Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Machine learning based extension of explainable model order reduction

Shubham Khatri

Computational Science and Engineering (International Master's Program)

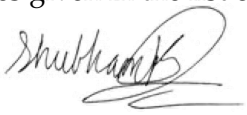
Technische Universität München

Master's Thesis

Machine learning based extension of explainable model order reduction

Author:	Shubham Khatri
Examiner:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Advisor:	Dr. Felix Dietrich, Dr. Dirk Hartmann, Dr. Lukas Failer
Submission Date:	June 14th, 2021

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.



June 14th, 2021 Shubham Khatri

Acknowledgments

I would like to extend my sincere gratitude to everyone who have helped and supported me throughout my thesis work. Without their constant support, guidance, motivation and cooperation it would have been difficult to complete this thesis work.

First, I would like to thank my advisors Dr. Dirk Hartmann, Dr. Felix Dietrich, and Dr. Lukas Failer for believing in me and providing me this wonderful opportunity. Their supervision, invaluable suggestions and motivation kept me constantly engaged with my research and brought my work to a higher level. I would like to extend special gratitude to Dr. Felix Dietrich and Dr. Lukas Failer for several discussions towards the beginning that helped me better my understanding of the subject and clarify the direction.

I would like to acknowledge my professor, Univ.-Prof. Dr. Hans-Joachim Bungartz for his valuable guidance and constant support throughout my studies and this thesis work.

I would also like to acknowledge Dr. Christoph Heinrich for accepting me as a part of his team and providing me with any support that was needed throughout the thesis work.

I am thankful to my colleagues Felix Sievers, for the insightful discussions and suggestion which helped me reach valuable conclusions as well as improved my result presentation. I would also like to acknowledge Dr. Juan Manuel Lorenzi, Diana Manvelyan, Liang Sin Go, and Rishith Ellath Meethal for providing me with the test data, server access and helpful technical discussions which helped me continue my research seamlessly.

I am grateful to my parents and my family members for their support and passionate encouragement. I would like to convey my gratitude to my friends for supporting me and always being there for me.

"Nature is written in mathematical language."

-Galileo Galilei

Abstract

One of the popular method used for Model Order Reduction (MOR) of non-linear Partial Differential Equations (PDEs) is the Proper Orthogonal Decomposition (POD) with Galerkin Projection (GP). The POD – GP method finds the optimal subspace in L_2 sense that encapsulates the system's dynamics and projects the governing PDE in this subspace. This method finds utility in large classes of problems. However, advection dominated PDEs require a large number of dimensions, or modes, to encapsulate the system dynamics properly. Therefore, the POD – GP method is observed to fail for advection dominated PDEs due to truncation of the higher modes. Often the alternative approach used for advection dominated PDEs are non-linear MOR, which are frequently based on Deep Learning methods and lack explainability. In this thesis work, we formulate a method that combines a coupled linear and non-linear MOR, such that it overcomes the issue of POD – GP while still retaining the formulation. In the proposed POD – CAE projection method, the first few POD modes are used for GP while the truncated modes are projected using CAE and are progressed using an LSTM model. We show that this formulation provides the linear MOR with necessary information about the dynamics of the truncated modes through coupling between the two models. We apply the POD – CAE formulation to solve the Burgers' equation and radiative heat equation to demonstrate the capability, flexibility, and explainability of the proposed formulation. Our results show that the formulation is able to tackle the challenges faced by the POD – GP method. Additionally, examining the low dimensional space allows the user to isolate the sources of error in the method, thereby providing explainability.

Abbreviations

MOR	Model Order Reduction
POD	Proper Orthogonal Decomposition
GP	Galerkin Projection
DEIM	Discrete Empirical Interpolation Method
DL	Deep Learning
FOM	Full Order Model
ROM	Reduced Order Model
CAE	Convolutional Autoencoder
LSTM	Long Short-Term Memory networks
PDE	Partial Differential Equations
RNN	Recurrent Neural Network
FEM	Finite Element Method

Contents

1 Introduction

Partial Differential Equations (PDEs) are ubiquitous since they describe several physical phenomena such as Fluid Flows, Electromagnetism and Quantum Mechanism. While some of these equations have an analytical solution, many require high fidelity simulations of parameterized PDEs. In general, high fidelity simulations have a very high associated computational cost and are often prohibitive for real-time or multi-query applications. The former case requires a solution of the *full order model* (FOM) or high fidelity simulation in a very short time, while the latter requires the solution of the FOM for several parameter values. To tackle these challenges *reduced order models* (ROM) are developed. The assumption behind the construction of ROM is the solution of any parameterized PDE, in the full (discrete) space, lies on a low dimensional solution manifold embedded in the full space [12]. The goal of ROMs are to approximate this solution manifold which encapsulates all the solution of the PDE for varying parameters in the parameter space.

There have been several works in the past two decades that are dedicated to the development of accurate and efficient MOR [8, 10, 23, 34, 9, 28]. One of the popular method used for model order reduction is Proper Orthogonal Decomposition (POD) with Galerkin Projection (GP) [34]. The POD method finds the optimal subspace in L_2 sense, which encapsulates the data dynamics and then the governing PDE is projected into this subspace. The projected equation can be used as a surrogate model for various applications. The POD – GP, by definition, is a linear projection method; therefore, when applied to non-linear PDE, the evaluation of non-linear term is still computationally intensive. There have been several methods which are designed to tackle this challenge [29, 22], one of the methods we would be focusing on the Discrete Empirical Interpolation Method (DEIM) [9]. DEIM reduces the computational cost of the non-linear term evaluation by empirically setting some probing points in the region of the space where system dynamics is maximum. Doing so, the non-linear term is only evaluated at a certain position in the space. The POD – GP with DEIM finds applicability in a wide array of problems. Nevertheless, the formulation is incapable of providing reliable results for advection-dominated problems since most advection dominated problems require a large dimension of the subspace, formulated using the POD method. In general, when performing a POD – GP MOR, some of these dominant modes are truncated, resulting in high error from surrogate model [12, 23].

Several data-driven MOR methods have been developed to tackle these truncation problem associated with POD – GP method [23, 10, 12]. These methods formulate a reduced space representation of the data using deep learning methods, which are then progressed (time-integrated) in the reduced space using another machine learning-based method. These methods have shown a better performance than POD – GP methods and are also

applicable to a wide array of problems. Despite the advantages of broad applicability, these methods suffer in terms of explainability, making them a less preferred choice in real application scenarios.

In this thesis work, we propose an alternative framework, POD – CAE, which combines the classical linear POD – GP based method to a data-driven method, solving the challenges of truncation as well as explainability. To achieve this goal, we define a two-part projection operator, one of which is linear and same as POD – GP, while the second is non-linear, CAE. Further, we formulate the linear and non-linear projection operator such that the projected data from the two operators lie in orthogonal subspaces. This formulation allows us to develop a coupled MOR surrogate model that overcomes truncation of the higher modes. For the non-linear projection, we perform the time integration in the encoding space using an LSTM model. We show that our formulation provides room for analysis in low dimensional space, such as error analysis, thereby providing explainability. Additionally, we also show that the proposed POD – CAE formulation is flexible and could be applied to a wide variety of problems.

The thesis work is structured as follows: In Chapter 2 we present an overview of several surrogate modelling methods. We will highlight the theoretical aspects that are important for the formulation of the POD – CAE method. In addition to this, we also highlight the work of Maulik et al. 2021 which aids in adding explainability to our formulation. In Chapter 3 we provide the complete formulation of the POD – CAE method. We also formulate the algorithms for developing an intrusive and non-intrusive surrogate model using the POD – CAE projection method. Additionally, we also put some general remarks that should be useful for applying the POD – CAE method. In Chapter 4 we apply the proposed POD – CAE algorithm to solve Burgers' equation and radiative heat transfer equation to demonstrate that the proposed model solves the aforementioned issues associated with the POD – GP method and complete data-driven approaches. We also provide the overall simulation run-time performance comparison for FOM, POD – GP with DEIM, and POD – CAE to highlight the accuracy versus performance compromise for the two surrogate models.

2 State of The Art

In this chapter we develop the theoretical foundations necessary for this thesis work. The section highlights some of the existing projection methods, time integration schemes and reduced model approximation methods popularly used in modelling of dynamical systems. For the ease of the development of the theory we consider a general PDE modelling a dynamical system. Let us consider a solution variable q in the Hilbert space $\mathcal{H}([0,T],\Omega)$ with the domain $\Omega \subset \mathbb{R}^d$, and boundaries $\Delta\Omega$, and let the variable q depend on parameters $\Gamma = \{\gamma_1, \gamma_2, \cdots\} \in \mathcal{G}$, and time, t, then we regard, in this thesis, PDEs which can be expressed as

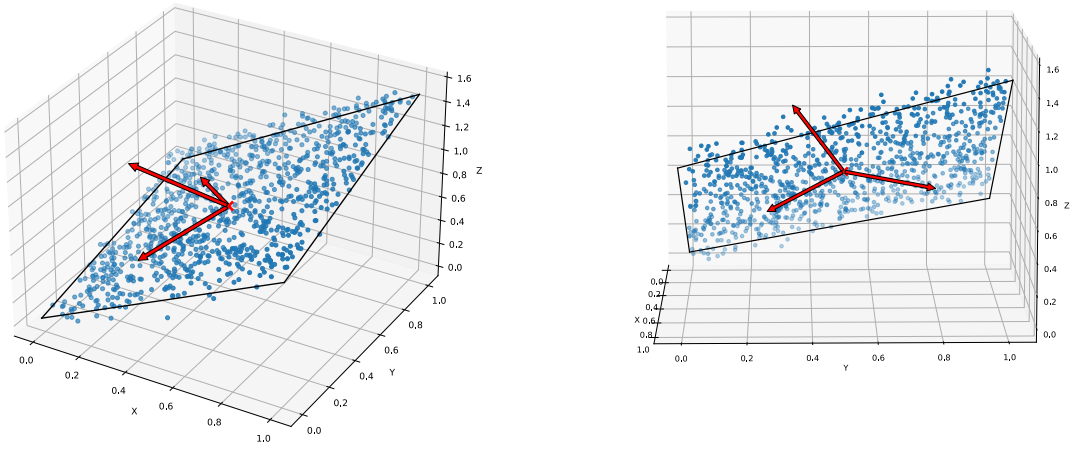
$$\begin{aligned} \dot{q}\left(\Omega,\Gamma,t\right) + \mathcal{L}\left[q\left(\Omega,\Gamma,t\right)\right] + \mathcal{N}\left[q\left(\Omega,\Gamma,t\right)\right] &= 0 \\ q\left(\Delta\Omega,\Gamma,t\right) &= q_b \\ q\left(\Omega,\Gamma,0\right) &= q_0 \quad \text{s.t.} \quad t \in [0,T)\,, \end{aligned} \tag{2.1}$$

where $\mathcal{L}$ and $\mathcal{N}$ are the linear and non-linear operators, respectively. The solution domain can be discretized using any method of choice such as Finite Elements or Finite Differences. The resulting discretized form of the Equation (2.1) can be expressed as

$$\begin{aligned} \dot{q}_h\left(\Gamma,t\right) + \mathcal{L}_d\left[q_h\left(\Gamma,t\right)\right] + \mathcal{N}_d\left[q_h\left(\Gamma,t\right)\right] &= 0 \\ q_{\Delta\Omega}\left(\Gamma,t\right) &= q_b \qquad q_b : [0,T) \times \mathcal{G} \to \mathbb{R}^{N_b} \\ q_h\left(\Gamma,0\right) &= q_0 \quad \text{s.t.} \quad q_h : [0,T) \times \mathcal{G} \to \mathbb{R}^{N}. \end{aligned} \tag{2.2}$$

2.1 Proper Orthogonal Decomposition

Proper Orthogonal Decomposition (POD) is one of the popular methods used in the domain of Model Order Reduction (MOR). The POD method finds the optimal subspace, that hold most of the information necessary for the dynamics to evolve, and are optimal in the least squares sense for a given dataset. In general, the POD method does not require any information on the process involved in generating data, such as underlying equation or discretization, making it applicable to a large variety of problems. The general intuition behind the POD method is that for a given dimension of data, we could find a low dimensional hyperplane that could embed most of the data dynamics. Figure 2.1 shows an example where the data is distributed in $3D$ space but most of the data variance lies in the shown $2D$ plane. The POD method finds the 2 directions shown by red arrows lying on the square and ignores the perpendicular direction since it contains very little information about the dynamics of data.



(a) Isometric View.

(b) YZ projected view.

Figure 2.1: The figure illustrates an example where the $3D$ data could be embedded in a $2D$ hyperplane since most of the data variance lies in the encapsulating 2D plane. The arrows represent the 3 principal directions which could be obtained with POD method.

2.1.1 Optimal Basis Construction

In this section we show that the optimal basis for the POD method is the left singularvector of the data itself. In order to show this, let $\mathcal{H}$ be the Hilbert space with inner product $\langle ., . \rangle$ and let $\mathbf{Q} \in \mathbb{R}^{N \times T}$ represent the snapshot matrix consisting of T system states, for example the solution of Equation (2.2) at T time points. Further, let the optimal subspace be $\mathcal{U}_l$, such that $l \ll N$. Then we can represent $\mathbf{Q}$ and $\mathcal{U}_l$ as follows

$$\begin{aligned} \mathbf{Q} &= [q_i \in \mathcal{H} | i = 1, 2, \cdots, T] \\ \mathcal{U}_l &= span\{\phi_1, \phi_2, \cdots, \phi_l\} \quad \text{s.t.} \quad \langle \phi_i, \phi_j \rangle = \delta_{ij}, \end{aligned} \tag{2.3}$$

where y_i represents the i^{th} snapshot of data and ϕ_j represents the j^{th} orthonormal basis. In general the snapshots are not linearly independent therefore the rank of the snapshot matrix, $\mathbf{Q}$, is $d \leq min(N, T)$. Further it can be shown there there exist a decomposition of following form also known as Singular Value Decomposition (SVD) [14]

$$\mathbf{Q} = \mathbf{U} \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{V}^T = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$$

with $\mathbf{U} = \{u_1, u_2, \cdots, u_N\} \in \mathbb{R}^{N\times N}$ being left singular vector, $\mathbf{V} \in \mathbb{R}^{T\times T}$ being right singular vector, and $\mathbf{D} = \{\sigma_1, \sigma_2, \cdots, \sigma_d\} \in \mathbb{R}^{d\times d}$ being the singular values of the snapshot matrix $\mathbf{Q}$.

Now we can pose the problem of finding the optimal subspace $\mathcal{U}_l$ as the following optimization problem

$$\max_{\phi_1,\cdots,\phi_l} \sum_{i=1}^{l}\sum_{j=1}^{T} |\langle q_j, \phi_i\rangle|^2 \quad \text{s.t.} \quad l \leq d. \tag{2.4}$$

It is shown in [34, p. 5-6] that for any choice of l the optimum $\{\phi_i\}_{i=1}^{l}$ is the left hand singular vector $\{u_i\}_{i=1}^{l}$.

2.1.2 Projection Operator

From the application point of view, we require a projection operator that projects the data to the already developed subspace, $\mathcal{U}_l$, and back from it. This projection operator can be constructed by column stacking the basis that span the subspace $\mathcal{U}_l$. Therefore the projection operator, $\mathbf{\Phi}$, can be represented as

$$\mathbf{\Phi} = [\phi_1, \phi_2, \cdots, \phi_l] = [u_1, u_2, \cdots, u_l] \in \mathbb{R}^{N\times l}, \tag{2.5}$$

and the snapshot matrix can be approximated using

$$\mathbf{Q} \approx \mathbf{\Phi} a, \tag{2.6}$$

where $a \in \mathbb{R}^{l\times T}$ are the projected snapshots. Since the projection operator, $\mathbf{\Phi}$, forms a basis the projected snapshots, a, can be computed using

$$a = \mathbf{\Phi}^T \mathbf{Q}. \tag{2.7}$$

The extent to how well the projected subspace, $\mathcal{U}_l$, represents the dynamics of the snapshot matrix is governed by the size of the subspace $\mathcal{U}_l$. The choice of l, for different applications, is rather heuristic and there is no a-priori rule, but the ratio of modelled to total energy of the snapshot can be computed using [34, p. 9]

$$\mathcal{E}(l) = \frac{\sum_{i=1}^{l} \lambda_i}{\sum_{i=1}^{d} \lambda_i}. \tag{2.8}$$

2.1.3 POD - Galerkin Projection

For any dynamical system defined by Equation (2.2) and subspace $\mathcal{U}_l$, Galerkin projection (GP) maps the dynamical system from the full order space to the subspace $\mathcal{U}_l$. This

mapping is obtained by orthogonal projection of the system using the operator defined in Section 2.1.2 [28]:

$$\begin{aligned}\mathbf{\Phi}^T\left(\dot{q}_h\left(\Gamma,t\right)+\mathcal{L}_d\left[q_h\left(\Gamma,t\right)\right]+\mathcal{N}_d\left[q_h\left(\Gamma,t\right)\right]\right)=0 \quad \forall \quad t\in[0,T) \\ \forall q_0.\end{aligned} \tag{2.9}$$

For a sufficiently large l, the POD projection of q_h on $\mathcal{U}_l$ can be described as $q_h \approx \mathbf{\Phi}\mathbf{\Phi}^T q_h = \mathbf{\Phi}a$ with $\mathbf{\Phi} \in \mathbb{R}^{N\times l}$ and $r \in \mathbb{R}^l$, whereby a solves the dynamical system. Then we can rewrite the above equation as follows

$$\begin{aligned}\mathbf{\Phi}^T\mathbf{\Phi}\dot{a}\left(\Gamma,t\right)+\mathbf{\Phi}^T\mathcal{L}_d\mathbf{\Phi}\left[a\left(\Gamma,t\right)\right]+\mathbf{\Phi}^T\mathcal{N}_d\left[\mathbf{\Phi}a\left(\Gamma,t\right)\right]=0 \\ \forall \quad t\in[0,T)\,.\end{aligned} \tag{2.10}$$

Since we formulate $\mathbf{\Phi}$ as a orthonormal projection operator we have $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}_l$, then we have

$$\begin{aligned}\dot{a}\left(\Gamma,t\right)+\mathcal{L}_p\left[a\left(\Gamma,t\right)\right]+\mathcal{N}_p\left[\mathbf{\Phi}a\left(\Gamma,t\right)\right]=0 \\ \forall \quad t\in[0,T)\,\forall \quad q_0,\end{aligned} \tag{2.11}$$

where $\mathcal{L}_p \in \mathbb{R}^{l\times l}$ and $\mathcal{N}_p \in \mathbb{R}^{l\times N}$ are the projected linear and non-linear operators, respectively.

2.2 The Discrete Empirical Interpolation Method

In Section 2.1.3 we developed the reduced order model using POD-GP method. In the Equation (2.11) , we observe that the application of linear operator has a computational complexity of $\mathcal{O}(l^2)$ while the the non-linear operator, $\mathcal{N}_p$, has a computational complexity of $\mathcal{O}(Nl)$ because of its form

$$\mathcal{N}_p = \underbrace{\mathbf{\Phi}^T}_{\mathbb{R}^{l\times N}}\underbrace{\mathcal{N}_d\left[q_h\left(\Gamma,t\right)\right]}_{\mathbb{R}^N}. \tag{2.12}$$

The Discrete Empirical Interpolation Method (DEIM) [9] overcomes this computational issue by approximating the non-linear term. For the sake of simplicity let $f(t) = \mathcal{N}_d\left[q_h\left(\Gamma,t\right)\right]$. Then DEIM aims to construct a low-dimensional approximation of $f(t)$ of the form

$$f(t) \approx \mathbf{W}c(t), \tag{2.13}$$

where $W = [w_1, w_2, \cdots, w_m] \in \mathbb{R}^{N\times m}$ is the projection operator and $c(t) \in \mathbb{R}^m$ is the coefficient vector. The DEIM constructs this projection operator and the coefficient matrix in similar fashion as POD. We consider an optimal m dimensional subspace, $\mathcal{D}_m = span\left\{w_1, w_2, \cdots, w_m\right\}$ such that $\langle w_i, w_j\rangle = 1$, which can represent the dynamics of the non-linear term $f(t)$. The optimal subspace is spanned by the left hand singularvector of the non-linear snapshot matrix.

In general, $f(t) = \mathbf{W}c(t)$ is an overdetermined system, therefore we could select certain rows from both side of the equation. Hence, we define a selection matrix $\mathbf{P} = [e_{\rho_1}, e_{\rho_2}, \cdots, e_{\rho_m}] \in \mathbb{R}^{N\times m}$, where $e_{\rho_i} = \left[\underbrace{0, 0, \cdots, 0}_{i-1}, 1, \underbrace{0, \cdots, 0}_{N-i}\right] \in \mathbb{R}^N$. If the value $\left(\mathbf{P}^T\mathbf{W}\right)$ is non-singular then we could compute the coefficient vector, $c(t)$, using the relation

$$\mathbf{P}^T f(t) = \left(\mathbf{P}^T\mathbf{W}\right) c(t). \tag{2.14}$$

Further, we could also write the approximation of non-linear term $f(t)$ from the subspace $\mathcal{D}_m$ as

$$f(t) \approx \mathbf{W}c(t) = \hat{f}(t) = \mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}\mathbf{P}^T f(t). \tag{2.15}$$

Here the term $\mathbf{P}^T f(t)$ is never calculated explicitly since it would require a matrix multiplication with computational cost of $\mathcal{O}(Nm)$. The term $\mathbf{P}^T f(t)$ can be seen as measurement of non-linearity at m selected points, therefore it has a complexity of $\mathcal{O}(m)$. Putting this approximation in Equation (2.12) we get

$$\begin{aligned} \mathcal{N}_p &= \underbrace{\mathbf{\Phi}^T\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}}_{\mathbb{R}^{l\times m}}\underbrace{\mathbf{P}^T\mathcal{N}_d\left[q_h\left(\Gamma, t\right)\right]}_{\mathbb{R}^m} \\ \mathcal{N}_p &= \underbrace{\mathbf{K}}_{\mathbf{\Phi}^T\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}}\underbrace{\mathcal{N}_d\left[\mathbf{P}^T q_h\left(\Gamma, t\right)\right]}_{\mathbb{R}^m}. \end{aligned} \tag{2.16}$$

The term $\mathbf{K}$ can be precomputed and therefore the evaluation of term $\mathcal{N}_p$ requires a matrix multiplication with complexity of $\mathcal{O}(lm)$. In general $m \ll N$ and $m > l$. While the subspace $\mathcal{D}_m$ is constructed by the left hand singularvector of non-linearity, the selection matrix $\mathbf{P}$ is constructed empirically using the Algorithm 1 .

Algorithm 1: DEIM

Input : $\{w_i\}_{i=1}^m \subset \mathbb{R}^N$ linearly independent
Output: $\rho = [\rho_1, \rho_2, \cdots, \rho_m]^T \in \mathbb{R}^m, \quad \mathbf{P} \in \mathbb{R}^{N\times m}$

$[|\rho|, \rho_1] = \max\{|w_1|\}$;
$\mathbf{W} = [w_1]$, $\mathbf{P} = [e_{\rho_1}]$, $\rho = [\rho_1]$;
for $i \leftarrow 2$ **to** m **do**
  Solve $\left(\mathbf{P}^T\mathbf{W}\right)c = \mathbf{P}^T w_l$ for c ;
  $r = w_l - \mathbf{W}c$;
  $[|\rho|, \rho_l] = \max\{|r|\}$;
  $\mathbf{W} \leftarrow [\mathbf{W}, w_l]$, $\mathbf{P} \leftarrow [\mathbf{P}, r_{\rho_l}]$, $\rho \leftarrow \begin{bmatrix}\rho \\ \rho_l\end{bmatrix}$;
end

Illustrative Example

An illustrative example taken from [9] is given by a non-linear parameterized function

$$s(x;\mu) = (1-x)\cos(3\pi\mu(x+1))e^{-(1+x)\mu}, \quad x \in [-1,1], \quad \mu \in [1,\pi] \tag{2.17}$$

which is used to construct a snapshot matrix S. The snapshot matrix is constructed using 100 equidistant discrete x points and 51 equidistant parameter values, μ. Figure 2.2 shows the shape of the first 6 POD modes and the DEIM points obtained using Algorithm 1 . It can be seen that DEIM selects the points in the region which covers most of the dynamics of the system. Figure 2.3 shows the approximation of $s(x;\mu)$ using 10 dimensional DEIM as compared to the exact function. The green lines show the DEIM measurement locations. We notice that 10 dimensional DEIM is able to approximate the behavior of the real function very well and the small error only occurs in the region beyond the last DEIM point.

