Int. J. Comput. Appl. Tech., Vol. X, No. Y, 2020

Reduced-order modelling of parameterised incompressible and compressible unsteady flow problems using deep neural networks

Oliviu Şugar-Gabor

Department of Engineering
School of Science, Engineering and Environment
University of Salford
M5 4WT, Salford, United Kingdom
Email: o.sugar-gabor@salford.ac.uk

Abstract: A non-intrusive reduced-order model for nonlinear parametric flow problems is developed. It is based on extracting a reduced-order basis from full-order snapshots via proper orthogonal decomposition and using both deep and shallow neural network architectures to learn the reduced-order coefficients variation in time and over the parameter space. Even though the focus of the paper lies in developing a reduced-order methodology for approximating fluid flow problems, the methodology is generic and can be used for the order reduction of arbitrary time-dependent parametric systems. Since it is non-intrusive, it is independent of the full-order computational method and can be used together with black-box commercial solvers. An adaptive sampling strategy is proposed to increase the quality of the neural network predictions while minimising the required number of parameter samples. Numerical studies are presented for two canonical test cases, namely unsteady incompressible laminar flow around a circular cylinder and transonic inviscid flow around a pitching NACA 0012 aerofoil. Results show that the proposed methodology can be used as a predictive tool for unsteady parameter-dependent flow problems.

Keywords: Non-intrusive parameterised reduced-order model; proper orthogonal decomposition; artificial neural networks; incompressible and compressible flow model order reduction.

Reference to this paper should be made as follows: Author(s) (2020) 'Reduced-order modelling of parameterised incompressible and compressible unsteady flow problems using deep neural networks', *Int. J. Comput. Appl. Tech.*, Vol. X, No. Y, pp. 000–000.

Biographical notes: Oliviu Şugar-Gabor received his PhD in Aerospace Engineering in 2015, from the University of Quebec, Montreal, Canada. He is currently working as Lecturer (Assistant Professor) at the university of Salford. His research interests include computational fluid dynamics, fast aerodynamic tools for aircraft design, non-intrusive reduced order models and aerodynamic shape optimisation techniques.

1. Introduction

Large-scale, high-fidelity numerical simulations are commonly used across a wide array of industrial applications. Increasingly, parametric studies and design optimisation studies are also being done using high-fidelity models, even if they are costly. For instance, aerodynamic shape optimisation based on the Reynolds-Averaged Navier-Stokes (RANS) equations is now common practice. Even with the constant increase in computer power, these models are still unfeasible for large-scale parametric studies. Surrogate models aimed at finding inexpensive to run approximations of a system's input-output relation can partly overcome this challenge (Guenot et al., 2013). Among them, reduced order models (ROMs) occupy a central place by generating a low-order representation of the system defined on a basis which optimally spans the space of the system's variables. ROMs have been successfully applied to a variety of problems requiring large-scale numerical simulations, including weather prediction (Fang et al., 2014), modelling of particle physics (Buchan et al., 2015), data assimilation (Cao et al., 2007), flow through porous media (Alotaibi et al., 2015), molecular dynamics (Hoang et al., 2016), or the study of advection-diffusion-reaction phenomena (Giere et al., 2015).

A popular technique for determining such a basis is Proper Orthogonal Decomposition (POD), originally introduced over a century ago (Pearson, 1901). POD uses a set of system outputs (commonly referred to as snapshots) to provide a set of orthonormal vectors which form the required basis. The solution coordinates in the POD basis are usually known as the POD coefficients. Popular techniques to determine the coefficients are Petrov-Galerkin projection, least-squares residual (or another error) minimisation or data-fitting techniques (Breitkopf and Coelho, 2010).

In the Petrov-Galerkin projection approach, the original equations describing the system are projected onto the basis in order to determine the reduced-order system, which can be solved for the coefficients. This approach is code intrusive (requires extensive source code modifications) and cannot be used with black-box solvers. Intrusive ROMs have been successfully applied to incompressible turbulent flows (Carlberg and Farhat, 2011), predicting the gust response of aerofoils (Zhou et al., 2017) or the design of control laws for flexible aircraft (Da Ronch et al., 2012) As highlighted by several authors, intrusive ROMs are generally tailored towards a very specific application, can suffer from instability and nonlinear efficiency issues, or may be very difficult to implement in the first place for legacy codes (see for example Amsallem and Farhat, 2012, Wang et al., 2019, Xiao et al., 2014, Schlegel and Noack, 2015, and related works).

An alternative non-intrusive approach is to project only the snapshots onto the basis and determine a finite set of POD coefficients corresponding to those snapshots, and then use other surrogate techniques such as Kriging, artificial neural networks (ANNs) or radial basis functions (RBFs) as closures to model the full POD coefficient space based on the available set. This approach has become increasingly popular over the last few years, as seen in the works of Noori et al. (2011), Xiao et al. (2015, 2017, 2019), Audouze et al. (2009, 2013), Amsallem et al. (2015) or Wang et al. (2019).

Regardless of whether the ROM is intrusive or non-intrusive, most approaches use an offline-online strategy. The offline stage refers to collecting the high-order model snapshots and generating the basis (plus any other fixed quantities involved in that specific ROM approach), while the online stage refers to using the ROM to get approximations for the system solution for varying configurations of interest.

Alternatively, machine learning techniques have the potential to generate more accurate reduced-order models, and recent research has been conducted to investigate deep neural network learning architectures in the context of fluid flow applications. A good review of the use of ML in model order reduction, together with representative examples from aerodynamics or structural mechanics can be found in (Swischuk et al., 2019). A supervised machine learning approach has been developed to predict regions in the parameter space where a predefined reduced order basis can achieve a target accuracy (Moosavi et al., 2015). A deep recurrent neural network has been introduced as an efficient model order reduction technique for nonlinear dynamical systems (Kani and Elsheikh, 2017). Other deep neural networks, such as long short-term memory (LSTM) architectures have been utilised to complement an imperfect reduced-order model with live data streams (Wan et al., 2018). Deep neural networks have also been used in conjunction with POD-based reduced order modelling for unsteady parametric fluid flow models but restricted to the viscous Burgers equations and having the Reynolds number as the only parameter, with a limited number of samples (San et al., 2019). Deep ANNs have also been shown to allow the relatively accurate prediction of the dynamics of chaotic systems (Vaidyanathan et al., 2019), showing good potential for application to complex, turbulent fluid flows.

A very recent application of deep neural networks is presented in Renganathan et al. (2020), where a parametric reduced-order model is generated for the inviscid steady-state transonic flow over a parameterised RAE 2822 aerofoil, the results showing that the neural network based model achieved accuracy comparable with intrusive ROM techniques. A non-intrusive ROM strategy based on POD, physics-informed neural networks (PINNs) and physics-reinforced neural networks (PRNNs) was proposed in Chen et al. (2020). Results shown better ROM accuracy compared to other ANN-based approaches, but the method was tested only for a low number of parameters as a single ANN is used to capture both the unsteady and the parameter space behaviour. Another ANN-ROM approach was developed in Regazzoni et al. (2019), formulated as a maximum-likelihood problem, but without any parameterisation. In addition to their potential use in obtaining more accurate non-intrusive reduced-order models, deep neural networks have provided alternatives to classical RANS-based turbulence models (Ling et al., 2016), (Maulik and San, 2017) or even to the full Navier-Stokes equations (Gautier et al., 2015).

It is seen that the focus in most of these works has been either to capture the nonlinear parameter space variation of a steady parameter-dependent system, or to accurately capture the unsteady behaviour of a nonlinear system. Indeed, the most common usage for ANNs in ROM frameworks are related to time evolution or data compression, examples of the latter being found in Kim et al. (2019), Thuerey et al. (2020) or Yan et al. (2019), fewer works (such as Renganathan et al. (2020)) focusing on the parametric space approximation. Work on ANN-ROMs for time-dependent phenomena involving arbitrarily large parameter spaces, as are common for many engineering design problems, is rarely covered in literature.

To the authors best knowledge, the work presented here represents the first utilization of an ANN as a closure model in the POD-ROM modelling of parametric and time-dependent CFD problems in which the size and complexity of the parametric space can be arbitrarily high. The novelty and flexibility of the proposed approach consists of an efficient decoupling of the time-domain dynamics approximation from the parameter space approximation. The most adequate ANN architecture can thus be selected for each approximation, based on the available number of POD modes, parameter samples, computational time constraints, etc. The full model is built by leveraging the advantages of both deep and shallow neural network architectures in learning highly-nonlinear embeddings both in time and across the parameter space. A novel adaptive sampling strategy is also proposed for minimising the number of required samples in the parameter space while ensuring the most optimal training set for the shallow ANN. Section 2 of the paper details the proposed ROM methodology, while section 3 presents the numerical applications attempted.

2. Methodology

2.1 Proper Orthogonal Decomposition

Let the full-order time-dependent parametric model be represented by:

$$\boldsymbol{R}(\boldsymbol{u}(\boldsymbol{x},t,\boldsymbol{\mu}),\boldsymbol{x},t,\boldsymbol{\mu}) = \boldsymbol{0} \tag{1}$$

Here, $\boldsymbol{R}: \mathbb{R}^N \times \mathbb{R}^P \times [0,\infty) \rightarrow \mathbb{R}^N$ with N typically being very large, $\boldsymbol{u}: \Omega \subset \mathbb{R}^N \times \mathbb{R}^P \times [0,\infty) \rightarrow \mathbb{R}^N$ are the system variables defined on a subspace Ω of $\mathbb{R}^N$, $\boldsymbol{x} \in \mathbb{R}^N$ are spatial coordinates, t is the time, defined in the semi-infinite interval $[0,\infty)$ and $\boldsymbol{\mu}: \mathrm{D} \subset \mathbb{R}^P \rightarrow \mathbb{R}^P$ are the problem parameters defined on a subspace D of $\mathbb{R}^P$.

It must be noted that (1) represents the discretized form of a single equation, for a single variable, but written for all grid points (finite volumes or finite elements, depending on the mathematical framework upon which the full-order solver whose solution will be approximated by the ROM is build). If the parametric model includes several equations for several variables, then the procedure described below is applied independently to each variable.

