \boldsymbol{q}^{\mathrm{I}} \rangle_{\Omega}}$.

re-written in the discrete space as

$$\boldsymbol{q}'(\boldsymbol{\chi}_j, t_k) = \boldsymbol{q}(\boldsymbol{\chi}_j, t_k) - \overline{\boldsymbol{q}}(\boldsymbol{\chi}_j), \quad j = 1, \ldots, N_\chi, \quad k = 1, \ldots, N_t. \tag{6}$$

Here, $\boldsymbol{q}'(\boldsymbol{\chi}_j, t_k) \in \mathbb{R}^{N_c \times 1}$, where N_c is the number of components of the input data (*e.g.* $N_c = 2$ for a two-dimensional velocity field, $N_c = 1$ for a pressure field). We also introduce $N_s = N_\chi \times N_c$ as the number of spatial points saved per time snapshot. The ensemble of snapshots $\boldsymbol{q}'(\boldsymbol{\chi}, t_k) \in \mathbb{R}^{N_s \times 1}$ can be summarized into a snapshot matrix $\boldsymbol{X}$ as

$$\boldsymbol{X} = [\boldsymbol{q}'(\boldsymbol{\chi}, t_1) \quad \boldsymbol{q}'(\boldsymbol{\chi}, t_2) \quad \cdots \quad \boldsymbol{q}'(\boldsymbol{\chi}, t_{N_t})] \in \mathbb{R}^{N_s \times N_t}. \tag{7}$$

The velocity fluctuation are essentially random but there exists some order in the randomness. The order is observed in the form of coherent structures which are identified via zones where the fluid motion is synchronized and the fluctuations $\boldsymbol{q}'(\boldsymbol{\chi}, t)$ are correlated. The correlation of the data is verified by computing its covariance matrix. In the direct method, the covariance matrix is calculated using the snapshot matrix $\boldsymbol{X} \in \mathbb{R}^{N_s \times N_t}$ defined in (7) to construct a spatial covariance matrix as $\boldsymbol{C}_s = \boldsymbol{X}\boldsymbol{X}^\top/(N_t - 1)$. However, the *snapshot POD* method introduced by Sirovich [73] is considered here. The method uses the snapshot matrix to construct a temporal correlation matrix as

$$\boldsymbol{C}_t = \frac{1}{N_s - 1}\boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{N_t \times N_t}. \tag{8}$$

The snapshot POD is preferred over the direct method as in most practical cases, the number of spatial measurement points is larger than the number of snapshots, *i.e.* $N_\chi \gg N_t$. This makes the covariance matrix $\boldsymbol{C}_t$ memory efficient and computationally more tractable as compared to the spatial covariance matrix $\boldsymbol{C}_s$.

It has been shown that the optimal basis vectors $\boldsymbol{\Phi}_i$ for data projection are obtained from the eigenvectors of the covariance matrix $\boldsymbol{C}_s$ [50]. However, the spatial coefficients and temporal modes of the snapshot method, differ from the spatial modes and temporal coefficients of the direct method by a multiplicative factor. The eigenvalues obtained using both the methods remain the same. The eigendecomposition of $\boldsymbol{C}_t$ provides deterministic temporal modes $\boldsymbol{A}_t \in \mathbb{R}^{N_t \times N_t}$ and eigenvalues $\boldsymbol{\Lambda} \in \mathbb{R}^{N_t \times N_t}$ as

$$\boldsymbol{C}_t = \boldsymbol{A}_t\boldsymbol{\Lambda}\boldsymbol{A}_t^{-1}. \tag{9}$$

The stochastic spatial coefficients $\boldsymbol{\Phi}_t$ are then obtained by projecting the fluctuation data matrix $\boldsymbol{X}$ on the temporal basis $\boldsymbol{A}_t$ as

$$\boldsymbol{\Phi}_t = \boldsymbol{X}\boldsymbol{A}_t^\top \in \mathbb{R}^{N_s \times N_t}. \tag{10}$$

The matrix $\boldsymbol{\Phi}_t$ contains the N_t spatial coefficients (equivalent to the spatial modes of the direct method), ordered energetically along the columns.

To match the results of both methods, we normalize each spatial coefficient of the snapshot POD[5] and scale the temporal modes accordingly. The spatial coefficients of the snapshot POD are normalized to obtain the spatial modes, which are orthonormal and equivalent to those obtained from direct method, as

$$\boldsymbol{\Phi} = \frac{1}{\sqrt{N_s - 1}}\boldsymbol{X}\boldsymbol{A}_t\boldsymbol{\Lambda}^{-1/2} \in \mathbb{R}^{N_s \times N_t}. \tag{11}$$

Finally, the temporal coefficients of the direct method are obtained from the spatial modes as

$$\boldsymbol{A} = \boldsymbol{\Phi}^\top\boldsymbol{X} \in \mathbb{R}^{N_t \times N_t}, \tag{12}$$

where an element A_{ik} of the matrix $\boldsymbol{A}$, represents the projection of the data measured at time t_k on

[5]The spatial modes $\boldsymbol{\Phi}$ in the direct method are orthonormal as they originate from the eigendecomposition of the symmetric correlation matrix $\boldsymbol{C}_s$. On the other hand, the spatial modes $\boldsymbol{\Phi}_t$ in the snapshot POD are not orthonormal as $\boldsymbol{\Phi}_t$ is not obtained directly from eigendecomposition but derived through projection (10).

mode Φ_i, given as

$$A_{ik} = a_i(t_k) = \boldsymbol{\Phi}_i(\boldsymbol{\chi})^\top \boldsymbol{q}'(\boldsymbol{\chi}, t_k) \tag{13}$$

$$= \sum_{j=1}^{N_s} (\boldsymbol{\Phi}_i)_{m(j)} (\boldsymbol{\chi}_{n(j)}) q'_{m(j)}(\boldsymbol{\chi}_{n(j)}, t_k), \quad \text{where} \quad m(j) = \lceil j/N_\chi \rceil. \tag{14}$$

Here $m(j) \in [1, N_c]$ indicates the component of the fluctuation vector, and $n(j) = j - (m(j) - 1)\, N_\chi \in [1, N_\chi]$ is the spatial coordinate index.

The eigenvalues rank the correlation with respect to the variance of the data. In fluid dynamics, if the fluctuations $\boldsymbol{q}'$ are obtained from velocity measurements, and if the inner product (4) is used, this corresponds to a ranking based on the turbulent kinetic energy (TKE) of the velocity fluctuations. The individual eigenvalues λ_i therefore represent the TKE associated with the i-th mode and are ordered in the matrix $\boldsymbol{\Lambda}$ based on their contribution to the total TKE. Based on this relationship between the eigenvalues and TKE, we introduce an energetic criterion to determine the number of modes that are necessary to sufficiently capture the flow dynamics. The criterion, known as *relative information content* or *RIC* [13], is the fraction of total energy that is represented by an ensemble of first n modes, given as

$$RIC(n) = \sum_{i=1}^{n} \lambda_i \Bigg/ \sum_{i=1}^{N_t} \lambda_i \in (0, 1] \quad \forall\, n \leq N_t. \tag{15}$$

2.2. Non-intrusive reduced-order modeling

In the proposed non-intrusive ROM (NIROM) framework, the projection step of intrusive ROM is bypassed with a neural network architecture which is capable of approximating the nonlinear functions and used as a forward model for dynamical prediction of the temporal POD coefficients. Unlike reduced-order models based on Galerkin projection, the NIROM is not physically interpretable. However, the ANN construction allows to represent nonlinear relationships that cannot be expressed explicitly in functional form. The strategy presented here corresponds to the approach where the neural network approximates the map between the time and state values at previous time steps as inputs, and the residuals of temporal POD coefficients as outputs [59, 78]. For a given time history of coefficients, the NIROM sequentially provides the projection coefficients at the subsequent time steps. Here, the main elements of the neural network-based ROM (NN-ROM) are presented.

2.2.1. Regression via neural network

The ability of deep neural network (DNN) to approximate nonlinear dynamical systems is used to obtain surrogate models that can iteratively predict the temporal dynamics of POD coefficients. In the proposed framework, the *multistep* and *residual-based* approaches have been combined.

- The multistep approach [66] consists of using the time history of a set of POD coefficients to predict a future state of the system, similar to classical time-stepping schemes like backward differentiation formulation (BDF). This approach allows to incorporate the memory effect, which helps to tackle the challenges of learning nonlinear temporal dynamics.

- The residual approach [63, 69] involves having the residuals calculated using consecutive values of POD coefficients, given as

  $$\boldsymbol{r}(t_{k+1}) = \boldsymbol{a}(t_{k+1}) - \boldsymbol{a}(t_k), \tag{16}$$

  instead of the sequential values of the solution trajectory given by $\{\boldsymbol{a}(t_k), \boldsymbol{a}(t_{k+1}), \ldots\}$. Although both formulations are mathematically equivalent, this simple transformation has been shown to yield relatively more stable and accurate results.

The regression setup includes *offline* and *online* stages. In the offline stage, the DNN model, henceforth referred as neural network-based ROM or NN-ROM, is trained in a supervised learning paradigm by using high-fidelity POD data in order to determine approximate maps between the

inputs (past projection coefficients and time) and residuals $\boldsymbol{r}(t)$. In the online stage, the learned NN-ROM is rapidly evaluated to obtain the predicted projection coefficients corresponding to the new sets of inputs.

Combining the above concepts, the NN-ROM $\boldsymbol{f}^{\mathrm{NN}}$ is formulated as a map between the *input features*, *i.e.* time step t_k and POD coefficients $\boldsymbol{a}^{\mathrm{POD}}(t_k)$, and the *targets* given in terms of the residuals $\boldsymbol{r}(t_{k+1})$. Let $\boldsymbol{W}$ and $\boldsymbol{b}$ be the weights and biases of the model. The discrete form of $\boldsymbol{f}^{\mathrm{NN}}$ is given by

$$\begin{aligned} \boldsymbol{r}(t_{k+1}) &= \boldsymbol{f}^{\mathrm{NN}}\left(\boldsymbol{a}^{\mathrm{POD}}(t_{k-p+1}), \ldots, \boldsymbol{a}^{\mathrm{POD}}(t_k); \boldsymbol{W}, \boldsymbol{b}\right) + \boldsymbol{\epsilon}_{k+1}, \\ &\forall\ p = 1, \ldots, N_t^{\mathrm{Train}} - 1, \quad \text{and} \quad k = p-1, \ldots, N_t^{\mathrm{Train}} - 2, \end{aligned} \tag{17}$$

where $\boldsymbol{\epsilon}_{k+1}$ is the modeling error at time step $k+1$. After training the DNN (offline stage, Sec. 2.2.2), the learned weights and biases ($\boldsymbol{W}^*$ and $\boldsymbol{b}^*$) are used to predict sequentially the temporal dynamics (online stage) as

$$\begin{aligned} \boldsymbol{a}^{\mathrm{NN}}(t_{k+1}) &= \boldsymbol{a}^{\mathrm{NN}}(t_k) + \boldsymbol{f}^{\mathrm{NN}}(\boldsymbol{a}^{\mathrm{NN}}(t_{k-p+1}), \ldots, \boldsymbol{a}^{\mathrm{NN}}(t_k); \boldsymbol{W}^*, \boldsymbol{b}^*), \\ &\forall\ p = 1, \ldots, N_t^{\mathrm{Pred}} - 1, \quad \text{and} \quad k = p-1, \ldots, N_t^{\mathrm{Pred}} - 1. \end{aligned} \tag{18}$$

In (17) and (18), the superscripts "Train" and "Pred" are used to distinguish the variables pertaining to the training and prediction regimes. Here, $\boldsymbol{a}^{\mathrm{NN}}(t_k) \in \mathbb{R}^{N_r \times 1}$ represents the POD projection coefficients obtained from the DNN framework at time instant t_k while $\boldsymbol{a}^{\mathrm{POD}}(t_k) \in \mathbb{R}^{N_r \times 1}$ represents the coefficients obtained from high-fidelity dataset used to train the NN-ROM. The model is trained using data from N_t^{Train} time steps and used to predict the evolution for N_t^{Pred} time steps. N_r is the number of modes considered in the reduced space based on the criteria (15).

