DEEP MULTILAYER CONVOLUTION FRAMEWORKS FOR DATA-DRIVEN LEARNING OF NONLINEAR DYNAMICS IN FLUID FLOWS

By

Shivakanth Chary Puligilla

Bachelor of Technology in Aeronautical Engineering
JNTU
2010

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
August 2, 2018

DEEP MULTILAYER CONVOLUTION FRAMEWORKS FOR DATA-DRIVEN LEARNING OF NONLINEAR DYNAMICS IN FLUID FLOWS

Thesis Approved:

Dr. Balaji Jayaraman

Thesis Advisor

Dr. He Bai

Dr. Omer San

Dr. Rushikesh Kamalapurkar

Dedicated to

my mother Suryamani

and

father Satyam Chary

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

ACKNOWLEDGMENTS

I thank Dr. Balaji, for providing support through my research work. Providing me with feedback that helped me to work efficiently and focus on improving knowledge. He would always say "Do the karma (deed) and the results will follow". I would like to thank my committee members Dr. He Bai, Dr. Rushi Kamalapurkar and Dr. Omer San for providing helpful feedback. I would like to thank Dr. San for all the ideological discussions on research ranging from machine learning to Turbulence.

Further, I would like to thank all my lecturers for imparting a great deal of knowledge without which my research would not have been successful.

I thank my friends and colleagues in lab, Romit, Chen Lu, Matt Mitchell, Saadbin, Prashant, Saurab, Vivek, Mansoor and Abdullah for all the time spent together learning and discussing ideas. Romit Maulik for the discussions on Machine learning theories and ideas we thought about.

I thank the MAE staff Beth, Chelsea, Diane and Daleene who have been very friendly and helpful. Special thanks to Beth for helping me swiftly at times of need.

I thank my partner in crime (life and research) my wife Vagbharathi for being there for me in every aspect of my life and providing me unconditional love and encouragement. I thank my parents and siblings for there love and support which gave me strength and courage to pursue my dreams. Especially, my mother who passed away an year ago during my study here, who against all odds took care of me and motivated me to follow my dreams. Helped me fulfill my ambitions and made me come to USA to pursue my graduate studies.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: SHIVAKANTH CHARY PULIGILLA

Date of Degree: JULY, 2018

Title of Study: Deep Multilayer Convolution Frameworks for Data-Driven Learning of Nonlinear Dynamics in Fluid Flows

Major Field: MECHANICAL AND AEROSPACE ENGINEERING

Abundance of measurement and simulation data has led to the proliferation of machine learning tools for model-based analysis and prediction of fluid flows over the past few years. In this work we explore globally optimal multilayer convolution models such as neural-networks(NNs) for learning and predicting dynamics from transient fluid flow data. While machine learning in general depends on data quality of the system, it is important for a given data-driven learning architecture to make the most of this available information. To this end, we cast the suite of recently popular data-driven learning approaches that approximate Markovian dynamics through a linear model in a higher-dimensional feature space as a multilayer architecture similar to neural networks, but with layer-wise locally optimal convolution mappings. As a contrast, we also represent the traditional neural networks with some slight modifications as a multilayer architecture, with convolution maps optimized to minimize the global learning cost. We show through examples, that globally optimal NN-like methods owe their success to leveraging the extended learning parameter space available in multilayer models to achieve a common goal of minimizing the cost function while incorporating nonlinear function maps between layers. While locally optimal methods allow for forward-backward convolutions, the standard globally optimal NNs can only handle forward maps which prevent their use as Koopman approximation tools. To this end, we developed novel deep learning neural network architecture, deep Koopman network which overcomes this limitation of symmetry by addition of penalty network. Further, we explored the feasibility of deep autoencoder networks (DAENs) as data-driven mappings into the observable space where the dynamics can be approximated as a linear time-invariant (LTI) system. The eigenmodes and the eigenvalues of the Koopman operator provide information about the structures in the data that are associated with their unique growth rate and frequency. The relevance of these structures and eigenvalues to the real system is tied to how closely the Koopman operator-based model approximates the real dynamics, which, in turn depends on the choice of observable. Traditional approaches for non-local optimization such as those in neural networks and deep learning are gradient-based and hence, limited to convolution basis functions whose derivatives are either known or computed accurately using numerical means. To realize the full potential of this deep learning framework, these algorithms need to be extended to arbitrary choice of convolution basis. To this end, we explored the use of gradient free optimization techniques for learning using a wider choice of functions. We illustrated these ideas by learning the dynamics from snapshots of training data and predicting the temporal evolution of canonical nonlinear fluid flows including the transient limit-cycle attractor in a cylinder wake and the instability-driven dynamics of buoyant Boussinesq flow.

Journal Publications

- Neural Networks as Globally Optimal Multilayer Convolution Architectures for Learning Fluid Flows, SC Puligilla, B Jayaraman - arXiv preprint arXiv:1806.08234, 2018

- Spectral Analysis of Nonlinear Fluid Flows using Deep Autoencoder Network-based Convolutions, SC Puligilla, B Jayaraman, (Manuscript in preparation)

- Deep Koopman Multilayer Networks for Data-driven Modeling and Spectral Analysis of Fluid Flows., SC Puligilla, B Jayaraman, (Manuscript in preparation)

Conference Proceedings

- Deep Multilayer Convolution Frameworks for Data-Driven Learning of Fluid flow Dynamics, SC Puligilla, B Jayaraman - 2018 Fluid Dynamics Conference, AIAA AVIATION 2018

- A convolution and Artificial Neural Network enabled Machine learning framework for fluid flows, P Shivakanth Chary, B Jayaraman, 37th Oklahoma AIAA/ASME Symposium

TABLE OF CONTENTS

LIST OF TABLES

Table Page

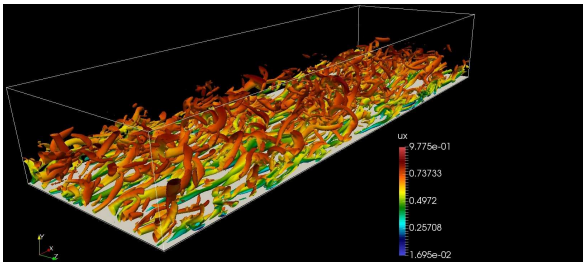
LIST OF FIGURES

CHAPTER 1

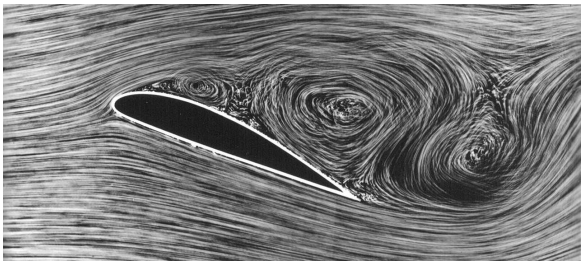
Introduction

1.1 Overview and Motivation

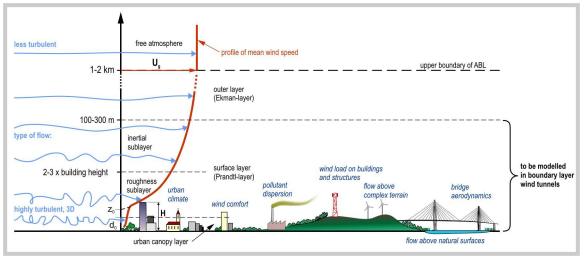
Fluid flows are predominantly multiscale phenomena occurring over a wide range of length and time scales such as transition (Edstrand et al. 2018), turbulence (Wu et al. 2017) and flow separation (Deem et al. 2018). Direct numerical simulation (DNS) of such realistic high Reynolds number flows even in their canonical forms is a challenge even with current computing capacity. On the other hand, advances in experimental techniques for visualization and data acquisition have led to an abundance of fluid flow measurement data, but these measurements are often sparse and in many cases the underlying phenomenology or governing model is not known. In both these cases, there is a need for efficient data-driven models to serve the twin goals of system modulation to achieve desired effects, i.e. flow control (Kim & Bewley 2007, Brunton & Noack 2015) or forecasting for improved and informed decision making (Cao et al. 2007, Fang et al. 2009, Benner et al. 2015) or both. Additionally, data-driven models also allow for extraction of dynamical and physical characteristics to generate novel insight (Bagheri 2013, Rowley et al. 2009) of the system behavior. A schematic of the general practice/methodology used in data driven techniques is illustrated in Fig. 1.2. Wherein, a full order model/experimental data can be used for learning data via unknown data driven basis or known basis. Learning using unknown basis generally are focused on predictive modeling, while the central point of learning via known basis is to improve the knowledge of the underlying physics or phenomenology. In flow control applications linear operator based control is often preferred due to



(a) Flow with multiple scales [In-house]



(b) Flow separation over airfoil [www.dlr.de-Imprint]



(c) Flow structures in Atmosphere boundary layer [www.bmeafl.com-AFL]

Figure 1.1: Engineering flows

the sheer volume of available expertise and their past success (Rowley & Dawson 2017) although nonlinear operator based flow control is an emerging area of research. Consequently, learning a linear system model from data is often preferred. The vast amount of recent literature (Schmid 2010, Williams et al. 2015) including that from our team (Lu et al. 2018) addresses this broad area of need, but do not perform adequately in data-sparse situations. In this work, we explore nonlinear machine learning frameworks that are capable of overcoming this limitation for canonical fluid flows which are inspired from the engineering flows presented in Fig. 1.1.

A good data-driven model should perform well in both system identification and

prediction using limited amounts of data. In addition, these models need to be computationally tractable which makes dimensionality reduction essential. System identification enables learning of stability and flow characteristics such as unstable modes and coherent structures for the purposes of understanding the flow. For example, proper orthogonal decomposition (POD) (Lumley 2007) via singular value decomposition (SVD) (Trefethen & Bau III 1997) and its close cousin, the Dynamic mode decomposition (DMD) (Schmid 2010) are well known methods to extract such relevant spectral information. However, the capacity of DMD for long-term prediction is underwhelming (Lu & Jayaraman 2017). POD-based methods that use Galerkin projection onto the flow governing equations are more successful, but require knowledge of the system. In this study, we focus on purely data-driven scenarios without knowledge of governing equations. By long-time predictions, we imply evolving the system model over multiple characteristic time-scales beyond the training regime. The other type is to employ the model to predict different system trajectories which is commonly referred using the generic term *forecasting*. Obviously, the precise definition of the 'long-time' prediction or forecasting depends on the flow physics of interest. For example, a limit-cycle system evolving on a stable attractor will be more amenable to forecasting from limited data as compared to more complex nonlinear mixing dynamics. In the case of cylinder wake flow explored in this study, forecasting represent predicting the limit cycle (Berkooz et al. 1993, Noack et al. 2003) dynamics using limited data in the transient region. We explore such cases as they are sensitive to error growth and hence, used to evaluate a given model. Errors in model learning can be attributed to limited training data, measurement noise, model over-fitting and insufficient validation (Bishop et al. 1995, Christopher 2016).

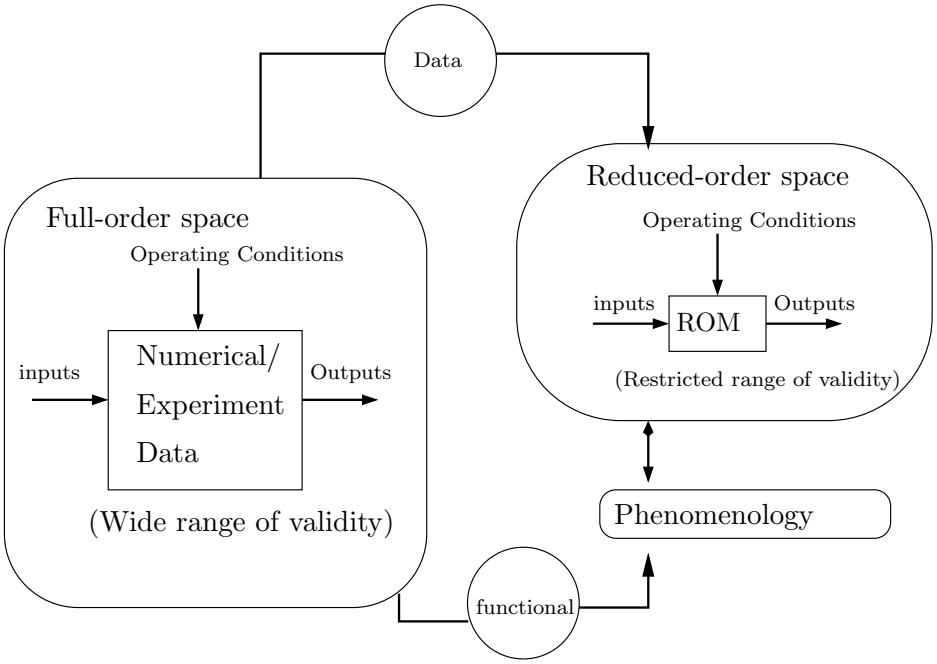


Figure 1.2: A schematic of the overview, describing the research approach used in developing the methodology for this work.

1.2 Markov Models

There are two classes of approaches for modeling dynamical systems from limited data, namely Markov and non-Markov models. For a given current state $\boldsymbol{x}_t$ and future state $\boldsymbol{x}_{t+T}$ of a dynamical system, a Markov model (Wu & Noé 2017), under some transformation $g,\ h$, evolves the system state as $g(\boldsymbol{x}_{t+T}) = \mathcal{K}h(\boldsymbol{x}_t)$. Learning such an operator $\mathcal{K}$ is a key component of building such a model. We consider a Markovian process to be minimally memory dependent and popular approaches for modeling such systems include dynamic mode decomposition (DMD) (Schmid 2010, Rowley et al. 2009) and Feed forward neural networks (FFNN). Recently, many of the linear operator (Rowley & Dawson 2017, Mezić 2005, Williams et al. 2015, Lu et al. 2018) methods for modeling nonlinear dynamics have been shown to be generalizable into a Koopman operator (Koopman 1931) theoretic framework. The Koopman approximation-based methods can be considered as a special case of Markov models if the transformations ($g = h$) to the feature space are identical. On the

other hand, if the model incorporates history (or copious amounts of memory) of the state variables to predict a future state, then it is considered non-Markovian. Recurrent neural networks(RNN) are good examples of non-Markovian models and have been employed for learning dynamical systems both in the past (Hopfield 1982, Hochreiter & Schmidhuber 1997) and in recent times (Soltani & Jiang 2016, Yu et al. n.d.). Although these have shown success, they are very hard to build and train (Bengio et al. 1994) as compared to standard feed forward neural networks (FFNNs) (Bengio et al. 2015). This is because, the standard backpropagation-based algorithms introduce over-fitting and the longer training times can lead to exploding or vanishing gradient problems.

1.2.1 Symmetric Markov Linear Model: Koopman

While Markov models are popular, especially the linear variants, their success is often tied to two aspects: (i) the ability of the projection or convolutions to the feature space (Rowley & Dawson 2017, Taira et al. 2017, Lu et al. 2018) to accurately map data without loss of information while incorporating the appropriate degree of nonlinearity and (ii) their ability to capture the evolution of the dynamics in the feature space (Lu et al. 2018). This renders many such learning methodologies into an exercise in identifying the optimal 'magic' feature maps. A common approach to building such nonlinear convolution operators is to layer multiple 'elementary' convolutions (Lu et al. 2018). While DMD (Schmid 2010, Rowley et al. 2009) employs a single-layer convolution operator based on singular value decomposition (SVD) of the training data, its multilayer variant EDMD (Williams et al. 2015) incorporates a second convolution (over a layer of SVD convolution) by embedding nonlinear functions. This approach is effective if one knows the nature of the nonlinearity *a priori*, but often results in a very high-dimensional function dictionary to approximate the data accurately. The kernel variant of this method, KDMD (Williams et al. 2014)

helps with dimensionality reduction, but is once again limited by the choice of the kernel function. A major limitation of all the above multi-layer methods is that the feature maps are treated independent of each other and obtained using local criteria, i.e. by direct function evaluation or projection onto a basis space that is optimal with respect to local features. To overcome these limitations there have been efforts to use deep neural networks (DNNs) to identify multilayer convolution maps (Puligilla & Jayaraman 2018*a*, Otto & Rowley 2017, Lusch et al. 2017) that embed the nonlinear dynamical system into a Koopman basis space with linear dynamics (Mezić 2005). The success reported from the use of such *deep learning-based Koopman observables* as nonlinear convolution operators to build linear Markov models can be attributed to finding the optimal transformation using multilevel convolution by identifying the appropriate combinations among the different layers in the architecture using an efficient algorithm. While the above work focuses on learning the optimal nonlinear convolution or feature map followed by the linear operator, the work in this article explores learning the overall dynamics using a nonlinear Markov model for the given architecture. Contrasting the two, the former enables prediction of nonlinear dynamics via a linear operator by *learning the appropriate map*, the latter *learns a nonlinear model for the dynamics as a whole* for a given multilayer convolution architecture. Both these Markovian approaches leverage multilayer convolution and deep learning in different ways and have broad implications for data-driven modeling.

1.3 Deep Neural Networks

Data-driven learning algorithms are being used successfully in many engineering applications ranging from optimal sensor placements (Manohar et al. 2017) and reconstruction (Brunton et al. 2013, Lu et al. 2018) to real time active flow control of flow separation over an airfoil (Deem et al. 2018), and also applications in identification of phenomenology or governing models. Specifically, in the realms of flow control and

system identification, Koopman theory has emerged as an important research direction of interest as a generalized framework for modeling nonlinear dynamics. This is generally accomplished through mapping to the Koopman invariant subspaces where the system dynamics can be approximated as linear. Dynamic mode decomposition (Schmid 2010, Rowley et al. 2009) and its variants (Williams et al. 2014, Williams et al. 2015) are popular Koopman approximation methods and typically employ data-driven proper orthogonal decomposition (POD) convolutions. These methods have had varying success in modeling dynamics of fluid flows and have proven to be successful as long as the convolution map is optimal, i.e. produces a mapping that both sparse and appropriately nonlinear. This has motivated the need to find the optimal convolution maps (Lusch et al. 2017, Otto & Rowley 2017, Williams et al. 2015, Wu et al. 2018) for a given nonlinear fluid flow physics.

In addition to learning the system dynamics, the primary goal of data driven methodologies is to extend relevant information from a given set of training measurements to an unknown state as a predictive modeling framework. Such predictive data driven modeling has great relevance in fields like weather forecasting, where the dynamics are multi-scale and turbulent (highly nonlinear) for which there exists no reliable low-order representation capable of capturing the dynamics. In many such cases, either the underlying governing model is unknown or the direct computation of these physics is far fetched, even with current computing power. Our goal through this effort is to develop data-driven models that allow for system identification and also provide reliable predictions of dynamics.

In particular, a Koopman theory based model should incorporate the following:

1. A convolution map that accurately projects the state data to and from the feature space without loss of information, while incorporating the appropriate degree of nonlinearity.

2. A system identification framework to capture the evolution of the dynamics in

the feature space.

While many learning algorithms try to find/identify the optimal 'magic' feature maps, a common practice is to build nonlinear convolution operators that are composed of multiple elementary convolutions layered on top of each other and generalized as a multilayer convolution framework (Lu et al. 2018). However, these operators often are an assumed form or optimized locally. The framework presented here utilizes Multilayer Global Optimal Convolution (MGOC) to identify the optimal combination of the convolution operators for an improved Koopman approximation of the dynamics (Puligilla & Jayaraman 2018*c*).

The learning process in MGOC framework is based on back-propagation algorithm (Rumelhart et al. 1988), which is widely used and helped in popularizing artificial neural networks. Artificial neural networks have been widely used as shallow networks that have universal approximation capabilities. The deep variant have demonstrated excellent accuracy in image classification and regression. More recently, artificial neural networks have been used in fluid dynamics community as model order reduction methods with main emphasis on extracting underlying physics (Raissi et al. 2017). Furthermore, artificial neural networks are used for regression and prediction. Neural networks extract the underlying physics or patterns with the help of activation or transformation functions, which are vital to this process. While back-propagation needs gradient information to optimize the weights connecting the neurons. Generally, conventional activation functions used in the machine learning community are not enough to model fluid flows or help in extracting the underlying physics. Methods like proper orthogonal decomposition (Schmid 2010), wavelet and Fourier transformation functions have been found to be optimal to varying degrees for modeling fluid flows. These transformation functions do not have derivative information based on which back-propagation works. There is a need for an optimization technique that potentially replaces back propagation and aid in the usage of physics based transfor-

mations which enable efficient modeling. To accomplish this, we have used genetic algorithm which is a global optimizer and do not require gradient information to optimize the system. There have been attempts to use genetic algorithm based methods to optimize neural networks with limited success (Yen & Lu 2003, Montana & Davis 1989), while review articles by Yao (Yao 1999) and Ding (Ding et al. 2013) have provided various possibilities using genetic algorithm in neural networks. This ranges from designing the neural network architecture by optimizing the hyper parameters involved while using back-propagation to optimize weights to both designing and optimizing the hyper parameters, weights and number of layers. In this article, we present our initial efforts into using genetic algorithm to train weights using non–dominated sorted genetic algorithm (Deb et al. 2002) with conventional transformation functions. We have investigated this framework on canonical flows like flow over cylinder.

The objectives of this study are as follows:

1. Identify convolution maps that can represent the dynamics to find linear koopman operator.

2. Modify to MGOC frameworks to find Koopman invariant subspaces and linear operator simultaneously.

3. Explore feasibility of genetic algorithm as a substitute for back propagation in MGOC frameworks.

1.4 Objectives and Scope of this Study

In chapter 2, we build accurate Markov models of complex nonlinear dynamics from limited data using feature maps whose layers are treated as dependent on each other and computed simultaneously by solving a 'global' or 'non-local' optimization problem such that the overall learning objective is realized. The aim then is to compare and contrast these approaches with ideas the employ local optimization. To this

end, we focus on the ability of a narrow class of SVD-based multi-layer convolution Markov models to capture nonlinear dynamics using globally or non-locally optimized features that we term as *globally optimal convolution (GOC) models* as against *locally optimal convolution (LOC)* . A popular example of multilayer GOC Markov models are feed forward neural networks (FFNN), a robust approach for learning the embedded nonlinearity in the dynamics from data. While shallow NN are known to possess universal function approximation properties (Hornik et al. 1989), it usually requires exponentially large number of neurons (features) for accurate prediction. To overcome this, deep neural networks (DNNs) offer a low-dimensional (short) and layered (deep) alternative for high (almost exponential) representational capacity of complex data. This low-dimensional feature space also helps reduce overfitting in a relative sense. In Chapter 3 and 4, we will present MGOC based modifications of neural networks that are symmetric networks which can learn Koopman subspace and linear operator. Deep Koopman Network (DKN) and Autoencoder Networks as way of learning the $\boldsymbol{g}$ and the Koopman operator for prediction and analysis of the flow. In doing so, we will highlight the important observations. In Chapter 5, we will present an alternate way to optimize the weights via Genetic algorithm (GA). Here the primary goal is to explore possible ways to incorporate derivative free activation functions (physics based) that can help learn the model. Further, to assess impact of the global vs local error used in the learning process. Finally in chapter 6, we will outline the key messages and important conclusions obtained from this study, further we will provide the future directions.

