International Journal for Numerical Methods in Engineering, Volume 124, Issue 13, July 2023, pp. 3087-3111
DOI:10.1002/nme.7240

A non-linear non-intrusive reduced order model of fluid flow by Auto-Encoder and self-attention deep learning methods

R. Fu[a], D. Xiao[b,*], I.M. Navon[c], F. Fang[d], L. Yang[f], S. Cheng[e], C. Wang[a]

[a]*ZCCE, Faculty of Science and Engineering, Swansea University, Bay Campus,Swansea, UK, SA1 8EN*
[b]*School of Mathematical Sciences,Tongji University, Shanghai, P.R. China,200092*
[c]*Department of Scientific Computing, Florida State University, Tallahassee, FL, 32306-4120, USA*
[d]*AMCG, Department of Earth Science and Engineering, Imperial College London, UK, SW7 2BP*
[e]*Data Science Institute, Imperial College London, London, SW7 2AZ, UK*
[f]*Division of Energy and Sustainability, Cranfield University, Bedford, MK43 0AL, UK*

Abstract

This paper presents a new nonlinear non-intrusive reduced-order model (NL-NIROM) that outperforms traditional Proper orthogonal decomposition (POD)-based reduced order model (ROM). This improvement is achieved through the use of Auto-Encoder (AE) and self-attention based deep learning methods. The novelty of this work is that it uses Stacked Auto-Encoder (SAE) network to project the original high-dimensional dynamical systems onto a low dimensional nonlinear subspace and predict fluid dynamics using an self-attention based deep learning method. This paper introduces a new model reduction neural network architecture for fluid flow problem, as well as a linear non-intrusive reduced order model (L-NIROM) based on proper orthogonal decomposition (POD) and self-attention mechanism. In the NL-NIROM, the SAE network compresses high-dimensional physical information into several much smaller sized representations in a reduced latent space. These representations are expressed by a number of codes in the middle layer of SAE neural network. Then, those codes at different time levels are trained to construct a set of hyper-surfaces using self-attention based deep learning methods. The inputs of the self-attention based network are previous time levels' codes and the outputs of the network are current time levels' codes. The codes at current time level are then projected back to the original full space by the decoder layers in the SAE network.

The capability of the new model, NL-NIROM, is demonstrated through two test cases: flow past a cylinder, and a lock exchange. The results show that the NL-NIROM is more accurate than the popular model reduction method namely POD based linear non-intrusive reduced order model (L-NIROM).

Keywords: NIROM, deep learning, Auto-Encoder, self-attention

1. Introduction

In engineering and physics, most physical phenomena are governed by conservation laws, which can be written as partial differential equations (PDEs) [1]. However, solving PDEs can be computationally intensive in particular for high-fidelity realizations[2]. In this case, the reduced-order models(ROMs) technology plays an important role as they are able to simulate physical systems accurately with several orders of magnitude CPU speed-up. The ROMs have been applied successfully to a number of research fields such as data assimilation [3], ocean modelling [4], shallow water equations [5, 6], air pollution prediction [7], polynomial systems [8], viscous and inviscid flows [9], Fluid-Structure Interaction [10], aerodynamic shape optimization [11], large-scale time-dependent systems [12], optimal control [13], circuit systems [14, 15], inverse problems [16], fluids [17], reservoir history matching [18] and turbulent flows [19, 20].

Among traditional model reduction methods, proper orthogonal decomposition (POD) combined with Galerkin projection is a popular model reduction method, and has been applied successfully to a number of fields. However, it lacks stability and it is highly inefficient for non-linear models[21, 22, 23]. Several stabilization methods such as calibration [24], Regularization[25], Petrov-Galerkin [26] have been presented. In addition, an empirical interpolation Method (EIM)[27], discrete empirical interpolation Method (DEIM) [23, 28, 29], a combination of Quadratic expansion and DEIM (residual DEIM method) [30], Petrov-Galerkin method [24] and a Gauss–Newton method with

*Corresponding author
Email address: xiaodunhui@tongji.edu.cn (D. Xiao)

approximated tensors (GNAT) [19] have been presented aimed at alleviating the nonlinear inefficiency of POD. The DEIM combined with POD method has been used to improve nonlinear term reduction efficiency, which means it improves ROM's CPU efficiency, but it does not improve the accuracy of the POD method.

Recently, in order to avoid the above mentioned issues, a data-driven non-intrusive reduced order modelling (NIROM) method was presented and has been applied to a number of research areas such as fluids, climate, turbulence, nonlinear problems [31, 32, 33, 34, 35]. Xiao et al. presented a number of NIROMs based on POD and interpolation methods and machine learning or deep learning methods [36, 31]. Wang et al. presented a NIROM based on POD and neural network and applied it to a combustion problem [37]. Jacquier et al. presented a NIROM based on deep neural network and POD and applied it to a flooding problem [38]. Ahmed presented a NIROM for non-ergodic flows using POD and a long short term memory neural network together with a principal interval decomposition [39].

As mentioned above, POD based model reduction methods restrict the state to evolve in a linear subspace (linear trial sub-spaces), which imposes a fundamental limitation on the efficiency and accuracy of the resulting ROM [40]. The deep learning method provides a feasible way of POD limitation of linear trial sub-spaces. Deep learning methods have been successfully applied into various fields. It is quite common to use deep learning for learning the fluid dynamics and parametric space (boundary or initial conditions). The deep learning methods for dimensionality reduction are not as many as traditional methods such as POD in ROM literature. One reason for this is that the recent algorithm advancements, such as back-propagation method, make the computation of neural network with deep hidden layers become possible. The Auto-Encoder is one type of deep learning method for dimensionality reduction, and can replace principal component analysis (PCA) or POD method. A number of work have proved that PCA has the same basis function space with the Auto-Encoder with linear activation function and one hidden layer [41]. The work paves the way to use Auto-Encoder with non-linear activation function as a non-linear dimensionality reduction method, which is like a non-linear version of PCA or POD method. Thus improving the limitation of POD method, which restricts the state to evolve in a linear subspace.

In this paper, we present a new data driven non-linear non-intrusive model reduction framework (NL-NIROM) using Stacked Auto-Encoder(SAE) network and self-attention based deep learning methods to tackle the above linear trial sub-spaces issue. The new NL-NIROM uses Stacked Auto-Encoder(SAE) network to map original high-dimensional dynamical systems into a nonlinear subspace and predict the fluid dynamics using an self-attention based deep learning method.

Deep learning method is a popular artificial neural network method as it has shown great potential in many research areas such as image recognition, materials, facial recognition, speech recognition, drug discovery, self-driving cars[42, 43, 44]. Auto-Encoder is a type of artificial neural network in which the input layer has the same dimensional size and data as the output layer. The high-dimensional inputs are passed into the network and are compressed in the network. The middle layer of the network has a smaller number of neurons compared to the input and output layers. Thus, the middle layer represents the inputs using reduced number of neurons (or referred to as 'codes' in Auto-Encoder network)[45]. Auto-Encoder has been applied to a number of areas such as image compression and denoising. Dimensionality reduction is the main application of Auto-Encoder network. Auto-Encoder has recently applied to Reduced Order Modelling [41, 46, 47, 48, 49]. In the work of [41], Auto-Encoder has been used for eigenvalue problems. In [46], the convolutional Auto-Encoder combined with self-attention has been used to reduce the dimensional size of the fluids images and Long short-term memory (LSTM) is used to predict the temporal fluid dynamics after reduction. Wiewel et al. uses a Auto-Encoder to conduct the dimensionality reduction and LSTM to predict the fluid dynamics [47].

The other deep learning method used in this work is the self-attention mechanism. It was presented to deal with long sequence prediction problems. As an outstanding deep learning algorithm, the attention mechanism has generated impressive results in natural language processing(NLP), computer vision and many other areas[50]. It imitates the process of organism observation behaviour, especially in human beings, which means that after scanning the whole image, a small target area in the image is noticed with more attention while other areas are neglected to various degrees. In encoder-decoder model, the general attention mechanism calculates attention scores of each item between the input(Source) sequence and the output(Target) sequence [51]. In self-attention-based models, each item in the input sequence has been assigned an attention score via the scaled dot-product of global Query vector(Q) and Key vector(K), which are linear projections from the inputs. The Q is obtained by transforming the input sequence into a information vector, which contains information needs to be searched or queried. The K vector is obtained by transforming the input sequence into a vector, which contains Keywords-like information. And the V is obtained by transforming the input sequence into a exact value-like vector. These three vectors Q, K and V are similar to a searching process from a searching website. For example, we input a word(Q) into a search engine, and the search engine searches the exact value

(V) via Keywords (K). The self-attention mechanism connects all items of the input sequence via those three vectors, even if the distance between the two items in the input sequence is large[52, 53]. Thus, the self-attention mechanism contains both of the long-term relationship and the local dependencies of items in the input sequence. Due to this advantage, it becomes popular recently in Natural Language Processing, visual tasks and speech recognition[54, 55]. Also, it has been used to construct a number of famous deep learning models such as Transformer[56] and Bert[57]. In this work, we use it to construct a NIROM.

In our previous work, we presented a number of NIROMs, such as NIROM based on POD and Taylor series [31], NIROM based on POD and Radial Basis Functions [58] and Long short-term memory (LSTM) [59]. This work extends our previous work via replacing dimensionality reduction stage in ROM, in particular, the POD process, with a deep learning method. In addition, self-attention based neural network architecture is used to represent the fluid dynamics in the reduced space and it is used to construct the linear non-intrusive reduced order model (L-NIROM) and NL-NIROM. We firstly use self-attention based deep learning architecture to construct a multi-variable response function (hypersurface) to predict the fluid dynamics in a latent space. The new presented NL-NIROM and L-NIROM are implemented under the framework of FLUIDITY [60]. The Fluidity is an open source, unstructured mesh, finite element computational fluid dynamics (CFD) three dimensional model and it is capable of numerically solving the Navier-Stokes (NS) equations. In order to illustrate the performance of this novel non-linear NIROM, two test cases, flow past a cylinder and a lock exchange are illustrated. In addition, the performance of this NL-NIROM is compared against the solutions of proper orthogonal decomposition (POD) based Linear NIROM(L-NIROM). The numerical results show that the new NL-NIROM is capable of capturing the details of flows while the CPU time is reduced by several orders of magnitude. In addition, it performs better than the POD based L-NIROM.

The structure of the paper is as follows. A detailed introduction of the governing equations is given in Section 2. Section 3 provides a brief overview of Linear non-intrusive reduced order model (L-NIROM) based on traditional POD method and self-attention based neural network architecture. Section 4 describes the newly non-linear non-intrusive reduced order model (NL-NIROM) based on SAE and self-attention. Section 5 illustrates the performance of this two types of NIROMs for two test cases: flow past a cylinder and lock exchange. Finally, in section 6, the summary and conclusions are presented.

2. Governing equations

The three dimensional (3D) non-hydrostatic and incompressible Navier-Stokes equations which describe the conservation of momentum and mass of fluids are given by

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nu\nabla^2\boldsymbol{u} + \nabla p = 0, \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{2}$$

where $\boldsymbol{u} \equiv (u, v, w)^T$ is the velocity field, p is the pressure, t is the time and $\nu > 0$ is the kinematic viscosity. The discretised form of the system can be written as

$$C^T\mathbf{u} = \mathbf{0}, \tag{3}$$

$$\mathcal{M}\frac{\partial \mathbf{u}}{\partial t} + \mathcal{A}(\mathbf{u})\mathbf{u} + \mathcal{K}\mathbf{u} + C\mathbf{p} = \mathbf{0}, \tag{4}$$

where C is a pressure gradient matrix, $\mathcal{M}$ denotes the mass matrix, $\mathcal{A}(\mathbf{u})$ denotes the solution dependent streaming operator and $\mathcal{K}$ denotes the matrix related to the remaining linear velocity terms. The velocity, $\mathbf{u}$, is a vector containing nodal values of all three components now, likewise, $\mathbf{p}$ is a vector containing the pressure nodal values.

3. Linear Non-intrusive reduced order modelling

This section presents a novel linear non-intrusive reduced order model based on traditional model reduction method: POD and self-attention based neural network architecture.

3.1. Proper orthogonal decomposition

In the POD method, any variables to be solved $\mathbf{u}^n$ (for example, the velocity, pressure or temperature) at a time level n can be expressed by the following expansion,

$$\mathbf{u} = \overline{\mathbf{u}} + \sum_{j=1}^{c} \alpha_j \boldsymbol{\phi}_j, \tag{5}$$

where α_j represents the jth POD coefficient, $\boldsymbol{\phi}_j$ denotes the jth POD basis function and $\overline{\mathbf{u}}$ is the mean of snapshots for the variable $\mathbf{u}$. c is the number of basis functions which can represent most of energy (99% for example) within the chosen snapshot solutions.

The solutions $\mathbf{x}$ of the discretised governing equations(3 and 4) during certain simulation time period can form a snapshot matrix A .

An SVD calculation is applied to the matrix $\boldsymbol{A}$ which results in three matrices,

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{6}$$

where $\boldsymbol{U}$ is left singular vectors with a size of $N \times N$, $\boldsymbol{V}$ is right singular vectors with a size of $\mathcal{N}_s \times \mathcal{N}_s$ and $\boldsymbol{\Sigma}$ is singular values matrix ($N \times \mathcal{N}_s$). The left singular vectors matrix $\boldsymbol{U}$ contains the POD basis functions. The singular values in $\boldsymbol{\Sigma}$ give an indication as to how important each basis function is. If the singular value is small its corresponding basis function can be discarded. A common criterion for choosing the number of POD basis functions is as follows: for a tolerance $\varepsilon \lesssim 1$, find the smallest integer c such that

$$\frac{\sum_{i=1}^{c} \lambda_i^2}{\sum_{i=1}^{\mathcal{N}_s} \lambda_i^2} \geqslant \varepsilon, \tag{7}$$

where λ_i represents the i^{th} singular value and $c \leqslant \mathcal{N}_s$. For example, if ε takes the value 0.99, this means 99% of the energy of the physical system would be captured by the first c leading POD basis functions.