The features (input) and target (output) data are arranged in matrices to facilitate the training of the NN-ROM. The computed POD coefficients $a_i^{\mathrm{POD}}(t_k)$, for all $i = 1, \ldots, N_r$ and $k = 0, \ldots, N_t^{\mathrm{Train}}$, are used to define the matrix of input features $\boldsymbol{A} \in \mathbb{R}^{(N_t^{\mathrm{Train}}-p)\times(N_r \times p+1)}$ given as

$$\boldsymbol{A} = \begin{bmatrix} t_p & a_1^{\mathrm{POD}}(t_0) & \cdots & a_{N_r}^{\mathrm{POD}}(t_0) & \cdots & a_{N_r}^{\mathrm{POD}}(t_{p-1}) \\ t_{p+1} & a_1^{\mathrm{POD}}(t_1) & \cdots & a_{N_r}^{\mathrm{POD}}(t_1) & \cdots & a_{N_r}^{\mathrm{POD}}(t_p) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ t_{N_t^{\mathrm{Train}}-1} & a_1^{\mathrm{POD}}(t_{N_t^{\mathrm{Train}}-p-1}) & \cdots & a_{N_r}^{\mathrm{POD}}(t_{N_t^{\mathrm{Train}}-p-1}) & \cdots & a_{N_r}^{\mathrm{POD}}(t_{N_t^{\mathrm{Train}}-2}) \end{bmatrix}. \tag{19}$$

As the input is made of non-homogeneous features, *i.e.* time and POD coefficients, the individual scale and distribution can be different for each variable. To avoid sensitivity to certain input values[6], *normalization* of the input matrix is performed [41]. The rescaled input features are obtained from row-wise normalization as

$$\boldsymbol{A}_{:,i} \leftarrow \frac{\boldsymbol{A}_{:,i} - \min(\boldsymbol{A}_{:,i})}{\max(\boldsymbol{A}_{:,i}) - \min(\boldsymbol{A}_{:,i})}. \tag{20}$$

In addition, we introduce $\boldsymbol{R} \in \mathbb{R}^{(N_t^{\mathrm{Train}}-p)\times N_r}$ the matrix of residuals $\boldsymbol{r}(t)$ as

$$\boldsymbol{R} = \begin{bmatrix} r_1(t_p) & r_2(t_p) & \cdots r_{N_r}(t_p) \\ r_1(t_{p+1}) & r_2(t_{p+1}) & \cdots r_{N_r}(t_{p+1}) \\ \vdots & \vdots & \ddots \\ r_1(t_{N_t^{\mathrm{Train}}-1}) & r_2(t_{N_t^{\mathrm{Train}}-1}) & \cdots r_{N_r}(t_{N_t^{\mathrm{Train}}-1}) \end{bmatrix}. \tag{21}$$

Each row of the input features in the matrix $\boldsymbol{A}$, *i.e.* current time instant and available POD coefficients at previous p time steps, corresponds to a row of target values in the matrix $\boldsymbol{R}$. The

[6]For instance, the training process might associate large weight values with large-valued inputs. Such a model with large weight values is often unstable.

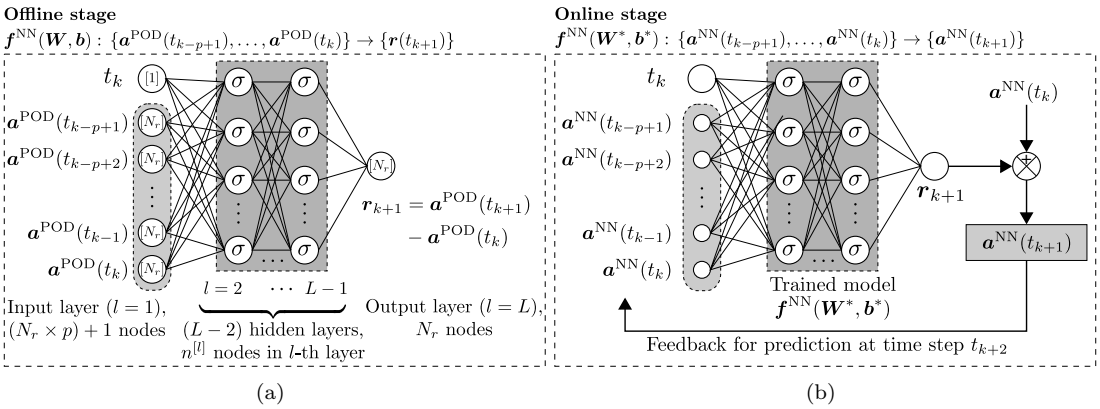


Figure 1: Architecture of NN-ROM $\boldsymbol{f}^{\mathrm{NN}}$ corresponding to (a) offline, and (b) online stages. In the offline training stage, the input features consist of the time (t) and POD modal coefficients ($\boldsymbol{a}^{\mathrm{POD}}(t)$), and the residuals ($\boldsymbol{r}(t)$) form the targets. For simplicity, only one node is shown for each input and output in vector form and the corresponding actual number of nodes is indicated within square brackets. For the hidden layers, σ is the activation function associated with each unit. In the online prediction stage, the learned forward model $\boldsymbol{f}^{\mathrm{NN}}(\boldsymbol{W}^*,\boldsymbol{b}^*)$ is used to obtain the sequential solution $\boldsymbol{a}^{\mathrm{NN}}(t)$.

regression problem (17) can be rewritten in matrix form as

$$\boldsymbol{R} \approx \widehat{\boldsymbol{R}} = \boldsymbol{f}^{\mathrm{NN}}(\boldsymbol{A};\boldsymbol{W},\boldsymbol{b}), \tag{22}$$

where we denote the output of the neural network as $\widehat{\boldsymbol{R}}$ in order to distinguish it from the matrix $\boldsymbol{R}$ used during training.

The NN-ROM (18) can be considered as a one-step numerical integrator which can be solved to obtain trajectories of the temporal POD coefficients $\boldsymbol{a}(t)$, similar to the Galerkin projection based POD-ROM [56]. Note that this integrator is "exact" in time (*i.e.* without temporal errors with respect to discretization, order of approximation, etc.) in the sense that the only error appears from the neural network approximation of the model operators defining the governing equations. This equation-free, non-intrusive regression framework offers an advantage by being purely data-driven, thus reducing the uncertainties associated with the model-form.

2.2.2. NN-ROM training and prediction

A supervised learning method, known as *backpropagation algorithm* (often simply called backprop), is used to train the NN-ROM. The gradient-based backprop approach attempts to optimize the internal weightings of the NN-ROM during training such that the error between input signals and the expected output signal is low. Here, the training is performed using the open-source machine learning library TensorFlow [1].

Backprop requires a DNN where each layer is densely connected, *i.e.* fully connected to the previous and next layers, as shown in Fig. 1(a). The DNN consists of L layers, with the number of nodes (also called neurons) in the l-th layer given as $n^{[l]}$ for all $l = 1,\ldots,L$. An input layer consists of $n^{[1]} = (N_r \times p) + 1$ nodes corresponding to features which include the current time instant $t_k \in \mathbb{R}^1$ and the available POD coefficients at previous p time steps $\boldsymbol{a}^{\mathrm{POD}}(t_{k'}) \in \mathbb{R}^{N_r \times 1}$, for all $k' = k-p+1,\ldots,k$. The output layer consists of $n^{[L]} = N_r$ nodes corresponding to the target residual vector $\boldsymbol{r}(t_{k'}) \in \mathbb{R}^{N_r \times 1}$. The hidden layers, indexed $l = 2,\ldots,L-1$, consist of $n^{[l]}$ hidden units each. It can also be noted that the the number of nodes $n^{[1]}$ and $n^{[L]}$ in the input and output layers are assigned vis-à-vis the number of columns in the feature matrix $\boldsymbol{A}$ (19) and the residual matrix $\boldsymbol{R}$ (21), respectively.

Supervised learning is performed during offline training to determine a set of regression parameters of the DNN model such that the error between the model output and targets is minimized. The regression parameters that are optimized are the linear weights $\boldsymbol{W}^{[l]}$ and biases $\boldsymbol{b}^{[l]}$ associated with

the neural network nodes in the hidden layers. The weights are coefficients associated with each of the node inputs to which the bias associated with the node is added. The result is passed through a node's predefined *activation function* (σ), indicated in the nodes in the hidden layers in Fig. 1(a), to obtain the node output. In this work, the rectified linear unit (ReLU) is used as the activation function, which offers ease of optimization with gradient-based methods [33] and is able to compute highly complex functions [55]. The weights and biases pertaining to each hidden layer can be stored in dictionaries as $\boldsymbol{W} \coloneqq \{\boldsymbol{W}^{[2]}, \ldots, \boldsymbol{W}^{[L-1]}\}$, with $\boldsymbol{W}^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}$, and $\boldsymbol{b} \coloneqq \{\boldsymbol{b}^{[2]}, \ldots, \boldsymbol{b}^{[L-1]}\}$, with $\boldsymbol{b}^{[l]} \in \mathbb{R}^{n^{[l]} \times 1}$, which together form the model training variables $\boldsymbol{\Theta} \coloneqq \{\boldsymbol{W}, \boldsymbol{b}\}$. The weights are initialized using normalized initialization [32] and the biases are initialized with small values (~ 0.1) to avoid loss of gradients caused by saturation of nodes [33]. The values of parameters in $\boldsymbol{\Theta}$ govern the accuracy with which the NN-ROM is able to map the features to the target.

After the initialization, *forward propagation* is performed by propagating the features $\boldsymbol{A}$ through layers of DNN and obtaining the residuals $\widehat{\boldsymbol{R}}$, which are compared with the targets $\boldsymbol{R}$ in order to optimize the training variables $\boldsymbol{\Theta}$. From another perspective, for a given input-target training pair $(\boldsymbol{A}, \boldsymbol{R})$, the training variables $\boldsymbol{\Theta}$ can be mapped to a loss function $\mathcal{L}(\widehat{\boldsymbol{R}}, \boldsymbol{R})$. The loss is generally formulated using disjoint, randomly selected subsets of the input features and targets called *minibatch*. Subsets of $\boldsymbol{A}$ provide N_{mb} minibatches $\{\boldsymbol{A}^{(j)} \in \mathbb{R}^{N_b \times (N_r \times p+1)} \,|\, j = 1, \ldots, N_{\text{mb}}\}$ of size $N_b < (N_t^{\text{Train}} - p)$. Similarly, the target minibatches $\{\boldsymbol{R}^{(j)} \in \mathbb{R}^{N_b \times N_r} \,|\, j = 1, \ldots, N_{\text{mb}}\}$ are also drawn from $\boldsymbol{R}$. Training on minibatches increases the number of optimization steps by a factor of N_{mb}. However, most optimization algorithms converge much faster in terms of the overall computation time if the approximate estimates of the gradient are rapidly computed over minibatches, as compared to when the exact gradients are computed over the entire training set [33]. The loss function for the j-th minibatch as a function of the DNN predictions and predefined targets is given as

$$\mathcal{L}(\widehat{\boldsymbol{R}}^{(j)}, \boldsymbol{R}^{(j)}) = \mathbb{E}[\|\widehat{\boldsymbol{R}}^{(j)} - \boldsymbol{R}^{(j)}\|^2] = \frac{1}{2N_b} \sum_{i=1}^{N_r} \sum_{k=1}^{N_b} (\widehat{R}_{k,i}^{(j)} - R_{k,i}^{(j)})^2, \quad \forall j = 1, \ldots, N_{\text{mb}}. \tag{23}$$

In order to limit the errors of the learning algorithm, a *regularization* factor is also defined in terms of the norm of the training variables as

$$\Omega(\boldsymbol{W}) = \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{w} = \frac{1}{2} \sum_{l=1}^{L} \sum_{q=1}^{n^{[l-1]}} \sum_{p=1}^{n^{[l]}} (W_{p,q}^{[l]})^2, \tag{24}$$

where $\boldsymbol{w}$ is the vector with weights contained in the dictionary $\boldsymbol{W}$ as elements. A L^2 regularized cost function (or objective function) for the j-the minibatch is defined by combining the loss function (23) and the penalty (24) as

$$\mathcal{C}^{(j)} = \mathcal{L}(\widehat{\boldsymbol{R}}^{(j)}, \boldsymbol{R}^{(j)}) + \lambda \Omega(\boldsymbol{W}), \quad \forall j = 1, \ldots, N_{\text{mb}}. \tag{25}$$

The hyperparameter $\lambda > 0$ weighs the contribution of the penalty term to the cost function with respect to the standard loss function. The weight-based regularization attempts to limit the generalization error of the learning algorithm by penalizing large weights that do not have a large contribution in the reduction of the cost function. The iteration over all the minibatches during learning constitutes one *training epoch*. The total number of training epochs N_{epochs} is defined by the user and should be sufficiently large such that convergence is reached.

After the forward pass, the backpropagation algorithm [68] allows the information from the scalar cost function $\mathcal{C}^{(j)}$ to flow backwards through the network for computing the gradients $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{C}^{(j)}$ and $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{C}^{(j)}$ with respect to the weights and biases respectively. The gradients computed from the minibatches of input data are subsequently used in a gradient descent-based optimization to update the training variables (weights and biases) such that the cost function (25) is minimized. In this work, the adaptive learning rate optimization algorithm known as Adam [44] is used. The algorithms of this class address the high sensitivity of the cost function to some directions in the space of training

variables by automatically adapting the learning rate throughout the course of training [33].

