33RD CONGRESS
OF THE INTERNATIONAL COUNCIL
OF THE AERONAUTICAL SCIENCES
STOCKHOLM, SWEDEN, 4-9 SEPTEMBER, 2022

ICAS 2022 SWEDEN

COMPARING REDUCED ORDER MODEL FORMS FOR NONLINEAR DYNAMICAL SYSTEMS

David Massegur[1], Declan Clifford[2], Andrea Da Ronch[3] & Sean Symon[4]

[1]PhD Student, Faculty of Engineering and Physical Sciences, University of Southampton.
[2]PhD Student, Faculty of Engineering and Physical Sciences, University of Southampton.
[3]Associate Professor, Faculty of Engineering and Physical Sciences, University of Southampton.
[4]Lecturer, Faculty of Engineering and Physical Sciences, University of Southampton.

Abstract

The time domain solution of a chaotic system governed by a set of nonlinear equations is computationally expensive and ill suited for parametric searches. This work investigates the use of reduced order models to distill, both from data and equations, an equivalent but more advantageous mathematical representation. Two types of reduced order model are presented, data-driven, non-intrusive approaches and a model-derived, intrusive alternative. Three test cases are used for assessing the predictive capability of the models: a) Lorenz 1963 model; b) Moehlis model; and c) Lorenz 1996 model. Various key performance indices are selected to quantify the accuracy of the reduced order models, including over the short and long time scales. The small size of the test cases, up to 220 states for Lorenz 1996 model, prevented us from executing a projection of the reduced order models onto a smaller basis. Hence, the focus was on recovering the underlying governing equations and on the reconstruction of the physical features. For each reduced order model, details concerning the practical implementation and the model generation are also given.

Keywords: reduced-order modelling, machine learning, model-based method, system identification, nonlinear dynamical systems.

1. Introduction

Nonlinear dynamic phenomena are predominant across most aeronautical sciences, such as structural dynamics, unsteady flow structures or control systems. Nonlinear systems are typically difficult to solve and frequently entail large computational resources to solve high-fidelity simulations, as is the case with computational fluid-dynamics.

In a fast-moving world with high market demands, there is increasing urge for accelerating simulation turnarounds to enable engineers rapid-decision making in industrial processes and higher quality products. Hence there is an ever-growing interest in reduced order model (ROMs) to replace high-fidelity, slow simulations.

Work on ROMs represent a well-known research field aimed at projecting the response of high-fidelity models onto smaller mathematical spaces, allowing analysis of the system which is substantially cheaper in computational terms and only marginally less accurate.

Multiple ROM approaches exist, depending on the type of application [1, 2, 3]. Traditionally, ROMs may be classified in *physics-derived* or *data-driven* approaches, without being complementary. A physics-derived framework transforms the equations governing a particular problem into a mathematical structure that is computationally less expensive to solve. By contrast, a data-driven framework attempts to infer a mathematical representation from data (acquired through simulations or experiments), often where the governing equations are unknown. Recent advances in machine learning, particularly deep learning [4], are now at the forefront of the latter approach [5]. While the knowledge of the governing equations traditionally results in more efficient models, data-driven techniques are

more suited in experimental scenarios to investigate less characterised phenomena. However, their applicability may overlap in simulation tasks, as well as hybrid approaches can be thought to boost the modelling performance. As a result, we want to answer: what are the performance benefits and detriments of each approach? Can we derive a set of recipes to help the engineering community choose the best suited model for a given task?

In the present work, we propose novel ROMs belonging to the two mentioned frameworks and we validate them in common complex nonlinear dynamical systems. We therefore aim to share the experiences gained throughout the ROM generation process and in particular provide insight on their suitability against the different tasks.

This paper continues with Section 2. to present the different ROMs developed, followed by a description of the underlying methodology in Section 3.and the selected validation cases in Section 4. . Section 5.reports the results obtained with each ROM and finally, Section 6.provides crucial conclusions extracted from our thorough investigation.

2. Reduced Order Models

In this Section, we introduce the ROM approaches we have developed for this study. We have chosen two ROMs for the comparison: one is built solely using data; and one is obtained directly from the governing equations after adequate assumptions are made.

2.1 Sparse System Identification

System identification is the approach to derive physical equations from gathered data. It is a central task for scientists and engineers to derive governing equations from experimentation, where the underlying physics are unknown or incomplete. Identification methods are more sophisticated than classic regression techniques because the mathematical structure is discovered by the model itself, rather than arbitrarily chosen beforehand. Such advantages, however, comes at the expense of a larger computational burden. Thus, a balance between computing power and model complexity is essential [6].

Recent advances in data-driven algorithms provide an excellent platform for complex identification tasks. Herein, the idea is to construct a machine-learning algorithm which infers interpretable mathematical expressions that best fit the measured data. Furthermore, to alleviate the computational burden, a plausible assumption is that most physics phenomena are described by compact expressions. Compactness can be achieved by promoting sparsity amongst the term space. A model with sparse mathematical form results in significant computational gains. The problem is that classic regularisation techniques, such as the Lasso L_1 [7, 8], are inadequate for identification purposes because these tend to dampen the larger parameters, causing incorrect discovery and loss of physics representation. A sparse machine-learning based identification method for non-linear dynamical systems was presented by Brunton et al. [9, 10]. They developed an efficient technique but it does not scale with dimensionality and is ill-suited to problems with multiple feasible solutions.

Herein, the first of the data-driven methods consists of a single-layer deep-learning framework, especially suited to high dimensionality and multiple solutions. Furthermore, we introduce a successful sparsity technique: a threshold filter constraint is included during the optimisation, which allows the optimiser to disable or enable each parameter at convenience.

2.2 Neural Networks for Spatio-Temporal Forecasting

A problem with the system identification approach is that it becomes impractical for systems of large dimensionality, due to the large number of parameters to identify. This is the case for our third validation problem, the Lorenz 1996 system. In such a scenario, it is more adequate to leverage more sophisticated deep-learning techniques, i.e. neural networks, to generate a lower-order approximate model. Neural networks are data-driven algorithms capable of predicting highly nonlinear systems at large scale, such as fluid-flow analysis.

Although neural-networks are especially designed to reconstruct large systems, it is common in literature to validate the proposed architectures with validation cases such as the nonlinear systems dealt in this paper, featuring a smaller-order form that replicate realistic flow phenomena. Zhang et al. studied generalisation performance of two different fully-connected neural networks (FCNN) with

small segments of the Lorenz solution [11]. Brajard et al. [12] proposed a convolutional neural network (CNN) architecture for the prediction of the Lorenz 1996 system, featuring a bilinear layer to include weighted quadratic terms and embedded in a data assimilation approach to account for noisy and unevenly distributed data. Fukami et al. [13] proposed a system identification method to reconstruct the Lorenz and Moehlis systems. Their method was subsequently embedded to a CNN-based autoencoder to predict the unsteady wake of a two-dimensional cylinder.
For spatio-temporal forecasting problems, an adequate architecture is the convolutional-based long short-term memory (Conv-LSTM). This is described in Section 3.2The advantage of such architecture lies on the deployment of convolutional operations, especially designed for spatial domains and the time-series prediction of the long short-term memory (LSTM) recurrent approach. The Conv-LSTM has successfully been adopted in other applications. For example, Mohan et al. [14] developed a forecasting model based on such architecture for the prediction of 3D turbulence. Han et al. [15] embedded a Conv-LSTM layer in a CNN-based autoencoder for the unsteady prediction of the wake behind a 2D cylinder.
In the present study, we propose a Conv-LSTM based architecture for the forecasting of large-scale nonlinear dynamical systems. This is the third case described in Section 4.3 To our knowledge, this is the first time this architecture has been adopted to solve such a problem.

2.3 Intrusive Techniques

Intrusive techniques are ROM methodologies which involve manipulating the governing full order equations into more compact and less computationally expensive systems. The approach is advantageous in that it produces a ROM which is guaranteed to be representative of the dynamics of the original system. However, manipulating the governing equations is often a non-trivial task, and furthermore, a closed access code will mean no ROM can be produced.
The intrusive technique used in this paper follows the approach in [1] which saw the formulation of a systematic procedure able to produce linear and nonlinear reduced order models. This approach has been used successfully to generate ROMs for flexible aircraft control design [16], and for gust load analysis [17]. Here, the method is adapted to make use of automatic differentiation, which avoids the round–off and truncation errors typical of finite–differencing and furthermore, enables generation of the nonlinear terms such that their evaluation is parametric with respect to the system parameters.
The method works by projecting a Taylor expansion of the full order equations onto a reduced basis of eigenvectors which are representative of the full-order dynamics. The method is referred to as the Taylor-Projection Reduced Order Model (TPROM) from hereon. The workflow in Fig. 1 provides a high-level overview of the TPROM methodology. It is important to stress that this high-level overview does not describe many intricacies involved in each step.
The TPROM is advantageous as ROMs of an arbitrary order in the perturbation variables can be generated by truncating the reduced basis up to a certain frequency content, in a similar manner to what is commonly done in modal analysis of structures. The ability to generate a TPROM that is parametric in one or more parameters is appealing as it provides a single model with the ability to capture the effect of changes in system parameters at a substantially reduced cost. A further advantage is that the approach does not require any data to be generated a–priori, bypassing the enormous task to generate or collect a database for data-driven models. This second point also means the method is not affected by noise, which any approach using a data-driven technique would suffer from. However, it should be noted that the parameterisation as presented in this work assumes a suitably rapid solution of the eigenvalue problem of the system Jacobian for any parameter change. Furthermore, it assumes the rapid reevaluation of the nonlinear terms for any parameter change. While the latter issue can be assisted through application of automatic differentiation, the former issue is prohibitive for high–dimensional systems. These issues are discussed in due course.
The challenge of using this technique arises mostly in the integration with the full order model code. Step 4.2 in Fig. 1 requires matrix-free finite differencing or automatic differentiation operations integrated with the full order code which is non-trivial. Reference [16] describes the method with higher order derivatives generated using matrix-free finite differencing. Here, these operations are replaced with automatic differentiation, using the `ADiGator MATLAB` toolbox [18]. Similar toolboxes are available for a wide variety of languages. `ADiGator` also provides a convenient way to parameterise

ROM terms through vectorised inputs to the derivative code.

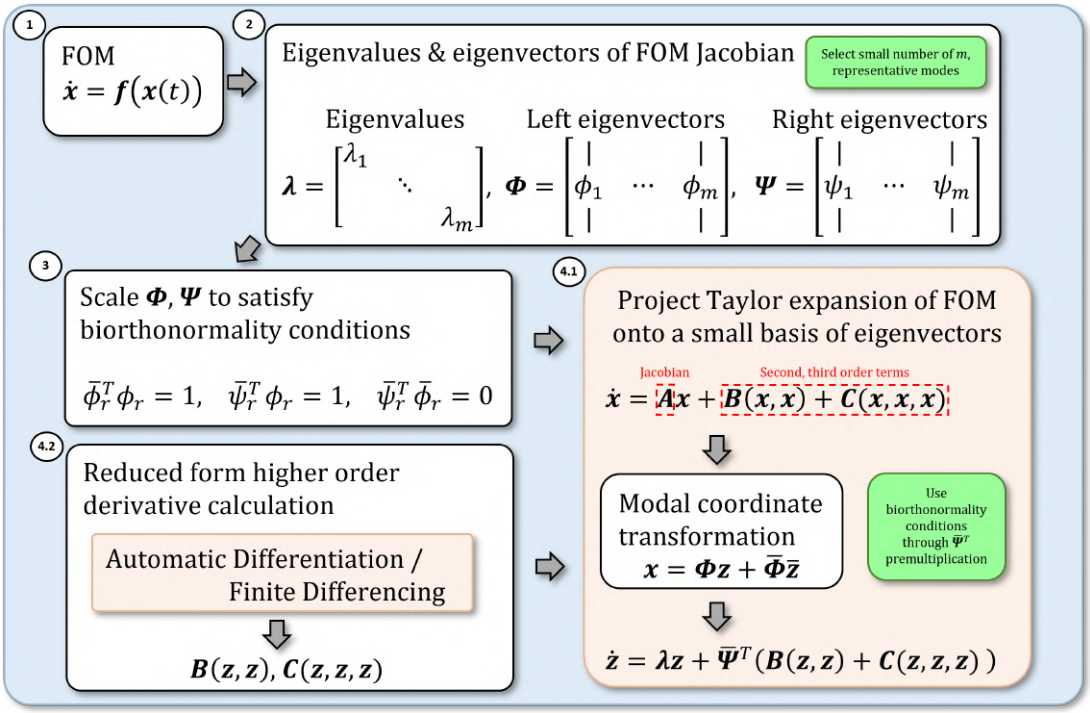


Figure 1 – TPROM workflow for dynamical systems.

3. Methodology

This Section provides the methodology and implementation details for the ROMs used in this work.

3.1 Data-Driven Machine-Learning Identification Model

We now introduce a data-driven and machine-learning technique to identify the terms governing a non-linear dynamical system, which we will denote as Machine-Learning System Identification (MLSI). A dynamical system can be expressed in the form:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ are the n-dimensional system states and $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^n$ are the governing equations that describe the physics system. The system is subject to initial conditions $\mathbf{x}(0) = \mathbf{x_0}$. The derivative $\frac{d}{dt}\mathbf{x}(t)$ can also be expressed as $\dot{\mathbf{x}}(t)$.

When $\mathbf{f}(\mathbf{x})$ is unknown, a data-driven tasks consists of finding an approximative functional $\hat{\mathbf{f}}(\mathbf{x})$ from a dataset containing discrete m samples of the system's states and the corresponding derivatives, $\mathscr{D} = \{\mathbf{x}_k, \dot{\mathbf{x}}_k \,|\, k = 1, ..., m\}$. The idea is to leverage machine learning [19] to infer $\hat{\mathbf{f}}(\mathbf{x})$ which best maps between the sampled inputs, $\mathbf{x}_k$, and the corresponding outputs, $\dot{\mathbf{x}}_k$. Such mapping is achieved via minimisation of a loss function $\mathscr{L}$, for example, the mean of the squared euclidean error of all sampled points [20]:

$$\min \mathscr{L} = \frac{1}{m}\sum_{k=1}^{m} ||\hat{\mathbf{f}}(\mathbf{x}_k) - \dot{\mathbf{x}}_k||_2^2 \tag{2}$$

At this point, particular consideration must be taken on the form that $\hat{\mathbf{f}}(\mathbf{x})$ should take. Indeed, no a-priori knowledge of the underlying law $\mathbf{f}(\mathbf{x})$ mandates infinite combinations of terms composing the approximate expression, but this is computationally prohibitive. To limit model size, it is reasonable to consider that many known dynamical systems are typically described by few components. This crucial assumption is achieved by promoting sparsity within the basis functions of our algorithm.