The contribution of this thesis is to explore feed forward DNN-like globally optimal multilayer convolution (multilayer GOC or MGOC) as an alternative to locally optimal multilayer convolution (multilayer LOC or MLOC) approaches. We focus our assessment on the following popular model architectures:

1. a 4 - level multi-layer LOC framework that mimics dynamic mode decomposition

(DMD);

2. a 6 - level multi-layer LOC with a nonlinear mapping that mimics extended DMD (EDMD);

3. a multi-level multi-layer GOC including nonlinear transfer functions that mimic a feed forward neural network (FFNN).

   (a) 6 - level multi-layer GOC (feed forward neural network)

   (b) 6 - level multi-layer symmetric markov GOC (Modified feed forward neural network)

   (c) Two step 9, 11 - level multi-layer symmetric markov GOC + LOC (Auto-Encoder network)

   (d) 8, 10 - level multi-layer symmetric markov GOC (Deep Koopman network)

   (e) 6 - level multi-layer symmetric markov GOC (Modified feed forward neural network) via derivative free optimization

In all of the above models, the convolution maps are carefully chosen so as to minimize variability so that we can focus purely on the effect of the *local versus global optimization* on the learning of the dynamics. Proper orthogonal decomposition (POD) is used as the first layer in all the above architectures so that we can operate in a low-dimensional feature space.

Through this work, we will show that MGOC models are accurate even with very limited input data as compared to MLOC models with both similar and dissimilar architectures for modeling dynamically evolving fluid flows. We will illustrate that the success of the MGOC architectures can be attributed to extending the dimension of the learning parameters and learning them concurrently using nonlinear optimization. This helps improve robustness and accuracy of the resulting predictions over short and long times as long as sufficient quality data is available. From the analysis

in the following sections, we show that MLOC models also benefit from the above design, but behave similar to two-layer shallow learning architectures requiring high-dimensional intermediate layers with slower convergence to the accurate result. On the other hand, MGOC being a 'deep learning' architecture produces more efficient learning. These ideas will be illustrated using different flow case studies including transient dynamical evolution of a cylinder wake towards a limit-cycle attractor and a transient buoyancy-driven mixing layer.

1.5 Fluid Flows and Data Generation

To assess the different modeling architectures and the learning algorithms, we build a database of snapshots of transient flow field data generated from high fidelity CFD simulations of a bluff body wake flow and also buoyancy-driven mixing layer. Both these flows are transient in their own way. The cylinder wake flow evolves on a stable attractor and approaches limit-cycle behavior rather quickly while the buoyancy-driven flow is a transient mixing problem with dynamics that dies out in the long-time limit. The former is an example of data-rich situation where the training data requirement to predict the dynamics is limited. On the other hand, the latter represents a data-sparse situation where the training data may not be sufficient to predict the future evolution. We explore the performance of the MLOC and MGOC architectures for both these situations with different combinations of training and prediction regimes. In the following section, we detail the data generation process for both these flows.

1.5.1 Transient Wake Flow of a Cylinder

Studies of cylinder wakes Roshko (1954), Williamson (1989), Noack et al. (2003), Rowley & Dawson (2017) have attracted considerable interest from the flow system learning community for its particularly rich flow physics content, encompassing many

of the complexities of nonlinear dynamical systems, while easy to simulate accurately on the computer using established CFD tools. In this exploration into the performance of different data-driven modeling frameworks we leverage both the unstable transient (evolution towards a limit cycle) and the stable limit-cycle dynamics of two-dimensional cylinder wake flow at two Reynolds numbers, $Re = 100, 1000$. To generate two-dimensional cylinder flow data, we adopt the spectral Galerkin method Cantwell et al. (2015) to solve incompressible Naiver-Stokes equations, as shown in Eq. (1.1) below:

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0, \tag{1.1a}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial P}{\partial x} + \nu\nabla^2 u, \tag{1.1b}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial P}{\partial y} + \nu\nabla^2 v, \tag{1.1c}$$

where u and v are horizontal and vertical velocity components. P is the pressure field, and ν is the fluid viscosity. The rectangular domain used for this flow problem is $-25D < x < 45D$ and $-20D < y < 20D$, where D is the diameter of the cylinder. For the purposes of this study, a reduced domain, i.e., $-2D < x < 10D$ and $-3D < y < 3D$, is used. The mesh was designed to sufficiently resolve the thin shear layers near the surface of the cylinder and transit wake physics downstream. For the case of $Re = 100$ the grid includes $24,000$ points whereas for $Re = 1000$ the grid is refined to include approximately $95,000$ points for the sampled flow region. The computational method employed is fourth order spectral expansions within each element in each direction. Each data snapshot output was sampled at $\Delta t = 0.2$ non-dimensional time units.

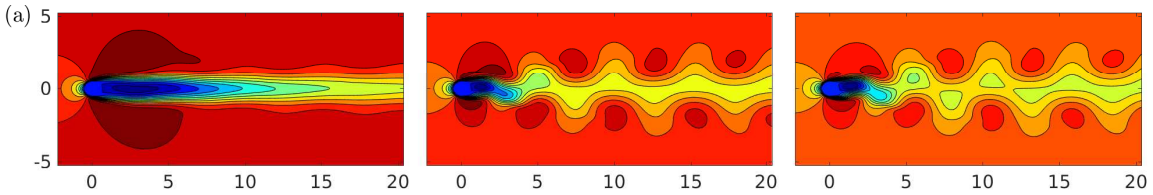


Figure 1.3: Velocity field snapshots of $Re100$ flow field.

1.5.2 2D Buoyant Boussinesq Mixing Flow

The above discussion pertains to a nonlinear wake flow dynamical system that transitions from a steady wake into a stable limit-cycle attractor. Such systems have seen success in predcition from data-driven models with the availability of limited data as demonstrated in Lu et al. (2018) . The instability-driven Bousinesq buoyant mixing flow Weinan & Shu (1998), Liu et al. (2003) exhibits strong shear and Kelvin-Helmholtz instability driven by thermal gradients. The convective dynamics in such a system cannot be compactly represented by POD modes. Further, the data-driven basis representing the low-dimensional manifold itself evolves temporally indicative of highly transient and data-sparse system. Such systems are sensitive to noise in the initial state that produce very different trajectories and conseuqently, a very different dynamical system with its own basis space. This renders such dynamical systems hard to predict even if one were to leverage equation-driven models such as POD-Galerkin Noack et al. (2003). In the corresponding author's earlier work Lu et al. (2018) it was shown that such problems are hard to predict accurately using MLOC methods. In this work, we explore this case to determine if MGOC methods can perform better.

The data set is obtained by modeling the dimensionless form of the two-dimensional incompressible flow transport equations Liu et al. (2003) augmented with buoyancy terms and thermal transport equations, as shown in Eq. 1.2 on a rectangular domain that is $0 < x < 8$ and $0 < y < 1$. To achieve this,we use a 6^{th}-order compact scheme Lele (1992) in space and 4^{th}-order Runge-Kutta method for the time-integration Got-

tlieb et al. (2001).

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0, \tag{1.2a}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial P}{\partial x} + \frac{1}{Re}\nabla^2 u, \tag{1.2b}$$

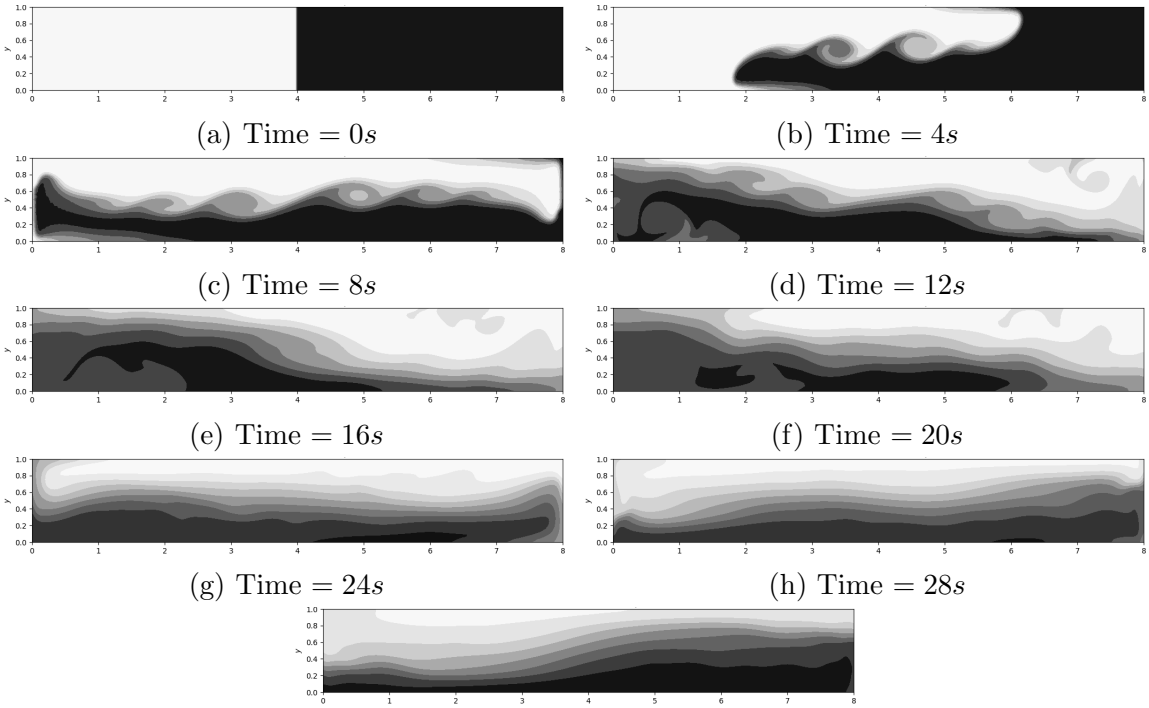
$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial P}{\partial y} + \frac{1}{Re}\nabla^2 v + Ri\theta, \tag{1.2c}$$

$$\frac{\partial \theta}{\partial t} + u\frac{\partial \theta}{\partial x} + v\frac{\partial \theta}{\partial y} = \frac{1}{RePr}\nabla^2 \theta, \tag{1.2d}$$

$$\tag{1.2e}$$

In the above system, u, v, and θ represent the horizontal, vertical velocity, and temperature field, respectively. The system is characterized by the following dimensionless parameters: Reynolds number, Re, Richardson number Ri, and Prandtl number, Pr with values of 1000, 4.0, and 1.0 respectively. The grid resolution employed is 256×33. The initial condition for the simulation is designed by vertically segregating the fluids at two different temperatures (uniformly distributed) at the middle of the domain. All the boundaries are adiabatic and friction generating walls. The thermal field evolution over the simulation duration of 32 seconds as shown in Fig. 1.4 illustrates the highly transient dynamics. To represent the system in a low-dimensional feature space, POD modes were computed from the entire 1600 snapshots.

The reduced feature set consisting of three POD features (capturing nearly 80% of the total energy) representing a low resolution measurement is shown in Fig. 2.8 is used to train the model and predict the trajectory.



(a) Time = $0s$ (b) Time = $4s$

(c) Time = $8s$ (d) Time = $12s$

(e) Time = $16s$ (f) Time = $20s$

(g) Time = $24s$ (h) Time = $28s$

(i) Time = $32s$

Figure 1.4: Time evolution of the isocontours of the temperature field in the 2D buoyant Boussinesq mixing layer is shown over a 32 *sec* time period.

CHAPTER 2

Linear vs. Nonlinear Modeling for Prediction

2.1 Introduction

A good data-driven model should perform well in both system identification and prediction using limited amounts of data. In addition, these models need to be computationally tractable which makes dimensionality reduction essential. System identification enables learning of stability and flow characteristics such as unstable modes and coherent structures for the purposes of understanding the flow. For example, proper orthogonal decomposition (POD) Lumley (2007) via singular value decomposition (SVD) Trefethen & Bau III (1997) and its close cousin, the Dynamic mode decomposition (DMD) Schmid (2010) are well known methods to extract such relevant spectral information. However, the capacity of DMD for long-term prediction is underwhelming Lu & Jayaraman (2017). POD-based methods that use Galerkin projection onto the flow governing equations are more successful, but require knowledge of the system. In this study, we focus on purely data-driven scenarios without knowledge of governing equations. By long-time predictions, we imply evolving the system model over multiple characteristic time-scales beyond the training regime. The other type is to employ the model to predict different system trajectories which is commonly referred using the generic term *forecasting*. Obviously, the precise definition of the 'long-time' prediction or forecasting depends on the flow physics of interest. For example, a limit-cycle system evolving on a stable attractor will be more amenable to forecasting from limited data as compared to more complex nonlinear mixing dynamics. In the case of cylinder wake flow explored in this study, forecasting represent

predicting the limit cycle Berkooz et al. (1993), Noack et al. (2003) dynamics using limited data in the transient region. We explore such cases as they are sensitive to error growth and hence, used to evaluate a given model. Errors in model learning can be attributed to limited training data, measurement noise, model overfitting and insufficient validation Bishop et al. (1995), Christopher (2016). The contributions from this paper are highly relevant within this context as we systematically explore and assess how nonlinear regression-based data-driven models perform relative to commonly used linear regression-based models for dynamically evolving fluid flows.

As a first step in this chapter, we build accurate Markov models of complex nonlinear dynamics from limited data using feature maps whose layers are treated as dependent on each other and computed simultaneously by solving a 'global' or 'nonlocal' optimization problem such that the overall learning objective is realized. The aim then is to compare and contrast these approaches with ideas the employ local optimization. To this end, we focus on the ability of a narrow class of SVD-based multi-layer convolution Markov models to capture nonlinear dynamics using globally or non-locally optimized features that we term as *globally optimal convolution (GOC) models* as against *locally optimal convolution (LOC)* . A popular example of multilayer GOC Markov models are feed forward neural networks (FFNN), a robust approach for learning the embedded nonlinearity in the dynamics from data. While shallow NN are known to possess universal function approximation properties Hornik et al. (1989), it usually requires exponentially large number of neurons (features) for accurate prediction. To overcome this, deep neural networks (DNNs) offer a low-dimensional (short) and layered (deep) alternative for high (almost exponential) representational capacity of complex data. This low-dimensional feature space also helps reduce overfitting in a relative sense. The contribution of this paper is to explore feed forward DNN-like globally optimal multilayer convolution (multilayer GOC or MGOC) as an alternative to locally optimal multilayer convolution (multi-

layer LOC or MLOC) approaches. We focus our assessment on three popular model architectures:

1. a 4 - level multi-layer LOC or 4-MLOC framework that mimics dynamic mode decomposition (DMD);

2. a 6 - level multi-layer LOC of 6-MLOC with a nonlinear mapping that mimics extended DMD (EDMD);

3. a 6 - level multi-layer GOC or 6-MGOC including nonlinear transfer functions that mimic a feed forward neural network (FFNN).

In all of the above models, the convolution maps are carefully chosen so as to minimize variability so that we can focus purely on the effect of the *local versus global optimization* on the learning of the dynamics. Proper orthogonal decomposition (POD) is used as the first layer in all the above architectures so that we can operate in a low-dimensional feature space.

Through this work, we will show that MGOC models are accurate even with very limited input data as compared to MLOC models with both similar and dissimilar architectures for modeling dynamically evolving fluid flows. We will illustrate that the success of the MGOC architectures can be attributed to extending the dimension of the learning parameters and learning them concurrently using nonlinear optimization. This helps improve robustness and accuracy of the resulting predictions over short and long times as long as sufficient quality data is available. From the analysis in the following sections, we show that MLOC models also benefit from the above design, but behave similar to two-layer shallow learning architectures requiring high-dimensional intermediate layers with slower convergence to the accurate result. On the other hand, MGOC being a 'deep learning' architecture produces more efficient learning. These ideas will be illustrated using different flow case studies including

transient dynamical evolution of a cylinder wake towards a limit-cycle attractor and a transient buoyancy-driven mixing layer. The organization of this paper is as follows.

In section 2.2 we present an overview of data-driven Markov models for transient dynamical systems and their connections to Koopman theoretic methods (section 2.2.1). In section 2.2.2, we introduce the concept of locally optimal convolution (MLOC) Markov modeling framework and describe the two variants that we consider in this study in subsections 2.2.2 and 2.2.1. In section 2.2.3 we introduce the feed forward DNN-like globally optimal convolution (MGOC) framework for building Markov models. The numerical examples and discussion of the modeling performance is presented in section 2.3 and the various outcomes are summarized with discussion in section 2.4.

2.2 Data-driven Markov Models for Transient Dynamical Systems

Extraction of high-fidelity Markov models from snapshot (time) data of nonlinear dynamical systems is a major need in science and engineering, where measurement data can be the only available piece of information. It is advantageous to learn the model in a low-dimensional feature space to both simplify the learning process and also improve efficiency. Most Markov models that are generally learned in the feature space, use linear operators/convolutions to take advantage of the powerful linear systems machinery for control Kim & Bewley (2007), optimization and spectral analysis Rowley et al. (2009). A Markov model, for given a current state $\boldsymbol{x}_t$ and future state $\boldsymbol{x}_{t+T}$ can be stated as $\boldsymbol{g}(\boldsymbol{x}_{t+T}) = \mathcal{K}\boldsymbol{h}(\boldsymbol{x}_t)$ where $\boldsymbol{g}$ and $\boldsymbol{h}$ are typically finite-dimensional transformations to the feature space and $\mathcal{K}$ represents the transition operator that can be linear or nonlinear. Without loss of generality, here we will use first order Markov process approximation of the dynamical system. That said, the algorithms presented here can easily be generalized to n^{th} order processes such as $\boldsymbol{g}(\boldsymbol{x}_{t+T}) = \mathcal{K}\boldsymbol{h}(\boldsymbol{x}_t, \boldsymbol{x}_{t-T}, \boldsymbol{x}_{t-2T}, \boldsymbol{x}_{t-3T}...\boldsymbol{x}_{t-(n-1)T})$. In section 2.2.1 we explore

the connections between the popular Koopman approximation-based methods and Markov linear models. Subsequently, in section 2.2.2, we introduce a class of locally optimal convolution (MLOC) based Markov models and draw similarities to existing popular models like DMD and EDMD. Finally, in section 2.2.3 we introduce the globally optimal convolution (MGOC) Markov models.

2.2.1 Koopman as Markov Linear Model

Linear, first order Markov models can be represented through the class of Koopman operator-theoretic methods Mezić (2005), Koopman (1931) as a framework for modeling nonlinear dynamics. In fact, a Markov process can approximate the Koopman operator Mezić (2005), Koopman (1931), Rowley et al. (2009) under certain conditions, namely, $\boldsymbol{g} = \boldsymbol{h}$ and $\mathcal{K}$ is linear. This representation is exact when $\boldsymbol{g}, \boldsymbol{h}$ and $\mathcal{K}$ are infinite-dimensional. Given a discrete-time dynamical fluid system that evolves as below:

$$\boldsymbol{y} = \boldsymbol{x}_{t+T} = \boldsymbol{F}(\boldsymbol{x}_t) = \boldsymbol{F}(\boldsymbol{x}) \tag{2.1}$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$ are N-dimensional state vectors, e.g., velocity components at discrete locations in a flow field at a current instant t, and separated by an appropriate unit of time T. To be precise, $\boldsymbol{x} \triangleq \boldsymbol{x}_t$ and $\boldsymbol{y} \triangleq \boldsymbol{x}_{t+T}$. Operator $\boldsymbol{F}$ evolves the dynamical system nonlinearly from $\boldsymbol{x}$ to $\boldsymbol{y}$, i.e. $\boldsymbol{F} : \mathcal{M} \to \mathcal{M}$. This representation can easily be made relevant to continuous time systems as well as in the limit $T \to 0$. A general linear Markov description of such a dynamical system is:

$$\boldsymbol{g}(\boldsymbol{y}) = \boldsymbol{g}(\boldsymbol{x}_{t+T}) = \mathcal{K}\boldsymbol{h}(\boldsymbol{x}). \tag{2.2}$$

Here, $\boldsymbol{g}(\boldsymbol{y})$ and $\boldsymbol{h}(\boldsymbol{y})$ are vector-valued transformations (components of $\boldsymbol{g}, \boldsymbol{h}$ are scalar-valued) to a feature space. In general, $\boldsymbol{g}, \boldsymbol{h} \in \mathcal{F}$ (where $\mathcal{F}$ is a function space) are infinite-dimensional, but approximated into a finite-dimensional vector in practice.