3.2. Self-attention-based deep learning method

Self-attention was introduced by Vaswani et al.[56] as the basis of Transformer model, to deal with the long sequence modelling and transduction challenges faced by the encoder-decoder network in natural language processing. The self-attention mechanism allows the source inputs to interact with each item (self process) and find out which item should be paid more attention ('attention' process). This makes the process of modelling long-range dependencies in a sentence faster and more accurate than recurrent neural network (RNN) and convolutional neural network (CNN) models[61]. The global information of the whole sequence could be captured with weights according to the computed attention map. It allows self-attention models to show great potentials in long sequence processing and prediction. The self-attention mechanism generates the relationship between each item in the input sequence such as POD coefficients or code information in the reduced latent space via three matrices: query(Q), key(K) and Value (V). The working process of those three matrices is similar to the process of a searching engine. The matrix Q, as the query, is the word we entered in the searching bar, aiming to find out the best matching content from the database. The matrix K, similar to the keywords, is the important information extracted from each item of the database, and it helps to accelerate the searching process and save energy. The matrix V is the real value that need to be searched. In self-attention mechanism, the correlation between two items is represented by the result of their similarity calculation, as shown in equation 11, dot-product calculations are selected to find out the similarity between the matrix Q of one item and matrices K of all items orderly in this sequence as the attention weights of this item. The self-attention neural network architecture has shown to perform better than other models in long sequence prediction, such as LSTM [62, 63, 64].

In this work, we use the self-attention mechanism for temporal fluid dynamics prediction in the reduced space. Unlike previous popular sequential models like LSTM, the self-attention mechanism learns the internal structure of the reduced temporal fluid dynamics by calculating dependencies regardless of the distance between two time levels in the inputs. For example, the source sequence (POD coefficients) $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_{\tau-1}$ generated by POD method at different time levels: $t = 0, t = 1, ..., t = \tau - 1$ are not necessarily inputted into the network in order. It is able to generate relationships between two items with the largest distance, for example, $\boldsymbol{\alpha}_{\tau-1}$ and $\boldsymbol{\alpha}_0$. τ is the number of time levels or source sequence length. The self-attention mechanism uses the position embedding to save the temporal position information for each time level. In this work, Time2vec[65], is used as the position embedding method (time embedding).

The attention block that represents fluid dynamics for our reduced order model is given in the Figure 1. It is similar to only encoder part of a complete Transformer architecture, which is a sequence prediction model. Six attention blocks, also called TF-Encoder blocks, are stacked as the main part of our model. In each block, instead of a single attention function, groups of queries(Q), keys(K) and value(V) are projected h times(h=8) respectively as the multi-head attention block with eight heads. This allows the model to jointly attend to information from different representation subspace at different positions[56]. In Figure 1, the POD coefficients $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_{\tau-1}$ are inputted into the multi-head attention in the left side of the figure. The output of the attention block is the next time level's POD coefficient $\boldsymbol{\alpha}_\tau$. And in the attention block, a position-wise fully connected feed forward network(FFN) block is following the multi-head attention block as another important sub-layer. One nonlinear activate function(ReLU) layer with larger scale and one linear layer are applied here aiming to enhance capability of extracted representations. The detailed operations inside the multi-head attention and feed forward layer can be found in Figure 2 (a) and (b) respectively. We also employed the residual connection[66] around multi-head attention block and feed forward block separately to avoid degradation of results in stacked model. The following layer, normalization layer helps to project the representations into the input range of the activate function and simplify learning process of our model. The residual connection and layer normalization functions are shown with similar layers but different inputs in step 5 and 7. The FFN block with two layers is shown in 6. This whole attention block considers each item in inputs with different weights matrices, then paying more attention on similar part and suppressing other useless information, which is important for the prediction process. Especially, both global and local connections of inputs are considered together with parallelize computation in multi-head attention part.

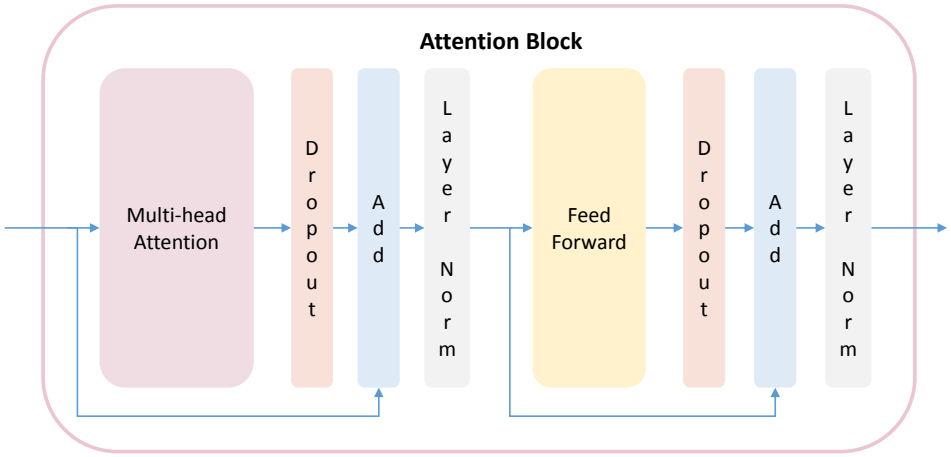


Figure 1: The figure shows attention block in the NIROM architecture

The main process of self attention block has been summarised in Algorithm 1. In step (1), each code α_t ($t = 0, 1, 2, ..., \tau - 1$) represents one time levels complete set $\alpha_t = (\alpha_t^1, \alpha_t^2, ..., \alpha_t^j, ...\alpha_t^c, \alpha_t^{t2v_{linear}}, \alpha_t^{t2v_{sin}})$. c is the number of basis functions in Equation 5 and $\alpha_t^{t2v_{linear}}, \alpha_t^{t2v_{sin}}$) are linear and periodic time presentations calculated by Time2Vec(t2v) mechanism [65]. The t2v mechanism is developed to save the temporal information in the form of a position embedding. A multi-head attention will be used then to gather different fluid dynamics in different subspace. After the multi-head attention, FNN prediction and normalisation will be performed. If self-attention mechanism is directly used to predict the fluid dynamics in the full original high dimensional space, then the dimensional size could be huge, and it could take long time to train the neural network. As such, the Auto-Encoder can be used to reduce the dimensional size of data in the full original space before performing the self-attention prediction.

3.3. Linear NIROM based on POD and self-attention

This section presents the method of construction of a Linear Non-Intrusive Reduced Order Model (L-NIROM) based on POD and self-attention neural network. POD based L-NIROM defines a linear projection between the field values $\boldsymbol{u}$ and the corresponding POD coefficients α in reduced space. The construction of this L-NIROM is similar to a NIROM based on POD and Gaussian Process Regression (GPR) [67]. The difference is that the L-NIROM presented in this work uses self-attention mechanism to represent the fluid dynamics in the reduced space.

Algorithm 1: Main process of self attention block

(1) Converting the inputs $\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_t, ..., \boldsymbol{\alpha}_{\tau-1}$ into a feature vector. $(t = 0, 1, 2, ..., \tau - 1)$ The inputs of attention mechanism include the concatenation of codes extracted by Autoencoder and positional embedding by Time2vec. The conversation process changes the inputs $\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{\tau-1}$ into $\tilde{\boldsymbol{\alpha}}$. $\alpha_t = (\alpha_t^1, \alpha_t^2, ..., \alpha_t^j, ...\alpha_t^c, \alpha_t^{t2v_{linear}}, \alpha_t^{t2v_{sin}})$

(2) Initialising h groups of trainable presentations, as multi-head, of the three vectors(Q_i, K_i, V_i), W_i^Q, W_i^K and W_i^V $(i = 1, 2, ..., h)$ by linear projections of input matrix $\tilde{\boldsymbol{\alpha}}$ respectively.

$$Q_i = \tilde{\boldsymbol{\alpha}} \times W_i^Q; \tag{8}$$

$$K_i = \tilde{\boldsymbol{\alpha}} \times W_i^K; \tag{9}$$

$$V_i = \tilde{\boldsymbol{\alpha}} \times W_i^V; \tag{10}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, and $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$.

(3) Calculating the weighted scaled attention scores by Attention function for each inputs using:

$$Attention = softmax(\frac{Q_i K_i^\top}{\sqrt{d_k}})V_i, i = 1, 2, ..., h \tag{11}$$

where $\sqrt{d_k}$ is a scaling factor. The softmax function converts the vector into a probability distribution within a range of [0, 1]. This score denotes the contribution from each input.

(4) Concatenating multi-head attention mechanism outputs: concatenate calculated representations from heads $head_i$ together.

$$Multihead(\tilde{\boldsymbol{\alpha}}) = Multihead(Q, K, V) = Concat(head_1, head_2, head_i..., head_h)\mathcal{W} \tag{12}$$

$$head_i = Attention(Q_i, K_i, V_i), i = 1, 2, ..., h \tag{13}$$

where the projection $W \in \mathbb{R}^{hd_v \times d_{model}}$.

(5) Adding and Layer normalization: a residual network and a normalization layer are implemented between the input matrix $\tilde{\boldsymbol{\alpha}}$ and its Multi-head attention mechanism outputs.

$$\tilde{\boldsymbol{\beta}} = LayerNorm(\tilde{\boldsymbol{\alpha}} + Multihead(\tilde{\boldsymbol{\alpha}})) \tag{14}$$

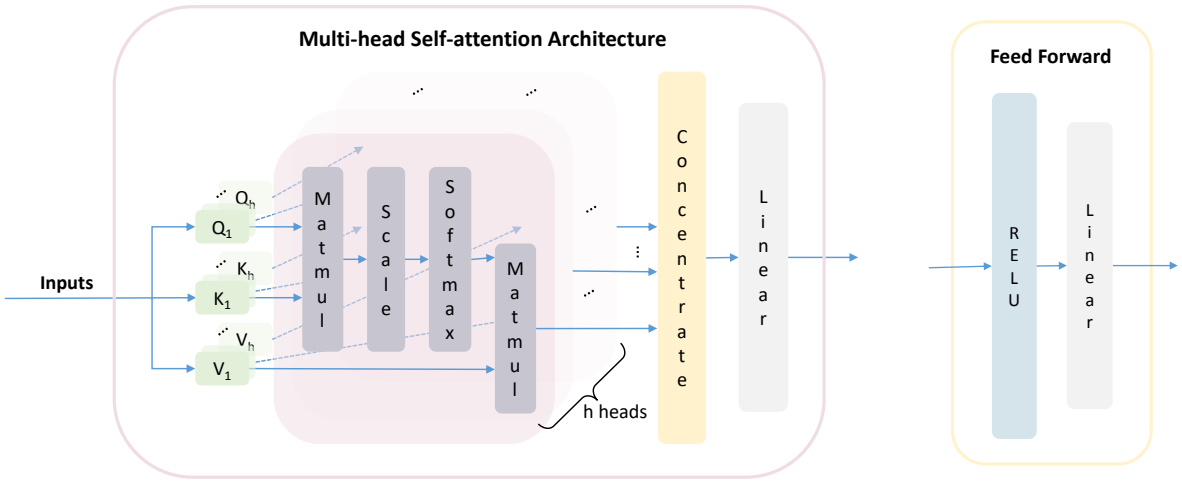
(6) Performing Feed-Forward Networks(FNN) output. This process consists of a Rectified Linear Unit (ReLU) and linear transformations. It can be calculated by

$$F(\tilde{\boldsymbol{\beta}}) = max(0, \tilde{\boldsymbol{\beta}}\mathcal{W}_1 + b_1)\mathcal{W}_2 + b_2 \tag{15}$$

where $\mathcal{W}$ and b are weights and bias that need to be optimised in the training process of this model.

