PROCEEDINGS A

royalsocietypublishing.org/journal/rspa

Research

Cite this article: Karbasian HR, van Rees WM. 2025 A parametric LSTM neural network for predicting flow field dynamics across a design space. *Proc. R. Soc. A* **481**: 20240055. https://doi.org/10.1098/rspa.2024.0055

Received: 22 January 2024
Accepted: 8 November 2024

Subject Category:
Engineering

Subject Areas:
fluid dynamics, artificial intelligence, computational physics

Keywords:
LSTM, flapping fin hydrodynamics, reduced order model, deep learning

Author for correspondence:
Wim M. van Rees
e-mail: wvanrees@mit.edu

A parametric LSTM neural network for predicting flow field dynamics across a design space

Hamid R. Karbasian[1,2] and Wim M. van Rees[1]

[1]Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA
[2]Department of Mechanical Engineering, Lyle School of Engineering, Southern Methodist University, 3101 Dyer Street, Dallas, TX 75275, USA

WMvR, 0000-0001-6485-4804

We develop a data-driven reduced-order model (ROM) to robustly predict the dynamics of fluid flows across a parametric design space. Our approach extends a long–short-term memory (LSTM) neural network with a new design gate, which enables the network to distinguish dynamic patterns associated with different design parameters. We first compare this parametric LSTM (pLSTM) with traditional LSTMs trained on a Van der Pol oscillator, where the design parameter is the nonlinear damping coefficient. The results show that pLSTM can provide accurate and robust predictions, whereas LSTMs fail to predict the correct dynamics when evaluated outside the immediate training data. Next, we use the pLSTM to predict the two-dimensional incompressible flow past a heaving and pitching ellipse. The pLSTM is trained with simulated flow field data compressed to a latent space, here defined through a proper orthogonal decomposition. The pLSTM can successfully predict the long-time dynamics of the flow field for unseen heaving/pitching kinematic parameters, while showing exceptional robustness to noise in the initial states. Taken together, the proposed pLSTM approach offers a three-aspect ROM approach (space, time and design space) to benefit prediction, optimization and control problems across parametric flow regimes.

1. Introduction

The ability to efficiently predict unsteady flow fields is imperative for exploration, control and performance optimization within the design space of a given problem

THE ROYAL SOCIETY PUBLISHING

[1]. For instance, in biologically inspired underwater propulsion, the use of optimization and learning has led to new insights into highly performant solutions as well as the fundamental physical processes [2–6]. Such studies rely on large amounts of numerical simulations or, in some cases, experimental runs [7–9], each one of which requires a spatio-temporal evolution of the flow field associated with a different set of design parameters. Data-driven reduced-order models (ROMs) pose an attractive pathway to capture low-dimensional patterns and system dynamics based on a small set of high-fidelity training data [10–13]. If successful, the ROMs can then be used in a design or optimization pipeline for low-cost, high-throughput prediction of flow results.

Model order reduction techniques for developing ROMs are typically categorized as projection based and data driven [12]. Among the projection-based ROMs, the Galerkin approach is a popular choice to convert the Navier–Stokes equations into a set of low-dimensional linear ordinary differential equations (ODEs) [14–16]. However, these models can encounter inaccuracies and instabilities, especially in nonlinear fluid physics [12,17]. To address this challenge, the least-squares Galerkin projection (LSGP) [16,18] has been proposed as an enhanced version of the Galerkin method, offering stability in solving ODEs for nonlinear fluid problems. Despite its enhanced stability, LSGP necessitates computing Jacobian matrices and solving Navier–Stokes equations during the ROM evaluation, leading to increased computational cost [16, 18]. For data-driven ROMs, deep learning algorithms have emerged as state-of-the-art tools in recent years [19]. Approaches using deep neural networks (DNNs), such as those employing sparse coding [20,21] and network-theoretic methods [22], have attracted attention for building nonlinear ROMs that are suitable for modelling nonlinear fluid flow problems [1,12,23].

DNN-based ROMs exist for time-averaged or steady-state predictions of fluid flow fields (here denoted one-aspect modelling in space, or $1A^{space}$), temporal prediction of scalar quantities such as hydrodynamic forces ($1A^{time}$) or spatio-temporal prediction of entire flow fields (two-aspect modelling, or 2A). In the category of $1A^{space}$, techniques like proper orthogonal decomposition (POD) [24], dynamic mode decomposition [25] and spectral POD [26] spatially decompose flow fields, utilizing their outputs in point-to-point feed-forward neural networks for data decoding. Recent advancements include various DNN models, like convolutional neural network autoencoders [27,28], sparse convolutional autoencoders [29], hierarchical autoencoders [30] and variational autoencoders [31], which extract fluid flow features to reconstruct high-dimensional flow fields. However, these $1A^{space}$ models have DNN architectures that are suitable for non-consecutive predictions, hindering their efficiency in addressing unsteady flows, such as vortex dynamics and wake structures behind bluff bodies. For sequential prediction of temporal solutions, these feed-forward networks lose efficacy, confined to point-to-point predictions, leading to neglect of input data's temporal dependencies and inadequate learning of dynamics [32].

In the category of $1A^{time}$, recurrent neural networks (RNNs) [33] are DNN models employed for sequential data analysis, particularly efficient in learning system dynamics due to their feedback mechanism [32]. The long–short-term memory (LSTM) network, a popular RNN variant, effectively mitigates the diminishing gradient problem often encountered in RNNs [34,35]. However, LSTMs are mainly suitable for low-dimensional sequential data. Two-aspect models, combining RNNs with the above-mentioned $1A^{space}$ models, have been developed to construct ROMs effectively predicting spatio-temporal dynamics for complex systems. With such 2A approaches, researchers have successfully used LSTMs for building ROMs for complex systems such as turbulence [19,34,36], unsteady wind turbine wakes [37], classifying vortex dynamics [33] and unsteady flow around various bluff bodies [38]. Moreover, regarding the dynamical discovery of nonlinear systems, LSTM networks were used in the main structure of ROMs as a nonlinear time-stepper [39]. The LSTM was also utilized as equation-free flux reconstructions of dynamical systems in extreme events [40]. Figure 1 illustrates the schematic representation of ROM frameworks based on the 1A and 2A models. In $1A^{time}$, a low-dimensional matrix of state is given to the LSTM and this LSTM can predict future solutions. In $1A^{space}$, where solutions remain unchanged over time (i.e. steady state), a matrix of high-dimensional states contains various

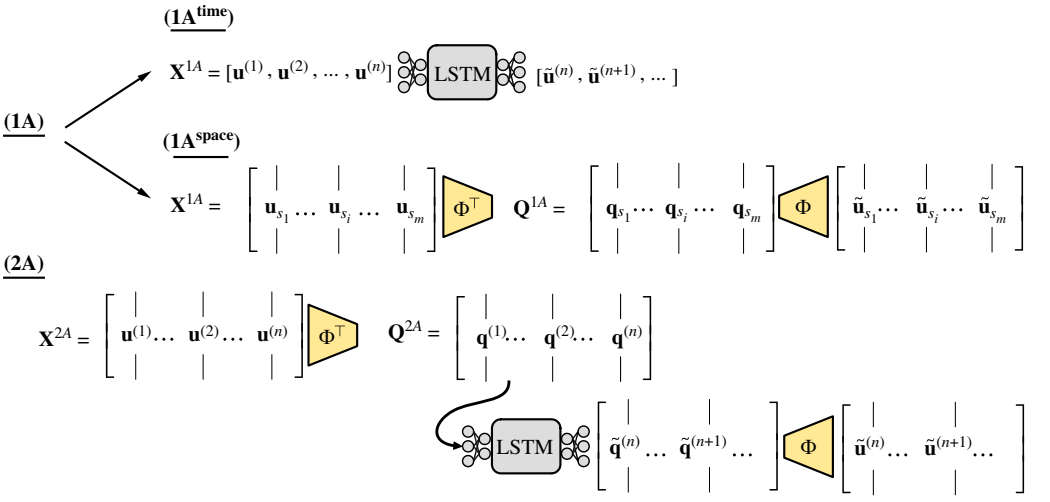

Figure 1. Schematic of ROM frameworks for 1A and 2A modelling using conventional LSTM. Here, **u**, **q** are vectors of the state and latent solution, respectively. Furthermore, $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{u}}$ are vectors of the predicted latent solution and decoded state, respectively. Superscript and subscript indicate the time interval and vector of the design parameter, s_i, respectively. The state and latent solutions are also stored in matrices **X** and **Q**, respectively. The autoencoder is represented by Φ and the temporal predictions are done via the LSTM model. Further details can be found in §2.

design parameters and an autoencoder can project high-dimensional states to a low-dimensional latent space and vice versa. For a high-dimensional unsteady problem, the 2A framework contains both an autoencoder and an LSTM model to predict latent solutions in the future. The predicted solutions are ultimately lifted back onto the high-dimensional space.

Most ROMs built based on 1A or 2A modelling encounter difficulties in learning prominent physics when predicting solutions across a wide range of design parameters. A naive merging of the design space with the latent space causes these models to learn all dynamical behaviours within the same latent space, often resulting in inaccurate predictions. To resolve this limitation, we propose a three-aspect (3A) modelling approach, enabling simultaneous learning of spatio-temporal flow fields across a design space. This method is particularly well suited for the optimization and control of high-dimensional nonlinear flow problems. Our proposed novel DNN architecture integrates a design gate within the LSTM, effectively increasing its learning memory capacity. This architecture functions as a versatile system capable of learning attractors (dynamical patterns) across various design parameters within the design space.

The rest of this article is structured as follows: §2 focuses on the mathematical explanations of a dynamical system, dimensionality reduction, ROM development and its associated DNN architectures. In §3, the pLSTM is proposed and explained as a new variant of the LSTM. In §4, we compare the predictions of the conventional and proposed variant of LSTM models for a low-dimensional problem, the Van der Pol oscillator, to show model performances for different linear and nonlinear dynamics. Subsequently, we pivot towards a high-dimensional Navier–Stokes problem and §5 provides insights into the flow solver used to construct a high-dimensional dataset and outlines the training details and framework architecture for ROM development. The comparative analysis between the proposed framework and conventional methods is presented in §6. Section 7 demonstrates the performance and robustness of the proposed ROM concerning augmented data in both initial conditions (ICs) and real-time modelling. The investigation of the high-dimensional results predicted by the proposed ROM and their validation against the ground truth CFD results is covered in §8. Finally, §9 summarizes the contents, concludes the study and discusses potential challenges and future avenues of work.

2. Reduced-order modelling

We consider a semi-discrete dynamical system with discrete state vector $\mathbf{u} \in \mathbb{R}^{N_u}$ evolving according to the full-order model

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, t), \quad \mathbf{u}_0 = \mathbf{u}(t = 0), \tag{2.1}$$

where $t \in \mathbb{T}$ is time in the time domain $\mathbb{T}$, $\mathbf{u}_0$ represents the IC at $t = 0$ and $\mathbf{f} : \mathbb{R}^{N_u} \to \mathbb{R}^{N_u}$ represents the discrete flux function. To derive a ROM, the main assumption is that the dynamics of equation (2.1) in the high-dimensional physical space can be reconstructed in a low-dimensional space (i.e. latent space). The latent space is spanned by basis functions containing spatial patterns. The projection of the state $\mathbf{u}$ onto the latent space yields the temporal evolution of latent solutions that contain information representing the dynamics of the problem. This information can be used to reconstruct the dynamics of the system in the latent space, identify the key features and build a ROM. Based on equation (2.1), the semi-discrete form of a ROM can be written as

$$\mathcal{R}_f(\mathbf{q}, t) = \frac{d\mathbf{q}}{dt} - \mathcal{F}(\mathbf{q}, t), \tag{2.2}$$

where $\mathbf{q} \in \mathbb{R}^r$ is the vector of the latent solution with a dimension of $r \in \mathbb{Z}$ indicating the rank of the basis functions (i.e. the number of solutions in the latent space). Also, $\mathcal{R}_f \in \mathbb{R}^r$ represents the residual of the ROM, and $\mathcal{F} : \mathbb{R}^r \to \mathbb{R}^r$ is the flux function that drives the ROM. The task in ROM is to find an appropriate function for $\mathcal{F}$, such that $\mathcal{R}_f$ remains minimum for all time in $\mathbb{T}$. This continuous form of the ROM is usually used in projection-based approaches, such as Galerkin projection [15,41] and LSGP [16,18]. For data-driven DNN models as considered in this study, instead the ROM is represented in a fully discrete form. Below, we will first describe our approach to obtain the basis of the latent space and subsequently proceed to the fully discrete dynamics of the system in latent space.