Consequently, the linear transition operator $\mathcal{K}$ is also a finite-dimensional approximation. In the Koopman framework, the feature space is the observable space and the feature maps are observable functions. The operator theoretic view interprets $\mathcal{K}$ as operating on the observable space Williams et al. (2015) $\mathcal{K} : \mathcal{F} \rightarrow \mathcal{F}$. When $\boldsymbol{g}$ and $\boldsymbol{h}$ are identical, then the linear operator $\mathcal{K}$ evolves the Markovian dynamics in Eq. (2.2) as a Koopman evolutionary model given by:

$$\mathcal{K}\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{y}) = g(\boldsymbol{F}(\boldsymbol{x})). \tag{2.3}$$

Since, the Koopman operator has the effect of operating on the functions of state space as shown in Eq. (4.1), it is commonly referred to as a composition operator where $\circ$ represents the composition between $\boldsymbol{g}$ and $\boldsymbol{F}$.

$$\mathcal{K}\boldsymbol{g} = \boldsymbol{g} \circ \boldsymbol{F}. \tag{2.4}$$

Being a linear operator, the products of Koopman spectral analysis such as the eigenfunctions (ϕ_j), eigenmodes (v_j) and eigenvalues (μ_j) can be leveraged to reconstruct the transformation $\boldsymbol{g}(\boldsymbol{x})$ as shown in Eqs. (4.3) - (4.4). This suggests that the transformation $\boldsymbol{g}$ should be spanned by Koopman eigenfunctions $\boldsymbol{\phi}$.

$$\boldsymbol{g}(\boldsymbol{x}) = \sum_{j=1}^{\infty} \phi_j \boldsymbol{v}_j \tag{2.5}$$

$$\boldsymbol{g}(\boldsymbol{y}) = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}) = \sum_{j=1}^{\infty} \phi_j \boldsymbol{v}_j \mu_j \tag{2.6}$$

In this discussion, we consider sequential snapshots of data separated by a constant time-step and denoted by $(\boldsymbol{x}^i, \boldsymbol{x}^{i+1} \dots)$ such that the dynamical system is given by:

$$\boldsymbol{x}^{i+1} = \boldsymbol{F}(\boldsymbol{x}^i) \tag{2.7}$$

The corresponding Koopman representation is:

$$\boldsymbol{g}(\boldsymbol{x}^{i+1}) = \boldsymbol{g}(\boldsymbol{F}(\boldsymbol{x}^{i})) = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}^{i}) \tag{2.8}$$

where, $\boldsymbol{g}$ is some convolution operator yet to be identified (or assumed for a given model). We can split this sequence of snapshots into pairs as $X = (\boldsymbol{x}^1 \dots \boldsymbol{x}^{M-2}, \boldsymbol{x}^{M-1})$ and $Y = (\boldsymbol{x}^2 \dots \boldsymbol{x}^{M-1}, \boldsymbol{x}^{M})$ such that $(X, Y) \in \mathbb{R}^{N \times M}$, then the dynamical system in Eq. (4.5) can be represented as

$$Y = \boldsymbol{F}(X) \tag{2.9}$$

and its quasi-linear form is

$$Y \approx \boldsymbol{A}(X)X. \tag{2.10}$$

Here, $\boldsymbol{A}(X) \in \mathbb{R}^{N \times N}$ is the quasi-linear operator describing the evolution of the discrete dynamical system that we are trying to approximate from data. Where N, M represent the dimensions of the instantaneous system state and the number of available snapshots respectively. Typically, $N \gg M$ and for a nonlinear system $\boldsymbol{A}(X)$ evolves with X. The observable function $\boldsymbol{g}$ is usually unknown and approximated with a finite-dimensional convolution map C that can have a functional or data-driven form. We define a convolution operation as a projection of the input state onto an appropriate basis such that the dynamics evolves in a feature space that is often low-dimensional. This would require $\boldsymbol{x}^i$ be spanned accurately by the choice of basis used to construct the columns of C. To build a finite-dimensional $\mathcal{K}$ in Eq. (4.6) we define a finite-dimensional convolution $C \in \mathbb{R}^{N \times K}$such that $\boldsymbol{g}(\boldsymbol{x}^i) \approx C\boldsymbol{x}^i$ and generalized as:

$$X = C\bar{X}, \tag{2.11}$$

$$Y = C\bar{Y}, \tag{2.12}$$

where $\bar{X} \in \mathbb{R}^{K \times M}$ and $\bar{Y} \in \mathbb{R}^{K \times M}$ are the corresponding features for X and Y in the

feature space, and K is the feature dimension. Note that C is truly nonlinear and should evolve with X as $C(X)$. Eq. (4.8) can be rewritten as:

$$\boldsymbol{A}(X)C\bar{X} = C\bar{Y}. \tag{2.13}$$

Rearranging Eq. (2.13) gives the relationship below with C^+, where $()^+$ is the Moore-Penrose pseudo-inverse

$$C^+\boldsymbol{A}(X)C\bar{X} = \bar{Y}. \tag{2.14}$$

Defining $\Theta \triangleq C^+\boldsymbol{A}(X)C$ as the linear finite-dimensional Koopman operator in the feature space we get:

$$\Theta\bar{X} = \bar{Y}. \tag{2.15}$$

The fidelity of the above approximation to the dynamical system in Eq. (4.5) depends on the choice of C as an approximation to observable $\boldsymbol{g}$. For Θ to be truly linear, C will have to evolve with the state X as $C(X)$ and $C^+(X)$ needs to exist. For detailed discussion on the architecture and choice of convolution maps we refer to Lu and Jayaraman Lu et al. (2018). Given C, X and Y, we can learn Θ by minimizing the frobenius norm of $\|\bar{Y} - \Theta\bar{X}\|_F$ via $\Theta = \bar{Y}\bar{X}^+$. In principle one could minimize the 2-norm $\|\bar{Y} - \Theta\bar{X}\|_2$ at the risk of added complexity, but the Frobenius norm serves an efficient alternative.

2.2.2 Markov Model using Multilayer Locally Optimal Convolution (MLOC)

The basis vector onto which the convolution projects the input state needs to be a function acting on the instantaneous state i.e., C should be $C(X)$ so that Θ can be linear. However, this often leads to a futile search for 'magic' functions when the nature of $\boldsymbol{A}(X)$ is not known. Alternative approaches Williams et al. (2015), Williams et al. (2014) include approximating the functional form of the convolution

map from data using a dictionary of basis functions. However, the dependence of C on the choice of functions populating the dictionary and the relative ease with which the feature dimension can grow, limits these approaches. Lu and Jayaraman Lu et al. (2018) propose an alternate approach of building complex and efficient convolution maps through layering of elementary operators based on the hypothesis that *deeper and shorter is better than taller and shallow* operators. This is also the key motivation behind the recent boom in *deep learning* ideas for artificial intelligence Bengio (2007). It is worth noting that both strategies increase the number of model parameters to be learned, but deep layering offers a sequential way to build model complexity and learn complex patterns much more efficiently than shallow architectures from limited data. A generalized way of building C is to layer recursively multiple convolution operators such as:

$$X = C_1 C_2 \bar{\bar{X}} = \mathcal{C}_{ML} \bar{\bar{X}}, \tag{2.16}$$

$$Y = C_1 C_2 \bar{\bar{Y}} = \mathcal{C}_{ML} \bar{\bar{Y}}, \tag{2.17}$$

where $\bar{\bar{X}}$ and $\bar{\bar{Y}}$ represent the features at the 2^{nd} layer and $\mathcal{C}_{ML}$ represents the multi-layer convolution operator, a schematic of such process is presented in Fig. 2.1. Where a state vector X is operated by two convolution operators C_1^+, C_2^+ respectively to yield $\bar{\bar{X}}$. Similarly, C_1^+, C_2^+ operate on Y, $\bar{Y}$ respectively to yield $\bar{\bar{Y}}$. An approximate linear Koopman operator Θ is then learned using $\bar{\bar{X}}$ and $\bar{\bar{Y}}$. Substituting Eq. (2.16) and (2.17) into Eq. (4.8), we have:

$$A\mathcal{C}_{ML} \bar{\bar{X}} = \mathcal{C}_{ML} \bar{\bar{Y}}. \tag{2.18}$$

Pre-multiplying the pseudoinverse of $\boldsymbol{\mathcal{C}}_{ML}$, we have:

$$\mathcal{C}_{ML}{}^{+} A \mathcal{C}_{ML} \bar{\bar{X}} = \Theta \bar{\bar{X}} = \bar{\bar{Y}}, \tag{2.19}$$

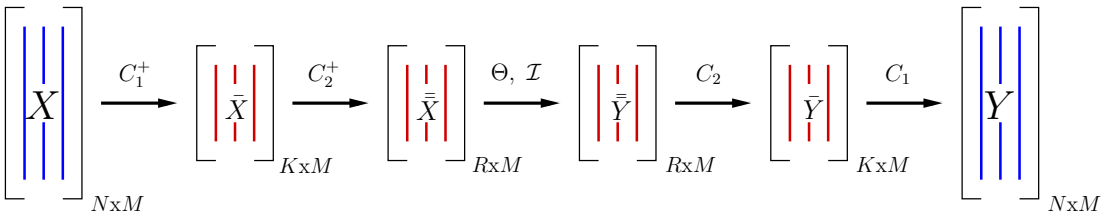


Figure 2.1: Schematic of a six-layer representation of the multilayer locally optimal convolution (MLOC) framework to approximate the Koopman operator. X, Y represent state space matrices and C_i^+, C_i, represent the convolution and reconstruction operations respectively. The arrows indicate direction of the convolution, i.e., C_1^+ acts on X to yield $\bar{X}$ and C_1 acts on $\bar{Y}$ to yield Y. Θ represents an approximate Koopman operator shown in Eq. (4.6). The size of data matrices in the high (X) and low dimensional($\bar{X}$ or $\bar{\bar{X}}$) space is also shown.

with Θ defined as:

$$\Theta = {\mathcal{C}_{ML}}^{+} A \mathcal{C}_{ML}. \tag{2.20}$$

In Eq. (2.20) the convolution can be generalized and consolidated into an effective map given by ${\mathcal{C}_{ML}}^{+} = C_L^+ ... C_1^+ C_2^+ C_3^+$ and $\mathcal{C}_{ML} = C_3 C_2 C_1 ... C_L$ to include sufficient number of elementary operators such that an optimal Θ is realized. L represents the number of layers in the design. The forward map or encoder ${\mathcal{C}_{ML}}^{+}$ can be computed as long as the elemental convolution maps, C_i, are invertible in a generalized sense. We note that although the multilayer formulation above is built from a Koopman approximation point of view, i.e. $\boldsymbol{g} = \boldsymbol{h} = \mathcal{C}_{ML}$, one can have a generalized Markov version of this model i.e. $\boldsymbol{g} = \mathcal{C}_{ML1}$ & $\boldsymbol{h} = \mathcal{C}_{ML2}$. Further, C_i and consequently $\mathcal{C}_{ML}$ are usually predetermined convolution maps (or functions) or computed using only the locally available information. Hence, we call this class of methods as *locally optimal convolution* or LOC for short. In the following subsections 2.2.2 and 2.2.1 we will view the popular Koopman approximation methods in the context LOC class of methods.

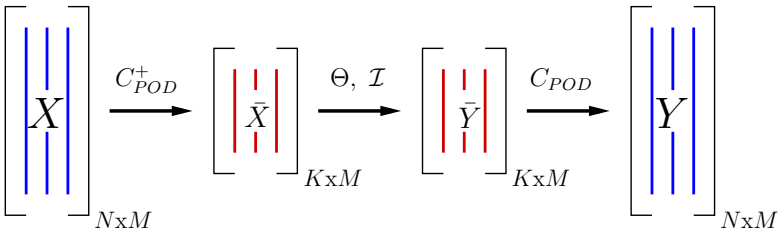


Figure 2.2: A four-level Koopman approximation LOC framework with linear maps.

A Four - level Multilayer Locally Optimal Convolution (4-MLOC) Map

There exist many methods to approximate the Koopman tuples including DMD Rowley & Dawson (2017), Schmid (2010), EDMD Williams et al. (2015) and its kernel variants Williams et al. (2014) and generalized Laplace analysis (GLA) Mezić (2005). DMD Schmid (2010) employs a linear identity map between the observable and input state spaces and hence, also represents a linear model in the original input space. The architecture for DMD is shown in Fig. 2.2 as a four-level architecture (4-MLOC) with a single POD convolution map obtained via SVD Trefethen & Bau III (1997) from snapshots of data (X). Given data snapshots separated in time X, Y as before, we can apply the convolution operator as:

$$X = C_{POD}\bar{X}; \quad Y = C_{POD}\bar{Y}. \tag{2.21}$$

Substituting the above in Eq. (2.19), we get the linear model below

$$\Theta\bar{X} = \bar{Y} \tag{2.22}$$

where $\bar{X}$ and $\bar{Y}$ are snapshots of POD coefficients evolving in time and represent the dynamics of the nonlinear fluid flow in the feature space. Θ is a finite dimensional approximation of the Koopman operator, a linear transition operator governing the

dynamics of the flow in the feature space. The pairs $\bar{X}$ and $\bar{Y}$ are setup as follows:

$$\bar{X} = \begin{bmatrix} a_1^1 & a_1^2 & \dots & a_1^{M-1} & a_1^M \\ a_2^1 & a_2^2 & \dots & a_2^{M-1} & a_2^M \\ . & . & \dots & . & . \\ . & . & \dots & . & . \\ a_K^1 & a_K^2 & \dots & a_K^{M-1} & a_K^M \end{bmatrix}_{(K \text{x} M)} \quad \text{and}$$

$$\bar{Y} = \begin{bmatrix} a_1^2 & a_1^3 & \dots & a_1^M & a_1^{M+1} \\ a_2^2 & a_2^3 & \dots & a_2^M & a_2^{M+1} \\ . & . & \dots & . & . \\ . & . & \dots & . & . \\ a_K^2 & a_K^3 & \dots & a_K^M & a_K^{M+1} \end{bmatrix}_{(K \text{x} M)} \tag{2.23}$$

Knowing Θ allows one to model the evolution of this dynamical system through the weights. Within the context of a multilayer framework, DMD can be viewed as a 4-level set-up (or 2−layer without the POD-convolution) with the first and last layers mapping in and out of the POD feature space respectively. Given that the convolution operator C_{POD} is independent of X, we can treat this model as a local optimization between the two layers in the middle as seen in Fig. 2.2. This local optimization problem tries to find an optimal mapping between all pairs of features $\boldsymbol{a}^t, \boldsymbol{a}^{t+1}$ which are column vectors in $\bar{X}, \bar{Y}$ as shown below,

$$\begin{aligned} \bar{Y} &= \begin{bmatrix} \boldsymbol{a}^2 & \boldsymbol{a}^3 & \dots & \boldsymbol{a}^{t+1} & \dots & \boldsymbol{a}^{M+1} \end{bmatrix} \\ &= \Theta \begin{bmatrix} \boldsymbol{a}^1 & \boldsymbol{a}^2 & \dots & \boldsymbol{a}^t & \dots & \boldsymbol{a}^M \end{bmatrix} \\ &= \Theta \bar{X} \end{aligned} \tag{2.24}$$

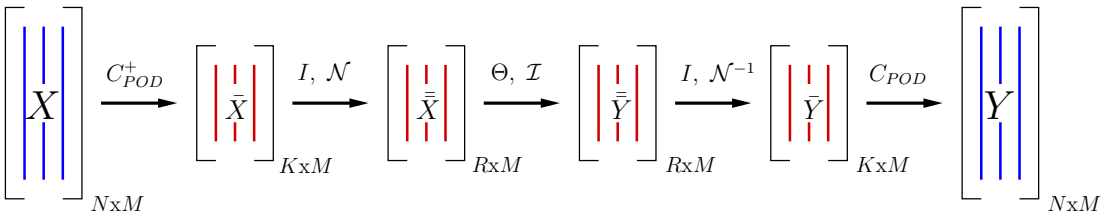


Figure 2.3: A six-level Koopman approximation LOC framework with nonlinear maps ($\mathcal{N}$, $\mathcal{N}^{-1}$) with I representing identity convolution operation.

such that $(\bar{Y} - \Theta\bar{X})$ Trefethen & Bau III (1997) is minimized. While there exists many approaches to solving this optimization problem, we minimize the Frobenius norm of $\|\bar{Y} - \Theta\bar{X}\|_F$ via,

$$\Theta = \bar{Y}\ (\bar{X} + \lambda I)^{+} \tag{2.25}$$

where $()^{+}$ denotes the generalized Moore-Penrose pseudo-inverse Trefethen & Bau III (1997) and λ is regularization Scholkopf & Smola (2001) parameter used to generate a unique solution from the pseudo-inverse and also avoid overfitting. In this 4−layer architecture, the convolution maps between any two layers are treated independently , i.e. C_{POD} does not depend on Θ which also implies that minimization of $\|\bar{Y} - \Theta\bar{X}\|_F$ is not the same as minimizing $\|Y - AX\|_F$. The equivalence between these two minimization problems depend on the choice of the convolution C.

A Six - level Multilayer Locally Optimal Convolution (6-MLOC) Map

Dynamic mode decomposition (DMD) being a linear model with a fixed POD features has difficulties predicting transient nonlinear fluid flows Rowley & Dawson (2017), Lu et al. (2018). Extensions to DMD such as EDMD Williams et al. (2015), Rowley & Dawson (2017) help alleviate this problem to some extent by introducing nonlinear convolution layer(s) as a wrapper to the POD-layer in DMD. It can also be viewed as combining POD convolution with a transfer function (TF) that creates an extended polynomial basis up to a desired order. In this way, it is a multilayer convolution

framework with nonlinear transfer functions as shown Fig. 2.3. This architecture can be viewed as a 6−level framework (4− layer without the POD-convolution) or 6-MLOC with the first and fifth pair of layers representing a POD-convolution while the 2^{nd} and 4^{th} pair of layers representing the nonlinear feature maps. Because, the architecture represented in Fig. 2.3 has a 'forward' depiction, the nonlinear mapping in the 4^{th} layer is denoted with an inverse function $\mathcal{N}^{-1}$. It is worth noting that in practice, $\mathcal{N}^{-1}$ may not be well behaved and hence the backward operation is preferred. This is another way of representing Koopman approximation methods which are intrinsically forward-backward maps. The linear Koopman operator is then approximated through a least squares minimization problem, $\|\bar{\bar{Y}} - \Theta\bar{\bar{X}}\|_F$ solved in a similar way as Eqn. (2.25).

In this study, we present two variants of this method, namely 6-MLOC-P (EDMD-P) Williams et al. (2015) which uses polynomial basis dictionary for incorporating nonlinearity into the convolution map and 6-MLOC-TS (EDMD-TS) which uses tan-sigmoid as a nonlinear feature map. An illustrative representation of the 6-MLOC-P (EDMD-P) with 2^{nd} order polynomial is presented in Fig.2.5(a), where $\boldsymbol{a}$ and $\bar{\boldsymbol{a}}$ represent the columns of $\bar{X}, \bar{\bar{X}}$ and are related as

$$\bar{\boldsymbol{a}}^i = \mathcal{N}(\boldsymbol{a}^i) \tag{2.26}$$

Here $\mathcal{N}$ is the nonlinear operator representing both polynomial generation for 6-MLOC-P (2^{nd} order or higher) and tansigmoid function evaluation for 6-MLOC-TS (EDMD-TS) as the choice may be. A mathematical representation of polynomial with $P = 2$ and tansigmoid operators are presented in Eqns. (2.27) & (2.28),

$$\bar{\boldsymbol{a}} = \mathcal{N}(\boldsymbol{a}) \quad = \quad \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{a} \otimes \boldsymbol{a} \end{bmatrix} \tag{2.27}$$

$$\bar{\boldsymbol{a}} = \mathcal{N}(\boldsymbol{a}) \quad = \quad \tanh \boldsymbol{a} \tag{2.28}$$

where $\bar{\boldsymbol{a}}$ represent the features in the $\mathcal{N}$ space. It can be easily seen from above that MLOC-P (EDMD-P) leads to a quadratic growth in the dimension of the features for 2^{nd} order nonlinearity. This will be even worse for higher order polynomial basis. On the other hand, MLOC-TS (EDMD-TS) does not lead to increase in the number of features. A key aspect of both the DMD and EDMD-type architectures is that the different convolution operators (layers) are independent of each other, i.e. C_{POD}, $\mathcal{N}$ and Θ do not depend on each other and hence classified as 'local' operators. As a consequence of this layer-wise independence, we can perform both backward and forward convolutions which allows for learning the Koopman operator and allows one to solve for Θ directly (non-iteratively). In the following section, we will focus on globally optimal convolution (GOC) operators with dependent convolutions that forces one to solve for Θ collectively (globally).

2.2.3 Markov Model using Multilayer Globally Optimal Convolution (MGOC)

In principle, multilayer convolution increases the number of hyperparameters in the model. In the earlier discussion on layer-wise LOC models, we bypassed this additional complexity by learning the convolution map offline (i.e. learning C_{POD} from training data) or assumed a functional form of the convolution ($\mathcal{N}$). To constrain the model to the data, we still had to solve the local optimization problem via Eq. (2.25) within a single-layer. While this approach makes MLOC methods efficient but they do not take advantage of the extended hyperparameter space for learning the system. Consequently, such methods work only for predicting select dynamics and fails to capture highly nonlinear transient dynamical systems. To truly take advantage of the extended hyperparameter space offered by the multilayer convolution framework, the convolution maps relating each layer need to be optimized for improved learning and prediction. Further, this optimization needs to be performed in a coordinated fashion so that a global objective can be minimized. This framework can be interpreted as a

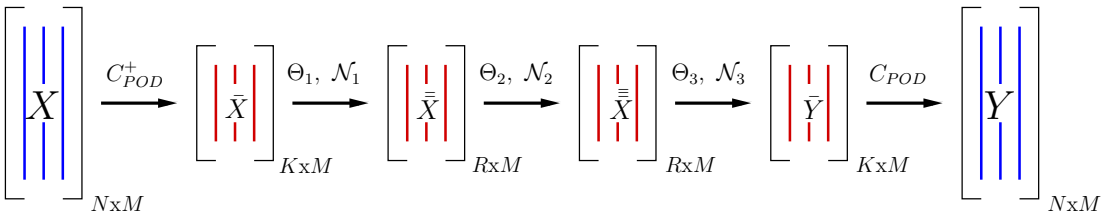


Figure 2.4: A six-level Multi layer GOC (6-MGOC) framework inspired from feed forward neural network architectures in machine learning and artificial intelligence, where Θ_l, $\mathcal{N}_l$ with arrow represents a convolution operation followed by a nonlinear mapping (see Eq. (2.32)) respectively.

multilayer globally optimal convolution (MGOC) based Markov model. It turns out that the MGOC architectures are like neural networks which allows us to leverage the various algorithmic advancements Bengio et al. (2015) . In this study we present a MGOC architecture inspired from feedforward neural networks (FFNN) as shown in Fig. 2.4. Fig. 2.5(b) presents this architecture as is commonly observed in the machine learning except for the absence of a bias term. We deliberately removed the bias term in order to facilitate better comparison with conventional MLOC architectures such as DMD and EDMD.