(7) Adding and Layer normalization: a residual network and a normalization layer are implemented between the input matrix of FNN block $\tilde{\boldsymbol{\beta}}$ and its outputs.

$$output = LayerNorm(\tilde{\boldsymbol{\beta}} + F(\tilde{\boldsymbol{\beta}})) \tag{16}$$



(a) Multi-head attention architecture (b) Feed Forward Block

Figure 2: Multi-head attention architecture and Feed forward block architecture

In equation 5, the POD coefficient α_j are solutions in reduced space and it can be solved by

$$\alpha_j(t+\Delta t) = f_j^{ATN}(\boldsymbol{\alpha}(t-\tau+1), ..., \boldsymbol{\alpha}(t-1), \boldsymbol{\alpha}(t)) \quad \forall j \in \{1, 2, \ldots, c\}. \tag{17}$$

subject to the initial condition

$$\alpha_j(0) = ((\mathbf{u}(0) - \overline{\mathbf{u}}), \phi_j), \tag{18}$$

In above equation 17, the time step, Δt, will coincide with that of the high-fidelity full model. τ denotes the number of time levels in the input sequence in training data pair. α_j denotes the j^{th} POD coefficient. c denotes the number of POD basis functions and it means the dimensional size in the reduced space. f_j^{ATN} is the response function that needs to approximated by self-attention neural network. In order to construct the self-attention neural network, $\mathcal{N}_s$ pairs of input and output training data have been used to form the response function f_j^{ATN}.

$$\text{input:} \quad \boldsymbol{\alpha}^{t-\tau+1}, \boldsymbol{\alpha}^{t-\tau+2}, \ldots, \boldsymbol{\alpha}^{t-1}, \boldsymbol{\alpha}^{t} \tag{19}$$

$$\text{output:} \quad \alpha_j^{t+1}, \tag{20}$$

for all $t \in \{1, 2, \ldots, \mathcal{N}_s\}$. The bold $\boldsymbol{\alpha}$ is one complete code information vector ($\boldsymbol{\alpha} = \alpha_1, \alpha_2, ..., \alpha_c$). The input consists of τ time levels' complete POD coefficients. The output α_j^k denotes j^{th} POD coefficients for time level t. This procedure is repeated for each code information (i.e. for $j \in \{1, 2, \ldots, c\}$), and once all the functions $\{f_j^{ATN}\}_{j=1}^{c}$ have been determined, the off-line stage is complete. The f_j^{ATN} will be optimised by the self-attention neural network architecture. This optimisation problem can be mathematically expressed as follows:

$$\begin{aligned} \arg\min_{\mathcal{W}, \boldsymbol{b}} \quad \mathcal{L}_{\text{oss}}(\mathcal{W}, \boldsymbol{b}) &= \frac{1}{\mathcal{N}_s} \sum_{j=1}^{\mathcal{N}_s} \left\| \boldsymbol{\alpha}_{\text{POD}} - \boldsymbol{\alpha}_{\text{attention}} \right\|_2^2 \\ &= \frac{1}{\mathcal{N}_s} \sum_{j=1}^{\mathcal{N}_s} \left\| \boldsymbol{\alpha}_{\text{POD}} - \left((\boldsymbol{\alpha}_i \mathcal{W}_1 + b_1) \mathcal{W}_2 + b_2 \right) \right\|_2^2, \end{aligned} \tag{21}$$

where $\mathcal{W}$ and $\mathbf{b}$ are weights and bias that need to be optimised in the neural network architecture. $\boldsymbol{\alpha}_{\text{POD}}$ denotes the POD coefficients, and $\boldsymbol{\alpha}_{\text{attention}}$ is the POD coefficients that are predicted by self attention based neural network. If three hidden layers used in the self-attention neural network, then the $\boldsymbol{\alpha}_{\text{attention}}$ will be $((\boldsymbol{\alpha}_i \mathcal{W}_1 + b_1) \mathcal{W}_2 + b_2) \mathcal{W}_3 + b_3$. Larger number of hidden layers, that process will be repeated. The prediction process can be found in Algorithm 1.

The procedure of on-line prediction using the L-NIROM is summarized in Algorithm 2. The number of time steps ($\mathcal{T}$) can be different from that used in the training period. That is, the L-NIROM can be run for a longer or shorter time

than the high-fidelity full model.

Algorithm 2: On-line Linear NIROM calculation

The response functions, $\left\{f_j^{ATN}\right\}_{j=1}^{c}$ are known.

The initial condition ($\boldsymbol{\alpha}^0$), time step (Δt), initial time (t_0) and number of time steps ($\mathcal{T}$) are given.

for $n = 1$ *to* $\mathcal{T}$ **do**
  $t = t_0 + n\Delta t$ current time

  Step (a): calculate the POD coefficients, $\boldsymbol{\alpha}^n$, at the current time step:
  for $j = 1$ *to* c **do**
    $\alpha_j^n = f_j^{ATN}(\boldsymbol{\alpha}^{n-\tau}, \boldsymbol{\alpha}^{n-\tau+1}, \ldots, \boldsymbol{\alpha}^{n-1})$
  endfor

  Step (b): obtain the solutions, velocity $\mathbf{u}^n$ for example, in the full space at the current time, t, by projecting α_j^n back onto the full space using Equation 22.

$$\mathbf{u} = \overline{\mathbf{u}} + \sum_{j=1}^{c} \alpha_j \boldsymbol{\phi}_j, \tag{22}$$

endfor

4. Non-linear NIROM based on autoencoder and self-attention

This section presents a new non-linear NIROM using stacked Auto-Encoder self-attention neural networks. In the dimensionality reduction process, we use a non-linear dimensionality reduction method, Auto-Encoder neural network, to project the full original physical system into a non-linear subspace.

4.1. Stacked Auto-Encoder neural network

The basic Auto-Encoder (AE) is an unsupervised feed-forward neural network that can be used to reduce the dimensional size. Instead of labelling the outputs for training, the AE network sets the output the same number of nodes (neurons), shape and values as the inputs. It first maps the input data into a reduced dimensional latent space (represented by a number of codes in the middle layer of the neural network) and then projects the latent representation with reduced dimensional size to the output, see Figure 3.

The purpose of this particular neural network architecture is to reconstruct the input data into reduced dimensional size. The input data will be velocity $\mathbf{u}$, pressure $\mathbf{p}$ in flow past a cylinder case and temperature for the lock exchange test case. An Auto-Encoder consists of two main parts, the encoder and the decoder, which can be described as transitions $\mathcal{E}$ and $\mathcal{D}$, such that:

$$\mathcal{E} : \mathbf{X} \rightarrow \mathcal{H} \tag{23}$$

$$\mathcal{D} : \mathcal{H} \rightarrow \mathbf{X} \tag{24}$$

$$\mathcal{E}, \mathcal{D} = \underset{\mathcal{E},\mathcal{D}}{\arg\min} \|\mathbf{X} - (\mathcal{E} \circ \mathcal{D})\mathbf{X}\|^2 \tag{25}$$

In the simplest case (only one hidden layer), the encoder takes the input $\mathbf{x} \in \{t_0, t_1, t_2, ..., t_m\} = \mathbf{X}$ and maps it to $\boldsymbol{\varrho} = \mathcal{H}$:

$$\boldsymbol{\varrho} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}). \tag{26}$$

In equation 26, the $\boldsymbol{\varrho}$ is the code, or latent variables, or a reduced representation of the full system and will be used for training the self-attention deep neural network. These representations are similar to the POD coefficients in POD

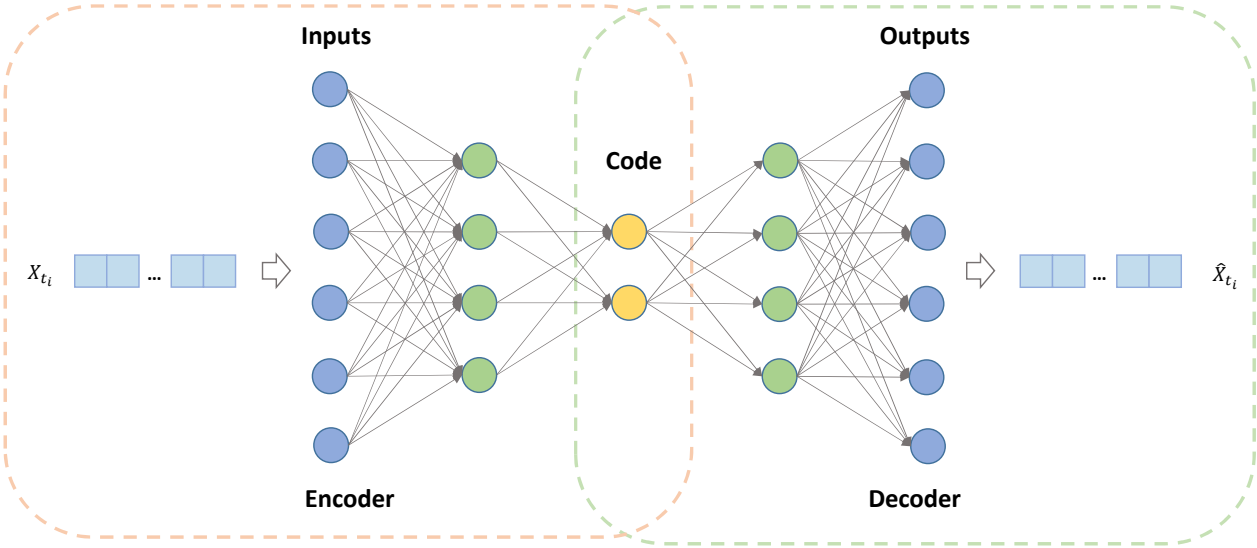


Figure 3: SAE Structure

based L-NIROM, but not identical. The dimensional size m in the reduced latent space can also be determined by SVD method, which means that we can still use equation 7 to calculate m in Auto-Encoder neural network. One reason is that Auto-Encoder neural network will capture more non-linear information using the same number of dimensional size.

σ in equation 26 is an activation function such as a *sigmond* function:

$$\sigma(x) = 1/(1 + e^{-x}) \tag{27}$$

or *tanh* function:

$$tanh(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x}). \tag{28}$$

$\mathbf{W}$ in equation 26 is the weights vector and $\mathbf{b}$ is a bias vector. The weights and biases are both initialised randomly, and then updated iteratively using back-propagation. Then, at the decoder stage, the Auto-Encoder maps $\boldsymbol{\varrho}$ to $\mathbf{x}'$ with the same shape as the input $\mathbf{x}$:

$$\mathbf{x}' = \sigma'(\mathbf{W}'\boldsymbol{\varrho} + \mathbf{b}'), \tag{29}$$

where $\mathbf{W}'$, σ' and $\mathbf{b}'$ are weights, activation function and bias vector respectively for the decoder. The Auto-Encoder is trained to minimise the errors between the inputs $\mathbf{x}$ and the reconstruction $\mathbf{x}'$. The loss function or cost function minimising the errors is then can be described as:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2 \tag{30}$$

The stacked Auto-Encoder(SAE) is a neural network architecture with more then one hidden layers.

4.2. Non-linear non-intrusive reduced order modelling based on Auto-Encoder and Self-attention

This work presents a novel neural network architecture for reduced order model. The architecture can be described in Figure 4. This Non-linear non-intrusive reduced order modelling (NL-NIROM) architecture consists of two main parts: Auto-Encoder network and self-attention part. The Auto-Encoder network (above part in the figure) is constructed for projecting the full high-dimensional space into a reduced space (latent space). The multi-head scaled-dot self-attention part (bottom part in the figure) is a structure that is used to represent the fluid dynamics in the reduced space. Its inputs are the codes from the middle layer of Auto-Encoder network and its outputs will be next time level's predicted codes and then they are projected back to the full space via decoder. It includes input embedding, positional embedding, attention blocks, flatten, Rectified Linear Unit (ReLU) and linear modules. This is similar to half of the complete transformer architecture (Encorder part of the transformer architecture).

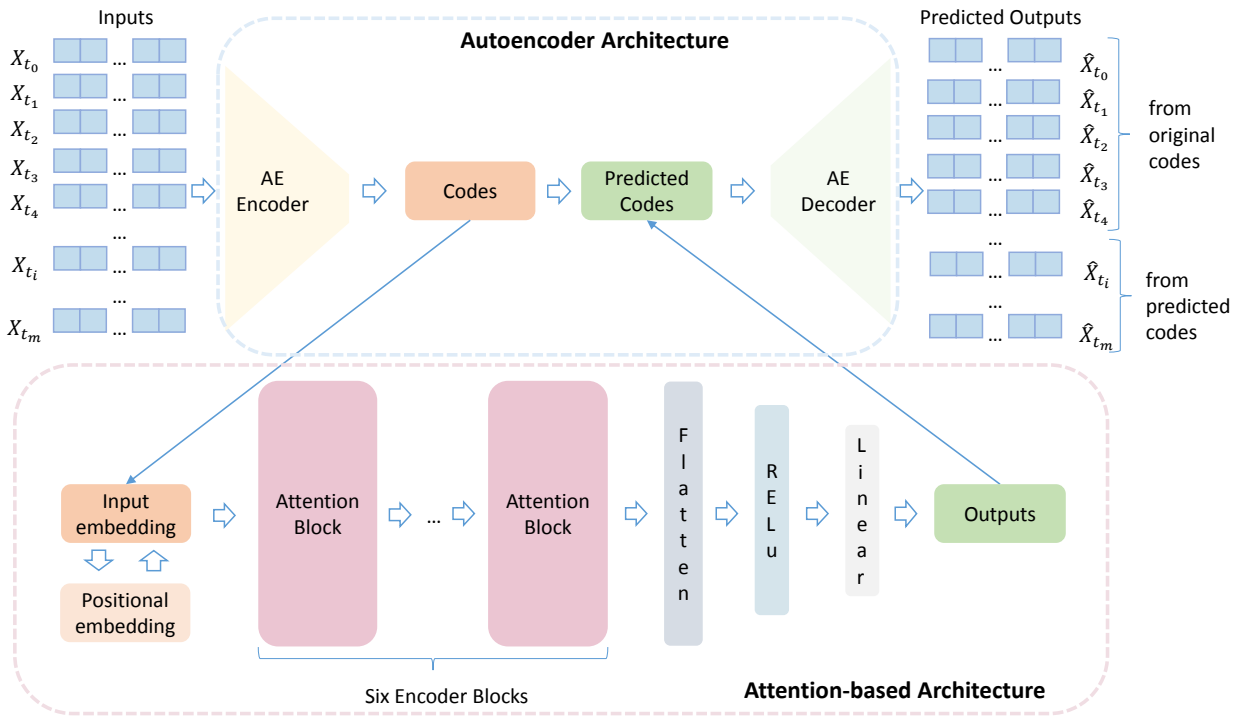


Figure 4: The figure shows neural network architecture for NL-NIROM

4.2.1. Constructing the NL-NIROM (offline stage)

Having obtained the codes in the SAE network (middle layer of the SAE network) in section 4.1, the offline stage of NL-NIROM will be completed by approximating the governing equations. This is achieved by using trained self attention based deep neural network to predict how the governing equations would behave. All simulation results generated by Fluidity at different time levels during a time period are projected onto the reduced space by the Auto-Encoder neural network architecture. The dimensional size of the codes in the middle layer of SAE is much smaller than that of the inputs, thus decreasing the dimensional size of the problem drastically. The codes information is similar to the POD coefficients in the POD method. The weights of neurons in other layers (excluding input, output and middle layers) store the projection information (projecting full original space into a reduced space and projecting back), which are similar to the POD basis functions in the POD method, see [58]. The codes are then used to train the self-attention deep neural network. This training results in a function f_j for each code value (reduced coefficient), which maps the set of code values from τ time levels $(\boldsymbol{\varrho}^{k-\tau-1}, \boldsymbol{\varrho}^{k-\tau}, \ldots, \boldsymbol{\varrho}^{k-1})$ to the associated code value at the next time level (ϱ_j^k), i.e.