A set of M points (or samples) in the parametric space $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_M,\}$, with M typically being much smaller than N, is initially chosen. The full-order model $\boldsymbol{R}$ from equation (1) is then solved (marched) in time for each parameter sample $\boldsymbol{\mu}_p$. The generation of the POD basis uses the method of snapshots introduced in (Sirovich, 1987). During the time-marching of the high-order model, at a set of MT points (or samples) in time $\{t_1, t_2, \dots, t_{MT},\}$, snapshots of the full solution are captured and arranged in the $N \times MT$ snapshots matrix:

$$\boldsymbol{S}(\boldsymbol{\mu}_p) = \begin{bmatrix} \boldsymbol{u}(\boldsymbol{x},t_1,\boldsymbol{\mu}_p), \boldsymbol{u}(\boldsymbol{x},t_2,\boldsymbol{\mu}_p), \dots, \\ \boldsymbol{u}(\boldsymbol{x},t_{MT},\boldsymbol{\mu}_p) \end{bmatrix} \tag{2}$$

The deviation matrix $\boldsymbol{D}(\boldsymbol{\mu}_p) = [\boldsymbol{u}(\boldsymbol{x},t_1,\boldsymbol{\mu}_p) - \overline{\boldsymbol{u}}, \boldsymbol{u}(\boldsymbol{x},t_2,\boldsymbol{\mu}_p) - \overline{\boldsymbol{u}}, \dots, \boldsymbol{u}(\boldsymbol{x},t_{MT},\boldsymbol{\mu}_p) - \overline{\boldsymbol{u}}]$ is then built, where $\overline{\boldsymbol{u}}$ is the snapshots mean vector, whose components are evaluated as $\bar{u}_i = \frac{1}{MT}\sum_{j=1}^{MT} u_i(t_j,\boldsymbol{\mu}_p), i = 1,2,\dots,N$. A singular value decomposition (SVD) of $\boldsymbol{D}$ is computed, $\boldsymbol{D} = \boldsymbol{U\Sigma V}^T$, where $\boldsymbol{U} \in \mathbb{R}^{N\times N}$, $\boldsymbol{V} \in \mathbb{R}^{MT\times MT}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{N\times MT}$ is a diagonal matrix containing the ordered singular values $\sigma_i \in \mathbb{R}$, $\sigma_1 > \sigma_2 > \dots > \sigma_{MT}$. The POD basis vectors (or modes) $\varphi_i(\boldsymbol{x},\boldsymbol{\mu}_p) \in \mathbb{R}^N$ are obtained by extracting the first K column vectors of $\boldsymbol{U}$. The problem solution can then be approximated as:

$$\boldsymbol{u}(\boldsymbol{x},t,\boldsymbol{\mu}_p) \cong \overline{\boldsymbol{u}}(\boldsymbol{x},\boldsymbol{\mu}_p) + \sum_{i=1}^{K} \alpha_i(t,\boldsymbol{\mu}_p)\varphi_i(\boldsymbol{x},\boldsymbol{\mu}_p) \tag{3}$$

Here, $\alpha_i(t,\boldsymbol{\mu}_p)$ are the POD coefficients. In a non-intrusive approach such as used in this paper, equation (3) is considered for each snapshot in turn, and the POD coefficients are determined by a Petrov-Galerkin projection of the snapshots onto the POD basis vectors:

$$\alpha_i(t_j,\boldsymbol{\mu}_p) = \left(\boldsymbol{u}(\boldsymbol{x},t_j,\boldsymbol{\mu}_p) - \overline{\boldsymbol{u}}(\boldsymbol{x},\boldsymbol{\mu}_p)\right)^T \varphi_i(\boldsymbol{x},\boldsymbol{\mu}_p), \quad i = 1,2,\dots,K, j = 1,2,\dots MT \tag{4}$$

For many problems of interest, the most important part of the energy distribution is concentrated in just the first few modes. The truncation of the basis to K modes is done by selecting the smallest possible $K \ll MT$ such that a desired level of energy capture $\varepsilon \in [0,1]$ is achieved:

$$\frac{\sum_{i=1}^{K}\sigma_i^2}{\sum_{i=1}^{MT}\sigma_i^2} \geq \varepsilon \tag{5}$$

In order to obtain a POD approximation of the solution $\boldsymbol{u}(\boldsymbol{x},t,\boldsymbol{\mu})$ at an arbitrary parameter value not included in the sampling $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_M,\}$ and at an arbitrary time not coinciding with any of the snapshot times, the variation of the POD coefficients over both the time domain and the entire parametric space is captured using artificial neural networks (ANNs).

2.2 Brief overview of Artificial Neural Networks

An artificial neural network, often referred to as a neural network, is a computational model able to learn from a provided data set (for a thorough overview of ANNs see for example Haykin (2004)).

Assume an arbitrary neuron j in the network receives M input signals $\{x_1, x_2, \ldots, x_M\}$ (either from M sending neurons or as the network input data) and produces a scalar output signal y_j (which can in turn be sent to N receiving neurons or represent the network output data). The propagation function of neuron *j* converts the vector input into a scalar input, often referred to as the net input. The most common choice is a weighted sum taking the form:

$$u_j = f_{propagation}(x_1, x_2, \ldots, x_M) = \sum_{k=1}^{M} w_{k,j} x_K \tag{6}$$

Here, u_j is the net input of neuron j and $w_{k,j}$ are the set of M weights for each of the neuron's vector input components. The activation function quantifies to what extent a given neuron is active (generating an output value). The activation function combines the net input with some threshold $t_j \in \mathbb{R}$ and is responsible for creating the activation state of the neuron:

$$a_j = f_{activation}(u_j, t_j) = f_{activation}\left(\sum_{k=1}^{M} w_{k,j} x_K \,, t_j\right) \tag{7}$$

Various choices of activation function exist in literature, among which the most common are so-called sigmoid functions (Haykin, 2004), the hyperbolic tangent or the rectified linear unit function being widely used examples:

$$a_j = \frac{e^z - e^{-z}}{e^z + e^{-z}} \qquad a_j = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{9}$$

The output function determines the neuron's scalar output y_j based on the activation state but is often taken as the identity function so that the output is directly given by a_j.

$$y_j = f_{output}(a_j) \tag{10}$$

Neurons are arranged in layers, with one input layer, K hidden layers and one output layer, with deep neural network typically having several hidden layers. The neurons in the input layer to not perform any computation tasks, having an identity activation function, while the output layer neurons typically use simpler, linear activation functions.

The training of the neural network is the iterative process through which the neuron weights are determined. This is achieved by learning from a provided training set. Let $f: \mathbb{R}^I \to \mathbb{R}^O$ be a nonlinear function which the neural network must approximate. Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{NTR}$, with $\boldsymbol{x}_i \in \mathbb{R}^I$ be a set of NTR training points, and $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{NTR}$, with $\boldsymbol{y}_i = f(\boldsymbol{x}_i) \in \mathbb{R}^I$ be the function values corresponding to these points. The set $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}, i = 1,2, \ldots, NTR$ represents the training set which must be supplied to the neural network. The goal is to approximate f up to a certain tolerance ε. The performance of the network is determined based on a performance function, typically the mean squared error (MSE):

$$Performance = \sum_{i=1}^{NTR} (\|\boldsymbol{y}_i - F(\boldsymbol{x}_i)\|)^2 \leq \varepsilon \tag{11}$$

Where $F(\boldsymbol{x}_i)$ is the neural network prediction at training point $\boldsymbol{x}_i$. The training iterations, usually called epochs, are aimed at calculating the optimal neuron weights such that (11) is satisfied and can be conducted using nonlinear regression algorithms such as the Levenberg-Marquardt algorithm (Marquardt, 1963) or stochastic gradient backpropagation approaches such as the ADAM algorithm (Kingma and Ba, 2014).

2.3 Generating the Artificial Neural Networks

Constructing the non-intrusive ROM requires a two-stage process. In the first stage, a deep ANN (D-ANN) is trained for each parameter value, D-ANN which captures the unsteady behaviour of model (1) for that specific sample in the parametric space:

$$\{t_1, t_2, \ldots, t_{MT},\} \xrightarrow{D-ANN(\boldsymbol{\mu}_p)} \{\boldsymbol{\alpha}(t_1, \boldsymbol{\mu}_p), \boldsymbol{\alpha}(t_2, \boldsymbol{\mu}_p), \ldots, \boldsymbol{\alpha}(t_{MT}, \boldsymbol{\mu}_p)\}, \quad \boldsymbol{\mu}_p = \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M \tag{12}$$

With the D-ANNs of (12) generated and trained, the solution at any moment in time can be calculated as:

$$\tilde{\boldsymbol{u}}(\boldsymbol{x}, t, \boldsymbol{\mu}_p) = \bar{\boldsymbol{u}}(\boldsymbol{x}, \boldsymbol{\mu}_p) + \sum_{i=1}^{K} \tilde{\alpha}_i(t, \boldsymbol{\mu}_p) \varphi_i(\boldsymbol{x}, \boldsymbol{\mu}_p), \quad \boldsymbol{\mu}_p = \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M \tag{13}$$

Where $\tilde{\alpha}_i(t, \boldsymbol{\mu}_p)$ are the POD coefficients approximated at time t by each of the D-ANNs trained in (12) and are given by $\tilde{\boldsymbol{\alpha}}(t, \boldsymbol{\mu}_p) = D - ANN(\boldsymbol{\mu}_p)(t)$. The second stage of the process is centred around constructing the approximation in the parameter space. Let t_D be the desired time for which the solution of (1) must be determined. Using equation (13), the approximation at t_D of the high-order solution $\tilde{\boldsymbol{u}}(\boldsymbol{x}, t_D, \boldsymbol{\mu}_p), p = 1,2, \ldots, M$ is calculated at all available parameter samples. Next, an ANN is trained using the available set of $\tilde{\boldsymbol{u}}$ values to build the correspondence:

$$\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M,\} \xrightarrow{S-ANN} \left\{ \begin{matrix} \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_D, \boldsymbol{\mu}_1), \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_D, \boldsymbol{\mu}_2), \ldots, \\ \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_D, \boldsymbol{\mu}_M) \end{matrix} \right\} \qquad (14)$$

A shallow ANN (S-ANN) with one single hidden layer with a relatively low number of neurons and using Radial Basis Functions (RBFs) is used for this step. Finally, the solution of (1) at the desired time t_D and an arbitrary parameter value $\boldsymbol{\mu}_D$ is given by the S-ANN trained in (14) and can formally be expressed as:

$$\hat{\boldsymbol{u}}(\boldsymbol{x}, t_D, \boldsymbol{\mu}_D) = S - ANN(\boldsymbol{\mu}_D) \qquad (15)$$

2.4 An adaptive sampling strategy

Ensuring the quality of the ANN-ROM approximation is adequate requires a judicious choice of parameter samples. This is especially true in instances where the system response depends nonlinearly on the values of some or all of the parameters. An adaptive sampling strategy can be utilised to this effect.