A function emanated from a machine-learning algorithm is typically composed by a chain of linear and non-linear functions, whose depth depends on the complexity of the architecture chosen. However, for the identification of simple dynamical systems, a FIR linear regression [21] may be adopted:

$$\hat{\mathbf{f}}(\mathbf{x}(t)) = \mathbf{W}^T \mathbf{x}(t) + \mathbf{b} \tag{3}$$

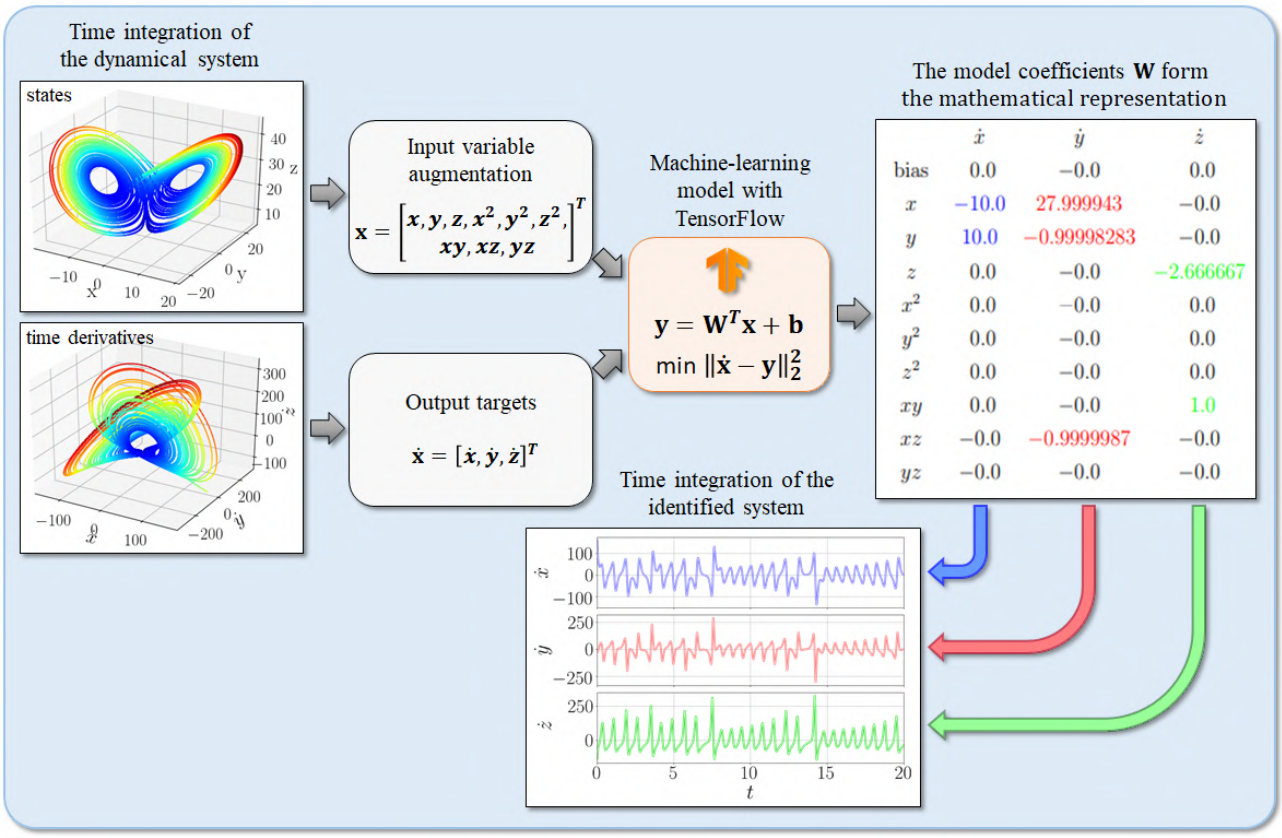
Where $\mathbf{x}(t)$ is the state of the variables at the time step t, $\mathbf{W}$ is the matrix of weights or parameters which creates the linear mapping between the input variables and the output functions $\hat{\mathbf{f}}(\mathbf{x}(t))$, and $\mathbf{b}$ are the constant terms.
Non-linear behavior in the FIR algorithm is introduced via term augmentation of the states $\mathbf{x}$. For example, adopting polynomial or trigonometric expansions. In the three-dimensional space $n=3$ with state variables $\{x,y,z\}$, the synthetic augmentation can yield the following basis functions:

$$\mathbf{x} = [x, y, z, x^2, y^2, z^2, xy, xz, yz, x^3, \ldots, x^2y, \ldots, x^n, \ldots, \sin(x), \cos(x), x^2\sin(x)\ldots]^T \tag{4}$$

The dimension of the regression expression is still considerably large. Therefore, to reduce the identification burden, an arbitrary choice of the basis functions can be introduced. With such choice, there is increasing risk the model may fail to extract the correct governing equations.
We now describe the process adopted in our MLSI, schematised in Figure 2 Panel (a). System identification is adequate for describing experimental data of an unknown phenomenon. In the present work, the source of the training dataset is from time integrating the governing equations of the three test cases. The states at each time step were fed as inputs to the model to predict the resulting time derivatives. The computational graph adopted for the linear regression is shown in Panel (b) of Fig. 2. The first-order gradient-based algorithm Adam [22] was chosen to minimise the Huber loss function [20]. As a result, back-propagation was used to compute the gradients of the cost function with respect to the learnable weights [4, 19]. The inputs were normalised to enhance the training. The model weights were subject to a high-pass filter to set to zero the least contributing terms and promote sparsity. The machine-learning system was implemented in `TensorFlow/Keras` [23, 24], an optimised deep-learning library in `Python` code developed by `Google`.

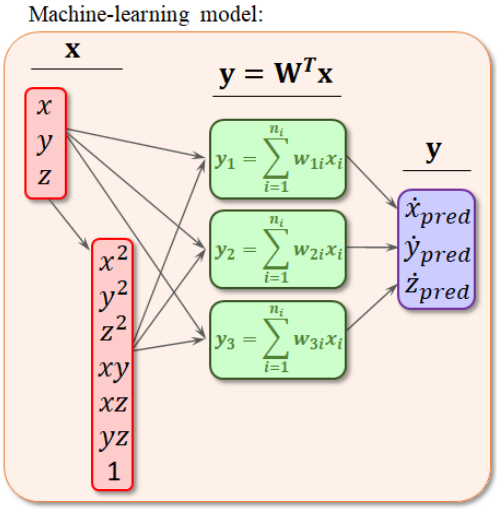
3.2 Convolutional Long Short-Term Memory

For the prediction of large and nonlinear dynamical systems, such as the Lorenz 1996 system (test case 3), a convolutional-based long short-term memory (Conv-LSTM) model is used [25]. Figure 3 illustrates the schematics of the Conv-LSTM model framework. This model was validated with the multi-scale Lorenz 1996 test case. Panel (a) shows the time-marching scheme adopted. The idea is to arrange the input as a sequence of the states at the current steps and a chosen number of time delays n. The machine-learning based model finds the best functional to predict the value of the states at the next time step. The output is then used to update the input sequence to execute the following prediction and step in time.
In the current work, the model comprises a Conv-LSTM [25]. Convolutional neural networks are designed to analyse evenly discretised spatial domains, where the state at a given location is only influenced by neighboring points. On the other side, LSTM layers are tailored to time-evolved predictions [26]. LSTM is an advanced recurrent neural network, designed to execute sequential predictions based on selected past history of the response. A recurrent layer takes as inputs a sequence of data ordered in time and two additional states, the hidden state carrying the information from the previous timestamp (short-term memory) and a cell stat carrying long-term information. Additionally, the layer incorporates two gates to select or discard the memory from those states [27]. LSTMs are widely used for time-series because of alleviating the numerical instabilities typical in standard recurrent networks.
A problem with the standard LSTM is that they adopt fully-connected operators. As a result, they are impractical in large spatial domains. Therefore, the combined Conv-LSTM architectures have become especially adequate for spatio-temporal predictions [15].
The large-scale ODE system of the Lorenz 1996 study case, can be conceived as a discretised one-dimensional space. As a result, we adopted one-dimensional convolutional-based LSTM (Conv1D-



	$\dot{x}$	$\dot{y}$	$\dot{z}$
bias	0.0	−0.0	0.0
x	−10.0	27.999943	−0.0
y	10.0	−0.99998283	−0.0
z	0.0	−0.0	−2.666667
x^2	0.0	−0.0	0.0
y^2	0.0	−0.0	0.0
z^2	0.0	−0.0	0.0
xy	0.0	−0.0	1.0
xz	−0.0	−0.9999987	−0.0
yz	−0.0	−0.0	−0.0

(a) Data-driven identification workflow.



(b) Computational graph adopted in the machine-learning-based identification algorithm.

Figure 2 – Machine-Learning System Identification workflow of dynamical systems.

LSTM) layers. In vector (1D) form, the discrete convolution operation is defined as [4]

$$\text{1D convolution}: \quad (f \star w)_i \stackrel{\text{def}}{=} \sum_p f(x_{i+p})\, w_p \tag{5}$$

where $\mathbf{x} \in \mathbb{R}^d$, $f(x_i)$ is the value of the inputs at the position x_i, and $\mathbf{w}$ denotes the kernel function or filter of weights of the chosen size. The formulation for the embedded Conv-LSTM layer is provided in Appendix A and the reader is referred to the original paper [25], as well as the original paper for

the standard LSTM [26].
The architecture adopted to model large spatio-temporal systems is illustrated in Panel (b) of Figure 3. The model takes a sequence of states of a chosen number of contiguous time steps. The inputs are standardised using the average over the whole dataset and then normalised with the standard deviation. Furthermore, a periodic padding is added at the extremes of the first sequence fed to the first Conv1D-LSTM layer. The discrete convolution operation implies a loss of spatial resolution, depending on the size of the filter. To avoid this, input-data sequences are extended at the extremes, operation known as padding. In the present work, we propose a bespoke periodic padding before the first layer, consisting of padding the data from the opposite extreme. Sufficient padding is added initially so no additional padding is required on the rest of the network layers so the output spatial dimension is the same as the original. Schematic examples of a periodic padding and a convolution operation Eq. (5) are illustrated in Panel (c) of the same Figure.
The core of the model architecture comprises Conv1D-LSTM layers of different number of channels (filters) in each layer. The hyperbolic tangential operation is adopted as activation at the output of all the layers except the last one. In the last layer, the sequence is reduced to a single time step. The output represents the values of the states at the next time step.
In the present work, we used `TensorFlow/Keras`*. The Huber was adopted as loss function [20] and the Adam algorithm [22], as the gradient-descent optimiser.

3.3 Taylor-Series Projected ROM

Here the methodology for the TPROM approach is introduced. The TPROM is able to generate a ROM of any dynamical system taking the form of a time–invariant, multi–input and multi–output, nonlinear dynamical system. This is written in state–space form as

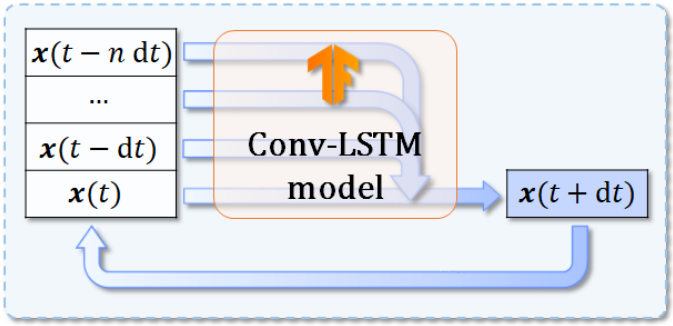
$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t)) \\ \mathbf{y} = \mathbf{L}^T \mathbf{x}(t) \end{cases} \tag{6}$$

with initial condition $\mathbf{x}(0) = \mathbf{x}_0$. Here, t is the time variable, $\mathbf{x}(t) \in \mathscr{R}^n$ is a state vector, $\mathbf{f} : \mathscr{R}^n \to \mathscr{R}^n$ is the state evolution function, and $\mathbf{y} \in \mathscr{R}^q$ is the output vector. The matrix $\mathbf{L} \in \mathscr{R}^{n \times q}$ relates the measurement outputs to the input vector. The symbol n is the state–space dimension, and p and q are the number of inputs and outputs, respectively. In most practical cases, it can be assumed that p and q are much smaller than n.
It is assumed that the nonlinear residual $\mathbf{f}$ is smooth, i.e. $\mathscr{C}^\infty$, and has an equilibrium at $\mathbf{x}_{eq}$, i.e. $\mathbf{f}(\mathbf{x}_{eq}) = \mathbf{0}$. Without loss of generality, the equilibrium is taken to be $\mathbf{0}$.
The desired attributes of reduced–order modelling of the nonlinear dynamical system (6) include replacing the large–scale system by a system of the same type but with a much smaller state–space dimension such that it has an admissible error between the full–order model and the reduced–order model. Furthermore, the reduced–order model should also preserve essential properties of the full–order system. Such a reduced–order model would let designers efficiently analyse and synthesize the dynamical behaviour of the original system within a tight design cycle. Specifically, given the nonlinear dynamical system of Eq. (6), a reduced–order nonlinear system of the form

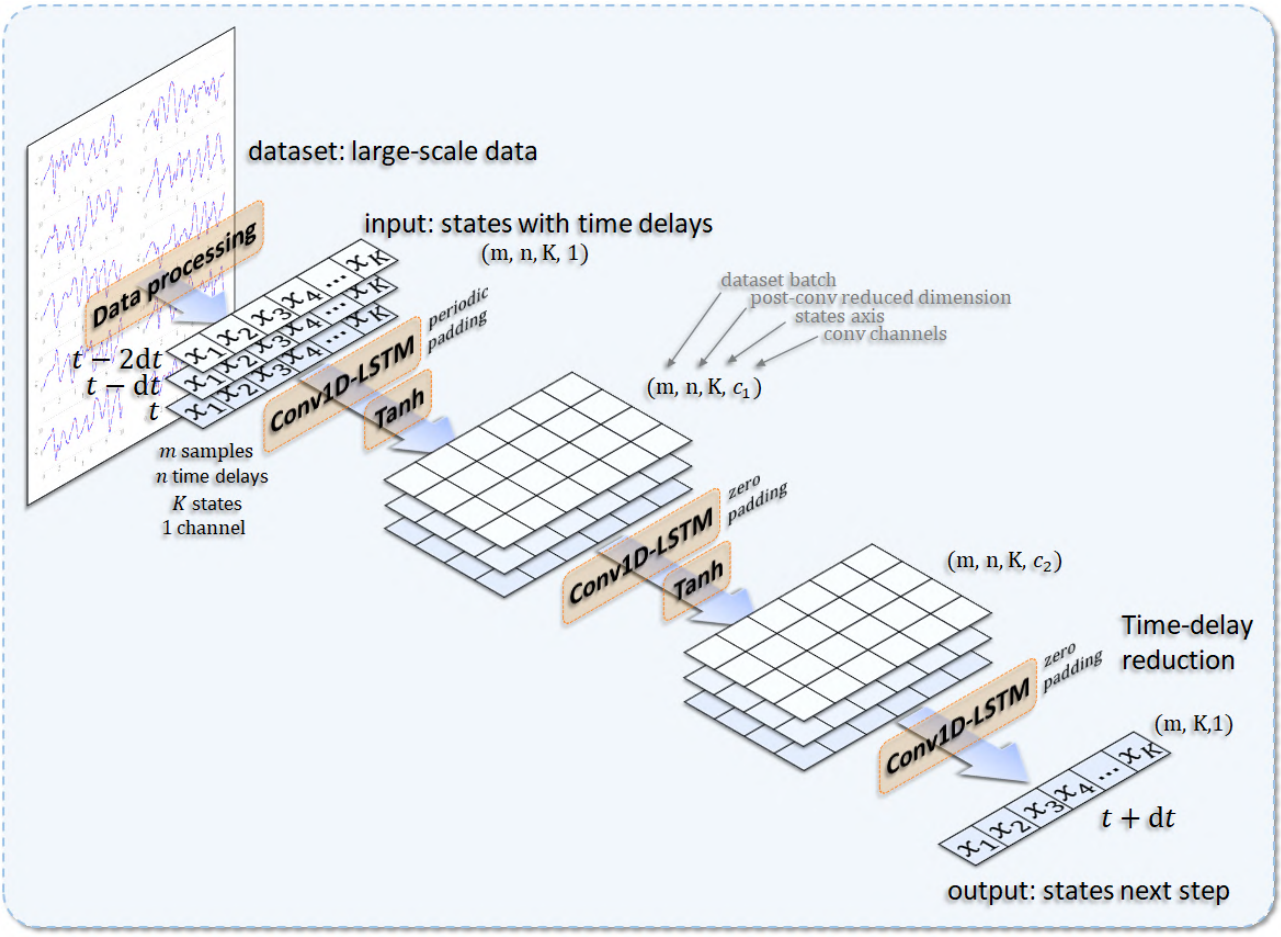
$$\begin{cases} \dot{\mathbf{z}} = \mathbf{f}_m(\mathbf{z}(t)) \\ \tilde{\mathbf{y}} = \mathbf{L}_m^T \mathbf{z}(t) \end{cases} \tag{7}$$

is desired to be found, where $\mathbf{z}(t) \in \mathscr{C}^m$ is the state vector of the reduced–order model, $\mathbf{L}_m \in \mathscr{C}^{m \times q}$, and $\mathbf{f}_m : \mathscr{C}^m \to \mathscr{C}^m$ is an approximation to the nonlinear state evolution function. The state–space dimension m should generally be much smaller than the state–space dimension n of Eq. (6), i.e. $m \ll n$.
It is now shown how to construct a reduced–order model of the nonlinear system (6) in the time domain for transient analysis.

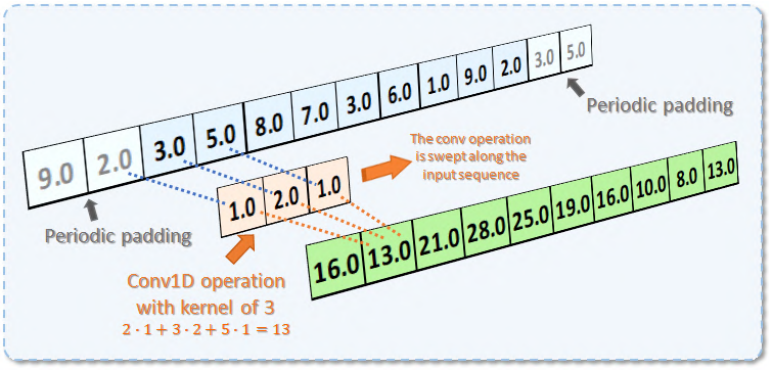
*`https://www.tensorflow.org/`



(a) Time-marching scheme for data-driven spatio-temporal simulations.



(b) Conv-LSTM based model architecture.



(c) Periodic padding and discrete convolution operations.

Figure 3 – Convolutional-based LSTM framework for the spatio-temporal prediction of large dynamical systems.