We chose a 6−layer architecture to compare with the 6-MLOC in section 2.2.1 so that a fair assessment of the different models can be made. Further, we retain the POD-based map as the first and last convolution operators so that the feature dimensions remain manageable for the fluid flow examples considered in this study. Each interior convolution map includes a linear map Θ_l and a nonlinear transfer function $\mathcal{N}_l$ that is predetermined for each version of the model. To mimic the 6-MLOC model exactly, one will need to set $\Theta_1 = \Theta_3 = I$, the identity tensor & $\mathcal{N}_1 = \mathcal{N}$, $\mathcal{N}_3 = \mathcal{N}^{-1}$ along with $\Theta_2 = \Theta$ & $\mathcal{N}_2 = \mathcal{I}$ as an identity map. For this FFNN-like MGOC architecture that only supports forward maps, building a convolution map with $\mathcal{N}^{-1}$ is not explored currently. This is because, for many common choices of $\mathcal{N}$, $\mathcal{N}^{-1}$ is not always bounded. It is for this reason, even in the

MLOC architectures, the backward operation is preferred. However, in general the various Θ_l are computed simultaneously by solving a coupled optimization problem. As before, $\bar{X}, \bar{Y}$ represent the time snapshots of POD weights of the data separated by the same unit of time as shown in Eq. (2.24) with columns made of $\boldsymbol{a}^t$ and $\boldsymbol{a}^{t+1}$ respectively. In this case, we predict $\bar{Y}$ using $\bar{X}$ as shown below:

$$\bar{\bar{X}} = \mathcal{N}_1(\Theta_1 \bar{X}), \tag{2.29}$$

$$\bar{\bar{\bar{X}}} = \mathcal{N}_2(\Theta_2 \bar{\bar{X}}), \tag{2.30}$$

$$\bar{Y}_p = \mathcal{N}_3(\Theta_3 \bar{\bar{\bar{X}}}). \tag{2.31}$$

In general, we have for a general multilayer network

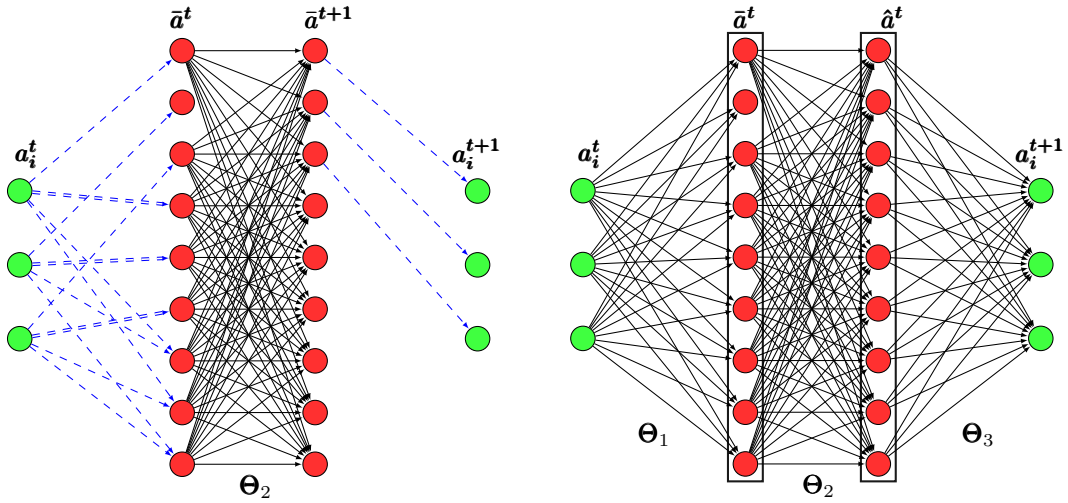
$$X_l = \mathcal{N}_l(\Theta_l X_{l-1}). \tag{2.32}$$

where $\bar{Y}_p$ is the predicted data and $X_l, \Theta_l, \mathcal{N}_l$ represents the mapped features, linear operator and nonlinear map relating the l^{th} and $l+1^{th}$ layers. The linear operator Θ_l, with $l = 1 \ldots (L-3)$, for a $L-$layer framework is computed by minimizing the overall cost function defined as:

$$\begin{aligned} \mathcal{J}(\Theta) = & \underbrace{\frac{1}{2M} \sum_{i=1}^{N} \sum_{j=1}^{M} (\bar{Y}_p(j,i) - \bar{Y}(j,i))^2}_{\text{Feed forward Cost}} \\ & + \underbrace{\left(\frac{\lambda}{2M} \sum_{l=1}^{L-3} \sum_{s=1}^{S} \sum_{q=1}^{Q} (\Theta_l(s,q))^2 \right)}_{\text{Regularization term}} \end{aligned} \tag{2.33}$$

where $\bar{Y}$ is the original data, S, Q represent the dimension of the features in layers l and $l+1$ respectively. In this architecture, we number the layers of features, l from $0 \ldots L-2$. For example, in the 6-MGOC network shown in Fig. 2.4 we have

$L = 6$ including the first ($l = 0$) and last ($l = 5$) POD convolution and deconvolution layers. This architecture will consist of $\Theta_l,\ l$ from $0 \ldots 4$ with $\Theta_0 = C^{+}_{POD}$ and $\Theta_4 = C_{POD}$ while Θ_1, Θ_2 and Θ_3 are computed simultaneously. The optimal solution for Θ_l is obtained using backpropagation algorithm with a gradient descent framework employing a Polack-Ribiere conjugate gradient algorithm Golub & Van Loan (2012) that employs a Wolfe-Powell stopping criteria. The use of back propagation to find Θ's is the most important distinction between MLOC and MGOC methods. This gradient descent framework requires $\mathcal{N}_l$ to be infinitely differentiable which is not always guaranteed when choosing $\mathcal{N}_l = \mathcal{N}^{-1}$. Consequently, we choose the nonlinear functions $\mathcal{N}_l$ in 6-MGOC as $\mathcal{N}_{1,2} = \mathcal{N}$ and $\mathcal{N}_{0,3,4} = \mathcal{I}$. To avoid overfitting in Eq. (2.33), we use $\mathcal{L}_2$ norm based regularization while computing Θ's, with λ as the tuning parameter similar to that used in the MLOC architectures. In the case of learning linear operators, this l_2 penalty boils down to the well known Tikhonov regularization. For assessment purposes, we use two intermediate layers in the MGOC (when learning the dynamics between $\bar{X}$ and $\bar{Y}$) as illustrated in figure 2.5 to resemble the construct of 6-MLOC (EDMD). Standard FFNN employs a bias term in Eq. (2.29), but is not considered here for this comparison study. To determine the dimension of the intermediate layer features in MGOC we use a factor (N_f) that is multiplied with the input feature dimension, i.e., $S, Q = N_f \times$ input feature dimension. The key distinction between MLOC and the FFNN-like MGOC is the lack of a forward-backward mapping in the latter which does not support the learning of the Koopman operator. To accomplish this, the MGOC framework (FFNN) needs to be modified with feedback loops that have similarities to RNNs. Such a modified architecture is presented in Puligilla & Jayaraman (2018*a*).



(a) A six-level MLOC (EDMD) with $P = 2$ (b) A six-level MGOC (FFNN) with $N_f = 3$

Figure 2.5: A representative comparison of architectures (a)6-MLOC-P (EDMD-P) and (b)6-MGOC (FFNN) methods which use local and non-local optimization, respectively

2.3 Numerical Experiments and Discussion

In this section, we compare the predictive capabilities of MLOC framework that uses local optimization with FFNN-like MGOC frameworks based on global optimization. Our hypothesis is that learning an extended hyperparameter space by minimizing the training error-based cost function allows for improved predictions of time-series flow data. Consistent with the earlier sections, we adopt the nomenclature 'L-Method-$\mathcal{N}M$' to denote the different architectures and their respective parameters, where the 'L' represents the total number of layers used to map from state space to state space ($X \to Y$), $\mathcal{N}$ defines the intermediate (nonlinear) convolution and M represents any specific parameters that supplement $\mathcal{N}$. For example, we describe a 6-level multilayer-locally optimal convolution (EDMD) with a polynomial convolution of 2^{nd} order as 6-MLOC-P2, while a 6-level MLOC (EDMD) with a tansigmoid function is described as 6-MLOC-TS1 where the number followed by TS represents the feature growth factor (N_f). A 4-level MLOC representing the DMD architecture is denoted as 4-MLOC-$\mathcal{I}1$, where $\mathcal{I}$ defines identity mapping and $M = 1$ defines the feature growth factor.

In this study, we have used four MGOC architectures with different feature growth factors so as to compare and assess the abilities of the GOC framework over their LOC counterparts. The MGOC methods used are 6-MGOC-TS with $N_f = 1, 3, 9, 20$. The various model possibilities are delineated in section 2.3.2. Section 2.3.1 details the generation of flow data from high fidelity computations for use in this study, namely the cylinder wake flow (Sec. 2.3.1) and the buoyancy-driven mixing flow (Sec. 2.3.1).

2.3.1 Experiments

Transient Wake Flow of a Cylinder

With the velocity data arranged as described in section 2.2.1, a SVD of state vectors was performed to obtain POD coefficients and their respective modes. The most dominant POD coefficients corresponds to eigenvalues at $St = 0.16$ and 0.23 for $Re = 100$ and 1000, from which we have deduced the number of data points that encompass one limit cycle. Each limit cycle contains approximately 31 and 21 data points of POD coefficients. In this study we have used the convention of cycles to specify the training region, so as to provide a better physical representation to the training data used, while this also helps in making observations on long time predictions (cycles) based on limited number of data(cycles) provided.

Although, a minimum > 15 POD modes are required for nearly 100% energy reconstruction of cylinder flows at $Re = 100$ and 1000, the large scale coherent structures which govern the flow dynamics are represented by the first 3 modes and account for approximately 95% and 90% energy respectively, see Figs. 2.6(a) and 2.7(a). The eigenfunctions corresponding to these three modes are presented in Figs.2.6(b) and 2.7(b) respectively and show qualitatively similar flow structures for both $Re = 100$ and $Re = 1000$. In Figs. 2.6(c) and 2.7(c) we show the phase portrait for the flow dynamical system, wherein the flow transitions from a steady wake through an instability and settles on a limit cycle attractor.

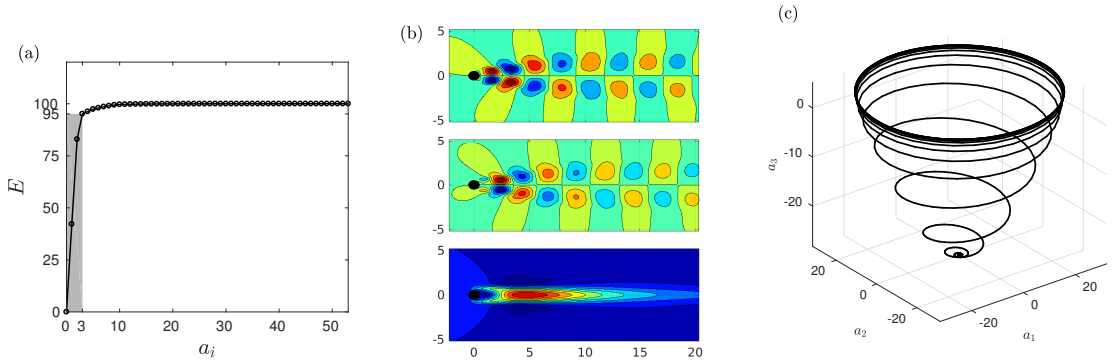


Figure 2.6: Energy content in POD features selected (a) 3 coefficients (b) eigen modes/functions corresponding to 3 POD features (c) phase portrait of $Re = 100$ flow.

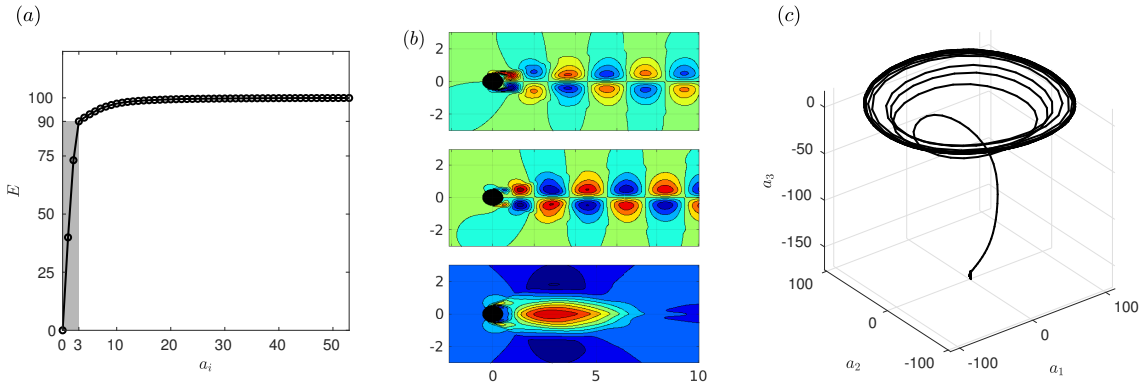


Figure 2.7: Energy content in POD features selected (a) 3 coefficients (b)eigen modes/functions corresponding to 3 POD features (c) phase portrait of $Re = 1000$ flow.

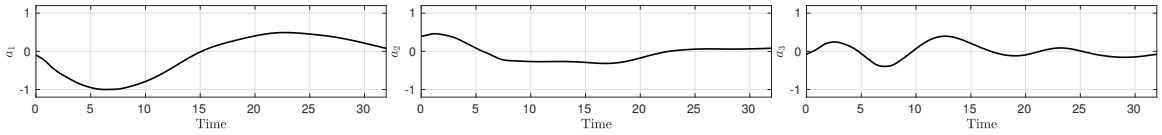


Figure 2.8: Time evolution of the POD weights, a_i^t for the buoyant mixing flow.

2D Buoyant Boussinesq Mixing Flow

Similar to the previous section, a low-dimensional feature space of 2D Buoyant Boussinesq Mixing Flow are obtained, POD modes were computed from the entire 1600 snapshots.

The reduced feature set consisting of three POD features (capturing nearly 80% of the total energy) representing a low resolution measurement is shown in Fig. 2.8 is used to train the model and predict the trajectory.

2.3.2 Analysis Framework

In this section, we summarize the different candidate model architectures and the corresponding learning algorithms. Table 2.1, lists the different MLOC architectures and their corresponding MGOC architectures with the total number of learning parameters ($\mathcal{LP}$) used. The first sub-column under each local (LOC) and globally (GOC) architecture in table 2.1 represents the choice of nonlinear part of the convolution map and the second sub-column represents the different layers with the corresponding feature dimension. For readability and conciseness, we have excluded the first and last layers corresponding to the input and output state vectors acted on by a POD-convolution. The rightmost column represents the total number of learning parameters ($\mathcal{LP}$) in a model. For example, when using 6-MLOC-P2 in table 2.1 we learn an operator (Θ) with 81(9x9) parameters to model the flow and similarly, using 6-MGOC-TS for TS3 we learn operators($[\Theta_1, \Theta_3, \Theta_3]$) totaling $135(27 + 81 + 27)$ parameters.

The six-level 6-MLOC-P2 (quadratic polynomial features) method produces 9 fea-

	Local optimization		$\mathcal{LP}$	Global optimization		$\mathcal{LP}$
1	4-MLOC-$\mathcal{I}$			6-MGOC-$\mathcal{I}$		
	$\mathcal{I}1$	3-3	9	$\mathcal{I}1$	3-3-3-3	27
2	6-MLOC-TS			6-MGOC-TS		
	TS1	3-3-3-3	9	TS1	3-3-3-3	27
3	6-MLOC-P			6-MGOC-TS		
	P2	3-9-9-3	81	TS3	3-9-9-3	135
	P7	3-125-125-3	15,625	TS9	3-27-27-3	891
				TS20	3-60-60-3	3960

Table 2.1: Overview of the methods used as part of this analysis. The dimensions of the different layers correspond to that used for cylinder wake flow model.

tures in the intermediate layer which is then used to learn a locally optimal mapping between the 9 features at the next intermediate layer followed by reverse map to the penultimate layer with 3 POD features. A similar construct can be observed in globally optimal framework using 6-MGOC-TS with $N_f = 3$. However, the 6-MLOC-P2 consists of 81 paramters while the 6-MGOC-TS has 135. In our following analysis of the various predictions, we find that using just 3 POD features with a 2nd order polynomial expansion in 6-MLOC-P2 does not produce accurate results. So, in addition to P2, we also explore higher order polynomial basis to ascertain if better predictions can be realized. In the following subsection, we describe the prediction and analysis methodology.

2.3.3 Prediction Framework and Error Metrics

The data generated from the computer simulations described above are separated into training and prediction regimes. The training data set is used for learning the optimal Θ's using which future time predictions are computed with input specified from the

previous time step alone to mimic a practical usage of the model. In this study , we assess model performance based on both qualitative representation of the dynamics and prediction errors. We quantify model errors using the $\mathcal{L}_2$ norm of the prediction error from the data-driven model relative to the truth using only the initial condition $\boldsymbol{a}_0$ specified. To bypass the complexities of computing the $2-$norm, the Forbenius norm of the error is computed instead as below.

$$\mathcal{E}_i = \frac{1}{2M_i} \left|\bar{Y}_p - \bar{Y}\right|_2^2 . \tag{2.34}$$

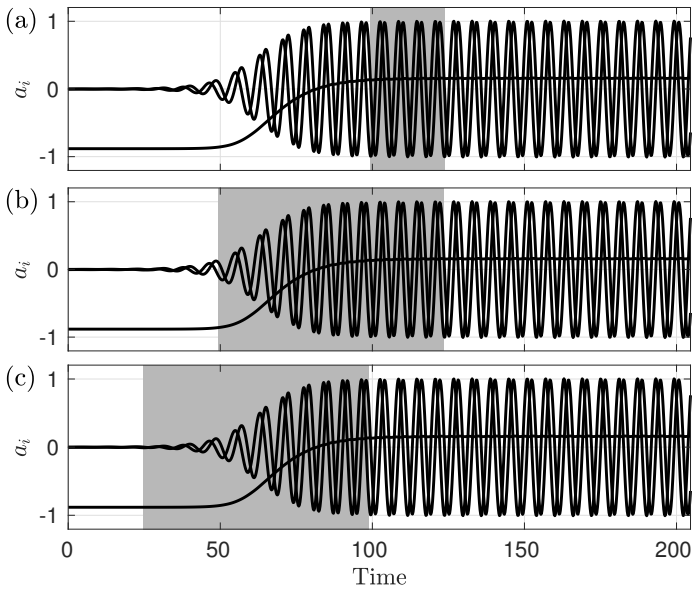
In the above equation $\bar{Y}_p$ represents the data-driven model predictions and $\bar{Y}$ the true data. We separately quantify the errors in the training region where the data-driven model is more of an interpolation and in the prediction region where model performs an extrapolation role. The prediction error in the training region is denoted by $\mathcal{E}_t$ and combined error in both the training and prediction regions is denoted by $\mathcal{E}_p$. As a way to assess the robustness of the different models discussed in table 2.1 we explore their ability to predict the flow dynamics given an initial condition in any part of the flow dynamics, for example, in transient or limit cycle regions for the cylinder wake flow. To this end, we designed three training regions (see Fig. 2.9a) corrresponding to three different time windows. In Figs. 2.9a and 2.9b we show these three training regions used for data with $Re = 100$ and $Re = 1000$ respectively, shaded in grey. The figures in row (a) represent the training region in the limit-cycle regime and rows (b) and (c) show two regions in the transition part of the dynamics and denoted by region I (TR-I) and region II (TR-II). A challenging test for a model is to learn the dynamics in the steady wake flow as shown in Figs. 2.9a and 2.9b and predict the instability growth that ultimately results the limit cycle. In our experience using training data that only contains information of the steady wake produces models that are highly unstable. Consequently, we designed two different training regions where the flow

transitions across flow regimes, but with different proportions of limit-cycle (vortex shedding) and steady wake content. In the following sections we will highlight and discuss the key results from our data-driven predictions.

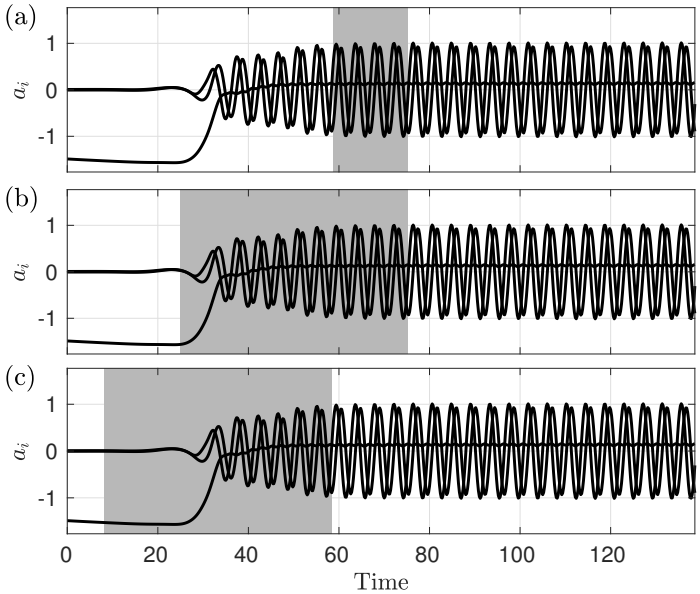
2.3.4 Learning and Predicting Limit-cycle Cylinder Wake Dynamics

The focus of this section is to learn from limit-cycle training data and predict the corresponding limit-cycle physics over long-times. Successful prediction of this case is considered a benchmark for data-driven models. The underlying theme in this article is to explore whether globally optimal learning of the model parameters ($\mathcal{LP}$) can outperform locally learnt model parameters for predictions. To verify this we compare the following four models namely: 4-MLOC-$\mathcal{I}1$ (DMD), 6-MGOC-$\mathcal{I}1$ (compares favorably with FFNN-linear), 6-MLOC-TS1 (EDMD-TS1) and 6-MGOC-TS1 (compares favorably with FFNN-TS). It is well known that 4-MLOC-$\mathcal{I}1$ or DMD performs well in the limit cycle region as shown in Lu et al. (2018), Rowley & Dawson (2017) and under performs in the strongly nonlinear transition regimes on account of being a linear model. In Fig. 2.10, the time series predictions of the first three POD features are shown. with rows $1-4$ (top-to-bottom) representing the prediction outcomes from the learned parameters (Θ) obtained using the 4-MLOC-$\mathcal{I}1$ (DMD), 6-MGOC-$\mathcal{I}1$ (FFNN-linear), 6-MLOC-TS1 (EDMD-TS1) and 6-MGOC-TS1 (FFNN-TS1) architectures respectively. It is worth noting that 4-MLOC-$\mathcal{I}1$ (DMD) and 6-MGOC-$\mathcal{I}1$ are a pair of local and global models with a linear transfer (convolution) function. In the same vein, 6-MLOC-TS1 (EDMD-TS1) and 6-MGOC-TS1 (FFNN-TS1) are a pair of linear and global models with a nonlinear tansigmoid transfer function. Specifically, we assess the role of local(LOC) versus global (GOC) optimization of the parameters as well as the impact of nonlinear mapping on model prediction.

The first major observation is that both the GOC as well as the LOC models with linear mapping predict the overall dynamics relatively accurately while the LOC



(a) Times series plot of the weights corresponding to the three most energetic POD modes with different training regions (a) Limit cycle $(16-20)$: 124 data points, (b) transient region-I $(8-20)$: 372 data points and (c) Transient region-II $(4-16)$: 372 data points, where each cycle consists of 31 data points.



(b) Times series plot of the weights corresponding to the three most energetic POD modes with different training regions (a) Limit cycle $(14-18)$: 84 data points, (b) transient region-I $(6-18)$: 252 data points and (c) Transient region-I $(2-14)$: 252 data points, where each cycle consists of 21 data points.

Figure 2.9: Schematic showing the different training regions chosen for prediction using the different models.

model with nonlinear sigmoid mapping damps the POD features over time. The second observation is that all the models show gradual error growth with time except the 6-MGOC-TS1 architecture which is closer to a FFNN-TS. The plots in Fig. 2.10 convey that a nonlinear mapping is not essential to capture the limit-cycle dynamics, but if used, should be carefully designed. For example, it was shown in Lu et al. (2018) that EDMD-P2 (6-MLOC-P2) can predict such dynamics very well while the current results (Fig. 2.10(c)) show that the same architecture with a tansigmoid function produces errors. The TS function is primarily used in machine learning for classification and has a *squashing* nature to it, i.e. it has the effect of compressing the features which explains its inability to predict the dynamics. This occurs in spite of using $\mathcal{N}^{-1}$ as the 'reverse squashing' map in the architecture as discussed in section 2.2.2. A plausible reason could be that the TS nonlinearity does not extend the space of learning parameters in contrast to polynomial basis. Nevertheless, when the TS nonlinearity (using the $\mathcal{N}$) is combined with a GOC framework such as in 6-MGOC-TS1, the prediction drastically improves as learning the mapping parameters in $\Theta_1, \Theta_2, \Theta_3$ simultaneously while applying the TS nonlinearity produces a compensatory and powerful outcome. Further, this FFNN-like 6-MGOC-TS1 model can predict over long times without growth in error as seen from the evolution of third POD feature (shift mode) in Fig. 2.10(d).