Assume a small, initial set of parameter samples $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M,\}$ is chosen (selected a priori by methods such as Latin Hypercube Sampling), the POD basis $\varphi_i(\boldsymbol{\mu}_p), i = 1,2, \ldots, K$ and coefficients are determined and the D-ANNs which capture the unsteady behaviour of the system at each of these parameter samples are built as described earlier, namely $\{t_1, t_2, \ldots, t_{MT},\} \xrightarrow{D-ANN(\boldsymbol{\mu}_p)}$

$$\{\boldsymbol{\alpha}(t_1, \boldsymbol{\mu}_p), \boldsymbol{\alpha}(t_2, \boldsymbol{\mu}_p), \ldots, \boldsymbol{\alpha}(t_{MT}, \boldsymbol{\mu}_p)\},$$

$\boldsymbol{\mu}_p = \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M$. Next, a small number NT of representative instances in time is chosen, and the set of S-ANNs is built as shown in (14) for each of the NT instances:

$$\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M,\} \xrightarrow{S-ANN_i} \{\tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_1), \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_2), \ldots, \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_M)\}, \quad i = 1,2, \ldots, NT \qquad (16)$$

The relative influence of each parameter sample on the overall quality of the approximation is defined as:

$$Infl_{Coeff}^{Rel}(\boldsymbol{\mu}_j) = \frac{\sum_{i=1}^{NT} \left\| \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_j) - \tilde{\boldsymbol{u}}^{-j}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_j) \right\|}{\sum_{k=1}^{M} \sum_{i=1}^{NT} \left\| \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_j) - \tilde{\boldsymbol{u}}^{-j}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_j) \right\|}, \quad j = 1,2, \ldots, M \qquad (17)$$

Here, $\tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_j) = S - ANN_i(\boldsymbol{\mu}_j)$ is the approximation of the solution at time instance t_i and parameter sample $\boldsymbol{\mu}_j$ using the S-ANNs constructed in (16), while $\tilde{\boldsymbol{u}}^{-j}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_j) = S - ANN_i^{-j}(\boldsymbol{\mu}_j)$ represents the same approximation, but as obtained with S-ANNs constructed by leaving out the j-th parameter sample from the training set:

$$\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_{j-1}, \boldsymbol{\mu}_{j+1}, \ldots, \boldsymbol{\mu}_M,\} \xrightarrow{S-ANN_i^{-j}} \left\{ \begin{matrix} \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_1), \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_2), \ldots, \\ \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_{j-1}), \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_{j+1}), \ldots, \\ \tilde{\boldsymbol{u}}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_M) \end{matrix} \right\}, \quad i = 1,2, \ldots, NT \qquad (18)$$

The quantity $Infl_{Coeff}^{Rel}(\boldsymbol{\mu}_j)$ is a measure of the total sensitivity of the S-ANN approximations with respect to the j-th parameter sample, and is calculated based on the well-known Leave One Out (LOO) verification strategy.

When selecting a new parameter sampling point, it is important to simultaneously improve the quality of the ANN approximations for all variables. To achieve this outcome, treating the system as a multi-response system and adopting a suitable Lipschitz constant as an indicator of the local response complexity (Lovison and Rigoni, 2011) are used.

In the study of partial differential equations, Lipschitz constants are employed for bounding the nonlinear character, and therefore the complexity, of the functions involved:

$$L_{f,D} = \sup_{x_1, x_2 \in D} \frac{|f(x_1) - f(x_2)|}{|x_1 - x_2|} \qquad (19)$$

After equation (17) is evaluated for all parameters in the initial set $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M,\}$ and for all variables (for example, $p - Infl_{Coeff}^{Rel}(\boldsymbol{\mu}_j)$ for pressure, $u - Infl_{Coeff}^{Rel}(\boldsymbol{\mu}_j)$ for the u velocity components, etc.), the parametric space is heavily populated (again using a method such as Latin Hypercube Sampling) with a set of candidate sample points $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_Q,\}$, with $Q \gg M$. The potential of enrichment of each candidate sample is then evaluated as:

$$\begin{aligned} &Pot(\boldsymbol{v}_i) = max(T_1, T_2, T_3), \\ &T_1 = p - Infl_{Coeff}^{Rel}(\boldsymbol{\mu}_j) \cdot \frac{L_p(\boldsymbol{v}_i)}{\sum_{\boldsymbol{v}_l} L_p(\boldsymbol{v}_l)} \cdot d(\boldsymbol{v}_i, \boldsymbol{\mu}_j), \\ &T_2 = u - Infl_{Coeff}^{Rel}(\boldsymbol{\mu}_j) \cdot \frac{L_u(\boldsymbol{v}_i)}{\sum_{\boldsymbol{v}_l} L_u(\boldsymbol{v}_l)} \cdot d(\boldsymbol{v}_i, \boldsymbol{\mu}_j), \\ &T_3 = v - Infl_{Coeff}^{Rel}(\boldsymbol{\mu}_j) \cdot \frac{L_v(\boldsymbol{v}_i)}{\sum_{\boldsymbol{v}_l} L_v(\boldsymbol{v}_l)} \cdot d(\boldsymbol{v}_i, \boldsymbol{\mu}_j), \\ &i = 1,2, \ldots, Q, j = argmin_k d(\boldsymbol{v}_i, \boldsymbol{\mu}_k) \end{aligned} \qquad (20)$$

Here, $d(\boldsymbol{v}_i, \boldsymbol{\mu}_j)$ is the Euclidean distance between two sample points and $L_p(\boldsymbol{v}_i)$, $L_u(\boldsymbol{v}_i)$ and $L_v(\boldsymbol{v}_i)$ are the Lipschitz constants calculated as $L_p(\boldsymbol{v}_i) = \sup_{\boldsymbol{\mu}_k \in \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M,\}} \sum_{i=1}^{NT} \left\| \begin{matrix} \tilde{p}(\boldsymbol{x}, t_i, \boldsymbol{v}_i) - \\ \tilde{p}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_k) \end{matrix} \right\| / \|\boldsymbol{v}_i - \boldsymbol{\mu}_k\|$.

Through (20), a balance is achieved between three distinct effects: a) improvement of the local quality of the S-ANN model by focusing on those samples having the highest relative influence; b) focusing on the regions of stronger nonlinearity as defined by the Lipschitz constants; c) parameter space exploration through the Euclidean distance, resulting in a decrease in the potential of enrichment if the candidate sample is too close to an already existing sample.

Finally, the candidate sample having the highest potential $\boldsymbol{\mu}_{new} = argmax_k Pot(\boldsymbol{v}_k)$ is added to the set, which becomes $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_M, \boldsymbol{\mu}_{M+1} = \boldsymbol{\mu}_{new}\}, M \leftarrow M + 1$ and the procedure continues until the desired number of samples have been added to the set.

2.5 The ANN-ROM algorithm

The offline procedure of constructing the ANN-ROM can be summarised as following:

(a) Select initial set of parameter samples $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_M,\}$ using Latin Hypercube Sampling.

(b) For each parameter sample in the set:

(b.1) run the full-order CFD solver and capture the sequence of snapshots.

(b.2) for each variable in the snapshots:

(b.2.1) obtain a set of POD basis function by using SVD of the snapshot matrix in equation (2).

(b.2.2) determine the POD coefficients by projecting the snapshots on the POD basis as shown in equation (4).

(b.2.3) train the deep ANNs (12) which capture the unsteady behaviour of model.

(c) Use the adaptive sampling procedure to add new parameter samples until the desired number has been added to the set:

(c.1) for each sample already in the set:

(c.1.1) train the shallow ANNs as shown in (16) and (18).

(c.1.2.) calculate the relative influence coefficient given by equation (17).

(c.2) populate the parameter space with the candidate samples $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_Q,\}$ using Latin Hypercube Sampling. For each candidate sample:

(c.2.1) calculate the Lipschitz constants with respect to all samples already in the set $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_M,\}$ using the S-ANNs trained in step (c.1.1).

(c.2.2) calculate the enrichment potential using equation (20).

(c.3) add the candidate sample having the highest potential to the set and:

(c.3.1) run the full-order CFD solver and capture the sequence of snapshots for the new sample.

(c.3.2) obtain the POD basis, POD coefficients and deep ANN as in steps (b.1) – (b.2).

The online procedure of using the ANN-ROM can be summarised as following:

(a) Select an arbitrary parameter value $\boldsymbol{\mu}_D$ and a series of time instances $\{t_1, t_2, \dots, t_p\}$ at which the solution is to be approximated.