3.3.1 How to Obtain a Reduced Basis

The large–order system is projected onto a subspace formed by a small number m of eigenvectors of the Jacobian matrix evaluated at the equilibrium point. Given a Jacobian matrix $\mathbf{A}_1 \in \mathscr{R}^{n\times n}$, suitable basis vectors are obtained by solving the right and left eigenvalue problems, respectively

$$\mathbf{A}\,\phi_r = \lambda_r\,\phi_r, \quad \mathbf{A}_1^T\,\psi_r = \lambda_r\,\psi_r \quad \text{for } r = 1,\ldots,n \tag{8}$$

The eigenvalues of $\mathbf{A}$ are the same as the eigenvalues of $\mathbf{A}^T$, whereas the eigenvectors of $\mathbf{A}$ are different from the eigenvectors of $\mathbf{A}^T$. If all eigenvalues are distinct, the right and left eigenvectors corresponding to different eigenvalues are biorthogonal ($\bar{\psi}_r^T\,\phi_s = 0$ for all $r \neq s$). It is suggested to construct these eigenvectors to satisfy the following biorthogonality conditions (see Appendix B)

$$\begin{array}{l} \bar{\phi}_r^T\,\phi_r = 1 \\ \bar{\psi}_r^T\,\phi_r = 1 \\ \bar{\psi}_r^T\,\bar{\phi}_r = 0 \end{array} \quad \text{for } r = 1,\ldots,n \tag{9}$$

If these properties are satisfied, then it can be viewed that the projection of the Jacobian matrix on the left and right eigenvectors yields the following relations

$$\bar{\psi}_r^T\,\mathbf{A}\,\phi_r = \lambda_r, \quad \bar{\psi}_r^T\,\mathbf{A}\,\bar{\phi}_r = 0 \quad \text{for } r = 1,\ldots,n \tag{10}$$

A rational choice to extract a small number m of basis vectors is to retain only the slow modes since they are likely to dominate the system dynamics, with the exception of unstable fast modes leading to the instability of the system. The diagonal matrix of eigenvalues, and the right and left modal matrices are written

$$\text{diag}\,(\lambda) = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{bmatrix}, \quad \Phi = \begin{bmatrix} | & & | \\ \phi_1 & \ldots & \phi_m \\ | & & | \end{bmatrix}, \quad \Psi = \begin{bmatrix} | & & | \\ \psi_1 & \ldots & \psi_m \\ | & & | \end{bmatrix} \tag{11}$$

The r–th column of the matrices Φ and Ψ contains the right and left eigenvector, respectively, associated with the eigenvalue λ_r.

It is important to note that for a purely real eigenvalue and its corresponding eigenvectors, the biorthogonality conditions cannot be met. This is since $\phi = \bar{\phi}$. This requires the eigenvectors to instead be scaled according to

$$\bar{\psi}_r^T\,\phi_r = \frac{1}{2} \tag{12}$$

which then leads to the relation

$$\bar{\psi}_r^T\,\mathbf{A}\,\phi_r = \frac{\lambda_r}{2} \tag{13}$$

The left and right modal matrices contain the desired information about the dynamics of the system. An approximation of the state vector $\mathbf{x}(t)$ can be considered with another state vector, constrained to stay in the subspace spanned by the columns of Φ,

$$\mathbf{x} \approx \Phi\mathbf{z} + \bar{\Phi}\bar{\mathbf{z}} \tag{14}$$

for some $\mathbf{z}(t) \in \mathscr{C}^m$. The complex conjugate of $\mathbf{z}$ is denoted by $\bar{\mathbf{z}}$. Substituting the transformation of coordinates (14) into the nonlinear large–order system (6) yields an over–determined system of equations with respect to the state vector $\mathbf{z}$

$$\begin{cases} \Phi\dot{\mathbf{z}} + \bar{\Phi}\dot{\bar{\mathbf{z}}} = \mathbf{f}(\mathbf{z}(t)) \\ \tilde{\mathbf{y}} = \mathbf{L}^T\left(\Phi\mathbf{z} + \bar{\Phi}\bar{\mathbf{z}}\right) \end{cases} \tag{15}$$

After left–multiplying the first equation by Ψ^T, we have

$$\begin{cases} \Psi^T\Phi\dot{\mathbf{z}} + \Psi^T\bar{\Phi}\dot{\bar{\mathbf{z}}} = \Psi^T\mathbf{f}(\mathbf{z}(t)) \\ \tilde{\mathbf{y}} = \mathbf{L}^T\left(\Phi\mathbf{z} + \bar{\Phi}\bar{\mathbf{z}}\right) \end{cases} \tag{16}$$

Then, recalling the biorthogonality conditions (9), an m–th reduced–order model of the nonlinear system in Eq. (6) in the time domain is defined as

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{f}_m(\mathbf{z}(t)) \\ \tilde{\mathbf{y}} = \mathbf{L}^T 2\left(\Re(\Phi)\,\Re(\mathbf{z}) - \Im(\Phi)\,\Im(\mathbf{z})\right) \end{cases} \tag{17}$$

where $\mathbf{f}_m = \Psi^T \mathbf{f}$. The symbols $\Re(\bullet)$ and $\Im(\bullet)$ represent the real and imaginary parts, respectively.

3.3.2 How to Retain Nonlinearities

We discuss here an approach to create an approximation of the nonlinear state evolution function. The approach supposes that Taylor's series expansion of $\mathbf{f}(\mathbf{x})$ about the equilibrium point $\mathbf{0}$ is written

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}\,\mathbf{x} + \mathbf{B}(\mathbf{x},\mathbf{x}) + \mathbf{C}(\mathbf{x},\mathbf{x},\mathbf{x}) + \ldots \tag{18}$$

where $\mathbf{A} \in \mathscr{R}^{n\times n}$ is the Jacobian or the first derivative matrix of $\mathbf{f}$, and $\mathbf{B} \in \mathscr{R}^n$ and $\mathbf{C} \in \mathscr{R}^n$ are, respectively, the second and third order expansion terms. Using Einstein notation, one has

$$\mathbf{B}(\mathbf{x},\mathbf{x}) = \frac{1}{2!}\left.\frac{\partial^2 \mathbf{f}}{\partial x_r\,\partial x_s}\right|_{\mathbf{0}} x_r x_s = \frac{1}{2!}\,\partial^x_{rs}\mathbf{f}|_{\mathbf{0}}\,x_r x_s \tag{19}$$

and

$$\mathbf{C}(\mathbf{x},\mathbf{x},\mathbf{x}) = \frac{1}{3!}\left.\frac{\partial^3 \mathbf{f}}{\partial x_r\,\partial x_s\,\partial x_t}\right|_{\mathbf{0}} x_r x_s x_t = \frac{1}{3!}\,\partial^x_{rst}\mathbf{f}|_{\mathbf{0}}\,x_r x_s x_t \tag{20}$$

where $\partial^x_{rs}\mathbf{f} \in \mathscr{R}^{n\times n\times n}$ is the Hessian or the second derivative of $\mathbf{f}$, and $\partial^x_{rst}\mathbf{f} \in \mathscr{R}^{n\times n\times n\times n}$ is the third order derivative of $\mathbf{f}$. The approximation of the large–scale nonlinear system, which retains the original size, is obtained by using the expansion of $\mathbf{f}$

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{B}(\mathbf{x},\mathbf{x}) + \mathbf{C}(\mathbf{x},\mathbf{x},\mathbf{x}) \\ \tilde{\mathbf{y}} = \mathbf{L}^T\,\mathbf{x}(t) \end{cases} \tag{21}$$

The reduced–order model is obtained by substituting the expansion of Eq. (18) into Eq. (17). It is assumed herein that the eigenvectors are scaled according to Eq. (9). It can be immediately observed that a linear reduced–order model is obtained by using only the first term of the expansion, $\mathbf{A}_1\,\mathbf{x}$. This term is left–multiplied by Ψ^T and will generate a diagonal matrix with elements equal to the eigenvalues, $\mathrm{diag}(\lambda) \in \mathscr{C}^m$, yielding a linear reduced–order model of the form

$$\dot{\mathbf{z}} = \mathrm{diag}(\lambda)\,\mathbf{z} \tag{22}$$

In the following, two approaches for the derivation of the nonlinear terms in the reduced–order model that arise from the second and third order expansion terms, $\mathbf{A}_2$ and $\mathbf{A}_3$, respectively are discussed. The first approach involves obtaining the second and third derivatives of $\mathbf{f}$ with respect to the state vector $\mathbf{x}$. Once the derivatives $\partial^x_{rs}\mathbf{f}$ and $\partial^x_{rst}\mathbf{f}$ are extracted, the state vector $\mathbf{x}$ is replaced by $\mathbf{z}$ via the transformation of coordinates. Considering that the r–th element of $\mathbf{x}$

$$x_r = \phi_{r,:}\,\mathbf{z} + \bar{\phi}_{r,:}\,\bar{\mathbf{z}} \tag{23}$$

where $\phi_{r,:}$ is the r–th row of Φ, the second and third order expansion terms are of the form

$$\mathbf{B}(\mathbf{z},\mathbf{z}) = \frac{1}{2!}\left.\frac{\partial^2 \mathbf{f}}{\partial x_r\,\partial x_s}\right|_{\mathbf{0}} \left(\phi_{r,:}\,\mathbf{z} + \bar{\phi}_{r,:}\,\bar{\mathbf{z}}\right)\left(\phi_{s,:}\,\mathbf{z} + \bar{\phi}_{s,:}\,\bar{\mathbf{z}}\right) \tag{24}$$

and

$$\mathbf{C}(\mathbf{z},\mathbf{z},\mathbf{z}) = \frac{1}{3!}\left.\frac{\partial^3 \mathbf{f}}{\partial x_r\,\partial x_s\,\partial x_t}\right|_{\mathbf{0}} \left(\phi_{r,:}\,\mathbf{z} + \bar{\phi}_{r,:}\,\bar{\mathbf{z}}\right)\left(\phi_{s,:}\,\mathbf{z} + \bar{\phi}_{s,:}\,\bar{\mathbf{z}}\right)\left(\phi_{t,:}\,\mathbf{z} + \bar{\phi}_{t,:}\,\bar{\mathbf{z}}\right) \tag{25}$$

respectively. One critical issue associated with this approach is the rapid growth of the dimension of the higher order derivatives of $\mathbf{f}$. For example, the dimension of the second derivative $\partial^x_{rs}\mathbf{f}$ is $\mathscr{O}(n^3)$ and the dimension of the third derivative $\partial^x_{rst}\mathbf{f}$ is $\mathscr{O}(n^4)$. However, these derivatives are symmetric

due to the equality of mixed partials and generally extremely sparse, so one can exploit these facts and produce a useful reduced–order model

$$\dot{\mathbf{z}} = \text{diag}\left(\lambda\right)\mathbf{z} + \bar{\Psi}^T\left(\mathbf{A}_2\left(\mathbf{z},\mathbf{z}\right) + \mathbf{A}_3\left(\mathbf{z},\mathbf{z},\mathbf{z}\right)\right) \tag{26}$$

The second approach is intended to explicitly incorporate the higher order derivatives of $\mathbf{f}$ with respect to the state vector $\mathbf{z}$ into the construction of a nonlinear reduced order model. The second order expansion term is

$$\mathbf{B}^z\left(\mathbf{z},\mathbf{z}\right) = \frac{1}{2!}\left(\left.\frac{\partial^2\mathbf{f}}{\partial z_r\,\partial z_s}\right|_{\mathbf{0}} z_r z_s + \left.\frac{\partial^2\mathbf{f}}{\partial \bar{z}_r\,\partial z_s}\right|_{\mathbf{0}} \bar{z}_r z_s + \left.\frac{\partial^2\mathbf{f}}{\partial z_r\,\partial \bar{z}_s}\right|_{\mathbf{0}} z_r \bar{z}_s + \left.\frac{\partial^2\mathbf{f}}{\partial \bar{z}_r\,\partial \bar{z}_s}\right|_{\mathbf{0}} \bar{z}_r \bar{z}_s\right) \tag{27}$$

and the third order expansion term is written

$$\begin{aligned}\mathbf{C}^z\left(\mathbf{z},\mathbf{z},\mathbf{z}\right) = \frac{1}{3!}\Bigg(&\left.\frac{\partial^3\mathbf{f}}{\partial z_r\,\partial z_s\,\partial z_t}\right|_{\mathbf{0}} z_r z_s z_t + \left.\frac{\partial^3\mathbf{f}}{\partial z_r\,\partial z_s\,\partial \bar{z}_t}\right|_{\mathbf{0}} z_r z_s \bar{z}_t + \left.\frac{\partial^3\mathbf{f}}{\partial z_r\,\partial \bar{z}_s\,\partial z_t}\right|_{\mathbf{0}} z_r \bar{z}_s z_t + \left.\frac{\partial^3\mathbf{f}}{\partial \bar{z}_r\,\partial z_s\,\partial z_t}\right|_{\mathbf{0}} \bar{z}_r z_s z_t + \\ &\left.\frac{\partial^3\mathbf{f}}{\partial z_r\,\partial \bar{z}_s\,\partial \bar{z}_t}\right|_{\mathbf{0}} z_r \bar{z}_s \bar{z}_t + \left.\frac{\partial^3\mathbf{f}}{\partial \bar{z}_r\,\partial \bar{z}_s\,\partial z_t}\right|_{\mathbf{0}} \bar{z}_r \bar{z}_s z_t + \left.\frac{\partial^3\mathbf{f}}{\partial \bar{z}_r\,\partial z_s\,\partial \bar{z}_t}\right|_{\mathbf{0}} \bar{z}_r z_s \bar{z}_t + \left.\frac{\partial^3\mathbf{f}}{\partial \bar{z}_r\,\partial \bar{z}_s\,\partial \bar{z}_t}\right|_{\mathbf{0}} \bar{z}_r \bar{z}_s \bar{z}_t\Bigg)\end{aligned} \tag{28}$$

Then, a nonlinear reduced–order model is written in state–space form

$$\dot{\mathbf{z}} = \text{diag}\left(\lambda\right)\mathbf{z} + \bar{\Psi}^T\left(\mathbf{B}^z\left(\mathbf{z},\mathbf{z}\right) + \mathbf{C}^z\left(\mathbf{z},\mathbf{z},\mathbf{z}\right)\right) \tag{29}$$

with initial condition $\mathbf{z}\left(0\right) = \bar{\Psi}^T\mathbf{x}_0$. An advantage of this approach is the reduced dimension of the higher order derivatives of $\mathbf{f}$. For example, $\partial^z_{rs}\mathbf{f}$ has dimension $\mathscr{O}\left(nm^2\right)$ and $\partial^z_{rst}\mathbf{f}$ has dimension $\mathscr{O}\left(nm^3\right)$ which are significantly smaller than $\mathscr{O}\left(n^3\right)$ and $\mathscr{O}\left(n^4\right)$ for the second and third order derivatives of $\mathbf{f}$ in $\mathbf{x}$, respectively, if $m \ll n$ as assumed initially.

3.3.3 Implementation Challenges

Accurate approximations of the higher order derivatives of $\mathbf{f}$ that are included in the terms $\mathbf{B}$ and $\mathbf{C}$ appearing in Eqs. (24) and (25), respectively (and in terms $\mathbf{B}^z$ and $\mathbf{C}^z$, in Eqs. (27) and (28), respectively) are needed.

Derivative approximation techniques fall into three categories: numerical approximation, symbolic differentiation, and automatic differentiation. Numerical approximation techniques include the classical methods of finite–differencing (forward and central differencing) and the complex–step derivative approximation. In the case of finite–differencing, the derivative is replaced by a computation where the function is evaluated at two neighbouring values of the ordinate and the difference between these function values is divided by the difference in the ordinate values. Unfortunately, finite–differencing suffers from round–off and truncation errors, often with catastrophic cancellation of the numerator of the approximation. The use of extended order arithmetic is an alternative that finds use only for small–scale problems.

A second differentiation approach that is available as part of a more general computer algebra system, is symbolic differentiation. Symbolic differentiation is performed using symbolic representations of the actual quantities or variables being manipulated. Symbolic differentiation is most often implemented by accumulating the function into a single expression and differentiating the expression using defined rules of algebra and calculus. While symbolic differentiation is appealing, it can become computationally difficult for very large problems due to the possible explosion in expression size that can arise when computing the derivatives of a complicated function.

The third approach uses automatic differentiation to obtain machine precision accuracy derivatives in a computationally efficient manner. Automatic differentiation exploits the fact that a computer code that implements a general function $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{x})$ can be decomposed into a sequence of elementary function operations. The derivative is then obtained by applying the standard differentiation rules (e.g., product, quotient, and chain rules). Here the `ADiGator MATLAB` toolbox is used [18].