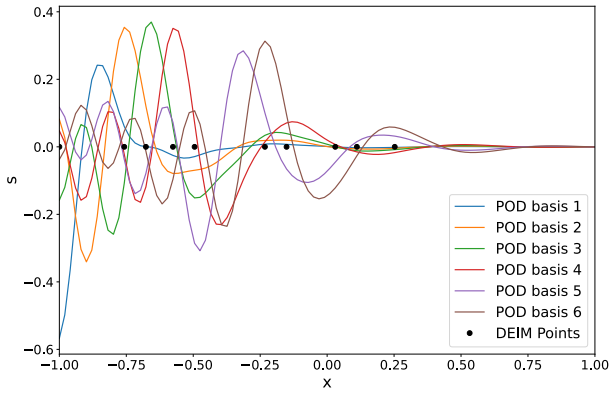


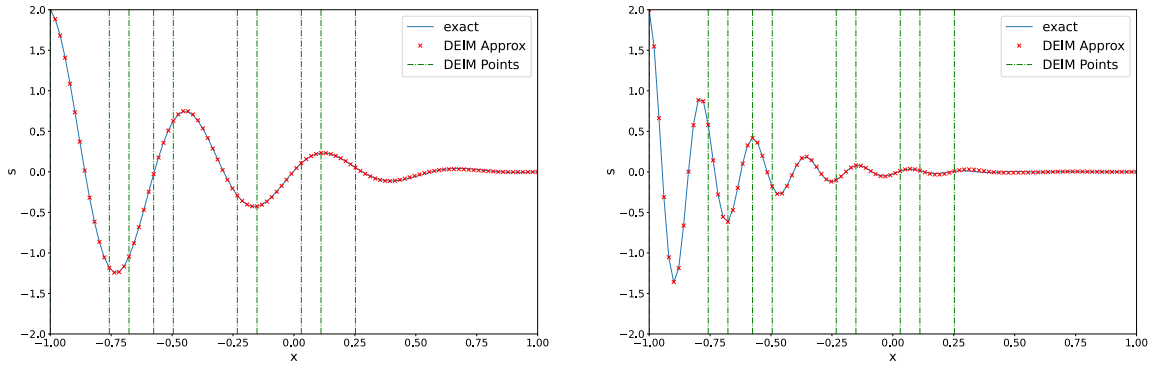
Figure 2.2: The figure shows the first 6 POD bases and the selected DEIM points computed from Algorithm 1

2.3 Operator Inference

Operator Inference (OI) is a class of non-intrusive data-driven method which can be used to approximate the operators involved in a given form of PDE [24]. In order to infer the operators the OI solves a regression problem which yields the operators that provide best reconstruction of the snapshots.

We can represent any semi-discretized PDE with leading quadratic term as follows

$$\begin{aligned} \dot{q}_h(t) &= c + \mathbf{A}q_h(t) + \mathbf{H}(q_h(t) \otimes q_h(t)) + \mathbf{B}u(t) \\ \dot{q}_h(t_0) &= q_0, \qquad t \in [t_0, t_f] \end{aligned} \tag{2.18}$$



(a) Exact vs DEIM for $\mu = 1.17$

(b) Exact vs DEIM for $\mu = 3.1$

Figure 2.3: DEIM approximation for problem 2.17 using 10 dimensions as compared to exact solution for $\mu = 1.17$ and 3.1

Further, any PDE in a polynomial form can be expressed in the leading quadratic form by application of lifting [24]. Therefore, for any PDE in above mentioned form we can express the regression problem as follows

$$\min_{c,\mathbf{A},\mathbf{H},\mathbf{B}} \sum_{j=1}^{k} ||c + \mathbf{A}q_h(t) + \mathbf{H}(q_h(t) \otimes q_h(t)) + \mathbf{B}u(t) - \dot{q}_h(t)||_2^2 \tag{2.19}$$

The data used for prediction of operators in above equation is generally noisy, due to numerical error, and the problem itself is overdetermined. Therefore overfitting of operators on data leads to bad predictive performance in the domain of interest $[t_0, t_f]$. We can therefore introduce Tikhonov regularization to the above problem and write it in matrix form as follows

$$\min_{\mathbf{O}} ||\mathbf{D}\mathbf{O} - \mathbf{R}||_2^2 + ||\mathbf{\Xi}\mathbf{O}||_2^2 = \left|\left| \begin{bmatrix} \mathbf{D} \\ \mathbf{\Xi} \end{bmatrix} \mathbf{O} - \begin{bmatrix} \mathbf{R} \\ 0 \end{bmatrix} \right|\right|_2^2 \tag{2.20}$$

Where

- $\mathbf{O} = [c \quad \mathbf{A} \quad \mathbf{H} \quad \mathbf{B}]$: Unknown operators
- $\mathbf{D} = \begin{bmatrix} 1_k & \mathbf{Q}^T & (\mathbf{Q} \otimes \mathbf{Q}) & \mathbf{U}^T \end{bmatrix}$: Known Data
- $\mathbf{Q} = [q_0, q_1, \cdots, q_k]$: Snapshot matrix of the quantity of interest q
- $\mathbf{U} = [u_0, u_1, \cdots, u_k]$: Snapshot matrix of the inputs
- $\mathbf{R} = [\hat{q}_0, \hat{q}_1, \cdots, \hat{q}_k]$: Snapshot matrix of time derivatives
- $\mathbf{\Xi}$: Full rank regularizer. For L_2 regularization $\mathbf{\Xi} = \lambda\mathbf{I}$, with $\mathbf{I}$ being the identity matrix.

Solving the optimization problem in Equation (2.20) enables to infer the operators in a non-intrusive fashion, given only the data snapshots.

2.4 Autoencoders

Autoencoders also sometimes referred as Non-Linear Principal Component Analysis are part of the family of Deep Learning (DL) based MOR methods. Autoencoders are a class of Artificial Neural Networks which learns a low dimensional representation of the data employing unsupervised learning methods [21]. The Autoencoders are a composition of two approximating functions the first being the encoder, which projects the data to latent space, and second being the decoder, that reprojects data from the latent space to the original space.

For the task of MOR we are generally interested in constructing a network such that the encoder projects the data to a lower dimensional subspace, such a network is called Undercomplete Autoencoders [15]. We can now formally formulate autoencoders as

$$\begin{aligned} X &\approx g\left(f\left(X\right)\right) =: \hat{X} \\ \kappa &= f(X), \quad X \in \mathcal{X}, \quad \kappa \in \mathcal{F}, \end{aligned} \tag{2.21}$$

where $g : \mathcal{F} \to \mathcal{X}$ and $f : \mathcal{X} \to \mathcal{F}$ are the decoder and encoder functions, respectively, and κ is the projection of the data. In order to construct these projection operators we define the learning task

$$\min_{W_1, W_2} \left\{L\left(X, g_{W_1}\left(f_{W_2}\left(X\right)\right)\right)\right\} \tag{2.22}$$

where L is the loss that penalizes dissimilarity between $\hat{X}$ and X. For a continuous value function this loss could be the mean squared difference. Depending on the choice of architecture for f and g as well as the choice of loss function, we could have several types of autoencoders[15] such as Convolutional Autoencoders (CAE) [26], Variational Autoencoders (VAE) [20], etc. For this thesis work we focus specifically on CAE.

Convolutional Autoencoders

Convolutional Autoencoders (CAE) are the class of autoencoders where the encoder function, f, and the decoder function, g, are constructed using convolutional layers. Each of these convolutional layers are constructed using convolutional filters, $\{\Upsilon_i \in \mathbb{R}^{F_1 \times F_2 \times \cdots \times F_m} \mid i = 1, 2, \cdots, K\}$, which convolve with the inputs, $I \in \mathbb{R}^{N_1 \times N_2 \times \cdots N_m}$, followed by an activation function, φ. Since the input data is always discrete we can formally define the

output, $\mathbf{G}$, of the convolution layer as follow[13, p. 113]

$$\begin{aligned}\mathbf{G}\left[r_1, r_2, \cdots, r_K\right] &= \varphi\left(\Upsilon \otimes I + B\right) \\ \mathbf{G}\left[r_1, r_2, \cdots, r_K\right] &= \varphi\left(\sum_i \sum_j \cdots \sum_q \Upsilon_k\left[i, j, \cdots, q\right] I\left[r_1 - i, r_2 - j, \cdots, r_m - q\right]\right. \\ &\qquad \left. + B_k\left[r_1, r_2, \cdots, r_m\right]\right)_{k=1}^{K}\end{aligned} \tag{2.23}$$

The dimensions of the output along each axes is determined by the size of the input, convolution filter and number of filters using following

$$D_i = \frac{N_i - F_i}{S} + 1 \tag{2.24}$$

Where S is the stride length of the convolution filter application [1]. Further, application of more than one filter adds an additional dimension to the output. Therefore the output $\mathbf{G} \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_m \times K}$. For the sake of simplicity we refer to the last axis obtained by application of multiple filters as the *depth* of the output. Figure 2.4 shows a simple case where $3D$ filter is applied on a $3D$ input. We can observe that the size of the output is smaller than that of input. Further the dimensionality of the output is same as input. Additionally, for each filter, we obtain one $3D$ output which can be stacked along the 4^{th} dimension.

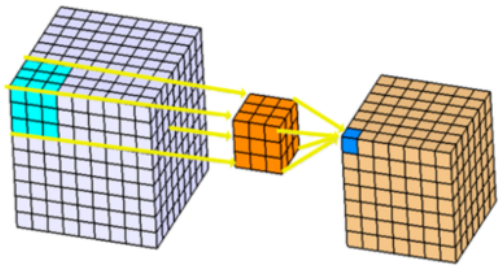


Figure 2.4: The figure shows how a $3D$ convolution filter is applied on a $3D$ input. For each $3D$ filter we generate a $3D$ output which is stacked along the 4^{th} dimension. The cyan block on the left represents a part of the $3D$ input tensor, the orange block represents a $3D$ filter which is applied on the input. The blue cube represents the output obtained by the application of the convolution filer on the cyan block.
The image is taken from [30] to aid in explanation.

Although the convolution layer performs dimensionality reduction on its own it is common to periodically insert pooling layers between successive convolutional layers. The

pooling layers perform dimensionality reduction independently on every depth slice of the input and resizes it along all the other axes. Pooling operations help in reduction of training parameters significantly. A pooling layer operates in a similar fashion as a convolution layer except instead of application of convolution filter on a features of input it performs certain specified aggregation operation, such as computation of average, maximum or L^2 norm. One of the most popular pooling operation is max-pooling. Figure 2.5 shows an example of $2D$ maxpooling operation where pooling is done with a filter size of 2 and stride of 2. The output dimension of pooling operation can also be computed using Equation (2.24) .

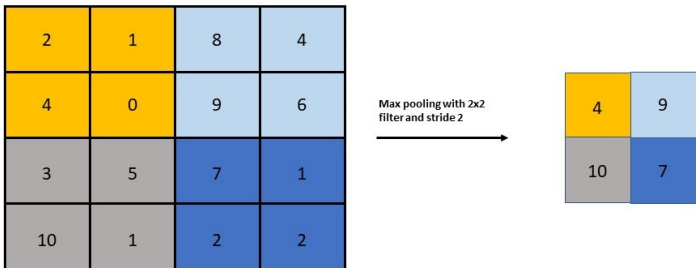


Figure 2.5: The figure shows maxpooling operation performed on a $2D$ input with stride 2 and filter size 2×2. The matrix on left shows the input where the four highlighted areas are the region where pooling operations are performed. The right matrix is the output after the pooling application.
The image is taken from [1] to aid in explanation.

The two parts of the convolutional autoencoders are constructed by stacking blocks convolutional layers followed by pooling layer. The architecture of each block is determined heuristically and is generally problem dependent. Figure 2.6 shows a generic architecture of a convolutional autoencoder. The grey box on the left represents an encoder unit which is constructed by stacking of convolutional blocks as described before. The decoder unit on the right is constructed in a similar fashion such that the output of the network has same size as the input. Although the output of the entire network is the reconstruction of the input we are mainly interested in the encoded inputs for this thesis work.

2.5 Long short-term memory networks

A Recurrent Neural Network (RNN) is a class of neural networks which are designed for temporal prediction task. As the name suggests RNN are recurrent in nature and therefore the output for any given input not only depends on the current input but also on the past states. It is easy to observe the similarities between a RNN and a classical time integration scheme such as Runge–Kutta methods. Therefore we could also see an RNN as a data

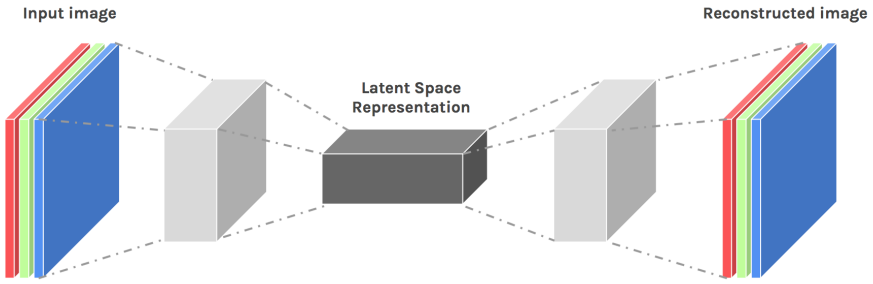


Figure 2.6: The figure depicts a general setup for convolutional autoencoder. The grey box on the left side of Latent Space Representation represents an encoder unit while the right side represents a decoder unit. The Latent Space Representation is the projection of inputs on a low dimensional manifold.
The image is taken from [25] to aid in explanation.

driven time integration method. Any RNN can be expressed as follows

$$\begin{aligned} O_n &= O(I_n) = f(I_n, O_{n-1}, \cdots, O_{n-k}) \\ O_1 &= O(I_1) = f(I_1, I_0, \cdots, I_{1-k}) \end{aligned} \tag{2.25}$$

Where:

- f: The function approximated by RNN
- I_n: n^{th} input state
- O_n: n^{th} output state
- k: the size of temporal window that impacts any output state.

We can observe that, because of inherent formulation of RNN we need at least k initial state to make the first prediction.

A Long short-term memory (LSTM) networks is a special kind of Recurrent Neural Network (RNN) which is designed to tackle the issue with the traditional RNN. Similar to any Neural Network, RNN also adjust the weights during training step to formulate the approximation function. The traditional RNN tend to saturate the adjustment of weights at an early stage of optimization process leading to a short memory and causing an inability for the RNN to make accurate predictions for the cases where output is dominated by one or more of the older states. The LSTM network overcomes these issues by network formulation that allows non-zero gradients [18]. For any arbitrary input a we can express

LSTM network output as follows

$$\begin{aligned}
\text{input gate:} \quad & \mathbf{G}_i = \sigma_g\left(W_i a + U_i h_{t-1} + b_i\right) \\
\text{forget gate:} \quad & \mathbf{G}_f = \sigma_g\left(W_f a + U_f h_{t-1} + b_f\right) \\
\text{output gate:} \quad & \mathbf{G}_o = \sigma_g\left(W_o a + U_o h_{t-1} + b_o\right) \\
\text{cell input state:} \quad & s_t = \mathbf{G}_f \otimes s_{t-1} + \mathbf{G}_i \otimes \sigma_c\left(W_c a + U_c h_{t-1} + b_c\right) \\
\text{output :} \quad & h_t = \mathbf{G}_o \otimes \sigma_h(s_t)
\end{aligned} \tag{2.26}$$

Where the initial value $c_0 = 0$ and $h_0 = 0$ and operator $\otimes$ is the Hadamard Product. Further the variables are as follows [18]

- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$: Weight matrices
- $b \in \mathbb{R}^h$: bias vector
- d: number of input features
- h: number of hidden units
- σ_g: sigmoid function
- σ_c: hyperbolic tangent function
- σ_h: hyperbolic tangent function

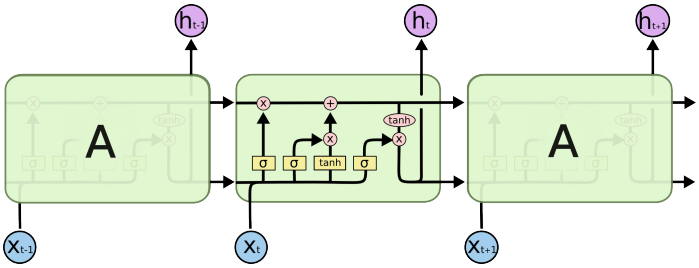


Figure 2.7: The figure shows a LSTM block with input, forget and output gates multiple such blocks are stacked to form the complete LSTM network. x_t represents the input at any time stamp t and h_t represents the output of network at time stamp t.
The image is taken from [2] to aid in explanation.

From Equation (2.26) we can compute the total number of parameters in an LSTM block

as follows

$$\begin{aligned}
\text{input gate:} &\quad h \times d + h \times h + h = (h + d + 1) \times h \\
\text{forget gate:} &\quad h \times d + h \times h + h = (h + d + 1) \times h \\
\text{output gate:} &\quad h \times d + h \times h + h = (h + d + 1) \times h \\
\text{cell input state:} &\quad h \times d + h \times h + h = (h + d + 1) \times h \\
\text{total parameters:} &\quad 4 \times (h + d + 1) \times h
\end{aligned} \tag{2.27}$$

Figure 2.7 represents a block of LSTM network which takes an input x_t at any time t and output from previous time stamps to generate the output at the current time h_t.

2.6 Solving partial differential equations using Autoencoder and Long short-term memory networks

In this section we describe the NN based MOR method from the work of Maulik et al., 2021. This method is particularly interesting for the thesis work since it develops a reduced order surrogate model using CAE and LSTM. The author uses a CAE model which constructs an encoder that can be used to project the data to a lower dimensional space. The obtained projections are then integrated using an LSTM model and therefore the overall surrogate model consists of a CAE projector and LSTM integrator. The CAE and LSTM models are trained using the provided snapshot matrix for various choices of parameters and time. Figure 2.8 describes the proposed idea for solution of 1D burgers' equation in the encoding space.

It should be noted that the previous attempts on using NN based surrogate model have focused on a combined training of the CAE and LSTM making it difficult to analyse the inputs and outputs in the embedded space. This is primarily because the output of the LSTM does not necessarily lie in the same domain as the encoded inputs leading to added difficulties in the analysis, as well as explainability. Section 4.1 describes through an experiment how the combined training can lead to difficulties in explainability. Therefore, to ease the analysis of the encoding space the proposed CAE – LSTM network trains the CAE and the LSTM models separately. We now consider an example from the work of Maulik et al. 2021

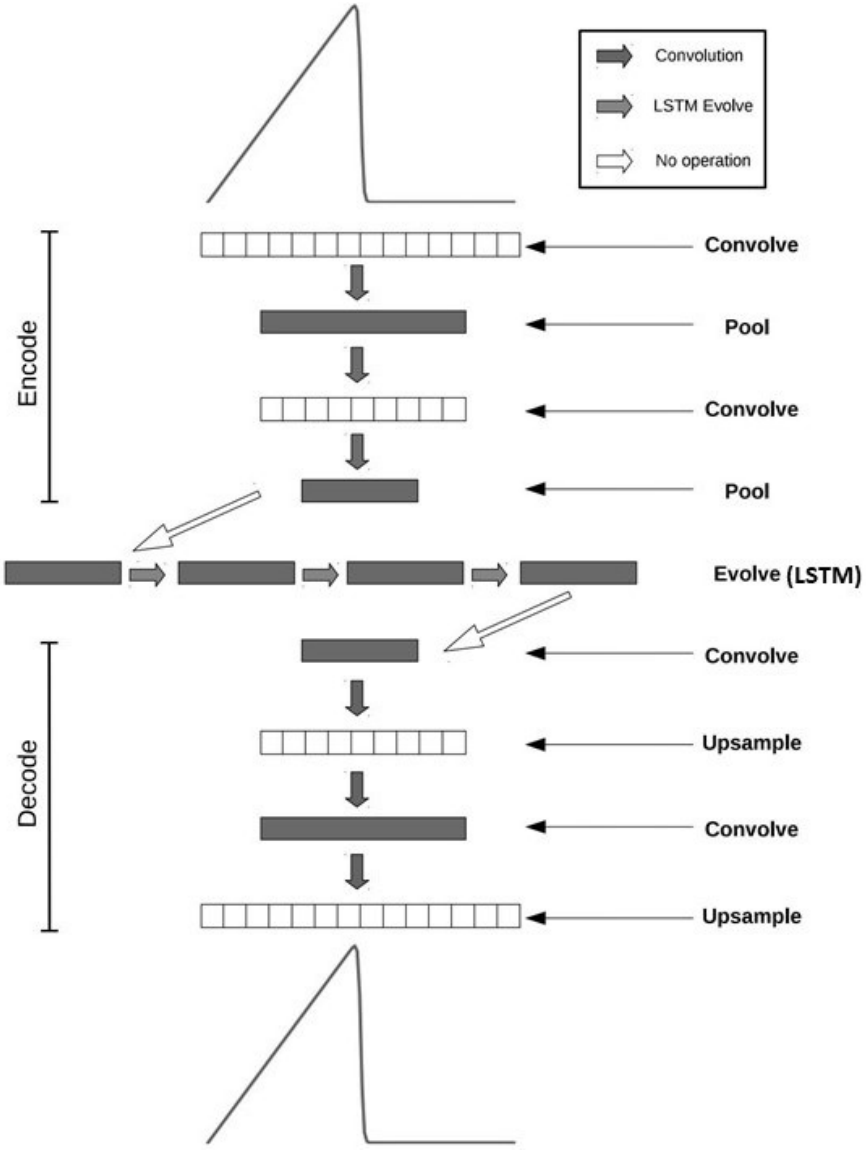


Figure 2.8: The figure shows the process of dimensionality reduction and time integration using CAE – LSTM model for 1D viscous burgers' equation. The initial condition is first projected to lower dimensional space using the encoder, which is then evolved in time domain using LSTM. The final state is then projected back to full space using the decoder.
The image is taken from [23] to aid in explanation.