(b) for each time instance in the series

(b.1) calculate the POD coefficients corresponding to that time instance using the trained deep ANNs for all parameter values in the set and for all variables of interest (p, u, etc).

$$\{\tilde{\boldsymbol{\alpha}}(t_i, \boldsymbol{\mu}_p), \tilde{\boldsymbol{\alpha}}(t_i, \boldsymbol{\mu}_p), \dots, \tilde{\boldsymbol{\alpha}}(t_i, \boldsymbol{\mu}_p)\}, \quad \boldsymbol{\mu}_p = \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_M \tag{21}$$

(b.2) Obtain the solutions corresponding to that time instance for all parameter values in the set:

$$\begin{gathered}\tilde{p}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_p) = \\ \bar{p}(\boldsymbol{x}, \boldsymbol{\mu}_p) + \sum_{i=1}^{K} \tilde{\alpha}_{p,i}(t_i, \boldsymbol{\mu}_p) \varphi_{p,i}(\boldsymbol{x}, \boldsymbol{\mu}_p), \\ \tilde{u}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_p) = \\ \bar{u}(\boldsymbol{x}, \boldsymbol{\mu}_p) + \sum_{i=1}^{K} \tilde{\alpha}_{u,i}(t_i, \boldsymbol{\mu}_p) \varphi_{u,i}(\boldsymbol{x}, \boldsymbol{\mu}_p), \\ \boldsymbol{\mu}_p = \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_M\end{gathered} \tag{22}$$

(b.3) For all variables of interest, train the shallow ANN as shown in (14).

(b.4) Use the shallow ANN to obtain the predictions for the arbitrary parameter value $\hat{p}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_D)$, $\hat{u}(\boldsymbol{x}, t_i, \boldsymbol{\mu}_D)$, etc.

2.6 Governing flow equations

The open-source SU2 solver (Economon et al., 2016) is used for the applications presented in the paper. The incompressible Navier-Stokes equations, governing the low-speed flow of a viscous fluid, written in differential form, are (for two-dimensional flows):

$$\begin{gathered}\nabla \cdot \boldsymbol{V} = 0 \\ \frac{\partial \boldsymbol{V}}{\partial t} + \nabla(\boldsymbol{V} \otimes \mathbf{V}) = -\frac{\nabla p}{\rho} + \upsilon \nabla^2 \boldsymbol{V}\end{gathered} \tag{23}$$

With:

$$\boldsymbol{V} = [u \quad v]^T$$

Where $\boldsymbol{V}$ is the velocity vector having components u and v (assuming a Cartesian reference system), ρ is the (constant) density, p is the pressure and υ is the kinematic viscosity.

The Euler equations, governing the compressible flow of an inviscid fluid, written in differential form, are (for two-dimensional flows):

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F} = 0 \tag{24}$$

With:

$$\boldsymbol{U} = [\rho \quad \rho u \quad \rho v \quad \rho E]^T$$
$$\boldsymbol{F} = \boldsymbol{F}_x \boldsymbol{i} + \boldsymbol{F}_y \boldsymbol{j}$$
$$\boldsymbol{F}_x = [\rho u \quad \rho uu + p \quad \rho uv \quad \rho uH]^T$$
$$\boldsymbol{F}_y = [\rho v \quad \rho vu \quad \rho vv + p \quad \rho vH]^T$$

Where $\boldsymbol{U}$ are the conservative variables, ρ is the density, u and v are the velocity components (assuming a Cartesian reference system), E is the internal energy, $\boldsymbol{F}$ is the flux vector having components $\boldsymbol{F}_x$ and $\boldsymbol{F}_y$, p is the pressure and H is the enthalpy.

3. Applications

The proposed ANN-ROM methodology is demonstrated on the prediction of two canonical test cases, which are unsteady incompressible laminar flow around a circular cylinder and the unsteady transonic flow around a sinusoidally pitching NACA 0012 aerofoil. It must be noted that the main focus of this work rests with verifying the capabilities of the ANN-ROM method against a variety of flow problems. As such, the test cases are selected to capture various flow phenomena of interest such as vortex shedding and shock waves.

3.1 Unsteady incompressible laminar flow around a circular cylinder

The cylinder has a diameter $D = 1m$ and is placed in a domain of length $L/D = 20$ and height $H/D = 10$. A fully structured grid of approximately 12000 cells is generated as shown in Figure 1. The six dots visible in the figure indicate the positions of monitoring points that will be used for a quantitative evaluation of the ROM accuracy, the exact coordinates of these points being provided at the relevant point in the paper.

The velocity at the inlet is set to a constant value $u_{inlet} = 1m/s$. An implicit second-order accurate

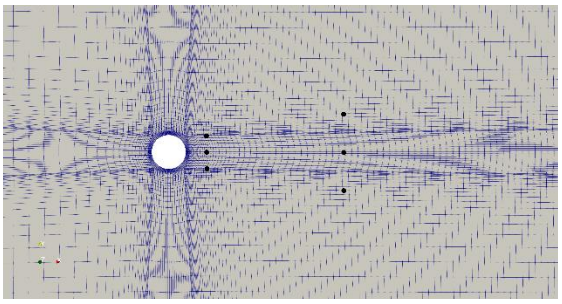


Figure 1. The circular cylinder structured grid

Flux Difference Splitting (FDS) scheme is used for the momentum equations.

Time marching is done using a second-order accurate dual time stepping method. At each physical time step, to accelerate dual time convergence to steady stage, a 3-level multigrid approach is used. The unsteady flow problem is run with a time step $\Delta t = 0.005s$ for a total physical time of $t_{max} = 50s$.

In order to verify that the selected grid density is sufficient, a grid convergence study is performed. A coarse grid of 4000 cells and a fine grid if 36000 cells are generated. To verify for grid convergence, the average value of the drag coefficient C_D^{Avg} is determined for the time interval between $t = 40s$ and $t_{max} = 50s$. The grid convergence study results for $u_{inlet} = 1m/s$ and a Reynolds number $Re = 30000$ are included in Table 1. It is seen that the C_D^{Avg} error between the 12000 cells medium grid and the Richardson extrapolation is 2.9%. As the primary focus of this work is to verify the capabilities of the ANN-ROM in approximating the full-order results, the RANS results obtained on the medium grid are considered to be sufficient for achieving this verification.

A time step size study is performed to select the required time step value. The study is performed on the medium grid of 12000 cells. Three values are chosen, $\Delta t = 0.0025s$, $\Delta t = 0.005s$ and $\Delta t = 0.01s$, with the results being summarised in Table 2.

Table 1. Grid convergence study for circular cylinder problem

Coarse Grid C_D^{Avg}	Medium Grid C_D^{Avg}	Fine Grid C_D^{Avg}	Order of Convergence	Richardson Extrapolation	Grid Convergence Index
0.688	0.715	0.727	1.476	0.736	1.017

Table 2. Time step study for circular cylinder problem

	$\Delta t = 0.0025s$	$\Delta t = 0.005s$	$\Delta t = 0.01s$
Strouhal Number	0.621	0.615	0.559

The Strouhal number $St = fD/u_{inlet}$ is calculated based on the vortex shedding frequency f for the time interval between $t = 40s$ and $t_{max} = 50s$. Reducing the time step size below $\Delta t = 0.005s$ has only a small effect on the obtained unsteady characteristics, and so the time step value is considered for the subsequent calculations.

Simulations are done at 3 values of the Reynolds number corresponding to the laminar flow regime, $Re = [25000, 30000, 35000]$, with Re representing the only parameter of this application. The adaptive sampling strategy is not used for this application. Snapshots of the pressure p and the two velocity components u and v are taken every 10 time steps, for a total of 1000×3 snapshot vectors for each parameter sample point.

Figure 2 shows the first 140 singular values of the u velocity snapshots matrix, nondimensionalised with respect to the largest singular value σ_1, as well as the energy content capture (calculated using (5)) as function of truncating the POD basis to various numbers K of modes. It was decided to proceed with $K = 50$ for generating the ROM.

The D-ANN which captures the unsteady behaviour of model has five hidden layers, with 8, 16, 32, 64 and 64 neurons, and one output layer. Neurons in

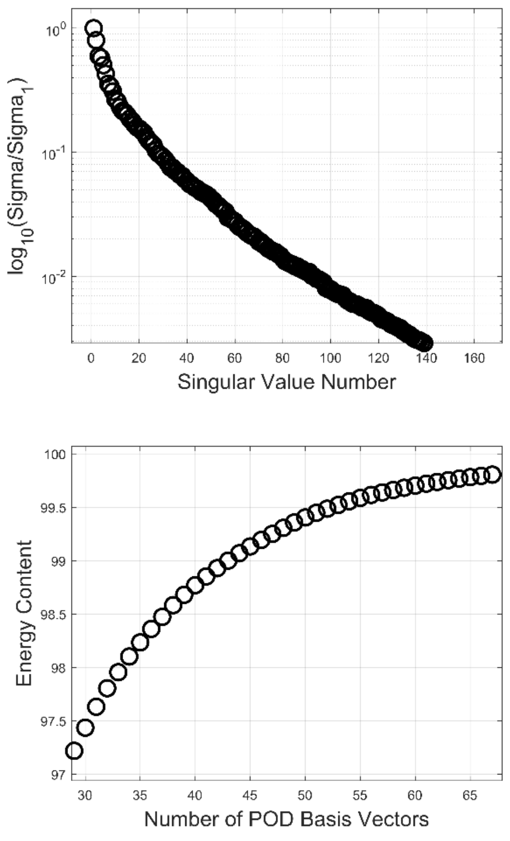


Figure 2. Singular values of u velocity snapshots matrix and the energy content capture as function of the retained number of POD modes

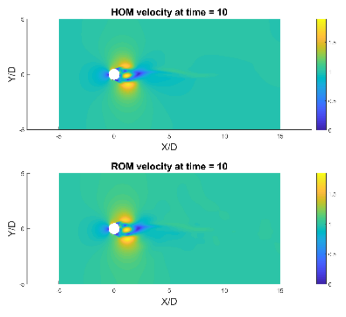
hidden layers 1, 3 and 5 use the hyperbolic tangent sigmoid activation function, hidden layers 2 and 4 use the rectified linear unit activation function while the output layer neurons use a linear activation function. Training is done using the ADAM algorithm for a maximum of 5000 epochs. From the 1000 snapshots in each training set, 70% are used for the actual training, and 15% each for validation and testing.