The numerical implementation of the nonlinear reduced–order model, Eq. (29), requires calculating high order derivatives with respect to a complex variable. This is not a trivial task, but fortunately

it may be carried out recursively by application of a fundamental relationship. To start a concise notation is chosen, denoting the complex variable $z \in \mathbb{C}$ as

$$z = \Re(z) + i\Im(z) = z_{Re} + i z_{Im} \tag{30}$$

The complex derivatives of a real function, $f : \mathbb{R} \to \mathbb{R}$, with respect to z may be written in the form:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial z_{Re}}\bigg|_{z_{Im}} - i \frac{\partial f}{\partial z_{Im}}\bigg|_{z_{Re}} \right) \tag{31}$$

and

$$\frac{\partial f}{\partial \bar{z}} = \frac{1}{2}\left(\frac{\partial f}{\partial z_{Re}}\bigg|_{z_{Im}} + i \frac{\partial f}{\partial z_{Im}}\bigg|_{z_{Re}} \right) \tag{32}$$

The calculation of any higher order derivative is facilitated by recursive application of Eqs. (31) and (32). Hereafter, it is implicitly understood that the partial derivative is obtained keeping all other variables constant. For example, second order derivatives with respect to z and $\bar{z}$ are:

$$\frac{\partial^2 f}{\partial z \partial z} = \frac{1}{4}\left(\frac{\partial^2 f}{\partial z_{Re}\partial z_{Re}} - i\frac{\partial^2 f}{\partial z_{Re}\partial z_{Im}} - i\frac{\partial^2 f}{\partial z_{Im}\partial z_{Re}} - \frac{\partial^2 f}{\partial z_{Im}\partial z_{Im}} \right) \tag{33}$$

$$\frac{\partial^2 f}{\partial z \partial \bar{z}} = \frac{1}{4}\left(\frac{\partial^2 f}{\partial z_{Re}\partial z_{Re}} + i\frac{\partial^2 f}{\partial z_{Re}\partial z_{Im}} - i\frac{\partial^2 f}{\partial z_{Im}\partial z_{Re}} + \frac{\partial^2 f}{\partial z_{Im}\partial z_{Im}} \right) \tag{34}$$

$$\frac{\partial^2 f}{\partial \bar{z} \partial z} = \frac{1}{4}\left(\frac{\partial^2 f}{\partial z_{Re}\partial z_{Re}} - i\frac{\partial^2 f}{\partial z_{Re}\partial z_{Im}} + i\frac{\partial^2 f}{\partial z_{Im}\partial z_{Re}} + \frac{\partial^2 f}{\partial z_{Im}\partial z_{Im}} \right) \tag{35}$$

$$\frac{\partial^2 f}{\partial \bar{z} \partial \bar{z}} = \frac{1}{4}\left(\frac{\partial^2 f}{\partial z_{Re}\partial z_{Re}} + i\frac{\partial^2 f}{\partial z_{Re}\partial z_{Im}} + i\frac{\partial^2 f}{\partial z_{Im}\partial z_{Re}} - \frac{\partial^2 f}{\partial z_{Im}\partial z_{Im}} \right) \tag{36}$$

For brevity, the third order terms are left as an exercise for the reader.
There is a final sophistication to mention before deploying any automatic differentiation toolbox. One observes that while the system dynamics f is defined for real variables (recall, for example, Eq. (6)), the above derivatives are taken with respect to a complex variable, z. This situation can be dealt with using the transformation of coordinates, Eq. (14). It is recommended to create a wrapper file for the system dynamics, as illustrated in Appendix C for a simple dynamical system.
The higher order terms appearing in the nonlinear reduced–order model are formed by combinations of simple derivatives. For example, the second order term can be expressed as

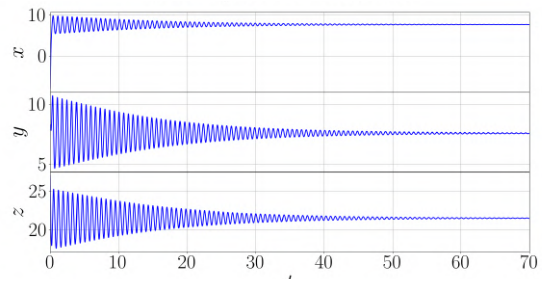
$$B^z = \frac{\partial^2 f}{\partial z^2} z^2 + \frac{\partial^2 f}{\partial z \partial \bar{z}} z\bar{z} + \frac{\partial^2 f}{\partial \bar{z} \partial} \bar{z} z + \frac{\partial^2 f}{\partial \bar{z}^2} \bar{z}^2 \tag{37}$$

An additional advantage of using the ADiGator automatic differentiation package is that ROMs can be generated which are parametric with respect to the coefficients of the governing full order dynamical equations. This can be achieved by passing the function coefficients as an auxiliary variable of differentiation into the derivative generation code. This allows a single ROM to be generated which is valid for any coefficient input. This however requires the eigenvalue problem of the Jacobian matrix to be recomputed as the Jacobian will change based on any coefficients of the full–order equations.

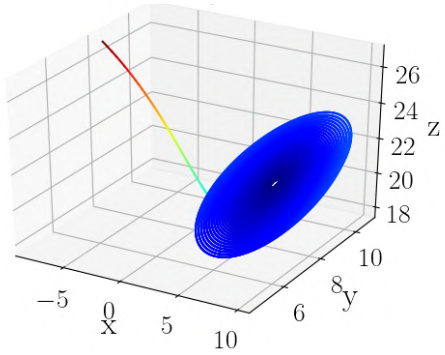
4. Validation Cases

To validate the developed ROMs, complex non-linear dynamical systems have been carefully selected, which emulate flow phenomena present in many fluid problems. Test cases in analytical form were selected to facilitate the validation amongst the different ROMs. The conclusions drawn from this study should be applicable to other less known systems.
This work sees the stated methods applied to three test cases:

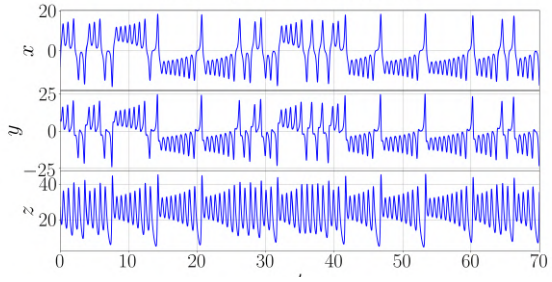
- Lorenz 1963 model [28].
- Moehlis low–dimensional model for turbulent shear flows [29].
- Lorenz 1996 model [30].



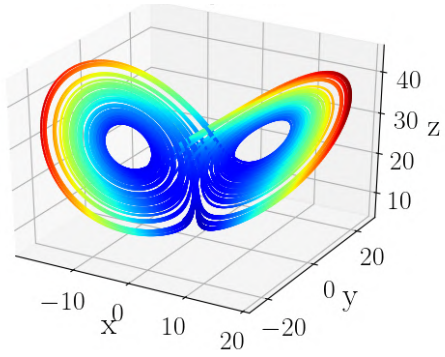
(a) $\rho = 22.5$. Stable case.



(b) The solution gradually converges to a single attractor and eventually stops.



(c) $\rho = 28$. Chaotic case, widely studied in literature.



(d) The solution keeps on switching from one attractor to the other endlessly.

Figure 4 – Lorenz 1963 model: in Panels (a) and (b), deterministic response; in Panels (c) and (d), chaotic response. Right-hand side panels are colored by velocity magnitude.

4.1 Lorenz 1963 System

The Lorenz 1963 system was formulated by Edward Lorenz in 1963 [31]. The system takes the form of three nonlinear ordinary differential equations (ODEs):

$$\begin{cases} \dot{x} &= \sigma(y-x) \\ \dot{y} &= x(\rho-z)-y \\ \dot{z} &= xy-\beta z \end{cases} \tag{38}$$

This system is particularly challenging since the solution changes radically depending on the coefficients chosen [32, 33]. Figure 4 shows the Lorenz solution for two different ρ values in temporal evolution (left column) and spatial coordinates (righ column). Indeed, the most interesting form takes $[\sigma = 10,\ \rho = 28,\ \beta = 8/3]$ and initial conditions $[x_0, y_0, z_0] = [-8, 8, 27]$, shown in the bottom row. The solution emulates the chaotic behavior of turbulent flows. Hence why it is widely studied for ROM investigation [9, 11].

In the present work we extend the investigation to address the stable case $\rho = 22.5$, first row in Fig. 4. The verification on such contrasted behaviors but with similar governing parameters is the ideal scenario to assess the robustness of the developed methods.

4.2 Low–dimensional model for turbulent shear flows

The low–dimensional model for turbulent shear flows formulated by Moehlis [29], henceforth referred to as the Moehlis model, describes a flow with fluid between two free–slip walls experiencing a sinusoidal body force. The velocity field can be expressed as a superposition of temporal amplitude coefficients with nine Fourier modes such that

$$u(x,t) = \sum_{j=1}^{9} a_j(t) u_j(x) \tag{39}$$

where a_j represents the j-th temporal amplitude coefficient and u_j the j-th Fourier mode. The variation of the amplitude coefficients with respect to time can be determined using a series of nine ODEs which are:

$$\dot{a}_1 = \frac{\beta^2}{Re} - \frac{\beta^2}{Re}a_1 - \sqrt{\frac{3}{2}}\frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}}a_6a_8 + \sqrt{\frac{3}{2}}\frac{\beta\gamma}{\kappa_{\beta\gamma}}a_2a_3, \tag{40}$$

$$\begin{aligned}\dot{a}_2 = &-\left(\frac{4\beta^2}{3}+\gamma^2\right)\frac{a_2}{Re} + \frac{5\sqrt{2}}{3\sqrt{3}}\frac{\gamma^2}{\kappa_{\alpha\gamma}}a_4a_6 - \frac{\gamma^2}{\sqrt{6}\kappa_{\alpha\gamma}}a_5a_7 \\ &- \frac{\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}}a_5a_8 - \sqrt{\frac{3}{2}}\frac{\beta\gamma}{\kappa_{\beta\gamma}}a_1a_3 - \sqrt{\frac{3}{2}}\frac{\beta\gamma}{\kappa_{\beta\gamma}}a_3a_9,\end{aligned} \tag{41}$$

$$\begin{aligned}\dot{a}_3 = &-\frac{\beta^2+\gamma^2}{Re}a_3 + \frac{2}{\sqrt{6}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}\left(a_4a_7+a_5a_6\right) \\ &+ \frac{\beta^2\left(3\alpha^2+\gamma^2\right)-3\gamma^2\left(\alpha^2+\gamma^2\right)}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}}a_4a_8,\end{aligned} \tag{42}$$

$$\begin{aligned}\dot{a}_4 = &-\frac{3\alpha^2+4\beta^2}{3Re}a_4 - \frac{\alpha}{\sqrt{6}}a_1a_5 - \frac{10}{3\sqrt{6}}\frac{\alpha^2}{\kappa_{\alpha\gamma}}a_2a_6 \\ &- \sqrt{\frac{3}{2}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}a_3a_7 - \sqrt{\frac{3}{2}}\frac{\alpha^2\beta^2}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}}a_3a_8 - \frac{\alpha}{\sqrt{6}}a_5a_9\end{aligned} \tag{43}$$

$$\begin{aligned}\dot{a}_5 = &-\frac{\alpha^2+\beta^2}{Re}a_5 + \frac{\alpha}{\sqrt{6}}a_1a_4 + \frac{\alpha^2}{\sqrt{6}\kappa_{\alpha\gamma}}a_2a_7 \\ &- \frac{\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}}a_2a_8 + \frac{\alpha}{\sqrt{6}}a_4a_9 + \frac{2}{\sqrt{6}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}a_3a_6\end{aligned} \tag{44}$$

$$\begin{aligned}\dot{a}_6 = &-\frac{3\alpha^2+4\beta^2+3\gamma^2}{3Re}a_6 + \frac{\alpha}{\sqrt{6}}a_1a_7 + \sqrt{\frac{3}{2}}\frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}}a_1a_8 \\ &+ \frac{10}{3\sqrt{6}}\frac{\alpha^2-\gamma^2}{\kappa_{\alpha\gamma}}a_2a_4 - 2\sqrt{\frac{2}{3}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}a_3a_5 + \frac{\alpha}{\sqrt{6}}a_7a_9 \\ &+ \sqrt{\frac{3}{2}}\frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}}a_8a_9\end{aligned} \tag{45}$$

$$\begin{aligned}\dot{a}_7 = &-\frac{\alpha^2+\beta^2+\gamma^2}{Re}a_7 - \frac{\alpha}{\sqrt{6}}\left(a_1a_6+a_6a_9\right) \\ &+ \frac{1}{\sqrt{6}}\frac{\gamma^2-\alpha^2}{\kappa_{\alpha\gamma}}a_2a_5 + \frac{1}{\sqrt{6}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}}a_3a_4\end{aligned} \tag{46}$$

$$\begin{aligned}\dot{a}_8 = &-\frac{\alpha^2+\beta^2+\gamma^2}{Re}a_8 + \frac{2}{\sqrt{6}}\frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}}a_2a_5 \\ &+ \frac{\gamma^2\left(3\alpha^2-\beta^2+3\gamma^2\right)}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}}a_3a_4\end{aligned} \tag{47}$$

$$\dot{a}_9 = -\frac{9\beta^2}{Re}a_9 + \sqrt{\frac{3}{2}}\frac{\beta\gamma}{\kappa_{\beta\gamma}}a_2a_3 - \sqrt{\frac{3}{2}}\frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}}a_6a_8 \tag{48}$$

The constant terms are such that, $\kappa_{\alpha\gamma} = \sqrt{\alpha^2+\gamma^2}$, $\kappa_{\beta\gamma} = \sqrt{\beta^2+\gamma^2}$ and $\kappa_{\alpha\beta\gamma} = \sqrt{\alpha^2+\beta^2+\gamma^2}$, where $\alpha = 2\pi/L_x$, $\beta = \pi/2$ and $\gamma = 2\pi/L_z$. The terms L_x and L_z denote the size of the domain, where $0 \leq x \leq L_x$ and $0 \leq z \leq L_z$. Re is the Reynolds number. The reader is referred to [29] for the formulation of the Fourier modes.

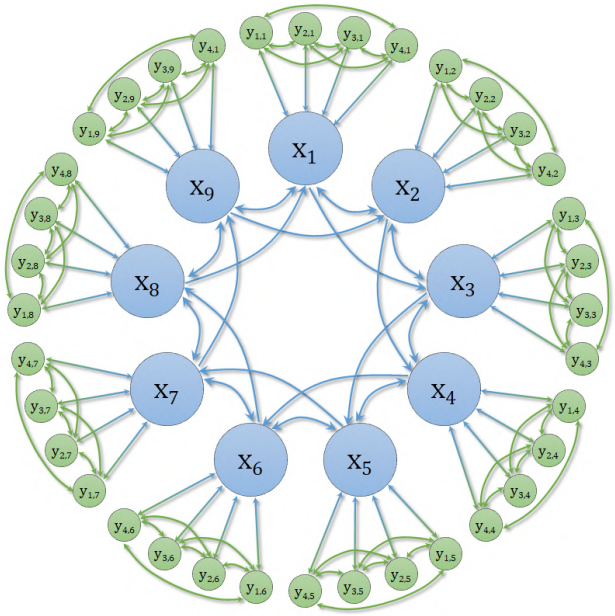


Figure 5 – Representation of the multi-scale Lorenz 1996 system as two one-dimensional rings.

The solutions of the set of equations are dependent on the size of the domain and the Reynolds number. Similarly to the Lorenz 1963 system, the Moehlis model can exhibit both chaotic and deterministic behaviour depending on the aforementioned parameters. Moehlis studies a system with domain size $L_x = 4\pi$ and $L_z = 2\pi$ at $Re = 400$. This results in transient chaotic behaviour, where the system has irregular, non–periodic oscillations that decay to a laminar steady state over time. The parameters mentioned are used for the ROM investigation in this paper.

4.3 Lorenz 1996 system

In 1996, Lorenz developed a multi-scale ODE system to study forecasting [34]. It features a multi-scale coupling of slow and fast variables similar to what is observed in weather and climate phenomena [35]. It is also being used to study chaotic behaviors as it replicates turbulent flow. In its original form, it consists of two-tier equations, comprising a coupled set of K low time-scale variables x_k, for $k = 1, 2, \ldots, K$, each of which is also coupled to J fast variables $y_{j,k}$, for $j = 1, 2, \ldots, J$, with the governing equations [30]:

$$\begin{cases} \dot{x}_k &= -x_{k-1}\,(x_{k-2} - x_{k+1}) - x_k + f - \frac{hc}{b}\sum_{j=1}^{J} y_{j,k} \\ \dot{y}_{j,k} &= -by_{j+1,k}\,(y_{j+2,k} - y_{j-1,k}) - y_{j,k} + \frac{h}{j}x_k \end{cases} \tag{49}$$

where h determines the strength of the coupling between the two systems, b controls the amplitude of the nonlinear interactions, c controls the damping speed of the fast variables relative to the slow and f is a force term that influences the chaoticity of the response. These parameters are fine tuned so that the slow variables oscillate with large amplitude and the fast variables contain a high-frequency range with low amplitude.

From the equations, we observe that the system describes a set of states that evolve in a circle. A visual representation of the multi-scale coupling of the system is provided in Figure 5. Each tier of variables are arranged in separate one-dimensional rings. The arrows indicate the directional influence among the variables. Note how on the outer ring, formed by the small-scale variables, only a small subset influences the corresponding large-scale variable.