We had mentioned earlier that the success of the MGOC frameworks comes from learning an extended parameter ($\mathcal{LP}$) space, but the following discussion shows that this is true only in the presence of a nonlinear function as part of the mapping. In the DMD (4-MLOC-$\mathcal{I}1$) framework, there are 9 learning parameters in Θ to predict the limit cycle dynamics as compared to 27 parameters for 6-MGOC-$\mathcal{I}1$ architecture. However, in the absence of an nonlinear function in the convolution maps, the linear operator computed from the two methods are the same, i.e. the product of the different Θ_l for $l = 1 - 3$ in the 6-MGOC will trivially turn out to be the same as Θ

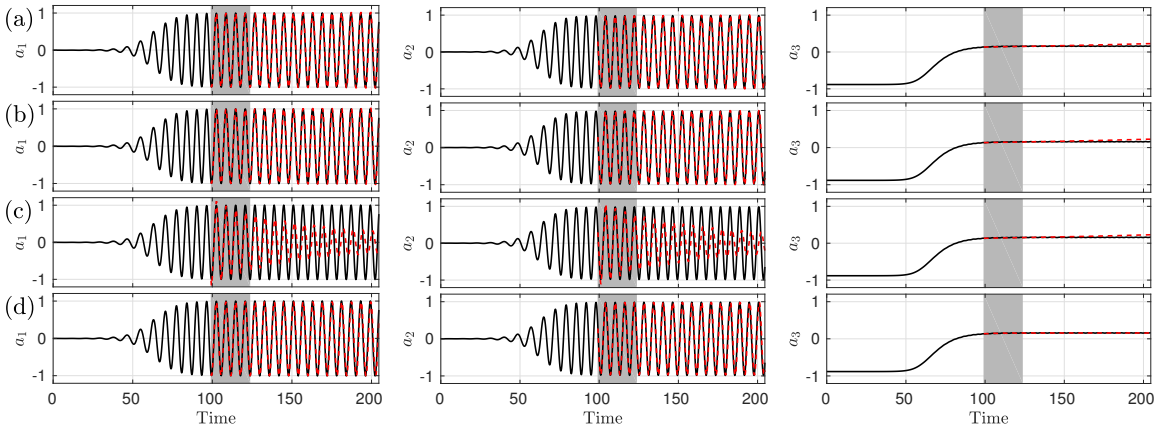


Figure 2.10: Times series of predicted POD features (– – –) obtained from (a) 4-MLOC-$\mathcal{I}1$, (b) 6-MGOC-$\mathcal{I}1$, (c) 6-MLOC-TS1 and (d) 6-MGOC-TS1 are plotted with their respective original data (——) in the limit cycle regime.

learned from the DMD (4-MLOC-$\mathcal{I}1$) framework. In Fig. 2.10, we use 4-cycles of (124 points) data in the limit cycle region for training and predict upto 17 cycles (527 data points). We see that the predictions obtained using 4-MLOC-$\mathcal{I}$ and 6-MGOC-$\mathcal{I}$ in Fig. 2.10(a) and (b) are similar as the same linear transition operator Θ is estimated. However, with limited training data, the predictions start to diverge from the truth over large times as is clearly seen from the evolution of the third POD feature, a_3.

While the addition of nonlinear functions in the convolution map aids the prediction of nonlinear dynamics, employing this formulation with a local optimization of the $\mathcal{LP}$ does not always guarantee good results. We see an illustration of this in the performance of the 6-MLOC-TS1 architecture as seen from Fig. 2.10(c), where all the three input features are incorrectly predicted in contrast to predictions by the 6-MGOC-TS1 in Fig. 2.10(d). The prediction error quantifications for the limit-cycle regime in the training and prediction regions are shown in the first two rows of the table 2.2. These show that the linear DMD (4-MLOC-$\mathcal{I}$) architecture and the FFNN-like 6-MGOC-TS architecture produce error magnitudes of 7.4×10^{-3} and 1.6×10^{-2} respectively outside the training region. However, these errors are higher than the $O(1e^{-4})$ errors in the training region as one would expect. In spite of being inaccurate in the prediction regime, the MGOC models do not allow for

growth in error which is a desirable feature. As additional benchmarks we also include prediction errors for other architectures including 6-MLOC-P2, 6-MGOC-TS1 and 6-MGOC-TS3 which generate comparable prediction accuracy with 6-MLOC-P2 being the smallest.In summary, except for the EDMD-TS (6-MLOC-TS1) all the other models display reasonable accuracy for this limit-cycle dynamics in both the training and prediction regimes. However, we observe a gradual growth of the third feature in all the models except the 6-MGOC-TS architectures which can impact long-time predictions. It is for this reason we consider the MGOC architectures to perform the best within this regime.

2.3.5 Learning and Prediction of a Transient Cylinder Wake Dynamics

In the earlier section, we highlighted the role of the choice of nonlinearity and the importance of combining this with a GOC framework for stable long-time predictions. In this section, we focus on learning from transient wake flow data and predict the resulting limit-cycle system. It is well known that DMD (4-MLOC-$\mathcal{I}$) performs better on limit cycle problems and under performs in the transient regime due to its inability to handle the enhanced nonlinearity of the underlying dynamical system. In particular, if the limit-cycle dynamics represents a nonlinearity of order k then the transient wake regime corresponds to a nonlinearity of order $\geq k+1$ Noack et al. (2003). Consequently, models that incorporate nonlinearity in the convolution maps such as the EDMD with polynomial basis Williams et al. (2015) (or equivalently the 6-MLOC-P as adopted in this article) or the corresponding kernel representation Williams et al. (2014) perform better for such problems, but only when using significant number of input features. In this section, we show that global optimization with nonlinear multilayer convolution provides much better learning and prediction capabilities from as little amount of input data as three features which the minimum amount of data one needs to capture the wake instability behind a cylinder Noack et al. (2003).

For this analysis, we used two training regions in the unstable transition regime, namely transient region-I (TR-I) and transient region-II (TR-II) as shown in Fig. 2.9a corresponding to $8-20$ and $4-16$ cycles respectively with both regions consisting of 372 data points.

Train		4-MLOC-$\mathcal{I}$	6-MLOC-TS	6-MLOC-P		6-MGOC-TS			
cycles	$\mathcal{E}$	$N_f = 1$	$N_f = 1$	p = 2	p =7	$N_f = 1$	$N_f = 3$	$N_f = 9$	$N_f = 20$
$16 - 20$(LC)	$\mathcal{E}_t$	$1.6e^{-4}$	$3.3e^{-2}$	$6.7e^{-5}$	$--$	$2.7e^{-4}$	$2.1e^{-4}$	$--$	$--$
	$\mathcal{E}_p$	$7.4e^{-3}$	0.269	$3.5e^{-4}$	$--$	$1.6e^{-2}$	$8.1e^{-3}$	$--$	$--$
$08 - 20$(TR-I)	$\mathcal{E}_t$	0.417	0.467	0.475	$6.4e^{-6}$	0.320	$3.7e^{-2}$	$1.9e^{-2}$	$2.1e^{-2}$
	$\mathcal{E}_p$	0.513	0.483	0.776	$3.9e^{-4}$	0.686	0.146	0.153	0.148
$04 - 16$(TR-II)	$\mathcal{E}_t$	0.246	0.238	0.223	0.191	$--$	0.106	0.182	0.385
	$\mathcal{E}_p$	0.551	0.530	0.683	0.977	$--$	0.883	0.948	0.720

Table 2.2: Prediction error estimates for layer-wise local (MLOC) and global optimization methods (MGOC) for $Re = 100$.

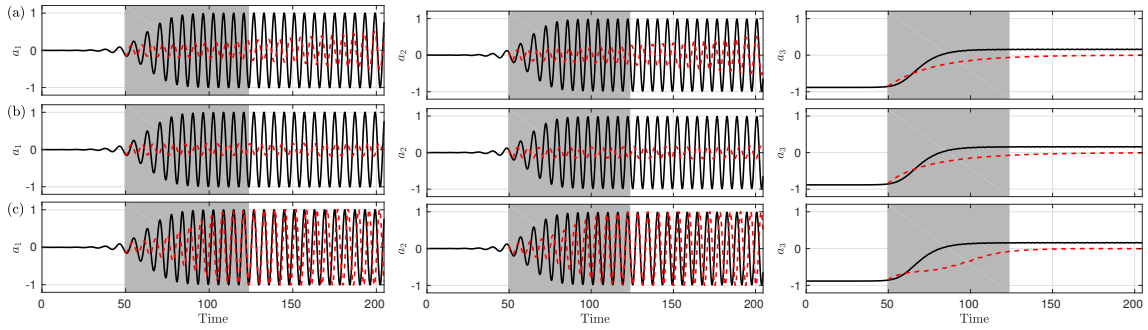


Figure 2.11: Times series of predicted POD features obtained from (a) 4-MLOC-$\mathcal{I}1$, (b) 6-MLOC-TS1 and (c) 6-MGOC-TS1 for TR-I as training region.

TR-I is relatively less challenging as almost all of the training data incorporates vortex shedding, but with an amplitude that is growing. In TR-II the first 30% of the training data includes a stable wake with symptoms of instability that grows in amplitude all through the regime. This has implications for predictions using machine learning models where the training data almost always determines what kind of dynamics the model can predict. If one were to rank the level of difficulty in predicting the resulting limit-cycle dynamics from different sets of training data then the most difficult would be TR-II followed by TR-I and lastly, the limit-cycle training data used in the previous section.

Figure 2.11 shows the predictions obtained from the different locally optimal 4-MLOC-$\mathcal{I}1$ (DMD), 6-MLOC-TS1 (EDMD-TS1) and the FFNN-like globally optimal 6-MGOC-TS1 methods for the TR-I training region. We can see that, all these methods fail to learn the nonlinear dynamics and predict the resulting limit-cycle system to varying levels of inaccuracy - MGOC being the closest. This can be attributed to the lack of sufficient nonlinearity in the models and insufficient learning parameters to represent the dynamics. It is worth pointing out that the EDMD-TS (6-MLOC-TS1) does not extend the $\mathcal{LP}$ space as against its polynomial basis variant, EDMD-P2 (6-MLOC-P2). Also, the choice of P2 basis is physics-driven to account for the quadratic nonlinearity of the POD features as embedded within the Navier-

Stokes equations that describe the flow. On the other hand, for the MGOC methods, a logical way to extend the $\mathcal{LP}$ space is to increase the number of features in the intermediate layers by increasing N_f. Consequently, we use 6-MLOC-P2 (EDMD-P2) as the baseline case and design a MGOC architecture with similar sized $\mathcal{LP}$ space with feature factor, $N_f = 3$. This approach of choosing N_f based on the dimension of the quadratic polynomial features is a logical way to design FFNN-like MGOC architectures as against more *ad hoc* choices. A schematic comparing the four intermediate layers in the 6-MLOC-P2 and the 6-MGOC-TS3 architectures is shown in Fig. 2.5. For 6-MLOC-P2, the three input POD features are mapped onto a polynomial basis space with nine features. In 6-MGOC-TS3, the three input features are mapped onto an unknown basis space, but guaranteed to be optimal for the chosen architecture. In this spirit of exploration, we also try a a 7^{th}-order polynomial feature map, i.e. a 6-MLOC-P7 and corresponding MGOC architectures with an expanded $\mathcal{LP}$ space ($N_f = 9$ and 20) to assess the effect of $\mathcal{LP}$ dimensionality on the predictions.

Figure 2.12 shows the predictions from 6-MLOC-P2 and 6-MGOC-TS3 using TR-I data. In spite of the embedded quadratic nonlinearity, the 6-MLOC-P2 framework fails to the predict the correct limit-cycle dynamics using just three input features. On the other hand, 6-MGOC-TS3 with a similar architecture with global optimization learns and predicts the flow dynamics more accurately. These prediction error trends are quantified in table 2.2. This is consistent with our earlier discussions that a larger $\mathcal{LP}$ dimension improves predictions as 6-MGOC-TS3 learns 135 parameters compared to 81 for the 6-MLOC-P2 case. Although these numbers are not vastly different, the 6-MLOC-P2 fails to even predict qualitatively accurate results. It is also worth noting that 6-MGOC-TS3 predicts the first two POD features accurately (see Fig. 2.12), but the third coefficient is biased towards a zero magnitude. We have found that this is due to the absence of bias term which when incorporated into the MGOC architectures corrects for this error as discussed and shown in Dig. 7.1

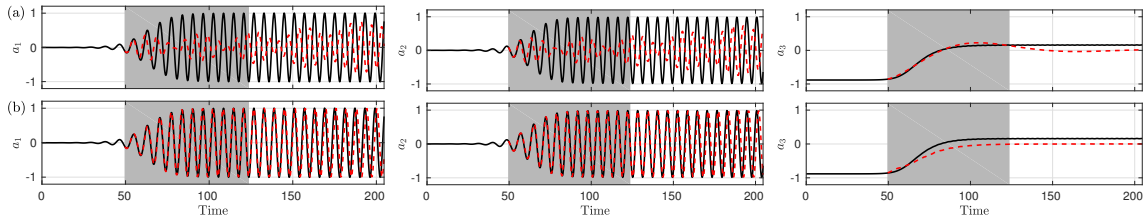


Figure 2.12: Times series of predicted POD features obtained from (a) 6-MLOC-P2, (b) 6-MGOC-TS3 for TR-I as training region.

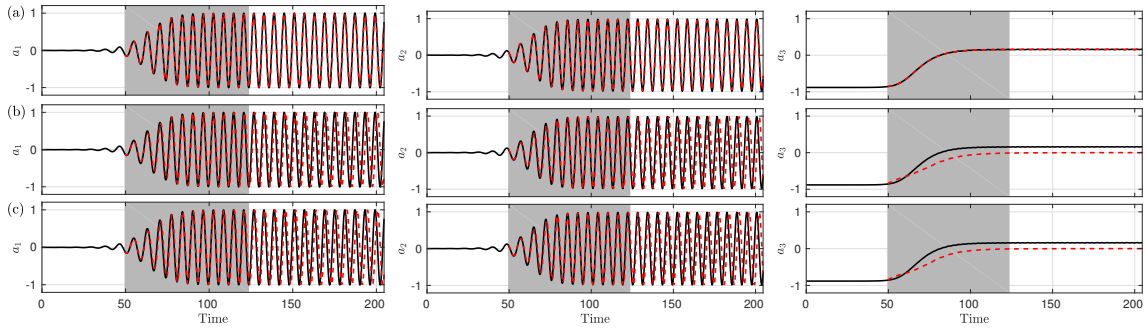


Figure 2.13: Times series of predicted POD features obtained from (a) 6-MLOC-P7, (b) 6-MGOC-TS9 and (c) 6-MGOC-TS20 for TR-I as training region.

included in Appendix 7.1.

As a reference, prior work Lu et al. (2018) on locally optimal architectures (6-MLOC-P2) has shown that 50 input features, 1325 quadratic nonlinear features with $\mathcal{LP}$ dimension of $1.7e^6$ can generate accurate predictions, thus indicating the need for a much larger $\mathcal{LP}$ dimension. Here, we explore whether expanding the $\mathcal{LP}$ dimension with just 3 input features improves the model performance. We accomplish this by increasing the order of polynomial to 7^{th}-degree i.e. we consider a 6-MLOC-P7 architecture method with 3 POD features and a $\mathcal{LP}$ dimension of $15,625$- a nearly ≈ 200 factor increase. Increasing the $\mathcal{LP}$ dimension by a couple of orders of magnitude produces accurate predictions of the nonlinear dynamics as shown in Fig. 2.13a. We note that choices of polynomial smaller than degree seven did not produce accurate predictions. We also explore the effect of increasing the $\mathcal{LP}$ dimension for the MGOC architectures by changing N_f as shown in table 2.1. The predictions obtained using 6-MGOC-TS9 and 6-MGOC-TS20 (see Figs. 2.13(b) and (c)) with $\mathcal{LP}$ dimensions

of 891 and 3960 respectively (factors of $\approx$ 10 & 40) also showed improved performance and compare favorably to the 6-MLOC-P7 architecture. However, both these GOC variants show similar results indicating that performance improvements have saturated, possibly due to the non-inclusion of the bias term (Appendix 7.1). Summarizing, for both the MLOC and MGOC architectures, increasing the $\mathcal{LP}$ dimension improves learning and prediction performance. However, MGOC requires relatively modest increases in $\mathcal{LP}$ dimension as compared to MLOC methods which provides them an advantage. In a way, this result reinforces the underlying principles behind the success of deep learning architectures Bengio et al. (2015). The MLOC can be viewed as a two-layer shallow learning architecture requiring larger intermediate layer dimensions while the MGOC is its deep learning counterpart requiring smaller number of intermediate layer features, but layered over each other.

We use the same modeling architecture's 6-MLOC-P2, 6-MLOC-P7 along with with 6-MGOC-TS3,-TS9, -TS20 on the more challenging TR-II data and the resulting predictions of the POD features are shown in figures 2.14 and 2.15. In this case both the MLOC architectures, i.e. 6-MLOC-P2 and 6-MLOC-P7 perform inadequately in spite of the extended $\mathcal{LP}$ space. On the other hand, predictions obtained using 6-MGOC-TS offer better qualitative results and predict the limit cycle dynamics, but display perceptible quantitative inaccuracy without a bias term and is insensitive to extension of learning parameter space (see table 2.2). However, as before, we observe that this quantitative inaccuracy, especially in the third POD feature is mitigated through the inclusion of a bias term as the plots clearly show in Fig. 7.2 in Appendix 7.1.

We note that computing the error metrics using a simple L_2 norm do not always represent the observed qualitative nature of the predictions accurately for such repetitive limit-cycle dynamics. For example, the predictions which qualitatively learn the dynamics but with a phase error tend show larger errors than some of the non-

qualitative predictions. The other aspect worthy of mention is that learning of all the MLOC/MGOC models is based on learning the 'local' errors and not the global errors that takes into account error propagation using predictions. Such a 'local' cost function misleads the learning process as the minimization of ($\mathcal{J}$) does not reflect minimization in prediction errors. To illustrate, in table 2.2, although the cost functions ($\mathcal{J}$) in most of the methods considered here are reduced to $O(1e^{-6})$, the associated prediction errors are of $O(1e^{-1})$. Improved regularization methods that use Jacobian of the cost function have been proposed in Pan & Duraisamy (2018) and will need to be explored.

To relate the observed deviation in the POD features to the predicted flow field of interest, we show in Fig. 2.16 the reconstructed solution (i.e. the actual predicted state vector) for $Re = 100$ obtained using the different methods considered in this paper. These plots are generated based on learning and prediction using TR-I ($cycles: 8-20$) data, and shown at $\approx T = 86.2$ (first column) which is the midpoint of the training region. Columns 2 and 3 in Fig. 2.16 represent predictions at $T = 124$, the last point in TR-I and $T = 205$, the last point in the prediction regime. These results clearly show that the MLOC methods with low $\mathcal{LP}$ dimension such as 4-MLOC-$\mathcal{I}1$, 6-MLOC-TS1 and 6-MLOC-P2 show delayed onset of wake instability and incorrect vortex shedding while the 6-MGOC-TS1 predicts the instability growth more accurately.

In summary, one or more strategies of extending the $\mathcal{LP}$ space, learning the parameters using a global optimization and improved regularizations help enhance the efficiency of the learning process and accuracy of the resulting predictions only when sufficient data is available. These strategies work much better with the TR-I data as against the TR-II. For the TR-II regime, the limited quantity of information about the limit-cycle dynamics in the training data is harder to overcome by the design of the machine learning architecture. Fortunately, in this case the MGOC architectures offer a significant improvement over MLOC, for a given $\mathcal{LP}$ dimension (computational

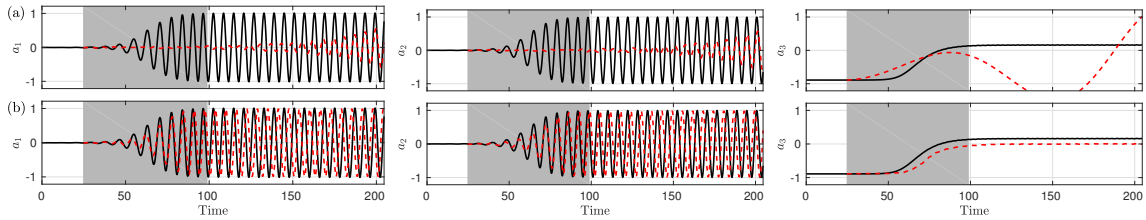


Figure 2.14: Times series of predicted POD features obtained from (a) 6-MLOC-P2, (b) 6-MGOC-TS3 for TR-II data.

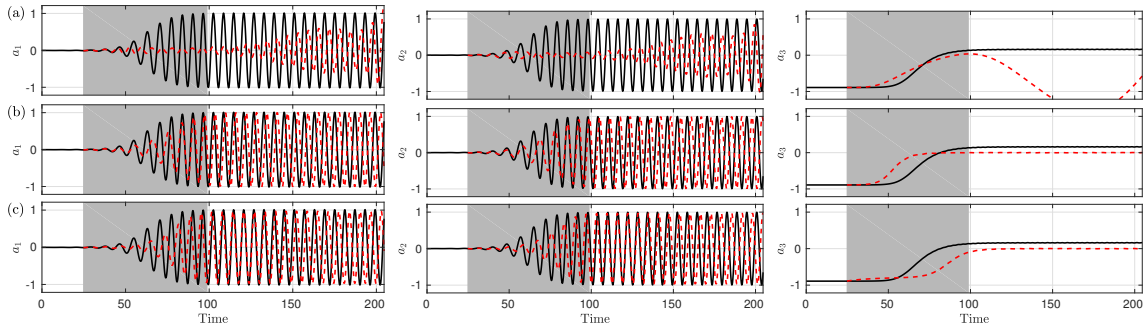


Figure 2.15: Times series of predicted POD features obtained from an extended $\mathcal{LP}$ space (a) 6-MLOC-P7, (b) 6-MGOC-TS9 and (c) 6-MGOC-TS20 for TR-II data.

cost), especially with the inclusion of a bias term.