The performance of the optimization algorithm over the course of the training epochs is monitored by plotting an error metric evaluated over a validation dataset that is kept hidden from the training algorithm. The validation dataset is constructed by splitting the available input features and targets into disjoint subsets, one of size N_t^{Train}, which has been used to train the model variables, and another of size N_t^{Val} used for validation[7]. The validation set is used to estimate the generalization error during training, allowing the training hyperparameters to be updated accordingly. The model is evaluated throughout the training epochs to obtain the validation set estimates as

$$\begin{aligned} \boldsymbol{a}^{\text{Val,NN}}(t_{k+1}) = \boldsymbol{a}^{\text{Val,POD}}(t_k) + \boldsymbol{f}^{\text{NN}}(\boldsymbol{a}^{\text{Val,POD}}(t_{k-p+1}), \ldots, \boldsymbol{a}^{\text{Val,POD}}(t_k); \boldsymbol{W}^*, \boldsymbol{b}^*), \\ \forall\ p = 1, \ldots, N_t^{\text{Val}} - 1, \quad \text{and} \quad k = p-1, \ldots, N_t^{\text{Val}} - 1. \end{aligned} \tag{26}$$

Here, $\boldsymbol{f}^{\text{NN}}(\boldsymbol{W}^*, \boldsymbol{b}^*)$ is the DNN model with trained weights and biases. The number of time lags p is the same as the one used during training. The validation subset $\boldsymbol{a}^{\text{Val,POD}}(t_k) \in \mathbb{R}^{N_r \times 1}$ is considered to be separate from the training dataset, and the NN-ROM estimation is represented by $\boldsymbol{a}^{\text{Val,NN}}(t_{k+1})$. The error metric, in terms of root-mean-square error ($RMSE_{\text{Val}}$), used to gauge the performance of the optimization algorithm is given as

$$RMSE_{\text{Val}} = \sqrt{\frac{1}{N_t^{\text{Val}}} \sum_{k=0}^{N_t^{\text{Val}}-1} \|\boldsymbol{a}^{\text{Val,NN}}(t_k) - \boldsymbol{a}^{\text{Val,POD}}(t_k)\|_2^2}. \tag{27}$$

A comparison of this metric with the training error $RMSE_{\text{Train}}$ determines if the model is *underfitting* or *overfitting* the dataset during the course of the training. Underfitting occurs when the model is not able to obtain a sufficiently low training error value on the training set, while overfitting occurs when the gap between the training error and test error is too large. The objective is therefore to obtain a model with sufficiently low generalization error at the end of the offline training stage, which often requires tuning the regularization coefficient in (25) and the model hyperparameters.

Lastly, the trained NN-ROM is used to obtain online predictions of the POD coefficients using (18), as outlined in Fig. 1(b). The output of the model in terms of residuals $\boldsymbol{r}(t_k)$ is transformed to POD coefficients at the time step t_{k+1} by adding it to the POD coefficients at time step t_k, already available as inputs, as follows

$$\boldsymbol{a}^{\text{NN}}(t_{k+1}) = \boldsymbol{a}^{\text{NN}}(t_k) + \boldsymbol{r}(t_k). \tag{28}$$

The prediction of the dynamics at subsequent time steps $(t_p, \ldots, t_{N_t^{\text{Pred}}-1})$ is obtained by recursive feedback of the output POD coefficients to the input sequence of p time lag steps.

2.3. Ensemble Kalman filter (EnKF) augmented NN-ROM estimation

As the NN-ROM framework is generated using the latent space variables obtained solely from the snapshot data, it may suffer from the fundamental challenges of model reduction that are typically faced by the traditional POD-Galerkin models, among which is the accurate prediction of dynamics over long-term horizons. One way to mitigate the resulting inaccuracy of the estimations is to merge this framework into a data assimilation algorithm. The ability of the DNN framework to take the memory effect into account in order to predict the future state of the system makes it amenable to be used as a forecast model in sequential data assimilation (DA) algorithms. This also gives a possibility to provide an optimal estimation of the dynamical system, by taking into consideration the statistical confidences of both the model outputs and the observations, and nudging the ROM solution towards the true dynamics. Here, the NN-ROM is therefore augmented with the sequential DA method of ensemble Kalman filter (EnKF). The combined framework is hereafter referred as NN-ROM-DA.

[7] A third disjoint subset of size N_t^{Test}, called the test set, is also drawn from the available input features and targets to estimate the generalization error after the training has been completed. Usually the splitting is done such that 70% of the available data is used for learning the model, which is a combination of training (55% of total) and validation (15% of total) datasets. The remaining 30% is used for testing the trained model.

In the current context, DA is defined as a methodology for an optimal estimation of the states by combining both the background solution from the *models* and incoming imperfect information from the *observations* while taking into account the respective statistical confidences of the two complimentary, but incomplete and/or inaccurate, sources of information. A class of DA methods, known as sequential (probabilistic) DA [4] has been considered. This approach seeks solutions with minimum variance and provides rich information in terms of its mean value and probability distribution. The sequential approach of *Kalman filter* (KF) is a the most well-known and often-used toolbox for stochastic estimation. However, it assumes the uncertainties in the stochastic states and observations to be Gaussian distributed, and requires the model and observation operators to be linear. For handling models following non-Gaussian statistics and/or nonlinear operators, the use of *ensemble Kalman filter* (EnKF) was proposed by Evensen [29].

EnKF uses the Monte Carlo (MC) method to empirically represent the statistical characteristics of the estimator. The statistics during the course of sequential estimation are estimated from the propagation of a finite *ensemble* of samples (particles) generated using a forced random walk of the states of interest. The EnKF can be seen as a reduced-order KF as it only handles first two moments (mean and covariance) of the error statistics which loosely mimics the Gaussian filter. Generally, due to this truncation of statistics, the EnKF does not solve the Bayesian filtering problem. However, EnKF has been shown to provide a good approximate algorithm to the filtering problem.

In the NN-ROM-DA framework, the learned NN-ROM constitutes the forward model. The nonlinear, time-discrete model of the dynamical system and the observation equation at the time instant t_k are therefore given as

$$\begin{cases} \boldsymbol{a}^{\mathrm{NN}}(t_k) = \boldsymbol{a}^{\mathrm{NN}}(t_{k-1}) + \boldsymbol{f}^{\mathrm{NN}}(\{\boldsymbol{a}^{\mathrm{NN}}(t_l)\}_{l=k-p}^{k-1}; \boldsymbol{W}^*, \boldsymbol{b}^*) + \boldsymbol{\eta}_k, \quad \boldsymbol{\eta}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k), \\ \boldsymbol{y}^o(t_k) = \mathcal{H}_k(\boldsymbol{a}^{\mathrm{NN}}(t_k)) + \boldsymbol{\epsilon}_k^o, \quad \boldsymbol{\epsilon}_k^o \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k). \end{cases} \tag{29}$$

The term $\boldsymbol{\eta}_k \in \mathbb{R}^{N_r}$ is the model error of the true (unknown) process at time t_k. The error is represented here as a stochastic additive term and accounts for the cumulative errors in the the model parameters and the unresolved scales. The noisy discrete time observation is represented by the vector $\boldsymbol{y}^o \in \mathbb{R}^{N_o}$, where N_o is the number of components of observation (*e.g.* number of probes). The superscript "o" is used to differentiate the imperfect (noisy) observations from the actual (noise-free) observations $\boldsymbol{y}(t_k)$. The nonlinear observation function is given by $\mathcal{H}_k : \mathbb{R}^{N_r} \to \mathbb{R}^{N_o}$. An additive observation error term $\boldsymbol{\epsilon}_k^o \in \mathbb{R}^{N_o}$ is included which accounts for the instrument error, deficiencies in the observation operator, and the representation error arising from unresolved scales. The model error $\boldsymbol{\eta}_k$ and observation error $\boldsymbol{\epsilon}_k^o$ are assumed to be unbiased and uncorrelated. Next, the EnKF is performed sequentially by following a forecast-analysis (or prediction-correction) scheme.

2.3.1. Forecast step

During the *forecast step*, an ensemble of forecasted states is constructed by propagating each of the N_e members (particles) of the ensemble with the evolution model in (29). Each particle in the state-space evolves independently as

$$\boldsymbol{a}^{f,(n)}(t_k) = \boldsymbol{a}^{a,(n)}(t_{k-1}) + \boldsymbol{f}^{\mathrm{NN}}(\{\boldsymbol{a}^{a,(n)}(t_l)\}_{l=k-p}^{k-1}; \boldsymbol{W}^*, \boldsymbol{b}^*) + \boldsymbol{\eta}_k^{(n)}, \quad \boldsymbol{\eta}_k^{(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k), \tag{30}$$

where $n = 1, \ldots, N_e$ is the sample index. The superscript "NN" has been omitted from the temporal POD coefficients for simplicity. The vector $\boldsymbol{a}^{f,(n)}(t_k)$ represents the n-th member of the ensemble of forecast states at the instant t_k and $\boldsymbol{\eta}_k^{(n)}$ is the associated model error. The vector $\boldsymbol{a}^{a,(n)}(t_{k-1})$ is the corresponding member of the ensemble of corrected (analyzed) states at a previous time instant t_{k-1}. The unbiased empirical estimator of the forecast error covariance matrix $\boldsymbol{P}_k^f$, $\boldsymbol{P}_k^{f,e}$, is obtained as

$$\boldsymbol{P}_k^{f,e} = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} (\boldsymbol{a}^{f,(n)}(t_k) - \overline{\boldsymbol{a}}^f(t_k))(\boldsymbol{a}^{f,(n)}(t_k) - \overline{\boldsymbol{a}}^f(t_k))^\top, \tag{31}$$

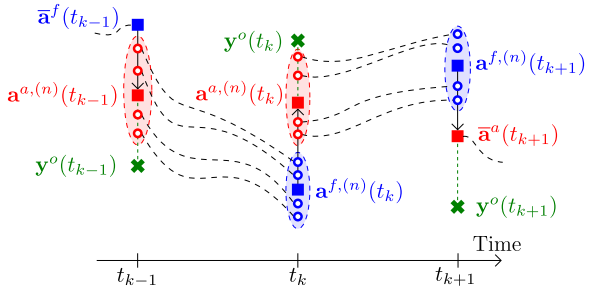


Figure 2: Schematic of the EnKF data assimilation method. The filled symbols represent the mean values and the hollow circles represent the ensemble members; the ensemble set itself is indicated by the shaded region around the respective mean values of the analyzed (red) and forecast (blue) states. The dashed lines represent the trajectories of the particles in the state ensemble propagated by the model from one time step to the next. The corrections of the propagated values obtained at the observation times is indicated by black arrows.

where the empirical mean $\overline{\boldsymbol{a}}^f(t_k)$ of the ensemble of forecast state is obtained as

$$\overline{\boldsymbol{a}}^f(t_k) = \frac{1}{N_e}\sum_{n=1}^{N_e} \boldsymbol{a}^{f,(n)}(t_k). \tag{32}$$

The observations are considered as random variables. This is ensured by keeping the value centered around the actual observation and adding a random perturbation drawn from a Gaussian distribution with covariance $\boldsymbol{R}_k$ such that

$$\boldsymbol{y}^{o,(n)}(t_k) = \boldsymbol{y}^o(t_k) + \boldsymbol{\epsilon}_k^{o,(n)}, \quad \boldsymbol{\epsilon}_k^{o,(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k). \tag{33}$$

The vector $\boldsymbol{y}^{o,(n)}(t_k)$ represents the n-th member of the observation ensemble at the instant t_k and $\boldsymbol{\epsilon}_k^{o,(n)}$ represents the associated perturbation. The unbiased (zero-mean) empirical estimator $\boldsymbol{R}_k^e$ of the observation error covariance matrix is given as

$$\boldsymbol{R}_k^e = \mathbb{E}[\boldsymbol{\epsilon}_k^{o,(n)}(\boldsymbol{\epsilon}_k^{o,(n)})^\top] = \frac{1}{N_e - 1}\sum_{n=1}^{N_e} \boldsymbol{\epsilon}_k^{o,(n)}(\boldsymbol{\epsilon}_k^{o,(n)})^\top. \tag{34}$$

As the number of MC elements tends to infinity, the empirical estimator $\boldsymbol{R}_k^e$ tends to the full-rank observation error covariance matrix $\boldsymbol{R}_k$.