(a) Building basis functions using POD

The high-dimensional data from physical space $\mathbb{P}$ are decomposed and projected onto a low-dimensional latent space $\mathbb{H}$ (i.e. Hilbert space) through the basis functions. In this study, the basis functions are built by computing a POD of a discrete numerical solution to the system. Denoting by n the total number of time steps in the numerical solution and recalling that N_u is the number of degrees of freedom in the state vector, we construct the matrix $\mathbf{X} \in \mathbb{R}^{N_u \times n}$. Each column of this matrix is the vector of state across all spatial degrees of freedom at a single time step, so that we can write

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^{(1)} - \bar{\mathbf{u}} & \mathbf{u}^{(2)} - \bar{\mathbf{u}} & \dots & \mathbf{u}^{(n)} - \bar{\mathbf{u}} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{N_u \times n}, \quad \bar{\mathbf{u}} = \frac{1}{N_u} \sum_{i=1}^{n} \mathbf{u}^{(i)}, \tag{2.3}$$

where $\bar{\mathbf{u}} \in \mathbb{R}^{N_u}$ represents the temporal mean value of the state vector. Subsequently, we build the POD modes using

$$\Phi = \mathbf{POD}(\mathbf{X}), \quad \text{and} \quad \mathbf{X} \approx \Phi\mathbf{Q}, \tag{2.4}$$

where the **POD** operator extracts the orthonormal basis functions (i.e. POD modes) denoted by $\Phi \in \mathbb{R}^{N_u \times r}$, such that $\Phi^\top \Phi = \mathbf{I}$. Further, r is the number of POD modes considered so that r is equal to the dimension of the solutions in the latent space. Additionally, $\mathbf{Q} \in \mathbb{R}^{r \times n}$ denotes the latent solutions that contain temporal information representing the dynamics of the system in the latent

space, defined as

$$\mathbf{Q} = \begin{bmatrix} | & | & & | \\ \mathbf{q}^{(1)} & \mathbf{q}^{(2)} & \dots & \mathbf{q}^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{r\times n}. \tag{2.5}$$

The state vector at any time step i is then approximated by $\mathbf{u}^{(i)} \approx \boldsymbol{\Phi}\mathbf{q}^{(i)} + \bar{\mathbf{u}}$, so that we can define the residual $\mathbf{r}^{(i)} \in \mathbb{R}^{N_u}$ as

$$\mathbf{r}^{(i)} = \mathbf{u}^{(i)} - \bar{\mathbf{u}} - \boldsymbol{\Phi}\mathbf{q}^{(i)}. \tag{2.6}$$

This residual represents the maximum accuracy of the state reconstruction that can be achieved during the development of a ROM using a latent space of r dimensions. In the next section, we will discuss the development of a ROM for the dynamics of the system in the latent space.

(b) Dynamical system in the latent space

Since the aim of this study is to develop a data-driven ROM, we convert the semi-discrete form of the ROM in equation (2.2) to a fully discrete form. Denoting the latent solution at the ith time step (i.e. $i\Delta t$) as $\mathbf{q}^{(i)}$, we write the fully discrete form of the ROM as

$$\mathcal{R}_q^{(i)} = \mathbf{q}^{(i)} - \mathcal{G}^{(i)}\left(\tilde{\mathbf{q}}^{(i-n_p)}, \dots, \tilde{\mathbf{q}}^{(i-2)}, \tilde{\mathbf{q}}^{(i-1)}\right), \tag{2.7}$$

where $\tilde{\mathbf{q}} \in \mathbb{R}^r$ is the predicted latent solution by the ROM. Furthermore, $\mathcal{G}^{(i)} : \mathbb{R}^{r\times n_p} \to \mathbb{R}^r$ is the DNN mapping function that advances the latent solution to the next time value i and n_p is the number of sequential time steps that we consider as input data for the DNN model. Finally, $\mathcal{R}_q^{(i)} \in \mathbb{R}^r$ is the residual vector of the ROM at time step i. Note that the performance of mapping function $\mathcal{G}^{(i)}$ is obtained by the matrix of learning coefficients and the vector of bias, both obtained during DNN training. Here, this mapping function is developed by LSTM architecture elaborated in the next subsection.

(c) Long–short-term memory

LSTM architecture includes forget and input gates that resolve the main issues in RNNs [35,42,43] (see §1). Here, we will briefly recall conventional LSTM architecture, followed by a discussion of some of the shortcomings of LSTMs for parametric design in the next section.

The conventional LSTM cell receives the latent solution $\tilde{\mathbf{q}}(t-\Delta t)$, cell state $\mathbf{c}(t-\Delta t)$ and the output of the previous LSTM cell, $\mathbf{h}(t-\Delta t)$. These inputs are given to neural networks as follows

$$\begin{aligned} I(t) &= \sigma\left(\tilde{\mathbf{q}}(t-\Delta t)\mathbf{U}_i + \mathbf{h}(t-\Delta t)\mathbf{W}_i + \mathbf{b}_i\right), \\ F(t) &= \sigma\left(\tilde{\mathbf{q}}(t-\Delta t)\mathbf{U}_f + \mathbf{h}(t-\Delta t)\mathbf{W}_f + \mathbf{b}_f\right), \\ O(t) &= \sigma\left(\tilde{\mathbf{q}}(t-\Delta t)\mathbf{U}_o + \mathbf{h}(t-\Delta t)\mathbf{W}_o + \mathbf{b}_o\right), \end{aligned} \tag{2.8}$$

where $I(t)$, $F(t)$ and $O(t)$ are input, forget and output gates, respectively. Moreover, σ is the sigmoid function, $\mathbf{U}$ and $\mathbf{W}$ are matrices of learning coefficients and $\mathbf{b}$ is the vector of bias. Note that the subscripts of matrices and vectors denote to what gate they belong. The candidate to update the cell state, $\hat{C}(t)$, at present instant is computed as

$$\hat{C}(t) = \tanh\left(\tilde{\mathbf{q}}(t-\Delta t)\mathbf{U}_c + \mathbf{h}(t-\Delta t)\mathbf{W}_c + \mathbf{b}_c\right). \tag{2.9}$$

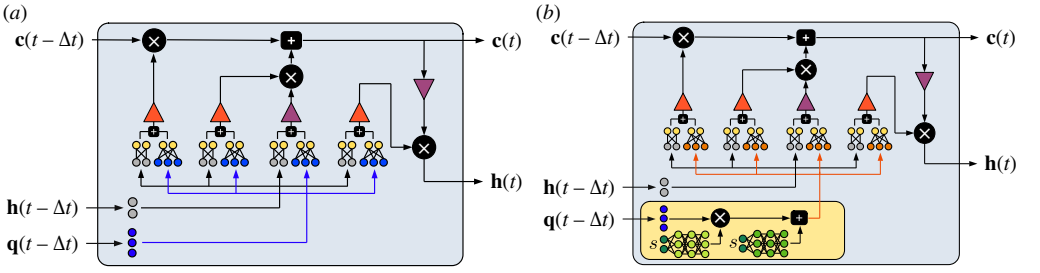

Figure 2. Architecture of (*a*) conventional and (*b*) parametric LSTM cells. The triangles are activation functions (red: sigmoid and purple: tanh). Note that the tanh activation function can be replaced by any other activation function based on the applications.

Subsequently, the cell state and the output values from the LSTM cell are computed as

$$\begin{aligned}\mathbf{c}(t) &= F(t) \times \mathbf{c}(t - \Delta t) + I(t) \times \hat{C}(t),\\ \mathbf{h}(t) &= \tanh\Big(\mathbf{c}(t)\Big) \times O(t),\end{aligned} \tag{2.10}$$

where $\times$ is the element-wise product between matrices. Figure 2*a* depicts the architecture of a conventional LSTM cell and its relative connections. Note that for n_p sequential data given to the input of the LSTM model, we need to connect n_p LSTM cells in series, and the final $\mathbf{h}(t)$ is projected onto a dense fully connected output layer to obtain $\tilde{\mathbf{q}}(t)$.

(d) Shortcomings of LSTM for parametric design

According to figure 2*a*, the conventional LSTM cell learns based on the inputs $\tilde{\mathbf{q}}(t - \Delta t)$, $\mathbf{h}(t - \Delta t)$ and $\mathbf{c}(t - \Delta t)$. There is no pattern recognition or classification gate in the conventional LSTM that can change dynamics based on differences in design parameters. Therefore, if the conventional LSTM model is trained for different case studies with different design parameters, the only adjustable parameter to change the dynamics of the problem to another one is the sequential latent solutions from previous time intervals. This is also visible in the formulation of equation (2.7), where the design space has no clear and independent identity in the LSTM architecture, and design parameters often appear as a ghost in the latent space. It is known that while LSTMs are designed to capture long-term dependencies, they may still struggle with noisy or ambiguous data [44], leading to instability of the model with error accumulation in the sequential solutions. This sensitivity to small perturbations, combined with the lack of any mechanism to constrain the latent-space dynamics to a specific design parameter, means that conventional LSTM is ill-suited to predict solutions across an entire design space.

3. Parametric LSTM

If the latent solutions in the ROM depend on a vector of design parameters $s \in \mathbb{R}^{N_s}$ with a dimension of N_s, then the discrete state vector and discrete latent solution can be written as $\mathbf{u}(t; s)$ and $\mathbf{q}^{(i)}(s)$, respectively. To add the design space as another aspect in the ROM, we propose to develop a design network that correlates the design parameters s with the attractors existing in the latent space $\mathbb{H}$. This design network is generically defined as follows

$$\mathcal{S} = \mathcal{D}(s), \tag{3.1}$$

where $\mathcal{S} \in \mathbb{R}^{r \times n_p}$ is the output of the network (i.e. the output layer), and $\mathcal{D} : \mathbb{R}^{N_s} \to \mathbb{R}^{r \times n_p}$ is the mapping function. The goal of the design network in equation (3.1) is to capture correlations

that exist in the design space, encode them and connect them to the latent solutions in the latent space. With this addition, the formulation of a DNN-based ROM can be represented as follows

$$\mathcal{R}_q^{(i)} = \mathbf{q}^{(i)} - \mathcal{G}_s^{(i)}\left(\tilde{\mathbf{q}}^{(i-n_p)}, \ldots, \tilde{\mathbf{q}}^{(i-2)}, \tilde{\mathbf{q}}^{(i-1)}; s\right), \tag{3.2}$$

where $\mathcal{G}_s^{(i)}$ is now parameterized by the design parameters through a new network gate proposed in this study.

Specializing this idea to an LSTM network, we can add a design gate to the LSTM cell as shown in figure 2*b*, leading to our proposed parametric LSTM (pLSTM) approach. This design gate directly involves design parameters in the learning process, which allows the ROM to learn different patterns of the dynamical system with respect to the design parameters in the design space. Therefore, the proposed pLSTM design gate is defined as follows:

$$\begin{aligned} \mathcal{S}_1 &= \mathcal{D}_1(s), \\ \mathcal{S}_2 &= \mathcal{D}_2(s), \\ \mathcal{Q}(t;s) &= \tilde{\mathbf{q}}(t) \times \mathcal{S}_1 + \mathcal{S}_2, \end{aligned} \tag{3.3}$$

where $\mathcal{Q}(t;s)$ is the output layer of the design gate. In this way, the pLSTM-gate has two separate design networks $\mathcal{D}_1$ and $\mathcal{D}_2$, and makes a linear correlation to obtain $\mathcal{Q}(t;s)$. This strategy provides better flexibility to find correlations between the state vector s and the latent solution $\tilde{\mathbf{q}}$, compared with alternative layer merging approaches (i.e. concatenation or element-wise adding). For a conventional merging method, including the effect of design parameters on all latent solutions as the input of the model requires concatenating all n_p sequential data. This would lead to an n_p fully connected network, increasing the risk of over-fitting and excessive sensitivity of the DNN model to the augmented latent solutions. Moreover, in preliminary tests, we observed that pure adding or pure multiplication does not perform well in capturing the dependency of the latent solutions on the design parameters. In the proposed solution, we therefore opt to use two design networks that complement each other to relate the latent solutions to the design parameters. The new architecture of the pLSTM, including this design gate, then turns into the following equations

$$\begin{aligned} I(t) &= \sigma\left(\mathcal{Q}(t-\Delta t;s)\mathbf{U}_i + \mathbf{h}(t-\Delta t)\mathbf{W}_i + \mathbf{b}_i\right), \\ F(t) &= \sigma\left(\mathcal{Q}(t-\Delta t;s)\mathbf{U}_f + \mathbf{h}(t-\Delta t)\mathbf{W}_f + \mathbf{b}_f\right), \\ O(t) &= \sigma\left(\mathcal{Q}(t-\Delta t;s)\mathbf{U}_o + \mathbf{h}(t-\Delta t)\mathbf{W}_o + \mathbf{b}_o\right), \\ \hat{C}(t) &= \tanh\left(\mathcal{Q}(t-\Delta t;s)\mathbf{U}_c + \mathbf{h}(t-\Delta t)\mathbf{W}_c + \mathbf{b}_c\right). \end{aligned} \tag{3.4}$$

Note that the rest of the architecture remains the same as the conventional LSTM. In general, the pLSTM uses the design gate to improve its fixed memory capacity that targets learning dynamics and can further memorize information about the attractors across different design parameters. Our proposed design gate approach contrasts with a more naive approach of creating a new network to directly control the weights inside the LSTM network, such as a hypernetwork [45,46]. The advantages of our approach are the increased tolerance of the model, reduced complexity in architecture and lower training costs. Specifically, by being integrated within the LSTM cell, pLSTM is able to suppress the noise in the solutions, leading to a stable DNN model for long-term future prediction of solutions. Further, to control all LSTM weights the output size of a hyper-DNN should be four times the input size of the conventional LSTM cell, which is much larger than our pLSTM design gates. Therefore, having such a DNN for tuning trainable parameters in the LSTM network may decrease the efficiency of the model both in training and prediction.