$$\begin{aligned} \varrho_j^k &= f_j(\boldsymbol{\varrho}^{k-\tau}, \boldsymbol{\varrho}^{k-\tau+1}, \ldots, \boldsymbol{\varrho}^{k-1}) \\ &= f_j(\varrho_1^{k-\tau}, \varrho_2^{k-\tau}, \ldots, \varrho_m^{k-\tau}, \varrho_1^{k-\tau+1}, \varrho_2^{k-\tau+1}, \ldots, \varrho_m^{k-\tau+1}, \ldots, \varrho_1^{k-1}, \varrho_2^{k-1}, \ldots, \varrho_m^{k-1}), \\ &\qquad \forall k \in \{1, 2, \ldots, \mathcal{N}_s\}. \end{aligned} \tag{32}$$

where τ denotes input data length in training data. ϱ_j denotes the j^{th} code information for SAE method. m denotes the number of codes used in the SAE network and it means the dimensional size in the reduced latent space. $\mathcal{N}_s$ denotes the total number of time levels, and it equals to the total number of snapshots. By including the initial condition, we have $\mathcal{N}_s$ pairs of input and output data that are used to form the response function f_j. The inputs of the function f_j is $\boldsymbol{\varrho}^{k-\tau}, \boldsymbol{\varrho}^{k-\tau+1}, \ldots, \boldsymbol{\varrho}^{k-2}, \boldsymbol{\varrho}^{k-1}$ and output is ϱ_j^k, for all $k \in \{1, 2, \ldots, \mathcal{N}_s\}$. The bold $\boldsymbol{\varrho}$ is one complete code information vector ($\boldsymbol{\varrho} = \varrho_1, \varrho_2, ..., \varrho_m$). The input consists of τ time levels' complete code information. The output ϱ_j^k denotes j^{th} code information for time level k. This procedure is repeated for each code information (i.e. for $j \in \{1, 2, \ldots, m\}$), and once all the functions $\{f_j\}_{j=1}^m$ have been determined, the off-line stage is complete. The f_j is optimised by the

Algorithm 3: Main process of NL-NIROM

(1) Generating N_t snapshots (for example, solutions $\boldsymbol{u}$ or $\boldsymbol{p}$) by running the high fidelity full model governed by Equation (3) and (4);

(2) Treating one time level's snapshot as one sample, and there are N_t snapshots samples in total to train the Auto-Encoder network;

(3) Calculating the code values using trained Auto-Encoder network;

(4) Constructing self attention based deep learning network;

(5) Predicting code values using trained self attention based deep learning network;

(6) Projecting the predicted code values in the reduced latent space back into the full space using trained Auto-Encoder network.

$$\mathbf{x}' = \sigma'(\mathbf{W}'\boldsymbol{\varrho} + \mathbf{b}'), \tag{31}$$

self-attention neural network architecture. This optimisation problem can be mathematically expressed as follows:

$$\begin{aligned} \arg\min_{\mathcal{W},\boldsymbol{b}} \quad \mathcal{L}_{\text{oss}}(\mathcal{W}, \boldsymbol{b}) &= \frac{1}{\mathcal{N}_{\text{s}}} \sum_{j=1}^{\mathcal{N}_{\text{s}}} \left\| \boldsymbol{\varrho}_{\text{AE}} - \boldsymbol{\varrho}_{\text{attention}} \right\|_2^2 \\ &= \frac{1}{\mathcal{N}_{\text{s}}} \sum_{j=1}^{\mathcal{N}_{\text{s}}} \left\| \boldsymbol{\varrho}_{\text{AE}} - \left((\boldsymbol{\varrho}_i \mathcal{W}_1 + b_1) \mathcal{W}_2 + b_2 \right) \right\|_2^2, \end{aligned} \tag{33}$$

where $\mathcal{W}$ and $\mathbf{b}$ represent weights and bias that need to be optimised in the self-attention neural network architecture. $\boldsymbol{\varrho}_{\text{AE}}$ denotes the code values in the Auto-Encoder neural network, and $\boldsymbol{\varrho}_{\text{attention}}$ is the code values that are predicted by self attention neural network. If three hidden layers are used in the self-attention neural network, then the $\boldsymbol{\varrho}_{\text{attention}}$ will be $((\boldsymbol{\varrho}_i \mathcal{W}_1 + b_1)\mathcal{W}_2 + b_2)\mathcal{W}_3 + b_3$. Larger number of hidden layers, that process will be repeated. The weight distribution of each head follows certain accuracy of the initial distribution, and then it is fine-tuned during the following epochs training. After the training converge step by step, the attention maps of heads are trained as learned weights with more precise and finer details.

4.2.2. Running simulations with the NL-NIROM (online stage)

For running the NL-NIROM, the functions $\{f_j\}_{j=1}^m$ are treated as response functions allowing the code information at one time level to be predicted given those at τ previous time levels

$$\varrho_j(t + \Delta t) = f_j(\boldsymbol{\varrho}(t), \boldsymbol{\varrho}(t-1), ..., \boldsymbol{\varrho}(t-\tau+1)) \quad \forall\, j \in \{1, 2, \ldots, m\}\,. \tag{34}$$

We remark that when running the NL-NIROM, the time step, Δt, will coincide with that of the high-fidelity full model. The procedure of on-line prediction using the NL-NIROM is summarized in Algorithm 4. The initial condition can be different than that used in the high-fidelity full model. After constructing the NL-NIROM, it can start from any time levels, which means the number of time steps ($\mathcal{T}$) can be different than that used in the training period. That is, the NL-NIROM can be run for a longer or shorter time than the high-fidelity full model.

Algorithm 4: On-line NL-NIROM calculation

The response functions, $\left\{f_j\right\}_{j=1}^{m}$ are known and Auto-Encoder network is already trained.

The initial condition ($\boldsymbol{\varrho}^0$), time step (Δt), initial time (t_0) and number of time steps ($\mathcal{T}$) are given.

for $n = 1$ *to* $\mathcal{T}$ **do**
  $t = t_0 + n\Delta t$ current time

  Step (a): calculate the code values, $\boldsymbol{\varrho}^n$, at the current time step:
  for $j = 1$ *to* m **do**
    $\varrho_j^n = f_j(\boldsymbol{\varrho}^{n-\tau}, \boldsymbol{\varrho}^{n-\tau+1}, \ldots, \boldsymbol{\varrho}^{n-1})$
  endfor

  Step (b): obtain the solutions, velocity $\mathbf{u}^n$ for example, in the full space at the current time, t, by projecting ϱ_j^n back onto the full space using the Auto-Encoder network using Equation 29.

  $\mathbf{u}^n = \sigma'(\mathbf{W}'\boldsymbol{\varrho} + \mathbf{b}')$

endfor

5. Illustrative numerical examples

In this section, we demonstrate the capability of NL-NIROM using two test problems, namely, a 2D flow past a circular cylinder and a lock exchange. The full model with simulation solutions of these two problems are generated via the finite element fluid model Fluidity[68]. In both test cases, unstructured triangular meshes were used with sufficient resolution to ensure accurate solutions. Using this snapshot data the NL-NIROM were constructed and then used to predict the problems.

In this demonstration a comparison between Auto-Encoder based NL-NIROM and POD based ROM has been carried out. In addition to comparing solution profiles, correlation coefficients and solution errors (root-mean-square errors(RMSE)) are analyzed. The formulation of CC can be described as,

$$CC(X(t), \hat{X}(t)) = \frac{cov(X(t), \hat{X}(t))}{\sigma_{X(t)}\sigma_{\hat{X}(t)}} = \frac{E[(X(t) - \mu_{X(t)})(\hat{X}(t) - \mu_{\hat{X}(t)})]}{\sigma_{X(t)}\sigma_{\hat{X}(t)}}, \tag{35}$$

where $\hat{X}(t)$ and $X(t)$ are ROM and high-fidelity full model at time level t respectively. The $\mu_{X(t)}$ and $\mu_{\hat{X}(t)}$ are the expected values of $X(t)$ and $\hat{X}(t)$, $\sigma_{X(t)}$ and $\sigma_{\hat{X}(t)}$ are standard deviations. The correlation coefficient measures the strength of relationship between two variables, for example, high-fidelity model solutions and NIROM solutions in this work. It shows how close the NIROM's solution is to the high fidelity model's solutions. If the values are greater than 0.8, it means that the relationship between two variables is considered to be significant.

The RMSE measures the differences between solutions and it is computed as

$$RMSE(t) = \sqrt{(\frac{1}{F})\sum_{i=1}^{F}(\hat{x}^i(t) - x^i(t))^2}, \tag{36}$$

where F is the number of nodes in the computational domain.

Also, the reconstruction descriptors $\hat{f}$ from ROM models are compared with the full physical space values f separately. The relative error $\triangle f$ can be measured in the relative L2-norm below:

$$\triangle f = \frac{\left\|\hat{f} - f\right\|_2}{\|f\|_2} \tag{37}$$

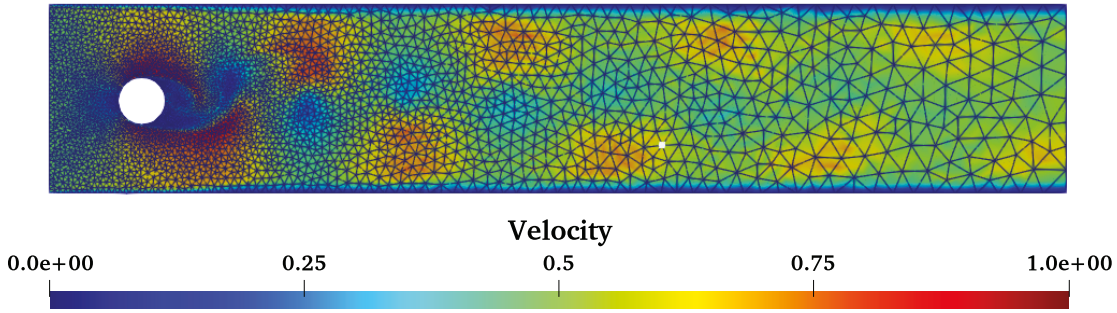


Figure 5: Case 1: computational domain of flow past a cylinder test case

5.1. Case 1: Flow past a cylinder

In the first numerical example, a two-dimensional flow past a cylinder is simulated. The computational domain is 2 units in length and 0.4 units in width, and it includes a cylinder of diameter 0.1 units (L) positioned over the centre point at $(0.2, 0.2)$, as shown in Figure 5. The dynamics of the fluid flow is driven by an in-flowing liquid that enters the domain via the left boundary with an inlet velocity of 0.5 meters per second ($V = 0.5$). The fluid flows past the cylinder and through the right boundary of the computational domain. No-slip and zero-outward-flow conditions are applied to the lower and upper edges of the domain, while Dirichlet boundary conditions are applied to the wall of the cylinder. The Reynolds number for this problem is set to 1000, with a kinematic viscosity of $5 \times 10^{-}5$.

This problem was simulated for a time period of 200 seconds, with a time step size of $\Delta t = 0.01$. From the full model simulation, with a mesh of 12568 nodes, 2000 snapshots were obtained at equal time intervals $\Delta t = 0.1$ for each of the velocity components (u, v) and pressure p solution variables. 70% of the simulation data is used for training model. 10% of the simulation data is used to validate the model and 20% of the simulation data is used to test the model. In deep or machine learning, the more training data is used, the better prediction results can be expected. Different test cases may need different amount of training data. It is case dependent as it is a data-driven modelling. In this test case, we generated enough simulation data including the periodic system in order to get accurate solutions. The input number of time levels is 40 in this case.

Figure 6 shows the field of velocity solutions obtained from the full model, NL-NIROM with 2, 3 and 4 codes and L-NIROM with 4 POD basis functions at trained time level $t = 100s$ and predicted time level $t = 180s$. As shown in the figure, the solutions of NL-NIROM are closer to the high-fidelity full model when a larger number of codes are used. From these flow patterns it is shown that both the NL-NIROM and L-NIROM methods can capture main structural details of the solutions and the NL-NIROM performs better using as few as 2 codes. Additionally, the simulation outputs in pressure field of the NL-NIROM also appears to be in closer agreement to the full model solutions than that of L-NIROM. This issue is highlighted in the graphs presented the lift coefficient C_L and drag coefficient C_D over time in Figure 7 which show the L-NIROM predicted solutions have more deviations from the full model solution than NL-NIROM. The lift and drag coefficient are calculated by

$$C_L(t) = -\frac{2}{LV^2} \int_S (\nu \frac{\partial u_{tS}}{\partial n} n_x + p n_y) dS \tag{38}$$

$$C_D(t) = \frac{2}{LV^2} \int_S (\nu \frac{\partial u_{tS}}{\partial n} n_y - p n_x) dS \tag{39}$$

where u_{tS} is the tangential velocity component $[n_y, -n_x]^T$ at the surface S, defined as $u_{tS} = \boldsymbol{u} \cdot [n_y, -n_x]^T$.

The streamline of velocity solutions of full model, NL-NIROM with 4 codes and L-NIROM with 4 POD basis functions at predicted time level t = 180s are shown in Figure 8.