In order to test the capabilities of the ANN-ROM, a new Reynolds number value of $Re = 32000$ is chosen. Figure 3 shows contour plots of the velocity magnitude $V = \sqrt{u^2 + v^2}$ as obtained with both the full-order CFD solver and the ANN-ROM, at six selected instances in time, namely $t = [10, 18, 25, 33, 42, 49]$. It is seen that the ANN-ROM predicts the flow filed with generally good accuracy. The loss in stability of the wake and the shedding of the first vortex are accurately captured at $t = 10$ and $t = 18$. The downstream advection of the initially-shed vortices is well tracked by the ANN-ROM, however at $t = 33$, as some of the vortices are pushed above the longitudinal centreline of the flow filed due to the influence of the strong first vortex, some discrepancies can be observed between the CFD solution and the ANN-ROM. As the flow filed progresses towards the characteristic periodic alternating vortex shedding pattern visible especially at $t = 49$, the quality of the ANN-ROM prediction increases, with a good capture of shedding frequency and typical vortex size.

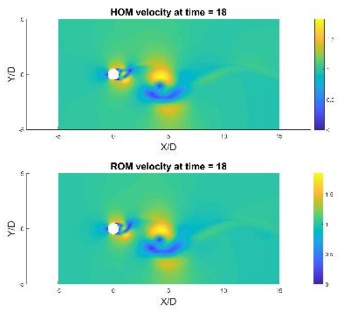
A quantitative overview of the ANN-ROM prediction is made by choosing a number of six points in the flow domain and plotting the time evolution of gauge pressure and velocity components at these locations. The chosen points have coordinates $(x, y) = (1, -0.49)$, (1,0), (1,0.49), $(5, -1)$, (5,0) and (5,1). Figure 4 presents the time evolution of the flow variables at these points. It is immediately observed that values predicted by the ANN-ROM approximate the values from the CFD well, but are somewhat nosy, with small instantaneous fluctuations appearing for all three variables, but especially visible for the pressure, and for the entire duration of the calculation. This behaviour is possibly due to the lack of sufficient training data, both for the unsteady D-ANN but especially for the parametric S-ANN. However, many realistic engineering scenarios operate within the constraints of reduced datasets, as the computational costs of running the unsteady full-order model for hundreds of parameter samples, or the storage requirements related to saving thousands of solutions for each unsteady run can be prohibitive.

The pressure coefficient distribution is shown in Figure 5 at the same six instances in time as those used for the velocity magnitude contour plots shown earlier. The quality of the ANN-ROM prediction is overall fair, showing low error in the region of attached flow and favourable pressure gradient on the front side of the cylinder, and higher error for the separated flow on the back of the cylinder, where the alternating vortex shedding initiates.

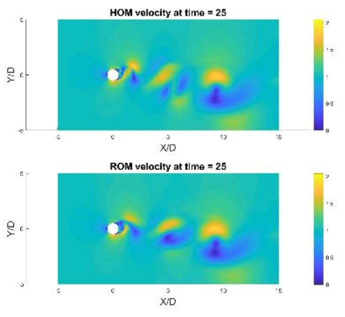
The variation in time of the cylinder's drag and lift coefficients are depicted in Figure 6. The ANN-ROM prediction of the lift coefficient is relatively good, although showing the same fluctuations as were seen for the flow variables at the monitoring points downstream of the cylinder. Again, this behaviour is possibly due to the lack of sufficient training data. The drag coefficient prediction is not accurate and would not represent an adequate substitute for the full-order simulation. It was decided to include these results nonetheless, as the laminar flow around a circular cylinder is widely used as test case for various ROM strategies, however quantitative results related to the aerodynamic coefficients or the surface pressure coefficient distribution are not often presented.



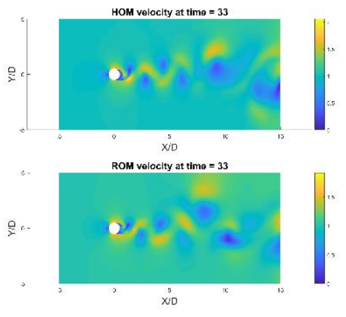
(a)



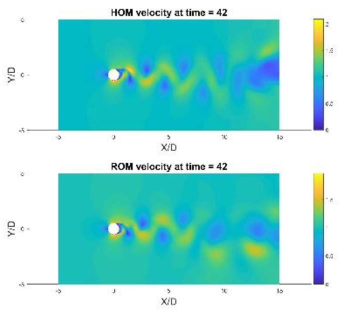
(b)



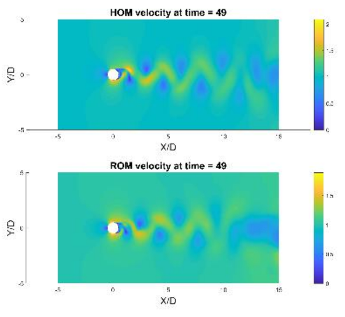
(c)



(d)

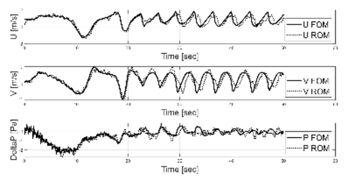


(e)

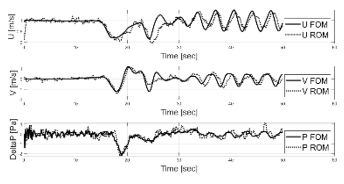


(f)

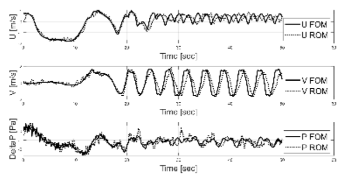
Figure 3. Plot of velocity magnitude for full-order CFD and ANN-ROM solutions for flow around a circular cylinder at a Reynolds number of 32000



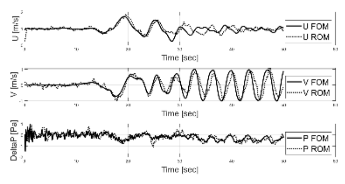
(a)



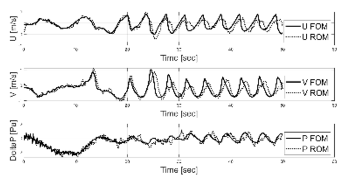
(b)



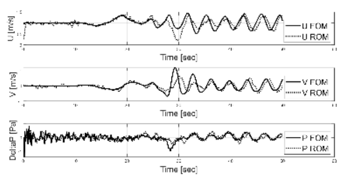
(c)



(d)



(e)



(f)

Figure 4. Time variation of velocity components and gauge pressure at points with (X, Y) coordinates of: (a) (1, -0.49), (b) (5, -1), (c) (1, 0), (d) (5, 0), (e) (1, 0.49), (f) (5, 1)

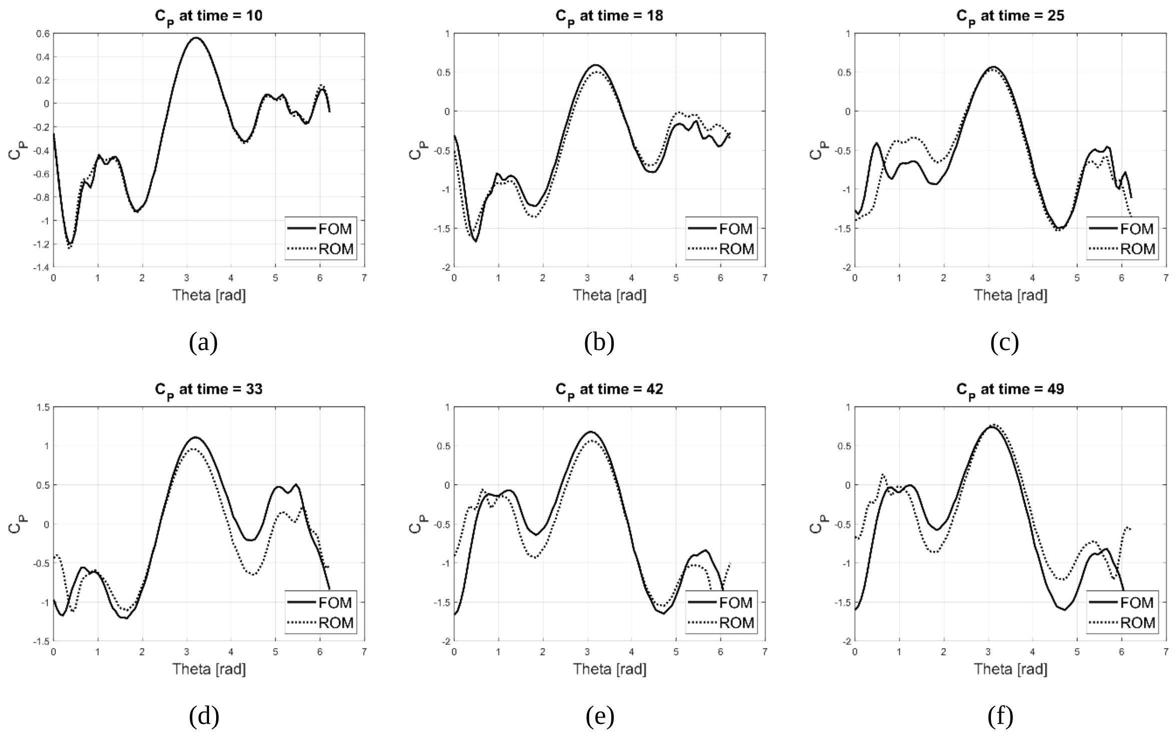


Figure 5. Pressure coefficient distribution for flow around a circular cylinder at a Reynolds number of 32000