To summarize our proposed pLSTM-based 3A modelling approach, figure 3 illustrates the schematic of the ROM framework. In the case of 3A modelling, a similar hybrid framework in 2A is used, but employing pLSTM instead of conventional LSTM to incorporate the design

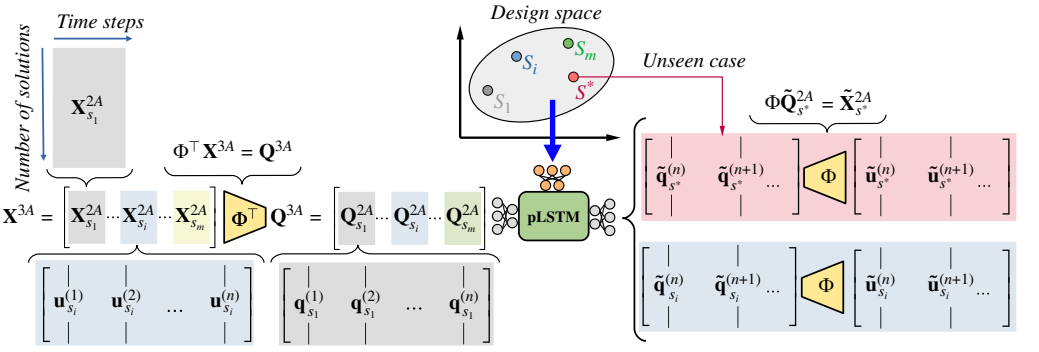

Figure 3. Schematic of ROM framework developed for 3A modelling using pLSTM.

space for ROM construction. The data matrix, $\mathbf{X}^{3A}$, is a multi-block matrix containing multiple $\mathbf{X}^{2A}_{s_i}$ matrices, each labelled with a distinct design parameter s_i with $i = 1, 2, \ldots, m$, where m is the number of cases with different design parameters, considered in the model training. Each $\mathbf{X}^{2A}_{s_i}$ matrix includes sequential states. Utilizing $\Phi^{\top}$, a multi-block matrix of latent solutions, $\mathbf{Q}^{3A}$ is derived, encompassing latent solutions matrices (i.e. $\mathbf{Q}^{2A}_{s_i}$) labelled with different design parameters. Subsequently, the ROM is trained by simultaneously feeding design parameters and the corresponding latent solutions into the pLSTM. Finally, the developed ROM can be utilized for predicting the dynamics of latent solutions in case studies involving unseen design parameters, denoted as s^*. Ultimately, these predicted low-dimensional results are projected back onto the high-dimensional physical space using the Φ basis functions.

4. Modelling of the Van der Pol oscillator

In this section, we aim to compare our proposed pLSTM approach with the conventional LSTM models for a low-dimensional nonlinear ODE. First, we construct a DNN based on the LSTM architecture leading to $1A^{\text{time}}$ aspect modelling, and then we add design space as the second aspect to the model by replacing the LSTM with our proposed pLSTM. Though the latter does not represent a 3A approach, applying pLSTM to this system does provide an opportunity to investigate the effect of adding design gates to an LSTM within a simple, low-dimensional setting.

The ODE considered is the Van der Pol oscillator, a non-conservative system widely used to simulate diverse engineering phenomena due to its nonlinear behaviour. The dynamics of the oscillator is governed by the second-order ODE

$$\frac{d^2g}{dt^2} - \mu(1 - g^2)\frac{dg}{dt} + g = 0, \quad g(t = 0) = \mathcal{N}(0, 1), \quad \frac{dg}{dt}(t = 0) = \mathcal{N}(0, 1), \tag{4.1}$$

where $g : \mathbb{T} \to \mathbb{R}$ denotes the oscillator's position as a function of time, and $\mu \in \mathbb{R}^+$ is the damping parameter defining the nonlinearity level in the oscillator. Moreover, $\mathcal{N}(0, 1)$ is a random number between 0 and 1. Small damping parameter values typically result in periodic or quasi-periodic motion, while surpassing a critical value around $\mu \approx 2$ leads to chaotic behaviour characterized by irregular oscillations and sensitivity to ICs.

We numerically marched the ODE forward in time with a time step of $\Delta t = 0.05$ using the fourth-order Runge–Kutta (RK44) for 18 different design parameters $s = \mu$, with values up to $s = 3$ contained within the design space. These 18 design parameters are random and uniformly distributed in the selected range. For the simulation of each case study, the IC is set to $\mathcal{N}(0, 1)$, and after 50 oscillation cycles, the temporal solution of the oscillator's position and its velocity, i.e. $\mathbf{q} = [g, \frac{dg}{dt}]$, are collected for eight extra cycles to build a dataset. We also provided the distribution of the mean squared error (MSE) error in the design space for the developed DNN model in

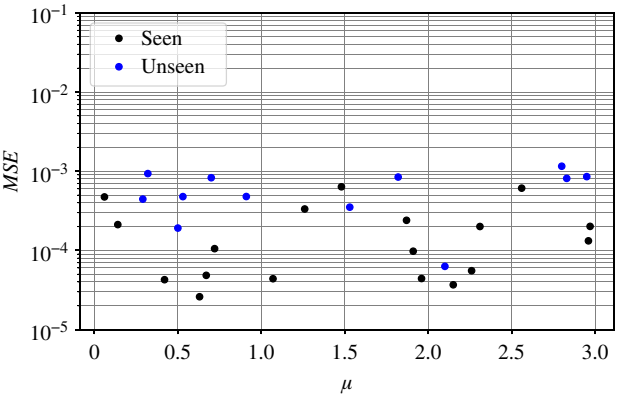

Figure 4. Distribution of the prediction error for the DNN used for the Van der Pol oscillator, distinguishing between the seen (black) and unseen (blue) data.

Table 1. Network architecture and parameters considered in building DNN for modelling of the Van der Pol oscillator.

parameter	value
network layers (LSTM)	[2×10, ReLU(30), Linear(2)]
network layers (pLSTM)	[2×10, ReLU(30), Linear(2)]
network layers (design network)	[1, ReLU(10), 2×10]
maximum epochs	1000
batch size	50
learning rate (LR)	$1 \times 10^{-2} 0.96^{i_e/i_d}$
test/training	0.1
optimizer	Adam
loss function	MSE
early-stopping callback	MSE (Test)

figure 4. As shown, the pLSTM is able to predict the solution of the dynamical system for all design parameters.

Given this training data, we explore two distinct LSTM approaches to compare against our proposed pLSTM method. First, we examine a scenario with an LSTM network trained on a single damping parameter, $s_1 = 0.14$, and denote this network LSTM_1 since it is trained on only one single parameter. Second, we consider an LSTM network that is trained on the entire training dataset (LSTM_{1+}) but does not include the design gate as proposed in the pLSTM approach. Both networks are tested in two scenarios. First, we provide each network the exact IC of s_1, which is contained in the training set for both LSTM_1 and for LSTM_{1+}, and compare the predictions with the simulated solution. Second, we provide each network the exact IC of another parameter $s_2 = 2.8$, which is an unseen case that is not contained in the training data for either LSTM. Meanwhile, the pLSTM approach is trained on the entire dataset and the associated design parameters, and then evaluated as well for both s_1 (seen) and s_2 (unseen).

The details regarding the final DNN architectures, hyper-parameters and settings employed for modelling the Van der Pol oscillator are presented in table 1. The network architecture is provided in a list format, where the initial and last items correspond to the DNN model's input and output, respectively. Intermediate items include the activation function alongside the number of units (neurons) within each hidden layer. For instance, in the LSTM architecture, the input is

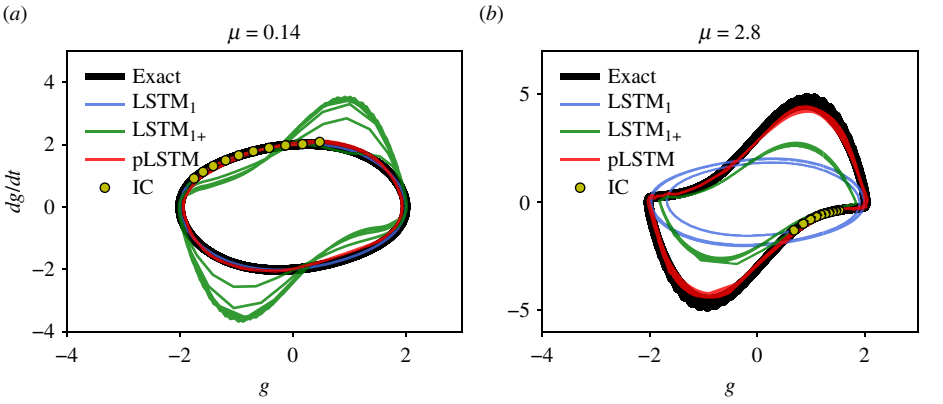

Figure 5. Comparing results of conventional LSTM and pLSTM architectures for the Van der Pol oscillator at (*a*) $s_1 = 0.14$ (seen) and (*b*) $s_2 = 2.8$ (unseen) design parameters.

structured as 2×10, where the first component denotes the number of solutions, and the second component, $n_p = 10$, means the number of sequential data. Subsequently, a rectified linear unit (ReLU) activation function is applied to a hidden layer comprising 30 units. The LSTM model's output consists of two units with a linear activation function, specifying the size of the output layer. The pLSTM network has the same architecture as the LSTM model. Additionally, the design network encompasses a single input, $s = \mu$, and its output layer matches the size of the allocated input layer for the solutions. The optimizer is configured as Adam with an adaptive learning rate defined by $LR(i_e) = 1 \times 10^{-2} 0.96^{i_e/i_d}$, where i_e stands for the epoch index, and i_d represents the decay step. The loss function is determined by the MSE, mathematically expressed as

$$MSE = \frac{1}{N_D} \sum_{i=1}^{N_D} \sum_{j=1}^{r} (q_j^{(i)} - \tilde{q}_j^{(i)})^2, \tag{4.2}$$

where $\tilde{q}_j^{(i)}$ denotes the predicted value by the DNN model indexed in vector $\tilde{\mathbf{q}}^{(i)}$, and N_D indicates the number of data utilized to calculate this error, applicable for both training and test datasets. The maximum epoch is set to 1000, yet the training may cease earlier due to a callback function monitoring the validation loss (i.e. test error).

Figure 5 shows the results of $LSTM_1$, $LSTM_{1+}$ and pLSTM for predicting solutions of the Van der Pol oscillator at $s_1 = 0.14$ (seen) and $s_2 = 2.8$ (unseen) in the design space. The IC is selected from the numerically advanced solution after the 50th cycle. In figure 5*a*, it can be seen that $LSTM_1$ and pLSTM effectively predict the limit cycle attractor solutions associated with their seen data, while $LSTM_{1+}$ fails. The reason for this failure is that $LSTM_{1+}$ is trained over the entire dataset, but it does not have a notion of the design parameters associated with the data. Therefore, $LSTM_{1+}$ is unable to extract the relations between the dynamical patterns and the design parameters. Consequently, $LSTM_{1+}$ is unable to recover the correct dynamics for either seen or unseen parameters. As depicted in figure 5*b*, when the damping parameter surpasses the critical value, nonlinear behaviour appears in the attractor, leading to chaos. In this case, to make sure the attractor is visible, the LSTM models predict the solutions from 50th to 98th cycles. Here, $LSTM_1$ and $LSTM_{1+}$ fail to predict accurate solutions compared with the exact one, while pLSTM is still able to provide a close prediction of the dynamics. The cloud of solutions in the upper and lower lobes of the attractors indicates the nonlinear behaviour that is expected at $s_2 = 2.8$.

Consequently, we can state that the $LSTM_1$, trained only on data for $s_1 = 0.14$, is understandably able to predict solutions at this parameter value but fails to predict solutions at $s_2 = 2.8$ when fed with the exact IC of s_2. The $LSTM_{1+}$, trained on the entire range of damping parameters, fails in both predictions despite using the exact IC associated with each of the design parameters. However, pLSTM, utilizing the design gate, encodes complex nonlinear

patterns across the design space, enabling predictions for dynamical behaviours even with unseen damping parameters in the Van der Pol oscillator.

Exploring this low-dimensional ODE highlighted the limitations of the conventional LSTM when applied to a system with a wide range of dynamic patterns. By contrast, the pLSTM demonstrated its ability to predict the dynamical system, even when unseen design parameters were provided.

In §5, we utilize the pLSTM for a true 3A modelling framework in complex fluid dynamic problems. Specifically, this involves a two-dimensional incompressible flow past a heaving and pitching ellipse with various kinematic design parameters, which will be elaborated in the following sections.

5. Set-up for applying pLSTM to a heaving and pitching ellipse

In this section, we detail the numerical flow solver, problem set-up, network architectures and training approach used for applying the pLSTM to a 3A analysis of the flow past a heaving and pitching ellipse.