From such interaction we realise that each subset of low-scale variables just acts to add high-frequency to the respective primary state. As a result, Lorenz 1996 can be seen as a dynamical system discretised in the one-dimensional space, $x_k(t)$. This motivates the use of the data-driven Conv-LSTM model in a spatio-temporal prediction scheme, described in Section 3.2

To our knowledge, this technique is unique in the prediction of this study case, even though this large-order system is widely used in literature to validate new data-driven approaches. For example,

Chattopadhyay et al. [36] proposed a reservoir-computing echo stat network and compared it to a standard neural network and a standard LSTM model. However, they used a disproportionate dataset size of a hundred million samples to generate their models. Their work expanded from the paper by Dueben et al. [37], which investigated local and global neural-network based models. Last, Brajard et al. [12] and Pawar et al. [38] used this multi-scale system to validate their respective LSTM-based models embedded in a data-assimilation scheme.
In the present work, we opted for $K=20$ slow states and $J=10$ fast states. The coupling parameters were set to $h=1$, the force term $F=10$ and the remaining coefficients were chosen as $b=c=10$. This parameter choice was arbitrary but it follows general practice in literature [39, 35, 36].
At this point, we reflect on the number of parameters and model size involved with the data-driven identification method proposed in Section 3.1for each validation case. In Table 1, we report the number of ODE coefficients for the first two test cases and the total number of parameters involved in their identification using a second-order polynomial basis function. The original Lorenz 1963 system comprises just 3 equations with 3 coefficients each, except the last one. A 2^{nd}-order identification comprises 10 parameters per equation. The 9-equation system proposed by Moehlis takes an average of 5 coefficients and 55 candidate parameters to include in the identification per equation. Last, the chosen Lorenz 1996 system of 20 fast and 10 slow modes, results in 11 equations per large-scale state, with 54 ODE coefficients and 1,044 model parameters per state. Furthermore, the sparsity ratio (ratio between the two values) becomes worse with the system size, which adds further difficulty for the identification approach.

Table 1 – Number of coefficients involved in each validation case for the identification method to find, considering just a second-order polynomial from.

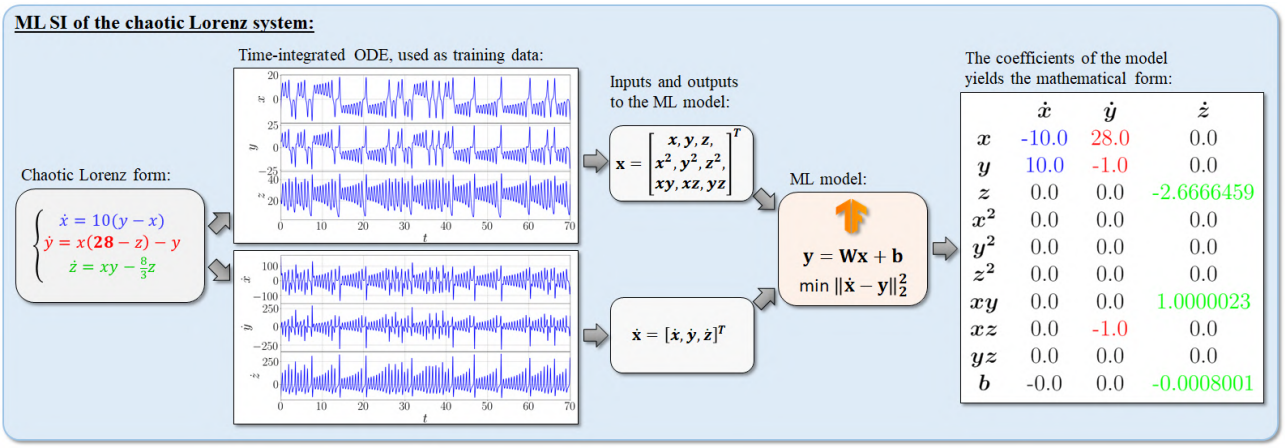
Case	States	ODE coefficients	SI Parameters	Sparsity Ratio
Lorenz	3	7	30	0.233
Moehlis	9	44	495	0.089
Lorenz'96	220	1080	20880	0.045

We conclude that for a single-scale system of n states, the number of parameters involved in a second-order identification grows to the order of $\mathcal{O}(\sim n\frac{(n+2)(n+1)}{2})$. The number of parameters to consider on the Lorenz 1996 system is disproportioned, even if constrained to a quadratic basis function. Consequently, a complete identification method, such as the first of the techniques adopted here (Section 3.1), becomes computationally impractical. This justifies the adoption of a data-driven regression approach instead, i.e. the convolutional-based LSTM method in Section 3.2
In addition, for the data-driven prediction, the order (number of states) of the system is reduced by excluding the high-frequency variables $y_{j,k}$ and only considering the slow x_k. Such strategy is also motivated by the fact that fast variables are typically not observable and only the low-frequency information is available from the gathered data. As such, this strategy represents a crucial dimensionality reduction of the system. More details are provided in Section 5.3.1

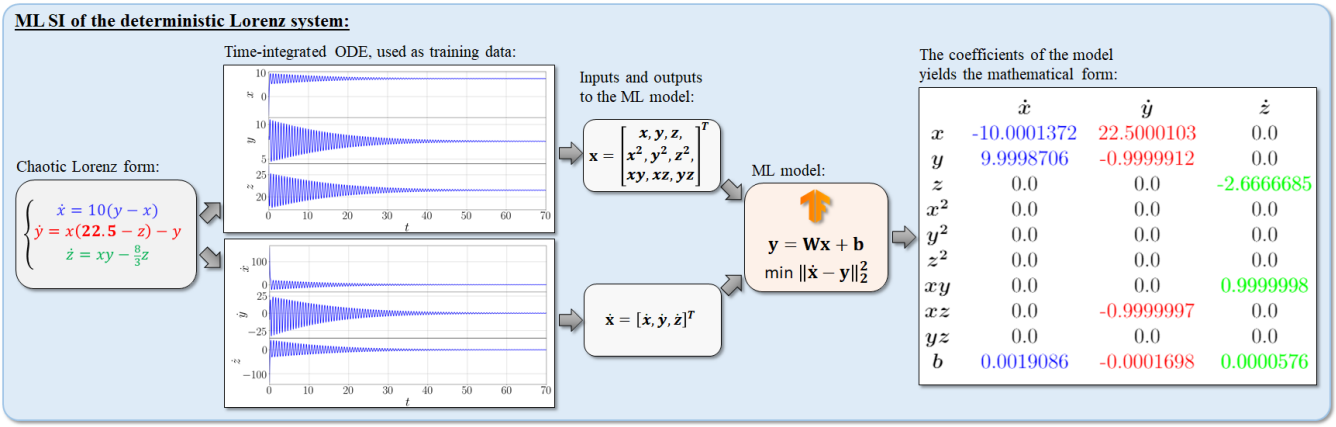
5. Results

This section presents the results obtained from the set of ROMs built for this work, as discussed in Sections 2and 3.
The underlying idea is to cross-check the ability of each framework to recover the basic physics of complex nonlinear systems. We will address the following questions: a) when will each ROM depart from the analytical solution? b) what algorithm provides better computational advantage? c) is the model easy to implement and/or generate? For an exhaustive comparison amongst the different frameworks, we propose the following performance indicators:

- Computational cost to generate the model.
- Amount of high-fidelity data required.
- Ability to reconstruct the real physics.



(a) Discovered coefficients for the chaotic Lorenz case. The actual equations are on the left block.



(b) Discovered coefficients for the deterministic Lorenz case. The actual equations are on the left block.

Figure 6 – System identification results for the two Lorenz study cases.

- Implementation complexity.
- Computing time.
- ROM ability to predict for cases other than those it was generated from.

The results for each test-case and corresponding discussion are now presented.

5.1 Lorenz 1963 Model

Results for the Lorenz 1963 system are based on two forms of the governing equations, these being the chaotic and deterministic solutions. As seen in Section 4.1, this change in behaviour comes from the variation of the parameter ρ.

5.1.1 Machine-Learning System Identification

First, we perform the system identification using our MLSI method, which outputs the coefficients of the mathematical expression for each state. The benchmark data was obtained via time-integration of the Lorenz equations (38) using a 4^{th}-order Runge-Kutta scheme [40] with a time step of $\mathrm{d}t = 0.001$ s and a total simulation time of 40 s. Therefore, the dataset consists of $40,000$ samples.

Figure 6 shows how the equation terms were predicted for each Lorenz form, just using the data from the real response, without any knowledge of the underlying equations. As can be seen, the MLSI model correctly found the coefficients of the equations in both cases, albeit with a rounding error. As a result, our proposed MLSI method proved successful for the discovery of this nonlinear dynamical system.

It was found that that the deterministic case proved more complicated to predict, evidenced by the slightly worse decimal precision on the identified coefficients. The reason for the case with $\rho = 28.0$

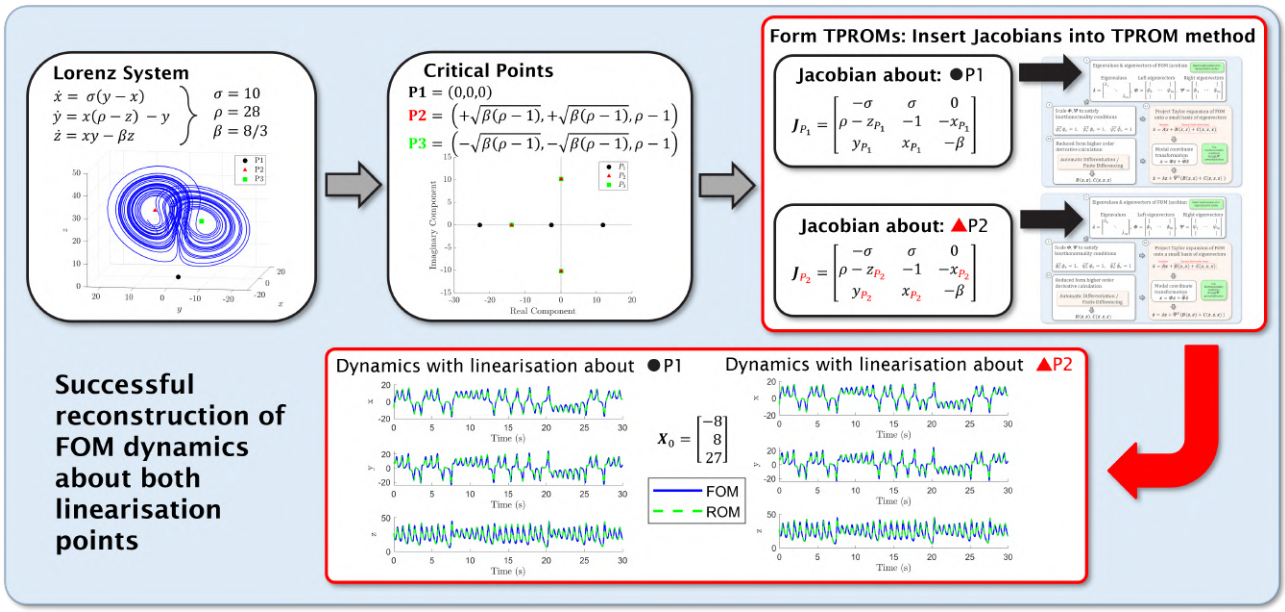


Figure 7 – High–level overview of the TPROM approach applied to the Lorenz 1963 system to generate ROMs about different equilibrium points.

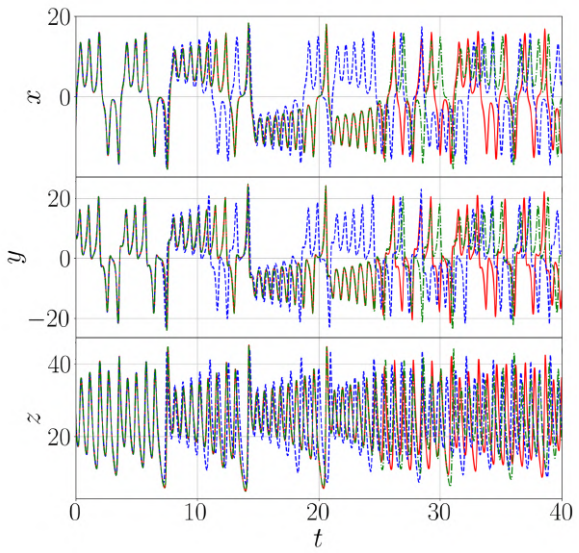
being easier to model is likely due to the large variability of the data, a key aspect to enhanced machine-learning performance. By contrast, the case with $\rho = 22.5$ resembles a damped harmonic oscillator. This type of signal can be well represented with a simpler mathematical formulation, albeit it derives from the Lorenz 1963 model. Despite this, the proposed method successfully found the correct forms.

5.1.2 Taylor-Series Projected ROM

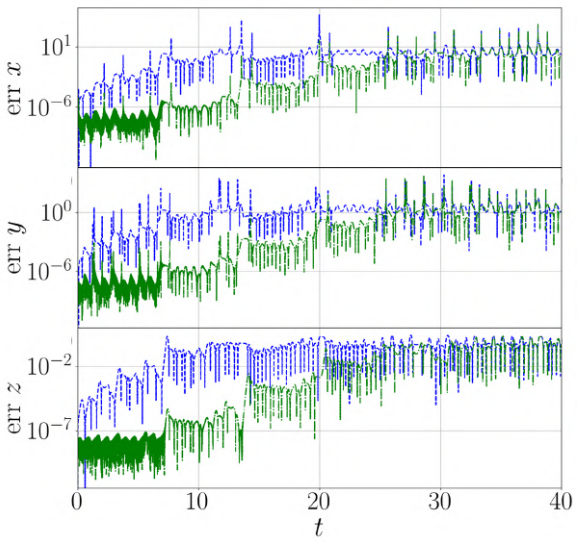
Since the Lorenz 1963 system is already low-dimensional, the application of the TPROM is to demonstrate the full order dynamics can be recovered providing that all of the system's modes are retained. A high–level overview showing the application of the TPROM to the Lorenz 1963 system is given in Figure 7. The reader is reminded that the TPROM approach is based on the full–order system being projected onto a subspace of eigenvectors of the Jacobian matrix evaluated at an equilibrium point. The Lorenz 1963 system has three equilibrium points, as noted in Figure 7. These exist at $O = (0,0,0)$ (black circle) and at $C^{\pm} = (\pm\sqrt{\beta(\rho-1)}, \pm\sqrt{\beta(\rho-1)}, \rho-1)$ (red triangle and green square). To demonstrate the TPROM method recovers the correct dynamical behaviour irrespective of the chosen equilibrium point, two TPROMs are generated, one about O and the other about C^+. It is worth noting the difference in the eigenvalue composition of the equilibrium points. At O the eigenvalues are purely real while at C^+ there is a single purely real eigenvalue and a complex conjugate pair. The result of this is shown in Figure 7 with both TPROMs recovering the same dynamical properties.

5.1.3 Comparison

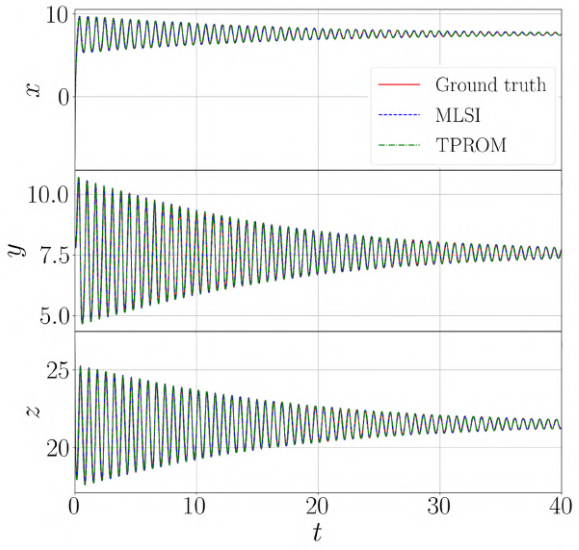
A comparison of the time-integrated responses from both the MLSI and TPROM methods against the full-order solution can be seen in Figure 8, for the two Lorenz cases respectively. As can be seen in Panel (a), the solutions bifurcate after a certain time step, with the MLSI response diverging earlier than the TPROM solution. This is caused by numerical error propagation in a system chaotically evolving around two attractors. A comparison of the propagated error is shown in Panel (b), which is significantly lower on the TPROM up to where both methods bifurcate. The TPROM method results in a more accurate model thanks to addressing the reconstruction from the full-order equations. A similar result was obtained for the deterministic case shown in Panel (c). In this case, however, both methods matched the full-order response remarkably well. MLSI consistently showed larger error as



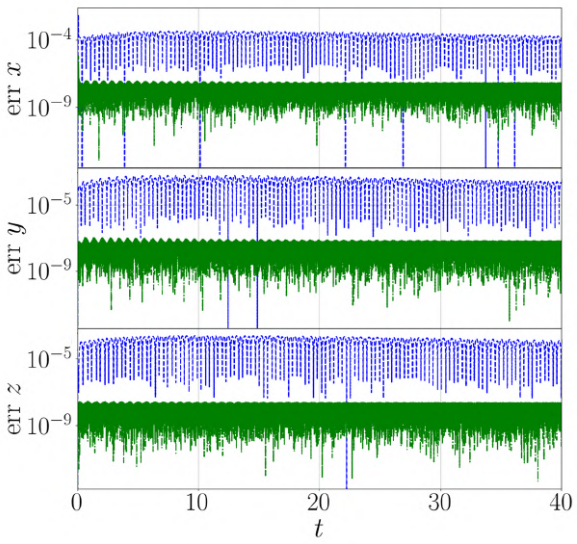
(a) System state response for the chaotic case.