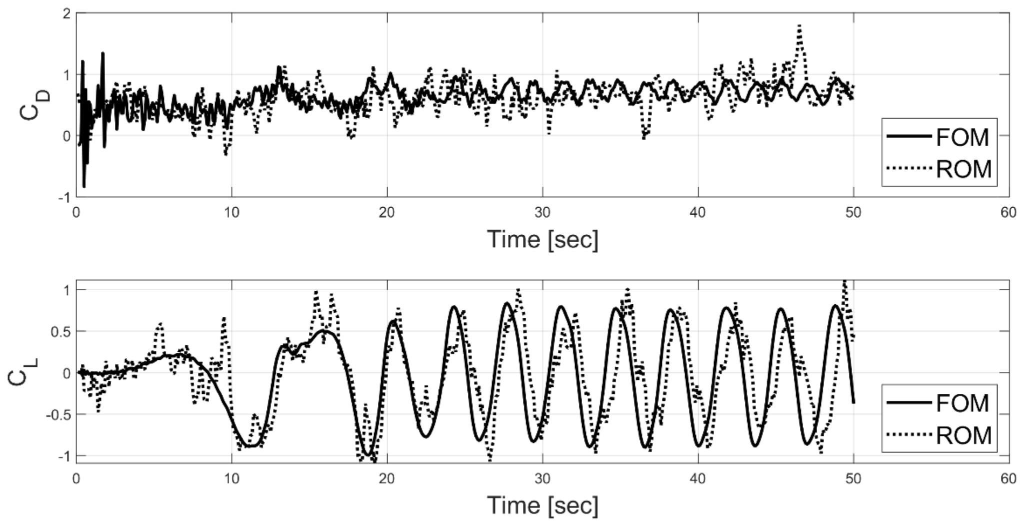


Figure 6. Time variation of cylinder drag and lift coefficients

3.2 Unsteady transonic flow around a sinusoidally pitching NACA 0012 aerofoil

The NACA 0012 aerofoil has a chord $c = 1$ and is placed in a relatively small domain of length $L/c = 8$ and height $H/c = 6$. A fully structured mesh of approximately 7,000 cells is generated and shown in Figure 7. The three dots visible in the figure indicate the positions of monitoring points that will be used for a quantitative evaluation of the ROM accuracy, the exact coordinates of these points being provided at the relevant point in the paper.

The inviscid compressible flow is run at Mach number values between $M = 0.80$ and $M = 0.85$, the Mach number representing the first parameter of this problem. The Euler equations are discretised using an implicit second-order accurate Jameson-Schmidt-Turkel (JST) scheme, while time marching is done using a second-order accurate dual time

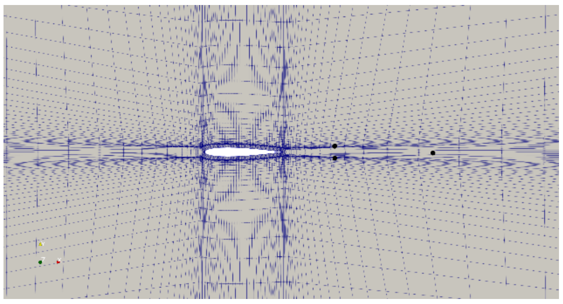


Figure 7. The structured grid around the NACA 0012 aerofoil

stepping method. At each physical time step, to accelerate dual time convergence to steady stage, a 3-level multigrid approach is used. The unsteady flow problem is run with a time step $\Delta t = 0.01s$ for a total physical time of $t_{max} = 1s$.

The farfield boundaries are positioned relatively close to the aerofoil. While this might affect the accuracy of the numerical flow solution, it will not affect the ANN-ROM generation and thus is considered sufficient for assessing the capabilities of the ANN-ROM. In order to verify that the selected grid density is sufficient, a grid convergence study is performed. A coarse grid of 3000 cells and a fine grid if 15000 cells are generated. The study is performed for steady-state flow conditions, at a Mach number of $M = 0.82$ and an aerofoil angle of attack of $\alpha = 3°$. The grid convergence results are included in Table 3. It is seen that for the medium grid of 7000 cells, the drag coefficient error compared to the Richardson extrapolation is 3.97%, while the lift coefficient error is 1.6%. For the inviscid flow case considered, the medium grid is considered for the rest of calculations required to generate the ANN-ROM.

Table 3. Grid convergence study for NACA0012 aerofoil in transonic flow problem

Coarse Grid C_D	Medium Grid C_D	Fine Grid C_D	Order of Convergence	Richardson Extrapolation	Grid Convergence Index
0.0752	0.0706	0.0689	2.472	0.0679	0.976
Coarse Grid C_L	Medium Grid C_L	Fine Grid C_L	Order of Convergence	Richardson Extrapolation	Grid Convergence Index
0.625	0.598	0.591	3.353	0.589	0.988

The singular values of the density snapshots matrix and the energy content capture (calculated using (5)) as function of truncating the POD basis to various numbers K of modes are depicted in Figure 8. This problem shows a much higher concentration of energy in the first few modes, as compared with the flow around the circular cylinder, with $K = 6$ modes being sufficient to obtain an energy content capture above 99.9%. It was decided to select $K = 8$ modes (energy content above 99.95%) for generating the ROM. Density is the variable for which the energy content capture grows slowest as function of K, this being the reason for selecting the K value based on density.

The D-ANN which captures the unsteady behaviour of model has five hidden layers, but the number of neurons per layer is much reduced compared to the previous application, having only 4, 8, 16, 16 and 8 neurons in the hidden layers. The activation functions, division of training data samples and the training algorithm remain the same as they were for the circular cylinder application. The S-ANN used for providing the approximations in the parameter space is now configured with 125 neurons in the hidden layer, a much higher number than before, but much more adequate for capturing the nonlinearities present.

Testing of the ANN-ROM accuracy is done at a Mach number of $M = 0.835$, a pitching amplitude $\hat{\alpha} = 4.5°$ and an angular frequency $\omega = 1.05 \times 2\pi\, rad/sec$, after checking that this point in the parameter space was not included in the ROM generation by the adaptive sampling procedure. Figure 9 shows contour plots of the Mach number at six selected instances in time, $t = [0.15, 0.30, 0.45, 0.60, 0.75, 0.90]$. There is a qualitative agreement between the ROM prediction and the full-order solution is terms of shock wave location, extend of supersonic flow over the aerofoil surface as well as the alternating appearance and subsidence of the supersonic flow over the upper and lower surfaces as the sinusoidal pitching motion progresses in time.

In the case of flow around aerodynamic bodies such as aerofoils, obtaining a pressure coefficient distribution which is as accurate as possible over both upper and lower surfaces is very important, regardless of the distribution being obtained by a full-order or a reduced-order model. Figure 10 depicts C_p versus chordwise location at the same six instances in time as used for the qualitative overview presented earlier.

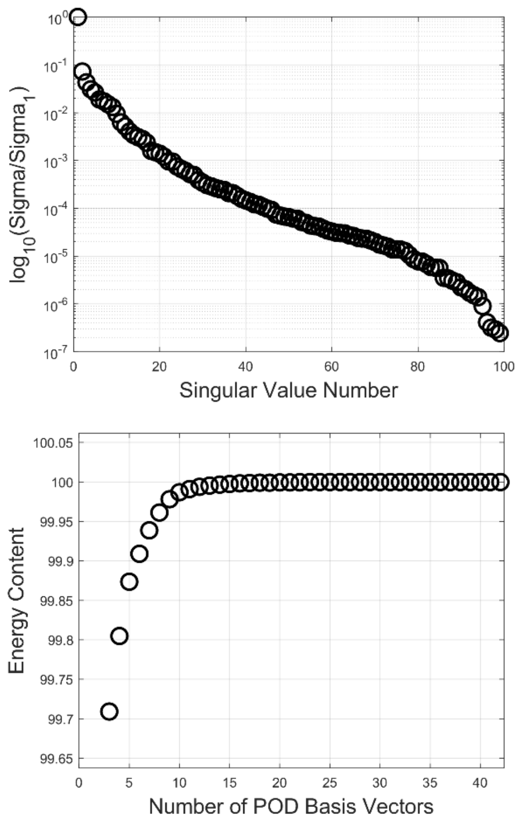


Figure 8. Singular values of density snapshots matrix and the energy content capture as function of the retained number of POD modes

The ANN-ROM prediction is very good, as the two curves are superimposed for most of the aerofoil chord. The pressure increase across the shock wave is not captured as sharply as in the full-order solution, a much smoother variation in the pressure coefficient being present both in front and after the shock. However, the chordwise location of the shock wave is well captured, as is the maximum gradient across the shock. This smoothing effect is due to the shallow RBF network approximation across the entire parametric space, being mostly eliminated when the comparison is made at one of the parameter samples chosen in the ANN-ROM generation phase.

The time variations of the force coefficients corresponding to the projection of the resultant aerodynamic force on the x-axis and z-axis are shown in Figure 11. The vertical force coefficient C_z is very well captured, while the ANN-ROM prediction of the horizontal force coefficient C_X has a maximum error of approximately 10% (at $t = 0.27sec$, the full-order value is 0.0538 while the reduced-order value is 0.0487). This is due to the smooth nature of the ANN-ROM pressure coefficient in front and after the shock wave.