2.3.2. Analysis step

The *analysis step* consists of updating each member from the ensemble of forecasted states using the actual observation by applying a linear correction as follows

$$\begin{aligned} \boldsymbol{a}^{a,(n)}(t_k) &= \boldsymbol{a}^{f,(n)}(t_k) + \boldsymbol{K}_k^e(\boldsymbol{y}^{o,(n)}(t_k) - \boldsymbol{y}^{f,(n)}(t_k)) \\ &= \boldsymbol{a}^{f,(n)}(t_k) + \boldsymbol{K}_k^e(\boldsymbol{y}^o(t_k) + \boldsymbol{\epsilon}_k^{o,(n)} - \mathcal{H}_k(\boldsymbol{a}^{f,(n)}(t_k))). \end{aligned} \tag{35}$$

The vector $\boldsymbol{a}^{a,(n)}(t_k)$ represents the n-th member of the ensemble of analyzed state. The ensemble Kalman gain $\boldsymbol{K}_k^e$ is expressed in terms of the ensemble covariances as

$$\boldsymbol{K}_k^e = \boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top(\boldsymbol{H}_k\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top + \boldsymbol{R}_k)^{-1}. \tag{36}$$

This terminates the analysis step at the instant t_k. We note that the linearized evolution model has not been used in the algorithm which makes it useful in a significantly nonlinear regime. The prediction-correction trajectory of the model state vector is shown in Fig. 2.

2.3.3. Kalman gain for nonlinear observation operator

The Kalman gain (36) requires the tangent linear model $\boldsymbol{H}_k$ of the observation operator $\mathcal{H}_k$ to be determined at each time step t_k for the terms $\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top$ and $\boldsymbol{H}_k\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top$, which can be expensive. However, these two terms can be estimated by using the full nonlinear observation operator. Following the derivation provided by Asch et al. [4], a fully ensemble-based Kalman gain is given as

$$\boldsymbol{K}_k^e = \boldsymbol{P}_{ay,k}^{f,e}(\boldsymbol{P}_{yy,k}^{f,e} + \boldsymbol{R}_k^e)^{-1}, \tag{37}$$

where the sample estimate $\boldsymbol{P}_{ay,k}^{f,e}$ of the term $\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top$ is given in terms of a cross-covariance of the forecast state $\boldsymbol{a}^{f,(n)}(t_k)$ and forecast observation $\boldsymbol{y}^{f,(n)}(t_k)$ as

$$\boldsymbol{P}_{ay,k}^{f,e} \coloneqq \frac{1}{N_e-1}\sum_{n=1}^{N_e}(\boldsymbol{a}^{f,(n)}(t_k) - \overline{\boldsymbol{a}}^f(t_k))(\boldsymbol{y}^{f,(n)}(t_k) - \overline{\boldsymbol{y}}^f(t_k))^\top, \tag{38}$$

and the sample estimate $\boldsymbol{P}_{yy,k}^{f,e}$ of the term $\boldsymbol{H}_k\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top$ is given in terms of a covariance of the forecast observation $\boldsymbol{y}^{f,(n)}(t_k)$ as

$$\boldsymbol{P}_{yy,k}^{f,e} \coloneqq \frac{1}{N_e-1}\sum_{n=1}^{N_e}(\boldsymbol{y}^{f,(n)}(t_k) - \overline{\boldsymbol{y}}^f(t_k))(\boldsymbol{y}^{f,(n)}(t_k) - \overline{\boldsymbol{y}}^f(t_k))^\top. \tag{39}$$

When the number of measurements is greater than the number of ensemble members, the inverse term in the calculation of the Kalman gain may become singular. In this case, a pseudo-inverse based on the singular value decomposition can be employed. However, the advantage of this representation mainly resides in the fact that both the linearization and the calculation of the covariance matrix of the prediction error are not required, therefore resulting in a considerable reduction in the computation cost and requirement of storage space.

3. Results

In this section, the neural network-based non-intrusive reduced-order modeling framework presented in Sec. 2 is applied for time series prediction of dynamical systems. In Sec. 3.1, the ability of the multistep, residual-based NN-ROM framework to predict the dynamics of the Lorenz-63 system is first demonstrated. In Sec. 3.2, the NN-ROM is then applied to obtain predictions of the dynamics of a cylinder wake flow at high Reynolds number. In addition, the sequential NN-ROM is augmented with the EnKF data assimilation algorithm to obtain long-term predictions.

3.1. Non-intrusive modeling of Lorenz-63 system

Before implementing the proposed framework for fluid flow applications, the performance of NN-ROM is evaluated here for the nonlinear and chaotic Lorenz-63 system.

3.1.1. Training and model-evaluation dataset

The Lorenz-63 system was developed to describe the phenomenon of natural convection in a heated rectangular cavity. The quadratic system in terms of the state vector $\boldsymbol{a} = [a_1\ a_2\ a_3]^\top$ of size $N_r = 3$ is given as

$$\begin{cases} \dot{a}_1(t) = \sigma(a_2(t) - a_1(t)), \\ \dot{a}_2(t) = a_1(t)(\rho - a_3(t)) - a_2(t), \\ \dot{a}_3(t) = a_1(t)a_2(t) - \beta a_3(t). \end{cases} \tag{40}$$

Here, the initial state is set as $\boldsymbol{a}(t_0 = 0) = [a_1(0)\ a_2(0)\ a_3(0)]^\top = [-8\ 7\ 27]^\top$. For the parameter values of $[\sigma\ \rho\ \beta] = [10\ 28\ 8/3]$, the system exhibits a chaotic dynamics characterized by a limit cycle and a strange attractor. The data is generated in the time range $t = [0, 20]$ with a time step of 0.01, giving $N_t = 2001$ snapshots. The dataset is split such that state evolution corresponding to the initial 85% of the time steps is used for training, *i.e.* $N_t^{\text{Train}} = 1700$, and the remaining data is used

for validation, *i.e.* $N_t^{\text{Val}} = 301$. At the end of the training stage, the performance of the trained model is evaluated based on the estimation capability with respect to the full solution trajectory, such that $N_t^{\text{Test}} = 2001$.

3.1.2. Hyperparameter selection

The training of NN-ROM, constructed as a feedforward network discussed in Sec. 2.2, requires making some design decisions. The performance of the NN-ROM is largely dictated by the selection of hyperparameters used to construct and optimize the neural network. The architecture is defined by the number of hidden layers, the connection between the consecutive layers, and the number of hidden units in each layer. The output of the hidden layer is governed by the choice of the activation function. The gradient-based learning also requires defining an appropriate cost function and selecting an optimizer.

Finding an optimal set of hyperparameters which minimizes the loss function over a hyperparameter space is a challenging task given the substantial number of free parameters involved. In this work, the hyperparameters have been selected heuristically by monitoring the influence of the assigned values on model performance 'on the fly'. More sophisticated methods for hyperparameter selection are random search [15] and Bayesian optimization [16]. However, following Goodfellow et al. [33], general guidelines can be formulated.

1. Number of layers (L): Increasing L augments the model's capacity to represent more complex functions with a simultaneous increase in the computational cost of training.
   ► In this example, the network is constructed with $L = 7$ layers, which includes the input and output layers and 5 hidden layers.

2. Number of units in each layer ($n^{[l]}$): Increasing $n^{[l]}$ has the similar effect as L on the representation ability of the model. The number of units in the input and output layers is fixed for a problem based on the size of the feature and target. Note that for a large value of either L or $n^{[l]}$, the model has a chance of overfitting the training data.
   ► In this example, the number of units in the input layer is $n^{[1]} = N_r p + 1 = 3p + 1$, where p is the number of past input steps defining the multistep framework (see Sec. 2.2.1). Note that the '+1' unit corresponds to the time variable and no additional units are assigned for parameters as the training is performed using data belonging to the solution set corresponding to the same parameters. The size of the hidden layers is assigned to $n^{[l=2,\ldots,6]} = 128$ units. The output layer contains $n^{[7]} = N_r = 3$ units.

3. Activation function (σ): The output of the hidden units is dictated by the activation function. The function σ must be chosen such that the layers do not saturate, *i.e.* the gradients of the cost function do not become very flat and remain large enough to guide the learning algorithm.
   ► In this example, the widely used and easy to optimize rectified linear unit (ReLU) is used as the activation function (see Sec. 2.2).

4. Size of minibatch (N_b): The stability and speed of the learning algorithm are generally optimized by dividing the input features (here, time and state evolution) and the target into subsets, the training being made on these minibatches. Minibatches typically contain two to several hundred samples, although for large models the choice may be constrained by computational resources.
   ► In this example, the minibatches are randomly selected disjoint subsets of the training data of size $N_b = 501$.

5. Number of minibatches (N_{mb}): The number of minibatches dictates the number of updates of the training variables during each training epoch. The number is fixed by monitoring the learning curve. A higher value of N_{mb} may lead the model to overfit the training data.
   ► In this example, $N_{\text{mb}} = 10$ is used.

6. Learning rate (α): The success of the gradient-based optimization depends on the choice of the learning rate. A small α slows down the computation while a large α may lead to overshooting

the local minima of the cost function. The value of α is fixed by monitoring the learning curve.
► In this example, the learning rate $\alpha = 0.001$ is used.

7. Regularization parameter (λ): The generalization error of the NN-ROM is tuned by the regularization parameter in the cost function. A lower value of λ may make the model more prone to overfit to the training data, while a higher λ may cause the weights of the neural network to vanish, severely underfitting the training data. The value of λ is fixed by monitoring the learning curve.
► In this example, the regularization parameter $\lambda = 1.0 \times 10^{-5}$ is used.

8. Number of training epochs (N_{Epochs}): The number of training epochs is related to the learning rate. As N_{Epochs} increases, the model transforms from underfitting, to optimal, and to potentially overfitting the training data. The value of N_{Epochs} is fixed by monitoring the learning curve.
► In this example, the model is trained over $N_{\text{Epochs}} = 20000$ cycles.

Tab. 1 summarizes the choice of these hyperparameters.

DNN architecture ($p = 1, 5, 10$)				Training data	Minibatch		Optimization parameters		
$n^{[1]}$	$n^{[2,\ldots,6]}$	$n^{[7]}$	σ	N_t^{Train}	N_b	N_{mb}	α	λ	N_{Epochs}
$3p+1$	128	3	ReLU	1700	501	10	1.0×10^{-3}	1.0×10^{-5}	20000

Table 1: Hyperparameters of NN-ROM for the Lorenz-63 system.

3.1.3. Online training

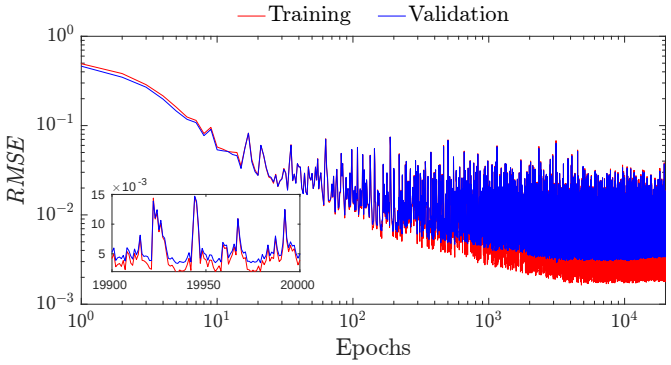


Figure 3: Evolution of the training and validation errors during the training epochs of NN-ROM for the Lorenz-63 system ($p = 10$). The error evolution in the last 100 epochs is also shown in the inset.

Once the hyperparameters are assigned, the training variables of NN-ROM (weights and biases) are optimized over the N_{Epochs} training epochs using the Adam algorithm (see Sec. 2.2). One training epoch involves the evaluation of the gradients of the cost function (25) for each minibatch by backpropagation. The deep neural network utilizes high-capacity architecture which, due to a large number of training variables, are susceptible to overfitting even when sufficient training data is available. This is monitored during the course of training using the learning curves which are plots of the error metric (27) calculated for the training set ($RMSE_{\text{Train}}$) and the validation set ($RMSE_{\text{Val}}$) as a function of the training epochs.

In order to demonstrate the influence of the number of past time steps p in the input on the performance of NN-ROM, three values of p are considered, namely $p = 1, 5, 10$. The learning curves for the training of NN-ROM for the Lorenz-63 system with $p = 10$ are shown in Fig. 3. Similar curves are obtained for other values of p. It is observed that the error for the validation set drops consistently with the corresponding decrease in the error for the training set. This implies that the hyperparameters selected for designing and training the neural network lead to a NN-ROM with

	$p=1$	$p=5$	$p=10$	$p=100$
Average *NRMSE*	0.7524	0.5481	0.4905	0.6282

Table 2: Time averaged *NRMSE* in the testing window corresponding to different numbers of past input steps (p) for the Lorenz-63 system.

a low generalization error, *i.e.* the model neither underfits not overfits the data used for training. Note that the noisy learning curve is common to the stochastic gradient-descent methods which can be attributed to the frequent updates of the training variables, allowing the model to avoid local minima and hence, an early convergence.