The errors considering all of the nodes in the computational domain is given in Figure 9. The graphs (a) and (b) in Figure 9 show correlation coefficients and RMSE errors of the NL-NIROM and POD based L-NIROM. Again, these exhibit that a noticeable improvement in accuracy is gained when using the Auto-Encoder network, whereby the errors are reduced in comparison to POD based L-NIROM method. The graphs also show that using larger number of codes results in improved accuracy. The errors between the two ROMs and the high-fidelity full solutions at two time levels t = 100s and t = 180s are presented in Figure 10, which shows that the errors decrease using the Auto-Encoder

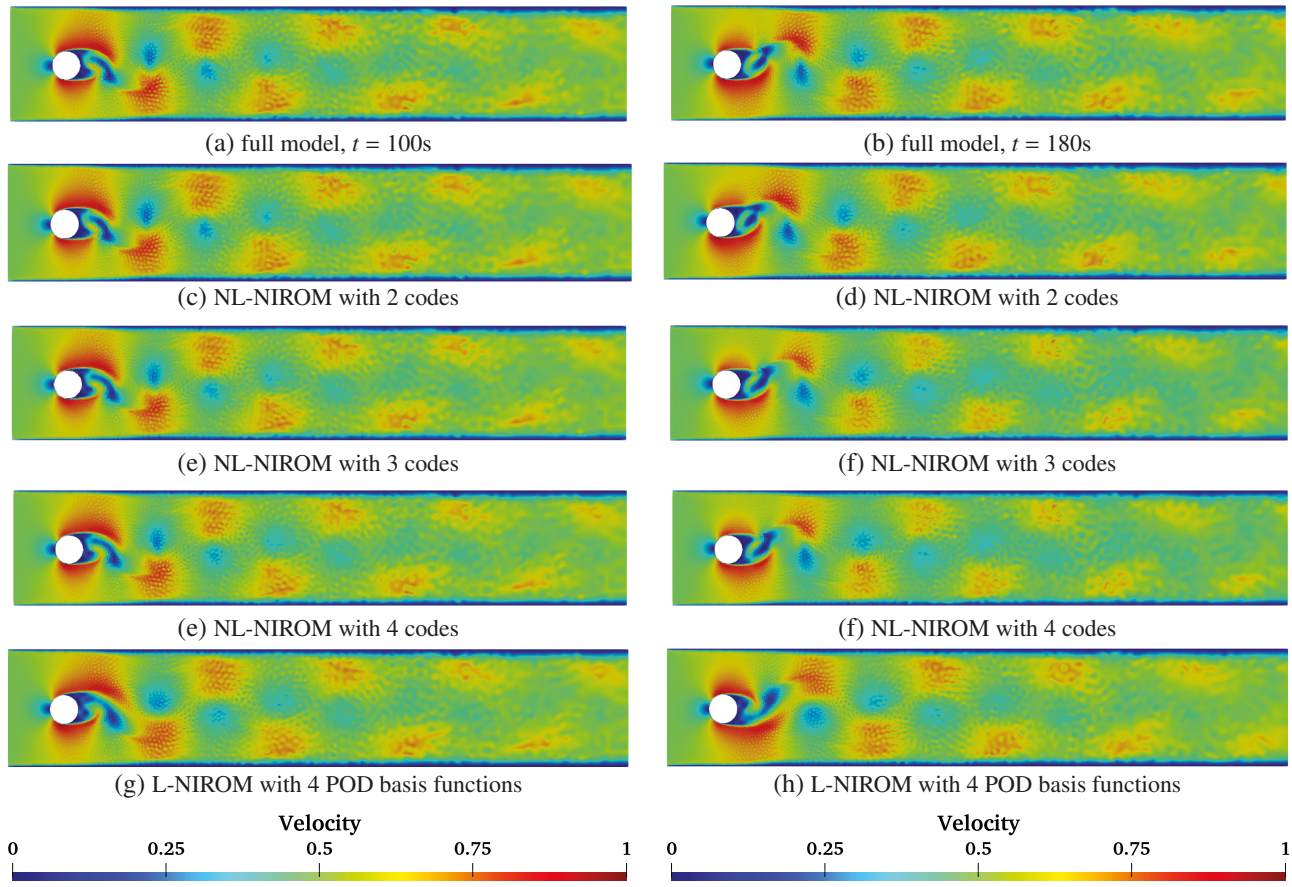
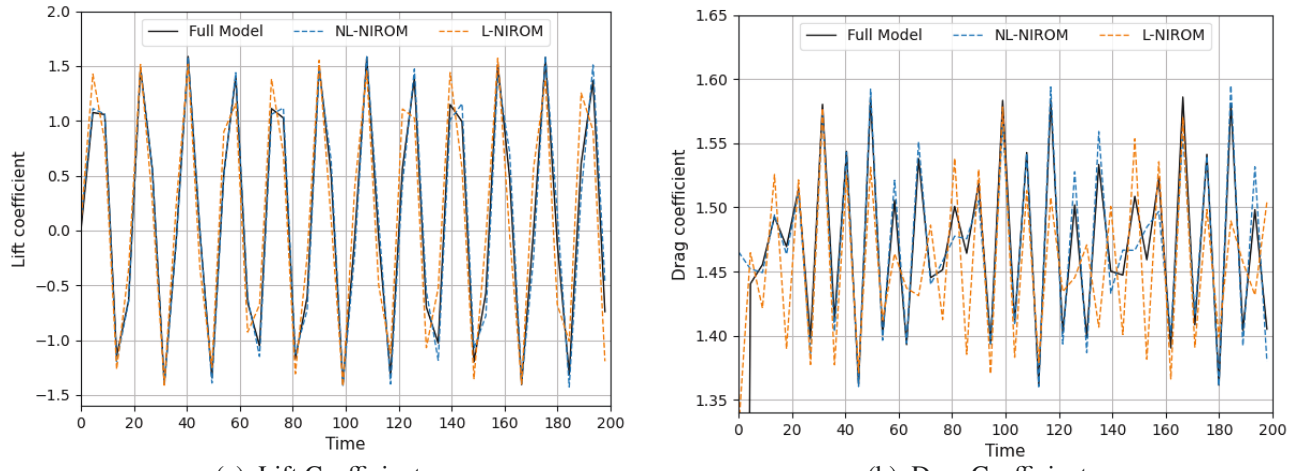
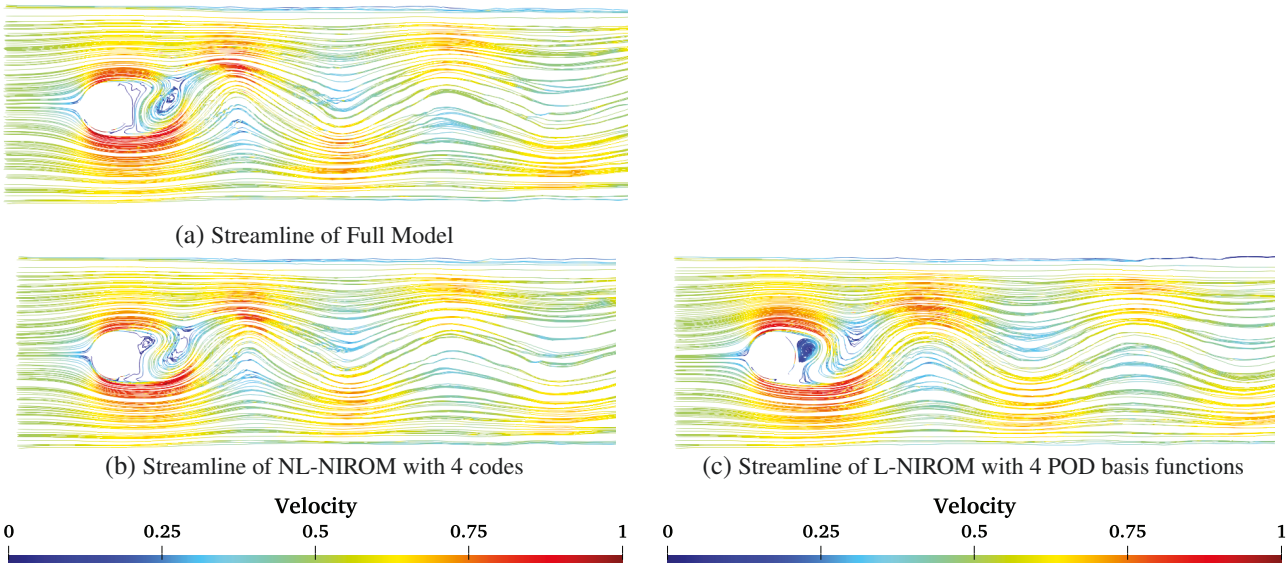


Figure 6: Case 1: flow past a cylinder. The graphs (a)-(h) show the field of velocity solutions obtained from the full model, NL-NIROM with 2, 3 and 4 codes and POD based L-NIROM at trained time level $t = 100s$ and predicted time level $t = 180s$.



(a) Lift Coefficient (b) Drag Coefficient

Figure 7: Case 1: Flow past a cylinder. Lift and drag coefficients over time are extracted from reconstructions of the full model, L-NIROM with 4 POD basis functions and NL-NIROM with 4 codes.



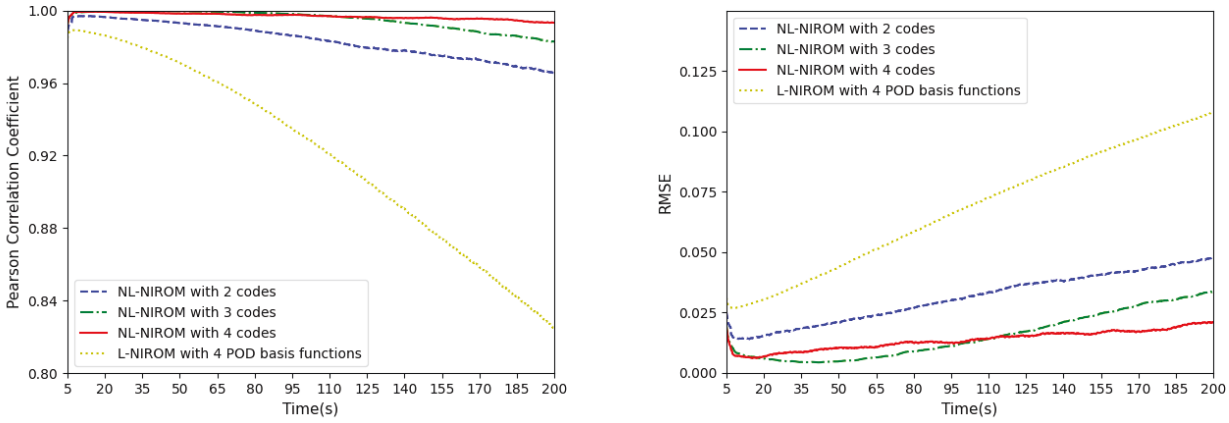
(a) Streamline of Full Model

(b) Streamline of NL-NIROM with 4 codes

(c) Streamline of L-NIROM with 4 POD basis functions

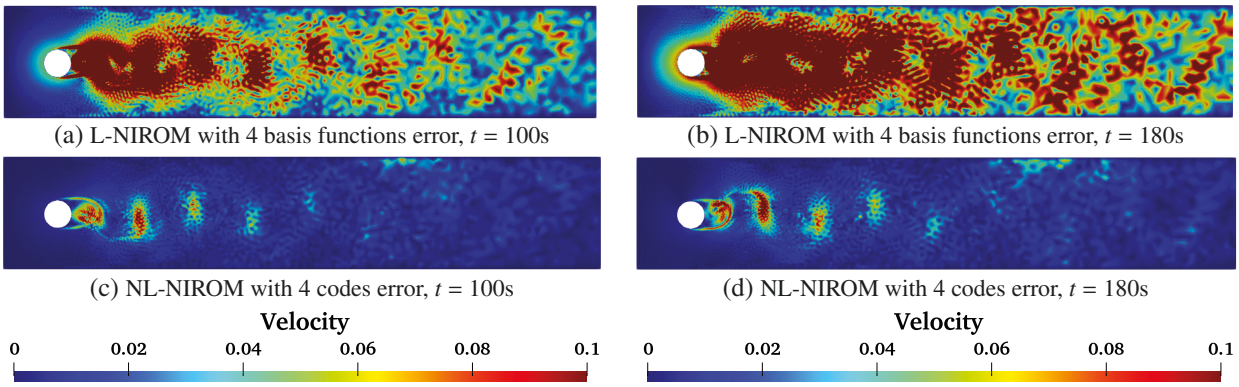
Figure 8: Case 1: flow past a cylinder. The streamline of velocity magnitude solutions of full model, NL-NIROM with 4 codes and L-NIROM with 4 POD basis functions at predicted time level t = 180s.

based NL-NIROM. The errors of the NL-NIROM are mainly from the dimensionality reduction and reconstruction of the flow field when the flow is stable. The Auto-Encoder based NL-NIROM is more accurate than that of POD based L-NIROM using identical dimensional size. We also compared NL-NIROM with POD based L-NIROM using larger number of dimensional size in the latent/reduced space. Figure 11 shows the relative error between the full model values and ROMs results against their dimension in the reduced latent space. The relative error is calculated by Equation 37. Compared with the POD based L-NIROM, NL-NIROMs display higher accuracy using less dimensional size in the reduced latent space. In addition, the errors reduces very fast in the first 12 codes, which means that the NL-NIROM is able to capture most of the energy of the full system using only 12 codes while L-NIROM needs more than 24 basis functions to capture the same energy.



(a) Correlation Coefficient (b) RMSE

Figure 9: Case 1: Flow past a cylinder. Correlation coefficient and the root-mean-square errors(RMSE) of velocity solutions calculated for Auto-Encoder based NL-NIROM with 2, 3, 4 codes and POD based L-NIROM with 4 basis functions.



(a) L-NIROM with 4 basis functions error, t = 100s (b) L-NIROM with 4 basis functions error, t = 180s

(c) NL-NIROM with 4 codes error, t = 100s (d) NL-NIROM with 4 codes error, t = 180s

Figure 10: Case 1: Flow past a cylinder. Velocity errors of flow past a cylinder problem at time levels 100 s (left) and 180 s (right). The solutions compare the error in POD based L-NIROM and Auto-Encoder based NL-NIROM. Both models are established using a dimensional size of 4.