(a) Numerical solver

The flow solver used in this work simulates the two-dimensional incompressible Navier–Stokes equations in vorticity transport form with a sharp treatment of immersed boundaries [47,48]. The vorticity transport equation is written as

$$\frac{\partial \omega}{\partial t} = -\nabla \cdot (\mathbf{v}\omega - \nu \nabla \omega), \tag{5.1}$$

where $\omega = \nabla \times \mathbf{v}$ is the scalar vorticity, $t \in \mathbb{T}$ is time in a time space $\mathbb{T}$ and ν represents the kinematic viscosity. The divergence-free velocity vector $\mathbf{v} = [v_x, v_y]$ can be expressed in terms of a scalar stream function ψ that satisfies a Poisson equation

$$\begin{aligned} \mathbf{v} &= \nabla \times \psi, \quad \text{on} \quad \Omega, \\ -\nabla^2 \psi &= \omega \quad \text{on} \quad \Omega, \end{aligned} \tag{5.2}$$

on the computational domain Ω with dimensions of L_∞ and H_∞ in x and y directions. The streamfunction is solved with free-space boundary conditions on the inflow and top/bottom domain boundaries, and an outflow boundary condition is applied on the downstream domain boundary [47]. The computation of the velocity field is supplemented with no-through boundary conditions and appropriate circulation constraints, whereas the vorticity transport equation is solved with a wall vorticity boundary condition to enforce no-slip [47,48]. Equations (5.1) and (5.2) are discretized using a second-order finite-difference method, and the solutions are evolved in time using a third-order Runge–Kutta temporal scheme.

(b) Problem definition

The flapping ellipse with an aspect ratio of 0.12 has a rigid-body motion with following harmonic heaving and pitching motion

$$y(t) = A_y \sin(2\pi f t), \tag{5.3}$$

$$\theta(t) = A_\theta \sin(2\pi f t + \phi_\theta), \tag{5.4}$$

where f is the flapping frequency, A_y represents the heaving amplitude, A_θ is the pitching amplitude and ϕ_θ is the phase angle between the heaving and pitching motion. The flapping

Table 2. Design parameters considered in this study.

design parameters	values (number of repetitions)
St [-]	0.3 (9), 0.35 (9), 0.45 (18), 0.55 (7), 0.6 (9)
A_θ [°]	20 (9), 25 (9), 30 (18), 35 (7), 40 (9)
ϕ_θ [°]	75 (9), 85 (9), 90 (9), 95 (8), 100 (8), 105 (9)

frequency is non-dimensionally expressed using the Strouhal number $St = 2fA_y/U_\infty$, where U_∞ is the free-stream velocity. The flow regime is characterized by the Reynolds number $Re = U_\infty c/\nu$, where c is the chord length of the ellipse. In this study, the Reynolds number is fixed to $Re = 500$ and the heaving amplitude is set to $A_y = 0.5c$.

The simulations are performed in a rectangular domain of resolution 1280×768 grid points, with a grid spacing $h = c/144$. The leading edge of the ellipse is placed at a distance of $c/2$ from the inflow plane. These settings are chosen based on extensive convergence analysis for problems at similar or higher Reynolds numbers in [48], as well as previous applications and benchmark studies using the same code [49,50].

The design parameters are the Strouhal number, the pitching amplitude and the phase shift between heave and pitch. Therefore, we define the vector of design parameters as $s = [St, A_\theta, \phi_\theta]$. We selected $m = 53$ parameter combinations, summarized in table 2 and simulated each one of them to generate training data for this study. Each case was simulated until 16 ellipse flapping cycles to ensure the solutions reached quasi-steady conditions. The corresponding dataset for each case study includes 40 samples per ellipse flapping cycle, captured during the last four cycles leading to 160 spatial flow fields per simulation.

(c) Architecture and training details

Figure 6 shows a schematic of the 3A-pLSTM framework applied to the current study. Equation (2.1), which can be understood as a general description of the Navier–Stokes equations, are solved in the (discrete) physical space $\mathbb{P}$ and the solutions are stored for temporal snapshots $t = [t_0, t_1, \ldots, t_n]$ in the time domain $\mathbb{T}$. Since, in the simulation, the ellipse is moving with respect to the domain, direct application of SVD on this data will not yield efficient data compression [51–53]. Therefore, using the approach proposed in [51,54], we perform a coordinate transform from the Eulerian coordinates x to Lagrangian coordinates $\tilde{x}$, moving and rotating with the heave and pitch motion of the ellipse. Therefore, using this transformation, the flapping ellipse remains stationary with respect to $\tilde{x}$. After transforming the flow fields data to the Lagrangian frame, we employ equations (2.6) with an orthogonality constraint to transform the data into a low-dimensional latent space. We employed SVD on the whole dataset built by snapshots from all case studies together, as sketched in figure 3. The magnitude of some of the POD modes, $|\phi_i|$, is shown in figure 6. Moreover, figure 7 shows the changes in the non-dimensional eigenvalues and accumulation with respect to the rank, i.e. the number of POD modes. Here, γ_i is the ith eigenvalue. As shown, after $r > 300$ the eigenvalues reduce significantly, but keeping this amount of solutions is too much in the context of ROM. Therefore, the rank is set to $r = 150$, for which truncated POD modes contain 98.2% of the energy.

Note that we can choose a higher number of ranks to increase the information accumulation in the model; however, there are a few technical issues that should be addressed. First, as explained before, the data are transformed into a Lagrangian frame to avoid spatial flow motions caused by the moving ellipse. However, by this transformation, the flow itself has some spatial motions in the domain, resulting in minor noise in the high frequency (i.e. high rank) POD modes. Therefore, adding extra modes may hurt the accuracy of the model and give fewer improvements in the data reconstructions. Second, by adding more ranks to the model, the number of input and output units in the LSTM network increases, leading to an essential need to increase the size of networks in the

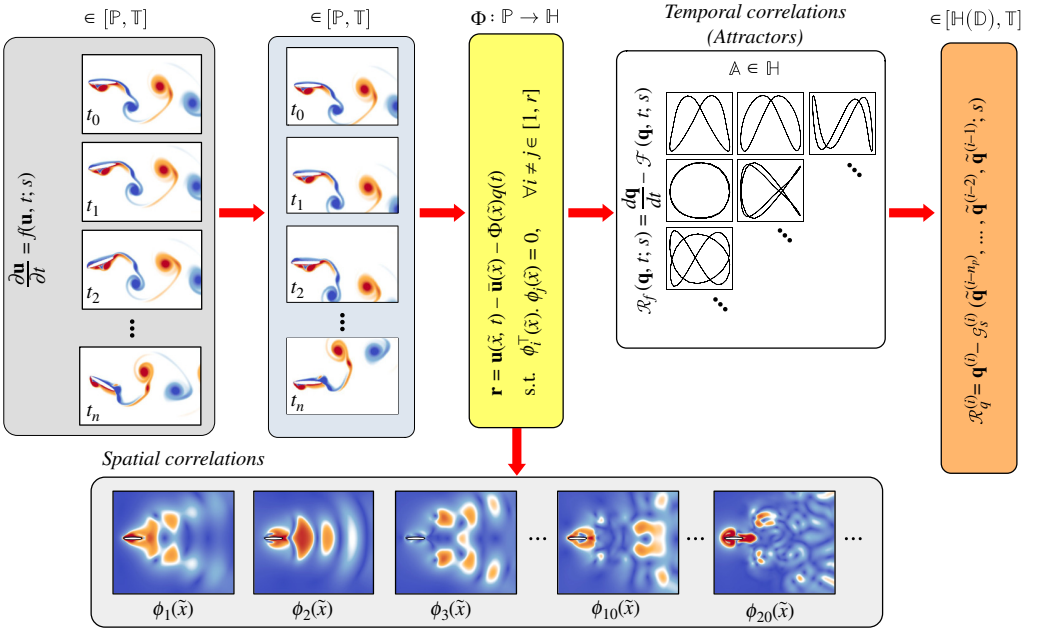

Figure 6. Schematic of the proposed framework for the current study that shows spatial and temporal correlations.

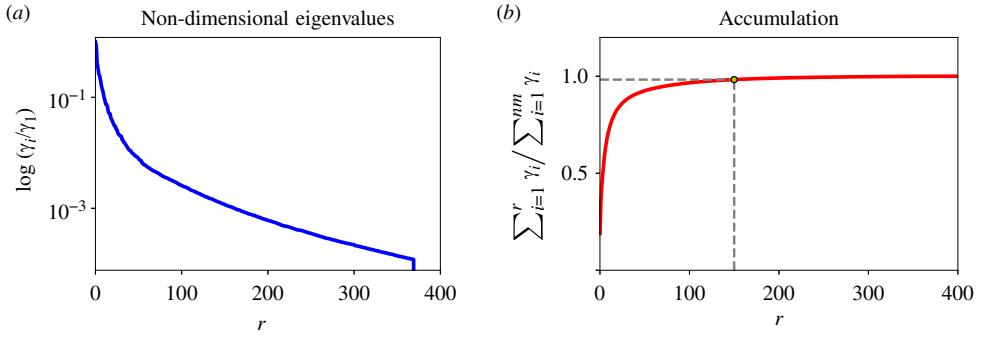

Figure 7. Variations in the non-dimensional eigenvalues (*a*) and accumulation (*b*) as a function of the rank. Here γ_i denotes the *i*th eigenvalue. At $r = 150$, the accumulation is 98.2%.

LSTM. Hence, this larger network brings new issues, such as over-fitting, less efficient prediction, difficulties in training and its higher computational costs and local optima in the loss function domain, leading to different model predictions after each training procedure.

To establish hyper-parameters for designing the neural network architecture, we conducted a comprehensive investigation into the number of time steps (n_p), the number of hidden layers (N_{ly}) and the number of units (N_{nr}) concerning the training and test errors, obtained by MSE. Table 3 displays the training and test errors for each architecture alongside the corresponding processing time per unit. The processing time denotes the time required to train a unit completely within the neural network. Table 3 reveals a significant reduction in training and test errors when the number of units is $N_{nr} = 640$, beyond which no substantial error reduction is observed. Notably, for $N_{nr} = 640$, the test error is lower for $N_{ly} = 1$ compared with $N_{ly} = 2$. Thus, adding more hidden layers does not notably affect the model's accuracy and we selected $N_{ly} = 1$ and $N_{nr} = 640$ for each LSTM cell. When the number of units is sufficiently large ($N_{nr} = 640$ or 1280) we can identify a trend in the number of time steps. For those cases, increasing from $n_p = 5$ to $n_p = 10$ leads to a clear reduction in errors, indicating that the amount of information from five previous time

Table 3. Hyper-parameters and their effect on the training performance of the pLSTM cell.

time steps n_p	hidden layers N_{ly}	units N_{nr}	train error (*MSE*)	test error (*MSE*)	time (min/N_{nr})
5	1	160	1.55×10^{-3}	1.92×10^{-2}	0.036
5	1	320	2.49×10^{-4}	3.61×10^{-3}	0.020
5	1	640	3.32×10^{-5}	7.87×10^{-3}	0.036
5	1	1280	1.52×10^{-5}	1.36×10^{-2}	0.032
5	2	160	1.21×10^{-2}	1.48×10^{-1}	0.062
5	2	320	9.02×10^{-4}	1.74×10^{-2}	0.066
5	2	640	2.03×10^{-4}	5.25×10^{-3}	0.086
5	2	1280	1.14×10^{-4}	5.02×10^{-3}	0.108
10	1	160	4.47×10^{-2}	5.75×10^{-2}	0.18
10	1	320	2.24×10^{-2}	3.98×10^{-2}	0.10
10	1	640	3.11×10^{-5}	7.01×10^{-4}	0.091
10	1	1280	2.21×10^{-5}	9.21×10^{-4}	0.089
10	2	160	4.46×10^{-2}	5.87×10^{-2}	0.20
10	2	320	2.25×10^{-2}	3.92×10^{-2}	0.13
10	2	640	1.65×10^{-5}	1.16×10^{-3}	0.16
10	2	1280	1.65×10^{-5}	8.91×10^{-4}	0.19
20	1	160	1.66×10^{-3}	2.12×10^{-2}	0.114
20	1	320	2.40×10^{-4}	3.91×10^{-3}	0.113
20	1	640	3.49×10^{-5}	7.28×10^{-3}	0.136
20	1	1280	9.98×10^{-5}	9.33×10^{-4}	0.170
20	2	160	1.30×10^{-3}	1.60×10^{-2}	0.222
20	2	320	1.55×10^{-4}	2.25×10^{-3}	0.234
20	2	640	3.90×10^{-5}	8.78×10^{-4}	0.310

steps provided to the pLSTM is not enough for this system. Interestingly, the main effect of increasing n_p from 5 to 10 is in the reduction of the testing error. This implies that the architecture is able to predict the training data even with a small time history, consistent with the Markov property of the underlying equations. However, when applied to unseen parameter vectors, the pLSTM architecture requires a longer history to predict the associated dynamics with similar accuracy. Increasing further to $n_p = 20$ shows stagnating or increasing error values, while training and prediction costs increase. We thus choose $n_p = 10$. Finally, it is worth mentioning that the same procedure is applied to set hyper-parameters for LSTM cells and we obtained the same architecture.