Illustrative Example

We consider a one-dimensional viscous Burgers' equation example from [23] expressed as following

$$\begin{aligned} \dot{u} + u\frac{\partial u}{\partial x} &= \nu \frac{\partial^2 u}{\partial x^2} \\ u(x,0) = u_0 &= \frac{x}{1+\sqrt{\frac{1}{t_0}}e^{Re\frac{x^2}{4}}}, \quad x \in [0, L], \quad t \in [0, t_{end}] \\ u(0,t) &= 0 \\ u(L,t) &= \frac{\frac{L}{t+1}}{1+\sqrt{\frac{t+1}{t_0}}e^{Re\frac{L^2}{4t+4}}}, \quad t_0 = e^{\frac{Re}{8}} \quad \text{and} \quad Re = \frac{1}{\nu} \end{aligned} \tag{2.28}$$

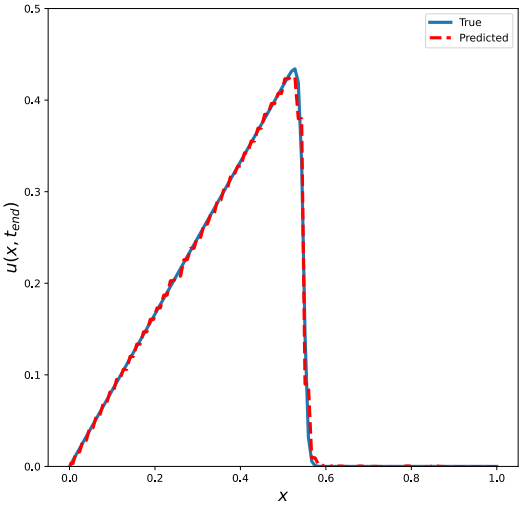
For Equation (2.28) the analytical solution exists and is given by

$$u(x,t) = \frac{\frac{x}{t+1}}{1+\sqrt{\frac{t+1}{t_0}}e^{Re\frac{x^2}{4t+4}}} \tag{2.29}$$

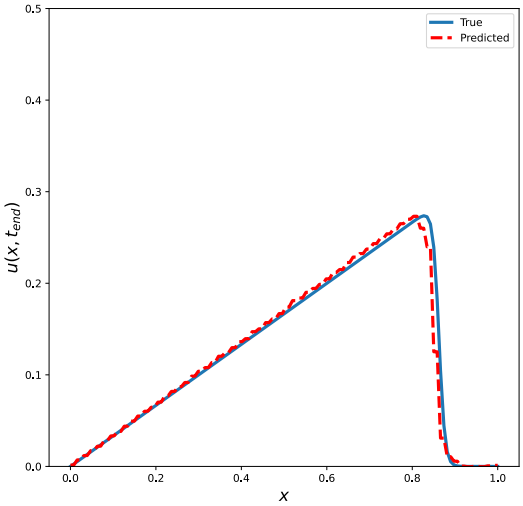
We solve the Equation (2.28) using POD – GP and CAE – LSTM model with the parameters listed in Table 2.1 . Figure 2.9 compares the results for the two method. We can notice for an advection dominated problem and non-smooth initial condition the POD projection for the initial condition itself consists of large errors. Further, these error are present at every time step t and accumulate during the time integration. We can also see that an increase in the number of modes reduces the oscillatory behaviour but an accurate prediction for the POD – GP method requires a large number of modes. On the other hand the CAE – LSTM model, constructed with 2 encoding modes, is able to accurately project the initial condition. As a result the prediction at the time t_{end} has small error arising from numerical error at each integration step and small projection error.

Table 2.1: Parameters used for development of POD – GP, and CAE – LSTM surrogate model in [23].

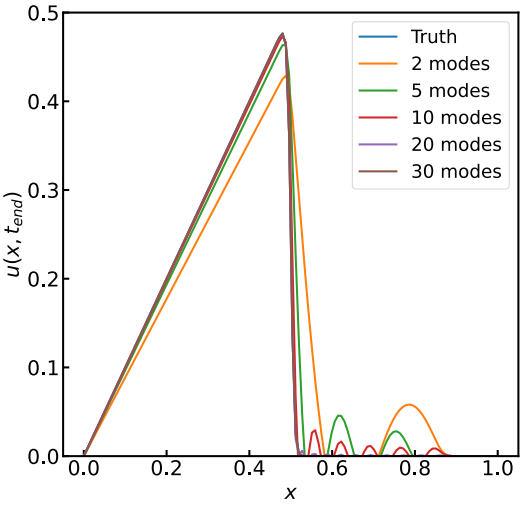
Parameters	Values
General	
L	1
t_{end}	2
Re	$\{100, 200, \cdots, 1900\}$
dx	$1/128$
dt	0.02
POD – GP	
POD Modes	$\{2, 5, 10, 20, 30\}$
CAE – LSTM	
Encoding Space Dimension	2
Convolution type	1D convolutions
Pooling	Max pooling
Number of Convolutional layers in encoder	6
Number of Convolutional layers in decoder	6
Size of LSTM output	2 (same as encoding space dimension)
LSTM temporal window size	10



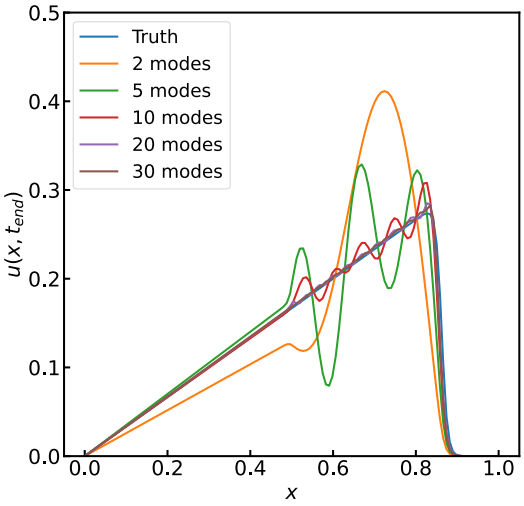
(a) Initial condition projection using CAE



(b) Integrated $u(x,t)$ at $t = t_{end}$ using CAE – LSTM



(c) Initial condition projection using POD



(d) Integrated $u(x,t)$ at $t = t_{end}$ using POD – GP

Figure 2.9: The figure compares the solution of Equation (2.28) using CAE – LSTM method and POD – GP method for $Re = 1000$. We can observe that the POD – GP method is unable to perform well even for a choice of high number of modes, on the other hand the CAE – LSTM method is able to capture the dynamics well due to non-linear projection.

3 Machine Learning based extension of explainable Model Order Reduction

Research in application of ML for solving PDE has shown that ML techniques can be used in different capacities such as a direct solver[32, 27, 19, 33], parameterization technique [6, 16], or model order reductor [10, 12, 5, 17, 23]. In the domain of Model Order Reduction (MOR), several studies have shown that the ML based methods perform far better than classical method both in terms of accuracy and online computational cost [23, 10, 17]. One of the common issues in these methods is the lack of explainability due to missing numerical analysis and related theory behind the Neural Networks (NN). We partially tackle this explainability issue in the proposed Proper Orthogonal Decomposition coupled with Convolutional Autoencoder (POD – CAE) method by taking predominantly POD based MOR and using the CAE as a corrector.

The fundamental assumption behind any MOR method is that the solution from simulations using large number of degrees of freedom lies close to a low dimensional manifold, also referred as solution manifold, embedded in the high-dimensional space. Therefore it is possible to express the solution of the PDE using the intrinsic coordinates on the solution manifold. For any non-stationary PDE of form Equation (2.2) with parameter $\Gamma \in \mathbb{R}^{n_p}$, if has a unique solution at every time t, has at most $n_p + 1 \ll N$ intrinsic dimension of the solution manifold [12]. In general for a complex problem, the solution manifold has a complicated shape and therefore the mapping of the solution from high-dimensional FOM space to solution manifold space requires a non-linear operator. Development of such a non-linear mapping is highly problem and parameter dependent, further there is no well established method to accurately and uniquely determine the non-linear mapping operator. This brings the necessity to use data-driven methods such as Autoencoders which find a non-linear mapping from the high-dimensional FOM space to the approximate solution manifold space. As mentioned above using Autoencoders alone leads to a non-explainable mapping therefore we propose a framework which develops a mapping that could be expressed as a sum of linear and non-linear mapping operator. This formulation allows the user to choose the number of linear dimensions as well as non-linear dimension, further for appropriate choice of reduced linear manifold, obtained through say Proper Orthogonal Decomposition, partial explainability can be easily achieved. In addition to this, we can inspect the intrinsic coordinates obtained from non-linear operator, Autoencoders, to understand the quality of projection and shape of the solution manifold. With these tools in hand we can easily monitor the cause of the error in the solution, and when necessary, alter the parameters to obtain better results.

3.1 Framework Construction

We now formulate the proposed idea in a more concrete mathematical form for intrusive and non-intrusive approaches to MOR. For the sake of ease, we will refer to the dimension of reduced linear manifold, obtained using the POD method, as the number of linear modes and represent it with k. Similarly, we will refer to the dimension of reduced non-linear manifold, obtained using Convolutional Autoencoders, as number of non-linear modes and represent it with m.

In order to formulate the method let us assume that the data matrix $\mathbf{Q} \in \mathbb{R}^{N\times T}$ is generated by some process modelled by the PDE expressed in Equation (2.1) . Further, let us reconsider Equation (2.5) and Equation (2.6) with $l = N$ (in contrast to general POD assumption, $l \ll N$) then we can write the full POD for a snapshot matrix $\mathbf{Q}$ by

$$\mathbf{Q} = \sum_{i=1}^{N} \mathbf{\Phi}_i a_{i*} = \mathbf{\Phi} a, \qquad \mathbf{\Phi} \in \mathbb{R}^{N\times N}, \;\; a \in \mathbb{R}^{N\times T}. \tag{3.1}$$

Now for any number of linear modes, we can rewrite the above equation as follows

$$\mathbf{Q} = \underbrace{\sum_{i=1}^{k} \mathbf{\Phi}_i a_{i*}}_{S_1} + \underbrace{\sum_{i=k+1}^{N} \mathbf{\Phi}_i a_{i*}}_{S_2}. \tag{3.2}$$

It should be noted that a full POD represents the complete dynamics of the data in FOM space and therefore the exact equality holds in Equation (3.2) .

For $k \ll N$, the S_1 in Equation (3.2) is equivalent to performing POD projection of the snapshot matrix, we take S_1 as the linear projection for our model. Let us define the subspace spanned by first k POD bases as follows

$$\hat{\mathbf{\Phi}} = \{\mathbf{\Phi}_i | i = 1, \cdots, k\} \tag{3.3}$$

The S_2 in Equation (3.2) represent the residual modes that are not covered by first k POD modes, further the S_2 lies in a space $\mathbf{\Psi} = \left\{\mathbf{\Phi} \setminus \hat{\mathbf{\Phi}}\right\} \in \mathbb{R}^{N\times N-k}$, which is high-dimensional. Now we can rewrite the Equation (3.2) in matrix form as follows

$$\mathbf{Q} = \underbrace{\hat{\mathbf{\Phi}} a_{\hat{\Phi}}}_{S_1} + \underbrace{\mathbf{\Psi} a_{\Psi}}_{S_2}. \tag{3.4}$$

Further, we can project the S_2 using a non-linear mapping constructed with CAE, as follows

$$\begin{aligned} S_2 &\approx \mathbf{\Psi}\left[\mathcal{D}(\mathcal{E}(\mathbf{\Psi}^T S_2))\right] = \mathbf{\Psi}\left[\mathcal{D}(\mathcal{E}(\mathbf{\Psi}^T\left[\mathbf{Q} - S_1\right]))\right], \\ \chi &= \mathcal{E}(\mathbf{\Psi}^T\left[S_2\right]), \qquad \chi \in \mathbb{R}^m, \end{aligned} \tag{3.5}$$

where:

- χ: The encoding space or the intrinsic coordinates space of reduced non-linear manifold.
- $\mathcal{E} : \mathbb{R}^{N-k} \rightarrow \mathbb{R}^m$: Encoder, projects the data from $\mathbf{\Psi}$ to χ.
- $\mathcal{D} : \mathbb{R}^m \rightarrow \mathbb{R}^{N-k}$: Decoder, project the data from the encoded space back to FOM space.

It should be noted that the S_2 lies in a subspace spanned by $\mathbf{\Psi} = \{\Phi_{k+1}, \Phi_{k+2}, \cdots, \Phi_N\}$ which is orthogonal to $\hat{\mathbf{\Phi}}$. Further, the CAE constructs a mapping from $\mathbf{\Psi}$ to the reduced non-linear manifold embedded in the subspace $\mathbf{\Psi}$ itself, therefore the S_2 as well as χ both are orthogonal to $\hat{\mathbf{\Phi}}$ and S_1. Additionally using the orthogonality properties we also have following

$$\begin{aligned} \left\langle \hat{\mathbf{\Phi}}, \mathbf{\Psi} \right\rangle = 0, \quad \left\langle \hat{\mathbf{\Phi}}, \hat{\mathbf{\Phi}} \right\rangle = \mathbf{I}_k, \quad \langle \mathbf{\Psi}, \mathbf{\Psi} \rangle = \mathbf{I}_m, \\ \left\langle \hat{\mathbf{\Phi}}, S_2 \right\rangle = 0, \quad \langle \mathbf{\Psi}, S_1 \rangle = 0. \end{aligned} \tag{3.6}$$

Now we can express our final POD – CAE decomposition as follows

$$\mathbf{Q} \approx \hat{\mathbf{\Phi}} \underbrace{a_{\hat{\Phi}}}_{\text{POD, lies in} \in \mathbb{R}^k} + \mathbf{\Psi} \left[\mathcal{D} \left(\underbrace{\mathcal{E} \left(\mathbf{\Psi}^T \left[\mathbf{Q} - \hat{\mathbf{\Phi}} a_{\hat{\Phi}} \right] \right)}_{\text{CAE lies in} \in \mathbb{R}^m} \right) \right], \qquad k + m \ll N,$$

$$\mathbf{Q} \approx \hat{\mathbf{\Phi}} a_{\hat{\Phi}} + \mathbf{\Psi} \left[\mathcal{D}(\chi) \right]. \tag{3.7}$$

With this decomposition we now construct the ROM for intrusive and non-intrusive method.

3.1.1 Intrusive: Proper Orthogonal Decomposition with Galerkin Projection

For a intrusive MOR we reformulate the POD - Galerkin Projection method with the POD – CAE decomposition. Let us consider Equation (2.2) , for a specific set of Γ and snapshot $\mathbf{Q}$ we can find a unique POD – CAE decomposition in the form of Equation (3.7) , therefore we can rewrite Equation (2.2) as follows

$$\begin{aligned} \frac{d}{dt} q_h(\Gamma, t) + \mathcal{L}_d \left[q_h(\Gamma, t) \right] + \mathcal{N}_d \left[q_h(\Gamma, t) \right] = 0, \\ \frac{d}{dt} \left(\hat{\mathbf{\Phi}} a_{\hat{\Phi}}(t) + \mathbf{\Psi} \left[\mathcal{D}(\chi(t)) \right] \right) + \mathcal{L}_d \left[\hat{\mathbf{\Phi}} a_{\hat{\Phi}}(t) + \mathbf{\Psi} \left[\mathcal{D}(\chi(t)) \right] \right] \\ + \mathcal{N}_d \left[\hat{\mathbf{\Phi}} a_{\hat{\Phi}}(t) + \mathbf{\Psi} \left[\mathcal{D}(\chi(t)) \right] \right] = 0. \end{aligned} \tag{3.8}$$

Now we can project Equation (3.8) in the subspace spanned by $\hat{\mathbf{\Phi}}$ and $\mathbf{\Psi}$ to perform the integration.

$$\begin{aligned}\hat{\mathbf{\Phi}}^T\Bigg[\frac{d}{dt}\left(\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right)+\mathcal{L}_d\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right]\\ +\mathcal{N}_d\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right]\Bigg]=0,\\ \mathbf{\Psi}^T\Bigg[\frac{d}{dt}\left(\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right)+\mathcal{L}_d\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right]\\ +\mathcal{N}_d\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right]\Bigg]=0.\end{aligned} \tag{3.9}$$

Using Equation (3.6) we can reduce the above equation in the following form

$$\frac{d}{dt}\left(a_{\hat{\Phi}}(t)\right)+\mathcal{L}_{\hat{\Phi}}\left[a_{\hat{\Phi}}(t)\right]+\mathcal{N}_{\hat{\Phi}}\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right]=0, \tag{3.10a}$$

$$\frac{d}{dt}\left(\mathcal{D}(\chi(t))\right)+\mathcal{L}_{\Psi}\left[\mathcal{D}(\chi(t))\right]+\mathcal{N}_{\Psi}\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right]=0, \tag{3.10b}$$

where:

$$\begin{aligned}\mathcal{L}_{\hat{\Phi}}&=\hat{\mathbf{\Phi}}^T\mathcal{L}_d\hat{\mathbf{\Phi}},\quad \mathcal{L}_{\Psi}=\mathbf{\Psi}^T\mathcal{L}_d\mathbf{\Psi},\\ \mathcal{N}_{\hat{\Phi}}&=\hat{\mathbf{\Phi}}^T\mathcal{N}_d,\quad \mathcal{N}_{\Psi}=\mathbf{\Psi}^T\mathcal{N}_d.\end{aligned} \tag{3.11}$$

We can observe that Equation (3.10a) takes the same form as shown in POD - Galerkin Projection with additional term $\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]$ in the non-linear operator. To solve this equation we use POD-GP with DEIM and reformulate the non-linear term, for d DEIM modes, as follows

$$\mathcal{N}_{\hat{\Phi}}=\hat{\mathbf{\Phi}}^T\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}\underbrace{\mathcal{N}_d\left[\mathbf{P}^T\cdot\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{P}^T\cdot\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right]}_{\mathbb{R}^d}, \tag{3.12}$$

where:

- $\mathbf{W}\in\mathbb{R}^{k\times d}$: Left hand singular vector of the snapshot matrix constructed from evaluation of non-linear term alone,
- $\mathbf{P}\in\mathbb{R}^{N\times m}$: Selection matrix that selects specific location for evaluation of non-linear term. It is obtained using Algorithm 1 ,
- $\mathbf{K}=\left(\hat{\mathbf{\Phi}}^T\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}\right)\in\mathbb{R}^{m\times d}$: Non-linearity coefficient matrix.

While Equation (3.10a) lies in $k \ll N$ dimensional subspace, the Equation (3.10b) lies in $N - k$ dimensional subspace, which is very close to the original space. Further, we are only interested in the evolution of $\chi(t) \in \mathbb{R}^m$ since that would allow us to perform the integration of Equation (3.10a) and obtain the approximate solution using Equation (3.7) . Therefore instead of directly integrating the Equation (3.10b) we utilize the LSTM network for time integration of $\chi(t)$.

Upon inspection of Equation (3.10b) , we can observe, for an accurate estimation of χ at any time t the LSTM network not only requires the previous states of χ but also $a_{\hat{\Phi}}$. Table 3.1 lists some of the possible methods and the associated issues for feeding LSTM network with $a_{\hat{\Phi}}$.

Table 3.1: Input method for $a_{\hat{\Phi}}$ to the surrogate LSTM model for Equation (3.10b) and the associated issues

Input Method	**Issues**
Full space Projection of $a_{\hat{\Phi}}$	Very high computational cost and extremely high number of LSTM parameters.
$a_{\hat{\Phi}}$ in $\hat{\mathbf{\Phi}}$ space	The LSTM network has to implicitly learn $k \times m$ additional parameters, making the training process difficult.

In addition to the issues mentioned in the Table 3.1 , another prominent issues is the extrapolation in the encoding space. When solving the Equation (3.10b) using LSTM model close to the boundary of the encoding space, any numerical error in Equation (3.10a) can change the problem from an interpolation task to an extrapolation task, thereby resulting in large errors [4]. Therefore, we decide not to input $a_{\hat{\Phi}}$ in to the LSTM model, and this could be one of the challenges to be tackled in the future work. Hence the final equation modelled by the LSTM network is

$$\frac{d}{dt}\left(\mathcal{D}(\chi(t))\right) + \mathcal{L}_{\Psi}\left[\mathcal{D}(\chi(t))\right] + \mathcal{N}_{\Psi}\left[\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right] = 0. \tag{3.13}$$

With this formulation the LSTM network remains oblivious to the evolution of the POD modes and hence the training and the prediction of χ can be performed independently for any given initial condition and POD – CAE projection. A major drawback of this unidirectional coupling can be observed in the case where the overall dynamics is majorly governed by the non-linear term. The LSTM being uninformed about the evolution of $a_{\hat{\Phi}}$ generates a unique value of χ at any time t and if the Equation (3.10a) makes an error in any integration step it continues to aggregate in all the following step. A bidirectional coupling could solve this by adjusting the χ based on the prediction of $a_{\hat{\Phi}}$ and vice versa. Another interesting implication of this bidirectional coupling is highlighted in the next section.

Algorithm 2 defines the final algorithm for development of surrogate model using intrusive method.

Algorithm 2: POD – CAE Intrusive ROM

Input : Solution snapshot matrix Q, Non-linear term snapshot matrix Q_{NL}, number of linear modes k, number of non-linear modes m, number of DEIM modes d, LSTM time window w, PDE

Output: Surrogate Model Linear Part M_L, and Non-Linear Part M_{NL}

POD – CAE:

Perform full POD on Q to obtain $\mathbf{\Phi} = \{\phi_i | i = 1, 2, \cdots, N\}$ and a

set $\hat{\mathbf{\Phi}} \leftarrow \{\phi_i | i = 1, \cdots, k\}$, $a_{\hat{\Phi}} \leftarrow \{a_i | i = 1, \cdots, k\}$, $\mathbf{\Psi} \leftarrow \{\phi_i | i = k+1, \cdots, N\}$, $a_\Psi \leftarrow \{a_i | i = k+1, \cdots, N\}$

Train CAE on a_Ψ with m encoding dimensions to obtain the encoder,$\mathcal{E}$, and decoder, $\mathcal{D}$

Project snapshot a_Ψ to obtain χ

POD – DEIM + LSTM:

Project the PDE using $\hat{\mathbf{\Phi}}$ in the form of Equation (3.10a)

Perform POD on Q_{NL} to obtain $\mathbf{W}$ with d modes

Use Algorithm 1 to obtain selection matrix $\mathbf{P}$

Define non-linear operator for the POD using Equation (3.12)

Train LSTM on χ

$M_{NL} \leftarrow$ {Projectors: $(\mathcal{E}, \mathcal{D})$, Integrator: Trained LSTM}

$M_L \leftarrow \left\{\text{Projectors: } (\hat{\mathbf{\Phi}}^T, \hat{\mathbf{\Phi}}), \text{ Integrator: User's Choice, PDE: Algorithm 5}\right\}$

3.1.2 Non–Intrusive: Proper Orthogonal Decomposition with Operator Inference

A non-intrusive MOR approach provides two possibilities for problem formulation. The first constructs the operators in the FOM space by solving Equation (2.19) with complete snapshot matrix. The second approach constructs the operators in a lower dimensional space by projecting the snapshots in the $\hat{\mathbf{\Phi}}$, and therefore omitting the necessity to explicitly project the operators. We describe both the approaches in the following two paragraphs

Full Space Operator Inference

The fundamental idea behind full scale operator inference is very similar to intrusive modelling. Here we assume a PDE of form Equation (2.18) and then infer the complete operators by solving Equation (2.19) . It should be noted that for the experiments discussed

later, we assume that the snapshots are generated through numerical simulations or physical experiments and therefore have some numerical or experimental error in them. This assumption justifies the solving of regularized minimization problem for inference of operators. If this is not the case and the snapshots is generated from analytic solution or from a very accurate simulation then the regularization term should be dropped in Equation (2.19) .

Once the operators are obtained we can use the Algorithm 2 for to obtain the reduced POD-CAE model.