3.1.4. Offline prediction

To elucidate the multistep, residual-based framework of the neural network, the learned NN-ROM models, corresponding to the three values of p in the input training data, are used to obtain the estimation of the state trajectories $\boldsymbol{a}^{\mathrm{NN}}(t)$ of the Lorenz-63 system using the forward model (18). The comparison of the estimated trajectories with the reference solution trajectories is shown in Fig. 4. The Lorenz system has a chaotic behavior. Hence, a small error in the estimated state of the system can lead to a larger error in the subsequent time steps. The time period for which the reproduced trajectory is the same as the reference trajectory varies for different values of p. In general, the inclusion of temporal history of the state of the system in the input to NN-ROM, *i.e.* $p > 1$, leads to an improved estimation where the reference trajectory is followed over a longer time span.

The performance of NN-ROM can be quantified in terms of the normalized root-mean-square error (*NRMSE*). Here, the *NRMSE* is calculated with respect to the reference solution $\boldsymbol{a}^{\mathrm{Ref}}(t)$, which is the solution obtained from the numerical integration of the Lorenz-63 system, as

$$NRMSE(t) = \frac{\sqrt{\sum_{i=1}^{N_r}(a_i^{\mathrm{NN}}(t) - a_i^{\mathrm{Ref}}(t))^2}}{\sqrt{\sum_{i=1}^{N_r}(a_i^{\mathrm{Ref}}(t))^2}}, \tag{41}$$

where $N_r = 3$. The averaged *NRMSE* over the testing time span is given in Tab. 2 for the three values of p. An extreme case where $p = 100$ time steps is also included to observe the effect of a long time past window on the accuracy of the estimation. It is observed that the estimation is more accurate for the cases with $p > 1$. This result encourages to take the memory effect into account in the input of NN-ROM for a more accurate reproduction of the transient dynamics. The effect is however not monotonous (*i.e.* higher number of time steps p does not correspond necessary to a lower value of error), as indicated by the higher value of estimation error corresponding to $p = 100$ as compared to $p = 10$. This is exacerbated by the sensitivity of the Lorenz-63 system to the state space, *i.e.* featuring a positive Lyapunov exponent. The number of past time steps (p) is thus a hyperparameter which can be optimized to obtain a sufficiently accurate estimation framework. For simplicity, this parameter has been selected heuristically in the subsequent test cases.

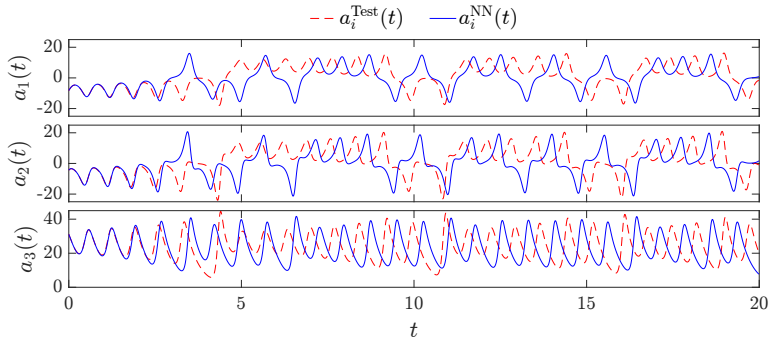
3.2. Application to experimental cylinder wake flow

In this section, we consider a cylinder wake flow configuration at high Reynolds number. A long-term estimation of the dynamics is envisaged by combining the sequential updates obtained by NN-ROM and data assimilation.

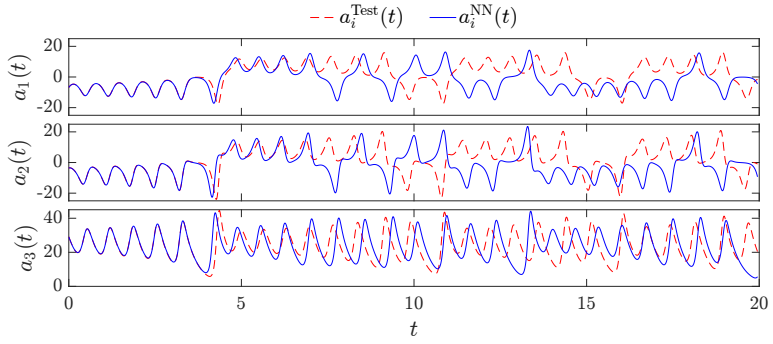
3.3. Snapshot dataset

The dataset were obtained from the experimental work performed by Benard et al. [11]. The experimental setup is briefly described in this section.

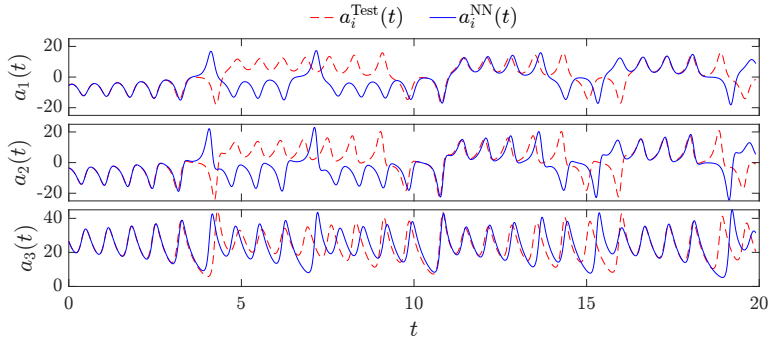
The cylinder wake flow was experimentally studied in an open Eiffel-type wind tunnel. The cylinder, which spans the entire test section, has a diameter of $D = 40$ mm and an aspect ratio of $L/D = 7.5$. The upstream velocity in the measurement section was set as $U_\infty = 5.6$ m/s, giving



(a) $p = 1$



(b) $p = 5$



(c) $p = 10$

Figure 4: Evolution of the states $\boldsymbol{a}(t)$ of the Lorenz-63 system obtained from the trained NN-ROM in the testing window corresponding to different numbers of past input steps (p) and compared with the true reference trajectories.

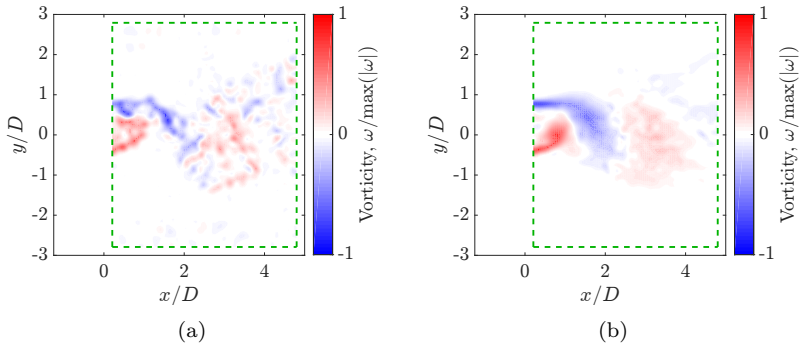


Figure 5: Normalized vorticity field at a time instant $t = 100$ s obtained from the PIV measurements (a), and reconstructed using the 10 most energetic POD modes (b) for the cylinder wake flow at $Re = 1.5 \times 10^4$. The PIV measurement field in the near-wake region is depicted by the green box.

a Reynolds number based on the cylinder diameter of $Re = 1.5 \times 10^4$. The Reynolds number is therefore an order of magnitude lower than the critical Reynolds number ($Re_c \sim 2.0 \times 10^5$, see Williamson 80) at which turbulence transition occurs in the detached shear layers. The flow over the cylinder remains in the sub-critical regime with laminar boundary layer separation, while a wide turbulent wake develops downstream. The natural frequency of the vortex shedding, obtained from the post-processing of the available data, is equivalent to a Strouhal number, based on the cylinder diameter and freestream velocity, of $St = 0.18$ (or a non-dimensional time period of $T_s U_\infty / D = 5.39$).

The two components of velocity in the near wake of the cylinder were measured with a particle image velocimetry (PIV) system consisting of a fast CCD camera (Photron, APX-RS), a fast dual oscillator single-head laser (Quantronix, Darwin-Duo), a synchronization unit (EG, R&D Vision) and an acquisition PC. The cylinder was illuminated in the mid-span by a 1 mm thin laser sheet. The measurement field, shown in Fig. 5(a), covers the region defined by the bounds $0.2 < x/D < 4.8$ and $-2.8 < y/D < 2.8$. A CCD sensor of resolution 1024×1024 pixel^2 was used. The acquisition was carried out at a frequency of 1000 Hz, which corresponds to a non-dimensional time interval between two consecutive snapshots of $\Delta t U_\infty / D = 0.14$. This implies that, in theory, a shedding cycle is discretized by about 38 instantaneous snapshots. The velocity vectors were obtained using LaVision's Davis software by computing cross-correlations with windows ranging in size from 64×64 pixel^2 to 16×16 pixel^2 in the final pass, each with a 50% overlap. This results in a uniform grid of size $N_y \times N_x = 108 \times 89$, *i.e.* $N_\chi = N_x N_y = 9612$ nodes, with a spatial resolution of $\Delta x = \Delta y = 2.09$ mm. The overall time sequence spans the time range $t \in [0, 1]$ s (*i.e.* 1001 snapshots), which is equivalent to $t U_\infty / D \in [0, 140]$ in terms of the non-dimensional variable.

3.3.1. NN-ROM setup and estimation

POD is first performed on the snapshots of velocity and the most dominant $N_r = 10$ modes, which represent 79% of total energy, are preserved for reduced-order modeling. The temporal POD coefficients corresponding to the initial 701 time steps are used for training the NN-ROM. The time series is split such that the data belonging to the first $N_t^{\text{Train}} = 595$ snapshots is used for training and that belonging to the next $N_t^{\text{Val}} = 106$ snapshots is used for validation of the learned model. The summary of the manually assigned hyperparameters associated with the construction and training of the neural network is given in Tab. 3. The NN-ROM is based on using the history of POD coefficients from $p = 10$ previous time steps in the input feature set. The details regarding the rest of the setup and optimization of the model are the same as the ones discussed in Sec. 3.1.2.

The learned NN-ROM is then used to obtain estimates of the POD coefficients. Due to the limited size of the data, the testing is performed on the whole time range of POD coefficients, $N_t^{\text{Test}} = 1001$. The evolution of the temporal POD coefficients for the testing dataset is shown in Fig. 6. Overall, the amplitudes of the estimated coefficients remain consistent with respect to the reference trajectory and the evolution remains bounded over the full testing time span. It is observed

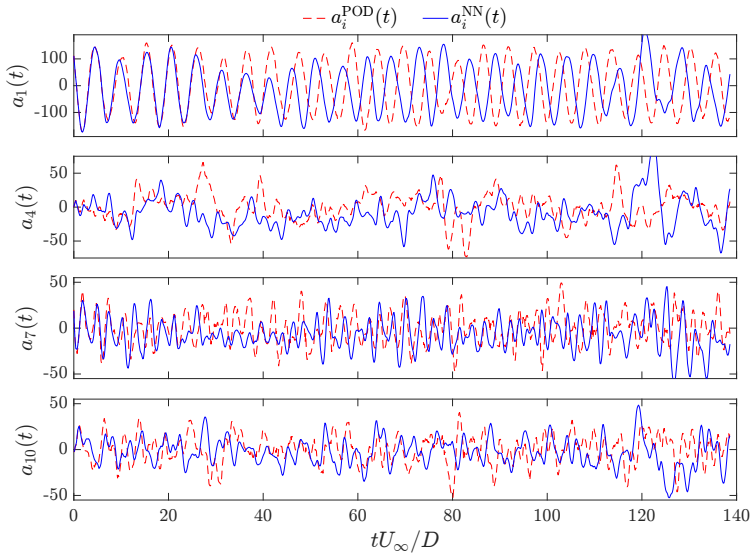


Figure 6: Evolution of the temporal POD coefficients $a_i(t)$ $(i = 1, 4, 7, 10)$ for the testing dataset for the cylinder wake flow at $Re = 1.5 \times 10^4$. The results are obtained from the NN-ROM and compared with the reference trajectory.

DNN architecture ($p = 10$)				Training data	Minibatch		Optimization parameters		
$n^{[1]}$	$n^{[2,\ldots,6]}$	$n^{[7]}$	σ	N_t^{Train}	N_b	N_{mb}	α	λ	N_{Epochs}
$10p+1$	256	10	ReLU	595	401	10	1.0×10^{-3}	1.0×10^{-2}	20000

Table 3: Hyperparameters of the NN-ROM for the cylinder wake flow at $Re = 1.5 \times 10^4$.

that in the short-term, the trajectory of the POD coefficients estimated from NN-ROM follows the reference trajectory of the POD coefficients directly obtained from the snapshots. However, the trajectories start to deviate after about the first 20 time units.

