Adv Comput Math (2018) 44:1717–1750
https://doi.org/10.1007/s10444-018-9590-z

Neural network closures for nonlinear model order reduction

Omer San[1] · **Romit Maulik**[1]

Received: 21 May 2017 / Accepted: 19 January 2018 /
Published online: 29 January 2018

Abstract Many reduced-order models are neither robust with respect to parameter changes nor cost-effective enough for handling the nonlinear dependence of complex dynamical systems. In this study, we put forth a robust machine learning framework for projection-based reduced-order modeling of such nonlinear and nonstationary systems. As a demonstration, we focus on a nonlinear advection-diffusion system given by the viscous Burgers equation, which is a prototypical setting of more realistic fluid dynamics applications due to its quadratic nonlinearity. In our proposed methodology the effects of truncated modes are modeled using a single layer feedforward neural network architecture. The neural network architecture is trained by utilizing both the Bayesian regularization and extreme learning machine approaches, where the latter one is found to be more computationally efficient. A significant emphasis is laid on the selection of basis functions through the use of both Fourier bases and proper orthogonal decomposition. It is shown that the proposed model yields significant improvements in accuracy over the standard Galerkin projection methodology with a negligibly small computational overhead and provide reliable predictions with respect to parameter changes.

Keywords Machine learning · Neural networks · Extreme learning machine · Model order reduction · Projection methods · Burgers equation

Mathematics Subject Classification (2010) 37N10 · 76M25 · 76F20 · 76D99

Communicated by: Peter Benner

✉ Omer San
osan@okstate.edu

[1] Department of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK 74078, USA

1 Introduction

The ever-increasing need for accurate prediction of many complex nonlinear processes leads to very large-scale dynamical systems whose simulations and analyzes make overwhelming and unmanageable demands on computational resources. Since the computational cost of traditional full-order numerical simulations is extremely prohibitive, many successful model order reduction approaches have been introduced [1, 16, 24, 29, 31, 32, 46, 58, 71, 78, 84]. The purpose of such approaches is to reduce this computational burden and serve as surrogate models for efficient computational analysis of such systems, especially in settings where traditional methods require repeated model evaluations over a large range of parameter values. Through a simplification of the computational complexity of an underlying governing law, these reduced-order models offer promises in many prediction, identification, design, optimization, and control applications. However, they are neither robust with respect to parameter changes nor low-cost with respect to nonlinear dependence handling for complex dynamical systems encountered in a wide range of physical phenomena. Therefore, reduced-order modeling (ROM) remains an open challenge for parametric time varying systems and the development of efficient and reliable model order reduction techniques is of paramount importance for both fundamental and applied science. In this study, we focus on the development of an accurate and robust machine learning framework for projection-based nonlinear model order reduction of such transient nonlinear systems.

The basic philosophy of projection-based approaches is an alleviation of the intense computational burden of partial differential equations obtained from governing laws [9]. These approaches endeavor to reduce the high degrees of freedom of a governing law through an expansion in a transformed space, traditionally with orthogonal bases. The choice of the transformation aids the user in the identification of sparseness and thus allows them to obtain a dense (but low dimensional) representation of the same governing law. Among the large variety of reduced-order modeling strategies, *proper orthogonal decomposition* (POD) has emerged as a popular technique for the study of dynamical systems [3, 6, 8]. With respect to terminology, POD is also referred to as the Karhunen-Loéve expansion [56], principal component analysis [42], or empirical orthogonal functions [57]. A successful construction of quality POD bases requires a collection of high-fidelity numerical simulations of the governing equations of the dynamic system being studied if an exact solution is not available. This information is used to devise optimal bases for the transformed space. Indeed, the construction of these bases also serves as a post-processing tool, for instance as a method of extraction of the large scale coherent structures in statistical pattern recognition [59]. This is because the global solution may be spanned through a considerably truncated number of optimal bases to resolve attractors and transients. This study, however, is oriented more toward the feedback flow control community which attempts to use POD as a degree of freedom reduction technique. In addition, other many-query applications such as uncertainty quantification and optimization may also benefit from more accurate ROMs at low model orders. We would like to emphasize here that there exist several noteworthy approaches to ROM implementation

such as dynamic mode decomposition (or DMD) which is a technique for determining a low-dimensional subspace from observed data similar to POD but with a description of the dynamics on that subspace [82]. Another popular technique for determining a reduced subspace is the extended DMD (or EDMD). The reader is directed to [77, 87] for an excellent discussion of these closely related approaches.

In our investigation, we implement the POD for our ROM analysis using a Galerkin projection (GP) approach. POD-GP has extensively been used to provide fast and accurate simulations of large nonlinear systems [13, 50, 92]. This approach is devised so as to generate a reduced model where the truncated modes are evolved through time as a set of ordinary differential equations. This may be considered to be a *projection* type approach as it is obtained by projecting the governing equations on to the truncated modes (obtained from our POD). We must remark here that an *orthogonal* projection has been used to move our problem to the reduced subspace. A non-orthogonal projection may also be employed which is generally known as the Petrov-Galerkin projection [5]. Another approach one may use for nonlinear systems is given by Koopman operator theory [33] which identifies a subspace where the nonlinear dynamics are linearized and uncoupled. However, in this investigation we limit ourselves to the GP approach alone. The process of truncation (which is necessary for degree of freedom reduction) introduces a limitation to our chosen approach. The truncation procedure causes a significant loss of information particularly for highly non-stationary, convective and nonlinear problems [21, 27]. Therefore, there have been considerable efforts to mitigate the drawbacks of this loss of information such as through the process of closure modeling [15, 80, 89, 90] where arguments are made in favor of modeling the effect of the discarded modes on the preserved ones. Another promising approach is to develop strategies for constructing more representative bases [16–18, 45, 51, 81]. Bases obtained from orthogonal Fourier modes have also been utilized in present investigation for added comparison.

By analogy with the large eddy simulations of turbulent flows, several *eddy viscosity* type of closure models have been introduced for ROMs [2, 6, 12, 14, 69, 80, 81, 85, 88, 90, 91]. Recently, a new closure modeling framework for stabilization of ROMs has been proposed by using a robust Lyapunov control theory. It has been demonstrated that the Lyapunov-based closure model is robust to parametric uncertainties and the free parameters are optimized using a data-driven multi-parametric extremum seeking algorithm [10, 11]. Other stabilization models for ROMs have been also suggested in literature [4, 7, 11, 21, 44, 52, 71, 76, 93, 96]. In addition to these models, Cazemier's closure modeling approach [19, 20, 80] uses the concept of energy conservation to account for the truncated modes. A data-driven filtered ROM framework has also been proposed to model the interaction between the resolved and unresolved modes solving a linear least squares problem [95]. To summarize, closure strategies for the stabilization of ROMs are devised with the specific intent of modeling the effect of the unrepresented spatial modes on the temporal evolution of the reduced system so as to prevent large deviations from the mean behavior of a hypothetical PDE evolution of the same physics. Therefore, any successful ROM implementation needs an intelligent selection of truncation as well as closure prior to deployment.

The focus of the present investigation will be to utilize a data-driven supervised *machine learning* framework for the closure (and stabilization) of a highly truncated POD implemented using the GP approach. A single layer feed-forward *artificial neural network* is used to estimate the effect of the discarded modes on the retained ones during the time integration of the ordinary differential equations obtained using POD-GP. In brief, an artificial neural network (ANN) can be used to set up a nonlinear relationship between a desired set of inputs and targets provided a large set of benchmark data for the underlying statistical relationship is available. It is therefore no surprise that this subset of the machine learning field has seen wide application in function approximation, data classification, pattern recognition and dynamic systems control applications [23, 94]. Before an ANN is deployed, it must be trained to accurately capture the nonlinear relationship between its inputs and outputs. For this purpose, a high-fidelity solution of the Burgers equation is used to generate the 'true' modes. An optimization problem is solved to minimize a desired performance function (which generally represents a measure of the suitability of a network for prediction). While we shall provide information about the machine learning framework implemented here in far greater detail within, we note that we have used the Bayesian regularization [30] and extreme learning machine [43] approaches for the training of the artificial neural network. Both methods have been developed in the machine learning community to accurately capture statistical trends in noisy data. While the former is famous for its accuracy, the latter is attractive due to its very low computational cost. Indeed, the extreme learning machine training (or ELM) proves to be very exciting as a possible means to address the need for fast learning in dynamically adaptive systems (although the topic of which is left to a separate investigation).

While there have been investigations into the suitability of machine learning techniques for the purpose of ROM implementations [47, 65–67, 79] and these approaches are quite well studied for use in feedback flow control where they are used to generate a direct mapping of flow measurements to actuator control systems [25, 26, 28, 34–36, 40, 53], it is our belief that the work presented in this document represents the first utilization of an ANN as closure model in the POD-GP framework. We also note that ANNs are also gaining popularity in the fields of turbulence modeling and hidden physics estimation where deep architectures with several hidden layers, coupled with physics-informed architectural modifications [55] or physics-informed regularization [73, 74] are utilized for the purpose of realizability preservation in ANN predictions and grid-free nonlinear PDE behavior estimation respectively. As mentioned previously, information from the time evolution of our PDE (from its analytical solution) is leveraged to provide a supervised learning framework for the ANN to approximate the error in the modal evolution through GP. A regularized training ensures that no localized behavior is observed from the trained ANN with respect to the governing equations and the data they have produced. Training data sets are generated through different realizations of the viscosity of the viscous Burgers equation with a high degree of freedom. The trained ANN architecture is then tested for its ability to stabilize modal evolution for values of viscosity that are not in the training data set. Our investigations can be summarized by the following bullet points

– A machine learning based closure model is implemented in the ROM framework (using a Galerkin projection) through the use of a single layer feedforward artificial neural network. Both optimal POD and Fourier modes are used as the bases of the transformation space.
– The closure is 'trained' using high-fidelity data obtained from the true solution of the time evolution of our partial differential equation. The Bayesian regularization and extreme learning machine approaches are used for a regularized training to capture underlying dynamics from the data.
– The proposed framework is tested through a ROM framework for the viscous Burgers equation for physical parameters which were not included in the training data set.

The rest of this paper is devoted to a mathematical development of the concepts we have introduced as well as a documentation of our results.

2 Mathematical model

2.1 Full-order simulations

For the development of a robust ROM, a high-fidelity dataset is generated using a fully resolved solution of the Burgers equation in a conservative formulation given by

$$\begin{aligned} &\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}, \\ &u \in \mathbb{R}^1, \quad x \in \Omega = [0, L], \quad t > 0, \end{aligned} \tag{1}$$

where Re is the Reynolds number and L is the extent of our one-dimensional domain Ω. This equation is generally considered a low-dimensional equivalent of the full Navier-Stokes equations due to its advective and diffusive behavior. It is therefore, quite common, to use the Burgers equation for the preliminary evaluation of ROM frameworks prior to deployment for fluid flow problems [44, 49, 80]. Indeed, it may be considered a challenging convective system for ROM assessments as it characterizes localized flow structures such as standing and advected shock waves as well as a cascade of energy following theoretical scaling laws [63] and generally requires additional stabilization in the form of a closure model that represents the effect of the discarded modal quantities. In practice, the full-order solutions are used to generate time snapshot data for generating POD bases either through discretization methods for complex problems or through closed-form solutions for simple benchmark problems. In the present study, we utilize an analytical solution to obtain snapshot data for the Burgers equation, the details of which are presented in Section 6.

2.2 Method of snapshots

The methodology of our ROM assessments is initiated by the decomposition of the solution field into a time invariant averaged $\bar{u}(x)$ and a fluctuating component $\hat{u}(x,t)$ through [41, 70],

$$u(x,t) = \bar{u}(x) + \hat{u}(x,t), \tag{2}$$

where the mean of the snapshot data is

$$\bar{u}(x) = \frac{1}{M}\sum_{i=1}^{M} u(x,t_i), \tag{3}$$

and M corresponds to a finite number of distinct snapshots at time t_i used for time averaging. The goal is to then devise a set of M bases

$$\phi_k(x) \in \mathbb{R}^1 \quad k = 1, 2, \ldots, M, \quad x \in \Omega = [0, L], \tag{4}$$

which can reproduce the fluctuating component of the above decomposition

$$\hat{u}(x,t) = \sum_{k=1}^{M} a_k(t)\phi_k(x), \tag{5}$$

where $a_k(t)$ are the time-dependent POD coefficients. A transformation to basis space with truncation (i.e., evolution for only a part of the M total bases) will lead to a reduced system of ODEs that endeavors to capture the spatial and temporal behavior of the PDE. As explained in further detail in Section 3, the method of snapshots [72, 86], which is an efficient way to compute the POD basis functions, is utilized to represent system dynamics. This is motivated by practical considerations for approximating the Fredholm-type integral eigensystem equation since data are limited to a finite number of frames or snapshots. Note that the number and location of the snapshots $u(x,t_i)$ are dependant on the dynamics of the system and vary from problem to problem. In this study, snapshots are sampled uniformly within the solution time interval.

3 Basis selection

In this section, we detail the two types of bases chosen for the transformation space of our PDE. Some of the bases obtained using our two approaches are shown in Fig. 1 and we note that they are orthogonal to each other.