Reduced Space Operator Inference

To formulate the reduced space OI, we follow a similar approach as that of the intrusive method. We consider the Equation (2.18) and insert the POD – CAE projection to obtain

$$\begin{aligned}\frac{d}{dt}\left(\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right) = c &+ \mathbf{A}\left(\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right)\\ &+ \mathbf{H}\left(\left(\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right)\otimes\left(\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right)\right)\\ &+ \mathbf{B}u(t).\end{aligned} \tag{3.14}$$

Let us define the leading square term as $\mathcal{N}$, then we can express the above equation as follows

$$\begin{aligned}\frac{d}{dt}\left(\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right) = c &+ \mathbf{A}\left(\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right)\\ &+ \mathbf{H}\cdot\mathcal{N}\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right]\\ &+ \mathbf{B}u(t),\end{aligned} \tag{3.15}$$

from this point we can follow the steps described in Section 3.1.1 to obtain the reduced equation in the form

$$\frac{d}{dt}\left(a_{\hat{\Phi}}(t)\right) = c_{\hat{\Phi}} + \mathbf{A}_{\hat{\Phi}}\left[a_{\hat{\Phi}}(t)\right] + \mathbf{H}_{\hat{\Phi}}\cdot\mathcal{N}\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right] + \mathbf{B}_{\hat{\Phi}}u(t), \tag{3.16a}$$

$$\frac{d}{dt}\left(\mathcal{D}(\chi(t))\right) = c_{\Psi} + \mathbf{A}_{\Psi}\left[a_{\hat{\Phi}}(t)\right] + \mathbf{H}_{\hat{\Psi}}\cdot\mathcal{N}\left[\hat{\mathbf{\Phi}}a_{\hat{\Phi}}(t)+\mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right]\right] + \mathbf{B}_{\Psi}u(t), \tag{3.16b}$$

where:

$$\begin{aligned}\mathcal{A}_{\hat{\Phi}} &= \hat{\mathbf{\Phi}}^T\mathcal{A}\hat{\mathbf{\Phi}}, \quad \mathcal{A}_{\Psi} = \mathbf{\Psi}^T\mathcal{A}\mathbf{\Psi},\\ c_{\hat{\Phi}} &= \hat{\mathbf{\Phi}}^T c, \quad c_{\Psi} = \mathbf{\Psi}^T c,\\ \mathbf{H}_{\hat{\Phi}} &= \hat{\mathbf{\Phi}}^T\mathbf{H}, \quad \mathbf{H}_{\Psi} = \mathbf{\Psi}^T\mathbf{H},\\ \mathbf{B}_{\hat{\Phi}} &= \hat{\mathbf{\Phi}}^T\mathbf{B}, \quad \mathbf{B}_{\Psi} = \mathbf{\Psi}^T\mathbf{B}.\end{aligned} \tag{3.17}$$

Following the similar argument described in Section 3.1.1 , the Equation (3.16a) lies in space the spanned by $\hat{\mathbf{\Phi}} \in \mathbb{R}^k$ and the Equation (3.16b) lies in space the spanned by $\mathbf{\Psi} \in \mathbb{R}^{N-k}$, which is high dimensional. Therefore, we again attempt to solve the Equation (3.16a) directly using any integration method of choice, and use an LSTM model, trained on snapshot encoding, to obtain χ at any time t. We will again consider a unidirectional coupling based on the arguments provided in the Section 3.1.1 . In order to solve the Equation (3.16a) we infer the projected operators $c_{\hat{\Phi}}$, $\mathcal{A}_{\hat{\Phi}}$, $\mathbf{H}_{\hat{\Phi}}$, and $\mathbf{B}_{\hat{\Phi}}$, by putting the projected snapshot matrix $a_{\hat{\Phi}}$ in the Equation (2.19) .

It is important to consider that the OI solves a minimum least square problem to obtain the operators and with the reduced space OI, we need to make sure that enough POD modes are used such that the dominant dynamics is covered by Equation (3.16a) . Usually this means considering a relatively larger number of POD modes as compared to the Full Space Operator Inference. Further, for an accurate prediction of operators almost all the modes contributing to the dynamics of the system should be taken into the linear part. This consideration somewhat undermines the fundamental idea behind using the POD – CAE decomposition, since the contribution of S_2 in the Equation (3.2) becomes negligibly small, and the overall dynamics is governed by the S_1. It can be shown when S_1 contains almost all the dominant POD modes the S_2 captures only very high frequency and low amplitude POD modes, these make the learning task complex for the CAE projector and the LSTM integrator. Further, the S_1 doesn't capture enough dynamics, the constructed operators are not accurate, and Equation (3.16a) dominates the error.

One possibility to remedy the situation of necessity to select large number of linear modes and negligible contribution of S_2 would be by considering a bidirectional coupling along with joint training of LSTM and OI. This could be considered as a possible direction for the future work. A bidirectional coupling allows the LSTM model to be aware of the states achieved by $a_{\hat{\Phi}}$, allowing it to produce the outputs accordingly. With this in hand, a joint training of OI and LSTM allows the OI operators to be influenced from the parameters of the LSTM model through χ in the non-linear term, $\mathcal{N}$, and vice versa. Therefore, the joint training can be viewed as a feedback mechanism that allows the operators to adjust based on the dynamics being covered by the LSTM and similarly the LSTM adjusts the parameters based on the operators predicted from the OI. Hence, this could lead to relaxed constraints on the choice of number of linear modes, and allow more dynamics to be encoded in the non-linear modes.

Algorithm 3 describes the final algorithm for development of surrogate model using POD – CAE projection and Reduced Space Operator Inference.

3.2 General Remarks

With the intrusive and non-intrusive formulation we can start experimenting on some examples. Here we list out some of the general considerations made throughout all the experiments

Algorithm 3: POD – CAE Non-Intrusive ROM

Input : Solution snapshot matrix Q, number of linear modes k, number of non-linear modes m, OI regularization factor λ, and LSTM time window w

Output: Surrogate Model Linear Part M_L, and Non-Linear Part M_{NL}

POD – CAE:

Perform full POD on Q to obtain $\mathbf{\Phi} = \{\phi_i | i = 1, 2, \cdots, N\}$ and a

set $\hat{\mathbf{\Phi}} \leftarrow \{\phi_i | i = 1, \cdots, k\}$, $a_{\hat{\Phi}} \leftarrow \{a_i | i = 1, \cdots, k\}$, $\mathbf{\Psi} \leftarrow \{\phi_i | i = k+1, \cdots, N\}$, $a_\Psi \leftarrow \{a_i | i = k+1, \cdots, N\}$

Train CAE on a_Ψ with m encoding dimensions to obtain the encoder,$\mathcal{E}$, and decoder, $\mathcal{D}$

Project snapshot a_Ψ to obtain χ

OI – LSTM:

Compute the Operators of Equation (3.16a) using projected snapshots $a_{\hat{\Phi}}$ and Equation (2.19)

Train LSTM on χ

$M_{NL} \leftarrow$ {Projectors: $(\mathcal{E}, \mathcal{D})$, Integrator: Trained LSTM} ;

$M_L \leftarrow \left\{\text{Projectors: } (\hat{\mathbf{\Phi}}^T, \hat{\mathbf{\Phi}}), \text{Integrator: User's Choice, PDE: Algorithm 5}\right\}$;

- About 95 – 98 % system energy is modeled by linear modes, $\hat{\mathbf{\Phi}}$, and the rest by $\mathbf{\Psi}$. For the cases, considered this implies considering 2 – 12 linear modes depending on the problem at hand and system size.
- The number of non-linear modes are chosen close to the dimension of the parameter space $\mathcal{G}$. For $\mathcal{G} \subset \mathbb{R}^{n_p}$, we iteratively increase the number of non-linear modes capping at $4 \times n_p$ and starting with n_p.
- The dimension of the convolution layer is decided based on the dimensions of the considered problem. For the non-intrusive cases where no-information on discretization or the dimensionality of the problem space is available a simple 1D convolution is performed.
- The convolution filter sizes are restricted to $\{3, 5, 7\}$ in order to keep the number of parameters low.
- The maxpooling filter size is restricted to $\{2, 3, 4\}$ in order to avoid too much data loss.
- The window size and the number of LSTM blocks are tuned based on the problem.
- To evaluate the quality of the surrogate model we examine the L_2 and L_∞ error over the time domain, and point-wise relative and absolute difference over the space

domain.

4 Experiments

In this section we apply the proposed POD – CAE method to the following two problems. We split them into cases, first belonging to intrusive method, and the second being the non-intrusive with full space OI. We follow the conventions described in Section 3.2 .

Before diving into the experiments concerning POD – CAE, we describe a study recommended by Dr. Romit Maulik[1] to justify the choice of separated training of CAE and LSTM model in [23] in contrast to the previous works where a combined training is performed.

4.1 CAE – LSTM Simultaneous vs Separated Training

In this experiment we consider two CAE – LSTM surrogate model for solving 1D Burgers' equation, differing only in the training process. The aim of the study is to examine the encoding space representation and the ease of explainability of two CAE – LSTM model differing only in training process, namely simultaneous and separated training. Here we show that a separated training provides a single representation of the data in the intrinsic coordinates, further we also show that for any initial condition u_0 and parameter Re lying in the domain of training set, not same as training data but any combination $Re \in \{100, 200, \cdots, 1900\}$ and $t \in [0, t_{end})$ such that we have an interpolation problem.

4.1.1 Problem Setup

For this experiment we use the Equation (2.28) and the parameters defined in Table 2.1 . We use Equation (2.29) to generate the snapshot matrix. Figure 4.1 shows the parameters used for construction of complete snapshot matrix.

We perform the combined training and the separated training keeping the parameters of the LSTM and CAE constant. Figure 4.2 shows the architecture of the constructed CAE and the Figure 4.3 shows the architecture used for LSTM model. In both the settings we examine the intrinsic coordinates spanned by the training snapshots before and after application of LSTM.

4.1.2 Results

For the above described problem setup we first examine the snapshots in the full space. Figure 4.4 shows some of the snapshots for different values of Re and t used for training

[1] This study was pointed out by Dr. Romit Maulik in personal correspondence about the topic

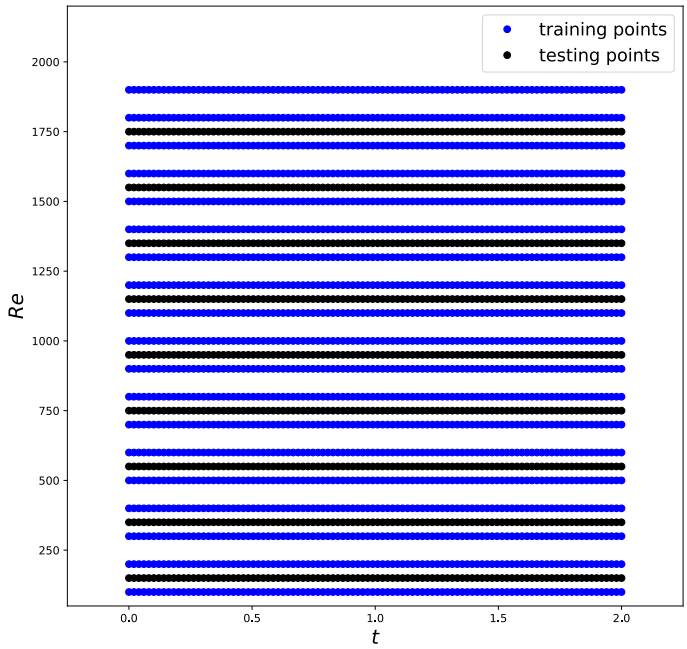


Figure 4.1: The points in the plot show the sampled value of t and Re from the parameter space to be used for construction of the complete snapshot matrix as well as for training and testing of the CAE – LSTM model. The points are equispaced along the time axis such that the LSTM model doesn't have to take into account another parameter, dt.

the CAE – LSTM models. We can observe that the snapshot matrix consists a mixture of smooth and sharp profiles spread across large parameter space. Using these snapshots we construct the encoding space for two cases.

Figure 4.5 shows the encoding space representation for the combined training and the separated training. We can see that for the combined training that the LSTM progressed points lie in disjoint region while in the case of the separated training the LSTM progressed points lie in the constructed encoding space by the encoder. It should be noted that we expect the LSTM progressed points to lie in the same encoding space, since the LSTM progressed point correspond to a training point of the encoder therefore the output of the LSTM has a unique representation in the encoded space formulated by the encoder. A disjoint region of the LSTM progressed point suggest that the LSTM maps the point from one manifold to another both of which have a overlapping region in the higher dimensional space (since the encoded coordinates for same point are different). Further, in case of combined training, the disjoint region suggests that the decoder and the encoder map to different manifolds.

Figure 4.6a highlights the two representation of the same point, the red point is the rep-

(a) Encoder Architecture

(b) Decoder Architecture

Figure 4.2: The figure shows the architecture of the constructed encoder and the decoder part of the CAE. This architecture is used for both combined and separated training. The input block to the network is at the bottom and the output block is at the top. The figure is generated using tensorboard toolkit available in the tensorflow library [3].

resentation corresponding to the manifold mapped to which the encoder maps the points and the black point is the representation corresponding to the manifold from which the decoder maps to the full space. For a given series of the encoded points, the LSTM network acts as a mapping from encoder's manifold to decoder's manifold. This is mainly because in a coupled training we do not have a control over the state of decoder or the LSTM network, sandwiched between the encoder and decoder, and the optimizer adjusts

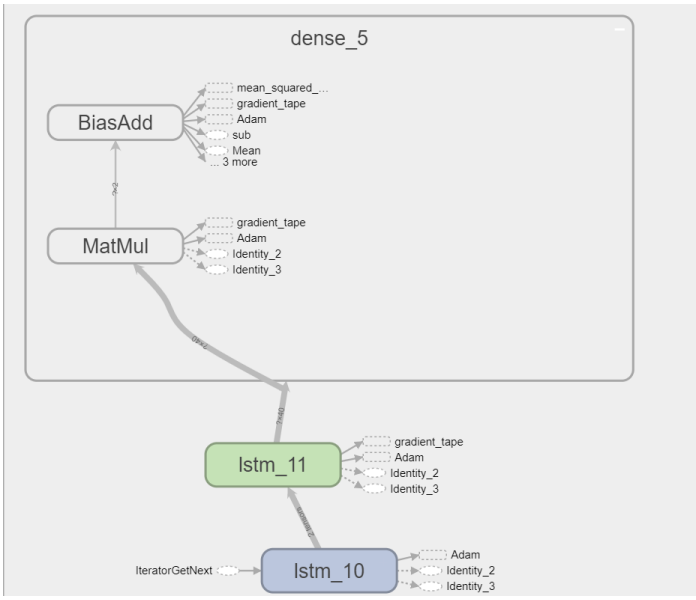
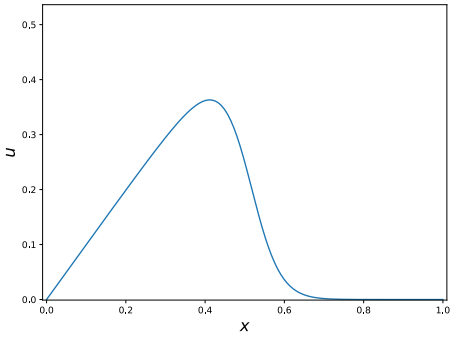


Figure 4.3: The figure shows the architecture used for the LSTM time integrator. Two consecutive LSTM blocks are used for the experiment followed by a dense layer. The architecture of each LSTM block is similar to that described in Section 2.5 . The first LSTM block has output dimension of 20, which is encoding space dimension times the window size, and the second LSTM block has an output dimension same as the encoding space.

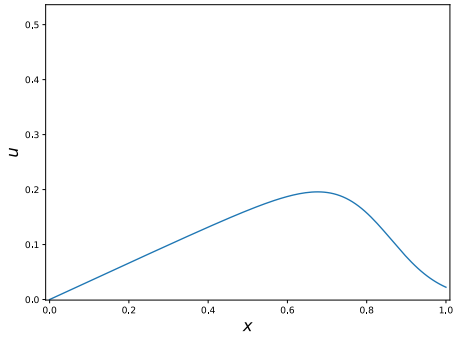
the weights in a way that the decoder can best reconstruct the output. To achieve this the LSTM network maps the encoded points to a manifold where they are well spaced, making the learning task for the decoder easier. In contrast to this, the separated training ensures that the encoder and decoder map to and from the same manifold, respectively. Doing so enforces the LSTM to map the output point, for a given series of encoded input, to the same manifold. Figure 4.6b shows the embedding space obtained from separated training of the CAE – LSTM network, and the corresponding expected and obtained output from LSTM. We can see that the mapping lies in the same manifold as expected.

A separated training is particularly interesting for this thesis work since it opens the possibility of explicit analysis of the projection operators and the time integrator. Further, it aids in the explainability of the model since an explicit construction of the embedding space allow to perform causal analysis of the error. For example when integrating a projected model using LSTM network we can examine the trajectory, if the path includes any point at the encoding space boundaries or outside the encoding space then we can conclude that the integration task performed by the LSTM includes extrapolation which can be source of error.

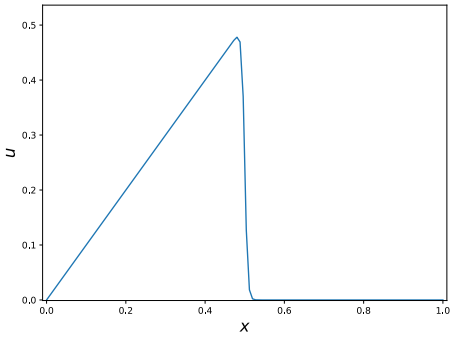
Following the above mentioned arguments we always perform a separated training for



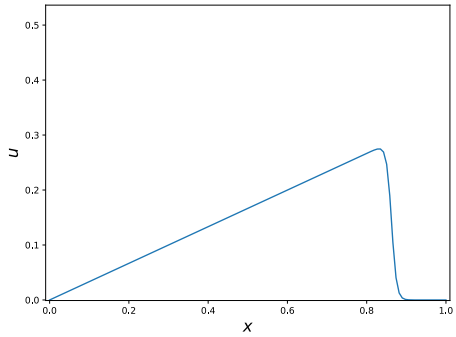
(a) $Re = 100$ and $t = 0$



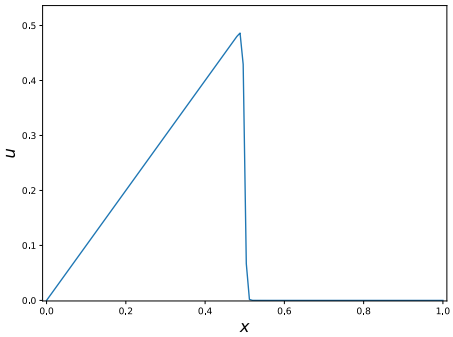
(b) $Re = 100$ and $t = 1.0$



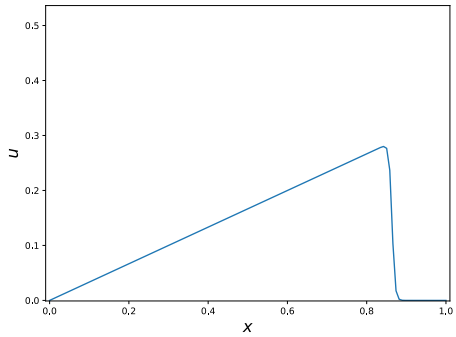
(c) $Re = 1100$ and $t = 0$



(d) $Re = 1100$ and $t = 1.0$



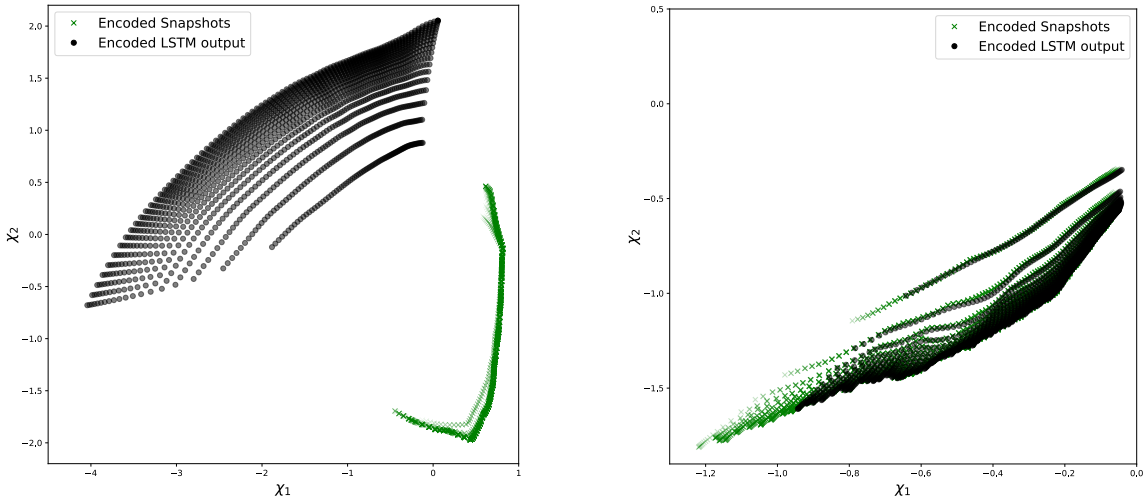
(e) $Re = 1900$ and $t = 0$



(f) $Re = 1900$ and $t = 1.0$

Figure 4.4: The figure shows different values of the profile of u for different choices of Re and t. We can observe that the data almost discontinuous behavior for higher Re values and small t values. For increasing t values and reduced Reynolds number the profile turns smooth as shown in Figure 4.4b . The snapshot matrix constructed with these data is used for training the CAE – LSTM model.

the proposed POD – CAE projection model.



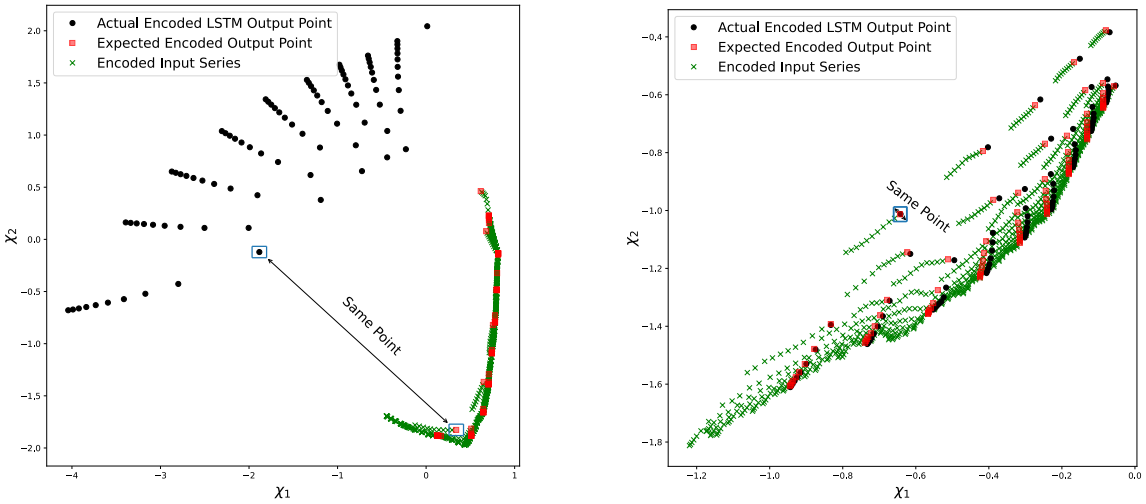
(a) Combined Training

(b) Separated Training

Figure 4.5: Encoding space representation of the snapshot matrix when CAE – LSTM model is trained in combined and separated fashion. We can observe that the LSTM progressed points in the combined training lie in a separate manifold in contrast to the separated training where the LSTM progressed points lie in the same manifold.

4.2 1D Burgers' Equation

In this experiment we implement the proposed POD – CAE projection method using intrusive ROM methodology as described in Section 3.1.1 . For this experiment we again consider 1D Burgers' equation as described in Equation (2.28) , we will project this equation to obtain the surrogate model using the Algorithm 2 . We further analyse the accuracy of the formulated surrogate model and compare it with classical POD – GP with DEIM and the exact solution. We will also attempt to add explainability by analysing the reduced space modes. Additionally, we also compare the runtime for the POD – CAE surrogate model, POD – GP with DEIM and a full order Finite Element Model (FEM).