3.4. EnKF augmented NN-ROM estimation

The NN-ROM learned in Sec. 3.3.1 is augmented with EnKF data assimilation. The framework is hereafter referred as NN-ROM-DA. The data assimilation is performed using point measurements of the streamwise velocity component at few locations in the flow field. As before, the observation vector in (29) is defined accordingly as the streamwise component of the velocity fluctuation, *i.e.* $\boldsymbol{y}^o(t_k) = \{u'_x(\boldsymbol{\chi}_\ell, t_k)\}_{\ell=1}^{N_o}$, where $N_o = 6$ is the number of fictive probes and $\boldsymbol{\chi}_\ell$ is the coordinate vector of the ℓ-th probe. The locations of the probes in the flow field are shown in Fig. 7 and the coordinates is listed in Tab. 4. The observation equation (42) maps the temporal POD coefficients obtained from the forward model to the observed velocity fluctuations. The spatial POD mode at each probe location serves as the linear observation operator $\boldsymbol{H}$ to map the temporal POD coefficients obtained from the forward model to the observed velocity fluctuations. The observation model (29) can thus be reformulated as,

$$\boldsymbol{u}'_x(\boldsymbol{\chi}, t_k) = \begin{bmatrix} u'_x(\chi_1, t_k) \\ \vdots \\ u'_x(\chi_{N_o}, t_k) \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi_1^x(\chi_1) & \cdots & \Phi_{N_{\mathrm{Gal}}}^x(\chi_1) \\ \vdots & \ddots & \vdots \\ \Phi_1^x(\chi_{N_o}) & \cdots & \Phi_{N_{\mathrm{Gal}}}^x(\chi_{N_o}) \end{bmatrix}}_{\boldsymbol{H}} \boldsymbol{a}(t_k), \tag{42}$$

where $\Phi_i^x(\chi_j)$ $(i = 1, \cdots, N_{\mathrm{Gal}})$ represents the streamwise component of the i-th spatial POD mode of the velocity fluctuations at the probe location χ_j.

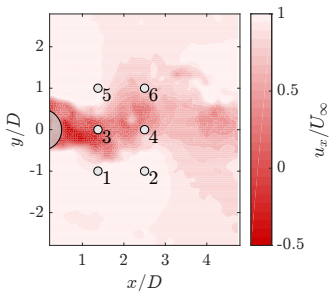


Figure 7: Visual representation of the probe locations for the streamwise velocity component measurements used for parameter identification. The flow field is illustrated with a contour plot of the instantaneous streamwise velocity at $tU_\infty/D = 50$.

ℓ	$\boldsymbol{\chi}_\ell := (x_\ell/D, y_\ell/D)$
1	$(1.4, -1.0)$
2	$(2.5, -1.0)$
3	$(1.4, 0.0)$
4	$(2.5, 0.0)$
5	$(1.4, 1.0)$
6	$(2.5, 1.0)$

Table 4: Probe coordinates.

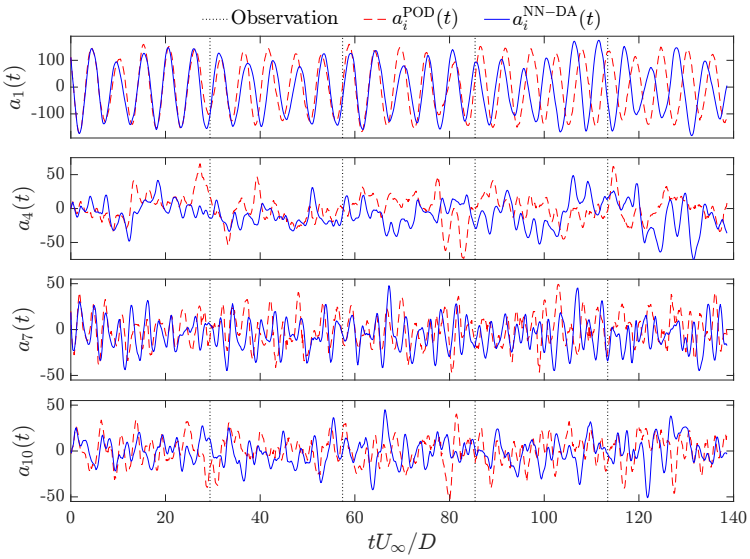


Figure 8: Same comparison as in Fig. 6 but for the evolution obtained from NN-ROM-DA with $p^{\mathrm{DA}} = 1$ consecutive assimilation.

An additional hyperparameter, p^{DA}, is introduced which is defined as the number of consecutive observations used in assimilation. The justification of this multistep assimilation approach is to augment the capability of the NN-ROM to take into consideration the memory effect of the temporal dynamics. In this case study, two values of p^{DA} are considered, namely $p^{\mathrm{DA}} = 1$, which corresponds to the classical single update method, and $p^{\mathrm{DA}} = 5$. Note that both are smaller than the number of time steps passed as the input to NN-ROM, $p = 10$.

For the DA augmented approach, the observations are assimilated every $\Delta t_o U_\infty/D = 28.1$. This value is 200 times larger than the PIV acquisition step size and spans 5 cycles of vortex shedding. This equates to performing 4 steps of equispaced assimilations over the estimation window (1001 time steps).

The time evolution of the temporal POD coefficients obtained by NN-ROM-DA for the two values of consecutive assimilations $p^{\mathrm{DA}} = 1$ and $p^{\mathrm{DA}} = 5$ is shown in Fig. 8 and Fig. 9, respectively. The plots show that the estimated trajectories of the POD coefficients obtained using NN-ROM-DA follow the reference trajectories more accurately as compared to the estimates obtained from NN-ROM (see Fig. 6). As observed, after each observation step, the EnKF assimilation procedure improves the accuracy of the trajectories in the time span between the observations. Moreover, a

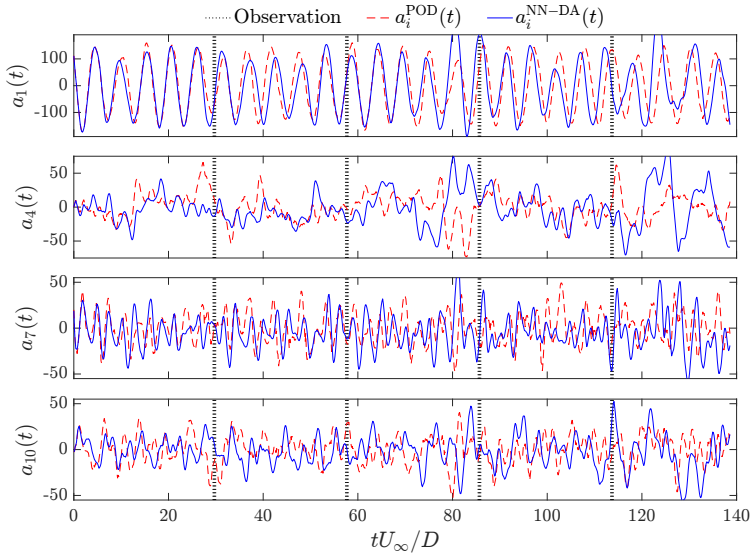


Figure 9: Same comparison as in Fig. 6 but for the evolution obtained from NN-ROM-DA with $p^{\mathrm{DA}} = 5$ consecutive assimilations.

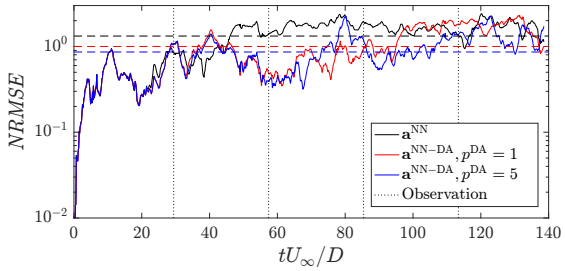
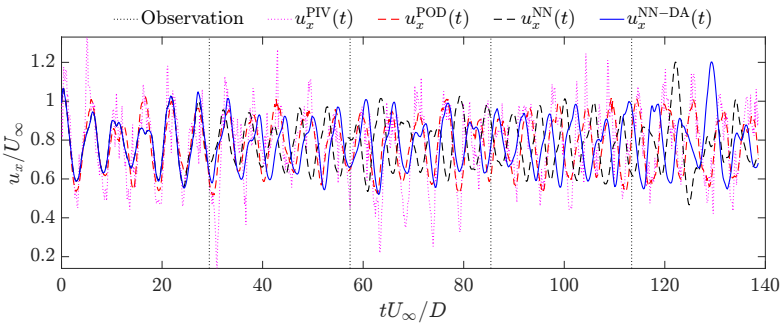


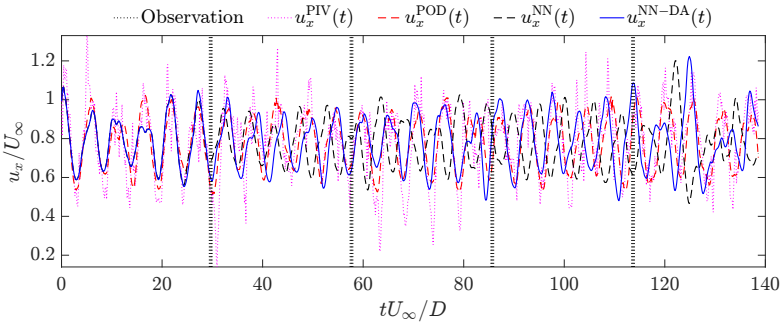
Figure 10: Time evolution of *NRMSE* in the testing window for the POD coefficients obtained from NN-ROM and NN-ROM-DA with $p^{\mathrm{DA}} = 1$ and 5. Cylinder wake flow configuration at $Re = 1.5 \times 10^4$. The time averaged *NRMSE* values over the testing window are indicated by horizontal dashed lines. The dotted lines represent the time steps at which the observations are initiated for the two NN-ROM-DA frameworks.

slight improvement in accuracy can be observed when multiple consecutive observations are available ($p^{\mathrm{DA}} = 5$) as compared to when only single observation is available at each assimilation step ($p^{\mathrm{DA}} = 1$). This indicates that the NN-ROM estimates benefit from the greater number of assimilated states available in the memory for forward propagation. In Fig. 10, the *NRMSE* defined by (41) is compared for the NN-ROM (*i.e.* without assimilation) and for the NN-ROM-DA with $p^{\mathrm{DA}} = 1$ and 5. A more accurate estimation is obtained from NN-ROM-DA than NN-ROM in the majority of the testing time span. The corresponding average errors are also lower.

The estimated POD coefficients are finally used to reconstruct the two-component velocity fields. In Fig. 11, the time evolution of the streamwise components of velocity, obtained from the NN-ROM estimates ($u_x^{\mathrm{NN}}(t)$) and the NN-ROM-DA estimates ($u_x^{\mathrm{NN-DA}}(t)$), are compared at a location in the flow field with the reference trajectories determined from the calculated POD coefficients ($u_x^{\mathrm{POD}}(t)$) and the PIV experiments ($u_x^{\mathrm{PIV}}(t)$). Owing to the more accurate estimation of the POD coefficients, the corresponding reconstruction of the velocity is also more accurate for the estimates from NN-ROM-DA as compared to those obtained from NN-ROM. Moreover, the phase shift observed in the time span between the assimilation steps is minimized when multiple consecutive observations are used to nudge the estimated trajectory towards the reference trajectory. To gauge the performance



(a) $p^{\mathrm{DA}} = 1$ consecutive assimilation



(b) $p^{\mathrm{DA}} = 5$ consecutive assimilation

Figure 11: Time evolution of the streamwise velocity components at a location $(x/D, y/D) = (4.8, 0.9)$ in the flow field. Comparison of the trajectory obtained from the PIV measurements with the reconstructions using the POD coefficients determined from the NN-ROM and NN-ROM-DA with $p^{\mathrm{DA}} = 1$ (a), and $p^{\mathrm{DA}} = 5$ (b). Cylinder wake flow configuration at $Re = 1.5 \times 10^4$.