(b) Normalised error from each ROM result against the ground truth, chaotic case.



(c) System state response for the deterministic case.



(d) Normalised error from each ROM result against the ground truth, deterministic case.

Figure 8 – Comparison of MLSI (blue) and TPROM (green) solutions against the ground truth (red) for the two Lorenz 1963 forms.

seen in Panel (d). This method is inevitably more liable to suffer from numerical error as it constructs the model from data, without any knowledge of the full-order representation.
As noted in Section 3.3the TPROM approach can be used to generate parametric ROMs. Here the model can be parameterised with respect to the Lorenz coefficients, σ, ρ and β. This results in a single TPROM generation that can be used to capture both the chaotic and deterministic regimes of the Lorenz 1963 system. The TPROM results in Figure 8 representing the chaotic and deterministic dynamics were formed using a single TPROM generation.

5.2 Low–dimensional turbulent shear flow results

The methods are here applied to the time evolution of the Moehlis model. The velocity field is recovered by multiplication of the coefficient with the corresponding mode. For each ROM, the process of generation and application follows what was previously presented for the Lorenz 1963 system.
To obtain the ground-truth data and generate the system identification, the Moehlis equations (40)

Table 2 – Number of coefficients involved in each validation case for the identification method to find, considering just a second-order polynomial from.

Case	Model Parameters	Training [CPU-h]
Lorenz $\rho = 28.0$	30	0.35
Lorenz $\rho = 22.5$	30	0.45
Moehlis	495	3.20

to (48) were integrated with a 4^{th}-order Runge-Kutta scheme with a time step $\mathrm{d}t = 0.1$ s for a total simulation of 1000 s. The dataset contains $10,000$ samples.

5.2.1 Machine-Learning System Identification

The number of parameters involved in the 2^{nd}-order MLSI of this system is 495, as we reported in Table 1. We now report in Table 2 the computational cost to identify the model, compared to the cost for the Lorenz 1963 model. This estimation is not straightforward as it comprises combined CPU and GPU usage. Regarding the Lorenz 1963 system, we found that the chaotic case could be correctly identified with a shorter dataset. The reason is because the data variability was already contained from the initial time steps. By contrast, a longer dataset was required for the deterministic case to include the convergence towards the equilibrium in the training data. This explains why the CPU hours required to generate the chaotic Lorenz is lower. In addition, the cost to identify the Moehlis system is ~ 8 times larger. It appears that with increasing dimension of the model to identify, the MLSI loses attractiveness due to the higher identification costs. This justifies the switch to the second data-driven method (from Section 3.2) for the last validation case.

5.2.2 Taylor-Series Projected ROM

With respect to the TPROM method, an equilibrium point is required about which the system Jacobian is formed. This equilibrium point is taken as $a_1 = 1$ and $a_2 = a_3 = ...a_9 = 0$ which corresponds to the laminar state of the system and is linearly stable for all values of Re. The generation of the TPROM nonlinear terms are also parameterised for the domain of the system, L_x and L_z, the system Reynolds number, Re, and the reduced basis. The strength of this parameterisation is that with the single generation, the TPROM nonlinear terms can be evaluated for varying domain sizes and varying values of Re at a significantly smaller cost than would otherwise be needed to regenerate the term. Likewise this parameterisation captures the correct dynamics of the turbulent and laminar states of the system. This has value for implementation with significantly larger systems, in CFD for example, where it may otherwise be necessary to generate many ROMs and interpolate between them.

5.2.3 Comparison

The results in Figures 9 and 10 have been generated using domain size $L_x = 4\pi$, $L_z = 2\pi$ and $Re = 400$. An initial condition $\mathbf{a}_0 = [1.0, 0.07066, -0.07076, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$, with a perturbation of magnitude $1.e-6$ to a_4 is used, which is a similar initial condition to that used in [13]. From these results it is clear that both the MLSI and TPROM approaches are equally successful in reconstructing the dynamics of the system. Both methods correctly capture the transient chaotic behaviour of the full order model, where the system exhibits non-periodic oscillations which then appear to decay into a laminar state. According to 10, both methods have low errors throughout the period of integration, though the TPROM maintains a consistently lower error. It is worth noting that the error is bounded for the entire duration of the analysis, indicating the good ability of the ROMs over long periods of time. As the system decays into the laminar state it is expected that these errors will settle.
Figure 11 demonstrates the parametric capability of the TPROM applied to the Moehlis model. Using a single TPROM generation, Panel (a) shows the flow field of the FOM and TPROM at 400 seconds for $L_x = 1.7\pi$, $L_z = 1.2\pi$ and $Re = 400$, and Panel (b) the flow fields at 400 seconds for $L_x = 6\pi$, $L_z = 4\pi$ and $Re = 300$. In both cases it can be seen that the mean velocity profiles (upper panel) of the FOM and TPROM are identical and that the flow–fields, downstream vortices (central panel) and midplane flow (lower panel), are likewise identical.

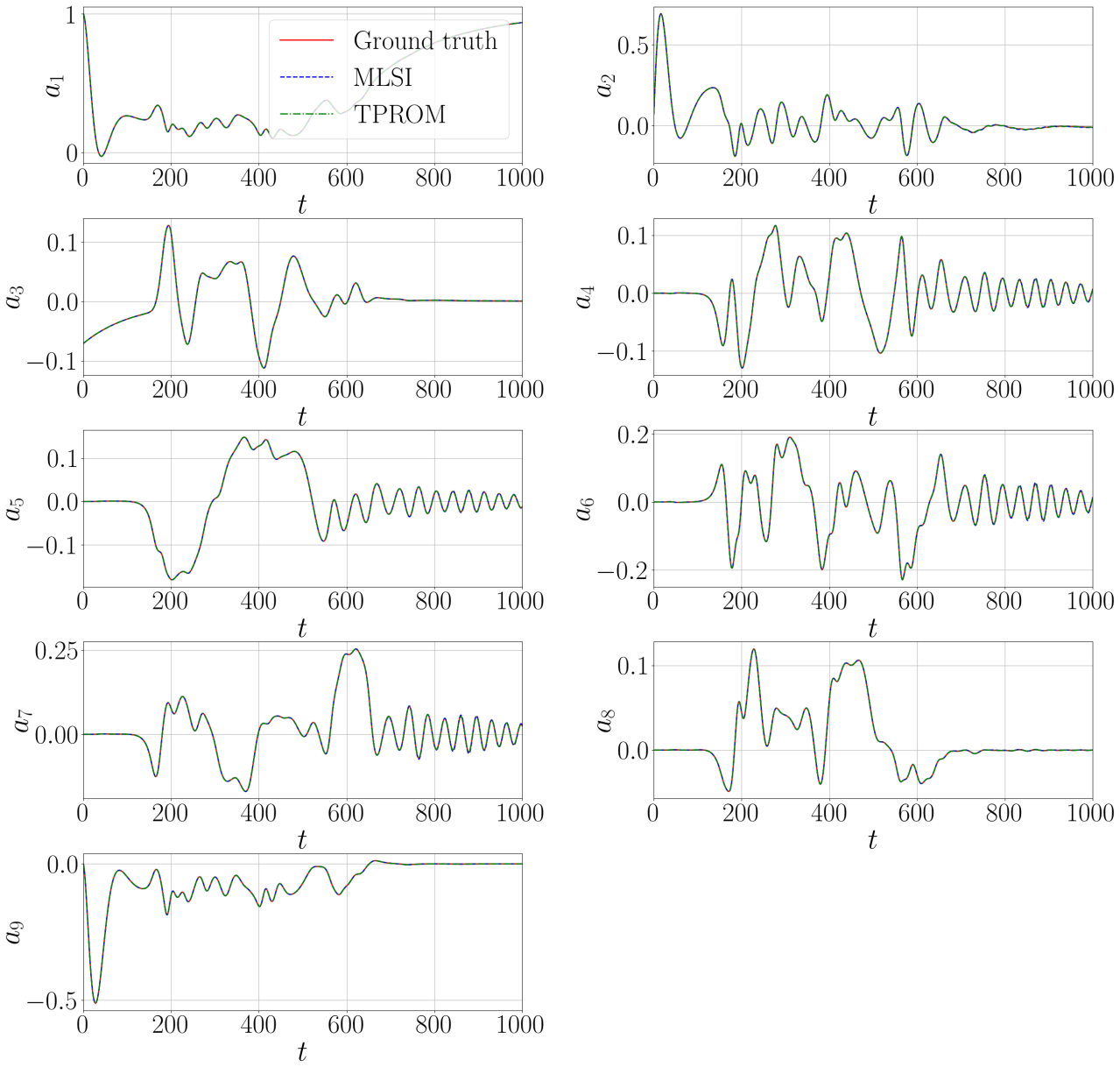


Figure 9 – Comparison of the prediction from the system identification (dashed blue) and the Taylor-projection ROM (dot-dashed green) against the true solution (solid red) of the Moehlis system.

5.3 Lorenz 1996 Model

The Lorenz 1996 test case is a system of larger dimension than those previously discussed. The test case consists of 20 large–scale states, with 10 fast states for each large–scale state. This results in a system of dimension 220.

As in the previous cases, the true data is generated via time-integration with a 4^{th}-order Runge-Kutta solver, using a time step of $\mathrm{d}t = 0.001$ time units for a total simulation time of 100 units. As a result, the total dataset size is of $100,000$ samples. This is three orders of magnitude smaller than the dataset used in [36] for their data-driven study of the same system.

5.3.1 Convolutional Long Short-Term Memory Model

In this section we provide the challenges addressed with the data-driven prediction of the Lorenz 1996 system as well as details on the final architecture and hyperparameters adopted for the Conv-LSTM neural network.

As part of the challenges, we reflect on the strategy to consider only the large-scale states x_k on the prediction. Despite each state being influenced by their respective subset of fast states $y_{j,k}$, Equa-

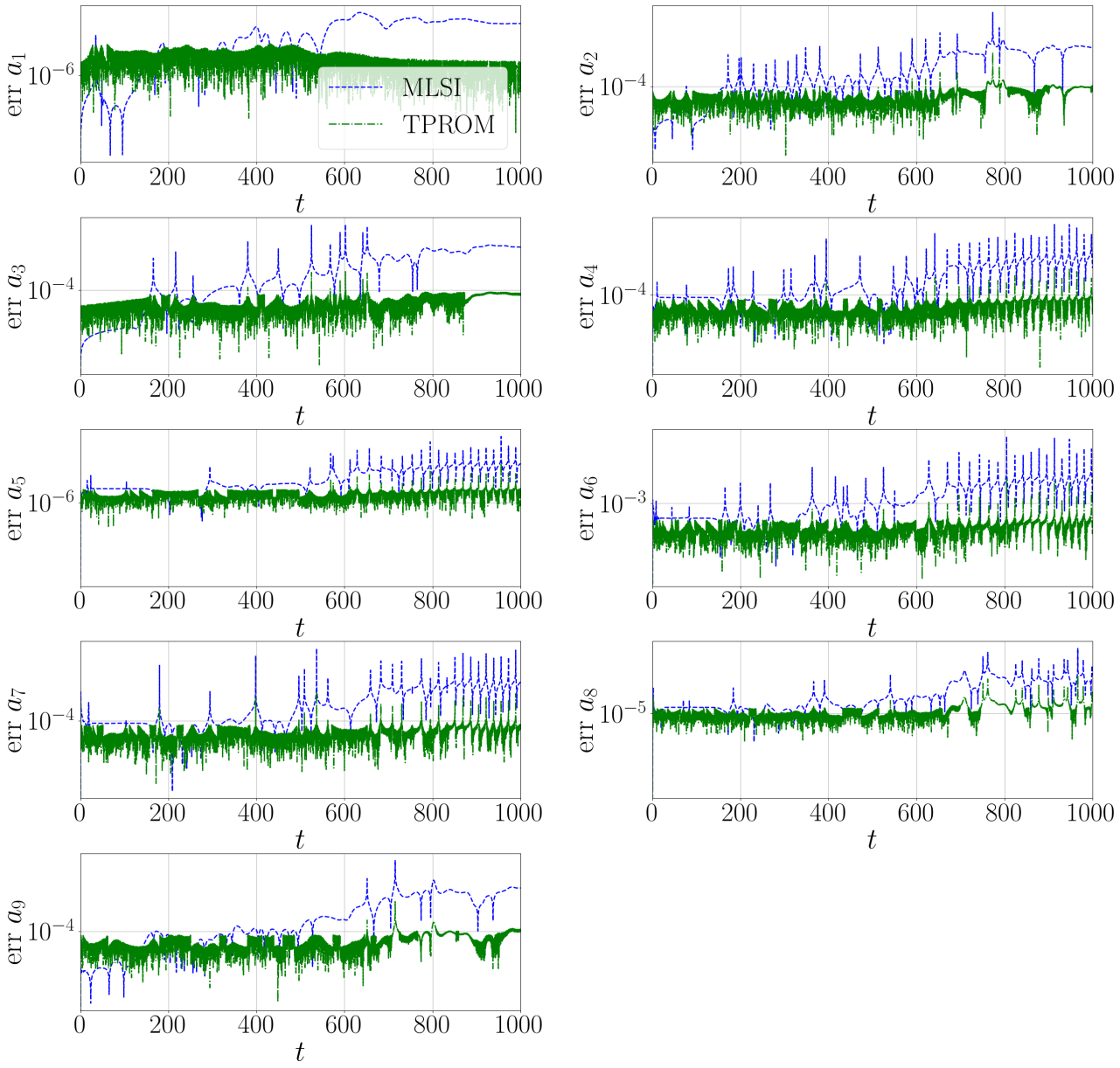
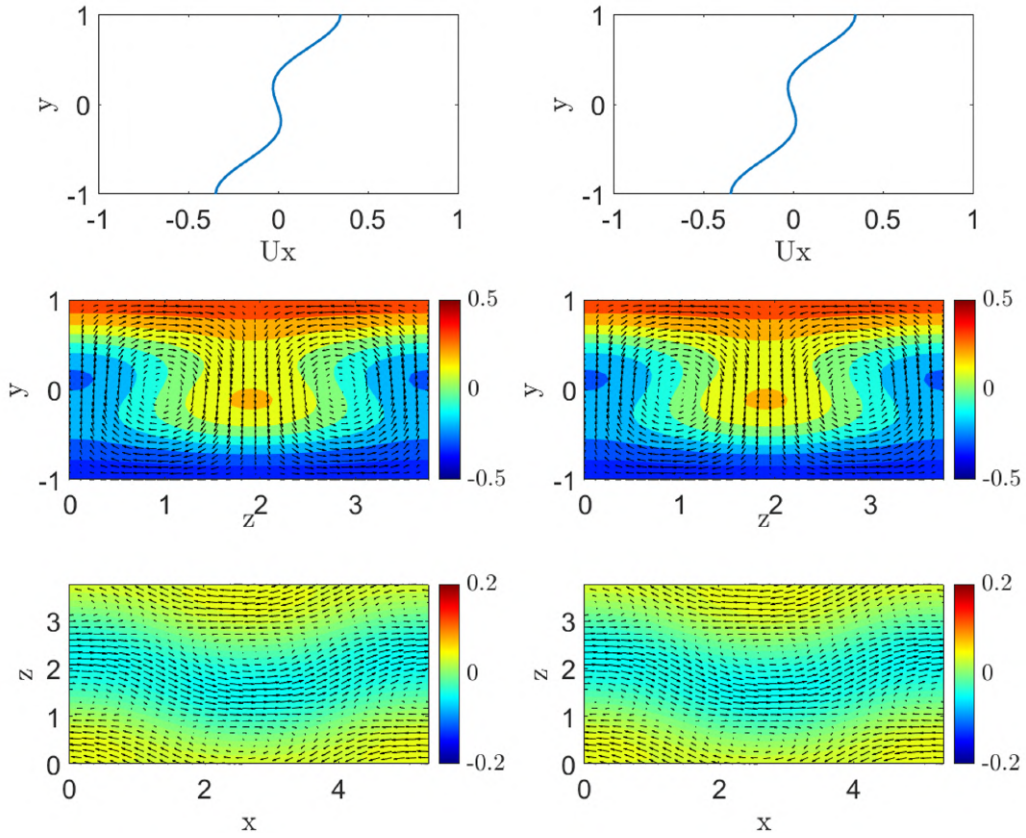