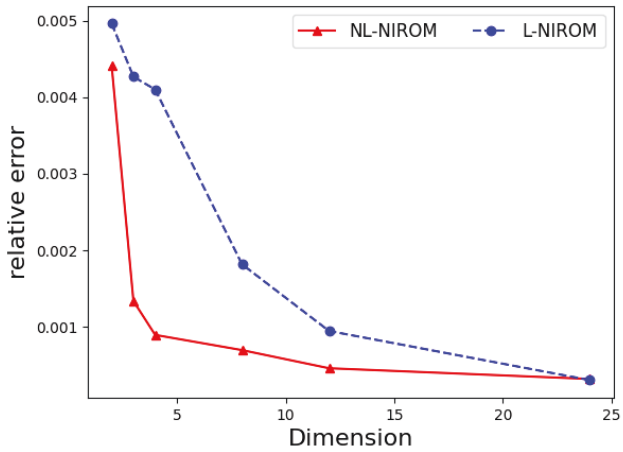


Figure 11: Case 1: Flow past a cylinder. The relative error of velocity between the full model and two NIROMs against their dimensional size in reduced space.

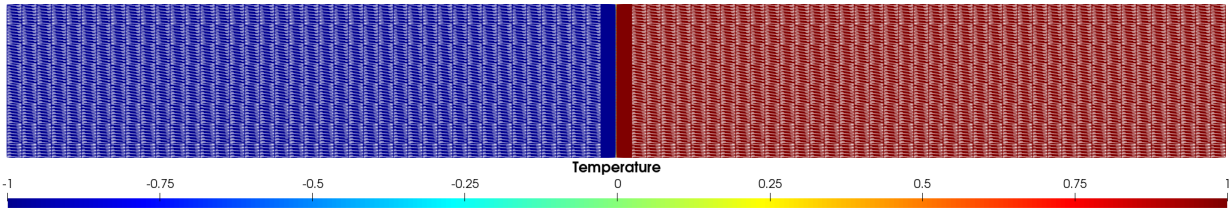


Figure 12: Case 2: Computational domain of a lock exchange case

5.2. *Case 2: Lock exchange*

In the second numerical example, a two dimensional lock exchange problem is demonstrated. The problem consists of cold and hot fluids with different densities, which are separated by a lock. The cold fluid is at the left and the hot fluid is at the right. Two gravity currents propagate along the tank after the lock is removed [69]. The computational domain is a non-dimensional rectangle with a size of 0.8×0.1, see Figure 12. The mesh has 13231 nodes. The initial non-dimensional temperatures are set to be $T = -1$ for the cold fluid and $T = 1$ for the hot fluid. The initial conditions for the pressure and velocity are set to be zero. The isotropic viscosity is 1×10^{-6}. The Crank–Nicolson method is used in the temporal discretisation. In this work, temperature field evolution is simulated via the Fluidity. From the full model simulation by Fluidity, 3200 snapshots were obtained at regularly spaced time intervals $\Delta t = 0.025$ for solution variables. The simulation period is 80s and 70% of the simulation data is used for training the attention based deep neural network in order to predict the fluid dynamics. 10% of the simulation data is used to validate the model. 20% of the simulation data is used to test the model. The input number of time levels (τ in Equation 33) is 60 in this case.

Figure 13 shows that the temperature solutions obtained from the full model, NL-NIROM with 8, 12, 18 and 24 codes and L-NIROM with 8, 12, 18 and 24 POD basis functions at time $t = 30$s and 60s respectively. As shown in the figure, both of the Auto-Encoder based NL-NIROM and POD based L-NIROM can capture dominant structural details of the numerical solutions.

In order to assess the predictive capability of the model, the predicted solutions at an unseen time level $t = 79$s are provided in Figure 14. The temperature solution profiles of these different models are close. In order to see clearly the differences, the temperature solution comparison at a particular nodal point $(x, y) = (0.3315, 0.042)$ in the computational domain is given in Figure 15. The reason why we chosen this particular point is because it contains the cold and hot fluids propagation information and the temperature exchanges actively at this point. The figure 16 shows the temperature solutions obtained from the full model, NL-NIROM with 8, 12, 18, 24 codes and L-NIROM with 8, 12, 18, 24 POD basis functions at that particular point $(x, y) = (0.3315, 0.042)$. In addition, the errors considering all points in the computational domain are given in Figure 17. These figures show that the NL-NIROM performs very well using as few as 8 codes. In addition, the temperature profile of the NL-NIROM appears to be in closer agreement to the full model solutions than that of POD based L-NIROM. Also, the NL-NIROM with larger number of codes exhibits more accurate solutions. This issue is highlighted in the graphs presented in Figure 17, which show correlation coefficient and the root-mean-square errors (RMSE) of temperature solutions calculated from NL-NIROM with 8,12,18,24 codes and L-NIROM with 8,12,18,24 POD basis functions. The correlation coefficient and RMSE consider errors of all of the points in the computational domain. The formulations of calculating correlation coefficient and RMSE are given in Equations 35 and 36 respectively. Figure 18 shows the residual errors between the high-fidelity full model and the different ROMs at time level t = 60s and t = 79s. As shown in the figure, the overall errors from L-NIROM with 24 POD basis functions are larger than those of Auto-Encoder based NL-NIROM with 8 codes. The graphs also show that using larger number of codes results in improved accuracy in Auto-Encoder based NL-NIROM. The relative error between the full model solutions and ROMs against their dimensionality size in the reduced space is shown in Figure 19. From the figure, we can see that the NL-NIROM has less error than the L-NIROM. The results of NL-NIROM with 8 codes almost have the same accuracy with the L-NIROM with 24 basis functions. The NL-NIROM (red line) converges much faster than the L-NIROM (blue line).

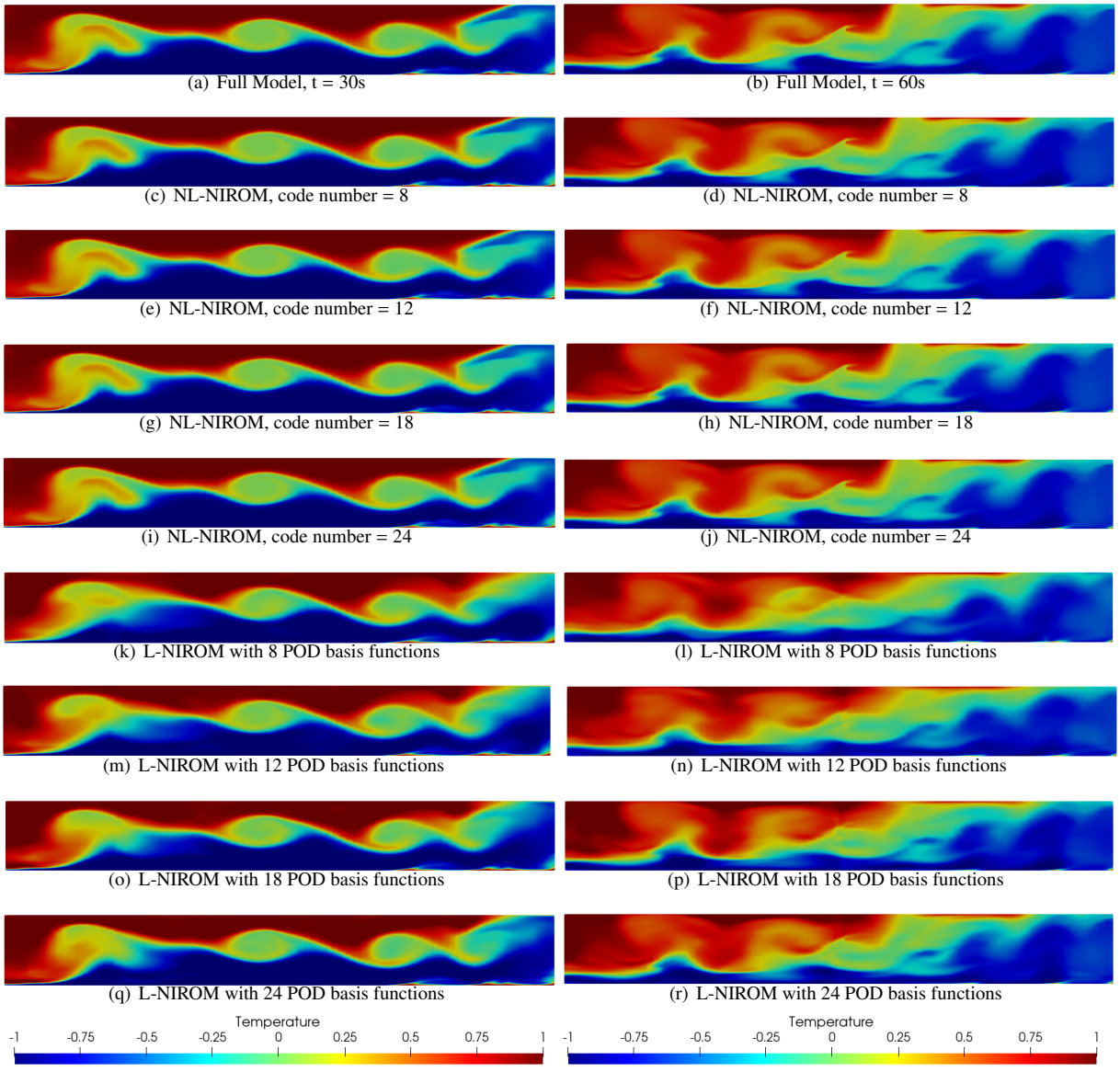
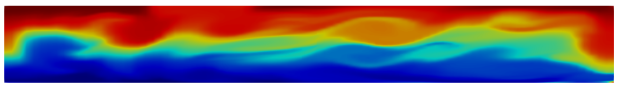
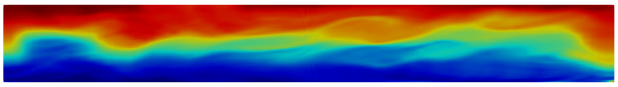


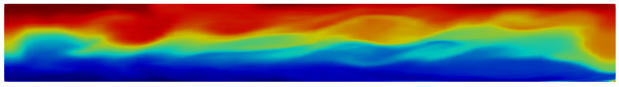
Figure 13: Case 2: Lock Exchange. The temperature solutions obtained at time t = 30s and 60s from the full model, NL-NIROM with 8,12,18 and 24 codes and L-NIROM with 8,12,18 and 24 POD basis functions respectively.



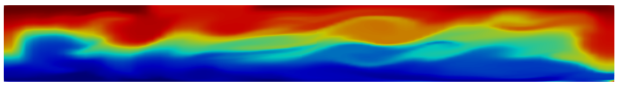
(a) Full Model, t = 79s



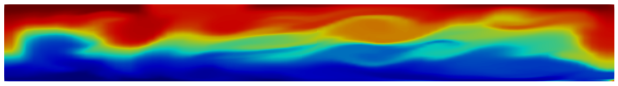
(b) NL-NIROM, code number = 8



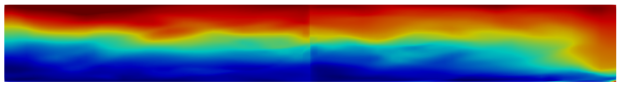
(c) NL-NIROM, code number = 12



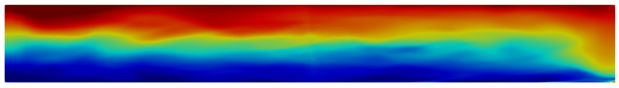
(d) NL-NIROM, code number = 18



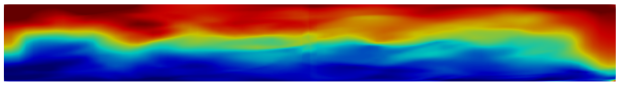
(e) NL-NIROM, code number = 24



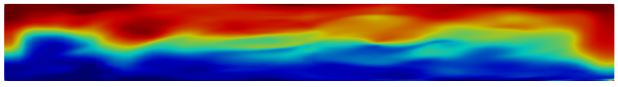
(f) L-NIROM with 8 POD basis functions



(g) L-NIROM with 12 POD basis functions



(h) L-NIROM with 18 POD basis functions



(i) L-NIROM with 24 POD basis functions

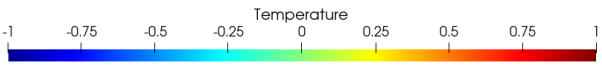


Figure 14: Case 2: lock exchange. The graphs (a) show the temperature solutions from the full model at predicted time level $t = 79s$, (b)-(i) show NL-NIROM with 8,12,18 and 24 codes and L-NIROM with 8,12,18 and 24 POD basis functions at the same time level

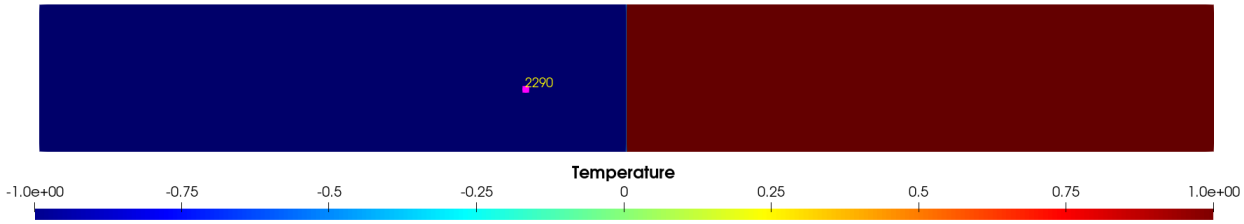


Figure 15: Case 2: lock exchange. Selected point $(x, y) = (0.4038, 0.095)$ in the computational domain