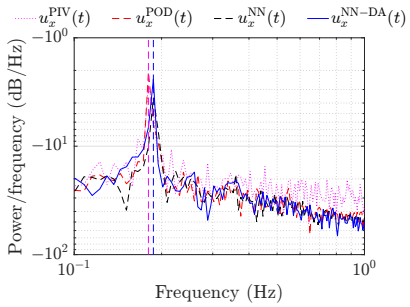


Figure 12: Power spectrum densities of the velocity at a location $(x/D, y/D) = (4.8, 0.9)$ in the flow field. Comparison of the results obtained from PIV measurements and the reconstructed velocity signals using the POD coefficients determined from the NN-ROM and NN-ROM-DA with $p^{\mathrm{DA}} = 5$. Cylinder wake flow configuration at $Re = 1.5 \times 10^4$. The peak frequencies of the measured signal and that reconstructed from the augmented ROM are indicated by vertical dashed lines.

in frequency domain, the power spectra of the reconstructed velocity signals using POD, NN-ROM and NN-ROM-DA (with $p^{\mathrm{DA}} = 5$) are compared in Fig. 12 with that using the signal obtained from experiment. It is observed that the peak frequency of the signal obtained from experiments (0.18 Hz) is correctly captured by the estimated dynamics using NN-ROM-DA (0.1874 Hz).

In order to quantify the performance of the different approaches, the normalized root-mean-square error of the velocity magnitude $|\boldsymbol{u}|$ with respect to the measured values is calculated over the

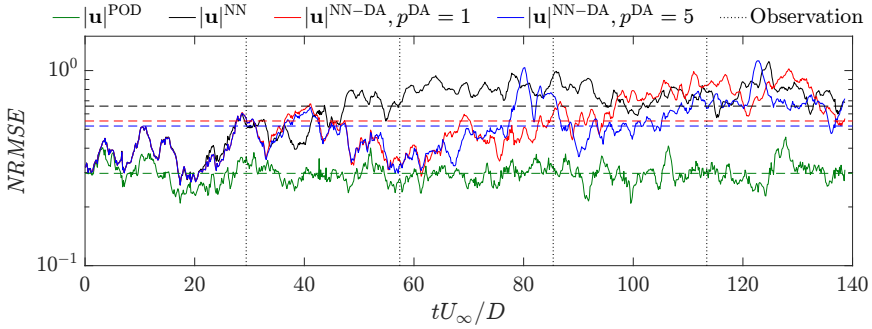


Figure 13: Time evolution of *NRMSE* in the testing window for the velocity magnitude calculated from the POD coefficients obtained from the reference values and those determined from the NN-ROM and NN-ROM-DA with $p^{\mathrm{DA}} = 1$ and 5. Cylinder wake flow configuration at $Re = 1.5 \times 10^4$. The time averaged *NRMSE* values over the testing window are indicated by horizontal dashed lines. The dotted lines represent the time steps at which the observations are initiated for the two NN-ROM-DA frameworks.

Average *NRMSE* $NRMSE(t;\boldsymbol{x},\boldsymbol{x}^{\mathrm{Ref}}) = \sqrt{\overline{(\boldsymbol{x}-\boldsymbol{x}^{\mathrm{Ref}})^2}/\overline{(\boldsymbol{x}^{\mathrm{Ref}})^2}}$		POD	NN-ROM	NN-ROM-DA	
$\boldsymbol{x}$	$\boldsymbol{x}^{\mathrm{Ref}}$			$p^{\mathrm{DA}} = 1$	$p^{\mathrm{DA}} = 5$
$\boldsymbol{a}^{\mathrm{NN}}$, $\boldsymbol{a}^{\mathrm{NN-DA}}$	$\boldsymbol{a}^{\mathrm{POD}}$	–	1.3234	1.0009	0.8625
u_x^{POD}, u_x^{NN}, $u_x^{\mathrm{NN-DA}}$	u_x^{PIV}	0.1332	0.2765	0.2203	0.1886
$\lvert\boldsymbol{u}^{\mathrm{POD}}\rvert$, $\lvert\boldsymbol{u}^{\mathrm{NN}}\rvert$, $\lvert\boldsymbol{u}^{\mathrm{NN-DA}}\rvert$	$\lvert\boldsymbol{u}^{\mathrm{PIV}}\rvert$	0.2973	0.6573	0.5516	0.5196

Table 5: Time averaged *NRMSE* values in the testing window corresponding to the POD coefficients $\boldsymbol{a}$, the streamwise velocity component u_x at location $(x/D, y/D) = (4.8, 0.9)$, and the full-field velocity magnitude $|\boldsymbol{u}|$ for the cylinder wake flow at $Re = 1.5 \times 10^4$.

prediction time span. This error is defined as

$$NRMSE(t) = \frac{\sqrt{\sum_{i=1}^{N_\chi}(|\boldsymbol{u}^{\mathrm{NN}}(\boldsymbol{\chi}_i,t)| - |\boldsymbol{u}^{\mathrm{PIV}}(\boldsymbol{\chi}_i,t)|)^2}}{\sqrt{\sum_{i=1}^{N_\chi}(|\boldsymbol{u}^{\mathrm{PIV}}(\boldsymbol{\chi}_i,t)|)^2}}. \tag{43}$$

In Fig. 13, we show the temporal evolution of *NRMSE* as obtained from NN-ROM and NN-ROM-DA on the one hand, and the coefficients determined by POD on the other hand. Again, the NN-ROM-DA estimates show an improvement over the NN-ROM estimates. Indeed, the reconstructions obtained from NN-ROM-DA have the same order of magnitude of error as those obtained from the direct reconstruction using POD modes. A summary of the averaged *NRMSE* values for the predictions in latent space and the reconstructions in physical space is given in Tab. 5. In the latent space, the use of NN-ROM-DA (with $p^{\mathrm{DA}} = 5$) leads to a 35% reduction in the error value as compared to NN-ROM. A corresponding reduction of 32% and 21% in the averaged error values are obtained in the physical space, respectively for the streamwise velocity measurement at a location in the farfield wake and the velocity magnitude. The lower error magnitudes obtained by NN-ROM-DA demonstrate the interest of coupling non-intrusive models determined by neural networks to the data assimilation framework.

4. Conclusion

Neural network-based, data-driven reduced-order model (NN-ROM) has been considered as a surrogate, non-intrusive alternative to the intrusive reduced-order models. Through applications to a nonlinear toy model and a fluid flow case, it was demonstrated that NN-ROM offers a viable replacement as a forward model to provide long-term prediction. The whole framework can be

considered in an offline-online paradigm. In the offline stage, NN-ROM is trained on variables in the latent space, namely the POD coefficients. These are obtained by projecting a large amount of high-fidelity snapshots onto the reduced space. In the online stage, the trained NN-ROM is rapidly evaluated to obtain the future state of the latent space variables. These variables are then used for reconstruction in the physical space, thus returning predicted high-fidelity flow field.

The NN-ROM is designed such that it learns on residual targets, *i.e.* the difference of the state of interest between two consecutive time steps, rather than the value of the future state itself. The model takes into account the memory effect by considering a sequential input of fixed length of the latent variable in order to provide estimates at subsequent time steps. Influence of the inclusion of temporal history of the dynamics on the performance of NN-ROM was evaluated for the standard Lorenz-63 system. It was observed that the accuracy of estimation increased with an increase in the length of the temporal history data (p) provided as input up till $p = 10$ but the accuracy deteriorated for a higher value of p. It is suggested that p should therefore be considered as one of the training hyperparameters.

In order to ensure accurate long-term predictions, NN-ROM is augmented with an Ensemble Kalman Filter (EnKF) algorithm. The ability of NN-ROM to provide sequential updates makes it amenable to be used as a forecast model in the EnKF algorithm. The performance of NN-ROM augmented with EnKF (NN-ROM-DA) in terms of reconstruction of the flow field is evaluated on experimental cylinder wake flow data at $Re = 1.5 \times 10^4$. Observations in the form of streamwise velocity components are used to improve the accuracy of NN-ROM estimates. NN-ROM-DA is found to be effective in mitigating the observed phase shift over a long time span. Consequently, a more accurate long-term prediction of dynamics in both the latent and physical space as compared to NN-ROM is obtained, implying an overall improvement in the estimation. Moreover, using multiple consecutive assimilations ($p^{\mathrm{DA}} > 1$), a decrease of 21% in the error metric was observed as compared to the initial NN-ROM estimation, indicating further improvement in the long-term predictions.

In this work, it was assumed that the reduced-order models are applicable only within the operating range of the snapshot dataset used to build the model. Robustness analysis of the models must be performed in order to have a detailed knowledge of the range of validity of the reduced-order model, *i.e.* to understand if the identified model will still be able to provide accurate estimates if the parameters like Reynolds number vary between specified minimum and maximum values. This can be challenging as, in governing nonlinear PDEs, the change of physical parameters results in a change in the spatial distribution of the solution which the initial POD modes may not be able to approximate. One of the scenarios in which the robustness analysis can be beneficial is when uncertainties are introduced in terms of time varying parameter disturbances, which in turn affect the estimation of the POD coefficients. Also, for NN-ROM architecture and optimization, more sophisticated methods like random search [15] and Bayesian optimization [16] should be introduced to tune the model hyperparameters according to the problem.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., et al., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467.

[2] Ahuja, S., Rowley, C.W., 2010. Feedback control of unstable steady states of flow past a flat plate using reduced-order estimators. Journal of Fluid Mechanics 645, 447–478. doi:10.1017/S0022112009992655.

[3] Alfio Quarteroni, G.R., 2014. Reduced Order Methods for Modeling and Computational Reduction. MS&A - Modeling, Simulation and Applications. 1 ed., Springer International Publishing. doi:doi.org/10.1007/978-3-319-02090-7.

[4] Asch, M., Bocquet, M., Nodet, M., 2016. Data assimilation. Fundamentals of Algorithms, Society for Industrial and Applied Mathematics.

[5] Aubry, N., 1991. On the hidden beauty of the proper orthogonal decomposition. Theoretical and Computational Fluid Dynamics 2, 339–352. doi:10.1007/BF00271473.

[6] Aubry, N., Holmes, P., Lumley, J.L., Stone, E., 1988. The dynamics of coherent structures in the wall region of a turbulent boundary layer. Journal of Fluid Mechanics 192, 115–173. doi:10.1017/S0022112088001818.

[7] Bagheri, S., Brandt, L., Henningson, D.S., 2009. Input-output analysis, model reduction and control of the flat-plate boundary layer. Journal of Fluid Mechanics 620, 263–298. doi:10.1017/S0022112008004394.

[8] Barbagallo, A., Sipp, D., Schmid, P.J., 2009. Closed-loop control of an open cavity flow using reduced-order models. Journal of Fluid Mechanics 641, 1–50. doi:10.1017/S0022112009991418.

[9] Baur, U., Beattie, C., Benner, P., Gugercin, S., 2011. Interpolatory projection methods for parameterized model reduction. SIAM Journal on Scientific Computing 33, 2489–2518. doi:10.1137/090776925.

[10] Baur, U., Benner, P., 2008. Cross-gramian based model reduction for data-sparse systems. Electronic Transactions on Numerical Analysis 31, 27.

[11] Benard, N., Debien, A., David, L., Moreau, E., 2010. Analyse par PIV rapide du sillage d'un cylindre manipulé par actionneurs plasmas, in: Congres Francophone de Techniques Laser, Vandoeuvre-les-Nancy, pp. 14–17.

[12] Bergmann, M., Bruneau, C.H., Iollo, A., 2009. Enablers for robust POD models. Journal of Computational Physics 228, 516–538. doi:10.1016/j.jcp.2008.09.024.

[13] Bergmann, M., Cordier, L., 2008. Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models. Journal of Computational Physics 227, 7813–7840. doi:10.1016/j.jcp.2008.04.034.

[14] Bergmann, M., Cordier, L., Brancher, J.P., 2005. Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model. Physics of fluids 17, 097101. doi:10.1063/1.2033624.

[15] Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. Journal of Machine Learning Research 13.

[16] Brochu, E., Cora, V.M., de Freitas, N., 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv:1012.2599.

[17] Brunton, S.L., Noack, B.R., 2015. Closed-loop turbulence control: Progress and challenges. Applied Mechanics Reviews 67. doi:10.1115/1.4031175.

[18] Brunton, S.L., Proctor, J.L., Kutz, J.N., 2016. Sparse identification of nonlinear dynamics with control (SINDYc). IFAC-PapersOnLine 49, 710–715. doi:10.1016/j.ifacol.2016.10.249.

[19] Cao, Y., Zhu, J., Navon, I.M., Luo, Z., 2007. A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition. International Journal for Numerical Methods in Fluids 53, 1571–1583. doi:10.1002/fld.1365.

[20] Casenave, F., Ern, A., Lelievre, T., 2015. A nonintrusive reduced basis method applied to aeroacoustic simulations. Advances in Computational Mathematics 41, 961–986. doi:10.1007/s10444-014-9365-0.

[21] Chiu, T.Y., 1996. Model reduction by the low-frequency approximation balancing method for unstable systems. IEEE transactions on Automatic Control 41, 995–997. doi:10.1109/9.508903.

[22] Cohen, K., Siegel, S., McLaughlin, T., Gillies, E., 2003. Feedback control of a cylinder wake low-dimensional model. AIAA journal 41, 1389–1391. doi:10.2514/2.2087.