3.1 Fourier basis functions

As a first choice for our orthogonal bases, we utilize the Fourier series coefficients. In order to ensure that the boundary conditions of the PDE are respected we discard the Cosine components of the series and are left with

$$\phi_k(x) = \sqrt{\frac{2}{L}}\sin(k\pi x/L), \tag{6}$$

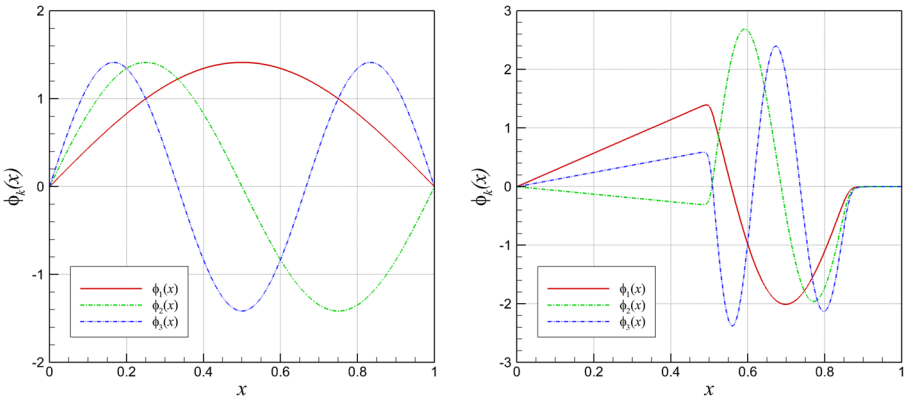


Fig. 1 Representative basis functions

where $\phi_k(x)$ is the k^{th} basis function. We remark here that these bases may *not* be considered optimal as they are devised without any information from our exact solutions but through intuition. Specifically, the choice of these bases imply ambitious assumptions regarding the spatial frequency content of the problem in question without any data assimilation from high-fidelity numerical solutions or experimental data (specifically as used in the POD approach). Combining these assumptions with a truncation make it very possible for the ROM to miss key aspects of the nonlinear interaction between spatial and temporal modes. However, for systems where data extraction (through experiments or high-fidelity numerical simulations) is infeasible, a good starting point would be to assume that the Fourier bases are able to capture a large proportion of the energies of the system at lower frequencies. The condition for orthonormality between bases can be expressed as:

$$(\phi_k, \phi_l) = \begin{cases} 1, & k = l \\ 0, & k \neq l. \end{cases}, \tag{7}$$

where we have defined an inner product of any two fields ϕ_1 and ϕ_2 as

$$(\phi_1, \phi_2) = \int_\Omega \phi_1(x)\phi_2(x)dx. \tag{8}$$

3.2 POD basis functions

In this subsection, we detail the POD approach for ROM to capture unsteady, convective and non-periodic dynamics of governing PDEs. The reader is directed to [80] for a far more detailed discussion of this topic. Section 2.2 introduces the method of snapshots utilized for the subsequent bases development.

A POD can be constructed from our scalar field $u(x, t)$ at different times (previously introduced as snapshots). These snapshots are either obtained by solving the governing equations through a DNS or (as in this case) through the exact solution. In the following, we will utilize the superscript i to indicate a particular snapshot in time.

For the POD approach we utilize a total of M snapshots for the field variable, i.e., $u(x,t_i) = u^i(x)$ for $i = 1, 2, \ldots, M$. A correlation matrix may be constructed using the fluctuating components of the snapshots, i.e., $\hat{u}(x,t_i) = \hat{u}^i(x)$, to give

$$C_{ij} = \int_\Omega \hat{u}^i(x)\hat{u}^j(x)dx, \tag{9}$$

where i and j refer to the i^{th} and j^{th} snapshots. The correlation matrix C is a non-negative symmetric square matrix of side M and may be expressed as $C_{ij} = (\hat{u}^i, \hat{u}^j)$. In this study, we use Simpson's 3/8 integration rule for a numerical computation of the inner products. The optimal POD basis functions are obtained by performing an eigendecomposition for the C matrix. This has been shown in detail in POD literature (see, e.g., [41, 75, 86]). The eigenvalue problem can be expressed in the following form:

$$CW = W\Lambda, \tag{10}$$

where $\Lambda = \text{diag}[\lambda_1, \lambda_2, ..., \lambda_M]$ is a diagonal matrix containing the eigenvalues of this decomposition and $W = [\, w^1, w^2, ..., w^M]$ is an orthogonal basis consisting of the eigenvectors of this decomposition. The eigenvalues are stored in descending order, $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_M$ and the POD basis functions can then be written as

$$\phi_k(x) = \sum_{i=1}^{M} w_i^k \hat{u}(x,t_i) \quad k = 1, 2, \ldots, M, \tag{11}$$

where w_i^j is the i^{th} component of eigenvector w^j. Therefore we emphasize that the POD modes are ranked according to the magnitude of their corresponding eigenvalue. The eigenvectors must also be normalized in order to satisfy the condition of orthonormality between bases given by Eq. (7). It can be shown that, for Eq. (7) to be true for the POD bases, the eigenvector w^j must satisfy the following equation:

$$\sum_{i=1}^{M} w_i^j w_i^j = \frac{1}{\lambda_j}. \tag{12}$$

In practice, most of the subroutines for solving the eigensystem given in Eq. (10) return the eigenvector matrix W having all the eigenvectors normalized to unity. In that case, the orthogonal POD bases are given by

$$\phi_j(x) = \frac{1}{\sqrt{\lambda_j}} \sum_{i=1}^{M} w_i^j \hat{u}(x,t_i), \tag{13}$$

where $\phi_j(x)$ is the j^{th} POD basis function. The main motivation behind the construction of a ROM using the optimal POD bases is due to the fact that modes with high magnitudes of eigenvalues retain a greater proportion of the energy of the system. Hence, it is possible to construct a lower degree of freedom approximation of our sparse PDE in the space transformed by the aforementioned eigenvectors.

4 Projection-based model order reduction

For reducing the computational expense of our system in the transformed space, we truncate the number of modes in our system to a value $Q \ll M$ with the assumption that these Q modes capture a large fraction of the total energy of the system. For our test case given by the Burgers equation, the Galerkin projection can be carried out in the following manner:

$$\hat{u}(x,t) = \sum_{k=1}^{Q} a_k(t)\phi_k(x), \tag{14}$$

where a_k are the time dependent coefficients, and ϕ_k are the space dependent modes. To derive the Galerkin projection, we first rewrite the Burgers equation (i.e., Eq. (1)) in the following form

$$\frac{\partial u}{\partial t} = L[u] + N[u;u], \tag{15}$$

where

$$L[f] = \frac{1}{Re}\frac{\partial^2 f}{\partial x^2}, \tag{16}$$

is the linear operator, and

$$N[f;g] = -\frac{\partial}{\partial x}\left(\frac{1}{2}fg\right), \tag{17}$$

is the nonlinear operator. By applying this projection to our nonlinear system (i.e., multiplying Eq. (15) with the basis functions and integrating over the domain), we obtain our ROM, denoted POD-GP-ROM when using the optimal POD modes and Fourier-GP-ROM when using Fourier modes:

$$\left(\frac{\partial u}{\partial t}, \phi_k\right) = (L[u], \phi_k) + (N[u;u], \phi_k), \quad \text{for} \quad k = 1, 2, ..., Q. \tag{18}$$

The above expression may then be further modified by substituting Eq. (2) into Eq. (13) followed by the use of the condition of orthogonality given in Eq. (7) to obtain the following ROM implementation of our system:

$$\frac{da_k}{dt} = B_k + \sum_{i=1}^{Q} \mathfrak{L}_{ik} a_i + \sum_{i=1}^{Q}\sum_{j=1}^{Q} \mathcal{N}_{ijk} a_i a_j, \quad \text{for} \quad k = 1, 2, ..., Q, \tag{19}$$

where

$$B_k = (L[\bar{u}], \phi_k) + (N[\bar{u};\bar{u}], \phi_k), \tag{20}$$

$$\mathfrak{L}_{ik} = (L[\phi_i], \phi_k) + (N[\bar{u};\phi_i] + N[\phi_i;\bar{u}], \phi_k), \tag{21}$$

$$\mathcal{N}_{ijk} = (N[\phi_i;\phi_j], \phi_k). \tag{22}$$

The GP-ROM given by Eq. (19) consists of Q coupled ODEs and can be solved by a standard numerical method (such as the third-order Runge-Kutta scheme that is used in this study). The number of degrees of freedom of the system is now significantly lower. The vectors, matrices and tensors in Eqs. (20)–(22) are also precomputed quantities, which results in a dynamical system that can be solved very efficiently.

To complete the dynamical system given by Eq. (19), the initial condition is given by using the following projection:

$$a_k(t=0) = (u(x,t=0) - \bar{u}(x), \phi_k)\,, \tag{23}$$

where $u(x, t = 0)$ is the initial condition of the problem given in Eq. (1).

5 Artificial neural networks

In the process of developing a reduced system, the truncation of modes leads to an inadequate capture of the energies of a system corresponding to the discarded modes and this motivates the development of closure modeling strategies. As mentioned previously, closure models aim to represent the contributions of the truncated modes on the system for which we have applied an artificial neural network.

5.1 ANN closure modeling for ROM

Equation (19) can be rewritten as

$$\frac{da_k}{dt} = R_k, \tag{24}$$

where

$$R_k = B_k + \sum_{i=1}^{Q} \mathfrak{L}_{ik} a_i + \sum_{i=1}^{Q}\sum_{j=1}^{Q} \mathcal{N}_{ijk} a_i a_j. \tag{25}$$

At this point, a closure term must be introduced to account for the effect of truncation. This gives us

$$\frac{da_k}{dt} = R_k + \tilde{R}_k, \tag{26}$$

where $\tilde{R}_k$ accounts for the residual effects of the discarded modes. An ANN architecture is introduced to model this term. For the purpose of training, we need to assess what our modal quantities would evolve like if there was *no* truncation (i.e., a pure evolution of the PDE in a transformed space). We may obtain this by using our underlying PDE

$$\frac{\partial u}{\partial t} = L[u] + N[u;u], \tag{27}$$

and applying our familiar projection to get

$$\left(\phi_k, \frac{\partial u}{\partial t}\right) = (\phi_k, L[u] + N[u;u])\,, \tag{28}$$

which gives us

$$\frac{da_k}{dt} = (\phi_k, L[u] + N[u;u])\,, \tag{29}$$

where the left hand side is computed by noting that

$$\left(\phi_k, \frac{\partial u}{\partial t}\right) = \left(\phi_k, \frac{\partial}{\partial t}\left(\bar{u}(x) + \sum_{j=1}^{Q} a_j(t)\phi_j(x)\right)\right), \tag{30}$$

and the definition of orthogonality given by Eq. (7). Our true projected right hand side thus becomes

$$\mathfrak{R}_k = (\phi_k, L[u] + N[u;u])\,. \tag{31}$$

One may now compare Eqs. (26) and (31) to obtain

$$R_k + \tilde{R}_k = \mathfrak{R}_k. \tag{32}$$

Thus to summarize: R_k is our standard truncated GP obtained using either POD or Fourier bases, $\tilde{R}_k$ is the closure we would like to model, $\mathfrak{R}_k$ is the true projection obtained by transforming the exact solution (i.e., full-order solution) to the new space spanned by the POD or Fourier bases. We note that $\mathfrak{R}_k$ are determined and utilized solely for the purpose of obtaining the closure terms $\tilde{R}_k$ for different physical parameters in the training phase. To that end, our ANN mechanism for the estimation of $\tilde{R}_k$ is also tested within and outside the physical range of the data used to obtain $\mathfrak{R}_k$.