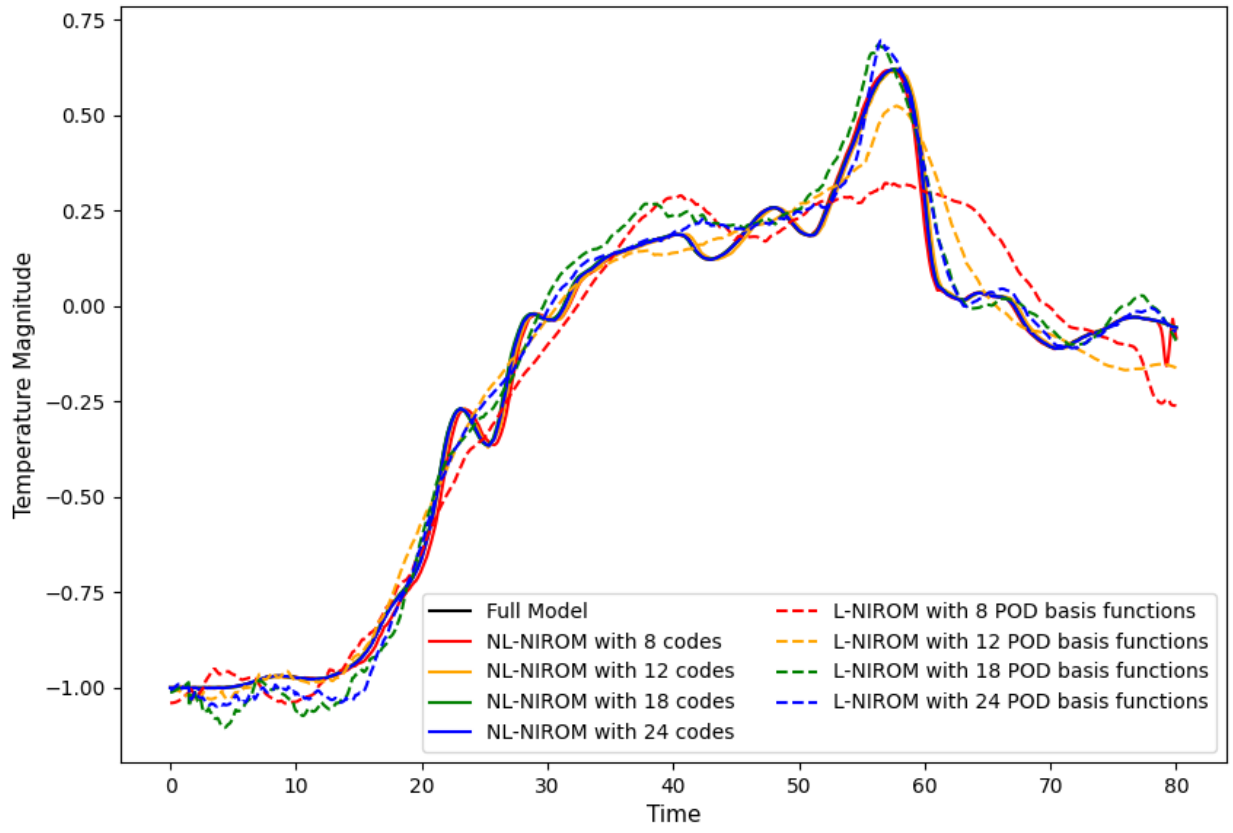
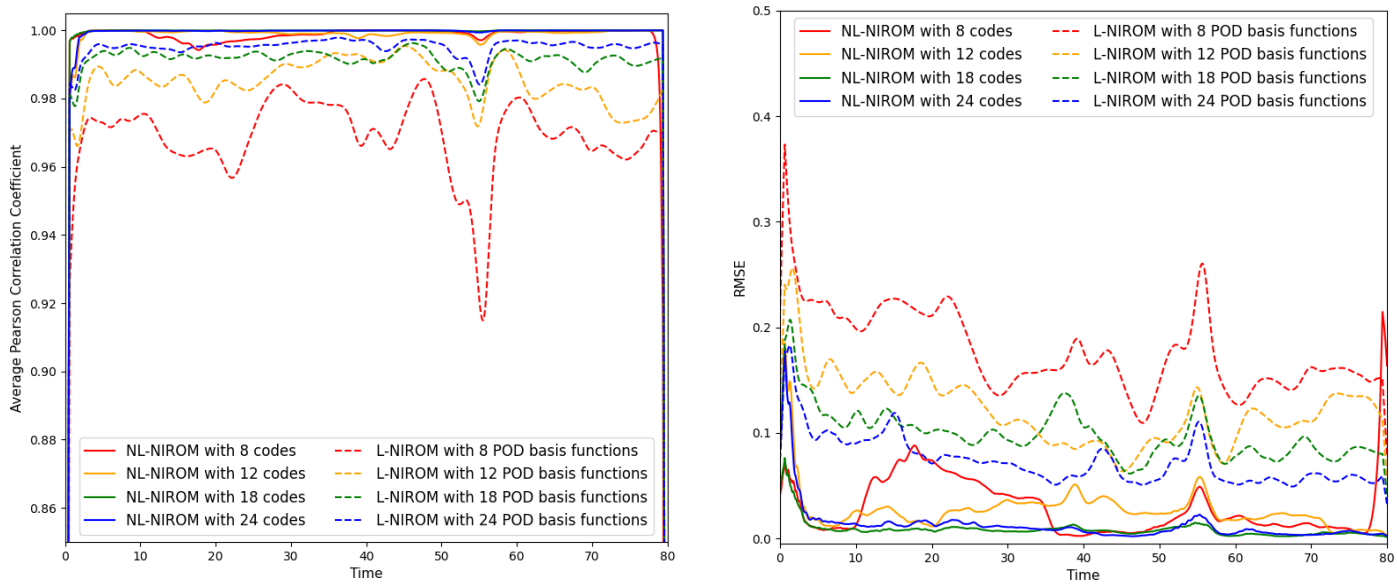
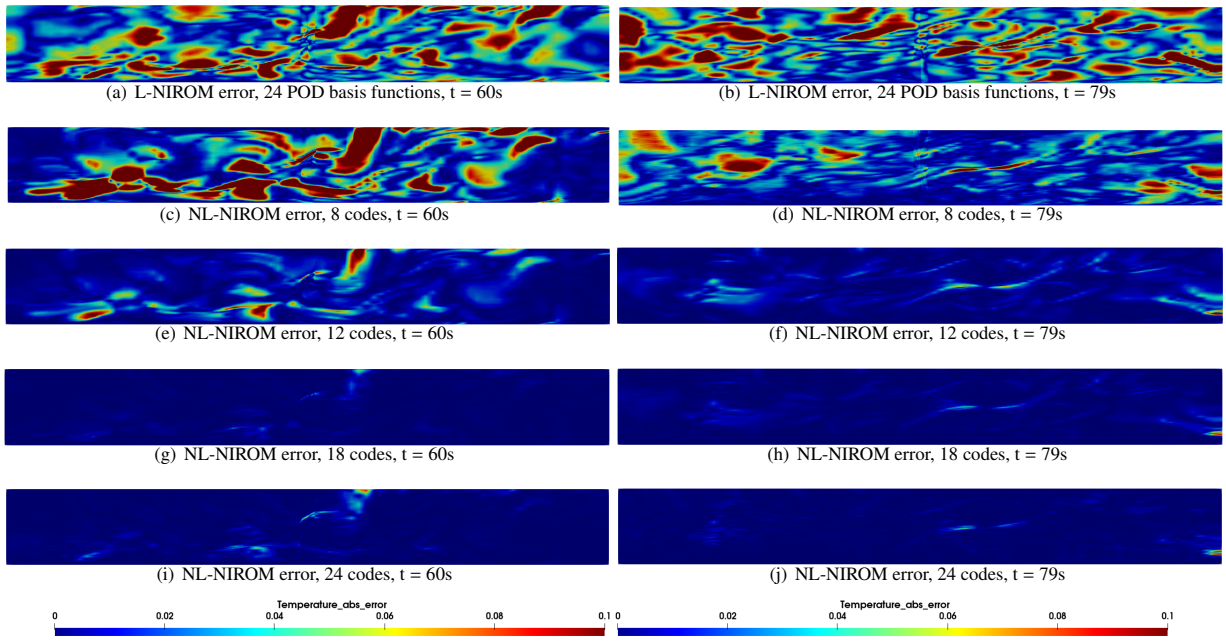


Figure 16: Case 2: lock exchange. The temperature solutions obtained from the full model, NL-NIROM with 8, 12, 18, 24 codes and L-NIROM with 8, 12, 18, 24 POD basis functions at that particular point.



(a) Correlation Coefficient (b) RMSE

Figure 17: Case 2: Lock exchange. Correlation coefficient and the root-mean-square errors(RMSE) of temperature solutions from Auto-Encoder based NL-NIROM with 8,12,18,24 codes and POD based L-NIROM with 8,12,18,24 basis functions.



(a) L-NIROM error, 24 POD basis functions, t = 60s

(b) L-NIROM error, 24 POD basis functions, t = 79s

(c) NL-NIROM error, 8 codes, t = 60s

(d) NL-NIROM error, 8 codes, t = 79s

(e) NL-NIROM error, 12 codes, t = 60s

(f) NL-NIROM error, 12 codes, t = 79s

(g) NL-NIROM error, 18 codes, t = 60s

(h) NL-NIROM error, 18 codes, t = 79s

(i) NL-NIROM error, 24 codes, t = 60s

(j) NL-NIROM error, 24 codes, t = 79s

Figure 18: Case 2: Lock Exchange. Temperature errors between full model and two different NIROMs at time levels 60s and 79s. The solutions compare the errors in L-NIROM with 24 POD basis functions(first row) and Auto-Encoder based NL-NIROM with 8,12,18 and 24 codes (following rows).

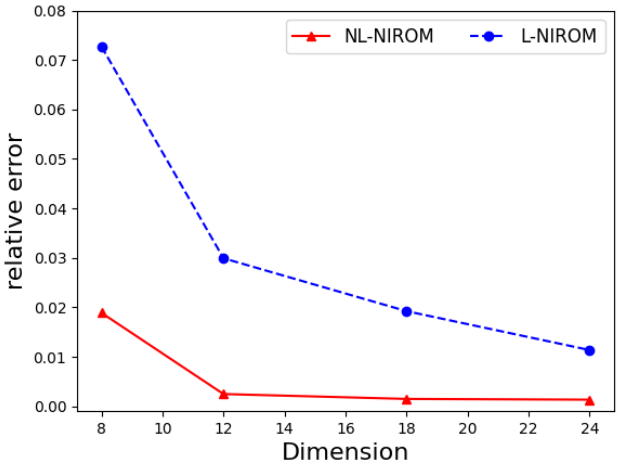


Figure 19: Case 2: lock exchange. The relative error of temperature between the full model and two NIROMs solutions against their dimensional size in the reduced space.

5.3. Computational efficiency

Table 1 shows the CPU cost (online cost and offline cost) required for running the high-fidelity full model and NL-NIROM for each time step. It is worth mentioning that the online CPU cost (dimensionless) required for predicting the NL-NIROM for one time step is only $3.0E-02$ s, while the high-fidelity full model for the flow past a cylinder case is 2.5 s. The computational cost of the high-fidelity full model will increase as the mesh size increases. For a complicated test case with larger number of nodes, the NL-NIROM can gain several orders of magnitude speed up. The training time of self-attention neural network is big, however, it is offline, which means it does not need to run again after is pre-computed. The training time of SAE neural network (dimensionality reduction stage) is comparatively less than that of self-attention stage. The reduction computational cost of training Auto-Encoder network is higher than that of obtaining POD basis functions. However, it has more or less the same time complexity with the POD stage, see the reduction cost column in the Table 1.

The full model simulations and NL-NIROM online predictions were performed on a workstation with an Intel 8 cores i7-9700 Processor (3.00GHz base frequency and 4.70GHz Max Turbo Frequency) and 24G RAM. The training of NL-NIROM were performed on Google's Colaboratory platform, which allows AI researchers to write and execute python scripts in the web browser. It also provides GPUs for deep learning training.

Table 1: CPU time required to run the full-fidelity full model, L-NIROM based on POD/Self-Attention(SE) and NL-NIROM based on AE/SE(s)

Cases	Models	Reduction*	Self-attention training**	Run	Nodes
Flow	full model	\	\	2.5	12568
past a	POD/SE(4)	8.7701	233.4754	$3.0387E-02$	12568
cylinder	AE/SE(2)	10.4528	186.3898	$3.0064E-02$	12568
	AE/SE(3)	10.4101	272.8074	$3.0296E-02$	12568
	AE/SE(4)	11.3381	218.3514	$3.0087E-02$	12568
Lock	full model	\	\	0.165	13231
exchange	POD/SE(24)	3.6774	774.8551	$3.2213E-03$	13231
	AE/SE(8)	12.6045	767.7650	$3.0513E-03$	13231
	AE/SE(12)	18.4910	745.5080	$3.0291E-03$	13231
	AE/SE(18)	27.7016	754.5975	$3.1131E-03$	13231
	AE/SE(24)	37.1502	789.9625	$3.1941E-03$	13231

* The training time of Auto-Encoder model for 5 epochs
** The training time of Self-attention model for 100 epochs

6. Conclusions

A new non-linear non-intrusive reduced order model (NL-NIROM) is presented in this work. This is achieved by constructing a new model reduction deep learning neural network architecture. The architecture consists of two main parts: Auto-Encoder and self-attention. The Auto-Encoder part is used to project the high-dimensional full space into a much lower dimensional space while the self-attention part involves constructing a number of functions representing the fluid dynamics in reduced space. The new NL-NIROM has been implemented under the framework of an advanced three dimensional finite element mesh fluid model (Fluidity). The performance of this NL-NIROM has been illustrated by two numerical problems: flow past a cylinder and a lock exchange test cases. A detailed comparison between the high-fidelity full model and two NIROMs has been made. The results show that Auto-Encoder based NL-NIROM performs better than POD based linear NIROM (L-NIROM). Significant CPU speed-up is achieved compared to the high-fidelity full model. The advantage of this NL-NIROM is that it is able to capture more non-linearity information than POD based L-NIROM.