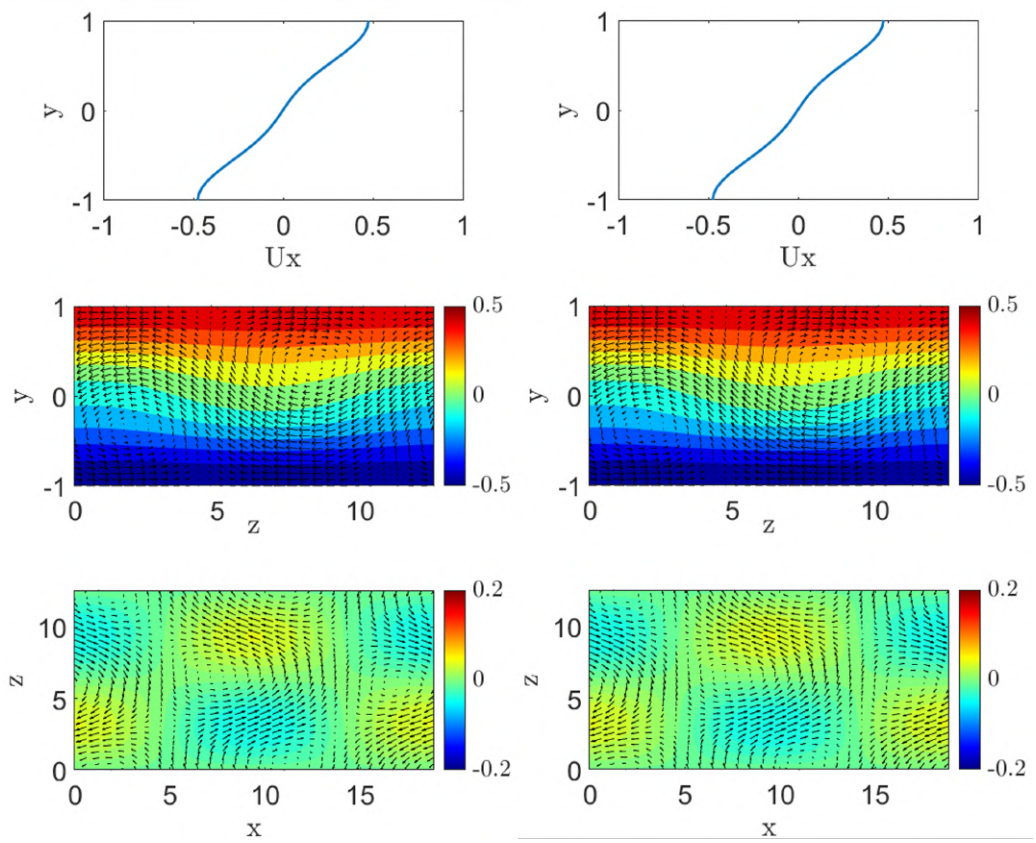
Figure 10 – Normalised error from the two models on the Moehlis system. Errors taken from the results illustrated in Fig. 9.

tion (49), the latter are typically difficult to measure because of their reduced scale and higher frequency content. By excluding the fast variables from the prediction, the dynamical system effectively undergoes a modal truncation. Indeed, the Lorenz formulation in two tiers allows for a dimensionality reduction of the system without having to resort to classical compression techniques, such as the proper orthogonal decomposition [41] or the autoencoder [42].

In Appendix D we provide a summary of the sensitivity study to fine tune the neural network. Following this study, the final model configuration chosen is reported in Table 3, based on the architecture from Figure 3. The dimension column refers to the tensor size at the output of the corresponding layer. The tensor dimension is arranged in (m samples $\times$ n time delays $\times$ k spatial dimension $\times$ c channels). The periodic padding is called at the beginning of the network, with the adeuate size so no additional padding is necessary through the remaining of the network. Three Conv1D-LSTM layers were chosen as they were found sufficient to correctly capture the dynamics of the system. A convolutional kernel (or filter) of size 5 was chosen for all three layers. The first two layers were assigned to 35 filters (channels). In the last layer, the filters and the time sequence are reduced to a single output of the same spatial dimension as the input. This amounts to a model size of $75,204$ trainable weights, as



(a) $L_x = 1.7\pi$, $L_z = 1.2\pi$ and $Re = 400$ at 400 seconds.



(b) $L_x = 6\pi$, $L_z = 4\pi$ and $Re = 300$ at 400 seconds.

Figure 11 – FOM (left) and TPROM (right) flow field comparison of the Moehlis low-order turbulent shear flow system. Upper, central and lower plots in each panel represent mean velocity profile, downstream vortices and midplane flow, respectively.

reported in Table 4. In this table we also report crucial hyperparameters that complete the whole machine-learning framework.

Layer	Dimension	Kernel	Channels	Padding
Input	$m \times 3 \times 20 \times 1$			
Padding	$m \times 3 \times 32 \times 1$			Periodic
Conv1D-LSTM 1	$m \times 3 \times 28 \times 35$	5	35	–
Tanh				
Conv1D-LSTM 2	$m \times 3 \times 24 \times 35$	5	35	–
Tanh				
Conv1D-LSTM 3	$m \times 1 \times 20 \times 1$	5	1	–
Output	$m \times 20$			

Table 3 – Layer information of the final Conv-LSTM model. Refer to Fig. 3 for the diagram of this NN architecture.

Parameter	Value
Trainable parameters	$75,204$
Sequence time steps n	3
Simulation $\mathrm{d}t$	0.005
Dataset samples	$20,000$
Training set (70%)	$14,000$
Batch size	$1,000$
Training epochs	250
Loss function	Huber
Optimiser	Adam
Learning rate	0.0025

Table 4 – Hyperparameter choice of the definitive Conv-LSTM model.

In this table we also report crucial hyperparameters that complete the whole machine-learning framework. The time sequence for the LSTM operations was chosen as $n=3$, which means that the network predicts the next state $x_k(t+\mathrm{d}t)$ from the previous three steps. In some tasks, this could be inconvenient because in a time-marching simulation, n time steps are required as initial conditions, rather than the usual single condition.
The simulation step was coarsened to $\mathrm{d}t = 0.005$ time units as it was found sufficient to capture the dynamics and boosted both training (thanks to a smaller dataset) and simulation speeds. Morever, from the resulting dataset size of $20,000$ time-steps, 70% were used for the training. Last, we chose the Huber loss function [20] for the minimisation problem, although no significant difference was observed when comparing to the mean-squared error function. See Appendix D.

5.3.2 Taylor-Series Projected ROM

Regarding the TPROM method, the exact same process can be applied as that used for the smaller dimension systems. More specifically, the definition of the system equilibrium point and then, the generation of the TPROM about this point. A further intricacy is the introduction of the constant forcing term, F, as seen in Eq. 49. This acts as a control parameter for the system. For the Lorenz 1996 parameters given, the system can be defined about the equilibrium where $F = 10$. To demonstrate the TPROMs ability to function with control parameters, the equilibrium point has been instead defined about $F = -1$, and then, a control perturbation $dF = 11$ is added and maintained for the time integration. This results in the expected dynamical behaviour.
The TPROM now presented is generated without modal truncation. This has resulted in an exact reconstruction of the system and its dynamics. Differences in the integrated solution over time, as seen in Figure 12c, are due to differences in the way the TPROM code is assembled compared to the

original FOM code. For highly coupled and chaotic systems, such as the Lorenz 1996 system, these differences are enough to cause eventual divergence from the FOM in the time integrated solution, though it is stressed that the dynamical behaviour remains accurate. Not performing modal truncation has however created a system of impractical size and dimension. The non–truncated TPROM matrices are dense and contain a total of nearly 13,000,000 elements. This results in substantial computational cost when performing time integration. For demonstrative purposes a truncated TPROM is generated. In this case, the truncation is performed by removing the 70 fastest modes of the system, thus using only 51 of the original system's 121 unique modes. This truncated TPROM is 18% of the size of the non–truncated TPROM and contains only 2,300,000 elements. The size and performance metrics of the TPROM are reported in more detail in Table 6.

5.3.3 Comparison

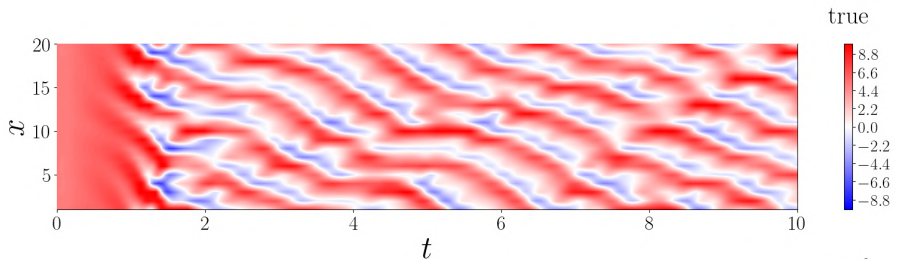
The results are presented for the Conv–LSTM (Section 3.2) and TPROM (Section 3.3) approaches. Figure 12 illustrates the results of the Lorenz 1996 in spatio-temporal evolution. Panel (a) shows the true solution; Panel (b) and (c) the results from the Conv-LSTM model and the non–truncated TPROM, respectively. With the data-driven model, Panel (b), the solution remains very accurate up to $t = 1$ s. After that, the solutions differ but the dynamics are well captured nonetheless. It is important to remember that the data-driven model was generated without using the the small-scale variables $y_{j,k}$ and only considering the large scale states x_k, effectively undergoing a modal reduction. This is an exemplary situation because the variables $y_{j,k}$ influence the large-scale variables but are unmeasurable from the data. On the other hand, it is observed that the TPROM, Panel (c), produces a result accurate to the true solution for a longer period of time, up to around $t = 2$ s. After that, differences appear due to the chaotic nature of the system. A further observation is that for the TPROM, error appears to propagate from the lower system states to the higher system states over time, with visible errors appearing at $t = 2$ s for the lowest index states and $t = 4$ s for the highest index states.

5.3.4 Statistical and Performance Analysis

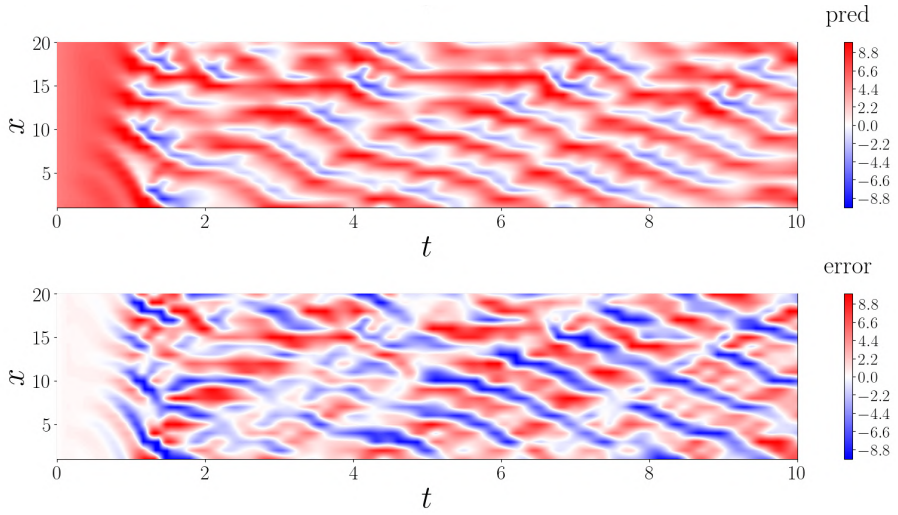
This section presents a comparison of the statistical and performance metrics of the TPROM and Conv–LSTM methods applied to the Lorenz 1996 system. Figure 13 shows the probability density functions (PDFs) of the time integrated solutions of the various model types. It can be seen that the PDFs of the Conv–LSTM and non–truncated TPROM closely match the PDF of the true solution. The truncated TPROM however has a narrower PDF with two distinct peaks forming. These peaks occur since the truncated TPROM has resulted in periodic rather than chaotic dynamics. The peaks indicate the maxima and minima of the periodic oscillations.

Table 5 provides further statistical measures of the system. Here it can be observed that the statistical metrics of the Conv–LSTM and non–truncated TPROM are close to those of the true solution, which indicates their dynamical properties are correct. The variance of the truncated TPROM is however much smaller than the variances of the true solution and other ROM methods. As identified from the PDF, this is due to the truncated TPROM exhibiting different dynamical behaviour and having a tighter grouping of data points.

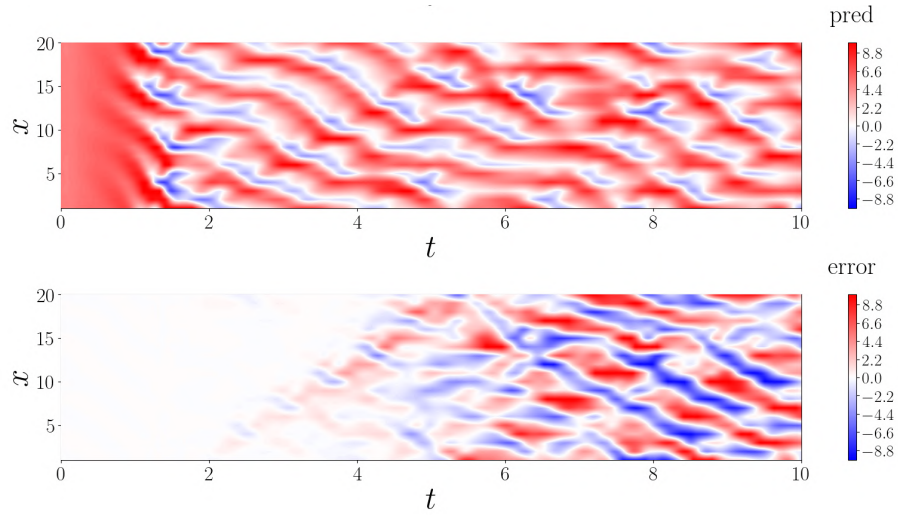
To complete the analysis of the different modelling approaches, we provide a summary of computing costs involved. The essential costs considered here are the computing burden to generate (or train) the models, the cost to complete the time-marching simulation of $t = 20$ time units and the storage memory requirements. Reporting CPU-hour requirements to train the neural network is non trivial because `TensorFlow` performs optimised management of simultaneous CPU and GPU usage. Nevertheless, we found that the simulation cost is significantly lower with the data-driven than with the intrusive approaches. This advantage is at the expense of loss of instantaneous accuracy, despite the statistics being well captured. A noteworthy point is the generation-cost improvement against the identification approach used in the previous cases, which was reported in Table 2, especially as we are now dealing with a system an order of magnitude larger.



(a) True solution of the Lorenz 1996.



(b) Prediction and error from the data-driven Conv-LSTM model. As a reminder, only the large-scale variables x_k are involved. The fast variables $y_{j,k}$ are excluded as part of the model reduction.



(c) Prediction and error from the TPROM.

Figure 12 – Spatio-temporal comparison of the two ROMs against the true Lorenz 1996 solution. Error denotes the delta between prediction and the ground truth.

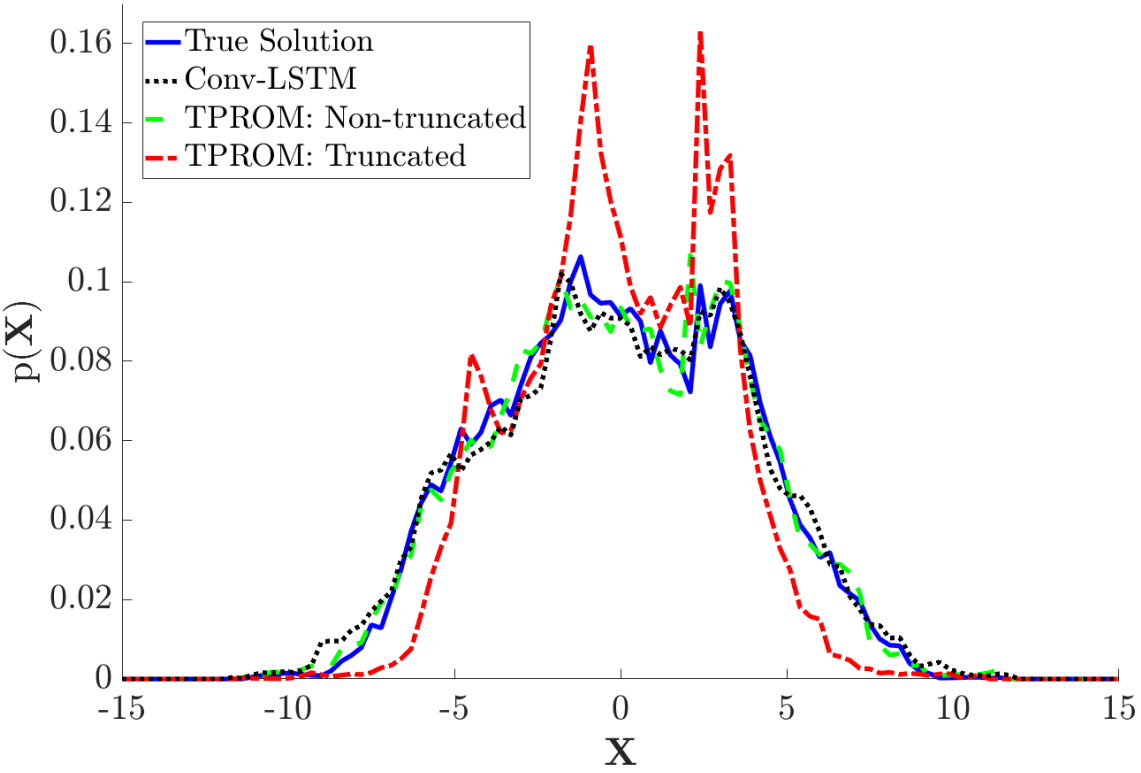


Figure 13 – Probability density functions of the integrated solutions of the various ROM types of true solution.