5.2 Network architecture

The basic structure of the simple feed-forward artificial neural network consists of $\mathbb{L}$ layers with each layer possessing a predefined number of unit cells called neurons. Each of these layers has an associated transfer function and each unit cell has an associated bias. Any input to the neuron has a bias added to it followed by activation through the transfer function. To describe this process using equations, we have [23]

$$\mathbf{S}^l = \mathbf{W}^l \mathbf{X}^{l-1}, \tag{33}$$

which represents a single neuron in the l^{th} layer receiving a vector of inputs $\mathbf{S}^l$ from the $(l-1)^{\text{th}}$ layer. Here $\mathbf{W}^l$ stands for a matrix of weights linking the $l-1$ and l layers with $\mathbf{X}^{l-1}$ being the output of the $(l-1)^{\text{th}}$ layer. The output of the l^{th} layer is now given by

$$\mathbf{X}^l = G_l(\mathbf{S}^l + \mathbf{B}^l), \tag{34}$$

where $\mathbf{B}^l$ is the matrix of biasing parameters for the l^{th} layer. Every node (or unit cell) at a particular layer l has an associated transfer function G_l which acts on its input and bias to produce an output which is 'fed forward' through the network. The nodes which take the raw input value of our training data set (i.e., the nodes of the first layer in the network) perform no computation (i.e., they do not have any biasing or activation through a transfer function). The next layers are a series of unit cells which have an associated bias and activation function which perform computation on their inputs. These are called the hidden layers with the individual unit cells known as neurons. Note that it is common in literature to consider these the only layers in the network. The final layer in the network is that of the outputs. The output layers generally have a linear activation function with a bias which implies a simple summation of inputs incident to a unit cell with its associated bias. We clarify the choice of linear activation in the output layers is generally typical of regression problems mapping from a set of real number inputs to real number outputs, i.e.,

$$f : \mathbb{R}^n \to \mathbb{R}^m, \tag{35}$$

unlike for classification purposes where some sort of hard limit function is generally used to classify outputs into one of k categorical values as

$$f : \mathbb{R}^n \to \{1, \ldots, k\}. \tag{36}$$

In this investigation, we have used one hidden layer of neurons between the set of inputs and targets with a Tan-Sigmoid activation function. The Tan-Sigmoid function can be expressed as

$$G_1(a) = \frac{2}{1 + \exp(-2a)} - 1. \tag{37}$$

The transfer function G_1 calculates the neuron's output given its net input. In theory, any differentiable function can qualify as an activation function in the hidden layer [97], however, only a small number of functions which are bounded, monotonically increasing and differentiable are used for this purpose. While it has been reported that sigmoidal activation functions saturate across a large portion of their domain [37], our reasoning behind the use of the classical Tan-Sigmoid activation was to leverage the benefit of the saturation behavior to obtain improved aggregate behavior. We note that our training data is presented to the neural network in the form of values normalized between -1 and 1 which suits the transformation provided by the activation function. The choice of using an ANN is motivated by its excellent performance as a forecasting tool [22, 48] and its general suitability in the machine learning and function estimation domain (e.g., see [39] and references therein). We note that our linear activation for the output layer is given by

$$G_2(a) = a. \tag{38}$$

For our study we shall utilized a 'trained' network architecture to estimate the statistical behavior of a set of previously obtained data, a-posteriori. Our input matrix is

$$\mathbf{X}^0 = \left[\mathbf{x}_1^0 \ \mathbf{x}_2^0 \ \ldots \ \mathbf{x}_{n_s}^0 \right], \tag{39}$$

where $\mathbf{x}_i^0$ is the i^{th} input sample (out of a total of n_s samples) of a multidimensional input vector. Our weights connecting the inputs to the middle (hidden) layer are given by

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{w}_1^{1\intercal} \\ \mathbf{w}_2^{1\intercal} \\ \vdots \\ \mathbf{w}_{n_q}^{1\intercal} \end{bmatrix}, \tag{40}$$

where $\mathbf{w}_i$ is the column vector of weights that when paired with an input vector produces the weighted sum of inputs to the i^{th} neuron (out of a total of n_q neurons) in the hidden layer and the superscript $\intercal$ represents a transpose. The study we have utilized here prescribes *only* one hidden layer of neurons and the biases for these may be given by

$$\mathbf{B}^1 = \begin{bmatrix} \mathbf{b}_1^{1\intercal} \\ \mathbf{b}_2^{1\intercal} \\ \vdots \\ \mathbf{b}_{n_q}^{1\intercal} \end{bmatrix}. \tag{41}$$

We clarify that the biases given by $\mathbf{B}^1$ consist of a column vector identically repeated across n_s columns. In other words, the same column of vector of first layer biases is added to the product of first weights and each individual input and can be denoted as $\mathbf{b}^1$, i.e.,

$$\mathbf{B^1} = \begin{bmatrix} \mathbf{b}^1 & \mathbf{b}^1 & \dots \end{bmatrix}. \tag{42}$$

The output of the lone hidden layer of neurons becomes

$$\mathbf{X}^1 = G_1(\mathbf{W}^1\mathbf{X}^0 + \mathbf{B}^1), \tag{43}$$

where $G_1(\mathbf{X})$ implies our tan-sigmoid activation procedure on each element of a matrix $\mathbf{X}$. The weight matrix of the second (i.e., the output) layer may be given as

$$\mathbf{W}^2 = \begin{bmatrix} \mathbf{w}_1^{2\intercal} \\ \mathbf{w}_2^{2\intercal} \\ \vdots \\ \mathbf{w}_{n_o}^{2\intercal} \end{bmatrix}. \tag{44}$$

We also prescribe biases for the second layer given by

$$\mathbf{B}^2 = \begin{bmatrix} \mathbf{b}_1^{2\intercal} \\ \mathbf{b}_2^{2\intercal} \\ \vdots \\ \mathbf{b}_{n_o}^{2\intercal} \end{bmatrix}, \tag{45}$$

where once again, our bias consists of the repetition of a single column vector of randomly distributed small numbers and we may define this vector as $\mathbf{b}^2$, i.e.,

$$\mathbf{B}^2 = \begin{bmatrix} \mathbf{b}^2 & \mathbf{b}^2 & \dots \end{bmatrix}. \tag{46}$$

Our outputs may thus be represented as

$$\mathbf{X}^2 = G_2(\mathbf{W}^2\mathbf{X}^1 + \mathbf{B}^2) = \mathbf{W}^2\mathbf{X}^1 + \mathbf{B}^2, \tag{47}$$

which must be trained against a set of targets corresponding to each input vector given by

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \dots & \mathbf{t}_{n_s} \end{bmatrix}. \tag{48}$$

An ANN architecture is devised which aids us in developing a nonlinear relationship between a set of inputs given by the Reynolds number (Re), the present time (t) and the truncated GP projections (i.e., $R_1, R_2, \dots, R_Q$). The outputs are given by the closures (i.e., $\tilde{R}_1, \tilde{R}_2, \dots, \tilde{R}_Q$). The architecture of our ANN is shown in Fig. 2 and n_q is our number of hidden neurons. We thus have $Q + 2$ inputs to our network with n_q neurons to obtain Q outputs. For instance if the number of inputs, outputs, neurons and samples is given by n_i, n_o, n_q and n_s respectively we have the following dimensions for our relevant matrices:

$$\begin{aligned} &\mathbf{X}^0 \to n_i \times n_s \quad \mathbf{X}^1 \to n_q \times n_s \quad \mathbf{X}^2 \to n_o \times n_s \\ &\mathbf{W}^1 \to n_q \times n_i \quad \mathbf{W}^2 \to n_o \times n_q \quad \mathbf{B}^1 \to n_q \times n_s \quad \mathbf{B}^2 \to n_o \times n_s \\ &\mathbf{H}^\intercal \to n_q \times n_s \quad \mathbf{S}^2 \to n_o \times n_s \quad \mathbf{T} \to n_o \times n_s. \end{aligned} \tag{49}$$

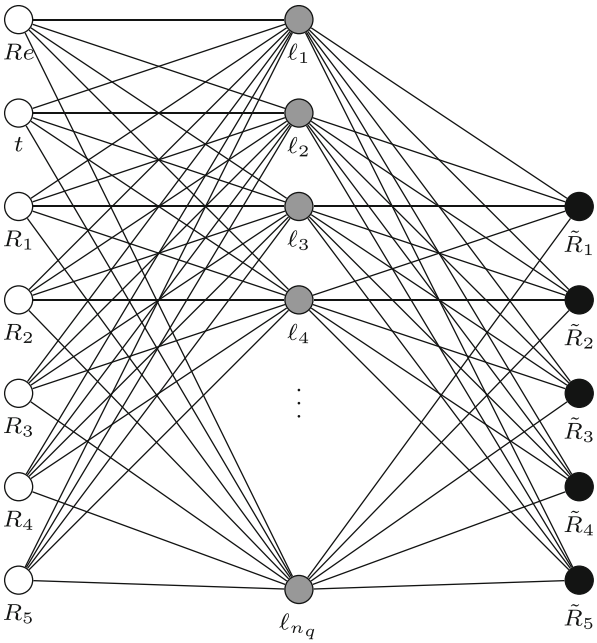


Fig. 2 The proposed ANN architecture for closure modeling of ROMs for a number of truncated modes $Q = 5$

5.3 Bayesian regularization

The training of a desired ANN is carried out by minimizing the error between the target and the outputs to determine a set of best-fit parameters. These best-fit parameters are the biases and linear weights that have captured the underlying relationship between the targets and inputs and may now be used to predict target data for inputs a posteriori. The advantage of using the ANN approach over traditional statistical regression models is that comparatively smaller data sets for training are suitable. Traditionally, the objective function of a single layer feed-forward neural network training is given by:

$$E_D = \sum_{i=1}^{n_s} \|\mathbf{t}_i - \mathbf{x}_i^2\|_2, \tag{50}$$

which is a classic element wise 2-norm error between the outputs (i.e., columns of $\mathbf{X}^2$ from layer $l = 2$) and the targets. There is an important caveat to the 'squared-error' based training of neural networks: regularization techniques are imperative for the prevention of *overfitting* noise in our data. In our investigation, we shall use the Bayesian regularization (BR) and extreme learning machine (ELM) approaches which have become popular for their regularization ability.

The Bayesian regularization training procedure augments the performance function of the ANN training by penalizing the sum of the squared weights of the network dynamically through the process of network error minimization [30, 60]. This is

in contrast to the traditional methods which are oriented to minimizing the sum of squared errors alone (which is dangerous for noisy data as it promotes excessive localization). This modified objective function is then implemented in a Bayesian framework where the parameters in the network (i.e., the weights and the biases) are considered as random variables. Mathematically we have, for the objective function of the BR framework:

$$F = \beta E_D + \alpha E_W, \tag{51}$$

where E_W is the sum of squared weights and α, β are the weight coefficients which are dynamically adjusted during training. The scalar E_W may be mathematically represented for our single hidden layer network as an element wise 2-norm of the weight matrices, i.e.,

$$E_W = \|\mathbf{W}^1\|_2 + \|\mathbf{W}^2\|_2. \tag{52}$$

If $\alpha \gg \beta$, the training aims to reduce weight sizes at the expense of network errors thereby producing a smoother network response. The main challenge is the setting of the weight coefficients after which a regular gradient based optimization technique may be used to adjust the model coefficients (i.e., the weights $\mathbf{W^1}$, $\mathbf{W^2}$ and the biases $\mathbf{B^1}$, $\mathbf{B^2}$) for one iteration since we have access to first-order information about the system through the process of backpropagation. For the choice of gradient based optimization, we utilize the Levenberg-Marquardt (LM) technique [54, 62] which is well suited to fast search for minima through its hybrid steepest-descent and Gauss-Newton's method approach. Once one step of the standard LM algorithm (the details of which are expressed below) is completed from the initial condition where $\alpha = 0$, $\beta = 1$ with layer weights initialized according to the Nguyen-Widrow method [68], an expression for the effective number of parameters is computed from the knowledge of the Hessian matrix available from the LM algorithm given by:

$$\gamma = P^{tot} - 2\alpha \mathrm{tr}(\mathbf{L})^{-1}, \tag{53}$$

where P^{tot} is the total number of parameters (i.e., weights and biases of the entire network fixed during the choice of network architecture) and the Hessian $\mathbf{L}$ is approximated by:

$$\mathbf{L} \approx 2\beta \mathbf{J}^\intercal \mathbf{J} + 2\alpha \mathbf{I}, \tag{54}$$

where $\mathbf{J}$ is the Jacobian matrix of the training set errors obtained from the chain rule of backpropagation [38]. We define a vector of all network parameters (corresponding to the model coefficients for optimization) given by $\mathbf{z}$ (refer equation 12.36 in [23]) as

$$\mathbf{z} = \left[\mathbf{w}_1^1 \ \mathbf{w}_2^1 \ \ldots \ \mathbf{w}_{n_q}^1 \ \mathbf{b}^1 \ \mathbf{w}_1^2 \ \mathbf{w}_2^2 \ \ldots \ \mathbf{w}_{n_o}^2 \ \mathbf{b}^2 \right]^\intercal. \tag{55}$$

We remind the reader that the bias vectors ($\mathbf{b}^1$ and $\mathbf{b}^2$) in the above expression constitute just two column vectors and therefore the total number of parameters for model optimization is considerably reduced. The LM algorithm, may then be utilized to obtain a new estimate of the network parameters:

$$\mathbf{z}^{new} = \mathbf{z} + \Delta \mathbf{z}, \tag{56}$$

where

$$\Delta \mathbf{z} = -[\mathbf{J}^\intercal \mathbf{J} + \mu \mathbf{I}]^{-1} \mathbf{J}^\intercal \mathbf{e}. \tag{57}$$

Note that this algorithm is a *hybrid* between the Gauss-Newton method and the steepest descent approach for function minimization with **e** being the absolute value vector of network errors given by

$$\mathbf{e} = \left| \mathbf{t}_i - \mathbf{x}_i^2 \right|. \tag{58}$$

When the scalar μ is small, the algorithm behaves more like the Gauss-Newton method of root finding using the Hessian matrix $\mathbf{J}^\intercal\mathbf{J}$. When μ is large, the algorithm becomes steepest descent with a small step size. In brief, the parameter μ is decreased after each successful step that reduces the objective function and increased only if a tentative step increases the objective function. The tentative step is recomputed using a larger value of μ. Eventually, a step that reduces the objective function will be obtained as an increasing μ favors the direction of steepest descent. Thus, the algorithm provides a compromise between the speed of the Gauss-Newton method and the guaranteed convergence of the steepest descent algorithm. New estimates for the weighting coefficients are then obtained as

$$\alpha = \frac{\gamma}{2E_W(\mathbf{z}^{new})} \quad \text{and } \beta = \frac{P^{tot} - \gamma}{2E_D(\mathbf{z}^{new})}, \tag{59}$$

where E_W and E_D are updated to account for newer values of $\mathbf{W}^1$, $\mathbf{W}^2$, $\mathbf{B}^1$, $\mathbf{B}^2$. An updated value of α and β, gives us a newly weighted performance function which may be utilized for the next step of the minimization process. Essentially the BR algorithm gives us a dynamic alteration of the objective function such that the sum of the squared weights and biases are kept low. Algorithm 1 summarizes the BR training methodology and we refer the interested reader to an excellent discussion of the implementation of LM into BR in [23]. The BR algorithm is equipped with a convergence criteria given by a maximum value of μ (implying that the minima of the objective function has been reached) or that a value of the gradient is obtained which is lower than the minimum gradient threshold $(\nabla \mathbf{z})_{min}$ specified prior to the start of the minimization process.