(a) Simultaneous Training of CAE – LSTM network

(b) Separated Training of CAE – LSTM network

Figure 4.6: The figure shows embedding space representation of snapshot matrix, generated from Equation (2.28) , and the integrated output from the LSTM for a simultaneously trained CAE – LSTM network and separately trained CAE – LSTM network. The green points are the encoded points from the snapshot matrix. The red points represent the expected output from the LSTM network for given time sequence of green points. The black points represent the actual outputs of the LSTM network.

4.2.1 Problem Setup

In this experiment we consider Equation (2.28) restated below

$$\begin{aligned} \dot{u} + u\frac{\partial u}{\partial x} &= \nu \frac{\partial^2 u}{\partial x^2} \\ u(x, Re, 0) = u_0 &= \frac{x}{1 + \sqrt{\frac{1}{t_0}} e^{Re\frac{x^2}{4}}}, \quad x \in [0, L], \quad t \in [0, t_{end}] \\ u(0, Re, t) &= 0 \\ u(L, Re, t) &= \frac{\frac{L}{t+1}}{1 + \sqrt{\frac{t+1}{t_0}} e^{Re\frac{L^2}{4t+4}}}, \quad t_0 = e^{\frac{Re}{8}} \quad \text{and} \quad Re = \frac{1}{\nu} \end{aligned} \tag{4.1}$$

The analytical solution to the equation is given by Equation (2.29) . In this experiment we will consider the following three scenarios

- **FEM Simulation**: We perform a full order FEM simulation for the solution of the 1D Burgers' equation. We compare the quality of the solution with the exact solution and

use the generated data to perform further studies. The details of the FEM simulation is provided in the Section 6 .

- **POD – GP with DEIM**: Next using the FEM snapshot data we perform a POD – GP with DEIM to solve the partial differential equation for $t \in [0, t_{end})$. We will compare the obtained results with the analytical solution. Further, we will extend this part to be used as the linear surrogate model in POD – CAE setup.
- **POD – CAE**: For the POD – CAE application we will again use the snapshot matrix developed from FEM simulation. Additionally we will also use the linear part of the model from the POD – GP with DEIM, and couple it with the non-linear solver developed in the Section 3.1.1 .

The parameters used for the three settings are described in the Table 4.1 .

Table 4.1: Parameters used for FEM simulation, POD – CAE surrogate model, and POD – GP with DEIM surrogate model.

Parameters	**Values**
General	
L	1
t_{end}	2
Re	$\{100, 200, \cdots, 1900\}$
FEM	
dx	$1/128$
dt	0.01
POD – GP with DEIM	
Number of POD modes	$\{2, 5, 10\}$
Number of DEIM modes	Number of POD modes $+ \{2, \cdots, 5\}$
POD – CAE	
Energy covered by linear modes	95 – 98 %
Number of non-linear modes m	4
Convolution type	1D convolutions
Pooling	Max pooling
Number of Convolutional layers in encoder	6 depending on m
Number of Convolutional layers in decoder	5 depending on m
Number of LSTM blocks	2
Size of LSTM output	m
LSTM temporal window size	10

4.2.2 Finite Element Method

As shown in Figure 4.4 the properties of the solution of Equation (4.1) is highly dependent on the choice of Re. For higher Re the non-linear term dominates the solution and the PDE would be *stiff*. This requires a smaller time stepping for time integration and a finer discretization to avoid discontinuity. To find a full order solution of the PDE we use the following form obtained from the formulation described in Section 6

$$\mathbf{M}\dot{u}(t) = -\frac{1}{2}\mathbf{D}u^2(t) - \nu\mathbf{B}u(t) + f(t) \tag{4.2}$$

where:

- **M**: mass matrix
- **B**: stiffness matrix
- **D**: Convection coefficient

Figure 4.7 shows the simulation results for the above equation we can observe that for higher Re the FEM solution shows error near the discontinuity. Further, Figure 4.8 highlights the relative error at each simulation step for different choices of Re. We can notice for higher Re some of the intermediate simulation steps have highest error, these are the simulation points where the newton iteration did not converge for the backward euler step. This is primarily because the PDE becomes increasingly stiff for higher values of Re, and therefore requires smaller time steps.

After solving the equation we collect the snapshots and use them for ROM development.

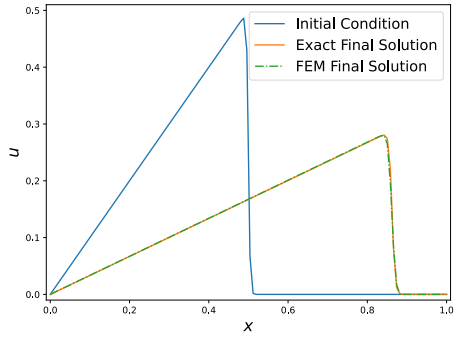
4.2.3 POD – GP with DEIM

The snapshots collected from the FEM simulations are now used to develop the POD – GP model with DEIM approximation of non-linearity. We begin with computation of POD basis for the model. We will examine the POD – GP with DEIM for multiple choices of POD modes and evaluate the quality of the surrogate model. For a snapshot generated from FEM model we can compute the POD modes using following [34]

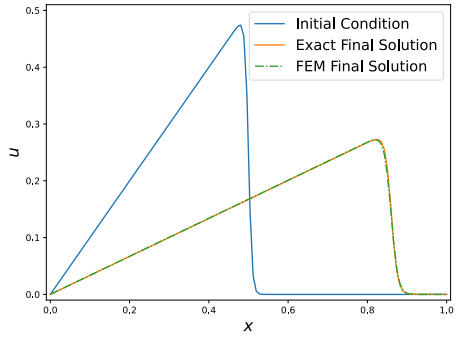
$$\mathbf{U}\mathbf{U}^T\mathbf{M}\phi = \sigma^2\phi \tag{4.3}$$

Since **M** is symmetric positive definite there exist a Cholesky decomposition of $\mathbf{M} = \mathbf{R}^T\mathbf{R}$ then we can rewrite the above equation as follows

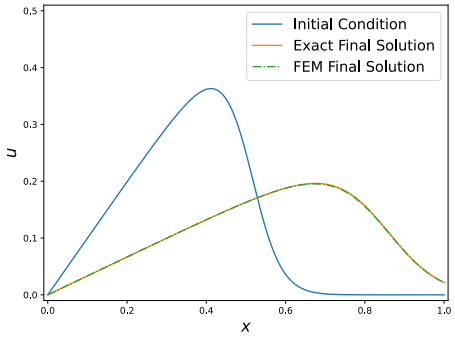
$$\begin{aligned} \mathbf{R}\mathbf{U}\mathbf{U}^T\mathbf{R}^T\mathbf{R}\phi &= \sigma^2\mathbf{R}\phi \\ \hat{\mathbf{U}}\hat{\mathbf{U}}^T\hat{\phi} &= \sigma^2\hat{\phi} \end{aligned} \tag{4.4}$$



(a) $Re = 1900$



(b) $Re = 900$



(c) $Re = 100$

Figure 4.7: The figure shows outputs for 3 FEM simulations for $Re = \{100, 900, 1900\}$. The blue line shows the initial condition used for the FEM simulation and the green line shows the obtained output at the end of simulation.

Now we perform SVD on $\hat{\mathbf{U}}$ to obtain the left hand singular vector, B, and select k modes. Using B and R we can obtain the POD basis:

$$\mathbf{\Phi} = \mathbf{R}^{-1}\mathbf{B}, \quad \text{where} \quad \mathbf{\Phi} = \{\phi_i | i = 1, \cdots, k\} \tag{4.5}$$

Using the obtained POD modes we can define the following ansatz for full-order approximation

$$u(t) \approx \mathbf{\Phi}\hat{u}(t), \quad \text{with} \quad \hat{u}(t) \in \mathbb{R}^k \tag{4.6}$$

Now we can apply POD Galerkin Projection with DEIM on the Equation (4.2) to obtain following reduced model

$$\frac{d}{dt}\hat{u}(t) = -\frac{1}{2}\mathbf{D}_{red}\left(\mathbf{C}\hat{u}\right)^2(t) - \nu\mathbf{B}_{red}\hat{u}(t) + f_{red}(t) \tag{4.7}$$

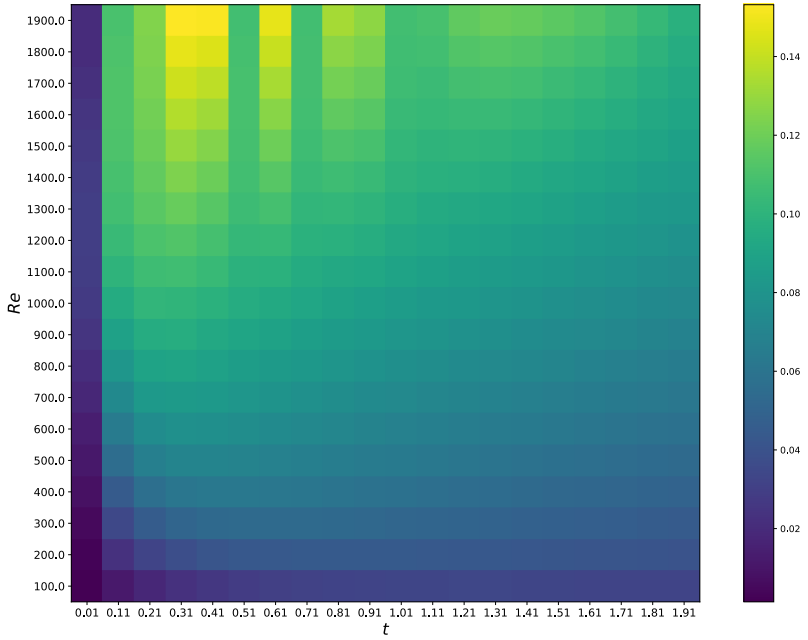


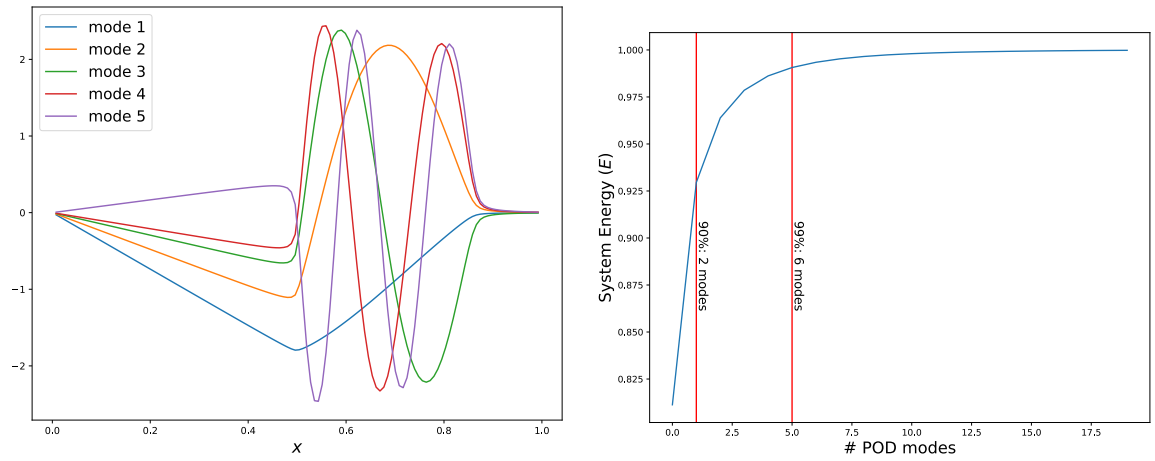
Figure 4.8: The figure shows the relative L_2 error for each integration point during the FEM simulation of Equation (4.2) . Each row represent all the states of one FEM simulation for a specific Re.

where

$$\begin{aligned} \mathbf{D}_{red} &= \mathbf{\Phi}^T \mathbf{D} \mathbf{W} \left(\mathbf{P}^T \mathbf{W} \right)^{-1}, \\ \mathbf{B}_{red} &= \mathbf{\Phi}^T \mathbf{B} \mathbf{\Phi}, \\ f_{red} &= \mathbf{\Phi}^T f, \\ \mathbf{C} &= \mathbf{P}^T \mathbf{\Phi}. \end{aligned} \tag{4.8}$$

The $\mathbf{W}$ is the left hand singular vector obtained from the snapshot matrix of non-linear term, $\mathbf{U}_2 = \left[u^2(t_1), u^2(t_2), \cdots, u^2(t_N) \right]$. Further, the $\mathbf{P}$ is the selection matrix obtained from the application of DEIM. Before applying the above formulation for different parameters listed in Table 4.1 , we investigate the POD modes of the snapshot matrix and the system energy they capture. Figure 4.9a shows the behaviour of the first 5 POD modes and the region of the dynamics. It also highlights the amount of system energy modelled by the addition of each POD modes. We can notice that a choice of 5 POD modes cover $\simeq 99\%$ of the system energy, and therefore modelling higher modes would contribute progressively less to the system dynamics. It should be noted that, for stiff PDE, although higher modes have diminishing contribution to the overall system energy but they have significant role in the system accuracy. For stiff PDEs usually a very high number of POD modes are necessary to accurately model the system. We show the results obtained for choice of

2, 5, 10 POD modes in Figure 4.10 .



(a) Behaviour of the first 5 POD modes across the domain.

(b) System energy covered by addition of each POD mode.

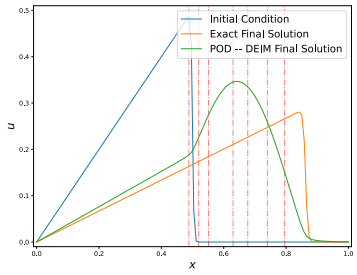
Figure 4.9: The figure shows the preliminary analysis on the POD modes. The behaviour of the modes across the domain highlights the region with maximum dynamics in the FEM snapshot matrix. Further, an inspection of the energy modelled by addition of each POD mode aids in deciding the sufficient number of modes to capture the major dynamics of the FEM snapshots.

Figure 4.10 suggests that addition of each POD mode reduces the low frequency error corresponding to each POD modes. Further the DEIM measurement points aid in making better prediction by adding more measurement points in the region of high dynamics. Figure 4.11 shows the regions of high error in the parameter space for different choice of POD modes. For 2 POD modes we can observe high error region around high Re and t because the projection is not able to capture the progression of non-linearity due to missing POD modes. This is effect is somewhat mitigated by introduction of additional POD modes and corresponding DEIM points. But we can observe that, for a stiff PDE, even the higher POD modes which do not contribute much to the system energy has significant impact on the accuracy of ROM.

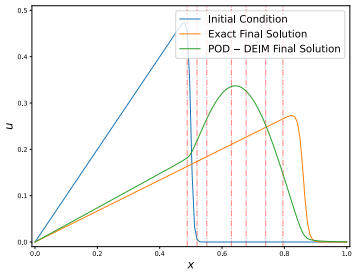
We now extend the shown POD – GP with DEIM framework to incorporate the POD – CAE type projection as described in Section 3.1.1 and develop the non-linear counterpart for coupling.

4.2.4 POD – CAE formulation

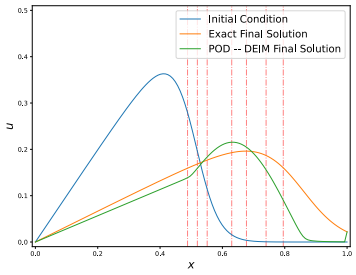
To formulate the POD – CAE projection of the model we have to express the Equation (4.2) in the Equation (3.10a) form. To achieve this let us reconsider Equation (4.4) , to obtain a



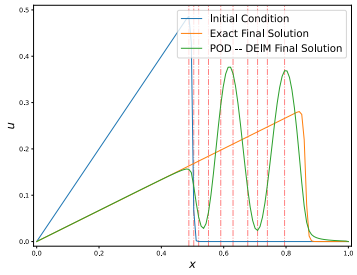
(a) $Re = 1900$ and 2 POD Modes 7 DEIM mode



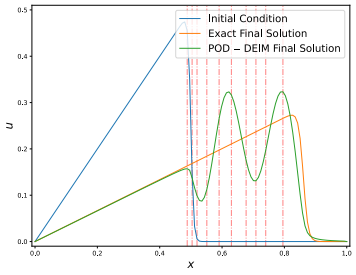
(b) $Re = 900$ and 2 POD Modes 7 DEIM mode



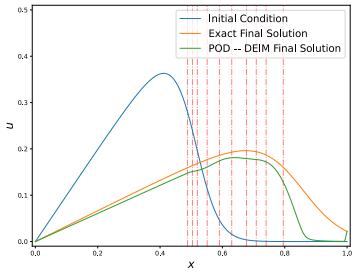
(c) $Re = 100$ and 2 POD Modes 7 DEIM mode



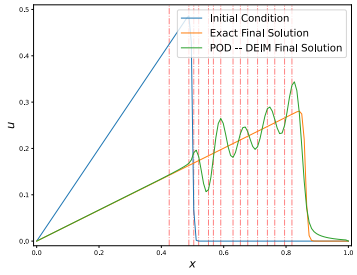
(d) $Re = 1900$ and 5 POD Modes 12 DEIM mode



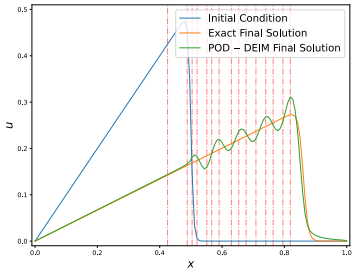
(e) $Re = 900$ and 5 POD Modes 12 DEIM mode



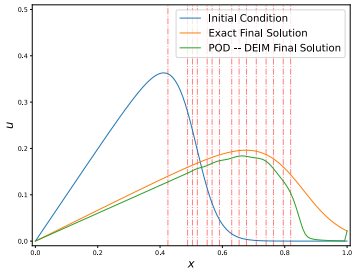
(f) $Re = 100$ and 5 POD Modes 12 DEIM mode



(g) $Re = 1900$ and 10 POD Modes 17 DEIM mode

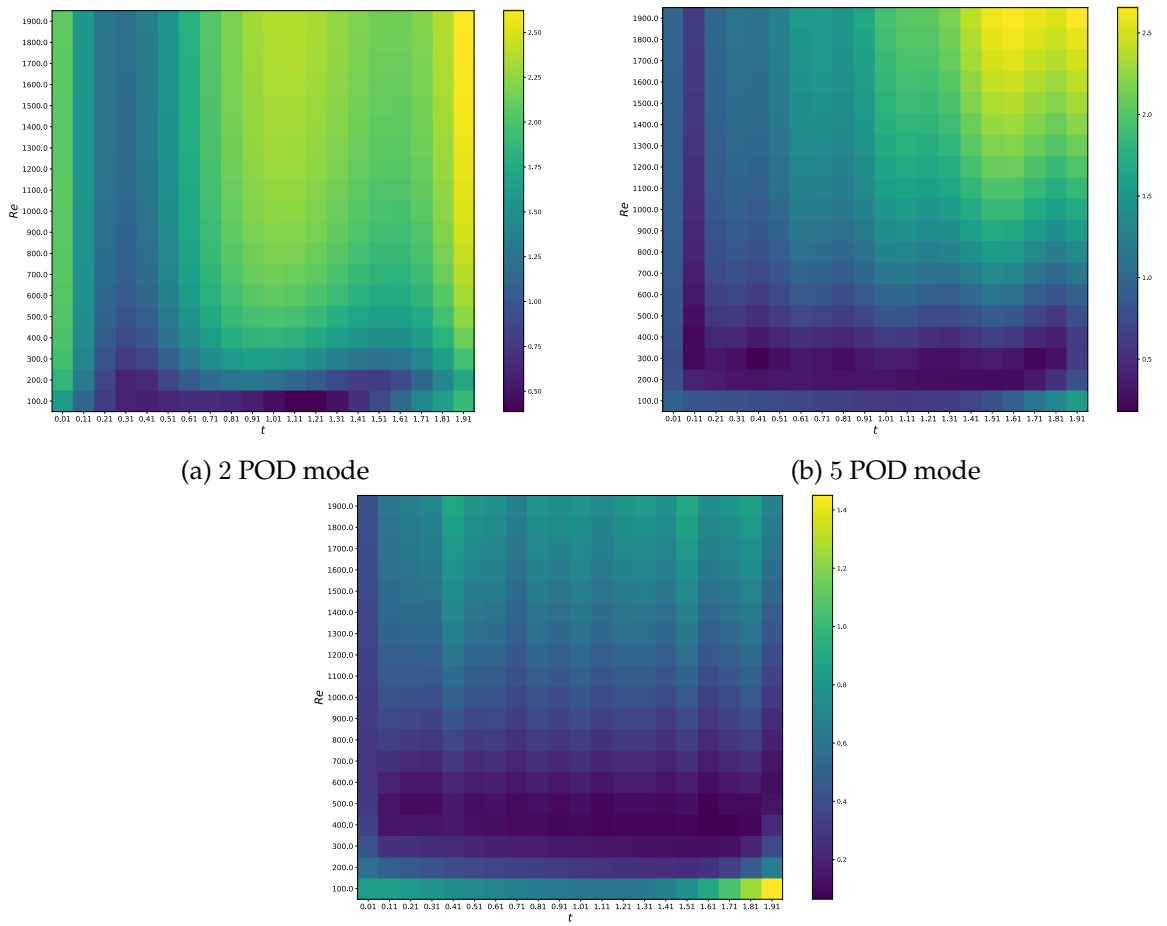


(h) $Re = 900$ and 10 POD Modes 17 DEIM mode



(i) $Re = 100$ and 10 POD Modes 17 DEIM mode

Figure 4.10: The figure shows the POD – GP with DEIM simulation output for Burgers' equation using FEM snapshot matrix. The red dashed lines represent the DEIM measurement locations, obtained from Algorithm 1 . We notice that the performance of the model is very poor for choice of 2 POD modes, although the DEIM measurements are performed close to the region of actual dynamics. Increasing the POD modes to 5 somewhat improves the performance of the model by removing the low frequency error arising from the missing modes, additionally DEIM modes increase the measurement points in the region of error aiding in the error reduction. A similar trend is observed for addition of next 5 POD modes.



(a) 2 POD mode

(b) 5 POD mode

(c) 10 POD mode

Figure 4.11: The figure shows the relative L_2 error for each integration point during POD – GP with DEIM simulation for the solution of Equation (4.2) using $2, 5, 10$ POD modes. We can notice that the overall relative error decreases with an increase of POD modes. For 2 POD modes we can notice the high error region to be concentrated around higher Re and t values. Which changes for 10 POD mode case where the high error region is concentrated around low Re and high t, this is because of the unavailability of DEIM measurement points toward the right boundary as shown in Figure 4.10i .

full POD, we compute the left hand singular vector, $\mathbf{B}$, of $\hat{\mathbf{U}}$ and compute the $\mathbf{\Phi}$

$$\mathbf{\Phi} = \mathbf{R}^{-1}\mathbf{B}, \quad \text{where} \quad \mathbf{\Phi} = \{\phi_i | i = 1, 2, \cdots, N\}. \tag{4.9}$$

We can now decompose the $\mathbf{\Phi}$ in $\hat{\mathbf{\Phi}} = \{\phi_i | i = 1, \cdots, k\} \in \mathbb{R}^{N\times k}$ and $\mathbf{\Psi} = \{\phi_i | i = k+1, \cdots, N\} \in \mathbb{R}^{N\times N-k}$. Using $\hat{\mathbf{\Phi}}$ and $\mathbf{\Psi}$ with Equation (3.7) we can define following ansatz for full order

approximation

$$u(t) \approx \hat{\mathbf{\Phi}}\hat{u}(t) + \mathbf{\Psi}\left[\mathcal{D}(\chi(t))\right], \quad \text{with} \quad \hat{u}(t) \in \mathbb{R}^k \text{ and } \chi \in \mathbb{R}^m, \tag{4.10}$$

where

- $\mathcal{D} : \mathbb{R}^m \to \mathbb{R}^{N-k}$ is the decoder of the CAE model
- $\mathcal{E} : \mathbb{R}^{N-k} \to \mathbb{R}^m$ is the encoder of the CAE model, which is used for encoding $u_\Psi = \mathbf{\Psi}^T\left(u(t) - \hat{\mathbf{\Phi}}\hat{u}(t)\right)$.