The information about the final DNN architectures, hyper-parameter and settings are provided in table 4. The LSTM has an input with dimensions of 150×10, where the first term is equal to the rank, $r = 150$ and the second term is $n_p = 10$. Then, the ReLU activation function is applied to a hidden layer with 640 units. The output of this LSTM model also has 150 units with a linear activation function, where the unit indicates the output layer size. The pLSTM network has the same core architecture as LSTM. The design network has three inputs, i.e. $s = [St, A_\theta, \phi_\theta]$ and the output layer of this network is the same size as the input layer allocated to the latent solutions (i.e. 150). The optimizer is set to Adam with an adaptive learning rate defined as

Table 4. Network architecture and parameters considered in building DNN reduced-order modelling.

parameter	value
network layers (LSTM)	[150×10, ReLU(640), Linear(150)]
network layers (pLSTM)	[150×10, ReLU(640), Linear(150)]
network layers (design network)	[3, ReLU(150), ReLU(150), 150×10]
maximum epochs	1000
batch size	164
learning rate (LR)	$1 \times 10^{-3} 0.96^{i_e/i_d}$
test/training	0.2
optimizer	Adam
loss function	MSE
early-stopping callback	MSE (Test)

$LR(i_e) = 1 \times 10^{-3} 0.96^{i_e/i_d}$, the maximum epoch is set to 1000, but the training may stop earlier since we set a callback function, monitoring the validation loss (i.e. test error).

6. Comparing pLSTM with generalizations of classical approaches

Here, we wish to compare our proposed pLSTM approach with naive applications of LSTM models across different parameters within a design space. To do so, we follow the approach of §4 and consider two different LSTM approaches. First, we consider a case with LSTM trained on a single parameter set s_1 (LSTM_1) and second, we consider the case with LSTM trained on the entire training set provided in table 2 (LSTM_{1+}). In both cases, we evaluate the LSTM's predictions after initializing them with either the IC of s_1 (contained within the training set) or the IC of s_2 (unseen data that are not contained within the training set). We choose parameter sets $s_1 = [0.3, 20°, 90°]$ and $s_2 = [0.55, 35°, 100°]$. We compare their performance with that of pLSTM trained on the entire dataset and initialized with the same ICs.

The results obtained with LSTM_1, LSTM_{1+} and pLSTM are shown in figure 8. The results for both s_1 (seen) and s_2 (unseen) design parameters are provided in two different columns. Here, *Prediction* denotes the history of the predicted solutions by the models in three-dimensional phase space and *stable* refers to the final solutions when the transition effect vanishes. Furthermore, the results in three-dimensional phase space are for $[q_1, q_2, q_3]$ and $[q_1, q_{30}, q_{40}]$, indicating the solutions in lower and higher frequencies, respectively. Also, the projections of the solutions are shown in two-dimensional phase spaces, plotted using lighter colours on the three interior faces of the plotting domain. The exact solution in latent space is obtained directly from the SVD step. Note that all models are initiated with the exact IC associated with either s_1 (left column) or s_2 (right column). As shown in the first row of figure 8, the LSTM_1 model with s_1 parameter set can capture the dynamics accurately for latent solutions at lower and higher frequencies. When initializing LSTM_1 with the exact solution of the unseen parameter set s_2, however, it fails to predict correct solutions. This is also expected: if the latent solutions of different dynamics are given as the IC to the LSTM_1 model, it cannot recover and reproduce the whole dynamics because it extrapolates the pattern of the dynamical system, which is not mathematically correct. Hence, when the solution of the LSTM_1 model becomes stable, we observe that the predicted dynamics are not correct.

In the next attempt, we build a ROM by training LSTM_{1+} using the training dataset that spans the entire design space. This would allow the LSTM_{1+} model to learn different dynamics based on the various latent solutions given as input to this model. The second row in figure 8 shows the predictions after initializing LSTM_{1+} with either s_1 or s_2 parameter sets. The plots

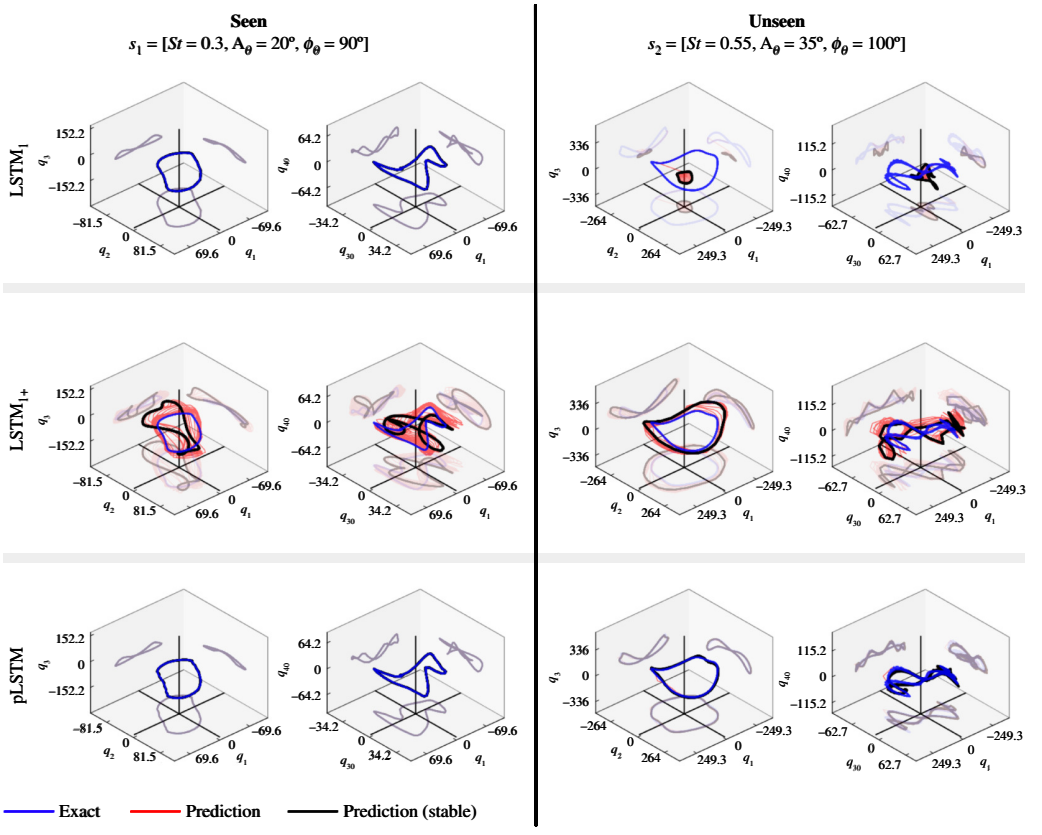

Figure 8. Temporal evolution of the latent solutions in the phase space diagram for the LSTM$_1$, LSTM$_{1+}$ and pLSTM models for s_1 (seen) and s_2 (unseen) case studies. These models are initiated by the exact ICs $\mathbf{q}_0(s_1)$ and $\mathbf{q}_0(s_2)$.

show that LSTM$_{1+}$ can predict the latent solutions accurately only for the first few cycles, after which the prediction diverges from the exact solutions. Eventually, the dynamics of the stable latent solutions will be different from the original ones. This issue can be attributed to the error accumulation in the results of LSTM$_{1+}$ at each time step. As mentioned before, the system identification in this LSTM$_{1+}$ model can only be done using the initial input. However, if the residual at the output of the LSTM$_{1+}$ model is considerable, the solution vector $\tilde{\mathbf{q}}^{(i)} + \mathcal{R}_q^{(i)}$ represents a perturbation of the reference dynamical system. As the error accumulates over the time steps, this deviation will become more significant, and eventually, the predictions converge to spurious latent solutions. As shown, the prediction of the dynamical system for s_1 has converged to a totally different solution, even though s_1 was part of the training data. The predicted stable solutions for s_2 are more similar to the exact ones, but they still do not represent the correct dynamics. This means that naively training an LSTM model on a parameter set that spans an entire design space will not yield reliable results, even when initialized with exact solutions of parameter combinations within the training data.

Finally, we consider the performance of our pLSTM model on the same case studies used for the previous LSTM models. The last row in figure 8 illustrates the results of the pLSTM model. As seen, the solutions predicted by the pLSTM model are in good agreement with the exact ones for both transient and stable conditions even after $40T$. For the unseen case study, s_2, the results of pLSTM for the latent solutions are also in good agreement with the exact ones for both lower and higher frequencies. At higher frequencies, the predicted results have some discrepancies, which are typical for unseen cases.

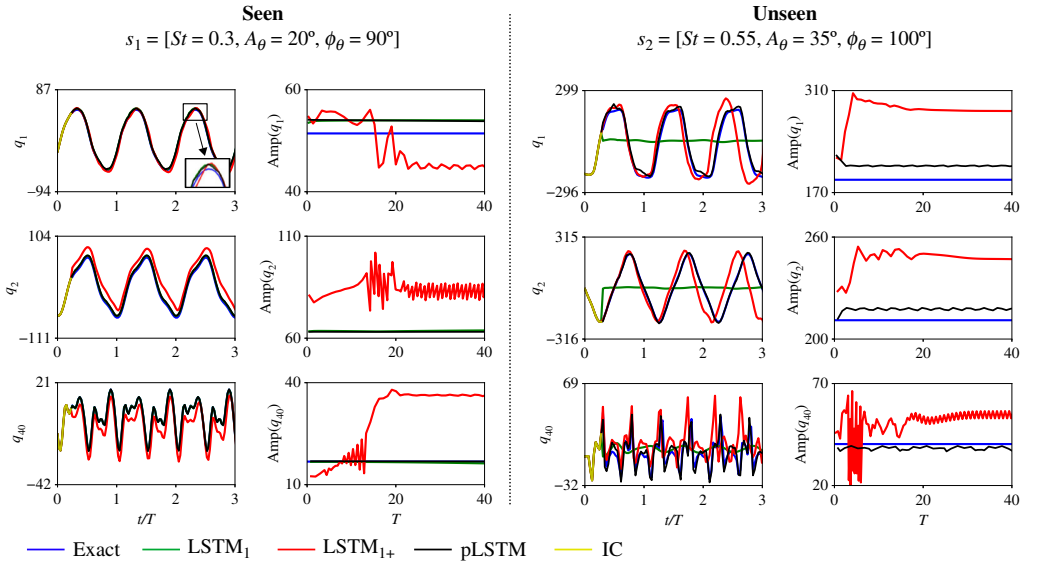

Figure 9. Comparing dynamical characteristics of different LSTM models. These models are tested by exact initial conditions $\mathbf{q}_0(s_1)$ and $\mathbf{q}_0(s_2)$ for seen and unseen case studies, respectively.

In order to investigate the performance of the LSTM models quantitatively, we measure the amplitude and frequency of the latent solutions for all models considered in this study and compare them with those of the exact latent solutions.

Figure 9 shows the results of LSTM models and the exact solutions for both the s_1 and s_2 parameter sets, for low-frequency (q_1, q_2) and high-frequency ($q_4 0$) modes. Additionally, the upper amplitude of the latent solution is computed for each cycle until $T = 40$, such that $\mathrm{Amp}(q) = \max\left(q(t_i)\right)$ for each cycle $T_i \leq t_i < T_{i+1}$ with $i = 1, \ldots, 40$. The first and third columns of figure 9 show the IC given to the models and the latent solutions are plotted until $t = 3T$. The results confirm what is seen in figure 8: LSTM$_1$ is stable and accurate for seen parameters s_1 but fails for unseen parameters s_2. For s_1, LSTM$_{1+}$ initially captures the behaviour well but switches to predict different dynamics after around $T = 10$ and its predictions significantly deviate from the exact latent solutions afterwards (second column). For the unseen parameters s_2, LSTM$_{1+}$ shows increasingly large errors during the first three cycles, and again switches to a different dynamical system, with much higher amplitudes compared with the exact solution (fourth column). The pLSTM results remain stable and relatively accurate for both s_1 and s_2, with some small errors in the amplitudes for s_2.

We can convert these transient predictions of the latent solutions to the frequency domain in order to evaluate stability and dispersion errors in the predicted results. Since our flow results are characterized by a limit cycle, we expect our latent solutions to remain periodic as well. Table 5 shows the frequency results for all cases in figure 9, with all frequencies computed using FFTs of the transient results and expressed non-dimensionally as Strouhal numbers. For the case study with s_1 parameter set, the flapping ellipse has a Strouhal number of $St = 0.3$. As shown, the LSTM$_1$ and pLSTM latent solutions have a primary frequency of $St = 0.302$, while LSTM$_{1+}$ has $St = 0.324$, resulting in errors of 0.66 and 8%, respectively. The exact secondary frequency for s_1 is $St = 0.6$, while this frequency for LSTM$_1$ and pLSTM is $St = 0.603$ (error of 0.5%), and for LSTM$_{1+}$ is $St = 0.646$, resulting in 7.6% error. The LSTM$_1$ and pLSTM thus predict the correct frequencies for the latent solutions with a negligible error, which match the time histories shown in the left column of figure 9. By contrast, the LSTM$_{1+}$ predicts higher St by default, resulting in faster variations in the dynamics of the predicted system. For the s_2 parameter set, the exact primary frequency also matches the frequency of the flapping ellipse, which is $St = 0.55$. As shown

Table 5. Comparing the dominant (primary and secondary) frequencies of the latent solutions obtained by different LSTM models for case studies with parameter sets of s_1 and s_2.

case	model	primary frequency (St)	error (%)	secondary frequency (St)	error (%)
s_1 (seen)	exact	0.3	–	0.6	–
	$LSTM_1$	0.302	0.66	0.603	0.5
	$LSTM_{1+}$	0.324	8	0.646	7.6
	pLSTM	0.302	0.66	0.603	0.5
s_2 (unseen)	exact	0.55	–	1.1	–
	$LSTM_1$	none	none	none	none
	$LSTM_{1+}$	0.605	10	1.22	11
	pLSTM	0.552	0.36	1.103	0.27

in figure 9, the $LSTM_1$ completely fails to predict the latent solutions for the specified design parameters, so its primary and secondary frequencies are incorrect and shown by *None*. The $LSTM_{1+}$ model predicts higher primary and secondary frequencies of $St = 0.605$ and $St = 1.22$, respectively (with 10 and 11% errors). However, the pLSTM predicts the primary frequency of $St = 0.552$ with an approximate error of 0.36% with reference to the St given in the parameter set. Also, the secondary frequency obtained for the results of the pLSTM model is $St = 1.103$, resulting in an approximate error of 0.27%.