Table 5 – Performance analysis against the benchmark solution for the various reduced order models.

Model	Mean	Variance	Skewness	Kurtosis
True Solution	2.66	13.48	-0.01	2.35
Conv-LSTM	2.79	15.13	-0.05	2.49
TPROM: Non–truncated	2.81	14.03	-0.03	2.49
TPROM: Truncated	2.54	8.79	-0.05	2.58

6. Conclusions

This paper presents three novel reduced-order model techniques based on distinct methods. One method is a data-driven identification which reduces the model order with a fully-integrated sparsity promotion technique. A second method is a data-driven regression especially suited for spatio-temporal prediction of large systems. Further model reduction is achieved by involving only the large-scale states in the prediction. The third technique is a model-derived method adopting a reduced-basis Taylor expansion of the full-order equations leveraging on auto-differentiation. This model was also created with nonlinear terms that are parametric with respect to the system parameters and basis. The extent to which the technique is fully parametric however is limited by the speed that the eigenvalue problem of the system Jacobian can be solved. This is as the Jacobian changes with varying system parameters. For the system sizes considered here, the speed at which the eigenvalue problem can be solved is in the order of milliseconds for a standard desktop computer. For much larger systems the solution time may quickly become prohibitive for example where real–time parameter variation is necessary.

The reduced order models were tested on three systems of increasing complexity and presenting a chaotic nature. The test cases were the Lorenz 1963 model, a low–dimensional model for turbulent shear flows, and the Lorenz 1996 model. A set of key performance indices were used to quantify the accuracy and robustness of the reduced order models. In this work, the focus was on distilling the

Table 6 – Computing cost analysis for the various reduced order models.

Model	Generation Cost [CPU-h]	Simulation Cost [CPU-h]	Model Size [MB]
Conv-LSTM (GPU)	1.00	0.81	1.2
TPROM: Non–truncated	0.89	9.89	200.0
TPROM: Truncated	0.21	2.00	89.0

equations governing the test cases, which are of relatively small size.
We accomplished the task related to assessing the predictive capability of data-driven and physics-derived models. It was found that both approaches performed well in the short and long time scales compared to the ground truth. It is worth noting that each approach has disadvantages, notably: for the data-driven methods, the cost to generate a sufficient and dynamically–rich training dataset; and for the physics-derived model, the loss of sparsity of the operators in the projected space. We will consider next test cases of larger size that require reduction onto a smaller basis to gain a deeper understanding of the advantages and disadvantages of the two approaches.

7. Contact Author

This work is submitted as a student paper. For queries, please contact Declan Clifford (dsc1g17@soton.ac.uk) and the project supervisor, Dr. Andrea Da Ronch (A.Da-Ronch@soton.ac.uk).

8. Copyright Statement

References

[1] A. Da Ronch, K. J. Badcock, Y. Wang, A. Wynn, and R. Palacios, “Nonlinear Model Reduction for Flexible Aircraft Control Design,” tech. rep.

[2] R. Kurniawan, “Numerical study of flutter of a two-dimensional aeroelastic system,” *ISRN Mechanical Engineering*, vol. 2013, 2013.

[3] S. L. Brunton and C. W. Rowley, “Modeling the unsteady aerodynamic forces on small-scale wings,” tech. rep.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, “Machine Learning for Fluid Mechanics,” *Annual Review of Fluid Mechanics*, vol. 52, pp. 477–508, 2020.

[6] J. R. Koza and R. Poli, “Genetic programming,” in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pp. 127–164, Springer US, 2005.

[7] W. C. Krolick, Y. Wang, and K. Pant, “Nonlinear aeroelastic reduced order modeling based on nested optimization and regularization,” *AIAA Scitech 2021 Forum*, no. January, pp. 1–17, 2021.

[8] R. Rubini, D. Lasagna, and A. Da Ronch, “The l 1 -based sparsification of energy interactions in unsteady lid-driven cavity flow,” *Journal of Fluid Mechanics*, vol. 905, 2020.

[9] S. L. Brunton, J. L. Proctor, J. N. Kutz, and W. Bialek, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 113, no. 15, pp. 3932–3937, 2016.

[10] S. L. Brunton1, J. L. Proctor2, and J. Nathan Kutz3, “Supporting Information for: Discovering governing equations from data: Sparse identification of nonlinear dynamical systems,” *Nature Communications*, vol. 8, no. 1, pp. 1–38, 2017.

[11] L. Zhang, “Evaluating the Training Performance of Artificial Neural Network Using Small Time Series Segments of the Lorenz Chaotic System,” *Proceedings of the International Joint Conference on Neural Networks*, vol. 2018-July, pp. 1–8, 2018.

[12] J. Brajard, A. Carrassi, M. Bocquet, and L. Bertino, "Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model," *Journal of Computational Science*, vol. 44, p. 101171, 2020.

[13] K. Fukami, T. Murata, and K. Fukagata, "Sparse identification of nonlinear dynamics with low-dimensionalized flow representations," tech. rep.

[14] A. T. Mohan, D. Daniel, M. Chertkov, and D. Livescu, "Compressed convolutional LSTM: An efficient deep learning framework to model high fidelity 3D turbulence," *arXiv*, no. Dl, pp. 1–27, 2019.

[15] R. Han, Y. Wang, Y. Zhang, and G. Chen, "A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network," *Physics of Fluids*, vol. 31, 12 2019.

[16] A. da Ronch, K. J. Badcock, Y. Wang, A. Wynn, and R. Palacios, "Nonlinear model reduction for flexible aircraft control design," in *AIAA Atmospheric Flight Mechanics Conference 2012*, 2012.

[17] A. Da Ronch, N. Tantaroudas, S. Timme, and K. J. Badcock, "Model reduction for linear and nonlinear gust loads analysis," *AIAA Paper 2013*, 2013.

[18] M. Patterson, M. Weinstein, and A. Rao, "An efficient overloaded method for computing derivatives of mathematical functions in matlab," *ACM Transactions on Mathematical Software*, 2013.

[19] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics, Springer, 2009.

[21] Billings, *Nonlinear System Identification*, ch. 2. Chichester, UK: John Wiley & Sons, 1 ed., 2013.

[22] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, and G. Research, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," tech. rep.

[24] F. Chollet, *Deep Learning with Python*. Manning, Nov. 2017.

[25] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," 6 2015.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[27] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," 2 2014.

[28] E. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, 1962.

[29] J. Moehlis, H. Faisst, and B. Eckhardt, "A low-dimensional model for turbulent shear flows," *New Journal of Physics*, 2004.

[30] E. Lorenz, "Predictability: a problem partly solved," in *Seminar on Predictability, 4-8 September 1995*, vol. 1, (Shinfield Park, Reading), pp. 1–18, ECMWF, ECMWF, 1995.

[31] L. M. Resler, "Edward n lorenz's 1963 paper, "deterministic nonperiodic flow", in journal of the atmospheric sciences, vol 20, pages 130–141: Its history and relevance to physical geography," *Progress in Physical Geography: Earth and Environment*, vol. 40, no. 1, pp. 175–180, 2016.

[32] A. Xin-lei and Z. Li, "Dynamics analysis and Hamilton energy control of a generalized Lorenz system with hidden attractor," *Nonlinear Dynamics*, vol. 94, no. 4, pp. 2995–3010, 2018.

[33] V. Pelino, F. Maimone, and A. Pasini, "Energy cycle for the Lorenz attractor," *Chaos, Solitons and Fractals*, vol. 64, no. 1, pp. 67–77, 2014.

[34] J. Kerin and H. Engler, "On the Lorenz '96 Model and Some Generalizations," 5 2020.

[35] T. Schneider, S. Lan, A. Stuart, and J. Teixeira, "Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations," *Geophysical Research Letters*, vol. 44, pp. 396–12, 12 2017.

[36] A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian, "Data-driven prediction of a multi-scale Lorenz 96 chaotic system using deep learning methods: Reservoir computing, ANN, and RNN-LSTM," 6 2019.

[37] P. D. Dueben and P. Bauer, "Challenges and design choices for global weather and climate models based on machine learning," *Geoscientific Model Development*, vol. 11, pp. 3999–4009, 10 2018.

[38] S. Pawar, S. E. Ahmed, O. San, A. Rasheed, and I. M. Navon, "Long short-term memory embedded nudging schemes for nonlinear data assimilation of geophysical flows," *Physics of Fluids*, vol. 32, 7 2020.

[39] E. N. Lorenz, *Predictability – a problem partly solved*, p. 40–58. Cambridge University Press, 2006.

[40] *Numerical Integration of Ordinary Differential Equations*, ch. 2, pp. 65–128. John Wiley Sons, Ltd, 2014.

[41] J. L. Lumley, *Stochastic tools in turbulence [by] John L. Lumley*. Academic Press New York, 1970.

[42] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, pp. 502–504, 7 2006.

Appendices

A Convolutional Long Short-Term Memory Formulation

In this Appendix we report the formulation for the Convolutional LSTM operation. This is extracted from the original paper by Shi et al. [25].

LSTM has become the state-of-the-art recurrent-neural-network (RNN) approach for prediction of sequential data such as time series [26]. RNNs are designed to embed past-history information (carry memory over) to the next prediction. A well known problem of standard RNN is the numerical instability due to vanishing gradients for long-sequence predictions. LSTM solves this issue by incorporating two states: a hidden state $\mathscr{H}_t$, which corresponding to the short-term memory or the output from the previous time step; and a cell state $\mathscr{C}_t$, carrying over the long-term memory. The philosophy behind the LSTM is to introduce three weighted gates: the input/update gate i_t, the forget gate f_t and the output gate o_t. In brief, the forget gate decides whether to keep the information from the previous timestamps in the longer term[†]. The input gate is used to weight the information coming from the input. Last, the output gate filters the information to output.

In the standard LSTM formulation, all the gates and states calculations involve weighted fully-connected operators. While this has proven very powerful for temporal predictions, the standard fully-connected layers have the known problem of redundancy and scalability in spatial domains. To this aim, the Convolutional-based LSTM introduces a variation of the LSTM adapted to spatial data. The idea is to replace the fully-connected operators for discrete convolutions, Equation (5).

As a result, the Convolutional-LSTM operator is defined as:

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi} \star \mathscr{X}_t + W_{hi} \star \mathscr{H}_{t-1} + W_{ci} \circ \mathscr{C}_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf} \star \mathscr{X}_t + W_{hf} \star \mathscr{H}_{t-1} + W_{cf} \circ \mathscr{C}_{t-1} + b_f\right) \\
\mathscr{C}_t &= f_t \circ \mathscr{C}_{t-1} + i_t \circ \tanh\left(W_{xc} \star \mathscr{X}_t + W_{hc} \star \mathscr{H}_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo} \star \mathscr{X}_t + W_{ho} \star \mathscr{H}_{t-1} + W_{co} \circ \mathscr{C}_t + b_o\right) \\
\mathscr{H}_t &= o_t \circ \tanh\left(\mathscr{C}_t\right)
\end{aligned}
\tag{50}
$$

The various W represent the convolutional filters, b are learnable constant terms, $\star$ denotes the convolution operator Eq. (5), $\circ$ is the Hadamart (element-wise) product and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. Subscripts t and $t-1$ correspond to the current and previous time steps.

To understand better the formulation, Figure 14 provides an intuitive diagram of the complete operations involved in the convolutional-based LSTM unit.

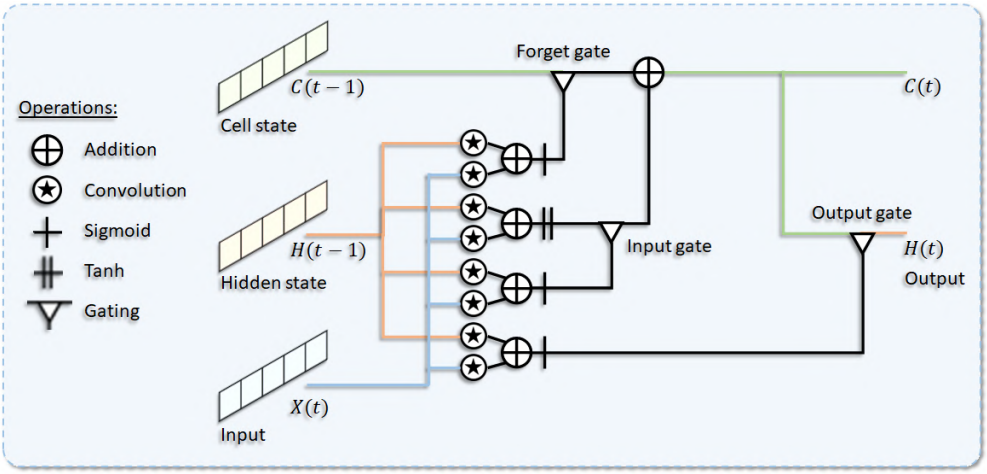


Figure 14 – Diagram of the Convolutional-LSTM operator.

[†]https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

B TPROM: Eigenvector Scaling

Matlab scales its eigenvectors such that the norm of each eigenvector is 1. Satisfying the normalisation conditions given in Eq. 9 requires a new normalisation routine. This routine could follow the Matlab code given below:

```
function[psi] = eigvec_scaling(phi, psi)

mrom = size(phi, 2);

% scale basis for projection
for i1 = 1:mrom
    % find coefficient
    C = 1./(psi(:,i1)'*phi(:,i1));

    % rescale psi
    psi(:,i1) = conj(C)*psi(:,i1);
end

end
```

As the $\bar{\phi}_r^T \phi_r = 1$ condition is already satisfied by Matlab, this code only scales to satisfy the condition $\bar{\psi}_r^T \phi_r = 1$.

C TPROM: Function Wrapper

```
function [Xdot] = func_wrapper(zr, zi, U, func_params, phi)

% coordinate transformation
z = zr + 1i*zi;
X = phi*z + conj(phi)*conj(z);

% force real values
X = real(X);

% note: first input to dynamics function (t) set to 0
[Xdot] = dynamics_function(0, X, U, func_params);

end
```

D Conv-LSTM Validation Study

This appendix expands on the justification for the final choice of architecture and hyperparameters reported in Section 5.3.1 A validation study was addressed to determine the best suited model for the prediction of the Lorenz 1996 system. In Table 7 we provide a summary of the most relevant tests. Mean and variance columns correspond to the statistics extracted from the respective probability density function for each case, with all modes merged together. The pointwise mean absolute error (MAE) agains the ground truth is also compared. The cost for training the NN model and for a time-marching simulation of $t = 20$ time units is also reported. In the first two rows we report the values for the reference ODE solution and the final data-driven model chosen. The rows underneath provide the results for the parameter change with respect to the chosen model of the second row. As a reminder, the complete configuration of the definitive model was provided in Tables 3 and 4.

We observe that there are no significant differences among the different configurations. Alternative valid candidates were the shorter dataset time and the kernel size. The shorter dataset option would be computationally more attractive but we could not verify whether the time-marching simulation could be extended beyond the final time of the training dataset. Thus, we opted for the safer option. With regards to the kernel size, both options resulted equally valid but the larger kernel size was

Table 7 – Summary of the validation study on the Conv-LSTM model.

Model Change	**Mean**	**Variance**	**MAE**	**Training [CPU-h]**	**Simulation [CPU-h]**
True solution	2.66	13.48	-	-	-
Final Conv-LSTM	2.78	15.13	3.82	1.00	0.81
Conv-LSTM layers: $3 \rightarrow 5$	2.69	19.75	4.64	1.81	1.19
Kernel size: $5 \rightarrow 3$	2.68	12.68	3.78	0.97	0.91
Channels: $35 \rightarrow 45$	2.42	9.07	4.00	1.10	0.82
Dataset time: $100 \rightarrow 50$ s	2.86	13.42	4.15	0.55	0.83
Epochs: $250 \rightarrow 500$	2.84	13.95	4.42	2.02	0.89
Loss function: Huber$\rightarrow$MSE	2.32	12.30	4.89	1.03	0.85

chosen as it was thought to provide further influence from neighboring states in space. On the contrary, the options involving the number of layers, channels and epochs implied an increase of model size without a clear performance advantage. Thus, such changes were unattractive. Last, the loss-function change resulted insensitive too.