So far, we explored the importance of training data quality (i.e. relevance to the data we are trying to predict) for MLOC methods while MGOC methods are relatively more robust to the data regime chosen for training. In this section, we briefly explore the impact of the inherent dimensionality of the system by considering a different flow Reynolds number, $Re = 1000$. In this case the first three POD features represent 90% of the total energy (Fig. 2.7(a)) as against 95% for $Re = 100$ flow. The phase portrait for the first three modes is shown in figure 2.7(b) which indicates the dynamics transitioning into a limit cycle much faster than observed for $Re = 100$. For the data-driven modeling assessment, we once again choose two different training regimes with different proportion of transient and limit-cycle information in the data as shown in Fig. 2.9b. The predictions for the different MLOC and MGOC models for both TR-I and TR-II regimes are shown in Figs. 2.17, 2.18 and 2.19 with quantifications reported in table 2.3. As observed for the low Reynolds number case,

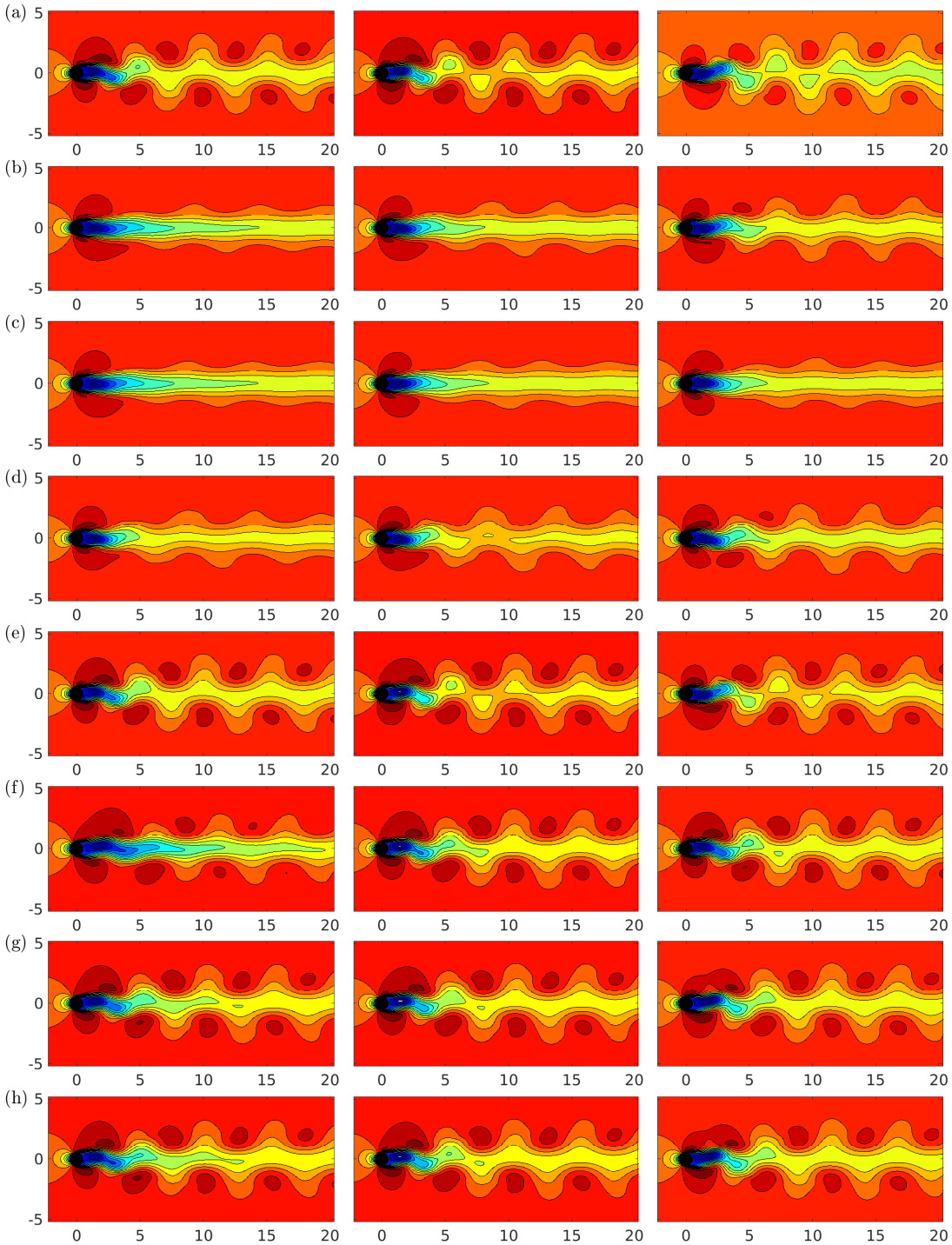


Figure 2.16: Reconstruction of $Re100$ flow field based on predicted POD features obtained from (a) Actual data, (b) 4-MLOC-$\mathcal{I}1$ (c) 6-MLOC-TS1 (d) 6-MLOC-P2 (e) 6-MLOC-P7 (f) 6-MGOC-TS1 (g) 6-MGOC-TS3 (h) 6-MGOC-TS9 comparison with 15 equally spaced contour levels ranging between $(-0.2645, 1.2963)$.

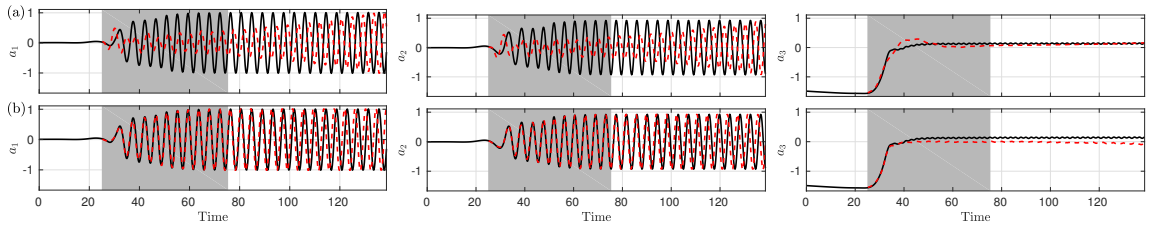


Figure 2.17: Times series comparison plot of the predicted POD features with the original data for TR-I data using different MLOC and MGOC modeling frameworks. (a) 6-MLOC-P2 (EDMD-P2) and (b) 6-MGOC-TS3.

the MGOC architectures perform better than the MLOC architecture in the TR-I regime for the similar $\mathcal{LP}$ dimensions as shown in Fig. 2.17. With increase in the $\mathcal{LP}$ dimension, the 6-MLOC-P7 or EDMD-P7 framework shows much improved accuracy (in Fig. 2.18 relative to the 6-MLOC-P2 while the MGOC performance improvement has saturated. The bias that shows up in the third POD feature for the MGOC models goes away when adopting a non-zero bias term in the architecture as shown in Fig. 7.3 in Appendix 7.1.

Moving on to the more challenging TR-II regime in Fig. 2.19, the 6-MLOC-P7 framework performs poorly although it made accurate predictions for the TR-I data. The different MGOC with large number of learning parameter,$\mathcal{LP}$, also perform poorly (see Fig. 2.19), but capture the overall qualitative behavior such as the dynamics settling into a limit cycle. However, predicted growth of the wake instability is faster than that for the true data. This is easily seen from table 2.3 where the prediction errors for the MGOC methods is greater than or comparable to the prediction errors. Unlike in the previous instances, these predictions do not improve with the addition of a bias unit as shown in Fig. 7.4 in Appendix 7.1. The prediction inaccuracies for this high Reynolds number case points to the possibility of the data missing dynamically relevant information contained in the lower energy containing POD modes. However, in practice, this is a realistic representation of the available data quality with the smaller scale features often not resolved with sufficient resolution.

Train		6-MLOC-P (EDMD-P)		6-MGOC-TS		
cycles	$\mathcal{E}$	p = 2	p =7	$N_f = 3$	$N_f = 9$	$N_f = 20$
$06 - 18$	$\mathcal{E}_t$	0.373	$2.6e^{-4}$	$1.5e^{-2}$	$2.6e^{-2}$	$3.8e^{-2}$
	$\mathcal{E}_p$	0.851	$2.6e^{-4}$	0.330	$2.2e^{-2}$	$9.7e^{-2}$
$02 - 14$	$\mathcal{E}_t$	0.320	0.191	0.379	0.695	0.606
	$\mathcal{E}_p$	3.414	0.375	0.183	0.395	0.654

Table 2.3: Prediction error estimates for layer-wise local (MLOC) and global optimization methods (MGOC) for $Re = 1000$.

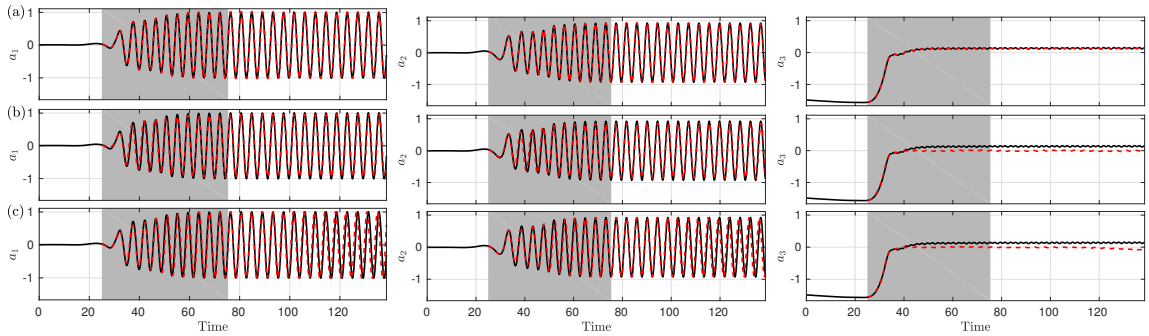


Figure 2.18: Time series comparison plot of the predicted POD features with the original data for TR-I data using different MLOC and MGOC modeling frameworks. (a) 6-MLOC-P7 (EDMD-P7), (b) 6-MGOC-TS9 and (c) 6-MGOC-TS20.

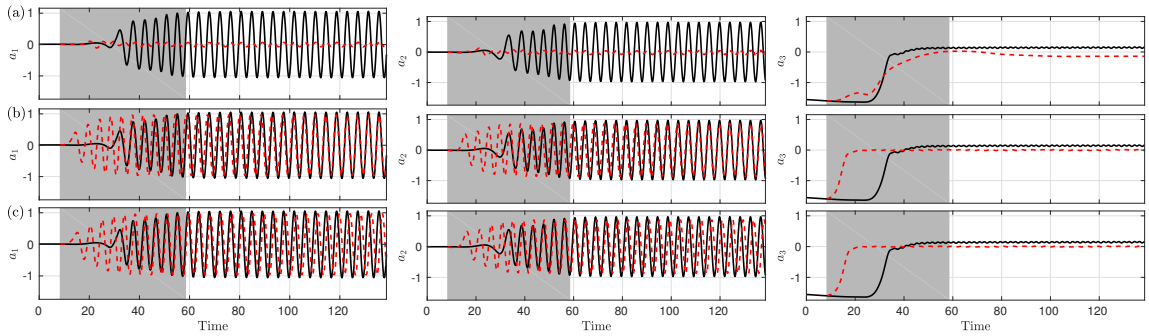


Figure 2.19: Time series comparison plot of the predicted POD features with the original data for TR-II data using different MLOC and MGOC modeling frameworks. (a) 6-MLOC-P7 (EDMD-P7), (b) 6-MGOC-TS9 and (c) 6-MGOC-TS20

2.3.6 Learning and Prediction of a Transient 2D Buoyant Boussinesq Mixing Flow

Unlike the low-dimensional limit-cycle attractor modeled in the earlier sections, here we explore a non-stationary and higher-dimensional buoyant Boussinesq mixing flow discussed in section 2.3.1. In fact, we observed previously that prediction of the transient evolution of the cylinder wake dynamics before it stabilizes into a limit-cycle is highly sensitive to the choice of training data. In addition, learning and predictability of these dynamics are also dependent on the flow data capturing sufficient dynamics for accurate prediction. For this study, we chose to retain just 80% of the total energy of the system captured in the CFD generated data (similar to a low resolution measurement) resulting in just 3 POD features in the 2^{nd} layer of the MLOC/MGOC architecture. Sensitivity to these aspects is stronger when trying to predict non-stationary phenomena that may settle into an unknown attractor over long times. The training data is almost always sparse for such dynamics and may not overtly show any evidence of the existence of such an attractor. Such instability-driven non-stationary problems are challenging for data-driven techniques that do not leverage knowledge of the underlying governing system and employ black box machine learning. Even if one were to diversify the training data-set with multiple realizations of the system, performance improvements are not guaranteed as the underlying dynamics will be different. We choose a single realization of such a data-sparse and low-dimensional representation of system for assessment of the MLOC and MGOC architectures .

For this case study, we consider the following locally optimal methods 4-MLOC-$\mathcal{I}$ (DMD), 6-MLOC-TS (EDMD-TS) and 6-MLOC-P (EDMD-P). We contrast these with the following globally optimal 6-MGOC-TS with growing $\mathcal{LP}$ dimensions for $N_f = 1, 3, 5$. As a preliminary step, we use the entire available data for training and assess the reconstruction performance of these models. Figure 2.21 compares

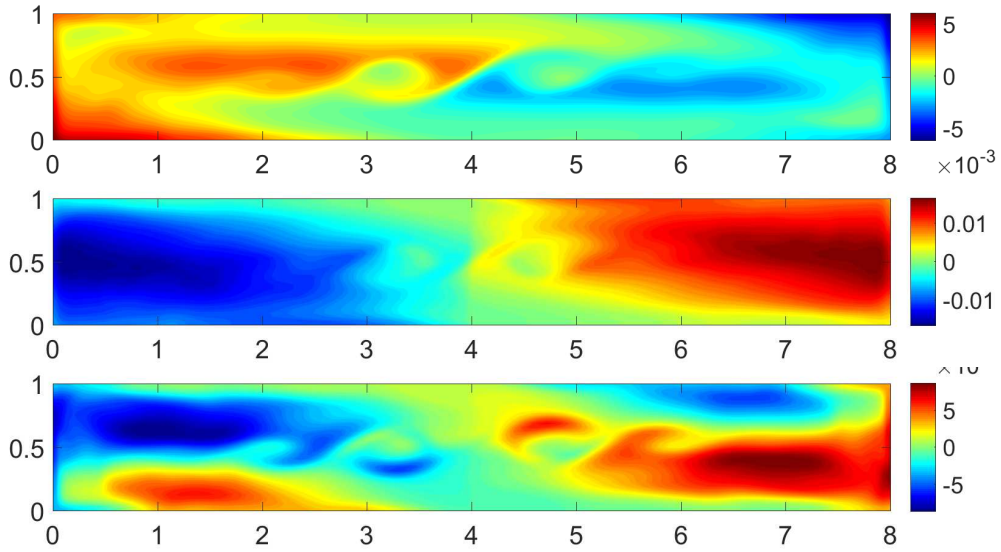


Figure 2.20: Visualization of the first three POD basis (in decreasing order of energy content) used to model the dynamics with the data-driven models.

the results for the linear 4-MLOC-$\mathcal{I}$ (DMD) model with the nonlinear 6-MLOC-TS1 (EDMD-TS1) and 6-MGOC-TS1 (EDMD-P) models with small number of learning parameters (9 and 27 respectively). Contrary to findings from the earlier section, all the LOC models including the linear DMD and EDMD with tansigmoid nonlinearity (EDMD-TS1) compare favorably to the GOC models with $N_f = 1$. All three models fail to predict the dynamics of the third POD feature which represents the secondary eddies from the Kelvin-Helmholtz instability generated by the mixing layer dynamics (see bottom plot in Fig. 2.20). The LOC models generate slightly better outcomes as compared to the MGOC(FFNN) for the first two POD features that represent transverse and vertical mixing (top two plots in Fig. 2.20). To improve the predictions of the third POD feature, we expand the learning parameter ($\mathcal{LP}$) dimension by comparing 6-MLOC-P3 (EDMD-P3), 6-MGOC-TS3 and 6-MGOC-TS5 as shown in Fig. 2.22. Consistent with earlier observations, this increase in $\mathcal{LP}$ improves the prediction of the third feature for both the MLOC and MGOC methods with MLOC performing better. Similar performance was also realized with the EDMD-P2 (6-MLOC-P2) architecture and is not reported here for brevity. This shows that for

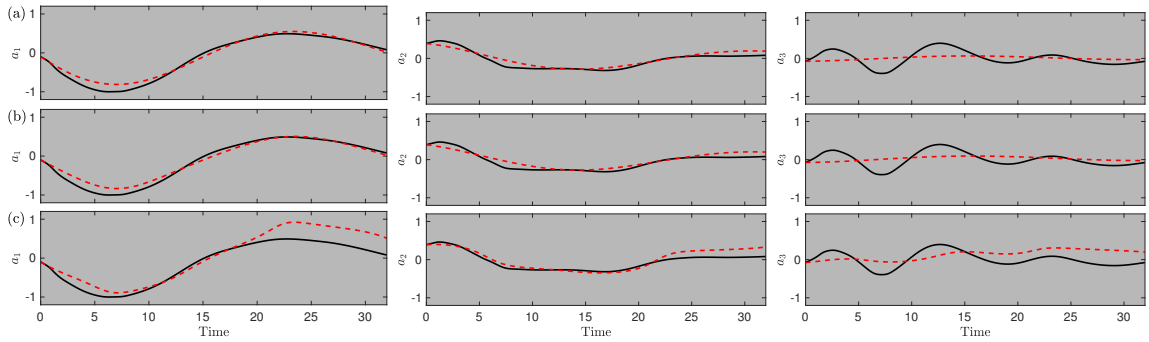


Figure 2.21: Times series comparison plot of the 3 POD weights with the original data with entire data used for training (a)4-MLOC-$\mathcal{I}$1 (DMD or 6-MLOC-P1) , (b) 6-MLOC-TS1 and (c) 6-MGOC-TS1

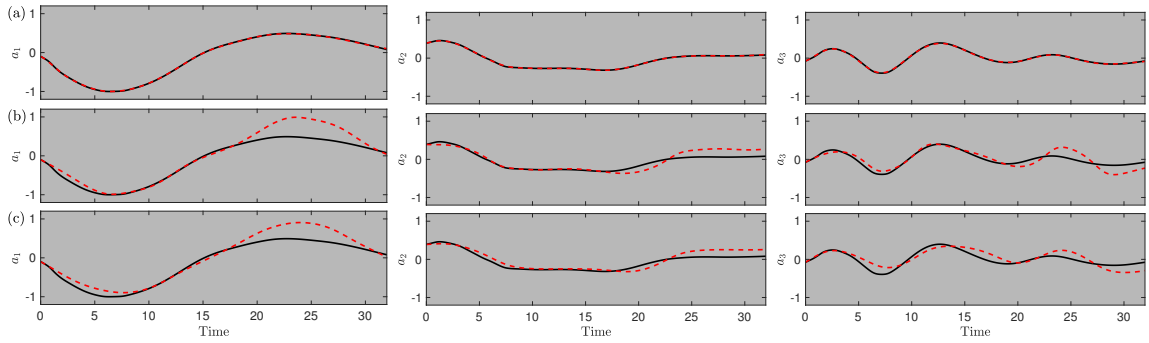


Figure 2.22: Times series comparison plot of the 3 POD weights with the original data with complete data used for training (a) 6-MLOC-P3 (EDMD-P3) , (b) 6-MGOC-TS3 and (c) 6-MGOC-TS5

reconstructing the dynamics, MLOC methods are more accurate as compared to the globally optimal FFNNs. This can be attributed to the choice of nonlinear mapping involved in MGOC architectures and exploration of other transfer functions is beyond the scope of this study.

To assess the ability of the models to learn the underlying system dynamics, we split the dataset equally into training and prediction regimes. Figure 2.23 compares the predicted output of the three POD features for 6-MLOC-TS1, 6-MLOC-P2 and 6-MGOC-TS3. For all the models, we clearly observe that reconstruction is better than prediction performance. So we focus on the latter in this discussion. The globally optimal FFNN (6-MGOC-TS3) outperforms the two LOC model architectures considered here in terms of stability and accuracy. Particularly, the MGOC predic-

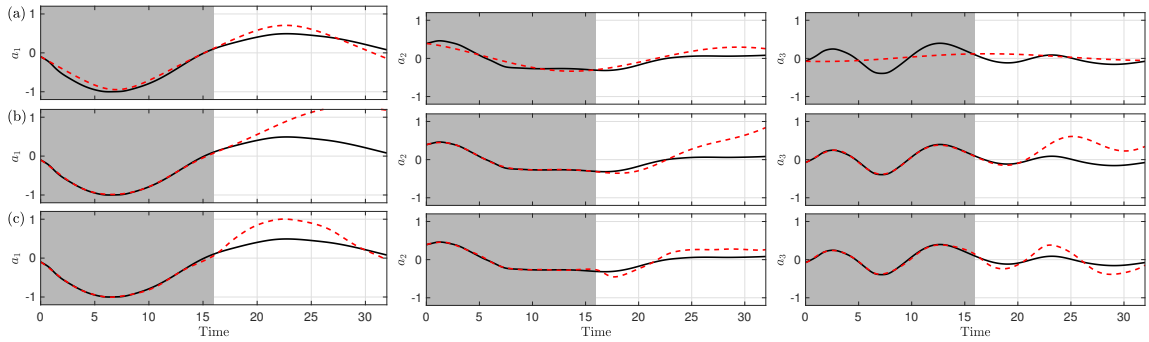


Figure 2.23: Times series comparison plot of the 3 POD weights with the original data with half data used as training (a) 6-MLOC-TS1 , (b) 6-MLOC-P2 and (c) 6-MGOC-TS3

tion using 50% data (Fig.2.23c) is highly similar to that using the entire dataset as the reconstruction experiment shown in Fig. 2.22c. This shows that these models offer robust and stable performance even with limited data. In summary, we see that MLOC methods offer competitive reconstruction performance, but MGOC models across the different architectures offer stable and robust model performance for long time predictions using limited data.

2.4 Summary

We explored the role of local (MLOC) versus global optimization (MGOC) of the multilayer convolution maps through the lens of learning parameter dimensionality and nonlinear transfer functions on their ability to reconstruct and predict over long times the transient, nonlinear dynamics of canonical fluid flows. The success of both the MLOC and MGOC architectures are tied to the nonlinearity in the mapping and the size of the learning parameter space in the multilayer architecture design. We observe that for prediction of limit-cycle dynamics and transient from different regions of data where all the different models show reasonable success, FFNN-like MGOC models control the growth of long-time prediction errors better than MLOC models.

CHAPTER 3

Deep Koopman Networks: Predictions

Deep Neural networks are very good universal function approximators (LeCun et al. 2015), even a shallow network given high number of neurons can approximate any given function to a very good accuracy. The deep neural networks advantage from the nonlinear transfer functions embedded in multiple hidden layers (Raissi et al. 2017). DNN have had a lot of success with classification problems, while using DNN for predictions dynamical systems have had very little success. DNN works very well for the classification problems very well, this is due to the fact that the transfer function like Rectified linear unit, log-sigmoid mainly focus to separate or distinguish the image pixels from zero intensity and higher intensity. This distinguished approach work very well for classification. But, if we use the same transfer functions for function approximation the transfer functions don't really get excitepd or provide any help with predictions, they are designed to work better in separation of intensities for classification problem, which indeed leads to lose of information that is critical physics in function approximation.

But we see that DNN are still providing in some cases accurate results, so when the lose of information happens during the manifold change, the weights have to do a lot of work to decompress the data to the exact physics. This puts a lot of onus on the weights for accurately prediction and this causes over fitting of the physics. Most function approx problems are predictive in nature and the input and output features are in the same manifold, but a general feed forward network does not guarantee the reverse mapping of the features. This motivated us to explore network architectures

that maps data into the original manifold using inverse transfer functions in a feed forward network. This approach retains the advantages of nonlocal optimization of a feed forward networks, but acknowledges the reverse mapping to the original manifold. This reverse mapping of manifold lies at the core of Koopman theory, wherein the identification of the optimal mapping from input space feature in which the dynamics can be approximated using a linear operator. Dynamic mode decomposition (Schmid 2010, Rowley et al. 2009) and its variants (Williams et al. 2014, Williams et al. 2015) are popular Koopman approximation methods and typically employ data-driven proper orthogonal decomposition (POD) convolutions. These methods have had varying success in modeling dynamics of fluid flows and have proven to be successful as long as the convolution map is optimal, i.e. produces a mapping that both sparse and appropriately nonlinear. This has motivated the need to find the optimal convolution maps (Lusch et al. 2017, Otto & Rowley 2017, Williams et al. 2015, Wu et al. 2018) for a given nonlinear fluid flow physics. This allows for the predictions to happen in the feature space and evolved features are back to the original manifold. We present the results that illustrate the performance in comparison to the linear models and multilayer convolution models with local optimization.