Finally, one may ask what if we use the concatenation or element-wise approaches to add the design parameters to the neural network model, instead of the design gate equipped in the pLSTM. To address this question, we trained models with these two traditional approaches. In the concatenation approach, the design parameters are concatenated to the solutions before feeding them to the conventional LSTM model. In the element-wise approach, on the other hand, the design parameters are expanded with a one-layer fully connected network (i.e. feedforward neural network) to have an output with the same size as the latent solutions. Then, this fully connected layer is added to the latent solutions element by element. Figure 10 shows the results of the latent solutions predicted by conventional LSTM with concatenation and element-wise adding, compared with the pLSTM results, both for seen (a) and unseen (b) data. As shown in figure 10*a*, the concatenation approach can predict the solution for four cycles, while the error of prediction increases in time. The element-wise approach fails to predict after about $t/T = 3$. On the other hand, the pLSTM outperforms two other approaches. Figure 10*b* also shows the predicted solution for the first latent solution for the unseen case. As shown, the concatenation and element-wise approaches fail to predict the solution within the first flapping cycle, while the pLSTM maintains accuracy and stability. Therefore, for both seen and unseen cases, the pLSTM outperforms two other approaches.

Based on these results, it can be concluded that LSTM models, regardless of the number of case studies used for training, cannot accurately predict solutions for different design parameters. This is because they solely rely on the latent solutions from previous time steps and any small error accumulation in the sequential latent solutions leads to different predictions of attractors, ultimately resulting in incorrect dynamics. On the other hand, pLSTM models can predict different dynamics and converge to the actual attractors of the system based on the given design parameters while remaining stable. This is due to the presence of a design gate in the pLSTM model, which allows for searching for different dynamics in the design space and acting as a solid constraint to ensure that the predicted latent solutions remain consistent with the specified design parameters. Moreover, the design gate limits the growth of error accumulation in the sequential prediction of the latent solutions.

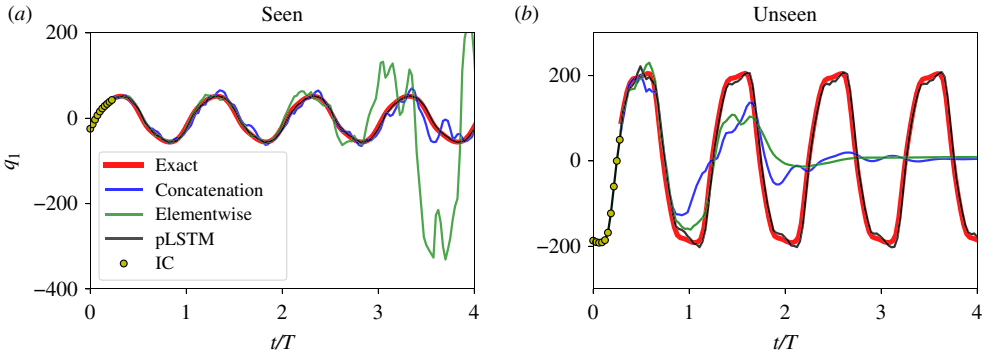

Figure 10. (*a*,*b*) Comparing results of pLSTM and the conventional LSTM with concatenation and element-wise adding of the design parameters to the solutions.

7. Robustness to the initial condition and real-time parameter variations

As discussed in the previous section, the design gate in the pLSTM model is responsible for ensuring that its solution predictions stay within the actual attractor defined by the design parameters. The design gate plays a crucial role in controlling the stability and accuracy of the dynamical system. However, it is important to determine the tolerance level of the design gate in the pLSTM model to errors in the solution. To investigate this, we perturb the ICs and feed them into the pLSTM model to observe its response to different levels of perturbations. After that, we investigate the effect of changes in the design parameters on the pLSTM model in real-time modelling.

(a) Initial condition

We add a scaled matrix of random noise, $\mathbf{Q}_{\text{rand}} \in \mathbb{R}^{r \times n_p}$ with $\text{Range}(\mathbf{Q}_{\text{rand}}) = [-0.5, 0.5]$ to the exact IC $\mathbf{q}_0(s)$, with s the parameter vector considered. The augmented IC of the latent solutions with a certain noise level is then defined as

$$\tilde{\mathbf{q}}_0(s) = \mathbf{q}_0(s)\left[1 + \varepsilon \mathbf{Q}_{\text{rand}}\right], \tag{7.1}$$

where ε is a scalar that tunes the noise amplitude.

Figure 11 displays the stable latent solutions predicted by pLSTM in the phase space, with noise amplitudes increasing from $\varepsilon = 0$ (exact) to $\varepsilon = 10$ (1000% noise). Interestingly, the results show that the pLSTM model can still recover the original attractors for both seen and unseen design parameters and the predicted solutions coincide with the corresponding exact solutions. This means that when the augmented latent solutions are given to the pLSTM, the design gate adds corrections to the input data before sending them to the other gates. These corrections result in a prediction in the output of the pLSTM model, which is more likely within the attractor represented by the specified design parameters. Therefore, after applying these corrections by the design gate for a few time iterations, the latent solution converges to the desired stable attractor. It is worth mentioning that the corrections applied by the design gate to the augmented solutions are rapid, leading to convergence in less than one ellipse flapping cycle.

(b) Real-time parameter changes

Since the model predictions are robust to noise in the initialization, we can further demonstrate the robustness of the pLSTM model to dynamic variations with respect to the design parameters. To do so, we identify three different design parameter sets and abruptly switch between them

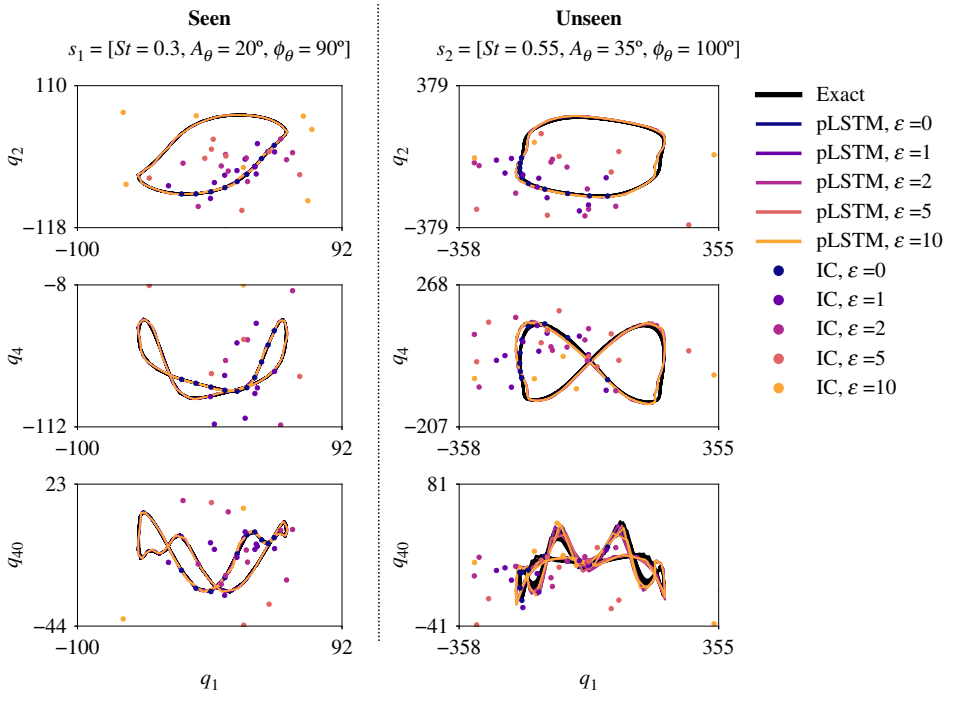

Figure 11. Temporal evolution of the latent solutions predicted by the pLSTM model with the augmented initial condition for the case studies with s_1 (seen) and s_2 (unseen).

every $tU_\infty/L_\infty = 500$ time units, according to the first row in figure 12. Among these three design parameter sets, $[St = 0.45, A_\theta = 20°, \phi_\theta = 90°]$ and $[St = 0.45, A_\theta = 30°, \phi_\theta = 95°]$ are among the seen design parameters, while $[St = 0.55, A_\theta = 35°, \phi_\theta = 100°]$ is unseen. Note that we did not use the gradual variation of design parameters since our approach is not designed to capture the physics associated with such transients. Rather, we wish to investigate the long-term stability of our predictions due to such real-time parameter changes. The second row in figure 12 illustrates the switch between parameter sets at instants $tU_\infty/L_\infty = 500$ (grey) and $tU_\infty/L_\infty = 1000$ (yellow). Moreover, the third row in figure 12 shows the time evolution of the predicted latent solutions (i.e. q_1, q_2 and q_3). As shown, the latent solutions have harmonic patterns and after sudden changes in the design parameters, their behaviours undergo a short spurious transient before they become stable again. In the last row, the latent solutions in the phase space are shown. The switch point (yellow) indicates the moment that changes are applied, after which the pLSTM model quickly finds the correct new attractors properly.

8. Physical analysis of the pLSTM predictions

So far, we have focused on the accuracy of the predicted pLSTM results in latent space. Here we investigate the high-dimensional results in physical space, i.e. the predicted (non-dimensional) vorticity $\omega c/U_\infty$ and velocity components v_x/U_∞ and v_y/U_∞ in the computational domain.

Figures 13 and 14 show contours of the normalized vorticity field, $\omega c/U_\infty$ and the normalized velocity magnitude, $(|\mathbf{v}| - U_\infty)/U_\infty$, at four different time instants within a cycle, for both the s_1 (seen) and s_2 (unseen) parameter combinations, respectively. The upper and lower contours belong to the prediction by the pLSTM model and the exact solutions, respectively. For s_1 (left column), the results of the pLSTM model are in good agreement with the simulation results. The pLSTM model for this case study captures and recovers the primary flow structures well. For s_2 (right column), operating at a higher Strouhal number, we observe higher velocity gradients

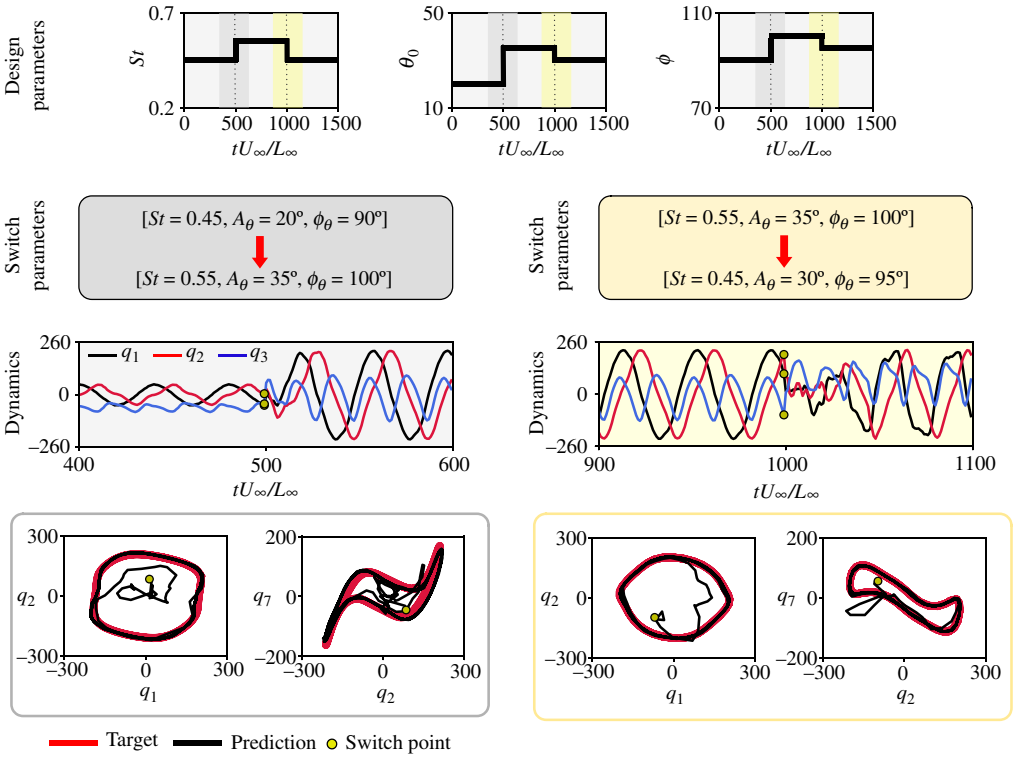

Figure 12. Results of real-time modelling for three sets of design parameters.

and magnitudes, leading to some discrepancies in the predictions. Still, the pLSTM predictions match well with the exact solutions. Note that some discrepancies exist in the vorticity contours, which may originate from the fact that pLSTM is trained by the velocity components, and hence, the vorticity values are computed by the derivatives of those velocity components predicted by pLSTM. Therefore, some errors in the vorticity contours are expected.