5.4 Extreme learning machine

In this section we detail the extreme learning machine approach to generalized single layer feedforward ANN training. This methodology was proposed in [43] for extremely fast training of a single layer feedforward ANN based on the principles of the least squares approximation. Using ELM terminology we may represent the output of the hidden layer of neurons as

$$\mathbf{H}^\intercal = G_1(\mathbf{W}^1\mathbf{X}^0 + \mathbf{B}^1). \tag{60}$$

The ELM approach mandates for zero biases in the output layer (i.e., $\mathbf{B}^2 = 0$). Our outputs of the ELM thus become

$$\mathbf{X}^2 = G_2(\mathbf{W}^2\mathbf{H}^\intercal) = \mathbf{W}^2\mathbf{H}^\intercal, \tag{61}$$

which must be trained against a set of targets **T**. The ELM training mechanism is given as follows: in order to calculate the matrix $\mathbf{W}^2$, we must recognize

Algorithm 1 Bayesian regularization

Given $\mathbf{X}^0$ and $\mathbf{T}$ ▷ Given training inputs and targets
$\alpha = 0$ and $\beta = 1$ ▷ Our regularization initial condition
$(\nabla \mathbf{z})_{min} = 10^{-7}$ and $\mu_{max} = 10^{10}$ ▷ Convergence criteria
$\mu_{init} = 0.001$ ▷ Initial μ
Initialize $\mathbf{z}$ ▷ Initialize non-zero random weights and bias
while not converged **do** ▷ Convergence criteria
    Calculate $\Delta \mathbf{z}$ from Eqs. (56–58)
    if F increases **then**
        Increase $\mu \rightarrow 10 \times \mu$ ▷ Shift toward steepest descent
    else
        Decrease $\mu \rightarrow 0.1 \times \mu$ ▷ Shift toward Gauss-Newton
        Calculate $\mathbf{z}_{new} = \mathbf{z} + \Delta \mathbf{z}$ ▷ Successful update
        Calculate new α and β from Eqs. (59) ▷ Penalize sum of squared coefficients
        Update objective function definition using Eq. (51) ▷ To ensure generalization
    end if
end while

that an optimal configuration of the weights in the second layer $\mathbf{W}^2_{op}$ should satisfy

$$\mathbf{W}^2_{op}\mathbf{H}^\intercal = \mathbf{T}, \tag{62}$$

or by taking a transpose of both sides

$$\mathbf{H}\mathbf{W}^{2^\intercal}_{op} = \mathbf{T}^\intercal, \tag{63}$$

which leads us to the following expression for the optimal weights

$$\mathbf{W}^{2^\intercal}_{op} = \mathbf{H}^\dagger \mathbf{T}^\intercal. \tag{64}$$

We clarify that the ELM training mechanism works on all of the training data at once through the calculation of the aforementioned pseudoinverse. The weights $\mathbf{W}^1$ and biases $\mathbf{B}^1$ are generated initially using random numbers. The matrix given by $\mathbf{H}^\dagger$ is calculated using a generalized Moore-Penrose pseudoinverse of $\mathbf{H}$ [83]. Once the optimal weights of the second layer are obtained, our network is trained for deployment. Due to the small random number values chosen for the weights in the first layer, our network is well suited to highly effective abstraction of the training process due to a smaller degree of freedom of the overall ANN. We must note however that the success of this method hinges on the utilization of small values for $\mathbf{W}^1$ and $\mathbf{B}^1$ and an improper choice of these initializations would lead to poor training and performance. In this investigation we have utilized a probability density function for our random numbers which is uniformly distributed between -1 and 1. The obvious advantage of this approach compared to the BR method is the substitution of an iterative minimization to a direct least squares approximation using the pseudoinverse but we note that we are restricted in terms of architecture to just one hidden layer for our network architecture. The procedure for the ELM method is given in Algorithm 2.

Algorithm 2 Extreme learning machine

1: Given $\mathbf{X}^0$ and $\mathbf{T}$ ▷ Given inputs and targets
2: Initialize $\mathbf{W}^1$ and $\mathbf{B}^1$ ▷ Initialize non-zero random parameters
3: Calculate $\mathbf{H}^\intercal$ ▷ From Eq. (60)
4: Transpose $\mathbf{H}^\intercal$ to obtain $\mathbf{H}$
5: Calculate $\mathbf{H}^\dagger$ ▷ Moore-Penrose pseudoinverse of $\mathbf{H}$
6: Calculate layer 2 weights $\mathbf{W}^{2^\intercal} = \mathbf{H}^\dagger \mathbf{T}^\intercal$ ▷ Least squares solution for optimal weights

5.5 Performance comparison between BR and ELM

Before proceeding to the results of our investigation, it would be useful to compare the computational cost of the BR and ELM training algorithms. In addition, since both algorithms are designed for regularization ability, it would also be helpful to see their training performance for noisy data. For this purpose, we choose a simple one dimensional problem to test both approaches given by:

$$f(x) = \begin{cases} \frac{\sin(\pi^2 x)}{\pi^2 x}, & x \neq 0 \\ 1, & x = 0\,. \end{cases} \tag{65}$$

A training data set is devised by obtaining 51 samples of the above function at equal intervals within $x \in [-2, 2]$. In addition two other training data sets are obtained with varying amounts of noise. We use a uniformly distributed pseudorandom number generator and multiply the random number with an amplitude to mimic noise which is added to each training data point. To sum up, our test cases are those with zero noise, with an amplitude of ± 0.05 and ± 0.1. The ELM and BR algorithms were then tested on all training data sets. Five trials for each data set were run and training times were recorded as given in Table 1. It is apparent that an increased noise in the data causes an increased duration for the convergence of the BR algorithm. On the other hand, the ELM method relies on the calculation of a pseudoinverse of a matrix, the dimensions of which depend solely on the number of hidden neurons and the size of the input data set. The regularization ability of the algorithms can be

Table 1 Comparison of training times (in seconds) for the BR and ELM training algorithms for our model test case given by Eq. (65)

Trial Number	Amplitude = 0		Amplitude = 0.05		Amplitude = 0.1	
	ELM	BR	ELM	BR	ELM	BR
Trial 1	0.002514	0.91155	0.002409	2.03174	0.002362	1.91132
Trial 2	0.001187	0.51236	0.001286	2.06923	0.001208	5.29371
Trial 3	0.001373	0.70040	0.001565	2.86361	0.001638	1.64957
Trial 4	0.001203	0.81693	0.001230	1.33273	0.001186	2.28243
Trial 5	0.001335	1.24820	0.001099	2.26790	0.001328	2.06521

Here amplitude refers to the approximate order of the random number that has been added to each data point. Note how increasing noise causes an increase in BR convergence times

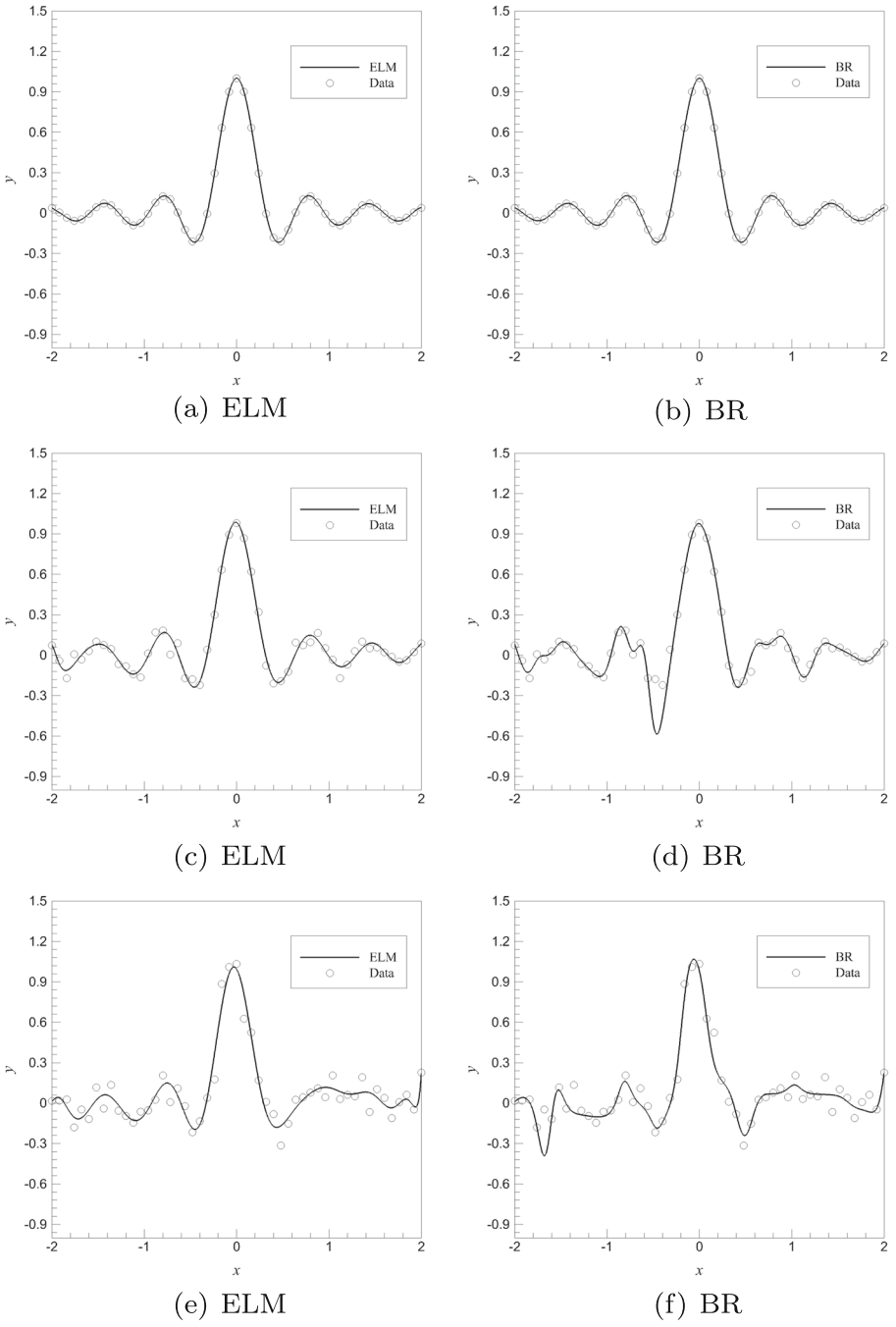


Fig. 3 Comparison of the regularization ability of ELM and BR algorithms with no noise (top), a noise of amplitude 0.05 (middle) and a noise of amplitude 0.1 (bottom). Note how the ELM generally tends to capture the true function behavior better than BR

examined through the performance of the trained ANN output as shown in Fig. 3. It can be observed that the ELM algorithm generally does a better job at extracting the underlying function behavior. However, we caution the reader that this behavior is not completely conclusive for either algorithm and a variety of test cases must be examined for stronger conclusions.

6 Results

This section details the results of our investigation for the viscous Burgers equation problem. Our test case is given by the following initial and boundary conditions:

$$\begin{aligned} u(x,0) &= \frac{x}{1+\sqrt{\frac{1}{A_0}}\exp\left(Re\frac{x^2}{4}\right)} \\ u(0,t) &= 0 \\ u(L,t) &= 0, \end{aligned} \tag{66}$$

where the length of our domain $L = 1$ and maximum time $t_m = 2$. The PDE for the Burgers equation with the aforementioned boundary and initial conditions may be

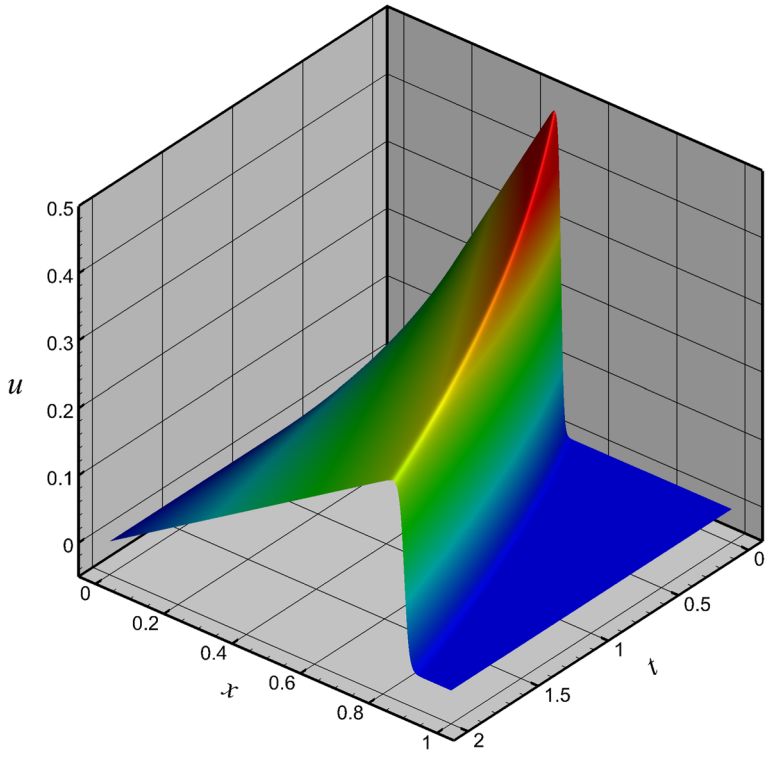


Fig. 4 Exact solution of Burgers equation at $Re = 1000$

solved exactly to obtain an analytical formulation for the time evolution for the field variable $u(x,t)$ given by [61]

$$u(x,t) = \frac{\frac{x}{t+1}}{1+\sqrt{\frac{t+1}{A_0}}\exp\left(Re\frac{x^2}{4t+4}\right)}, \tag{67}$$

where $A_0 = \exp(Re/8)$. This exact expression is used to generate snapshot data for our ROM assessments. We compute inner products using the well-known Simpson's rule [64] and a third-order Runge-Kutta scheme is utilized for solving the resulting ROM. The reader is directed to [80] for a more detailed discussion of the numerical methods used in this investigation. A space-time evolution of our solution field is presented in Fig. 4 for an $Re = 1000$. Figure 5 shows the distribution of the eigenvalues as a function of the total energy captured. It is apparent that retaining the eigenvectors corresponding to the first 5 eigenvalues causes a 94% capture of energy for the system and the first 20 modes contribute to 99.93% of the total energy of the system at $Re = 1000$. Note that a variation in Re would manifest itself with a higher number of modes needed to capture an equivalent amount of energy (if Re increases) and a lower number of modes (if Re decreases).