The NL-NIROM will be applied into more complicated fluid problems in the future, such as large scale urban flows, flooding, multi-phase flow and ocean modelling. The predictive capability of NL-NIROM has strong connections with the amount of training data. Larger number of training data generally leads to higher accuracy, while it can be case dependent. Future work will also involve developing a parametric NL-NIROM capable of handling varying initial and boundary conditions. In addition, our future work will also combine this data-driven NL-NIROM with physical equations to build a hybrid reduced order model to make simulations when training data is not enough[70].

Acknowledgments

The authors would like to acknowledge the support of EPSRC grant: PURIFY ($EP/V000756/1$) and the Royal Society International Exchanges 2019 Cost Share ($IEC\backslash NSFC\backslash 191037$). The work is supported by the Fundamental Research Funds for the Central Universities. Rui Fu acknowledges College of Engineering Centenary PhD Scholarship at Swansea University. We acknowledge the support of the Supercomputing Wales project, which is part-funded by the European Regional Development Fund (ERDF) via Welsh Government.

References

[1] Saddam Hijazi, Giovanni Stabile, Andrea Mola, and Gianluigi Rozza. Data-driven POD-Galerkin reduced order model for turbulent flows. *Journal of Computational Physics*, 416:109513, 2020.

[2] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[3] PTM Vermeulen and AW Heemink. Model-reduced variational data assimilation. *Monthly weather review*, 134(10):2888–2899, 2006.

[4] Yanhua Cao, Jiang Zhu, Zhendong Luo, and Ionel M Navon. Reduced-order modeling of the upper tropical pacific ocean model using proper orthogonal decomposition. *Computers & mathematics with Applications*, 52(8-9):1373–1386, 2006.

[5] DN Daescu and IM Navon. A dual-weighted approach to order reduction in 4dvar data assimilation. *Monthly Weather Review*, 136(3):1026–1041, 2008.

[6] Răzvan Ştefănescu and Ionel Michael Navon. POD/DEIM nonlinear model order reduction of an adi implicit shallow water equations model. *Journal of Computational Physics*, 237:95–114, 2013.

[7] D. Xiao, F. Fang, J. Zheng, C.C. Pain, and I.M. Navon. Machine learning-based rapid response tools for regional air pollution modelling. *Atmospheric Environment*, 199:463–473, 2019.

[8] Peter Benner and Pawan Goyal. Interpolation-based model order reduction for polynomial systems. *SIAM Journal on Scientific Computing*, 43(1):A84–A108, 2021.

[9] Immanuel Martini, Bernard Haasdonk, and Gianluigi Rozza. Certified reduced basis approximation for the coupling of viscous and inviscid parametrized flow models. *Journal of Scientific Computing*, 74(1):197–219, 2018.

[10] Gabriele Boncoraglio, Charbel Farhat, and Charbel Bou-Mosleh. Model reduction framework with a new take on active subspaces for optimization problems with linearized fluid-structure interaction constraints. *International Journal for Numerical Methods in Engineering*, 122(19):5450–5481, 2021.

[11] Youngsoo Choi, Gabriele Boncoraglio, Spenser Anderson, David Amsallem, and Charbel Farhat. Gradient-based constrained optimization using a database of linear reduced-order models. *Journal of Computational Physics*, 423:109787, 2020.

[12] Benjamin Peherstorfer, Serkan Gugercin, and Karen Willcox. Data-driven reduced model construction with time-domain loewner models. *SIAM Journal on Scientific Computing*, 39(5):A2152–A2178, 2017.

[13] Alessandro Alla, Carmen Grässle, and Michael Hinze. A posteriori snapshot location for POD in optimal control of linear parabolic equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 52(5):1847–1873, 2018.

[14] Nguyen Thanh Son and Tatjana Stykel. Model order reduction of parameterized circuit equations based on interpolation. *Advances in Computational Mathematics*, 41(5):1321–1342, 2015.

[15] Christoph Hachtel, Johanna Kerler-Back, Andreas Bartel, Michael Günther, and Tatjana Stykel. Multirate dae/ode-simulation and model order reduction for coupled field-circuit systems. In *Scientific Computing in Electrical Engineering*, pages 91–100. Springer, 2018.

[16] Răzvan Ştefănescu, Bernd R Noack, and Adrian Sandu. Model reduction and inverse problems and data assimilation with geophysical applications. a special issue in honor of i. michael navon's 75th birthday, 2016.

[17] Xuping Xie, Muhammad Mohebujjaman, Leo G Rebholz, and Traian Iliescu. Data-driven filtered reduced order modeling of fluid flows. *SIAM Journal on Scientific Computing*, 40(3):B834–B857, 2018.

[18] Cong Xiao, Olwijn Leeuwenburgh, Hai Xiang Lin, and Arnold Heemink. Efficient estimation of space varying parameters in numerical models using non-intrusive subdomain reduced order modeling. *Journal of Computational Physics*, 424:109867, 2021.

[19] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The gnat method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[20] Sebastian Grimberg, Charbel Farhat, Radek Tezaur, and Charbel Bou-Mosleh. Mesh sampling and weighting for the hyperreduction of nonlinear petrov–galerkin reduced-order models with local reduced-order bases. *International Journal for Numerical Methods in Engineering*, 122(7):1846–1874, 2021.

[21] Michael Schlegel and Bernd R. Noack. On long-term boundedness of Galerkin models. *Journal of Fluid Mechanics*, 765:325–352, 2 2015.

[22] Jan Osth, Bernd R. Noack, Sinisa Krajnovic, Diogo Barros, and Jacques Boree. On the need for a nonlinear subscale turbulence term in POD models as exemplified for a high-Reynolds-number flow over an Ahmed body. *Journal of Fluid Mechanics*, 747:518–544, 5 2014.

[23] S. Chaturantabut and D.C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput*, 32:2737–2764, 2010.

[24] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for numerical methods in engineering*, 86(2):155–181, 2011.

[25] Karen Willcox and Alexandre Megretski. Model reduction for large-scale linear applications. *IFAC Proceedings Volumes*, 36(16):1387–1392, 2003.

[26] D Xiao, F Fang, J Du, CC Pain, IM Navon, AG Buchan, Ahmed H Elsheikh, and G Hu. Non-linear Petrov–Galerkin methods for reduced order modelling of the Navier–Stokes equations using a mixed finite element pair. *Computer Methods In Applied Mechanics and Engineering*, 255:147–157, 2013.

[27] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[28] Saifon Chaturantabut and Danny C Sorensen. Application of POD and DEIM on dimension reduction of non-linear miscible viscous fingering in porous media. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4):337–353, 2011.

[29] Saifon Chaturantabut and Danny C Sorensen. A state space error estimate for POD-DEIM nonlinear model reduction. *SIAM Journal on numerical analysis*, 50(1):46–63, 2012.

[30] Dunhui Xiao, Fangxin Fang, Andrew G Buchan, Christopher C Pain, Ionel Michael Navon, Juan Du, and G Hu. Non-linear model reduction for the Navier–Stokes equations using residual DEIM method. *Journal of Computational Physics*, 263:1–18, 2014.

[31] D Xiao, F Fang, AG Buchan, CC Pain, IM Navon, and A Muggeridge. Non-intrusive reduced order modelling of the Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 293:522–541, 2015.

[32] Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

[33] Sk M Rahman, Suraj Pawar, Omer San, Adil Rasheed, and Traian Iliescu. Nonintrusive reduced order modeling framework for quasigeostrophic turbulence. *Physical Review E*, 100(5):053306, 2019.

[34] Mariella Kast, Mengwu Guo, and Jan S Hesthaven. A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems. *Computer Methods in Applied Mechanics and Engineering*, 364:112947, 2020.

[35] Changhong Mou, Birgul Koc, Omer San, Leo G Rebholz, and Traian Iliescu. Data-driven variational multiscale reduced order models. *Computer Methods in Applied Mechanics and Engineering*, 373:113470, 2021.

[36] Dunhui Xiao, Fangxin Fang, Claire E Heaney, IM Navon, and CC Pain. A domain decomposition method for the non-intrusive reduced order modelling of fluid flow. *Computer Methods in Applied Mechanics and Engineering*, 354:307–330, 2019.

[37] Qian Wang, Jan S Hesthaven, and Deep Ray. Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *Journal of computational physics*, 384:289–307, 2019.

[38] Pierre Jacquier, Azzedine Abdedou, Vincent Delmas, and Azzeddine Soulaïmani. Non-intrusive reduced-order modeling using uncertainty-aware deep neural networks and proper orthogonal decomposition: Application to flood modeling. *Journal of Computational Physics*, 424:109854, 2021.

[39] Shady E Ahmed, Sk Mashfiqur Rahman, Omer San, Adil Rasheed, and Ionel M Navon. Memory embedded non-intrusive reduced order modeling of non-ergodic flows. *Physics of Fluids*, 31(12):126602, 2019.

[40] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[41] Toby RF Phillips, Claire E Heaney, Paul N Smith, and Christopher C Pain. An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. *International Journal for Numerical Methods in Engineering*, 122(15):3780–3811, 2021.

[42] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[43] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, 18(6):463–477, 2019.

[44] Piyush M Tagade, Shashishekar P Adiga, Shanthi Pandian, Min Sik Park, Krishnan S Hariharan, and Subramanya Mayya Kolake. Attribute driven inverse materials design using deep learning bayesian framework. *npj Computational Materials*, 5(1):1–14, 2019.

[45] Duc Tran, Hung Nguyen, Bang Tran, Carlo La Vecchia, Hung N Luu, and Tin Nguyen. Fast and precise single-cell data analysis using a hierarchical autoencoder. *Nature communications*, 12(1):1–10, 2021.

[46] Pin Wu, Siquan Gong, Kaikai Pan, Feng Qiu, Weibing Feng, and Christopher Pain. Reduced order model using convolutional auto-encoder with self-attention. *Physics of Fluids*, 33(7):077107, 2021.

[47] Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pages 71–82. Wiley Online Library, 2019.

[48] Teeratorn Kadeethum, Francesco Ballarin, Youngsoo Choi, Daniel O'Malley, Hongkyu Yoon, and Nikolaos Bouklas. Non-intrusive reduced order modeling of natural convection in porous media using convolutional autoencoders: comparison with linear subspace techniques. *Advances in Water Resources*, page 104098, 2022.

[49] Rambod Mojgani and Maciej Balajewicz. Low-rank registration based manifolds for convection-dominated pdes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 399–407, 2021.

[50] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

[51] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[52] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[53] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[54] Huy Phan, Huy Le Nguyen, Oliver Y Chén, Philipp Koch, Ngoc QK Duong, Ian McLoughlin, and Alfred Mertins. Self-attention generative adversarial network for speech enhancement. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7103–7107. IEEE, 2021.

[55] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[56] A Waswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, AN Gomez, L Kaiser, and I Polosukhin. Attention is all you need. In *NIPS*, 2017.

[57] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[58] D Xiao, F Fang, C Pain, and G Hu. Non-intrusive reduced-order modelling of the navier–stokes equations based on RBF interpolation. *International Journal for Numerical Methods in Fluids*, 79(11):580–595, 2015.

[59] Zheng Wang, Dunhui Xiao, Fangxin Fang, Rajesh Govindan, Christopher C Pain, and Yike Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

[60] Imperial College London AMCG. Fluidity manual v4.1.12, Apr 2015.

[61] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention mechanisms in computer vision: A survey. *Computational Visual Media*, pages 1–38, 2022.

[62] Qingchao Liu, Tao Liu, Yingfeng Cai, Xiaoxia Xiong, Haobin Jiang, Hai Wang, and Ziniu Hu. Explanatory prediction of traffic congestion propagation mode: A self-attention based approach. *Physica A: Statistical Mechanics and its Applications*, 573:125940, 2021.

[63] Yaojie Zhang, Bing Xu, and Tiejun Zhao. Convolutional multi-head self-attention on memory for aspect sentiment classification. *IEEE/CAA Journal of Automatica Sinica*, 7(4):1038–1044, 2020.

[64] Jiangyan Yi and Jianhua Tao. Self-attention based model for punctuation prediction using word and speech embeddings. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7270–7274. IEEE, 2019.

[65] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

[66] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[67] Dunhui Xiao, CE Heaney, L Mottet, F Fang, W Lin, IM Navon, Y Guo, OK Matar, AG Robins, and CC Pain. A reduced order model for turbulent flows in the urban environment using machine learning. *Building and Environment*, 148:323–337, 2019.

[68] CC Pain, MD Piggott, AJH Goddard, F Fang, GJ Gorman, DP Marshall, MD Eaton, PW Power, and CRE De Oliveira. Three-dimensional unstructured mesh ocean modelling. *Ocean Modelling*, 10(1-2):5–33, 2005.

[69] JO Shin, SB Dalziel, and PF Linden. Gravity currents produced by lock exchange. *Journal of Fluid Mechanics*, 521:1–34, 2004.

[70] Jinlong Fu, Dunhui Xiao, Rui Fu, Chenfeng Li, Chuanhua Zhu, Rossella Arcucci, and Ionel M Navon. Physics-data combined machine learning for parametric reduced-order modelling of nonlinear dynamical systems in small-data regimes. *Computer Methods in Applied Mechanics and Engineering*, 404:115771, 2023.

A non-linear non-intrusive reduced order model of fluid flow by Auto-Encoder and self-attention deep learning methods

Fu, Rui

2023-04-03

Fu R, Xiao D, Navon IM, et al., (2023) A non-linear non-intrusive reduced order model of fluid flow by Auto-Encoder and self-attention deep learning methods. International Journal for Numerical Methods in Engineering, Volume 124, Issue 13, July 2023, pp. 3087-3111
https://doi.org/10.1002/nme.7240