A more quantitative comparison is presented in figure 15, which shows the non-dimensional velocity profiles at each of the instances in figure 14, measured along the y-axis at $x/L_\infty = 0.34$. Note that the exact velocity profiles are the reconstructed ones from 700 POD modes, containing 99.99% of the total kinetic energy. For s_1, the predicted velocity profiles for both v_x/U_∞ and v_y/U_∞ coincide with the exact corresponding non-dimensional velocity profiles (first and second columns). However, some differences result from the truncated errors of POD modes and numerical errors in the pLSTM model itself. Moreover, the non-dimensional velocity component profiles predicted by pLSTM for the case study with s_2 are also in good agreement with the exact solutions (third and fourth columns). At $t/T = 0.5$, pLSTM underpredicts the minimum value of v_y/U_∞ in the wake, which can be attributed to some information missed in the ROM for this unseen case study. We believe that this error can be reduced by increasing the rank, r, in order to increase the dimensionality of the latent solutions to the DNN model. However, we note that increasing r too much adds excessive complexity to the model so that, ultimately, the optimal value of r will be obtained from an application-specific trade-off between required accuracy and cost.

In general, pLSTM captures the primary flow structures well for both seen and unseen design parameters. Moreover, comparing the velocity profiles shows that the accuracy of the pLSTM model is in good agreement with the exact solutions. Therefore, high-fidelity flow fields can be reconstructed and reproduced for various design parameters to study physical phenomena in a

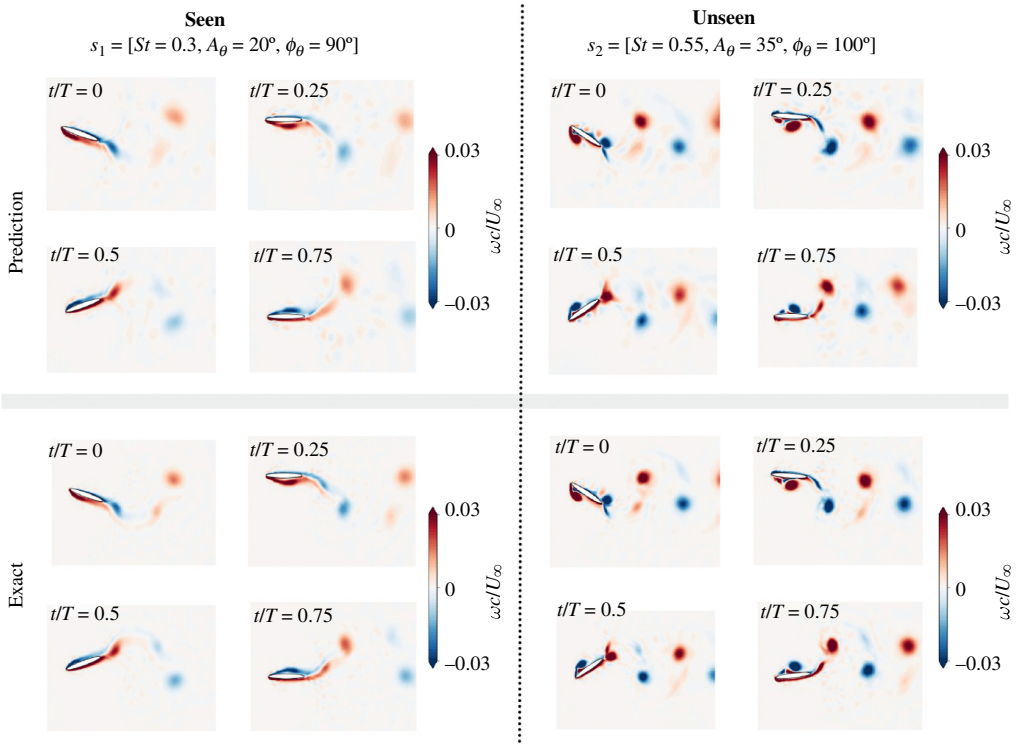

Figure 13. Contours of the non-dimensional vorticity field, $\omega c/U_\infty$, at different moments in one flapping cycle T for the cases with s_1 (seen) and s_2 (unseen).

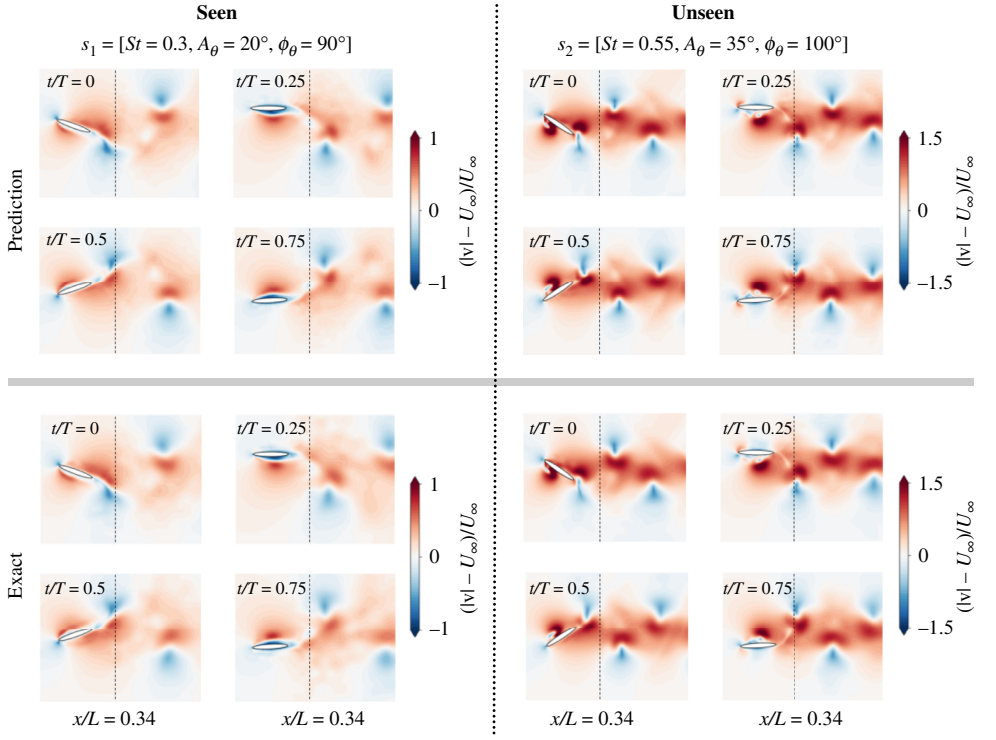

Figure 14. Contours of the normalized velocity magnitude, $(|\mathbf{v}| - U_\infty)/U_\infty$, at different moments in one flapping cycle T for the cases with s_1 (seen) and s_2 (unseen).

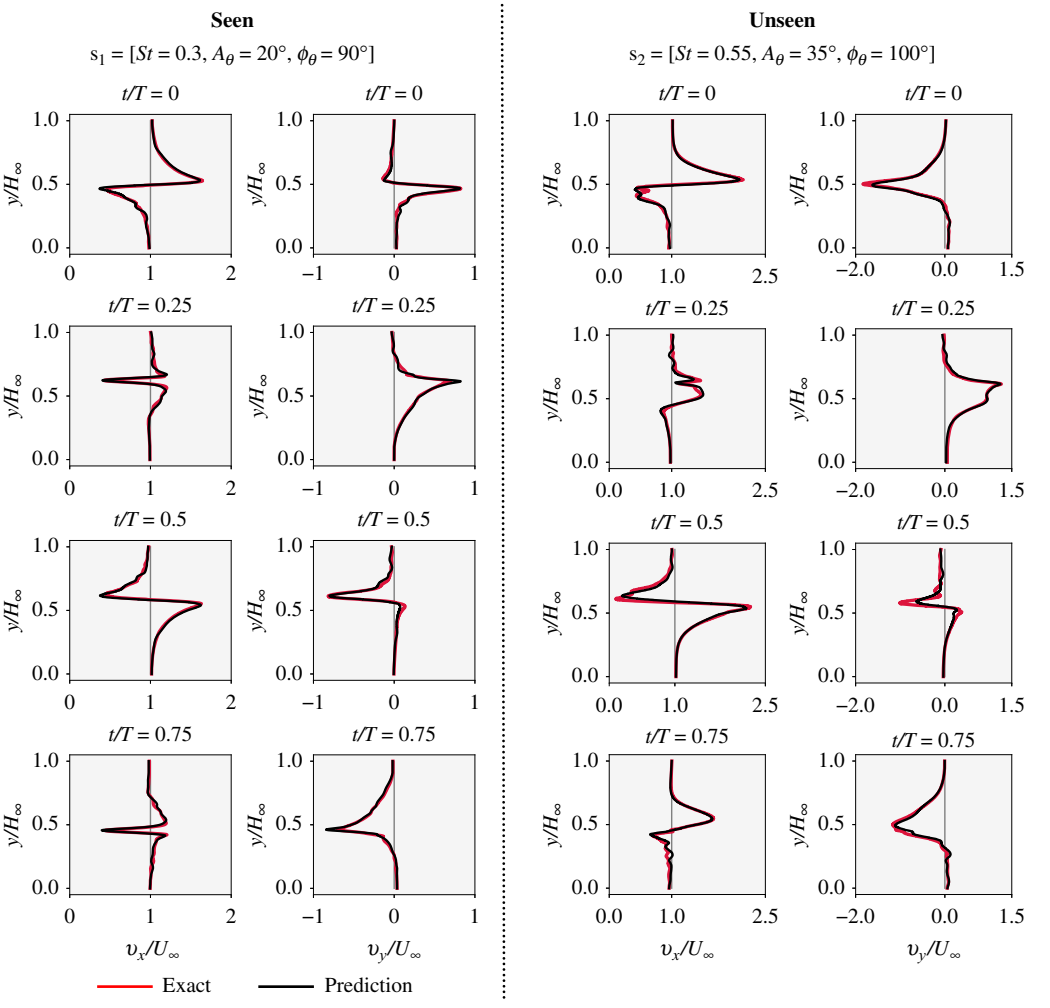

Figure 15. Non-dimensional velocity profiles for components v_x/U_∞ and v_y/U_∞ along the y-axis at $x/L_\infty = 0.34$ predicted by the pLSTM model at four different moments in a flapping cycle T. The results belong to the case studies with s_1 (seen) and s_2 (unseen).

design exploration or numerical optimization context. It is worth mentioning that we observed both 2S and 2P wake types [55], depending on the precise combination of the design parameters. These wake patterns were covered within the training dataset and were correctly predicted by pLSTM for both seen and unseen combinations of design parameters.

9. Conclusions

This study introduces a three-aspect reduced-order model (3A-ROM), enabling simultaneous prediction across spatio-temporal and design spaces. The main contribution is to propose a parametric LSTM (pLSTM) method, which integrates a design gate with the other gates (such as input, output and forget gates) in a conventional LSTM cell. This design gate serves as a crucial link between the design space and dynamical system attractors. It further addresses challenges found in conventional LSTM models, such as fixed memory capacity and sensitivity to error accumulation, which can lead to instability of the predicted solution. By incorporating the design gate, the pLSTM significantly enhances learning memory capacity, enabling the learning of attractors across diverse design parameters.

First, we examined the proposed pLSTM and conventional LSTM on a low-dimensional dynamical system, the Van der Pol oscillator. Second, we explored a high-dimensional fluid dynamic problem with complex flow structures. For this case, the numerical set-up involved a two-dimensional flapping ellipse within the domain, simulated using a high-fidelity flow solver for the incompressible Navier–Stokes equations with moving boundaries. The Strouhal number, pitch amplitude and phase angle difference between heave and pitch are three design parameters that alter the kinematics of the flapping ellipse. The Reynolds number was set to $Re = 500$, and the ROM was trained with 53 different case studies across the design space. For both applications, pLSTM was trained on an entire parametric dataset and shown to match the predictive performance of an LSTM trained and evaluated at a single design parameter. However, pLSTM is unmatched in its ability to predict accurate solutions across the entire design space, including for unseen parameter vectors. For the flapping ellipse problem, specifically, the proposed ROM with pLSTM thoroughly captures primary flow structures and reproduces high-fidelity flow fields.

The proposed pLSTM approach is uniquely suited for outer-loop optimization and uncertainty quantification problems where spatio-temporal flow field dynamics are required across a parameter space. Examples are one-way coupled flow and flow-driven multi-physics problems, such as scalar mixing or optimal sensor placement. Further, the pLSTM approach can provide initial insights into two-way coupled problems, enabling quick traversal of the parameter space for such problems as fish schooling. The proposed pLSTM approach can also pave the way towards predicting two-way coupled problems, initiating new lines of research into architecture details and training strategies for coupled problems. Further, while we only focused here on a simple POD-based expression, pLSTM can naturally be combined with more efficient order-reduction approaches for dynamic problems, such as dynamic mode decomposition [56] and different types of autoencoders that bring interpretability of latent expressions [57,58]. Finally, though the main application considered here is in fluid flows, the pLSTM approach is general and thus holds promise for data-driven ROMs of dynamical systems across many different applications.