Before proceeding with an elaboration of our results, it is worthwhile to elaborate on the definition of our models:

- Fourier-GP-ROM: Reduced-order model obtained by the standard Galerkin projection to Fourier modes.
- POD-GP-ROM: Reduced-order model obtained by the standard Galerkin projection to POD modes.

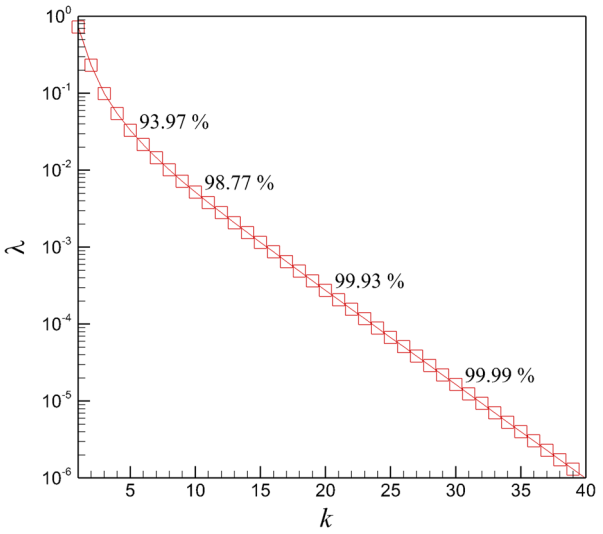


Fig. 5 Eigenvalues of the snapshot data matrix. Note that first five modes captures almost 94% of the energy

– Fourier-ANN-ROM: Proposed reduced-order model with ANN closure applied to Fourier modes.
– POD-ANN-ROM: Proposed reduced-order model with ANN closure applied to POD modes.
– Fourier-True: Projection of exact solution of PDE into space spanned by Fourier modes.
– POD-True: Projection of exact solution of PDE into space spanned by POD modes.

Figure 6 shows the time evolution of the solution field when using a ROM implementation of the Fourier modes. On increasing the number of retained modes Q, one can see a convergence to the true behavior of the PDE. However, for $Re = 1000$, some oscillations can still be observed for $Q = 30$ retained modes. When using

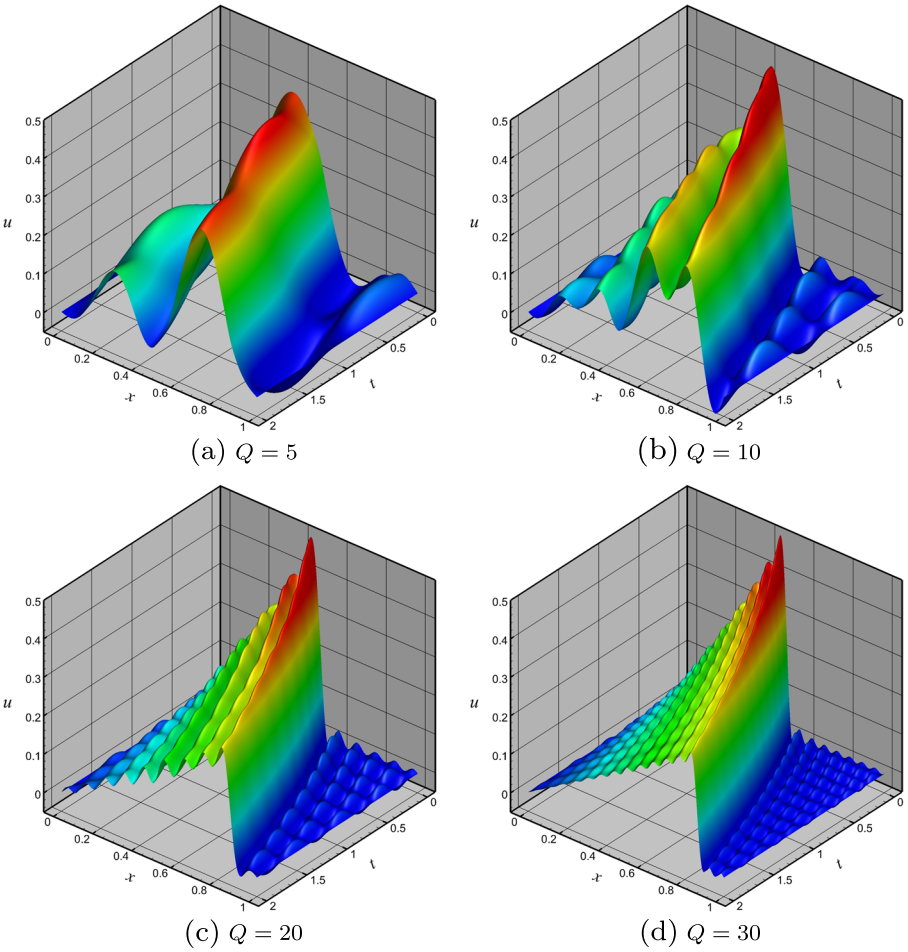


Fig. 6 Fourier-GP-ROM with different number of modes at $Re = 1000$

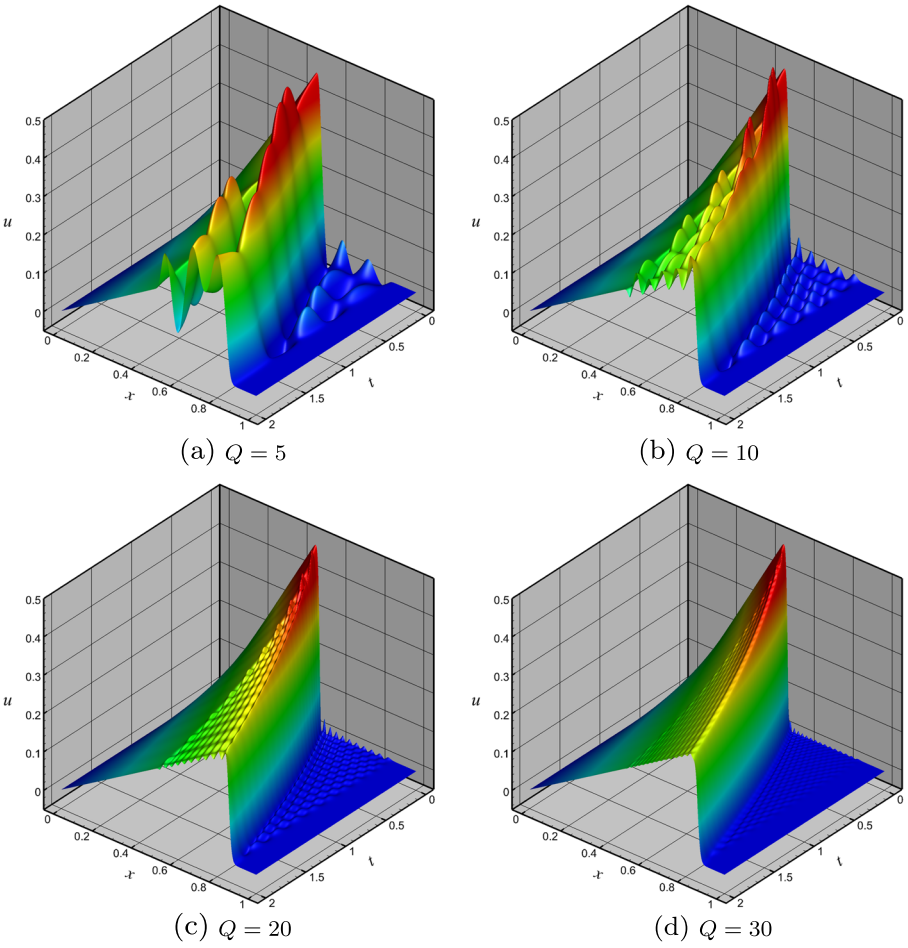


Fig. 7 POD-GP-ROM with different number of modes at $Re = 1000$

the POD modes instead, as shown in Fig. 7, a quicker convergence is observed with increasing modes. This is expected since POD bases are generated from exact snapshot data.

Figure 8 shows the performance of the Fourier-GP-ROM and POD-GP-ROM methods against the Fourier-ANN-ROM and POD-ANN-ROM approaches for the first five modes. The true projections are also shown for the purpose of comparison. It is immediately apparent that the use of the ANN closures lends a 'stabilization' effect to the modal evolution. We remark that the training data for the ANN architecture includes modal evolution data sets obtained by $Re \in [200, 300, \ldots, 1200]$. At $Re = 1000$, which is a value of Reynolds number within the data set, an excellent performance is observed with the ANN-ROM approaches virtually indistinguishable from the true modes. We note that the BR training technique has been used for these figures.