[23] Cordier, L., Bergmann, M., 2008. Proper Orthogonal Decomposition: an overview, in: Lecture series 2002-04, 2003-03 and 2008-01 on post-processing of experimental and numerical data. Von Kármán Institute for Fluid Dynamics, pp. 1–46. ISBN 978-2-930389-80-X.

[24] Deane, A.E., Kevrekidis, I.G., Karniadakis, G.E., Orszag, S.A., 1991. Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders. Physics of Fluids A: Fluid Dynamics 3, 2337–2354. doi:10.1063/1.857881.

[25] Degroote, J., Vierendeels, J., Willcox, K., 2010. Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis. International Journal for Numerical Methods in Fluids 63, 207–230. doi:10.1002/fld.2089.

[26] Delville, J., 1994. Characterization of the organization in shear layers via the proper orthogonal decomposition. Applied Scientific Research 53, 263–281. doi:10.1007/BF00849104.

[27] Dimitriu, G., Apreutesei, N., 2007. Comparative study with data assimilation experiments using proper orthogonal decomposition method, in: International Conference on Large-Scale Scientific Computing, Springer. pp. 393–400. doi:10.1007/978-3-540-78827-0_44.

[28] Ehlert, A., Nayeri, C.N., Morzynski, M., Noack, B.R., 2019. Locally linear embedding for transient cylinder wakes. arXiv e-prints arXiv:1906.07822.

[29] Evensen, G., 1994. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. Journal of Geophysical Research: Oceans 99, 10143–10162. doi:10.1029/94JC00572.

[30] Frangos, M., Marzouk, Y., Willcox, K., van Bloemen Waanders, B., 2010. Surrogate and reduced-order modeling: A comparison of approaches for large-scale statistical inverse problems. John Wiley & Sons, Ltd. chapter 7. pp. 123–149. doi:10.1002/9780470685853.ch7.

[31] Gillies, E.A., 1998. Low-dimensional control of the circular cylinder wake. Journal of Fluid Mechanics 371, 157–178. doi:10.1017/S0022112098002122.

[32] Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings. pp. 249–256.

[33] Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y., 2016. Deep Learning. volume 1. MIT Press, Cambridge.

[34] Guenot, M., Lepot, I., Sainvitu, C., Goblet, J., Coelho, R.F., 2013. Adaptive sampling strategies for non-intrusive POD-based surrogates. Engineering Computations: International Journal for Computer-Aided Engineering 30, 521–547. doi:10.1108/02644401311329352.

[35] Gugercin, S., Antoulas, A.C., 2004. A survey of model reduction by balanced truncation and some new results. International Journal of Control 77, 748–766. doi:10.1080/00207170410001713448.

[36] Haasdonk, B., Ohlberger, M., 2008. Reduced basis method for finite volume approximations of parametrized linear evolution equations. ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique 42, 277–302. doi:10.1051/m2an:2008001.

[37] Hesthaven, J.S., Ubbiali, S., 2018. Non-intrusive reduced order modeling of nonlinear problems using neural networks. Journal of Computational Physics 363, 55–78. doi:10.1016/j.jcp.2018.02.037.

[38] Himpe, C., Ohlberger, M., 2014. Cross-gramian-based combined state and parameter reduction for large-scale control systems. Mathematical Problems in Engineering 2014. doi:10.1155/2014/843869.

[39] Hoepffner, J., Akervik, E., Ehrenstein, U., Henningson, D.S., 2006. Control of cavity-driven separated boundary layer, in: 3rd Conference on Active Flow Control, Berlin, Germany.

[40] Holmes, P., Lumley, J.L., Berkooz, G., Rowley, C.W., 2012. Turbulence, Coherent Structures, Dynamical Systems and Symmetry. Cambridge Monographs on Mechanics. 2 ed., Cambridge University Press. doi:10.1017/CBO9780511919701.

[41] Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.

[42] Ito, K., Ravindran, S.S., 2001. Reduced basis method for optimal control of unsteady viscous flows. International Journal of Computational Fluid Dynamics 15, 97–113. doi:10.1080/10618560108970021.

[43] Iuliano, E., Quagliarella, D., 2013. Aerodynamic shape optimization via non-intrusive POD-based surrogate modelling, in: 2013 IEEE Congress on Evolutionary Computation, IEEE. pp. 1467–1474. doi:10.1109/CEC.2013.6557736.

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

[45] Kunisch, K., Volkwein, S., 1999. Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition. Journal of Optimization Theory and Applications 102, 345–371. doi:10.1023/A:1021732508059.

[46] Kunisch, K., Volkwein, S., Xie, L., 2004. HJB-POD-based feedback design for the optimal control of evolution problems. SIAM Journal on Applied Dynamical Systems 3, 701–722. doi:10.1137/030600485.

[47] Lall, S., Marsden, J.E., Glavaški, S., 2002. A subspace approach to balanced truncation for model reduction of nonlinear control systems. International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal 12, 519–535. doi:10.1002/rnc.657.

[48] LeGresley, P., Alonso, J., 2000. Airfoil design optimization using reduced order models based on proper orthogonal decomposition, in: Fluids 2000 Conference and Exhibit, p. 2545. doi:10.2514/6.2000-2545.

[49] Loiseau, J.C., Brunton, S.L., 2018. Constrained sparse Galerkin regression. Journal of Fluid Mechanics 838, 42–67. doi:10.1017/jfm.2017.823.

[50] Lumley, J.L., 1967. The structure of inhomogeneous turbulent flows, in: Yaglom, A.M., Tatarski, V.I. (Eds.), Atmospheric Turbulence and Radio Propagation. Nauka, pp. 166–178.

[51] Mathelin, L., Hussaini, M.Y., Zang, T.A., 2005. Stochastic approaches to uncertainty quantification in CFD simulations. Numerical Algorithms 38, 209–236. doi:10.1007/BF02810624.

[52] Mehrmann, V., Stykel, T., 2005. Balanced truncation model reduction for large-scale systems in descriptor form, in: Dimension Reduction of Large-Scale Systems. Springer, pp. 83–115. doi:10.1007/3-540-27909-1_3.

[53] Mendonça, G., Afonso, F., Lau, F., 2019. Model order reduction in aerodynamics: Review and applications. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 233, 5816–5836. doi:10.1177/0954410019853472.

[54] Mezić, I., 2013. Analysis of fluid flows via spectral properties of the Koopman operator. Annual Review of Fluid Mechanics 45, 357–378. doi:10.1146/annurev-fluid-011212-140652.

[55] Montúfar, G., Pascanu, R., Cho, K., Bengio, Y., 2014. On the number of linear regions of deep neural networks. arXiv:1402.1869.

[56] Noack, B.R., Morzyński, M., Tadmor, G. (Eds.), 2011. Reduced-order Modelling for Flow Control. CISM, Springer-Verlag.

[57] Östh, J., Noack, B.R., Krajnović, S., Barros, D., Borée, J., 2014. On the need for a nonlinear subscale turbulence term in POD models as exemplified for a high-Reynolds-number flow over an Ahmed body. Journal of Fluid Mechanics 747, 518–544. doi:10.1017/jfm.2014.168.

[58] Panzer, H., Mohring, J., Eid, R., Lohmann, B., 2010. Parametric model order reduction by matrix interpolation. at-Automatisierungstechnik Methodenund Anwendungen der Steuerungs- 58, 475–484. doi:10.1524/auto.2010.0863.

[59] Pawar, S., Rahman, S.M., Vaddireddy, H., San, O., Rasheed, A., Vedula, P., 2019. A deep learning enabler for nonintrusive reduced order modeling of fluid flows. Physics of Fluids 31, 085101. doi:10.1063/1.5113494.

[60] Peña, F.L., Casás, V.D., Gosset, A., Duro, R., 2012. A surrogate method based on the enhancement of low fidelity computational fluid dynamics approximations by artificial neural networks. Computers & fluids 58, 112–119. doi:10.1016/j.compfluid.2012.01.008.

[61] Peterson, J.S., 1989. The reduced basis method for incompressible viscous flow calculations. SIAM Journal on Scientific and Statistical Computing 10, 777–786. doi:10.1137/0910047.

[62] Podvin, B., Lumley, J., 1998. A low-dimensional approach for the minimal flow unit. Journal of Fluid Mechanics 362, 121–155. doi:10.1017/S0022112098008854.

[63] Qin, T., Wu, K., Xiu, D., 2019. Data driven governing equations approximation using deep neural networks. Journal of Computational Physics 395, 620–635. doi:10.1016/j.jcp.2019.06.042.

[64] Qiu, C., Chou, J., 2006. Four-dimensional data assimilation method based on SVD: Theoretical aspect. Theoretical and applied climatology 83, 51–57. doi:10.1007/s00704-005-0162-z.

[65] Raisee, M., Kumar, D., Lacor, C., 2015. A non-intrusive model reduction approach for polynomial chaos expansion using proper orthogonal decomposition. International Journal for Numerical Methods in Engineering 103, 293–312. doi:10.1002/nme.4900.

[66] Raissi, M., Perdikaris, P., Karniadakis, G.E., 2018. Multistep neural networks for data-driven discovery of nonlinear dynamical systems arXiv:1801.01236.

[67] Reiss, J., Schulze, P., Sesterhenn, J., Mehrmann, V., 2018. The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena. SIAM Journal on Scientific Computing 40, A1322–A1344. doi:10.1137/17M1140571.

[68] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533–536. doi:10.1038/323533a0.

[69] San, O., Maulik, R., Ahmed, M., 2019. An artificial neural network framework for reduced order modeling of transient flows. Communications in Nonlinear Science and Numerical Simulation 77, 271–287. doi:10.1016/j.cnsns.2019.04.025.

[70] Schlegel, M., Noack, B.R., 2015. On long-term boundedness of Galerkin models. Journal of Fluid Mechanics 765, 325–352. doi:10.1017/jfm.2014.736.

[71] Schmid, P.J., 2010. Dynamic Mode Decomposition of numerical and experimental data. Journal of Fluid Mechanics 656, 5–28. doi:10.1017/S0022112010001217.

[72] Schmid, P.J., Henningson, D.S., Jankowski, D.F., 2002. Stability and Transition in Shear Flows. volume 55 of *Applied Mathematical Reviews*. ASME. doi:10.1115/1.1470687.

[73] Sirovich, L., 1987. Turbulence and the dynamics of coherent structures, Parts I-III. Quarterly of Applied Mathematics 45, 561–590.

[74] Ştefănescu, R., Sandu, A., Navon, I.M., 2015. POD/DEIM reduced-order strategies for efficient four dimensional variational data assimilation. Journal of Computational Physics 295, 569–595. doi:10.1016/j.jcp.2015.04.030.

[75] Taira, K., Brunton, S.L., Dawson, S.T.M., Rowley, C.W., Colonius, T., McKeon, B.J., Schmidt, O.T., Gordeyev, S., Theofilis, V., Ukeiley, L.S., 2017. Modal analysis of fluid flows: An overview. AIAA Journal 55, 4013–4041. doi:10.2514/1.J056060.

[76] Theofilis, V., 2011. Global linear instability. Annual Review of Fluid Mechanics 43, 319–352. doi:10.1146/annurev-fluid-122109-160705.

[77] Tissot, G., Cordier, L., Benard, N., Noack, B.R., 2014. Model reduction using dynamic mode decomposition. Comptes Rendus Mécanique 342, 410–416. doi:10.1016/j.crme.2013.12.011.

[78] Wang, Q., Hesthaven, J.S., Ray, D., 2019. Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. Journal of Computational Physics 384, 289–307. doi:10.1016/j.jcp.2019.01.031.

[79] Wang, Z., Akhtar, I., Borggaard, J., Iliescu, T., 2012. Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison. Computer Methods in Applied Mechanics and Engineering 237, 10–26. doi:10.1016/j.cma.2012.04.015.

[80] Williamson, C.H., 1996. Vortex dynamics in the cylinder wake. Annual Review of Fluid Mechanics 28, 477–539. doi:10.1146/annurev.fl.28.010196.002401.

[81] Winter, M., Breitsamter, C., 2014. Reduced-order modeling of unsteady aerodynamic loads using radial basis function neural networks. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV.

[82] Wirtz, D., Haasdonk, B., 2012. Efficient a-posteriori error estimation for nonlinear kernel-based reduced systems. Systems & Control Letters 61, 203–211. doi:10.3182/20120215-3-AT-3016.00135.

[83] Xiao, M., Breitkopf, P., Coelho, R.F., Knopf-Lenoir, C., Sidorkiewicz, M., Villon, P., 2010. Model reduction by CPOD and Kriging. Structural and Multidisciplinary Optimization 41, 555–574. doi:10.1007/s00158-009-0434-9.