Our goal through this effort is to develop data-driven models that allow for system identification and also provide reliable predictions of dynamics.

In particular, a Koopman theory based model should incorporate the following:

1. A convolution map that accurately projects the state data to and from the feature space without loss of information, while incorporating the appropriate degree of nonlinearity.

2. A system identification framework to capture the evolution of the dynamics in the feature space.

While many learning algorithms try to find/identify the optimal 'magic' feature maps,

a common practice is to build nonlinear convolution operators that are composed of multiple elementary convolutions layered on top of each other and generalized as a multilayer convolution framework (Lu et al. 2018). However, these operators often are an assumed form or optimized locally. The framework presented here utilizes Multilayer Global Optimal Convolution (MGOC) to identify the optimal combination of the convolution operators for an improved Koopman approximation of the dynamics (Puligilla & Jayaraman 2018*c*). As shown in (Puligilla & Jayaraman 2018*b*), this severely limits the ability of the convolution map to represent the dynamics for a given layer dimension. Feedforward neural networks (FFNNs) provide an intriguing alternative as global optimal convolution architectures and are effective Markovian prediction tools (Puligilla & Jayaraman 2018*b*), but are purely forward maps, i.e they do not use symmetric convolutions which prevents learning of a Markov linear operator. To address this, the authors have developed a modified neural network architecture termed as *Deep Koopman Neural Networks (DKN)* (Puligilla & Jayaraman 2018*a*, Jayaraman & Puligilla 2018) that enforce symmetry and learn the Koopman operator simultaneously by combining two FFNNs trained in sync. An alternative is to leverage deep neural networks (DNNs) to learn multilayer convolution maps from data through the use of symmetric deep autoencoder networks or AENs (Hinton & Salakhutdinov 2006) that invariably allow for dimensionality reduction and encoding data into a transformed 'feature' space. Both these approaches are similar in using deep learning to compute data-driven embeddings, but the former learns the corresponding Koopman operator approximation simultaneously while learn the mapping. g. In the second approach, the Koopman (Markov linear) operator is learned in a separate step. Both these approaches perform similarly in most cases, but the DKN produced robust long-time predictions that has implications for data-driven modeling.

The objectives of this study are as follows:

1. Identify convolution maps that can represent the dynamics to find linear koop-

man operator.

2. Modify to MGOC frameworks to find Koopman invariant subspaces and linear operator simultaneously.

3.1 Methodology

In our previous efforts, it was observed that a MGOC framework enables learning of the dynamical system more efficiently compared to MLOC frameworks for a given hyper-parameters span (Puligilla & Jayaraman 2018*c*). But, the MGOC based on neural networks architectures are limited by the forward propagation which does not allow for the forward-backward (Puligilla & Jayaraman 2018*c*) convolutions which MLOC method leverage to approximate linear Koopman operator via least squares. In Eq. (4.1), the $\boldsymbol{g}$ map provides a forward and backward convolution of the state vectors and is important for coordinate transformation which are invariant. The forward and backward convolution is tied to the symmetry of the framework used to learn a dynamical system. For example, an optimal convolution set that helps linearize the dynamics maps the inputs to a manifold spanned by Koopman subspaces which should then be 'reverse mapped' to the original manifold. Autoencoders (Hinton & Salakhutdinov 2006) are a good example of MGOC frameworks that are widely used in dimensionality reduction. Recent efforts to use autoencoders in the context of Koopman theory have had interesting conclusions (Lusch et al. 2017, Otto & Rowley 2017). Where in, an autoencoder is used to map input data to itself to find lower dimensional representation of the input data. Using this knowledge of the convolution operators and maps once can build encoders which transforms the data in the original manifold to a lower dimensional embedding. Similarly a decoder can be built to transform the data back into the original manifold. Here the process of finding linear koopman operator is a two step process, wherein first an autoencoder is used to find the koopman embeddings and then a koopman operator is computed. We believe, this two step process will render learning process into an excise in identifying the optimal 'magic' convolutions. Here again, the convolution and Koopman operator are treated

independently. We have observed that by modifying the MGOC framework we can use the learning capabilities of back propagation to find both the convolution maps and Koopman operator simultaneously.

With this in mind, we have used two strategies that help ensure symmetry of the MGOC frameworks. There are three processes that enable learning of the dynamics efficiently, namely, the convolution operators (Θ), the nonlinear convolution maps ($\mathcal{N}$) and the back-propagation algorithm. The following strategies used to ensure symmetry in MGOC frameworks:

1. Deep Koopman Network: Constraining convolution operators (Θ)'s using an penalty network(Puligilla 2018) that ensures symmetry via encoder-decoder networks(see Fig. 3.1)

2. Auto-Encoder Network: A two step process involving MGOC to learn the encoder-decoder networks and using MLOC to learn the Koopman operator (see Figure 3.2).

In the following sections, we will present the two modified MGOC frameworks that have helped improve the learning and representational capabilities of the underlying nonlinear dynamics.

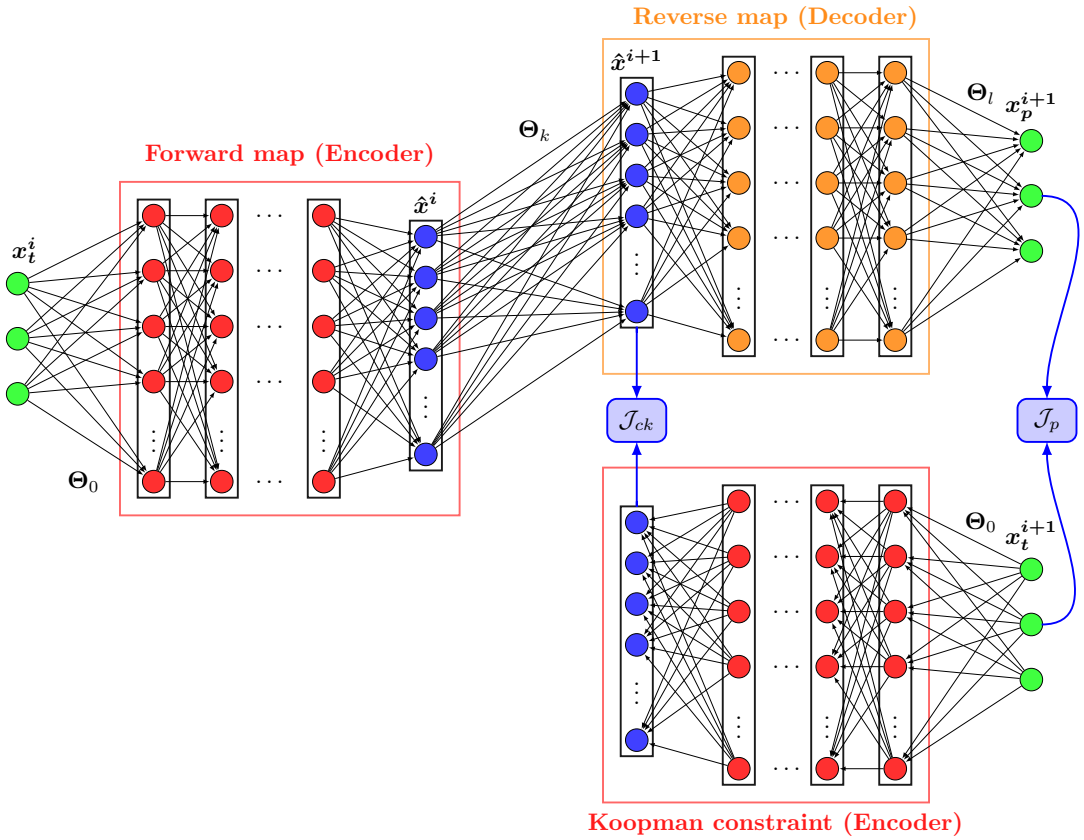


Figure 3.1: Deep Koopman Network (DKN) where the red bounding box represents encoder and the orange bounding box the decoder.

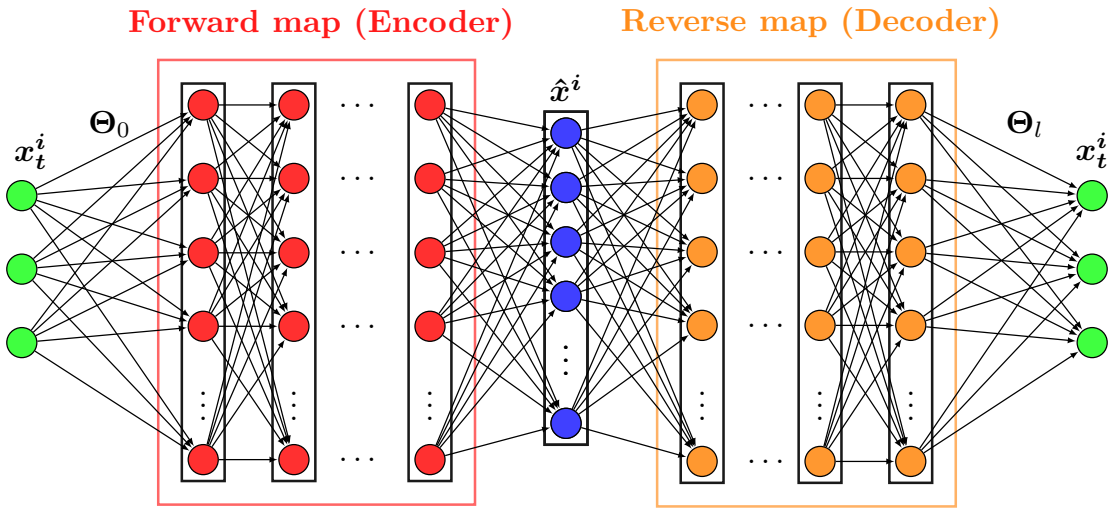


Figure 3.2: Two set Autoencoder Network (AEN) where the red bounding box represents encoder and the orange bounding box the decoder.

3.1.1 Forward Propagation for a 6 Hidden Layer Network

$$z_1 = \theta_0 a_0 = \theta_0 X$$
$$a_1 = g_1(z_1) \tag{3.1}$$

$$z_2 = \theta_1 a_1$$
$$a_2 = g_2(z_2) \tag{3.2}$$

$$a_3 = g_3(z_3) \rightarrow [\hat{X}] \tag{3.3}$$

$$z_4 = \theta_3 a_3 \rightarrow [\hat{Y}]$$
$$a_4 = g_4(z_4) \tag{3.4}$$

$$z_5 = \theta_4 a_4$$
$$a_5 = g_5(z_5) \tag{3.5}$$

$$z_6 = \theta_5 a_5$$
$$a_6 = g_6(z_6) \tag{3.6}$$

$$z_7 = \theta_6 a_6$$
$$a_7 = g_7(z_7) \tag{3.7}$$

Here the final layer has No squashing function g.

3.1.2 Penalty Network for Symmetry

$$cz_1 = \theta_0 Y$$
$$a_{1CKT} = g_1(cz_1) \tag{3.8}$$

$$cz_2 = \theta_1 a_{1CKT}$$
$$a_{2CKT} = g_2(cz_2) \tag{3.9}$$

$$a_{3CKT} = g_3(cz_3) \sim \hat{Y} \tag{3.10}$$

3.1.3 Cost Functions

$$J_{ff} = \sum \frac{1}{2m}(a_7 - Y)^2 \tag{3.11}$$

$$J_{CKT} = \sum \frac{1}{2m}(Z_4 - a_{3CKT})^2 \tag{3.12}$$

$$J = J_{ff} + J_{CKT} \tag{3.13}$$

where

$$J_{ff} = f(\theta_6, \theta_5.\theta_4, \theta_3, \theta_2, \theta_1, \theta_0, g) \tag{3.14}$$

$$J_{CKT} = f(\theta_3, \theta_2, \theta_1, \theta_0, g) \tag{3.15}$$

3.1.4 Proof of Symmetry

J_{CKT} is the Koopman constrain on the Network. If $J_{CKT} = 0$, then the network is symmetric.

$$Z_4 = a_{3CKT} \implies \theta_3 g(\theta_2 g_2(\theta_1 g_1(\theta_0 X))) = g_3(\theta_2 g_2(\theta_1 g_1(\theta_0 Y))) \tag{3.16}$$

$$\implies \theta_3 g \underbrace{(\theta_2 g_2(\theta_1 g_1(\theta_0 X)))}_{\mathcal{F}} = \underbrace{g_3(\theta_2 g_2(\theta_1 g_1(\theta_0 Y)))}_{\mathcal{F}} \tag{3.17}$$

$$\theta_3 \mathcal{F}(X) = \mathcal{F}(Y) \tag{3.18}$$

while J_{ff} is for the convergence to the solution.

$$Y = g_7(\theta_6 g_6(\theta_5 g_5(\theta_4 g_4(z_4)))) = a_7 \tag{3.19}$$

3.1.5 Back Propagation for 6 Hidden Layer Network

$$\frac{dJ}{d\theta_6} = \underbrace{\frac{1}{m}(a_7 - Y)\frac{dg_7}{d\hat{z}_7} \cdot \frac{d\hat{z}_7}{dz_7}}_{\delta_7} \cdot a_6 + 0 \tag{3.20}$$

$$\frac{dJ}{d\theta_5} = \underbrace{\theta_6\delta_7\frac{dg_6}{d\hat{z}_6}\cdot\frac{d\hat{z}_6}{dz_6}}_{\delta_6}\cdot a_5 + 0 \tag{3.21}$$

$$\frac{dJ}{d\theta_4} = \underbrace{\theta_5\delta_6\frac{dg_5}{d\hat{z}_5}\cdot\frac{d\hat{z}_5}{dz_5}}_{\delta_5}\cdot a_4 + 0 \tag{3.22}$$

$$\frac{dJ}{d\theta_3} = \underbrace{\theta_4\delta_5\frac{dg_4}{d\hat{z}_4}\cdot\frac{d\hat{z}_4}{dz_4}}_{\delta_4}\cdot a_3 + \underbrace{\frac{1}{m}(z_4 - a_{3CKT})a_3}_{\delta_{CBP4}} \tag{3.23}$$

$$\frac{dJ}{d\theta_2} = \underbrace{\theta_3\delta_4\frac{dg_3}{d\hat{z}_3}\cdot\frac{d\hat{z}_3}{dz_3}}_{\delta_3}\cdot a_2 + \underbrace{\theta_3\delta_{CBP4}\frac{dg_3}{d\hat{z}_3}\frac{d\hat{z}_3}{dz_3}}_{\delta_{CBP3}}\cdot a_2 + \underbrace{(-\frac{1}{m})(z_4 - a_{3CKT})\frac{dg_3}{dcz_3})}_{\delta_{EBP3}}\cdot a_{2CKT} \tag{3.24}$$

$$\frac{dJ}{d\theta_1} = \underbrace{\theta_2\delta_3\frac{dg_2}{d\hat{z}_2}\cdot\frac{d\hat{z}_2}{dz_2}}_{\delta_2}\cdot a_1 + \underbrace{\theta_2\delta_{CBP3}\frac{dg_2}{d\hat{z}_2}\frac{d\hat{z}_2}{dz_2}}_{\delta_{CBP2}}\cdot a_1 + \underbrace{\theta_2\delta_{EBP3}\frac{dg_2}{dcz_2}}_{\delta_{EBP2}}\cdot a_{1CKT} \tag{3.25}$$

$$\frac{dJ}{d\theta_0} = \underbrace{\theta_1\delta_2\frac{dg_1}{d\hat{z}_1}\cdot\frac{d\hat{z}_1}{dz_1}}_{\delta_1}\cdot a_0 + \underbrace{\theta_1\delta_{CBP2}\frac{dg_1}{d\hat{z}_1}\frac{d\hat{z}_1}{dz_1}}_{\delta_{CBP1}}\cdot a_0 + \underbrace{\theta_1\delta_{EBP2}\frac{dg_1}{dcz_1}}_{\delta_{EBP1}}\cdot a_{0CKT} \tag{3.26}$$

3.1.6 Conjugate Gradients

$$d_t = \nabla_\theta J(\theta) + \beta_t d_{t-1} \tag{3.27}$$

Fletcher - Reeves:

$$\beta_t = \frac{\nabla_\theta J(\theta_t)^T \nabla_\theta J(\theta_t)}{\nabla_\theta J(\theta_{t-1})^T \nabla_\theta J(\theta_{t-1})} \tag{3.28}$$

Polak - Ribiere:

$$\beta_t = \frac{(\nabla_\theta J(\theta_t) - \nabla_\theta J(\theta_{t-1}))^T \nabla_\theta J(\theta_t)}{\nabla_\theta J(\theta_{t-1})^T \nabla_\theta J(\theta_{t-1})} \tag{3.29}$$

3.2 Results

In this section, we will discuss the predictive capabilities of Deep Koopman Network(DKN) and Auto-Encoder Network. From the design of the architecture, we

Algorithm 1 The conjugate gradient method

Require: Initial parameters θ_0
Require: Training set of m examples
Initialize $\rho_0 = 0$
Initialize $g_0 = 0$
Initialize $t = 1$
1: **while** stopping criterion not met **do**
Initialize the gradient $g_t = 0$
Compute gradient: $g_t \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^i;\theta), y^i)$
Compute $\beta_t = \frac{(g_t - g_{t-1})^T g_t}{g_{t-1}^T g_{t-1}}$ (Polak - Ribiere) (Nonlinear conjugate gradient: optionally reset β_t to zero, for example if t is a multiple of some constant k, such as $k = 5$)
Compute search direction: $\rho_t = -g_t + \beta_t \rho_{t-1}$
Perform line search to find: $\epsilon^* = argmin_\epsilon \frac{1}{m} \sum_{i=1}^{m} L(f(x^i;\theta_t + \epsilon\rho_t), y^i)$ (On a truly quadratic cost function, analytically solve for ϵ^* rather than explicitly searching for it)
Apply update: $\theta_{t+1} = \theta_t + \epsilon^* \rho_t$
$t \leftarrow t + 1$
2: **end while**

can perform predictions in two ways as seen in Fig. 3.3. The inner loop predictions described in algorithm 2, here the decoding is performed once we perform the all predictions/time marching in the hidden features space. The inner loop depicts conventional way of predictions adopted in reduced order models like DMD, POD etc, where in the flow dynamics are evolved in a feature space. In case of the outer loop predictions, the encoding, prediction and decoding are all performed for each time step as described in algorithm 3. This approach depicts a forward propagation prediction employed in neural networks. The need for these predictions comes from the fact that, for a given problem, we would like to evaluate the performance of the Koopman networks for robustness in encoding and predictions. In the section also, we will use the cylinder flow dynamics and the 2D Boussinesq mixing flow to evaluate the performance of the Koopman networks. To this end, we have used a four hidden layer(4 - HL) and six hidden layer(6 - HL) Deep Koopman Networks and five(5 - HL) and seven(7 - HL) AutoEncoder networks for this study. The choice of the 4

- HL and 6 - HL is derived from the experience wherein we have found that deeper networks perform better than shallower networks. The architecture, nonlinear maps, optimization methodology and number of learning parameters for each network is detailed in table 3.1 In the following section, we will present the validation of the inner and outer loop predictions obtained from AEN and DKN methods.

Algorithm 2 Inner loop predictions

Require: Initial parameters $\bar{X}(1)$

Encode $\bar{X}(1) \xrightarrow{EN} \bar{\bar{X}}(1)$

1: **while** $i < N_p$ **do**

Perform predictions from $\bar{\bar{X}}(i) \xrightarrow{\Theta_k} \bar{\bar{X}}(i+1)$

i = i + 1

2: **end while**

Decode the $\bar{\bar{X}}(1:N_p) \xrightarrow{DEC} \bar{X}(1:N_p)$

Inner loop error $\mathcal{E}_{p-i} = \sum(\bar{Y}_a - \bar{Y}_p)^2/(2m)$

Algorithm 3 Outer loop predictions

1: **while** $i < N_p$ **do**

Encode $\bar{X}(i) \xrightarrow{EN} \bar{\bar{X}}(i)$

Perform predictions from $\bar{\bar{X}}(i) \xrightarrow{\Theta_k} \bar{\bar{X}}(i+1)$

Decode the $\bar{\bar{X}}(i+1) \xrightarrow{DEC} \bar{X}(i+1)$

i = i + 1

2: **end while**

Outer loop error $\mathcal{E}_{p-o} = \sum(\bar{Y}_a - \bar{Y}_p)^2/(2m)$

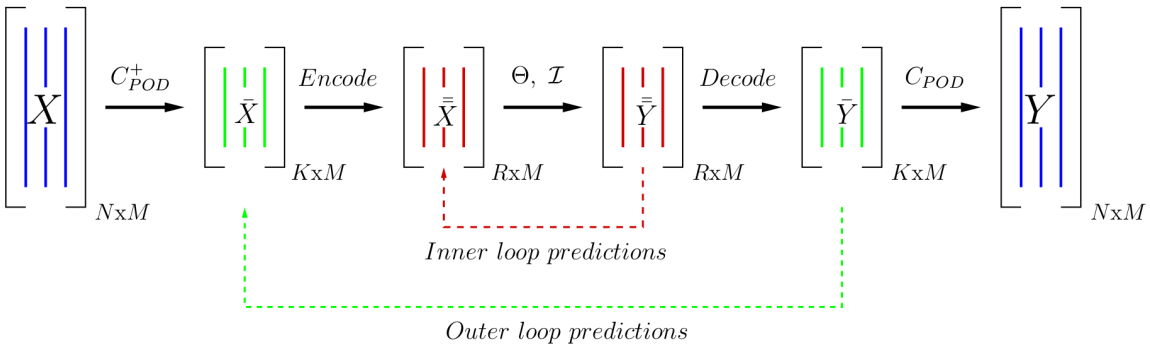


Figure 3.3: A schematic representation of the inner loop and outer loop prediction in Koopman networks (DKN and AEN)

	Architecture		$\mathcal{N}$	Optimization	$\mathcal{LP}$
1	DMD	55-55	$\mathcal{I}1$	2-MLOC	3025
		10-10	$\mathcal{I}1$	2-MLOC	100
		3-3	$\mathcal{I}1$	2-MLOC	9
2	EDMD	55-1595-1595-55	$P2$	4-MLOC	$2.5e^6$
		10-55-55-10	$P2$	4-MLOC	3025
		3-9-9-3	$P2$	4-MLOC	81
3	DKN	3-9-9-9-9-3	TS	6-MGOC	297
		3-9-9-9-9-9-9-3	TS	8-MGOC	459
4	AEN	3-9-9-9-9-9-3	TS	7-MGOC	378
		3-9-9-9-9-9-9-9-3	TS	9-MGOC	540

Table 3.1: Overview of the methods used as part of analysis in this chapter. The dimensions of the different layers correspond to that used for learning the model.