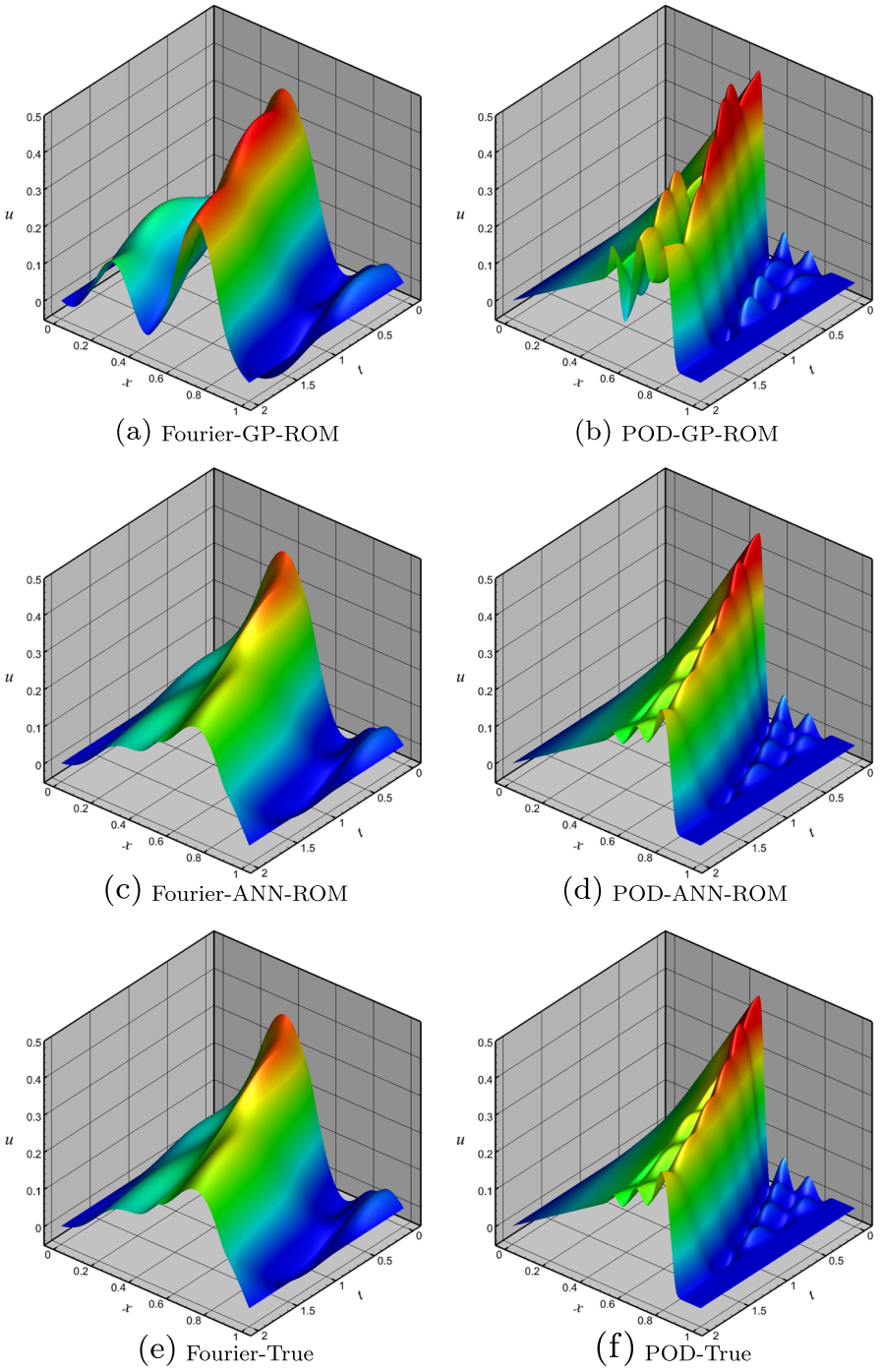
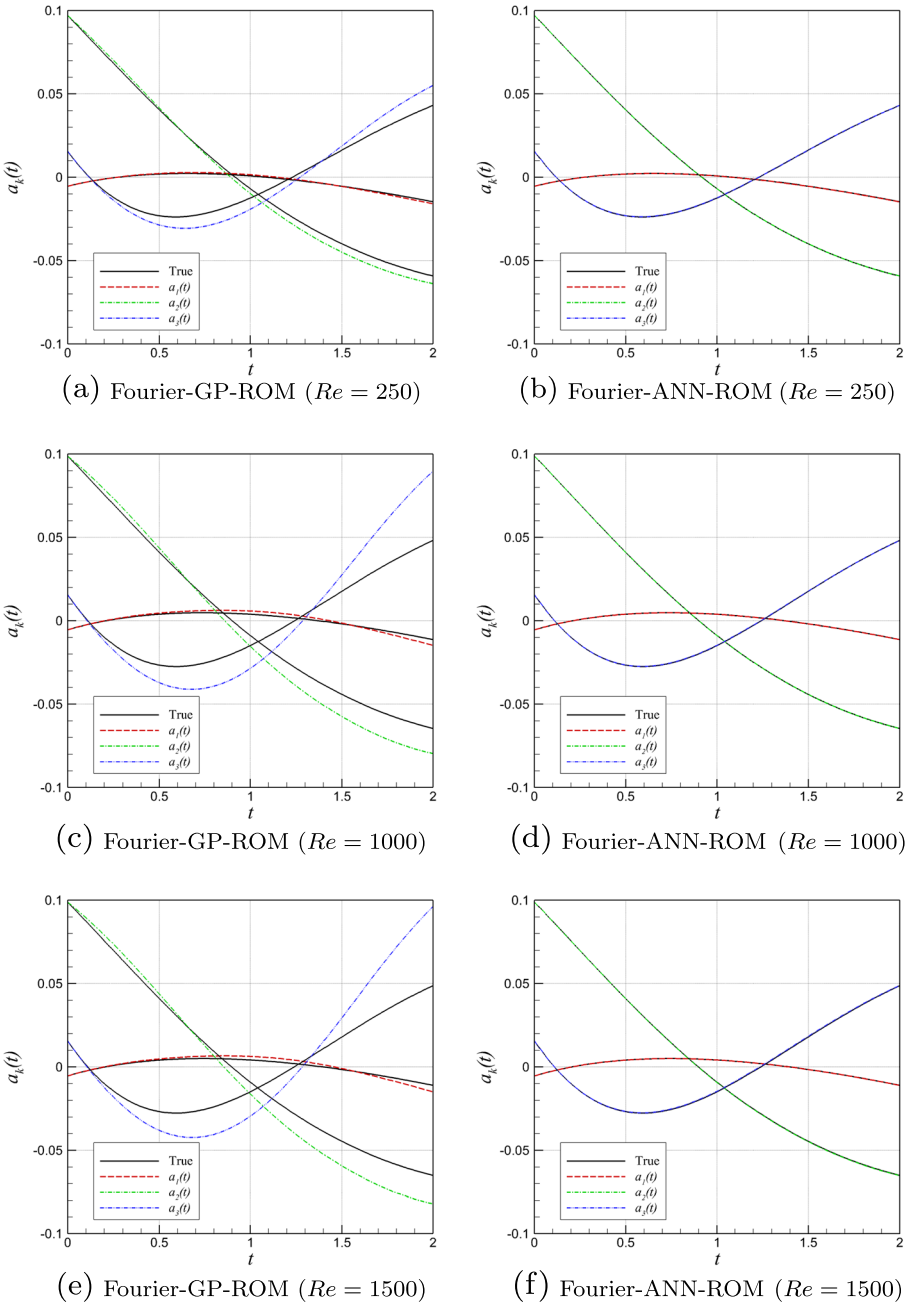


Fig. 8 Predictive performance of the models using $Q = 5$ modes at $Re = 1000$. ANN with Bayesian regularization with 10 neurons

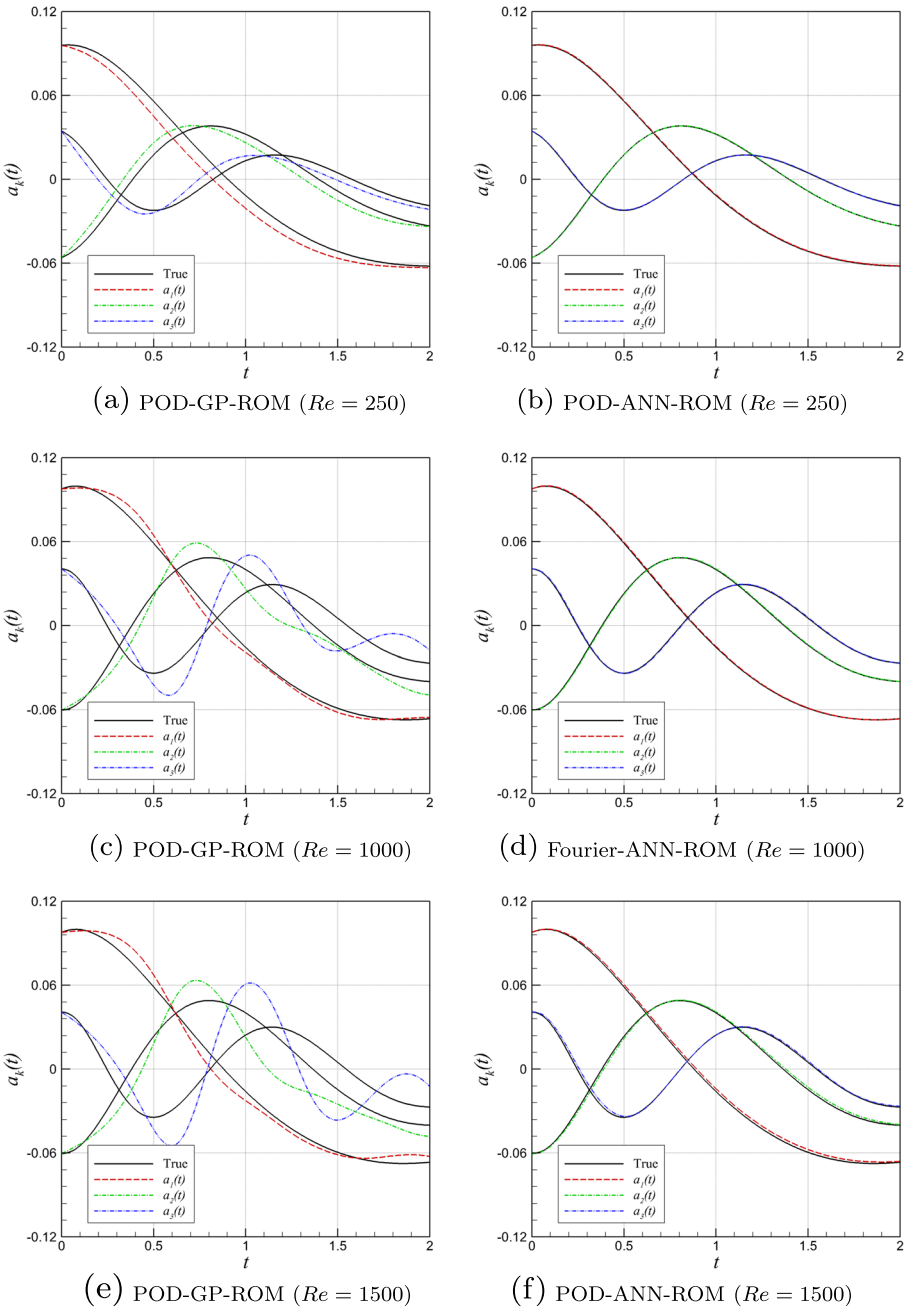


(a) Fourier-GP-ROM ($Re = 250$) (b) Fourier-ANN-ROM ($Re = 250$)

(c) Fourier-GP-ROM ($Re = 1000$) (d) Fourier-ANN-ROM ($Re = 1000$)

(e) Fourier-GP-ROM ($Re = 1500$) (f) Fourier-ANN-ROM ($Re = 1500$)

Fig. 9 Time evolution of first few modes when using the Fourier basis. ANN with Bayesian regularization with 10 neurons. Note that $Re = 250$ and $Re = 1500$ cases are not included in our training set of $Re \in [200 - 1200]$



(a) POD-GP-ROM ($Re = 250$) (b) POD-ANN-ROM ($Re = 250$)

(c) POD-GP-ROM ($Re = 1000$) (d) Fourier-ANN-ROM ($Re = 1000$)

(e) POD-GP-ROM ($Re = 1500$) (f) POD-ANN-ROM ($Re = 1500$)

Fig. 10 Time evolution of first few modes when using the POD basis. ANN with Bayesian regularization with 10 neurons. Note that $Re = 250$ and $Re = 1500$ cases are not included in our training set of $Re \in [200 - 1200]$

The real challenge of our proposed closure lies in its performance for values of physical parameters which do not lie in the training data set. The ability to predict modal evolution for values of the Reynolds number which are not a part of the training data set would be favorable for optimal control applications where small differences in physical parameters would not require a complete high-fidelity numerical simulation prior to ROM deployment. Figure 9 describes the time series evolution of the first three modes for the ROM implementations with and without the proposed closures. The bases used for the Galerkin projection here are the Fourier bases. The ANN closure clearly leads to a much closer alignment with the true evolution of the modes. What is notable here is that for $Re = 250$ and $Re = 1500$, the ANN closure performs excellently as well in comparison to the Fourier-GP-ROM which can be seen to deviate from the trends of the true evolution. A similar conclusion can also be drawn from Fig. 10, where we have now used the POD bases for our transformation space. It can also be observed that the POD modes show a larger deviation from the true evolution without the use of the closure.

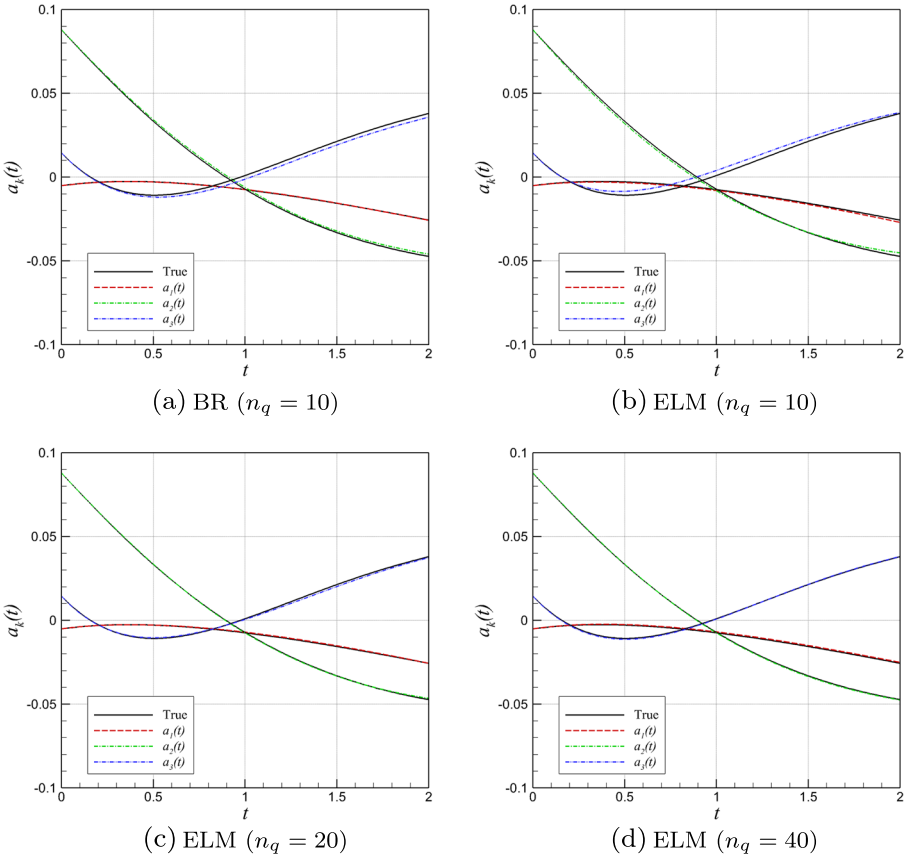


Fig. 11 Time evolution of first few modes when using the Fourier basis at $Re = 100$. Note that this Reynolds number is not included in our training set of $Re \in [200 - 1200]$