Using above formulations we can transform Equation (4.2) to

$$\frac{d}{dt}\hat{u}(t) = -\frac{1}{2}\mathbf{D}_{red}\left(\mathbf{C_1}\hat{u}(t) + \mathbf{C_2}\mathcal{D}(\chi(t))\right)^2 - \nu\mathbf{B}_{red}\hat{u}(t) + f_{red}(t), \tag{4.11}$$

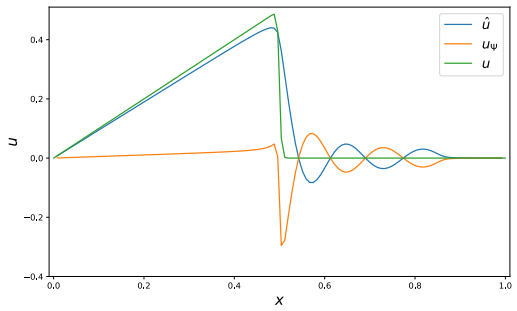
where

$$\begin{aligned}
\mathbf{D}_{red} &= \hat{\mathbf{\Phi}}^T\mathbf{D}\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}, \\
\mathbf{B}_{red} &= \hat{\mathbf{\Phi}}^T\mathbf{B}\hat{\mathbf{\Phi}}, \\
f_{red} &= \hat{\mathbf{\Phi}}^T f, \\
\mathbf{C_1} &= \mathbf{P}^T\hat{\mathbf{\Phi}}, \\
\mathbf{C_2} &= \mathbf{P}^T\mathbf{\Psi},
\end{aligned} \tag{4.12}$$

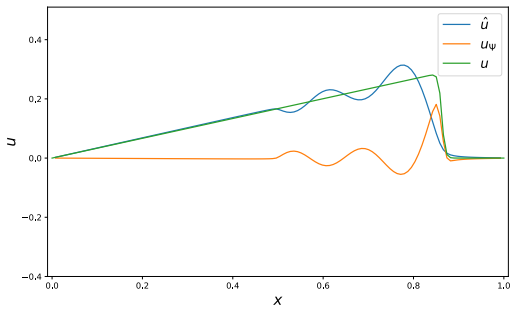
and $\mathbf{W}$ as well as $\mathbf{P}$ are the DEIM terms. W is the left hand singular vector of the non-linear term snapshot matrix, $\mathbf{U}_2 = \left[u^2(t_1), u^2(t_2), \cdots, u^2(t_N)\right]$ and $\mathbf{P}$ is the selection matrix obtained from the *Algorithm* 1.

As explained in the Section 3.1 , the non-linear projector is formulated using a Convolutional Autoencoders. Following the directions mentioned in Section 3.2 we decide k and m. For, this experiment we choose $k = 5$ which covers 98% system energy, and since the number of parameters in the Equation (4.2) are 2 so we take $m = 8$ for our experiment. Before constructing our non-linear projector we examine the $\hat{u}$ and u_Ψ. Figure 4.12 shows the decomposition of u in $\hat{u}$ and u_Ψ. As expected, the $\hat{u}$ corresponding to the first k POD modes forms a smooth approximation of u, while the u_Ψ captures the sharp changes in the profile since it lies in the subspace $\mathbf{\Psi}$ which consists of high frequency modes.

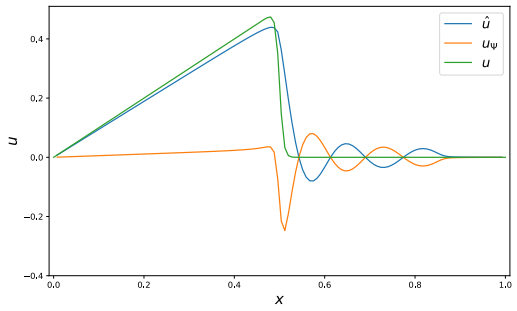
Now we construct the CAE based non-linear projector using the parameters defined in Table 4.1 . In our formulation, we perform a separated training of CAE and LSTM as mentioned in Section 4.1 , therefore we can ensure that the encoder, $\mathcal{E}$, and decoder, $\mathcal{D}$, map to and from same manifold, respectively. This also opens the possibility to perform a separate study of the quality of projection using CAE. Figure 4.13a shows the relative reconstruction error of the u_Ψ. We can notice that the relative error is high in the region with high Re and low t. Further, the Figure 4.13b shows the intrinsic coordinates of the obtained u_Ψ from the FEM simulation. The points are coloured by relative error, with the blue points being on the lower side of the relative error and the red points being on the higher side. We



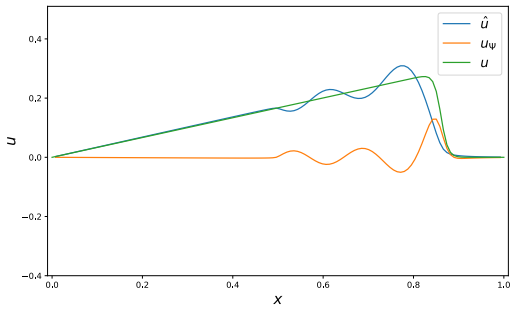
(a) $Re = 1900$, $t = 0$, 5 POD Modes



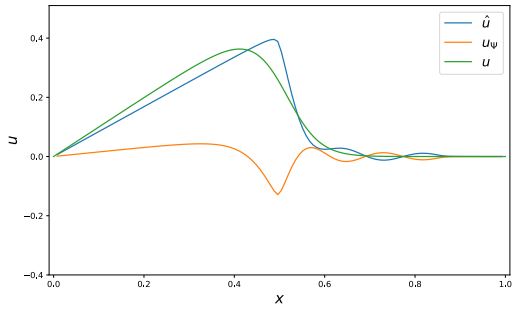
(b) $Re = 1900$, $t = 2.0$, 5 POD Modes



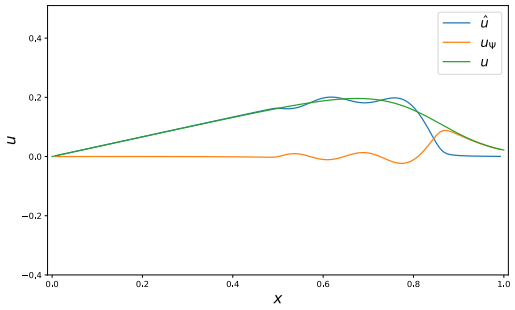
(c) $Re = 900$, $t = 0$, 5 POD Modes



(d) $Re = 900$, $t = 2.0$, 5 POD Modes



(e) $Re = 100$, $t = 0$, 5 POD Modes



(f) $Re = 100$, $t = 2.0$, 5 POD Modes

Figure 4.12: The figure shows the full POD decomposition of u in $\hat{u} = S_1$ and $u_\Psi = S_2$ form based on Equation (3.2) . We can observe that $\hat{u}$ always forms the smooth curve even in the cases where u have sharp change in value, due to small frequency of the first 5 POD modes. On the other hand the u_Ψ captures the sharp changes, due to the high frequency POD modes.

can notice that the red points are always at the boundary of the intrinsic domain spanned by the projected points. This behaviour is easily explainable since the interpolating function, CAE, does not have enough data at the boundaries to make an accurate prediction. This also suggests that u_ψ for high Re and low t values lie on the domain boundary of the

intrinsic coordinates. Therefore to improve the quality of projection of u_ψ for high Re and low t values we should sample more such points in the FEM snapshot matrix. A similar argument holds for the extended branches in the domain. Since we restrict the study to the choice of FEM parameters mentioned in Table 4.1 , we will continue with this CAE projector for further analysis.

Similar to the projector we can perform an analysis on the intergrator, LSTM, separately. We examine the relative reconstruction error for all the FEM snapshots and the intrinsic coordinates in the encoding space for a integrated example of u_Φ. Figure 4.14a highlights the region of high error, we observe a high error for low Re and high t values which correspond to the domain branch. Further, the integration path, in Figure 4.14b , for the $Re = 200$ suggests that the LSTM could have attempted prediction outside the domain, leading to additional error arising from extrapolation.

With the linear part formulated in Equation (4.11) , along with the non-linear projector and integrator we can use the POD – CAE intrusive surrogate model. For the integration of Equation (4.11) we use Backward Euler iteration with Newton fixed point iteration [7].

4.2.5 Results

In this part we investigate the solution of the POD – CAE projection based intrusive ROM, compare the accuracy and performance of the formulated surrogate model as well as postulate the possible source of errors.

Figure 4.15 shows the computed final state for 3 Re values, we observe that the POD – CAE intrusive model is able to predict the overall trend. Further, the overall oscillatory behaviour of the solution is reduced significantly as compared to POD – DEIM. Additionally, we also observe that the POD – CAE surrogate model reduces the L_2 error by an order of magnitude at a minuscule cost of performance as highlighted in Table 4.3 and Table 4.2 . Figure 4.16 shows the relative L_2 norm error for all choices of Re, we notice that the relative error trend is similar to that of Figure 4.14a . The reason behind such trend is that for the stiff PDE, Equation (4.2) , the non-linear term governs the dynamics and in Equation (4.11) the non-linear term has complete access to the modes in $\mathbf{\Psi}$ through $\mathbf{C_2}$. Therefore, the possible sources of error in computation of non-linear term in the Equation (4.11) are the error accumulation through time integration scheme and the computation of $\mathbf{C_2}$. Hence the dominant source of error in the resulting solution is the contribution from the non-linear projection and integration through CAE and LSTM, respectively. A possibility to improve the POD – CAE intrusive surrogate model could through enforcing the formulation of convex domain of intrinsic coordinates. Such a formulation would minimize the error made by LSTM model since any prediction inside the convex domain would always be an interpolation step, and there would very limited cases that are close to the boundary. Another possibility to control the error would be through projection operator. It can be concluded from Figure 4.13a that the CAE projector puts a lower bound on the error due to imperfect reconstruction. Any improvement in the decoder can reduce the overall error of the CAE – POD intrusive surrogate model.

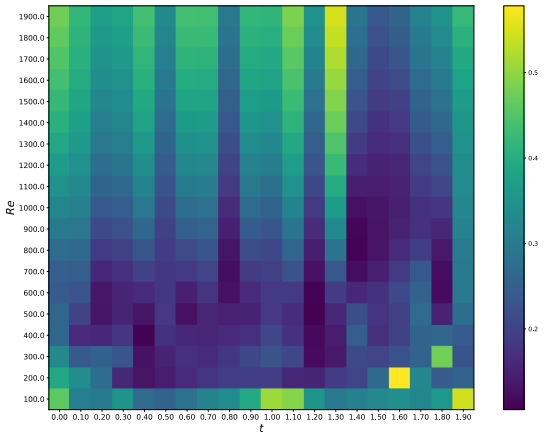
We can observe that for an order of magnitude loss in accuracy the proposed reduced order model provides a performance gain of about two orders of magnitude and solves the issues that come with the classical POD – GP with DEIM method.

Table 4.2: Runtime comparison for solution of Burgers' equation using the three methods. The average time is computed using 5 runs for each parameter value

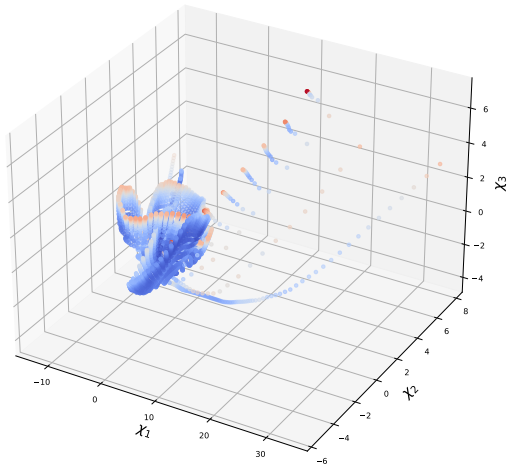
	FEM	**POD – GP with DEIM**	**Intrusive POD – CAE**
Worst case time (s)	14.310 ± 0.111	0.086 ± 0.009	0.148 ± 0.008
Average time (s)	9.932 ± 2.908	0.124 ± 0.015	0.133 ± 0.012
Speed-up w.r.t. FEM	1.0	79.4 ± 21.7	73.7 ± 18.8

Table 4.3: L_2 error norm comparison for solution of Burgers' equation using the three methods at final time step. The table outlines the results where the POD – CAE (5 POD modes, 4 CAE modes), FEM and POD – GP (10 POD modes) with DEIM are deployed anew for each Re. The L_2 norm is computed between the final time step result and the exact solution. The POD – CAE model error is lower for similar number of reduced dimensions.

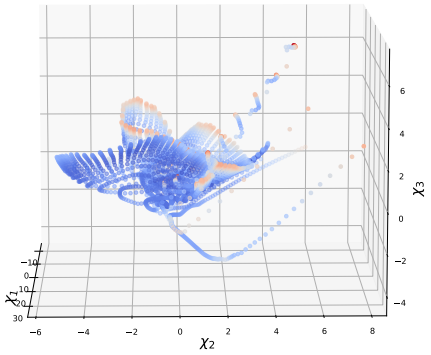
	$\mathbf{Re = 1750}$	$\mathbf{Re = 950}$	$\mathbf{Re = 150}$
Absolute L_2 error			
FEM	2.257×10^{-4}	1.445×10^{-4}	5.052×10^{-5}
POD – GP with DEIM	1.352×10^{-2}	1.337×10^{-2}	1.123×10^{-2}
Intrusive POD – CAE	1.160×10^{-3}	6.562×10^{-4}	1.561×10^{-3}
Relative L_2 error			
FEM	0.080%	0.053%	0.026%
POD – GP with DEIM	4.819%	4.903%	5.726%
Intrusive POD – CAE	0.413%	0.241%	0.795%



(a) Relative reconstruction error of u_Ψ

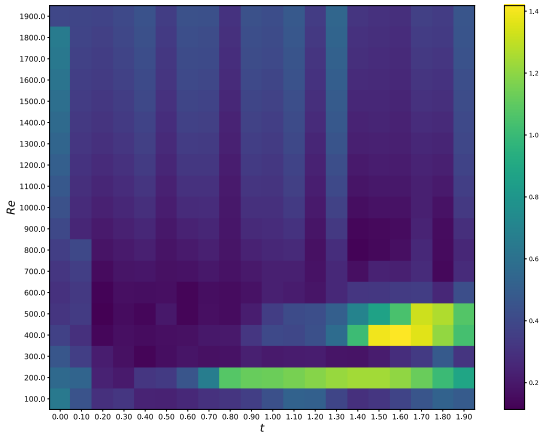


(b) Intrinsic Coordinates of the FEM snapshots: View 1

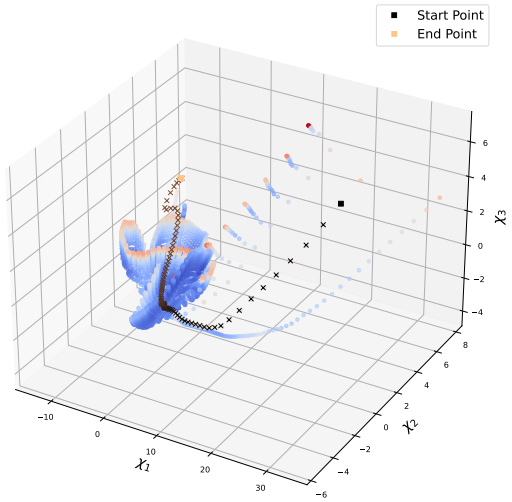


(c) Intrinsic Coordinates of the FEM snapshots: View 2

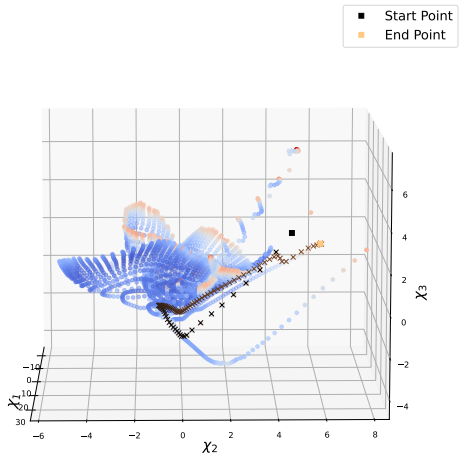
Figure 4.13: The figure shows the quality of non-linear projection operator, CAE. The first figure shows the relative L_2 error for reconstruction of u_Ψ. We can observe that the reconstruction quality is relatively poor for the higher Re and lower t values. Further, the lower figures show the intrinsic coordinates for the complete snapshot matrix. The points are coloured with relative reconstruction error, with red being the highest and blue being the lowest. We can observe that the highest relative error is obtained for the points which lie on the boundary of the encoding domain, suggesting that the points with high Re and low t are mapped on the boundary of the domain.



(a) Relative reconstruction error of u_Ψ after integrating with LSTM model

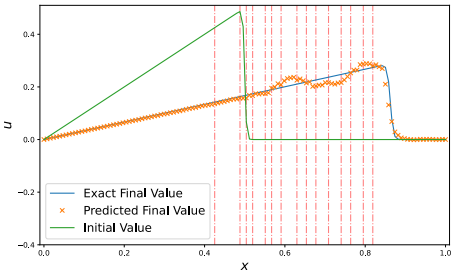


(b) Intrinsic Coordinates of the FEM snapshots with a sample integration case: View 1

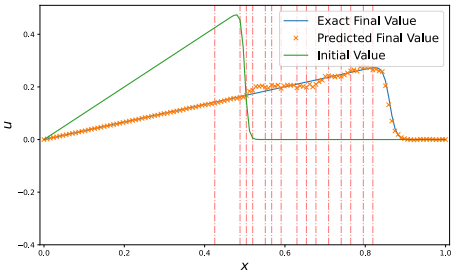


(c) Intrinsic Coordinates of the FEM snapshots with a sample integration case: View 2

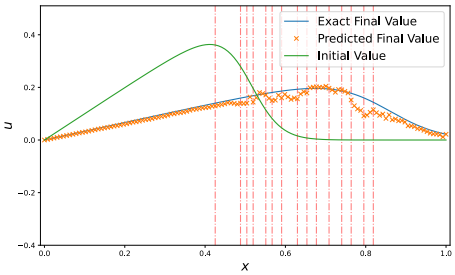
Figure 4.14: The figure shows the quality of non-linear projection and integration operators, CAE – LSTM. The first figure shows the relative L_2 error obtained on integration of $\chi(t = 0)$ over $t \in [0, t_{end})$, using LSTM, followed by projection to full space. We can observe that the integrated χ has maximum error for low Re and high t, these values correspond to the extended branches of the intrinsic coordinates domain. The lower figures highlight the intermediate integration points, obtained from LSTM, for $Re = 200$. We can observe that the initial integration points and the final integration point lie on the branch, and the corresponding points in the above figure also show high error. Further, the integration points pass through the boundary of the domain suggesting that LSTM could have attempted prediction outside the domain.



(a) $Re = 1750$



(b) $Re = 950$



(c) $Re = 150$

Figure 4.15: The figure shows outputs for 3 POD – CAE simulations for $Re = \{150, 950, 1750\}$. The green line shows the initial condition used for the simulation and the blue line shows the expected outcome, the red points show the computed output using the POD – CAE intrusive surrogate model.

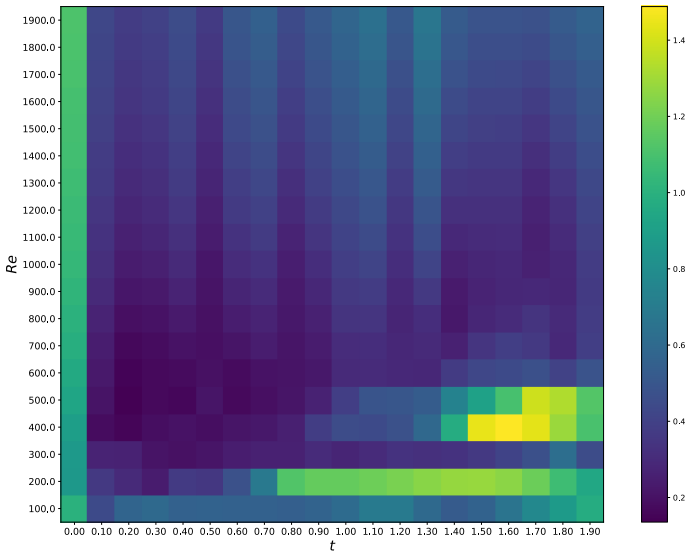
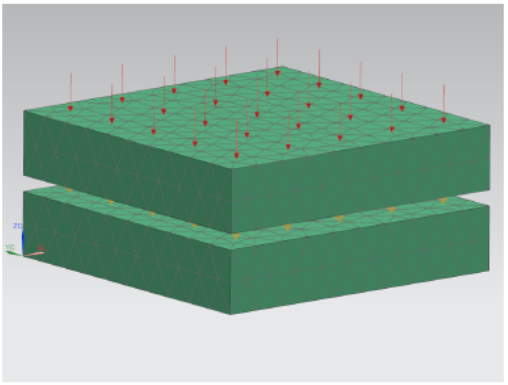
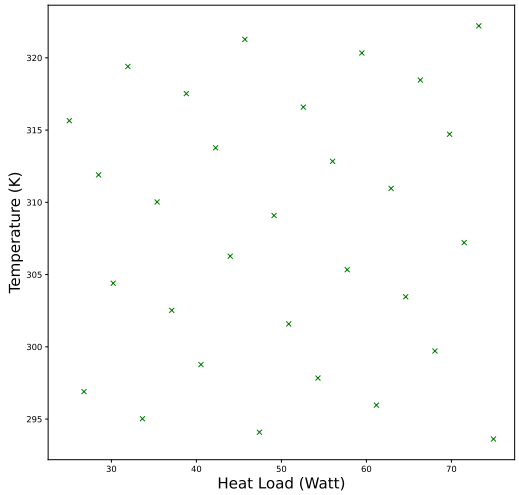


Figure 4.16: The figure shows the relative L_2 error for the formulated POD – CAE intrusive surrogate model. We observe that the error dominant region is still with low Re and high t, this is primarily because the initial χ points and final χ points lie on the extended branches. Additionally, a unidirectional coupling makes the LSTM network oblivious to the implicit Backward Euler method for solving the linear part, thereby disabling it to perform iterative improvement.



(a) Sample domain to be used in the gap – radiation model.



(b) Sampled Parameter values for Gap radiation model

Figure 4.17: The figure shows the domain being simulated in this experiment and the initial parameter values.

4.3 Gap – radiation problem

The Gap – radiation model is a thermal model including radiative coupling. For this model we consider a general heat – transfer equation with radition term as follows

$$\mathbf{C}_p\dot{T} = \mathbf{K}T + \mathbf{R}T^4 + \mathbf{B}u. \tag{4.13}$$

This equation represents several radiation dominated phenomenon occurring in aerospspace, energy and manufacturing domain [35]. For this experiment we will consider a simple case with two separated plates, with upper plate having an inward heat flux, Figure 4.17a shows the setup.