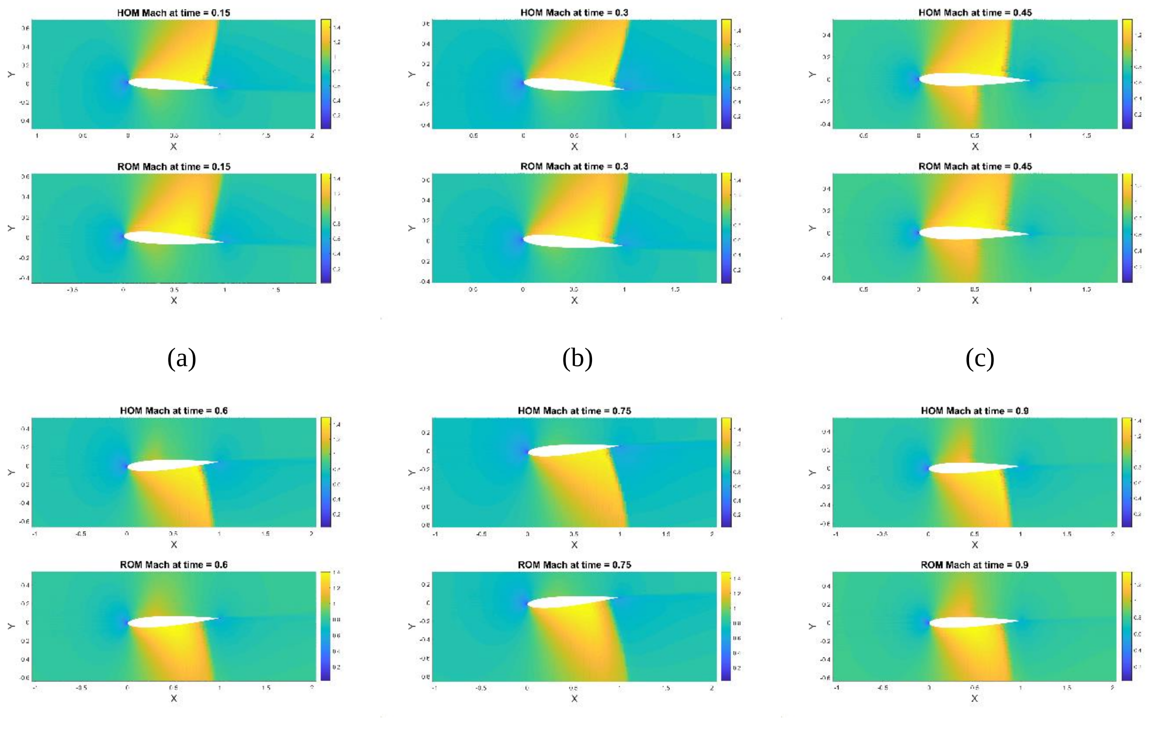
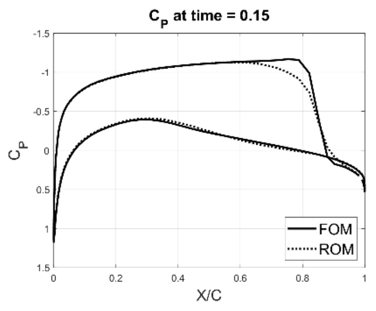
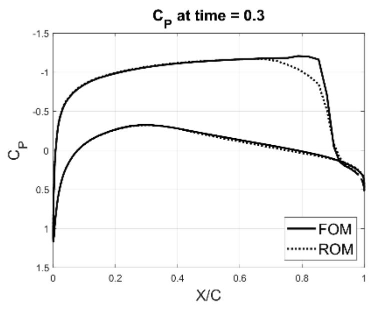


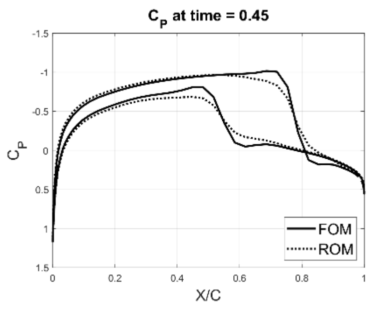
Figure 9. Plot of Mach number for full-order CFD and ANN-ROM solutions for the transonic flow around the pitching NACA 0012 aerofoil



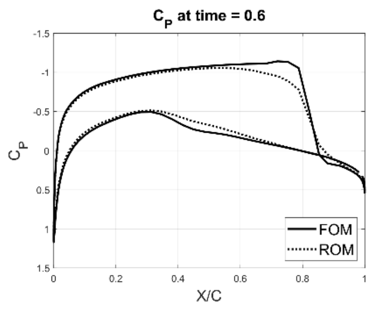
(a)



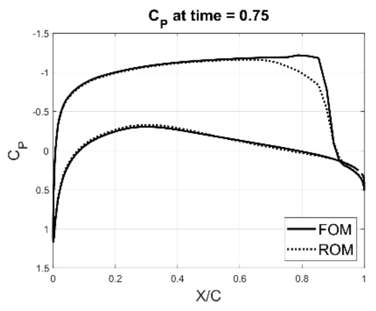
(b)



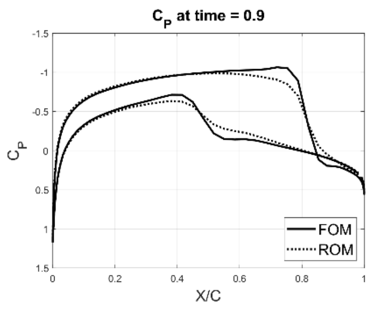
(c)



(d)



(e)



(f)

Figure 10. Pressure coefficient distribution over the pitching NACA 0012 aerofoil

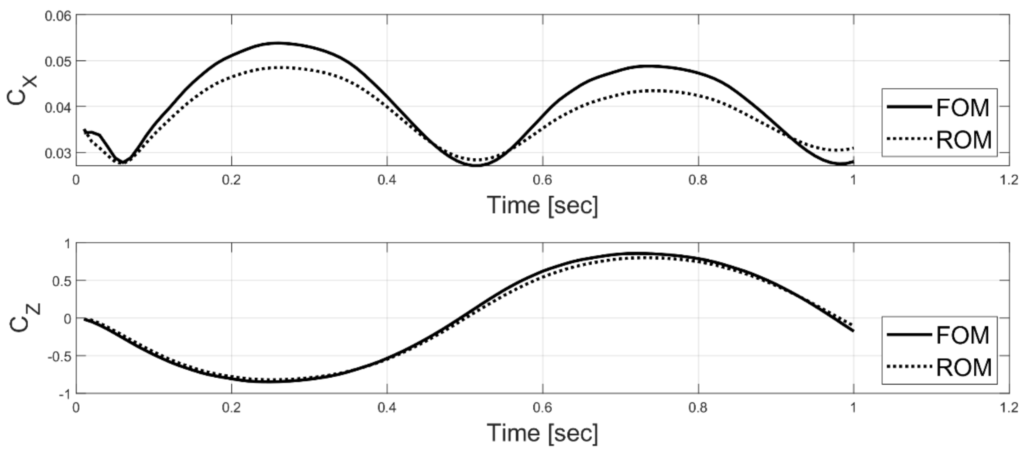


Figure 11. Time variation of aerodynamic force coefficients for the pitching NACA 0012 aerofoil

3.4 Computational costs

Table 4 summarises the computational cost incurred in the generation of the ROM. All CPU times are normalised by the CPU time required to run one unsteady CFD solution. The unsteady CFD solution times used for normalisation are different for each test case. It will be noted that all the numerical experiments shown in this paper were conducted on an Intel i5-9600K workstation with 16GB of RAM and an NVIDIA RTX2070 GPU.

By far, the largest amount of time of the offline phase of the algorithm is spent on running the full-order CFD solver and collecting the required snapshots. Each circular cylinder run took approximately 3 hours (including writing/reading data and saving all snapshots), while each run of the transonic NACA0012 analysis took approximately 15 minutes. Scaling the snapshots matrix and generating the POD basis is the least expensive step of the algorithm, amounting to only a small fraction of the CPU cost required to run one unsteady CFD solution. On the hardware used, this step took approximately $6 \times 10^{-2}s$ per parameter sample point and per flow variable, leading to a maximum time of around $30s$ for the second application, having 125 parameter sample points.

Table 4. Computational cost of ROM generation

	Circular Cylinder CPU Time	Transonic Inviscid NACA0012 CPU Time
Snapshots Collection	3	125
POD Basis	5.3×10^{-5}	3×10^{-2}
D-ANN Training	8.3×10^{-3}	5.55

Table 5. Computational cost of ROM execution

	Circular Cylinder CPU Time	Transonic Inviscid NACA0012 CPU Time
CFD Solution	9×10^{5}	4.4×10^{3}
ROM Solution (one time instance)	1	1

The second largest amount of time is spent on generating the D-ANNs. The CPU cost of this step is proportional to the number of networks to be trained and can be as costly as several CFD solutions. Leveraging the advantages of the massively-parallel architecture of the GPU, the average D-ANN training time was approximately $10s$ per network, leading to a maximum training time (again for the second test case) of 1.35 hours. During the adaptive sampling procedure, additional time is spend generating the S-ANNs required for the relative influence coefficients. The training time of the single layer RBF-based networks was very fast, typically around $5 \times 10^{-2}s$ per network for the 125 neuron architecture of the second test case.

The CPU costs required to run the ROM are presented in Table 5. CPU times are normalised by the CPU time required to run the ANN-ROM. The ROM execution time provided in the table is for predicting the solution at one time instance. For multiple time instances, the ROM CPU time in Table 5 is simply multiplied by the number of instances.

The CPU running time of the ROM is given by the number of parameter space samples. The CPU speed-up obtained for problems with a high number of samples and a low CFD solution time will be lower, as seen for second test case, but is still significant. For more complex CFD solutions, the CPU speed-up will be substantial, as observed for the first test case. The online computational costs are given by running the trained D-ANNs for all parameter samples, taking approximately $10^{-3}s$ per run, and training the S-ANN once for each time instance at which the solution is to be calculated. Training times depend on the total number of parameter samples and can take approximately $1s$ for the 125 neuron architecture.

It must be noted that the hyperparameters of the D-ANNs used in the current work were set based on a quick naïve study, and since the results obtained were satisfactory, a thorough search for optimal the network configuration and hyperparameter values was deemed unnecessary. Further improvement in D-ANN accuracy can be obtained by automating the search for the optimal number of hidden layers, neurons, batch size, learning rate, etc. This will significantly increase the D-ANN training time, but the procedure can be done for a single parameter sample point (a single network) and would be necessary for very complex unsteady flow fields.

4. Conclusions

In this work, a non-intrusive reduced-order model was proposed, based on proper orthogonal decomposition, and using both deep and shallow neural network architectures. The proposed approach consists of an efficient decoupling of the time-domain dynamics approximation, done with the deep network, from the parameter space approximation, done with the shallow network. The model developed is suitable for parametric unsteady flow problems. Numerical studies were conducted for incompressible laminar flow around a circular cylinder and inviscid transonic flow around a pitching NACA0012. Overall, the results indicate that the ANN-ROM model can achieve very good accuracy. The deep neural network can learn the unsteady variations of the POD coefficients to a good degree of accuracy even with as few as 100 training samples in the time domain, as indicated by the two test cases. It was observed that the shallow neural network is good at capturing the solution variations across the parameter space, but sufficient training samples must be provided to avoid random fluctuations in the predicted values, as seen for the circular cylinder test case. The pressure coefficient distribution is well approximated by the ANN-ROM in both compressible and incompressible flows.