Finally, we also detail the effect of the training algorithm (i.e., whether BR or ELM) for determining the ANN weights and consequent performance. We have utilized three values of $Re = 100, 750, 1500$. We remark that none of the values are a part of the training data set. Also, $Re = 100$ and $Re = 1500$ are outside the *range* of the training data. In particular, we are interested in the performance of ELM since it represents a massive opportunity for fast training requirements in dynamical systems. For the case of $Re = 100$, given by Fig. 11, we can see the ELM approach progressively converges to the true evolution with increasing number of neurons n_q. For $Re = 750$, which lies within the range of our training data set, we can see that the ELM and BR approaches are more or less equivalent with excellent agreement with the true evolution (as shown in Fig. 12). Figure 13 shows the challenging case of $Re = 1500$, where a slight deviation from the true solution can be seen for ELM at lower number of hidden neurons. Note that this behavior is expected since the ELM procedure effectively reduces the degrees of freedom of the single layer feedforward

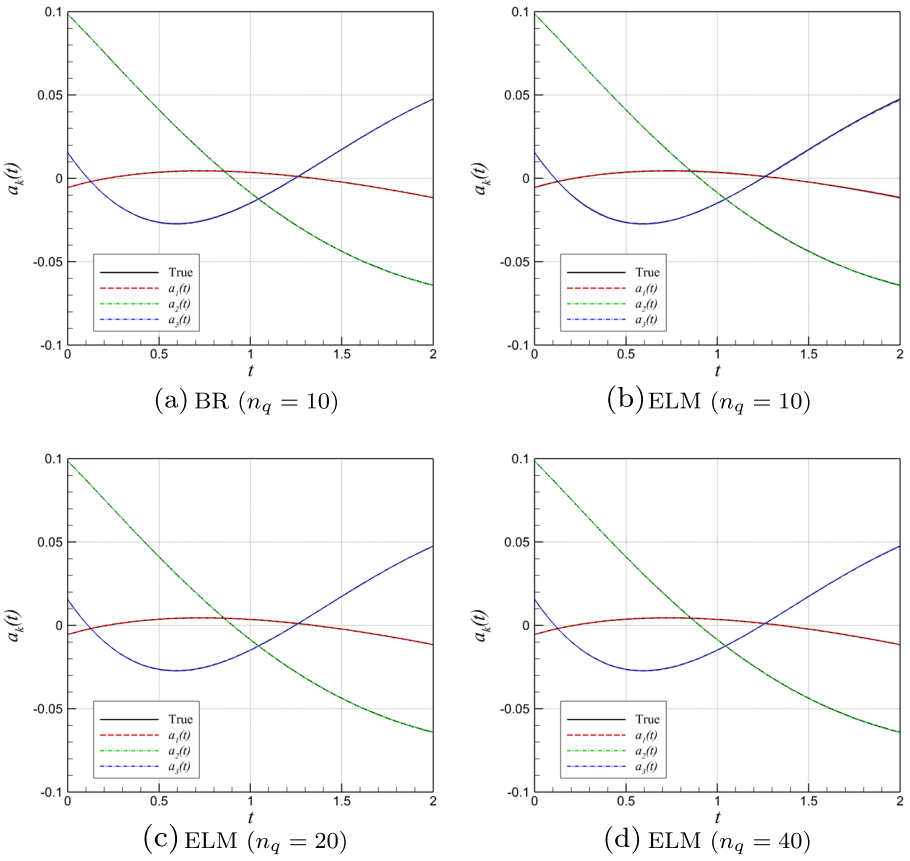
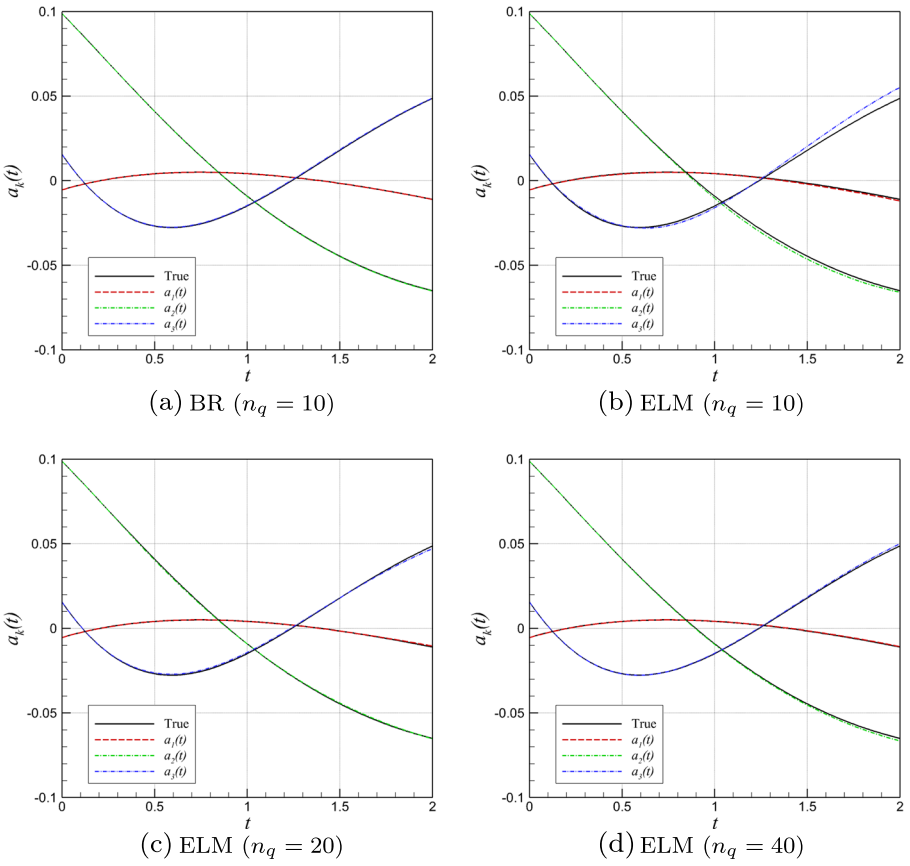


Fig. 12 Time evolution of first few modes when using the Fourier basis at $Re = 750$. Note that this Reynolds number is not included in our training set of $Re \in [200 - 1200]$



(a) BR ($n_q = 10$) (b) ELM ($n_q = 10$)

(c) ELM ($n_q = 20$) (d) ELM ($n_q = 40$)

Fig. 13 Time evolution of first few modes when using the Fourier basis at $Re = 1500$. Note that this Reynolds number is not included in our training set of $Re \in [200 - 1200]$

ANN which might prove to be a handicap for a lower number of hidden layer neurons. However, the fast training time effectively offsets the slightly higher necessity for neurons.

7 Conclusions

This work demonstrates the value of utilizing a single layer feedforward artificial neural network as a closure for reduced-order models of the viscous Burgers equation. A reduced-order model for the Burgers equation is created by transforming our partial differential equation using a new set of bases (given by either the optimal POD modes or the Fourier modes). While the Fourier modes are predefined, the POD space is constructed by using snapshot data generated from the exact solution of the system we are studying. The Galerkin projection methodology is then used to

evolve a severely truncated system with and without our proposed ANN closure. Our proposed closure consists of an ANN architecture with one hidden layer of neurons which is trained using 'true' modal evolution values generated from the exact solution of the governing law. The nonlinear relationship approximates the true value of the right hand side of the Galerkin projection system of ordinary differential equations, given the truncated modes at the current time step. It thus acts as a stabilization procedure.

Our numerical assessments reveal that our closure hypothesis proves efficient in the closure of the ROM for both Fourier and POD bases. The ANN closure is also seen to perform robustly for the modal evolution of systems with Reynolds numbers which are different from the training data set. This highlights its potential utility as a closure for ROM-GP based optimal control for fluid systems. In addition, two different training approaches are utilized prior to the deployment of the ANN for the purpose of closure. The BR and ELM training methodologies are selected for this study due to their ability to give *regularized* networks (i.e., networks which are less sensitive to noisy data). While the BR method is exceptional for its accuracy and has become a mainstay for ANN training for noisy data sets, the ELM approach is extremely popular due its high speed in training. It is observed that both approaches are capable of stabilizing modal evolution satisfactorily. In addition, an increasing number of neurons is seen to enhance the accuracy of the ELM training. This is because it has a slightly lower degree of freedom as compared to the BR method. The ELM approach is particularly promising due its application in dynamical systems which may require fast learning. A simple one dimensional training data is also examined with artificial noise using both BR and ELM approaches and the computational benefit of ELM is clearly established. This investigation leads us to several meaningful insights about the future of hybrid data and physics driven ROM. For the case of the one dimensional Burgers equation, our proposed closure may be considered to be an adequate augmentation to a POD-ROM. However, further investigations are necessary to test the performance of the closure (for example for noisy data). The use of BR and ELM (i.e., regularized training) is motivated by the desire to extract the underlying statistics of the flow (which are ultimately physics driven) and they must be ideally tested against experimental data sets with their inherent background noise.

An extension of the study described in this document lies through an investigation of the performance of this closure for higher dimensional systems and ultimately for real world flow control problems. Our ultimate goal is to utilize this methodology in flow situations with higher Reynolds number physics which prove more challenging for ROMs as well as predictive ideologies such as the regularized ANN ideas espoused here. In that situation, we may have to turn to more complicated regularization ideas which incorporate human intuition of physics (refer [73, 74]) to ensure closure behavior does not lead to predicted physics corruption. Thus our ultimate aim is to preserve statistics from a physical perspective without sacrificing the turnover time critical to flow control and ROM applications for challenging test-cases reflective of real world applications.

Acknowledgments The computing for this project was performed by using resources from the High Performance Computing Center (HPCC) at Oklahoma State University.

References

1. Akhtar, I., Borggaard, J., Burns, J.A., Imtiaz, H., Zietsman, L.: Using functional gains for effective sensor location in flow control: a reduced-order modelling approach. J. Fluid Mech. **781**, 622–656 (2015)
2. Akhtar, I., Wang, Z., Borggaard, J., Iliescu, T.: A new closure strategy for proper orthogonal decomposition reduced-order models. J. Comput. Nonlinear Dyn. **7**(3), 034,503 (2012)
3. Amsallem, D., Farhat, C.: Interpolation method for adapting reduced-order models and application to aeroelasticity. AIAA J. **46**(7), 1803–1813 (2008)
4. Amsallem, D., Farhat, C.: Stabilization of projection-based reduced-order models. Int. J. Numer. Methods Eng. **91**(4), 358–377 (2012)
5. Antoulas, A.C.: Approximation of Large-Scale Dynamical Systems. SIAM, Philadelphia (2005)
6. Aubry, N., Holmes, P., Lumley, J.L., Stone, E.: The dynamics of coherent structures in the wall region of a turbulent boundary layer. J. Fluid Mech. **192**(1), 115–173 (1988)
7. Balajewicz, M., Dowell, E.H.: Stabilization of projection-based reduced order models of the Navier–Stokes. Nonlinear Dyn. **70**(2), 1619–1632 (2012)
8. Banyay, G.A., Ahmadpoor, M., Brigham, J.C.: Proper orthogonal decomposition based reduced order modeling of the very high temperature reactor lower plenum hydrodynamics. In: ASME 2014 4th Joint US-European Fluids Engineering Division Summer Meeting Collocated with the ASME 2014 12th International Conference on Nanochannels, Microchannels, and Minichannels, pp. V01DT27A013–V01DT27A013. American Society of Mechanical Engineers (2014)
9. Benner, P., Gugercin, S., Willcox, K.: A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Rev. **57**, 483–531 (2015)
10. Benosman, M., Borggaard, J., San, O., Kramer, B.: Learning-based robust stabilization for reduced-order models of 2D and 3D Boussinesq equations. Appl. Math. Model. **49**, 162–181 (2017)
11. Benosman, M., Kramer, B., Boufounos, P.T., Grover, P.: Learning-based reduced order model stabilization for partial differential equations: application to the coupled Burgers' equation. In: American Control Conference (ACC), 2016, pp. 1673–1678. IEEE (2016)
12. Bergmann, M., Bruneau, C.H., Iollo, A.: Improvement of reduced order modeling based on POD. Computational Fluid Dynamics **2008**, 779–784 (2009)
13. Borggaard, J., Hay, A., Pelletier, D.: Interval-based reduced order models for unsteady fluid flow. Int. J. Numer. Anal. Model. **4**(3-4), 353–367 (2007)
14. Borggaard, J., Iliescu, T., Wang, Z.: Artificial viscosity proper orthogonal decomposition. Math. Comput. Model. **53**(1), 269–279 (2011)
15. Borggaard, J., Wang, Z., Zietsman, L.: A goal-oriented reduced-order modeling approach for nonlinear systems. Comput. Math. Appl. **71**(11), 2155–2169 (2016)
16. Buffoni, M., Camarri, S., Iollo, A., Salvetti, M.V.: Low-dimensional modelling of a confined three-dimensional wake flow. J. Fluid Mech. **569**, 141–150 (2006)
17. Bui-Thanh, T., Willcox, K., Ghattas, O., van Bloemen Waanders, B.: Goal-oriented, model-constrained optimization for reduction of large-scale systems. J. Comput. Phys. **224**(2), 880–896 (2007)
18. Carlberg, K., Farhat, C.: A low-cost, goal-oriented 'compact proper orthogonal decomposition' basis for model reduction of static systems. Int. J. Numer. Methods Eng. **86**(3), 381–402 (2011)
19. Cazemier, W.: Proper Orthogonal Decomposition and Low Dimensional Models for Turbulent Flows. Groningen (1997)
20. Cazemier, W., Verstappen, R., Veldman, A.: Proper orthogonal decomposition and low-dimensional models for driven cavity flows. Phys. Fluids (1994-present) **10**(7), 1685–1699 (1998)
21. Cordier, L., Majd, E., Abou, B., Favier, J.: Calibration of POD reduced-order models using Tikhonov regularization. Int. J. Numer. Methods Fluids **63**(2), 269–296 (2010)
22. Dawson, C.W., Wilby, R.: An artificial neural network approach to rainfall-runoff modelling. Hydrol. Sci. J. **43**(1), 47–66 (1998)
23. Demuth, H.B., Beale, M.H., De Jess, O., Hagan, M.T.: Neural Network Design. Martin Hagan (2014)
24. Dyke, S., Spencer, B. Jr., Sain, M., Carlson, J.: Modeling and control of magnetorheological dampers for seismic response reduction. Smart Mater. Struct. **5**(5), 565 (1996)
25. Efe, M., Debiasi, M., Yan, P., Ozbay, H., Samimy, M.: Control of subsonic cavity flows by neural networks-analytical models and experimental validation. In: 43rd AIAA Aerospace Sciences Meeting and Exhibit, p. 294 (2005)