We will consider both plates to be made of steel with thermal capacity 434 $J/(kg \cdot K)$ and thermal conductivity 14 $W/(m \cdot K)$. We take initial temperature,T_0, and heat load, h, as the system parameter. We consider the $T_0 \in [20 \circ C, 300 \circ C]$. We perform simulation for 3600 seconds. Figure 4.17b shows the sampled parameter values for the experiment. In this experiment we consider that the domain discretization is unknown and we use Section 3.1.2 to obtain the operators in Equation (4.13) .

4.3.1 Problem Setup

For this problem we will be begin with reconstruction of the full operators and then follow the steps described in Section 3.1.1 to obtain the final reduced surrogate model. For each

set of parameters we solve Equation (2.19) after application of lifting, to convert Equation (4.13) to have a leading quadratic term[31]. This is done by introducing following variables

$$\begin{aligned} T_1 &= T, \\ T_2 &= T^2, \\ T_3 &= T^3, \end{aligned} \tag{4.14}$$

then we have

$$\begin{aligned} \dot{T}_1 &= \dot{T} \\ \dot{T}_2 &= 2T_1\dot{T} \\ \dot{T}_3 &= 3T_2\dot{T}. \end{aligned} \tag{4.15}$$

Using above formulation we can rewrite the Equation (4.13) as follows

$$\begin{aligned} \mathbf{C}_p\dot{T}_1 &= \mathbf{K}T_1 + \mathbf{R}T_2^2 + \mathbf{B}u \\ \mathbf{C}_p\dot{T}_2 &= 2(\mathbf{K}T_2 + \mathbf{R}T_2T_3 + T_1\mathbf{B}u) \\ \mathbf{C}_p\dot{T}_1 &= 3(\mathbf{K}T_3 + \mathbf{R}T_3^2 + T_2\mathbf{B}u). \end{aligned} \tag{4.16}$$

We can express the combined system as follows

$$\begin{bmatrix} \mathbf{C}_p & & \\ & \mathbf{C}_p & \\ & & \mathbf{C}_p \end{bmatrix} \begin{bmatrix} \dot{T}_1 \\ \dot{T}_2 \\ \dot{T}_3 \end{bmatrix} \approx \begin{bmatrix} \mathbf{K} & & \\ & 2*\mathbf{K} & \\ & & 3*\mathbf{K} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} + \mathbf{H} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \otimes \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} + \begin{bmatrix} \mathbf{B} & & \\ & \mathbf{B} & \\ & & \mathbf{B} \end{bmatrix} \begin{bmatrix} u \\ 0 \\ 0 \end{bmatrix}, \tag{4.17}$$

where

$$\begin{aligned} \mathbf{H} &= \begin{bmatrix} & \mathbf{R} & \\ & & \mathbf{R} \\ & & \mathbf{R} \end{bmatrix}, \\ \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \otimes \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} &= \begin{bmatrix} T_1^2 \\ T_1T_2 \\ T_1T_3 \\ T_2^2 \\ T_2T_3 \\ T_3^2 \end{bmatrix}. \end{aligned} \tag{4.18}$$

In the above equation we drop the mixed terms of form T_iu for the sake of simplicity and possibility to convert it in the form of Equation (2.18) .

Using the this formulation we extract the operators $\mathbf{C}_p$, $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{B}$. Following this we can solve the above equation using the intrusive formulation.

Figure 4.18 show the reconstructed snapshots for different values of u, we can observe an increasing error with time, suggesting that the reconstructed operators aren't exact.

Table 4.4: Parameters used for Operator Inference, POD – CAE surrogate model, and POD – GP with DEIM surrogate model.

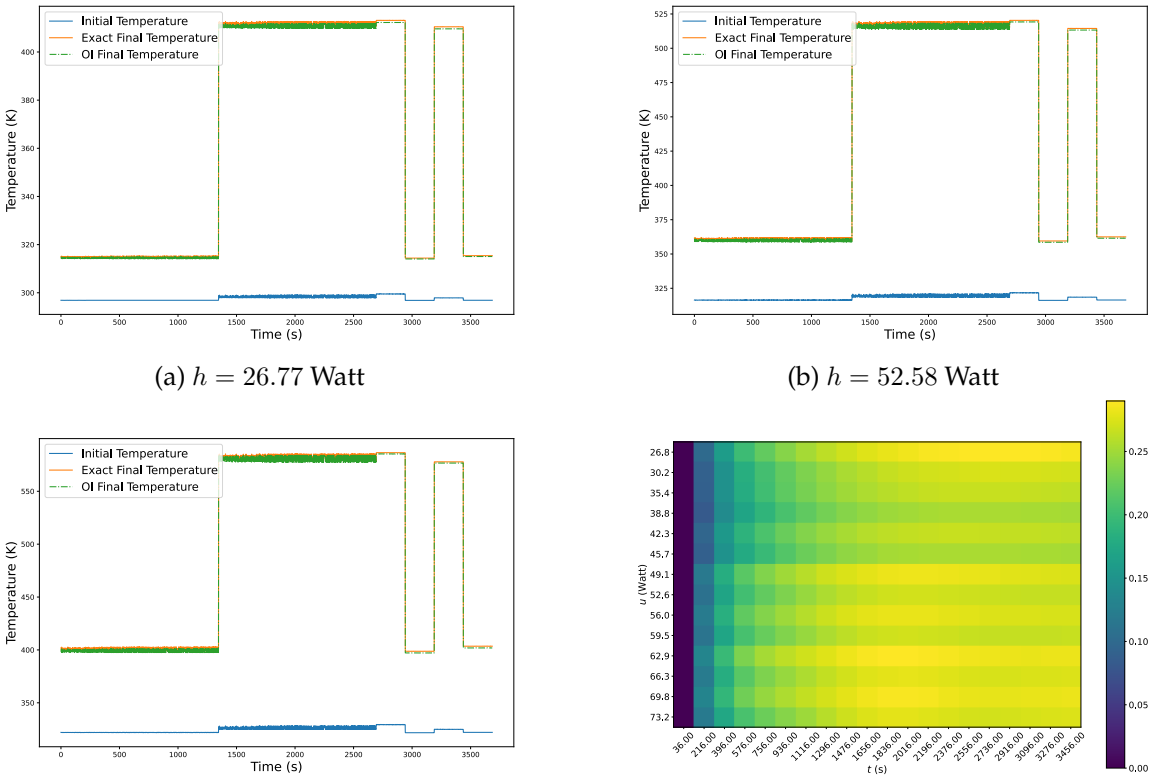
Parameters	Values
General	
Number of spatial grid points (Nx)	3686
t_{end}	3600
Heat load (u)	Sampling shown in Figure 4.17b
dt	36.0
OI	
Regularization search space	$[0.001, 10.0]$
POD – GP with DEIM	
Number of POD modes	8
Number of DEIM modes	Number of POD modes +6
POD – CAE	
Energy covered by linear modes	95 – 98 %
Number of non-linear modes m	4
Convolution type	1D convolutions (unknown discretization)
Pooling	Max pooling
Number of Convolutional layers in encoder	6 depending on m
Number of Convolutional layers in decoder	5 depending on m
Number of LSTM blocks	2
Size of LSTM output	m
LSTM temporal window size	10

This is expected since we drop the mixed terms of T and u as well as we use the regularization in the system. It should be noted that the 3D points are arranged linearly in this experiment because of unavailability of the discretization. We can now formulate the POD – DEIM model using the obtained operators.

4.3.2 POD – DEIM formulation

We can now perform POD on the temperature snapshot matrix and project the Equation (4.13) in the form of Equation (2.11) :

$$\begin{aligned} T &\approx \hat{\mathbf{\Phi}}\hat{T}, \\ \frac{d}{dt}\hat{T} &= \mathbf{K}_{red}\hat{T} + \mathbf{R}_{red}(\hat{\mathbf{\Phi}}\hat{T})^4 + \mathbf{B}_{red}u, \end{aligned} \tag{4.19}$$



(a) $h = 26.77$ Watt

(b) $h = 52.58$ Watt

(c) $h = 74.95$ Watt

(d) Relative L_2 error for reconstruction of temperature snapshot with the inferred operators

Figure 4.18: The figure shows the reconstruction of the temperature snapshot using inferred operators. We can see an increasing trend of error in the reconstructed snapshot suggesting that the operators aren't exact, and the error accumulation can be expected on time integration.

where

$$\begin{aligned} \mathbf{K}_{red} &= \hat{\mathbf{\Phi}}^T \mathbf{C}_p^{-1} \mathbf{K} \hat{\mathbf{\Phi}}, \\ \mathbf{R}_{red} &= \hat{\mathbf{\Phi}}^T \mathbf{C}_p^{-1} \mathbf{R}, \\ \mathbf{B}_{red} &= \hat{\mathbf{\Phi}}^T \mathbf{C}_p^{-1} \mathbf{B}. \end{aligned} \tag{4.20}$$

We define the non-linear term using the DEIM as follows

$$\begin{aligned} \mathbf{R}_{deim} &= \mathbf{R}_{red} \mathbf{W} \left(\mathbf{P}^T \mathbf{W} \right)^{-1}, \\ \mathbf{Z} &= P^T \hat{\mathbf{\Phi}}. \end{aligned} \tag{4.21}$$

Then we can rewrite the Equation (4.19) as following

$$\frac{d}{dt}\hat{T} = \mathbf{K}_{red}\hat{T} + \mathbf{R}_{deim}(\mathbf{Z}\hat{T})^4 + \mathbf{B}_{red}u, \tag{4.22}$$

where:

- $\mathbf{W}$: left hand singular vector of the non-linear snapshot matrix $T_{NL} = \left[T_1^4, T_2^4, \cdots, T_N^4\right]$
- $\mathbf{P}$: selection matrix obtained from Algorithm 1 .

We now analyse the POD modes of system, we can observe in Figure 4.19 that the first mode itself is able to capture more than 99% of system's energy but for the sake of comparison with the POD – CAE method, because of the choice of parameters, we proceed in the further analysis with 8 POD modes.

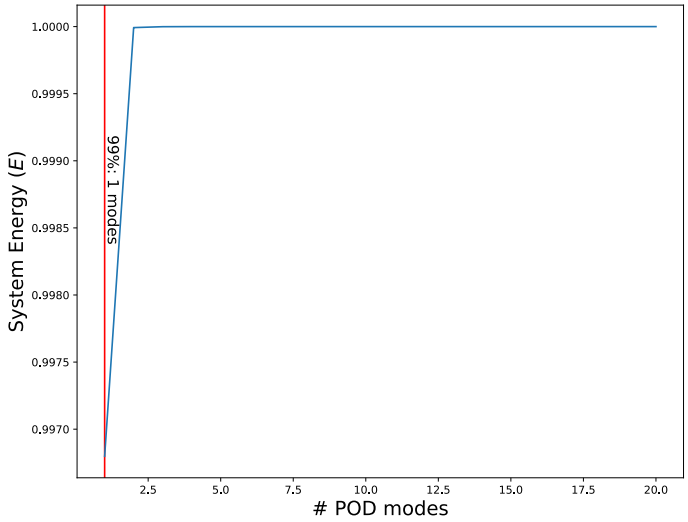
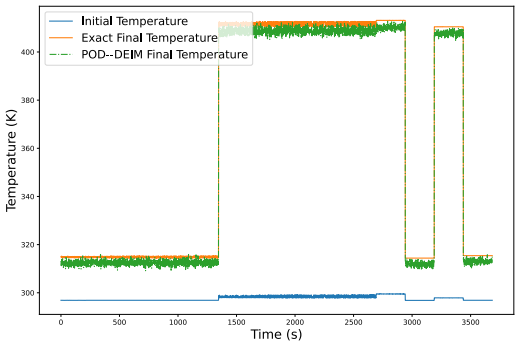
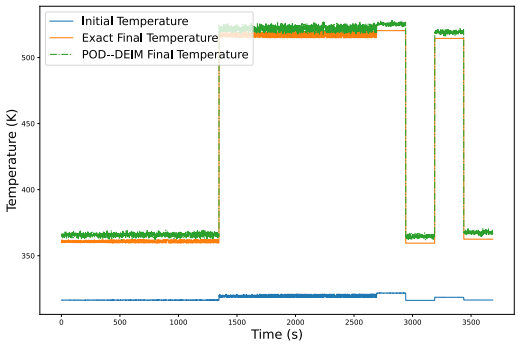


Figure 4.19: The figure shows the ratio of modelled system energy to the total system energy for addition of each POD mode. We can observe that the first POD mode itself is able to capture more than 99% of the system's energy.

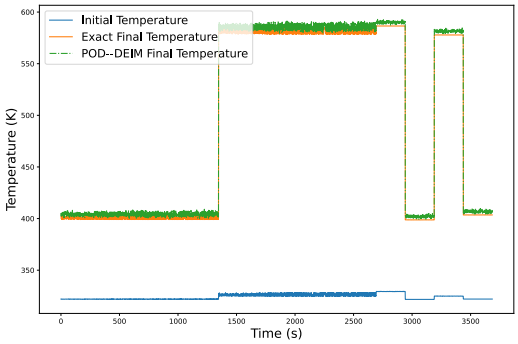
Figure 4.20 shows the results obtained from a POD – GP with DEIM surrogate model. We can notice that the surrogate model is able to capture the overall trend of the temperature from the given snapshot matrix but it fails to compute accurately the temperature values. Further, Figure 4.20d highlights that the projection of the initial condition is inaccurate which can be expected since most of the snapshots in the system have significantly different shape as compared to the initial condition. Therefore the miscalculated initial condition leads to an accumulation of the error during the integration steps.



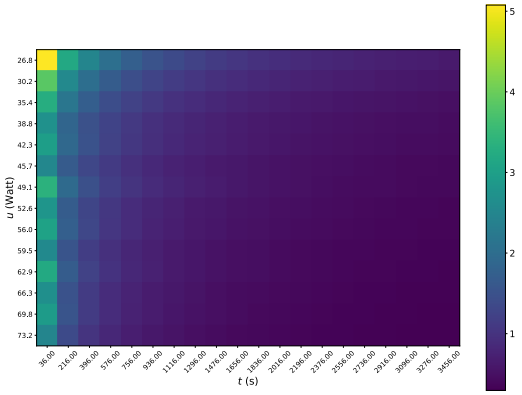
(a) $h = 26.77$ Watt



(b) $h = 52.58$ Watt



(c) $h = 74.95$ Watt



(d) Relative L_2 error for reconstruction of temperature snapshot with the inferred operators

Figure 4.20: The figure shows the reconstruction of the temperature snapshots using POD – GP with DEIM method. We can observe the difference in the exact solution and the POD – GP with DEIM method, further a comparison Figure 4.18 also shows apparent difference in the prediction. We can observe that the POD – GP with DEIM is able to capture the overall temperature trend but makes big error in the projection at an initial stage.

4.3.3 POD – CAE formulation

We now reformulate the Equation (4.13) using thr POD – CAE decomposition. Let the complete decomposition of the snapshot matrix be given as follows

$$T = \hat{\mathbf{\Phi}}\hat{T} + \mathbf{\Psi}T_{\Psi}, \tag{4.23}$$

where $\hat{\mathbf{\Phi}}$ consists of first k POD modes and the $\mathbf{\Psi}$ contains the remaining modes. Using this we can rewrite the Equation (4.13) as follows

$$\frac{d}{dt}\hat{T} = \mathbf{K}_{red}\hat{T} + \mathbf{R}_{red}(\hat{\mathbf{\Phi}}\hat{T} + \mathbf{\Psi}T_{\Psi})^4 + \mathbf{B}_{red}u, \tag{4.24}$$

where the operators $\mathbf{K}_{red}$, $\mathbf{R}_{red}$, and $\mathbf{B}_{red}$ have the same form as described in Equation (4.20) . We now apply DEIM to the non linear term

$$\frac{d}{dt}\hat{T} = \mathbf{K}_{red}\hat{T} + \mathbf{R}_{deim}(\mathbf{Z}_1\hat{T} + \mathbf{Z}_2T_{\Psi})^4 + \mathbf{B}_{red}u, \tag{4.25}$$

where

$$\begin{aligned} \mathbf{R}_{deim} &= \mathbf{R}_{red}\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}, \\ \mathbf{Z}_1 &= P^T\hat{\mathbf{\Phi}}, \\ \mathbf{Z}_2 &= P^T\hat{\mathbf{\Psi}}, \\ T_{\Psi} &= \mathcal{D}(\chi(t)). \end{aligned} \tag{4.26}$$

In the above equation the $\mathcal{D} : \mathbb{R}^m \to \mathbb{R}^{N-k}$ is the decoder of the CAE model and the $\mathcal{E} : \mathbb{R}^{N-k} \to \mathbb{R}^m$ is the encoder of the CAE model, with $\chi(t) = \mathcal{E}(T_{\Psi})$.

Following the directions mentioned in the Section 3.2 we choose $k = 2$ and $m = 4$, since the system has only 2 parameters. We also use a two block LSTM network with the parameters defined in the Table 4.4 . We will again perform a separated training for the CAE and LSTM. Figure 4.21 shows the quality of the CAE projector, we can observe that the CAE is able to reconstruct the field with worst case accuracy of 98.4% which happens due to under-representation of the initial condition profiles, which are significantly different than average profiles, in the snapshot matrix. Further, these initial conditions are mapped to the boundaries of the domain spanned by the snapshots in the encoding space, which also suggests that they are under-represented or have significantly different profile. Figure 4.22 highlights the relative L_2 error for integration of sampled points using LSTM model. We observe that the overall relative error is dominated by the projection operator and the LSTM model has a relatively insignificant error.

4.3.4 Results

Using the above described formulation with the parameters in Table 4.4 we perform integration for 10% of the sampled points that were not used in training of the CAE and LSTM model. Table 4.6 shows the performance of the POD – CAE projection with Full Space Operator Inference, we can observe the POD – GP with DEIM has about 6 times worse performance than the proposed Full OI POD – CAE method. It should be noted that the operators inferred from the OI puts a lower bound of error on both the ROM, POD – GP and POD – CAE, since these operators are projected to a lower dimensional subspace. We can notice that the overall error for the POD – CAE method, with only 2

POD modes, is very close to that of OI, in contrast to POD – GP with DEIM, using 8 POD modes. The main reason behind this behaviour is the capability of the formulated linear model to implicit access the dynamics covered in $\mathbf{\Psi}$. The term $\mathbf{Z}_2 T_\Psi$ allows the linear part to take into account the impact in space $\hat{\mathbf{\Phi}}$ from $\mathbf{\Psi}$ through the interaction caused due to non-linear term.

Table 4.5 highlights the overall simulation time and speedups for the OI, POD – GP with DEIM, and Full OI POD – CAE method. We observe that the POD – GP with DEIM provides the best performance on the cost of accuracy. On the other hand the POD – CAE method improves the relative L_2 error by an order of magnitude at a cost of small gain in simulation time. We can notice that the simulation time of Full OI POD – CAE method is still 3 orders of magnitude lower as compared to OI.

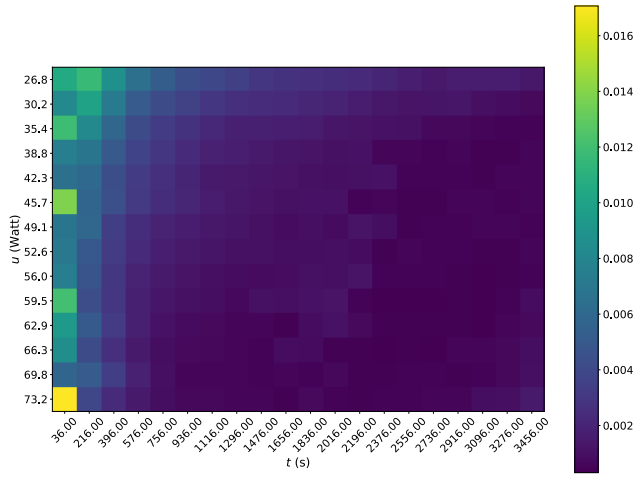
Additionally, Figure 4.23 shows the simulated output compared to the exact solution, for three test cases. We observe that the predicted solution is very close to the actual solution. Further, an inspection of relative error in Figure 4.23 and Figure 4.20d suggests that the dominant source of error in this experiment is the linear model. This is primarily because the initial projection in the linear part has high inaccuracy, as the POD method constructs the basis which covers maximum dynamics of the snapshots therefore, an under-representation of initial condition would construct a projection operator which ignores the variations in the initial condition. We can also notice that this inaccuracy is dissipates due to the coupling with the CAE – LSTM model.

Table 4.5: Runtime comparison for solution of heat equation using the three methods. The average time is computed using 5 runs for each parameter value

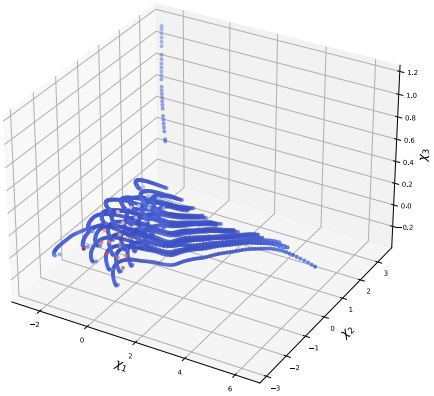
	OI	**POD – GP with DEIM**	**Intrusive POD – CAE**
Average time (s)	31.324 ± 1.351	0.053 ± 0.008	0.210 ± 0.012
Speed-up w.r.t. OI	1.0	591.0 ± 99.6	149.2 ± 15.0

Table 4.6: L_2 error norm comparison for solution of heat equation using the three methods at final time step. The table outlines the results where the POD – CAE (2 POD modes, 4 CAE modes), OI and POD – GP (8 POD modes) with DEIM are deployed anew for each u. The L_2 norm is computed between the final time step result and the exact solution. The POD – CAE model error is lower for similar number of reduced dimensions.