Future directions in this work include the use of more complex deep neural networks to learn both unsteady and parameter space variations by a single network, although reconstructing the full solution would require a way to approximate the POD basis at arbitrary points. Use of a second deep neural network to provide the POD basis approximation is an option being considered. Other future developments include tests on problems with higher number of degrees of freedom and more complex flow fields, as well as incorporation into unsteady aerodynamic optimisation frameworks in the longer term.

References

Alotaibi, M, Calo, VM, Efendiev, Y, Galvis, J and Ghommem, M. (2015) Global local nonlinear model reduction for flows in heterogeneous porous media. *Computer Methods in Applied Mechanics and Engineering*, Vol. 292, pp. 122-137. Special Issue on Advances in Simulations of Subsurface Flow and Transport (Honouring Professor Mary F. Wheeler).

Amsallem, D and Farhat, C. (2012) Stabilization of projection-based reduced-order models. *International Journal for Numerical Methods in Engineering*, Vol. 91, No. 4, pp. 358–377.

Amsallem, D, Zahr, M, Choi, Y and Farhat, C. (2015) Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, Vol. 51, pp. 919–940.

Audouze, C, De Vuyst, F and Nair, P. (2009) Reduced-order modelling of parameterized PDEs using time–space-parameter principal component analysis. *International Journal for Numerical Methods in Engineering*, Vol. 80, No. 8, pp. 1025-1057.

Audouze, C, De Vuyst, F and Nair, P. (2013) Nonintrusive Reduced-Order Modelling of Parametrized Time-Dependent Partial Differential Equations. *Numerical Methods for Partial Differential Equations*, Vol. 29, No. 5, pp. 1587-1628.

Breitkopf, P and Coelho, RF. (2010) Multidisciplinary Design Optimization in Computational Mechanics. Wiley, New York, NY.

Buchan, A, Calloo, A, Goffin, M, Dargaville, S, Fang, F, Pain, C and Navon, IM. (2015) A POD reduced order model for resolving angular direction in neutron/photon transport problems. *Journal of Computational Physics*, Vol. 296, pp. 138-157.

Cao, Y, Navon, IM and Luo, Z. (2007) A reduced order approach to four-dimensional variational data assimilation using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, Vol. 53, pp. 1571-1583.

Carlberg, K and Farhat C. (2011) Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, Vol. 86, pp. 155-181.

Chen W, Wang Q, Hesthaven JS and Zhang C. (2020) Physics-informed machine learning for reduced-order modelling of nonlinear problems. *Preprint*.

Da Ronch, A, Badcock, K, Wang, Y, Wynn, A and Palacios, R. (2012) Nonlinear model reduction for flexible aircraft control design. In: *AIAA Atmospheric Flight Mechanics Conference*, AIAA Paper 2012-4404.

Economon, TD, Palacios, F, Copeland, SR, Lukaczyk, TW and Alonso, JJ. (2016) SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, Vol. 54, No. 3, pp. 828-846.

Fang, F, Zhang, T, Pavlidis, D, Pain, C, Buchan, A and Navon, IM. (2014) Reduced order modelling of an unstructured mesh air pollution model and application in 2D/3D urban street canyons. *Atmospheric Environment*, Vol. 96, pp. 96-106.

Gautier, N, Aider, JL, Duriez, T, Noack, B, Segond, M and Abel, M. (2015) Closed-loop separation control using machine learning. *Journal of Fluid Mechanics*, Vol. 770, pp. 442-457.

Giere, S, Iliescu, T, John, V and Wells, D. (2015) SUPG reduced order models for convection-dominated convection diffusion reaction equations. *Computer Methods in Applied Mechanics and Engineering*, Vol. 289, pp. 454-474.

Guenot, M, Lepot, I, Sainvitu, S, Goblet, J and Coelho, RF. (2013) Adaptive sampling strategies for non-intrusive POD-based surrogates. *International Journal for Computer-Aided Engineering and Software*, Vol. 30, No. 4, pp. 521-547.

Haykin, S. (2004) Neural Networks: A Comprehensive Foundation. Prentice Hall, Upper Saddle River, NJ.

Hoang, K, Fu, Y and Song, J. (2016) An hp-proper orthogonal decomposition moving least squares approach for molecular dynamics simulation. *Computer Methods in Applied Mechanics and Engineering*, Vol. 298, pp. 548-575.

Kani, JN and Elsheikh, AH. (2017) DR-RNN: A deep residual recurrent neural network for model reduction. *arXiv preprint*, arXiv:1709.00939.

Kim, B, Azevedo, VC, Thuerey, N, Kim, T, Gross, M and Solenthaler, B. (2019) Deep fluids: A generative network for parameterized fluid simulations. In: *Computer Graphics Forum*, volume 38, pp. 59–70, Wiley Online Library.

Kingma, D and Ba, J. (2014) Adam: A method for stochastic optimization. *arXiv preprint*, arXiv:1412.6980.

Landon, RH. (1982) NACA0012 oscillatory and transient pitching. In: Compendium of Unsteady Aerodynamics, AGARD-R-702.

Ling, J, Kurzawski, A and Templeton, J. (2016) Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, Vol. 807, pp. 155-166.

Lovison, A and Rigoni, E. (2011) Adaptive sampling with a Lipschitz criterion for accurate metamodeling. *Communications in Applied and Industrial Mathematics*, Vol. 1, Issue 2, pp. 110-126.

Marquardt, DW. (1963) An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, Vol. 11, No. 2, pp. 431-441.

Maulik, R and San, O. (2017) A neural network approach for the blind deconvolution of turbulent flows. *Journal of Fluid Mechanics*, Vol. 831, pp. 151-181.

Moosavi, A, Stefanescu, R and Sandu, A. (2015) Efficient construction of local parametric reduced order models using machine learning techniques. *arXiv preprint*, arXiv:1511.02909.

Noori, R, Karbassi, A, Mehdizadeh, H, Vesali-Naseh, M and Sabahi, M. (2011) A framework development for predicting the longitudinal dispersion coefficient in natural streams using an artificial neural network. *Environmental Progress and Sustainable Energy*, Vol. 30, No. 3, pp. 439-449.

Pearson, K. (1901) On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, Vol. 2, No. 11, pp. 559-572.

Regazzoni, F, Dede, L and Quarteroni, A. (2019) Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics*, Vol. 397, 397: 108852.

Renganathan, SA, Maulik, R and Rao, V. (2020) Machine learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil. *Physics of Fluids*, Vol. 32, No. 4.

San, O, Maulik, R and Ahmed, M. (2019) An artificial neural network framework for reduced order modelling of transient flows. *Communications in Nonlinear Science and Numerical Simulation*, Vol. 77, pp.271-287.

Schlegel, M and Noack, BR. (2015) On long-term boundedness of Galerkin models. *Journal of Fluid Mechanics*, Vol. 765, pp. 325-352.

Sirovich, L. (1987) Turbulence and the dynamics of coherent structures. *Quarterly of Applied Mathematics*, Vol. 45, No. 3, pp. 561-571.

Swischuk R, Mainini L, Peherstorfer B and Willcox K. (2019) Projection-based model reduction: Formulations for physics-based machine learning. *Computers & Fluids*, Vol. 179, pp. 704-717.

Thuerey, N, Weißenow, K, Prantl, L and Hu, X. (2020) Deep learning methods for Reynolds-Averaged Navier–Stokes simulations of airfoil flows. *AIAA Journal*, Vol 58(1), pp. 25-36.

Vaidyanathan, S, Pehlivan, I, Dolvis, LG, Jacques, K, Alcin, M, Tuna, M and Koyuncu, I. (2019) A novel ANN-based four-dimensional two-disk hyperchaotic dynamical system, bifurcation analysis, circuit realisation and FPGA-based TRNG implementation. *International Journal of Computer Applications in Technology*, Vol. 62, No. 1, pp. 20-35.

Wan, ZY, Vlachas, P, Koumoutsakos, P and Sapsis, T. (2018) Data-assisted reduced-order modelling of extreme events in complex dynamical systems. *PloS One*, Vol. 13, No. 5.

Wang, Q, Hesthaven, JS and Ray, D. (2019) Non-intrusive reduced order modelling of unsteady flows using artificial neural networks with application to a combustion problem. *Journal of Computational Physics*, Vol. 384, pp. 289-307.

Xiao, D, Fang, F, Buchan, A, Pain, C, Navon, IM, Du, J and Hu, G. (2014) Non-linear model reduction for the Navier-Stokes equations using Residual DEIM method. *Journal of Computational Physics*, Vol. 263, pp. 1-18.

Xiao, D, Fang, F, Buchan, A, Pain, C, Navon, IM and Muggeridge, A. (2015) Non-intrusive reduced order modelling of the Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, Vol. 293, pp. 552-541.

Xiao, D, Fang, F, Pain, C and Navon, IM. (2017) A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications. *Computer Methods in Applied Mechanics and Engineering*, Vol. 317, pp. 868-889.

Xiao, D, Heaney, C, Fang, F, Mottet, L, Hu, R, Bistrian, D, Aristodemou, E, Navon, I and Pain, C. (2019) A domain decomposition non-intrusive reduced order model for turbulent flows. *Computers and Fluids*, Vol. 182, pp. 15-27.

Yan, R, Gong, S and Zhong, S. (2019) Crowd counting via scale-adaptive convolutional neural network in extremely dense crowd images. *International Journal of Computer Applications in Technology*, Vol. 61, No. 4, pp. 318-324.

Zhou, Q, Chen, G, Da Ronch, A and Li, Y. (2017) Reduced order unsteady aerodynamic model of a rigid aerofoil in gust encounters. *Aerospace Science and Technology*, Vol. 63, pp. 203-213.