26. Efe, M.O., Debiasi, M., Ozbay, H., Samimy, M.: Modeling of subsonic cavity flows by neural networks. In: Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM'04, pp. 560–565. IEEE (2004)
27. El Majd, B.A., Cordier, L.: New regularization method for calibrated POD reduced-order models. Math. Model. Anal. **21**(1), 47–62 (2016)
28. Faller, W.E., Schreck, S.J.: Unsteady fluid mechanics applications of neural networks. J. Aircr. **34**(1), 48–55 (1997)
29. Fang, F., Pain, C., Navon, I., Gorman, G., Piggott, M., Allison, P., Farrell, P., Goddard, A.: A POD reduced order unstructured mesh ocean modelling method for moderate Reynolds number flows. Ocean Model. **28**(1), 127–136 (2009)
30. Foresee, F.D., Hagan, M.T.: Gauss-Newton approximation to Bayesian learning. In: International Conference on Neural Networks, 1997, vol. 3, pp. 1930–1935. IEEE (1997)
31. Fortuna, L., Nunnari, G., Gallo, A.: Model Order Reduction Techniques with Applications in Electrical Engineering. Springer Science & Business Media, Berlin (2012)
32. Freund, R.W.: Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation. In: Applied and Computational Ccontrol, Signals, and Circuits, pp. 435–498. Springer (1999)
33. Gaspard, P.: Chaos, Scattering and Statistical Mechanics, vol. 9. Cambridge University Press, Cambridge (2005)
34. Gillies, E.: Low-dimensional characterization and control of non-linear wake flows. Ph.D. thesis, PhD. Dissertation, University of Glasgow, Scotland (1995)
35. Gillies, E.: Low-dimensional control of the circular cylinder wake. J. Fluid Mech. **371**, 157–178 (1998)
36. Gillies, E.: Multiple sensor control of vortex shedding. AIAA J. **39**(4), 748–750 (2001)
37. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT press, Cambridge (2016)
38. Hagan, M.T., Menhaj, M.B.: Training feedforward networks with the Marquardt algorithm. IEEE Trans. Neural Netw. **5**(6), 989–993 (1994)
39. Haykin, S.S., Haykin, S.S., Haykin, S.S., Haykin, S.S.: Neural Networks and Learning Machines, vol. 3. Pearson, Upper Saddle River (2009)
40. Hocevar, M., Sirok, B., Grabec, I.: Experimental turbulent field modeling by visualization and neural networks. J. Fluids Eng. **126**, 316–322 (2004)
41. Holmes, P., Lumley, J.L., Berkooz, G.: Turbulence, Coherent Structures, Dynamical Systems and Symmetry. Cambridge University Press, Cambridge (1998)
42. Hotelling, H.: Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. **24**(6), 417 (1933)
43. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1), 489–501 (2006)
44. Imtiaz, H., Akhtar, I.: Closure modeling in reduced-order model of Burgers' equation for control applications. J. Aerosp. Eng. **231**(4), 642–656 (2017)
45. Iollo, A., Lanteri, S., Désidéri, J.A.: Stability properties of POD–Galerkin approximations for the compressible Navier–Stokes equations. Theor. Comput. Fluid Dyn. **13**(6), 377–396 (2000)
46. Kazantzis, N., Kravaris, C., Syrou, L.: A new model reduction method for nonlinear dynamical systems. Nonlinear Dyn. **59**(1), 183–194 (2010)
47. Khibnik, A., Narayanan, S., Jacobson, C., Lust, K.: Analysis of low dimensional dynamics of flow separation. In: Continuation Methods in Fluid Dynamics, vol. 74, pp. 167–178. Vieweg (2000)
48. Kim, T.W., Valdés, J.B.: Nonlinear model for drought forecasting based on a conjunction of wavelet transforms and neural networks. J. Hydrol. Eng. **8**(6), 319–328 (2003)
49. Kunisch, K., Volkwein, S.: Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition. J. Optim. Theory Appl. **102**(2), 345–371 (1999)
50. Kunisch, K., Volkwein, S.: Galerkin proper orthogonal decomposition methods for parabolic problems. Numer. Math. **90**(1), 117–148 (2001)
51. Kunisch, K., Volkwein, S.: Optimal snapshot location for computing POD basis functions. ESAIM: Mathematical Modelling and Numerical Analysis **44**(3), 509–529 (2010)
52. Lassila, T., Manzoni, A., Quarteroni, A., Rozza, G.: Model order reduction in fluid dynamics: challenges and perspectives. In: Quarteroni, A., Rozza, G. (eds.) Reduced Order Methods for Modeling and Computational Reduction. Springer, Milano (2013)

53. Lee, C., Kim, J., Babcock, D., Goodman, R.: Application of neural networks to turbulence control for drag reduction. Phys. Fluids **9**(6), 1740–1747 (1997)
54. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Q. Appl. Math. **2**(2), 164–168 (1944)
55. Ling, J., Kurzawski, A., Templeton, J.: Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J. Fluid Mech. **807**, 155–166 (2016)
56. Loève, M.: Probability Theory; Foundations, Random Sequences. D. Van Nostrand Company, New York (1955)
57. Lorenz, E.N.: Empirical orthogonal functions and statistical weather prediction. Massachusetts Institute of Technology, Department of Meteorology, Cambridge (1956)
58. Lucia, D.J., Beran, P.S., Silva, W.A.: Reduced-order modeling: new approaches for computational physics. Prog. Aerosp. Sci. **40**(1), 51–117 (2004)
59. Lumley, J.: The structures of inhomogeneous turbulent flow. In: Yaglom, A., Tatarski, V. (eds.) Atmospheric Turbulence and Radio Wave Propagation, pp. 160–178 (1967)
60. MacKay, D.J.: Bayesian interpolation. Neural Comput. **4**(3), 415–447 (1992)
61. Maleewong, M., Sirisup, S.: On-line and off-line POD assisted projective integral for non-linear problems: a case study with Burgers' equation. International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering **5**(7), 984–992 (2011)
62. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11**(2), 431–441 (1963)
63. Maulik, R., San, O.: Explicit and implicit LES closures for Burgers turbulence. J. Comput. Appl. Math. **327**, 12–40 (2018)
64. Moin, P.: Fundamentals of Engineering Numerical Analysis. Cambridge University Press, Cambridge (2010)
65. Moosavi, A., Stefanescu, R., Sandu, A.: Efficient construction of local parametric reduced order models using machine learning techniques. arXiv:1511.02909 (2015)
66. Moosavi, A., Stefanescu, R., Sandu, A.: Multivariate predictions of local reduced-order-model errors and dimensions. arXiv:1701.03720 (2017)
67. Narayanan, S., Khibnik, A., Jacobson, C., Kevrekedis, Y., Rico-Martinez, R., Lust, K.: Low-dimensional models for active control of flow separation. In: Proceedings of the 1999 IEEE International Conference on Control Applications, 1999, vol. 2, pp. 1151–1156. IEEE (1999)
68. Nguyen, D., Widrow, B.: Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In: 1990 IJCNN International Joint Conference on Neural Networks, 1990, pp. 21–26. IEEE (1990)
69. Noack, B., Papas, P., Monkewitz, P.: Low-dimensional Galerkin model of a laminar shear-layer. Tech. rep., Tech. Rep. 2002-01. Laboratoire De Mecanique des Fluides, Departement de Genie Mecanique, Ecole Polytechnique Fédérale de Lausanne, Switzerland (2002)
70. Noack, B.R., Afanasiev, K., Morzyński, M., Tadmor, G., Thiele, F.: A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. J. Fluid Mech. **497**, 335–363 (2003)
71. Noack, B.R., Morzynski, M., Tadmor, G.: Reduced-Order Modelling for Flow Control, vol. 528. Springer, Berlin (2011)
72. Pinnau, R.: Model reduction via proper orthogonal decomposition. Model Order Reduction: Theory, Research Aspects and Applications **13**, 95–109 (2008)
73. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (Part I): data-driven discovery of nonlinear partial differential equations. arXiv:1711.10561 (2017)
74. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (Part II): data-driven discovery of nonlinear partial differential equations. arXiv:1711.10566 (2017)
75. Ravindran, S.S.: A reduced-order approach for optimal control of fluids using proper orthogonal decomposition. Int. J. Numer. Methods Fluids **34**(5), 425–448 (2000)
76. Rowley, C.W.: Model reduction for fluids, using balanced proper orthogonal decomposition. Int. J. Bifurc. Chaos **15**(03), 997–1013 (2005)
77. Rowley, C.W., Dawson, S.T.: Model reduction for flow analysis and control. Annu. Rev. Fluid Mech. **49**, 387–417 (2017)
78. Roychowdhury, J.: Reduced-order modeling of time-varying systems. IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **46**(10), 1273–1288 (1999)
79. Sahan, R., Koc-Sahan, N., Albin, D., Liakopoulos, A.: Artificial neural network-based modeling and intelligent control of transitional flows. In: Proceedings of the 1997 IEEE International Conference on Control Applications, 1997, pp. 359–364. IEEE (1997)

80. San, O., Iliescu, T.: Proper orthogonal decomposition closure models for fluid flows: Burgers equation. Int. J. Numer. Anal. Model. **5**, 217–237 (2014)
81. San, O., Iliescu, T.: A stabilized proper orthogonal decomposition reduced-order model for large scale quasigeostrophic ocean circulation. Adv. Comput. Math. **41**(5), 1289–1319 (2015)
82. Schmid, P.J.: Dynamic mode decomposition of numerical and experimental data. J. Fluid Mech. **656**, 5–28 (2010)
83. Serre, D.: Matrices: Theory and Applications. Springer, New York (2002)
84. Silveira, L.M., Kamon, M., White, J.: Efficient reduced-order modeling of frequency-dependent coupling inductances associated with 3-D interconnect structures. IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part B **19**(2), 283–288 (1996)
85. Sirisup, S., Karniadakis, G.E.: A spectral viscosity method for correcting the long-term behavior of POD models. J. Comput. Phys. **194**(1), 92–116 (2004)
86. Sirovich, L.: Turbulence and the dynamics of coherent structures. I-Coherent structures. II-symmetries and transformations. III-dynamics and scaling. Q. Appl. Math. **45**, 561–571 (1987)
87. Taira, K., Brunton, S.L., Dawson, S., Rowley, C.W., Colonius, T., McKeon, B.J., Schmidt, O.T., Gordeyev, S., Theofilis, V., Ukeiley, L.S.: Modal analysis of fluid flows: an overview. AIAA J. **55**(12), 4013–4041 (2017)
88. Ullmann, S., Lang, J.: A POD-Galerkin reduced model with updated coefficients for Smagorinsky LES. In: V European Conference on Computational Fluid Dynamics, ECCOMAS CFD, p. 2010 (2010)
89. Wang, Z.: Reduced-order modeling of complex engineering and geophysical flows: analysis and computations. Ph.D. thesis, Virginia Tech (2012)
90. Wang, Z., Akhtar, I., Borggaard, J., Iliescu, T.: Two-level discretizations of nonlinear closure models for proper orthogonal decomposition. J. Comput. Phys. **230**(1), 126–146 (2011)
91. Wang, Z., Akhtar, I., Borggaard, J., Iliescu, T.: Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. Comput. Methods Appl. Mech. Eng. **237**, 10–26 (2012)
92. Weller, J., Lombardi, E., Bergmann, M., Iollo, A.: Numerical methods for low-order modeling of fluid flows based on POD. Int. J. Numer. Methods Fluids **63**(2), 249–268 (2010)
93. Wells, D., Wang, Z., Xie, X., Iliescu, T.: An evolve-then-filter regularized reduced order model for convection-dominated flows. Int. J. Numer. Methods Fluids **84**, 598–615 (2017)
94. Widrow, B., Rumelhart, D.E., Lehr, M.A.: Neural networks: applications in industry, business and science. Commun. ACM **37**(3), 93–106 (1994)
95. Xie, X., Mohebujjaman, M., Rebholz, L., Iliescu, T.: Data-driven filtered reduced order modeling of fluid flows. arXiv:1709.04362 (2017)
96. Xie, X., Wells, D., Wang, Z., Iliescu, T.: Approximate deconvolution reduced order modeling. Comput. Methods Appl. Mech. Eng. **313**, 512–534 (2017)
97. Zhang, G., Patuwo, B.E., Hu, M.Y.: Forecasting with artificial neural networks: the state of the art. Int. J. Forecast. **14**(1), 35–62 (1998)