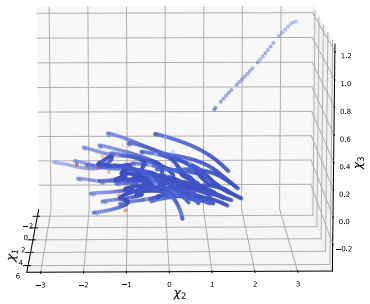
	$\mathbf{u = 28.49}$ **Watt**	$\mathbf{u = 49.13}$ **Watt**	$\mathbf{u = 68.06}$ **Watt**
Absolute L_2 error			
OI	10.641×10^{-3}	17.148×10^{-3}	23.467×10^{-5}
POD – GP with DEIM	79.643×10^{-3}	72.943×10^{-3}	67.977×10^{-3}
Full OI POD – CAE	11.957×10^{-3}	17.568×10^{-3}	24.066×10^{-3}
Relative L_2 error			
OI	0.006%	0.007%	0.007%
POD – GP with DEIM	0.051%	0.031%	0.022%
Full OI POD – CAE	0.007%	0.008%	0.008%



(a) Relative reconstruction error of T_Ψ

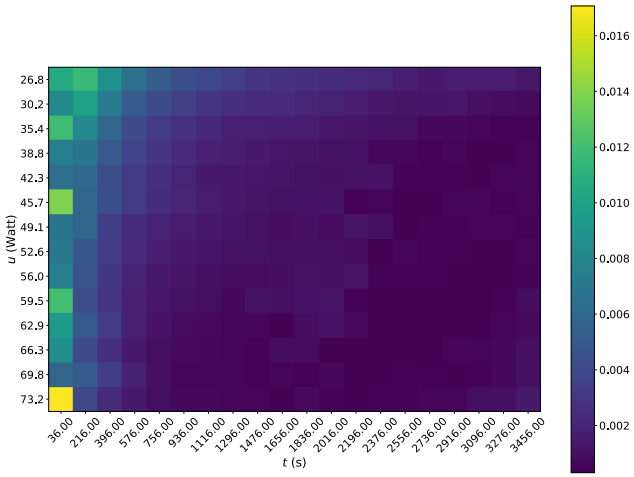


(b) Intrinsic Coordinates of the snapshots: View 1

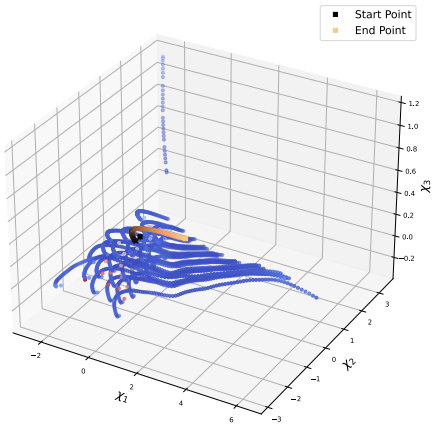


(c) Intrinsic Coordinates of the snapshots: View 1

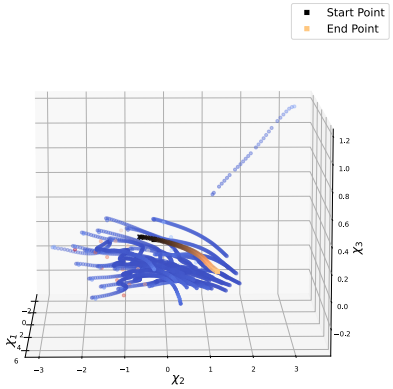
Figure 4.21: The figure shows the quality of non-linear projection operator, CAE. The first figure shows the relative L_2 reconstruction error for the T_Ψ. We can observe that the quality of reconstruction is very well, with worst case having an error of 1.6%. We can also notice that the reconstruction error is high for initial time-steps, which is primarily because the initial profile is underrepresented in the snapshot matrix. In the Figure 4.20a blue curve shows the initial profile, we can see that it differs significantly from the other profiles. The lower images shows the distribution of intrinsic coordinates in the encoding space. The points are colored by relative reconstruction error, with blue being the lowest error and red being the highest. We can observe that the initial conditions lie on the boundaries of the domain spanned by the snapshots in the encoding space.



(a) Relative reconstruction error of T_Ψ

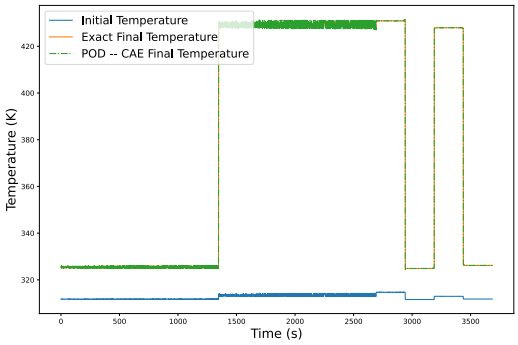


(b) Intrinsic Coordinates of the snapshots: View 1

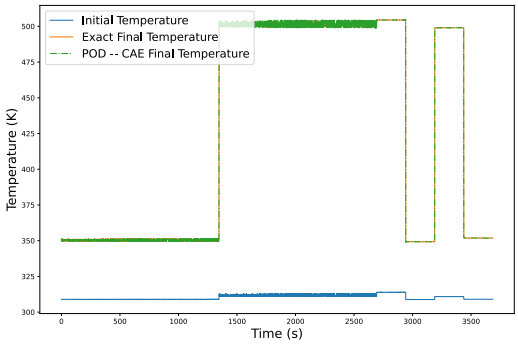


(c) Intrinsic Coordinates of the snapshots: View 2

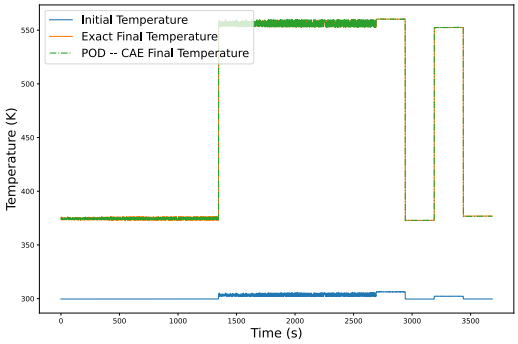
Figure 4.22: The figure shows the quality of non-linear projection and integration operator, CAE – LSTM. The first figure shows the relative L_2 error obtained on integration of $\chi(t = 0)$ over $t \in [0, t_{end})$ using LSTM, followed by projection to full space, using CAE. We can observe that the $\chi(t)$ has maximum error for low t, which arises from the CAE model, since the number of sample having similar profile to initial condition are relatively low in the snapshot matrix, making the reconstruction task difficult for the CAE. We can also observe the similarity between Figure 4.22a and Figure 4.21a which arises from the fact that between CAE and LSTM, the CAE is the dominant source of error. The lower figure highlights the trajectory of a test point, with $h = 59.5$ progressed with LSTM model in the encoding space. We can observe that the initial point is mapped to the boundary, which is high error location, of the encoding space, but the LSTM model progresses the point inside the domain with final state being in a low error region.



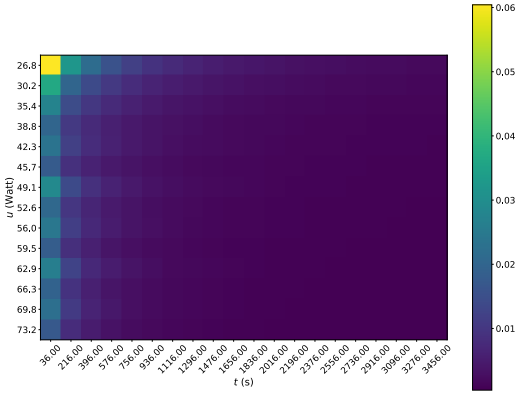
(a) $u = 28.49$ Watt



(b) $u = 49.13$ Watt



(c) $u = 68.06$ Watt



(d) Relative L_2 error for POD–CAE projection with Full Space Operator Inference surrogate model

Figure 4.23: The figure highlights the performance of the POD – CAE POD–CAE projection with Full Space Operator Inference surrogate model. We observe that the overall performance of the model increases significantly as compared to POD – DEIM alone. We observe two orders of magnitude improvement in the performance of the ROM. Further, we can notice that the predicted final state overlaps the exact solution even in the high oscillation regions.

5 Discussion and Conclusions

5.1 Summary and Conclusion

In this thesis work, we developed an explainable machine learning-based extension of the linear model order reduction method, specifically POD – GP with DEIM. This method is developed to tackle the following two problems

- Failure of POD – GP with DEIM for Advection dominated equations
- Failure of explainability in the machine learning based MOR.

The POD – GP with DEIM shows instability as well as slow convergence for advection dominated problems. Often for such problems, ML-based MOR methods are used, which suffer from a lack of explainability. To tackle these problems, we proposed a POD – CAE projection method, in Chapter 3 , which performs a two-part projection, first using POD and the second using the CAE. This projection method allows to inspect the data in low dimensional space and perform specific analysis without going to the full space. We use the POD – CAE projection method to formulate a generalized projected governing equation, a part of which is solved using a classical integrator, while the second is solved using the LSTM model. Further, this generalized formulation allows the flexibility to apply this method to a wide array of problems. One of the salient features of this framework is that the POD counterpart of the projected governing equation has a very similar form to POD – GP with DEIM, yet it doesn't suffer from the drawback of the POD – GP method. Using this formulation we developed the Algorithm 2 for intrusive MOR and Algorithm 3 for non-intrusive MOR. Further, we test the algorithm for intrusive and non-intrusive settings in Chapter 4 .

In Section 4.2 we solve the Burgers' equation for a variety of Reynold's Number values ranging from 100 to 1900. For high Reynold's Number value, the Burgers' equation has a dominant advection term. We solve the problem using the Finite Element Method (FEM), POD – GP with DEIM and POD – CAE intrusive MOR. We set the FEM simulation as the baseline for the Full Order Model (FOM) and use the data generated from the FEM simulation to formulate the two reduced surrogate model. We show that the POD – GP with DEIM is incapable of making an accurate prediction in this case. We then show that the proposed POD – CAE projection method solves the problem accurately and shows an overall reduction in the error. We also analyse the proposed framework's performance compared to FOM and POD – GP with DEIM. We show that the presented model makes a

minuscule compromise in performance, still being at least two orders of magnitude faster than the FOM, to obtain a significant reduction in error as compared to POD – GP with DEIM. Further, we also show that the POD – CAE formulation allows for the inspection of the low dimensional space, which opens the possibility of performing causal analysis for error and stability in the low dimensional space. For example, we inspect the regions of high projection and integration error in the low dimensional space to infer the cause of the error.

In Section 4.3 we solve the radiative heat transfer equation using non-intrusive method. We transform the PDE to a form with a leading quadratic term and use Operator Inference (OI) to obtain full operators. We then follow the formulation proposed in Section 3.1.2 to obtain POD – GP with DEIM model and POD – CAE surrogate model. We use the OI method as the baseline for the FOM and compare the performance and accuracy of the surrogate models to it. We show that the POD – GP with DEIM is able to perform well for this problem. Then we show that using the POD – CAE surrogate model provides a lower relative L_2 error showing the flexibility of the model to be applicable to a wide array of cases. We again show that the POD – CAE method makes a small compromise in the performance to gain overall accuracy.

With the two problems, we were able to show that the proposed POD – CAE method, in general, provides superior performance in terms of accuracy with a similar number of reduced dimensions with small compromises in the computational performance. Further, the proposed method is flexible enough to be applicable to a wide array of problems.

5.2 Future Work

At this point, we would like to highlight some of the possible directions of future work.

Bidirectional coupling of the projected governing equation

In Section 3.1.1 and Section 3.1.2 , we project the governing equation Equation (2.2) into the two spaces $\hat{\mathbf{\Phi}}$ and $\mathbf{\Psi}$. In order to avoid the computational complexity associated with the bidirectional coupling of Equation (3.10a) and Equation (3.10b) we relaxed the coupling condition for Equation (3.10b) , assuming only uni-directional coupling. We then used the relaxed equation for LSTM model. This assumption comes with a cost on overall accuracy and therefore is a possible direction for future work. One of the possible way to provide a bidirectional coupling would be to use the $a_{\hat{\Phi}}$ as an input feature to the LSTM model, therefore allowing the LSTM model to use the dynamics information in $\hat{\mathbf{\Phi}}$ and infer its impact in $\mathbf{\Psi}$ directly.

Error quantification metric in the low dimensional space

Another interesting direction to investigate would be to develop an error quantification metric for low dimensional space. Development of such metric could have several possible utilities such as

- Inspection of the low dimensional space to isolate the region of high error, without intrusive inspection from FOM space.
- Possibility to bifurcate the error of projector and integrator without performing individual analysis of the two from FOM space.

6 Appendix

Numerical Solution of Burger's Equation using Finite Element Method

Let us consider the Burgers' equation with non-homogeneous boundary condition described as follows

$$\dot{u} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{6.1a}$$

$$u(x, Re, 0) = u_0 = \frac{x}{1 + \sqrt{\frac{1}{t_0}}e^{Re\frac{x^2}{4}}}, \quad x \in [0, L], \quad t \in [0, t_{end}] \tag{6.1b}$$

$$u(0, Re, t) = 0 \tag{6.1c}$$

$$u(L, Re, t) = \frac{\frac{L}{t+1}}{1 + \sqrt{\frac{t+1}{t_0}}e^{Re\frac{L^2}{4t+4}}}, \quad t_0 = e^{\frac{Re}{8}} \quad \text{and} \quad Re = \frac{1}{\nu} \tag{6.1d}$$

We can rewrite the PDE in the conservative form as follows

$$\dot{u} + \left(\frac{1}{2}u^2 - \nu u_x\right)_x = 0 \tag{6.2}$$

Now we can consider a N point equidistant discretization of the space with step size h resulting in $x_h = \{x_0, x_1, \cdots, x_{N+1}\}$. Then we can define the following FEM ansatz,

$$u(t, x) = \sum_{i=1}^{N} u_i(t)\phi_i(x) + u_0(t)\phi_0(x) + u_{N+1}(t)\phi_{N+1}(x) \tag{6.3}$$

This ansatz in general fulfils any kind of Dirichlet boundary condition. We choose the trial function such that $\{\phi_i | i = 1, \cdots N\}$ vanish at the boundaries while ϕ_0 and ϕ_{N+1} do not vanish at the boundaries. For the trial and test function we choose a hat function as

described in [11].

$$\begin{aligned}
\phi_i(x) &= \begin{cases} \frac{x-x_{i-1}}{h}, & \text{for } x \in [x_{i-1}, x_i], \\ \frac{x_{i+1}-x}{h}, & \text{for } x \in [x_i, x_{i+1}], , \\ 0, & \text{otherwise.} \end{cases} \quad \text{for } i = 1, 2, \cdots, N \\
\phi_0(x) &= \begin{cases} \frac{x_1-x}{h}, & \text{for } x \in [x_0, x_1], \\ 0, & \text{otherwise.} \end{cases} \\
\phi_{N+1}(x) &= \begin{cases} \frac{x-x_N}{h}, & \text{for } x \in [x_N, x_{N+1}], \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{6.4}$$

With these test and trial functions we can compute the weak form of Equation (6.1a) . We put the FEM ansatz in the Equation (6.1a) , multiply it with the test function, and integrate it over the $x \in [0, L]$:

$$\underbrace{\int_0^L \dot{u}(t,x)\phi_j(x)dx}_{T_1} = -\frac{1}{2}\underbrace{\int_0^L \left(u^2(t,x)\right)_x \phi_j(x)dx}_{T_2} + \nu \underbrace{\int_0^L (u(t,x))_{xx}\phi_j(x)dx}_{T_3} \tag{6.5}$$

For the sake of clarity we now formulate each of the terms, described above, separately:

$$\begin{aligned}
T_1 &= \int_0^L \dot{u}(t,x)\phi_j(x)dx = \int_0^L \left(\sum_{i=1}^N \dot{u}_i(t)\phi_i(x) + \dot{u}_0(t)\phi_0(x) + \dot{u}_{N+1}(t)\phi_{N+1}(x)\right)\phi_j(x)dx \\
T_1 &= \sum_{i=1}^N \dot{u}_i(t) \underbrace{\int_0^L \phi_i(x)\phi_j(x)dx}_{\mathbf{M}_{i,j}} + \dot{u}_0(t) \underbrace{\int_0^L \phi_0(x)\phi_j(x)dx}_{\mathbf{M}_{0,j}} + \dot{u}_{N+1}(t) \underbrace{\int_0^L \phi_{N+1}(x)\phi_j(x)dx}_{\mathbf{M}_{N+1,j}}
\end{aligned} \tag{6.6}$$

$$\begin{aligned}
T_2 &= \int_0^L \left(u^2(t,x)\right)_x \phi_j(x)dx = \int_0^L \Bigg(\sum_{i=1}^N u_i^2(t)\phi_i(x) + u_0^2(t)\phi_0(x) \\
&\qquad + u_{N+1}^2(t)\phi_{N+1}(x) \Bigg)_x \phi_j(x)dx \\
T_2 &= \sum_{i=1}^N u_i^2(t) \underbrace{\int_0^L (\phi_i(x))_x \phi_j(x)dx}_{\mathbf{D}_{i,j}} + u_0^2(t) \underbrace{\int_0^L (\phi_0(x))_x \phi_j(x)dx}_{\mathbf{D}_{0,j}} \\
&\qquad + u_{N+1}^2(t) \underbrace{\int_0^L (\phi_{N+1}(x))_x \phi_j(x)dx}_{\mathbf{D}_{N+1,j}}
\end{aligned} \tag{6.7}$$

$$
\begin{aligned}
T_3 = \int_0^L \left(u(t,x)\right)_{xx} \phi_j(x) dx = \int_0^L \Bigg(\sum_{i=1}^{N} u_i(t)\phi_i(x) + u_0(t)\phi_0(x) \\
+ u_{N+1}(t)\phi_{N+1}(x) \Bigg)_{xx} \phi_j(x) dx \\
T_3 = \sum_{i=1}^{N} u_i(t) \int_0^L \left(\phi_i(x)\right)_{xx} \phi_j(x) dx + u_0(t) \int_0^L \left(\phi_0(x)\right)_{xx} \phi_j(x) dx \\
+ u_{N+1}(t) \int_0^L \left(\phi_{N+1}(x)\right)_{xx} \phi_j(x) dx
\end{aligned}
\tag{6.8}
$$

Now to resolve T_3 in the form of derivative of test and trial function we apply integration by parts and put the boundary condition:

$$
\begin{aligned}
T_3 = -\sum_{i=1}^{N} u_i(t) \underbrace{\int_0^L \left(\phi_i(x)\right)_x \left(\phi_j(x)\right)_x dx}_{\mathbf{B}_{i,j}} \\
\underbrace{+ u_0(t)\frac{1}{h}\phi_j(0) - u_0(t) \int_0^L \left(\phi_0(x)\right)_x \left(\phi_j(x)\right)_x dx}_{\mathbf{B}_{0,j}} \\
\underbrace{+ u_{N+1}(t)\frac{1}{h}\phi_j(L) - u_{N+1}(t) \int_0^L \left(\phi_{N+1}(x)\right)_x \left(\phi_j(x)\right)_x dx}_{\mathbf{B}_{N+1,j}}
\end{aligned}
\tag{6.9}
$$

Now we can define the matrices as follows:

$$
\begin{aligned}
\mathbf{M} &= \frac{h}{6} \begin{pmatrix} 1 & 4 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & 4 & 1 & \\ & & & 1 & 4 & 1 \end{pmatrix}, \\
\mathbf{D} &= \begin{pmatrix} -\frac{1}{2} & 0 & \frac{1}{2} & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} & \\ & & & -\frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \\
\mathbf{B} &= \frac{1}{h} \begin{pmatrix} -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 2 & -1 \end{pmatrix}
\end{aligned}
\tag{6.10}
$$

It should be noted that the matrix $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{B}$ are non-square and are in $\mathbb{R}^{N\times N+2}$. We can now substitute these terms back in Equation (6.5) :

$$\mathbf{M}\dot{u}(t) = -\frac{1}{2}\mathbf{D}u^2(t) - \nu\mathbf{B}u(t) \tag{6.11}$$

To make the matrices square in the above equation we can introduce a source term f which combines the contribution of the three terms at the boundaries. The f can be written as follows

$$f(t) = \begin{pmatrix} -\frac{h}{6}\dot{u(0,t)} + \frac{1}{4}u^2(0,t) + \frac{\nu}{h}u(0,t) \\ 0 \\ \vdots \\ 0 \\ -\frac{h}{6}\dot{u(L,t)} - \frac{1}{4}u^2(L,t) + \frac{\nu}{h}u(L,t) \end{pmatrix} \tag{6.12}$$

Now we can rewrite the equation in the following form

$$\mathbf{M}\dot{u}(t) = -\frac{1}{2}\mathbf{D}u^2(t) - \nu\mathbf{B}u(t) + f(t) \tag{6.13}$$

With this formulation we can integrate the above equation using any suitable explicit or implicit time integration scheme. For this thesis work we perform a Backward Euler time integration.

Bibliography

[1] Cs231n: Convolutional neural networks for visual recognition. http://cs231n.stanford.edu/.

[2] Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[4] J. Armstrong and F. Collopy. Causal forces: Structuring knowledge for time-series extrapolation. 1993.

[5] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model reduction and neural networks for parametric pdes, 2020.

[6] N. D. Brenowitz and C. S. Bretherton. Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, 45(12):6289–6298, 2018. doi: https://doi.org/10.1029/2018GL078510. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018GL078510.

[7] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Wiley-Interscience, USA, 1987. ISBN 0471910465.

[8] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011. doi: https://doi.org/10.1002/nme.3050. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.3050.

[9] Saifon Chaturantabut and Danny C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010. doi: 10.1137/090766498. URL https://doi.org/10.1137/090766498.

[10] Thomas Daniel, Fabien Casenave, Nissrine Akkari, and David Ryckelynck. Model order reduction assisted by deep neural networks (rom-net). *Advanced Modeling and Simulation in Engineering Sciences*, 7(1):16, Apr 2020. ISSN 2213-7467. doi: 10.1186/s40323-020-00153-6. URL https://doi.org/10.1186/s40323-020-00153-6.

[11] K. Eriksson, editor. *Computational differential equations*. Cambridge University Press, Cambridge ; New York, 1996. ISBN 9780521567381 9780521563123.

[12] Stefania Fresca, Luca Dede', and Andrea Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes. *Journal of Scientific Computing*, 87(2):61, Apr 2021. ISSN 1573-7691. doi: 10.1007/s10915-021-01462-7. URL https://doi.org/10.1007/s10915-021-01462-7.

[13] Aurelien Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA, 2017. ISBN 978-1491962299.

[14] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[16] Katja Hansen, Grégoire Montavon, Franziska Biegler, Siamac Fazli, Matthias Rupp, Matthias Scheffler, O. Anatole von Lilienfeld, Alexandre Tkatchenko, and Klaus-Robert Müller. Assessment and validation of machine learning methods for predicting molecular atomization energies. *Journal of Chemical Theory and Computation*, 9(8):3404–3419, July 2013. doi: 10.1021/ct400195d. URL https://doi.org/10.1021/ct400195d.

[17] J.S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.02.037. URL https://www.sciencedirect.com/science/article/pii/S0021999118301190.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[19] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations, 2019.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[21] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. doi: https://doi.org/10.1002/aic.690370209. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209.

[22] Yvon Maday and Olga Mula. A generalized empirical interpolation method: Application of reduced basis techniques to data assimilation. *Springer INdAM Series*, page 221–235, 2013. ISSN 2281-5198. doi: 10.1007/978-88-470-2592-9_13. URL http://dx.doi.org/10.1007/978-88-470-2592-9_13.

[23] Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3):037106, Mar 2021. ISSN 1089-7666. doi: 10.1063/5.0039986. URL http://dx.doi.org/10.1063/5.0039986.

[24] Shane A. McQuarrie, Cheng Huang, and Karen E. Willcox. Data-driven reduced-order models via regularised Operator Inference for a single-injector combustion process. *Journal of the Royal Society of New Zealand*, 2021. ISSN 03036758. doi: 10.1080/03036758.2020.1863237.

[25] Mrgrhn. Convolutional autoencoders (cae) with tensorflow. https://ai.plainenglish.io/convolutional-autoencoders-cae-with-tensorflow-97e8d8859c
Jan 2021.

[26] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles, 2017.

[27] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[28] Clarence W. Rowley, Tim Colonius, and Richard M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115–129, 2004. ISSN 01672789. doi: 10.1016/j.physd.2003.03.001.

[29] Arvind K. Saibaba. Randomized discrete empirical interpolation method for nonlinear model reduction, 2020.

[30] Shivamb. 3d convolutions : Understanding use case. https://www.kaggle.com/shivamb/3d-convolutions-understanding-use-case, Jan 2019.

[31] Felix Michael Sievers. Uncertainty quantification of reduced order models via stochastical reduced order basis and non-intrusive operator inference, 2021.

[32] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, Dec 2018. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.08.029. URL http://dx.doi.org/10.1016/j.jcp.2018.08.029.

[33] Ben Stevens and Tim Colonius. Finitenet: A fully convolutional lstm network architecture for time-dependent partial differential equations, 2020.

[34] S. Volkwein. Model reduction using proper orthogonal decomposition, 2011. lecture notes, university of konstanz. http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Vorlesung.pdf, 2011.

[35] Qinyu Zhuang, Juan Manuel Lorenzi, Hans-Joachim Bungartz, and Dirk Hartmann. Model order reduction based on runge-kutta neural network, 2021.