Data accessibility. The code and data used in this work are accessible in the repository https://zenodo.org/records/13993315.

Declaration of AI use. We have not used AI-assisted technologies in creating this article.

Authors' contributions. H.R.K.: conceptualization, investigation, methodology, software, validation, visualization, writing—original draft, writing—review and editing; W.V.R.: conceptualization, funding acquisition, methodology, supervision, validation, visualization, writing—review and editing.

Both authors gave final approval for publication and agreed to be held accountable for the work performed therein.

Conflict of interest declaration. We declare we have no competing interests.

Funding. This article received partial internal funding from the Department of Mechanical Engineering at Massachusetts Institute of Technology.

Acknowledgements. The authors gratefully acknowledge partial funding from the Department of Mechanical Engineering at Massachusetts Institute of Technology.

References

1. Brunton SL, Kutz JN. 2022 *Data-driven science and engineering: machine learning, dynamical systems and control*. Cambridge University Press. (doi:10.1017/9781009089517)
2. Gazzola M, van Rees WM, Koumoutsakos P. 2012 C-start: optimal start of larval fish. *J. Fluid Mech.* **698**, 5–18. (doi:10.1017/jfm.2011.558)
3. van Rees WM, Gazzola M, Koumoutsakos P. 2013 Optimal shapes for anguilliform swimmers at intermediate Reynolds numbers. *J. Fluid Mech.* **722**, R3. (doi:10.1017/jfm.2013.157)
4. van Rees WM, Gazzola M, Koumoutsakos P. 2015 Optimal morphokinematics for undulatory swimmers at intermediate Reynolds numbers. *J. Fluid Mech.* **775**, 178–188. (doi:10.1017/jfm.2015.283)

5. Novati G, Verma S, Alexeev D, Rossinelli D, van Rees WM, Koumoutsakos P. 2017 Synchronisation through learning for two self-propelled swimmers. *Bioinspir. Biomim.* **12**, 036001. (doi:10.1088/1748-3190/aa6311)
6. Karbasian HR, Vermeire BC. 2022 Application of physics-constrained data-driven reduced-order models to shape optimization. *J. Fluid Mech.* **934**, A32. (doi:10.1017/jfm.2021.1051)
7. Milano M, Gharib M. 2005 Uncovering the physics of flapping flat plates with artificial evolution. *J. Fluid Mech.* **534**, 403–409. (doi:10.1017/S0022112005004842)
8. Quinn DB, Lauder GV, Smits AJ. 2015 Maximizing the efficiency of a flexible propulsor using experimental optimization. *J. Fluid Mech.* **767**, 430–448. (doi:10.1017/jfm.2015.35)
9. Ramananarivo S, Mitchel T, Ristroph L. 2019 Improving the propulsion speed of a heaving wing through artificial evolution of shape. *Proc. R. Soc. A: Math. Phys. Eng. Sci.* **475**, 20180375. (doi:10.1098/rspa.2018.0375)
10. Lino M, Fotiadis S, Bharath AA, Cantwell CD. 2023 Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proc. R. Soc. A: Math. Phys. Eng. Sci.* **479**, 20230058. (doi:10.1098/rspa.2023.0058)
11. Brunton SL, Proctor JL, Kutz JN. 2016 Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl Acad. Sci. USA* **113**, 3932–3937. (doi:10.1073/pnas.1517384113)
12. Taira K, Hemati MS, Brunton SL, Sun Y, Duraisamy K, Bagheri S, Dawson ST, Yeh CA. 2020 Modal analysis of fluid flows: applications and outlook. *AIAA J.* **58**, 998–1022. (doi:10.2514/1.j058462)
13. Ghattas O, Willcox K. 2021 Learning physics-based models from data: perspectives from inverse problems and model reduction. *Acta Numer.* **30**, 445–554. (doi:10.1017/s0962492921000064)
14. Rempfer D. 2000 On low-dimensional Galerkin models for fluid flow. *Theoret. Comput. Fluid Dynamics.* **14**, 75–88. (doi:10.1007/s001620050131)
15. Noack BR, Papas P, Monkewitz PA. 2005 The need for a pressure-term representation in empirical Galerkin models of incompressible shear flows. *J. Fluid Mech.* **523**, 339–365. (doi:10.1017/s0022112004002149)
16. Carlberg K, Bou-Mosleh C, Farhat C. 2011 Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *Int. J. Num. Meth. Engng.* **86**, 155–181. (doi:10.1002/nme.3050)
17. Karbasian HR, Vermeire BC. 2022 Sensitivity analysis of chaotic dynamical systems using a physics-constrained data-driven approach. *Phys. Fluids* **34**. (doi:10.1063/5.0076074)
18. Carlberg K, Barone M, Antil H. 2017 Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction. *J. Comput. Phys.* **330**, 693–734. (doi:10.1016/j.jcp.2016.10.033)
19. Mohan A, Daniel D, Chertkov M, Livescu D. 2019 Compressed convolutional LSTM: an efficient deep learning framework to model high fidelity 3D turbulence. *arXiv preprint arXiv:1903.00033*.
20. Deshmukh R, McNamara JJ, Liang Z, Kolter JZ, Gogulapati A. 2016 Model order reduction using sparse coding exemplified for the lid-driven cavity. *J. Fluid Mech.* **808**, 189–223. (doi:10.1017/jfm.2016.616)
21. Sargsyan S, Brunton SL, Kutz JN. 2015 Nonlinear model reduction for dynamical systems using sparse sensor locations from learned libraries. *Phys. Rev. E* **92**, 033304. (doi:10.1103/physreve.92.033304)
22. Nair AG, Taira K. 2015 Network-theoretic approach to sparsified discrete vortex dynamics. *J. Fluid Mech.* **768**, 549–571. (doi:10.1017/jfm.2015.97)
23. Brunton SL, Noack BR, Koumoutsakos P. 2020 Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **52**, 477–508. (doi:10.1146/annurev-fluid-010719-060214)
24. Lumley JL. 1967 The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, pp. 166–178.
25. Schmid PJ. 2010 Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* **656**, 5–28. (doi:10.1017/s0022112010001217)
26. Sieber M, Paschereit CO, Oberleithner K. 2016 Spectral proper orthogonal decomposition. *J. Fluid Mech.* **792**, 798–828. (doi:10.1017/jfm.2016.103)
27. Lee Y, Yang H, Yin Z. 2017 PIV-DCNN: cascaded deep convolutional neural networks for particle image velocimetry. *Exp. Fluids* **58**, 1–10. (doi:10.1007/s00348-017-2456-1)
28. Fukami K, Fukagata K, Taira K. 2019 Super-resolution reconstruction of turbulent flows with machine learning. *J. Fluid Mech.* **870**, 106–120. (doi:10.1017/jfm.2019.238)

29. Obayashi W, Aono H, Tatsukawa T, Fujii K. 2021 Feature extraction of fields of fluid dynamics data using sparse convolutional autoencoder. *AIP Advances* **11**. (doi:10.1063/5.0065637)
30. Fukami K, Nakamura T, Fukagata K. 2020 Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. *Phys. Fluids* **32**. (doi:10.1063/5.0020721)
31. Wang J, He C, Li R, Chen H, Zhai C, Zhang M. 2021 Flow field prediction of supercritical airfoils via variational autoencoder based deep learning framework. *Phys. Fluids* **33**. (doi:10.1063/5.0053979)
32. Borrelli G, Guastoni L, Eivazi H, Schlatter P, Vinuesa R. 2022 Predicting the temporal dynamics of turbulent channels through deep learning. *arXiv preprint arXiv:2203.00974*. (doi:10.1016/j.ijheatfluidflow.2022.109010)
33. Rocha Ribeiro BL, Franck JA. 2021 A machine learning approach to classify kinematics and vortex wake modes of oscillating foils. In *AIAA Aviation 2021 Forum*, p. 2947. (doi:10.2514/6.2021-2947)
34. Mohan AT, Gaitonde DV. 2018 A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks. *arXiv preprint arXiv:1804.09269*.
35. Hochreiter S, Schmidhuber J. 1997 Long short-term memory. *Neural Comput.* **9**, 1735–1780. (doi:10.1162/neco.1997.9.8.1735)
36. Nakamura T, Fukami K, Hasegawa K, Nabae Y, Fukagata K. 2021 Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Phys. Fluids* **33**. (doi:10.1063/5.0039845)
37. Zhang J, Zhao X. 2020 A novel dynamic wind farm wake model based on deep learning. *Appl. Energy* **277**, 115552. (doi:10.1016/j.apenergy.2020.115552)
38. Han R, Wang Y, Zhang Y, Chen G. 2019 A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network. *Phys. Fluids* **31**. (doi:10.1063/1.5127247)
39. Vlachas PR, Arampatzis G, Uhler C, Koumoutsakos P. 2022 Multiscale simulations of complex systems by learning their effective dynamics. *Nat. Mach. Intell.* **4**, 359–366. (doi:10.1038/s42256-022-00464-w)
40. Wan ZY, Vlachas P, Koumoutsakos P, Sapsis T. 2018 Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one* **13**, e0197704. (doi:10.1371/journal.pone.0197704)
41. Noack BR, Morzynski M, Tadmor G. 2011 *Reduced-order modelling for flow control*, vol. 528. Springer Science & Business Media.
42. Hiriyannaiah S, Srinivas A, Shetty GK, Siddesh G, Srinivasa K. 2020 A computationally intelligent agent for detecting fake news using generative adversarial networks. In *Hybrid Computational Intelligence* , pp. 69–96. Elsevier. (doi:10.1016/b978-0-12-818699-2.00004-4)
43. Liu K, Zhang J. 2021 A dual-layer attention-based LSTM network for Fed-batch fermentation process modelling. In *31st European Symposium on Computer Aided Process Engineering*, Computer Aided Chemical Engineering, (eds Türkay M, Gani R), vol. 50, pp. 541–547. Elsevier. (doi:https://doi.org/10.1016/B978-0-323-88506-5.50086-3)
44. Bengio Y, Simard P, Frasconi P. 1994 Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* **5**, 157–166. (doi:10.1109/72.279181)
45. Ha D, Dai A, Le QV. 2016 Hypernetworks. *arXiv preprint arXiv:1609.09106*.
46. Pan S, Brunton SL, Kutz JN. 2023 Neural implicit flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data. *JMLR* **24**, 1–60.
47. Gabbard J, Gillis T, Chatelain P, van Rees WM. 2022 An immersed interface method for the 2D vorticity-velocity Navier–Stokes equations with multiple bodies. *J. Comput. Phys.* **464**, 111339. (doi:10.1016/j.jcp.2022.111339)
48. Ji X, Gabbard J, van Rees WM. 2023 A sharp immersed method for 2D flow-body interactions using the vorticity-velocity Navier–Stokes equations. *J. Comput. Phys.* **494**, 112513. (doi:10.1016/j.jcp.2023.112513)
49. Rhodes P, van Rees WM. Hydrodynamic forces on a side-by-side ellipse pair with and without relative motion. *FLOW*. In press. (doi:10.1017/flo.2024.21)
50. Wukie NA, Fidkowski K, Persson PO, Fujimoto TR, Wang ZJ, Gabbard J, van Rees WM. In *High-Fidelity CFD Verification Workshop 2024: Post-Workshop Mesh Motion Summary*. (doi:10.2514/6.2024-3696)
51. Mojgani R, Balajewicz M. 2017 Lagrangian basis method for dimensionality reduction of convection dominated nonlinear flows. *arXiv preprint arXiv:1701.04343*.

52. Reiss J, Schulze P, Sesterhenn J, Mehrmann V. 2018 The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena. *SIAM J. Sci. Comput.* **40**, A1322–A1344. (doi:10.1137/17m1140571)
53. Schulze P, Reiss J, Mehrmann V. 2019 Model reduction for a pulsed detonation combuster via shifted proper orthogonal decomposition. In *Active Flow and Combustion Control 2018: Papers Contributed to the Conference 'Active Flow and Combustion Control 2018', Berlin, Germany, 19–21 September*, pp. 271–286. Springer. (doi:10.1007/978-3-319-98177-2_17)
54. Mojgani R, Balajewicz M. 2021 Low-rank registration based manifolds for convection-dominated PDEs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35 pp. 399–407. (doi:10.1609/aaai.v35i1.16116)
55. Williamson CH, Roshko A. 1988 Vortex formation in the wake of an oscillating cylinder. *J. Fluid Struct.* **2**, 355–381. (doi:10.1016/s0889-9746(88)90058-8)
56. Schmid PJ. 2022 Dynamic mode decomposition and its variants. *Annu. Rev. Fluid Mech.* **54**, 225–254. (doi:10.1146/annurev-fluid-030121-015835)
57. Fukami K, Taira K. 2023 Grasping extreme aerodynamics on a low-dimensional manifold. *Nat. Commun.* **14**, 6480. (doi:10.1038/s41467-023-42213-6)
58. Smith L, Fukami K, Sedky G, Jones A, Taira K. 2024 A cyclic perspective on transient gust encounters through the lens of persistent homology. *J. Fluid Mech.* **980**, A18. (doi:10.1017/jfm.2024.16)