3.2.1 Validating with Limit-cycle Dynamics

Before performing analysis on more challenging data sets, we have used simple limit cycle data to validate the performance of the DKN and AEN networks. In Figs. 3.4 and 3.5, we have presented the results obtained from the inner and outer loop predictions, the first two rows in both figures are from the two step AEN, while the 3 and 4 rows are DKN results. In all the cases the long time predictions of first two POD weights are in good agreement with the actual data, but the third POD weight seems to diverge in DKN architecture. To asses the performance the prediction errors of these methods are compared with respect to the MLOC methods which perform excellently in limit cycle dynamics in table 3.2(16-20 cycles). In this table, we have also included the prediction errors obtained MLOC methods with POD weights which amounts to 100% energy content for analyzing the actual spectral content which is topic of next chapter and to assess the predictive capabilities of MLOC methods

even when high fidelity data is available. The architecture, features and optimization techniques used for MLOC methods (DMD and EDMD) are presented in table 3.1. In case of flow over the cylinder roughly ≈ 55 POD weights and for 2D Boussinesq Buoyant flow ≈ 10 accumulate $> 97\%$ of the flow energy. In table 3.2, the AEN and DKN methods are performed as better as the MLOC methods and this provides a good indication of the capability of DKN and AEN methods in predictions.

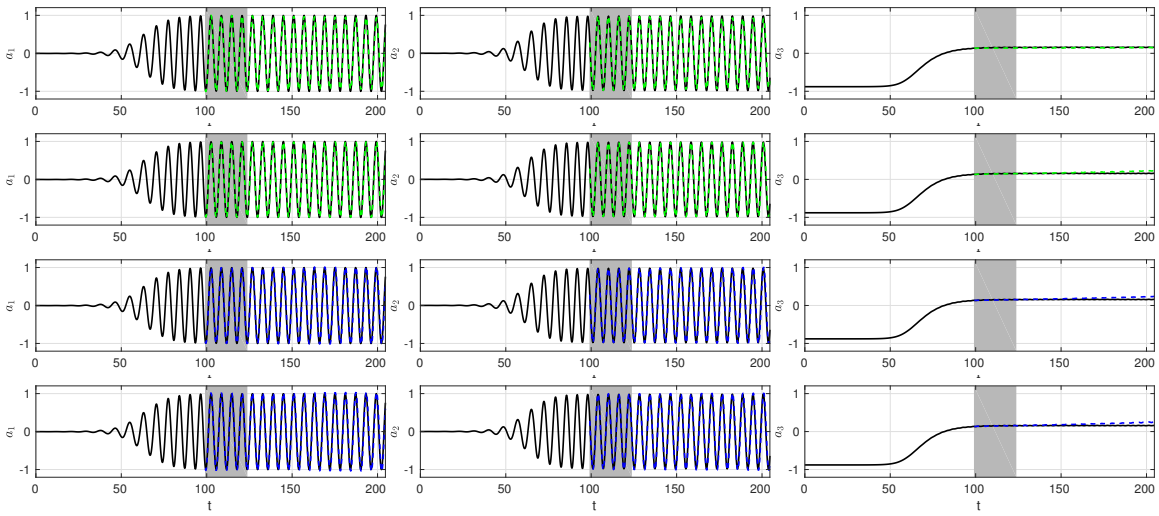


Figure 3.4: Comparison of inner loop predictions based POD weights, where (1^{st} and 2^{nd} row): 5 - HL and 7 - HL Auto-Encoder (AEN) and (3^{rd} and 4^{th} row): 4 - HL and 6 - HL Deep Koopman Network (DKN)

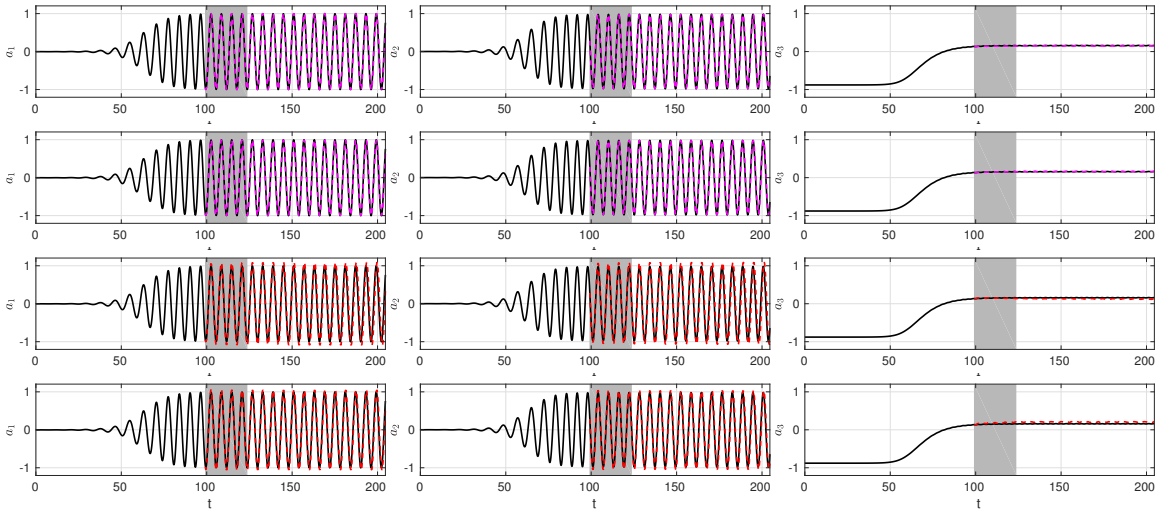


Figure 3.5: Comparison of outer loop predictions of POD weights, where (1^{st} and 2^{nd} row): 5 - HL and 7 - HL Auto-Encoder (AEN) and (3^{rd} and 4^{th} row): 4 - HL and 6 - HL Deep Koopman Network (DKN)

Further, in Figs. 3.6 and 3.7, we have presented the predictions in the Koopman space where the POD weights dynamics can be evolved or predicted linearly, obtained from AEN and DKN respectively. In both the figures the predicted dynamics using inner loop and outer loop seems to agree excellently with actual encoded data using the learned encoder. From the predictions plots for the AEN and DKN methods, both the inner loop and outer loop predictions seems to learn the dynamical behavior of the flow.

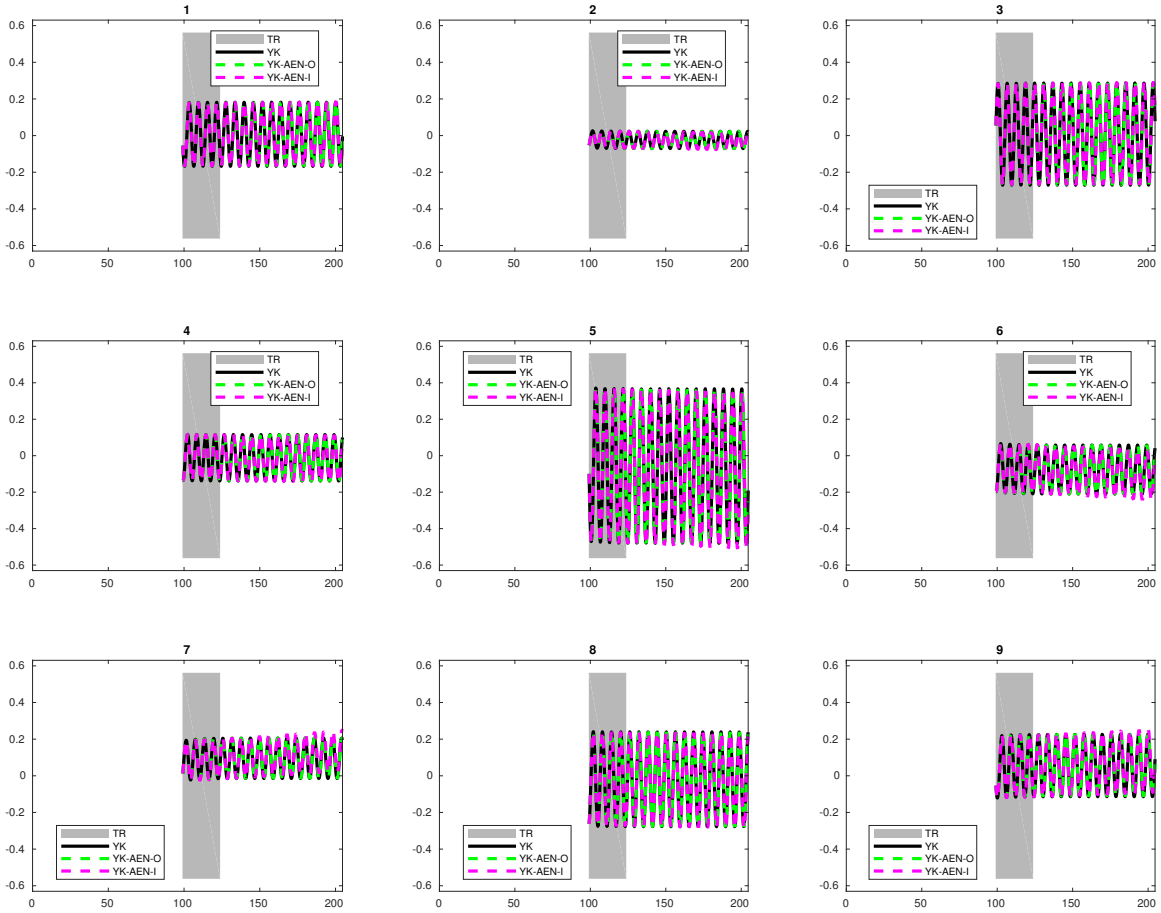


Figure 3.6: Comparison of inner loop vs outer loop predictions in Koopman space from 7- H L Auto-Encoder (AEN)

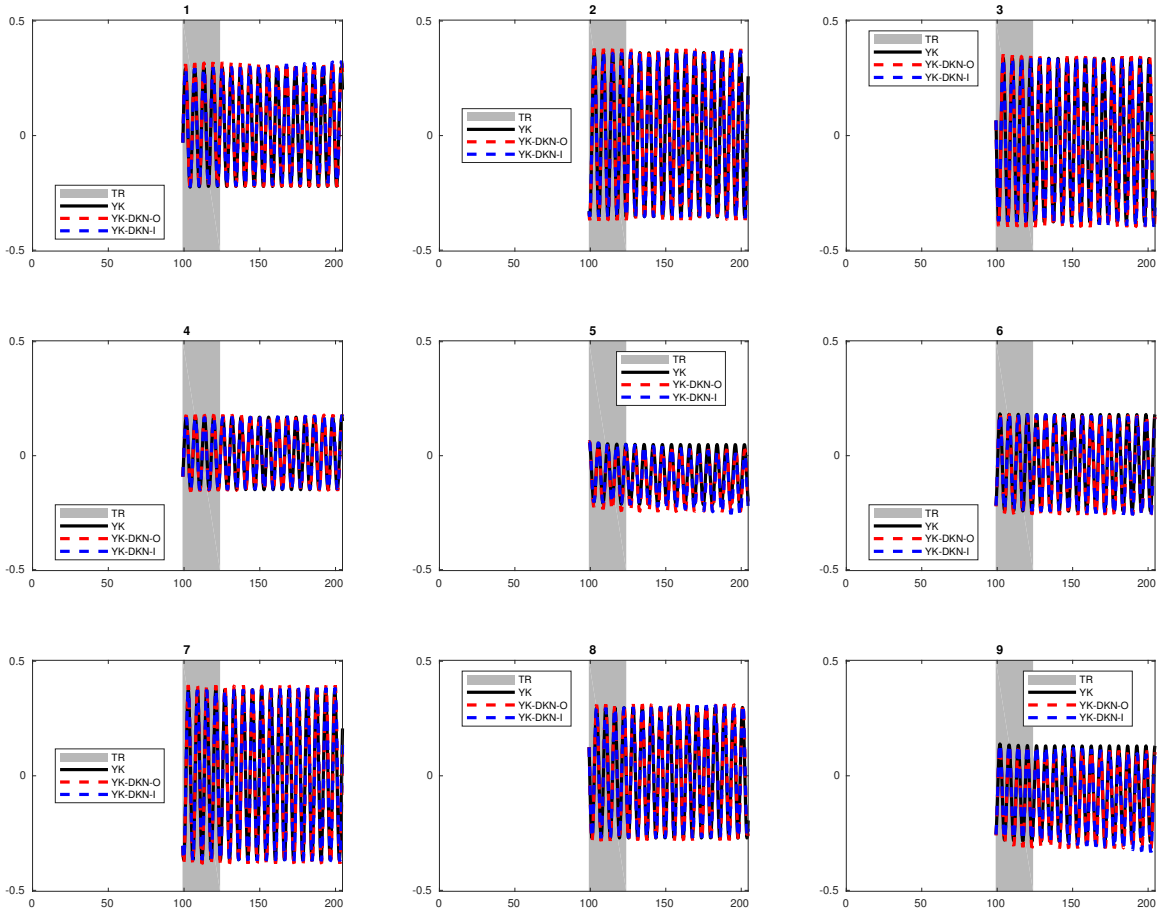


Figure 3.7: Comparison of inner loop vs outer loop predictions in Koopman space from 6 - HL Deep Koopman Network (DKN)

3.2.2 Learning and Prediction of a Transient 2D Buoyant Boussinesq Mixing Flow

In the chapter 2, we have seen that predictions in Fig. 2.22 from MLOC methods where in good agreement, while the MGOC method did not perform very well. In the case of MGOC, the method under performed due to the choice of nonlinear mapping (transfer functions) and non-symmetric architecture (forward propagation). In this section, DKN and AEN architectures which are symmetric networks perform excellently in the inner loop predictions of the POD weights of the Boussinesq mixing flow presented in Fig. 3.8. But, when we perform the outer loop predictions, the results (see Fig. 3.9)

seems to agree with the results obtained from the 6-MGOC-TS3(FFNN). Although this behavior is not expected, the application of encoder-decoder on every prediction step in outer loop is causing the predictions in the Koopman space to grow in error and eventually blowing up. On the other hand, the inner loop predictions do not encounter the error growth due to encoding and decoding. Further, from the table 3.2, we can see that the one step of encoding-decoding introduces an error of the order $\approx 1e^{-6}$ and from Fig. 3.10, we can see that the cost minimization that forces the encoding error J_{ff} in AEN and J_{ckt} in DKN is also of the same order. The inner loop predictions are comparable to the EDMD with 3 POD weights, while the outer loop does not perform well. This is in line with hypothesis initially made about the advantages of a symmetric networks over non-symmetric networks. Here again in Figs. 3.11 and 3.12, the inner loop predictions in the Koopman space agree very well with the actual data.

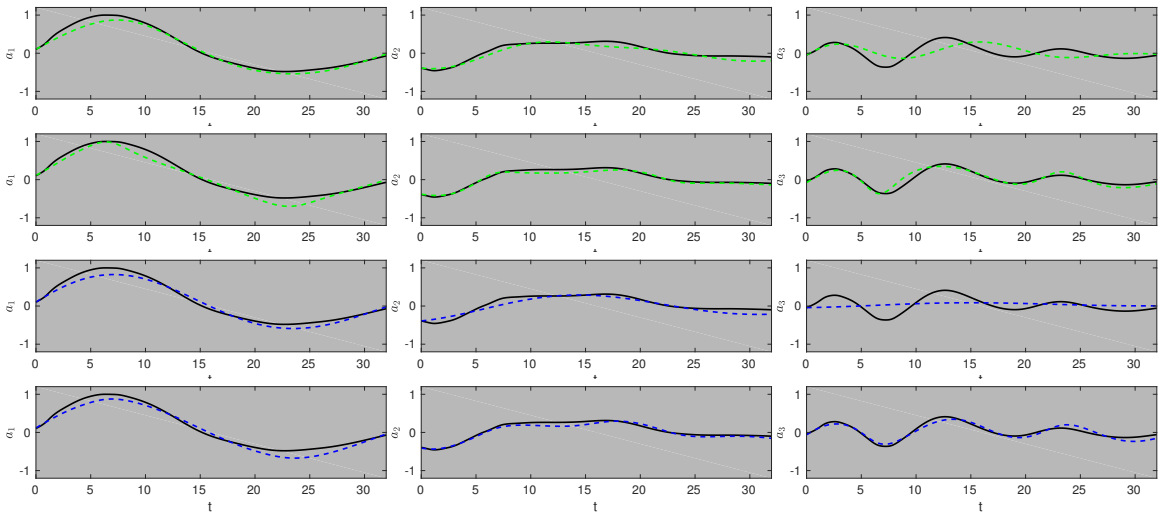


Figure 3.8: Comparison of inner loop predictions based POD weights, where (1^{st} and 2^{nd} row): 5 - HL and 7 - HL Auto-Encoder (AEN) and (3^{rd} and 4^{th} row): 4 - HL and 6 - HL Deep Koopman Network (DKN)for Buoyant Boussinesq Mixing Flow

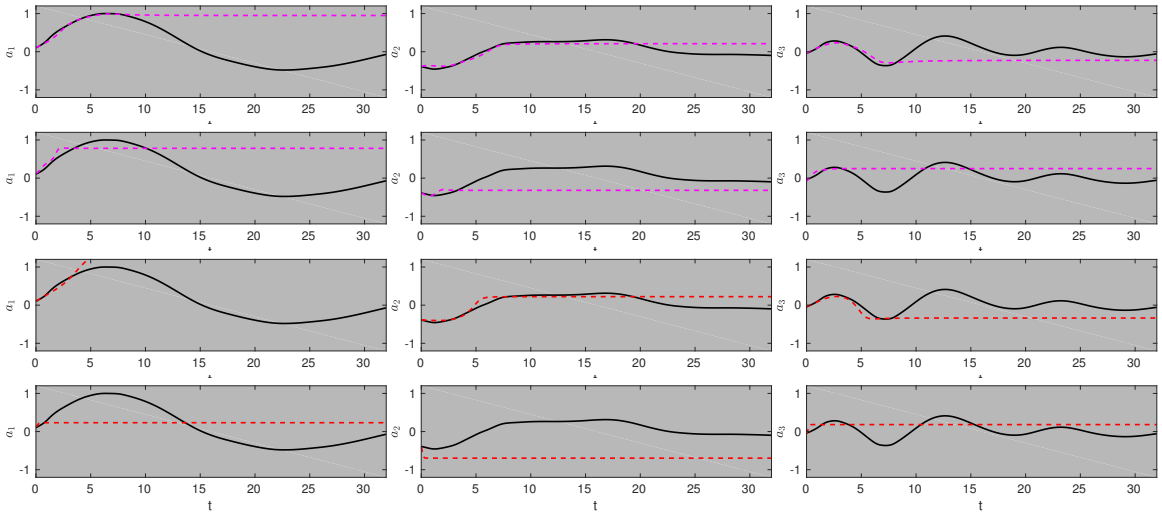


Figure 3.9: Comparison of outer loop predictions of POD weights, where (1^{st} and 2^{nd} row): 5 - HL and 7 - HL Auto-Encoder (AEN) and (3^{rd} and 4^{th} row): 4 - HL and 6 - HL Deep Koopman Network (DKN) for Buoyant Boussinesq Mixing Flow

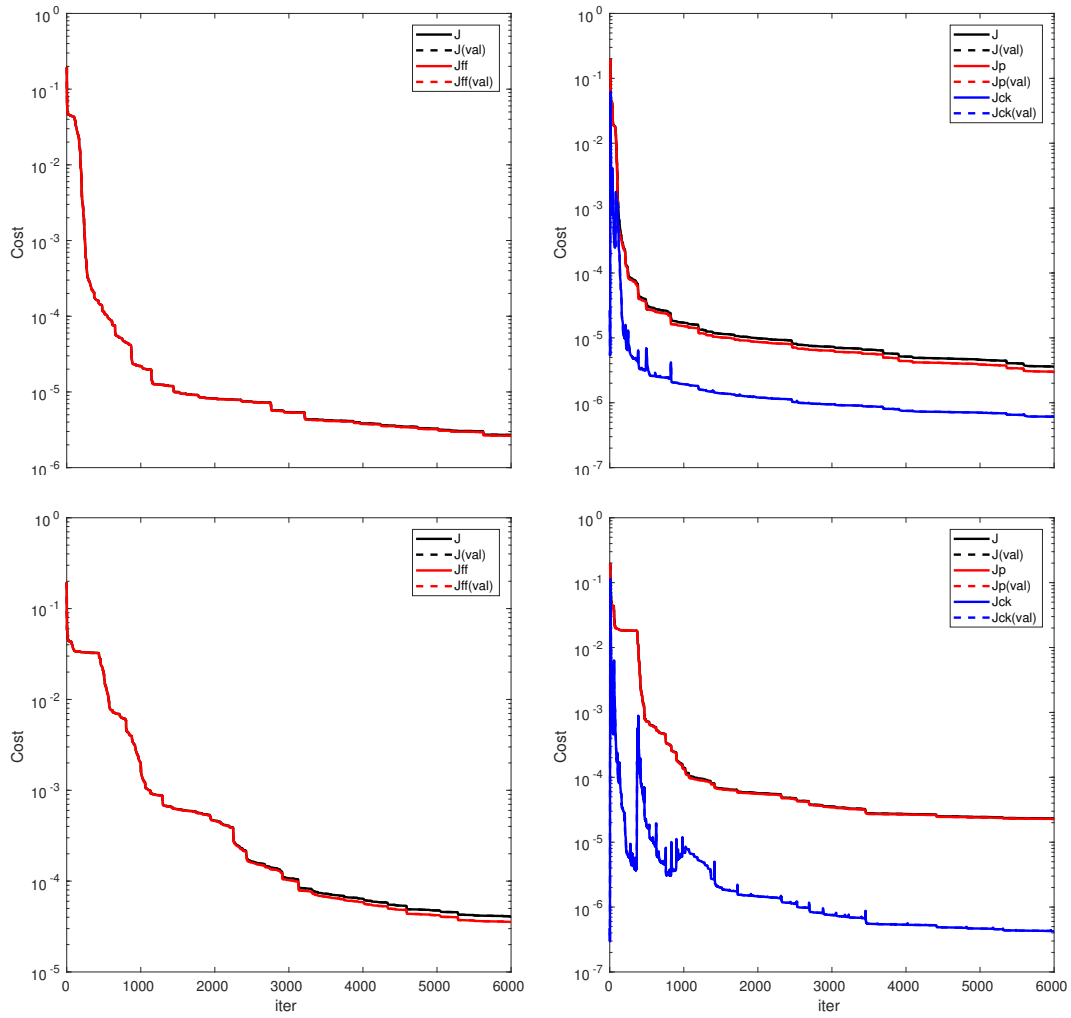


Figure 3.10: Plot of cost function minimization with respect to number of iterations, (top:left) 5 - HL AEN, (top: right) 4 - HL DKN , (bottom:left) 7 - HL AEN and (bottom: right) 6 - HL DKN for Buoyant Boussinesq Mixing Flow

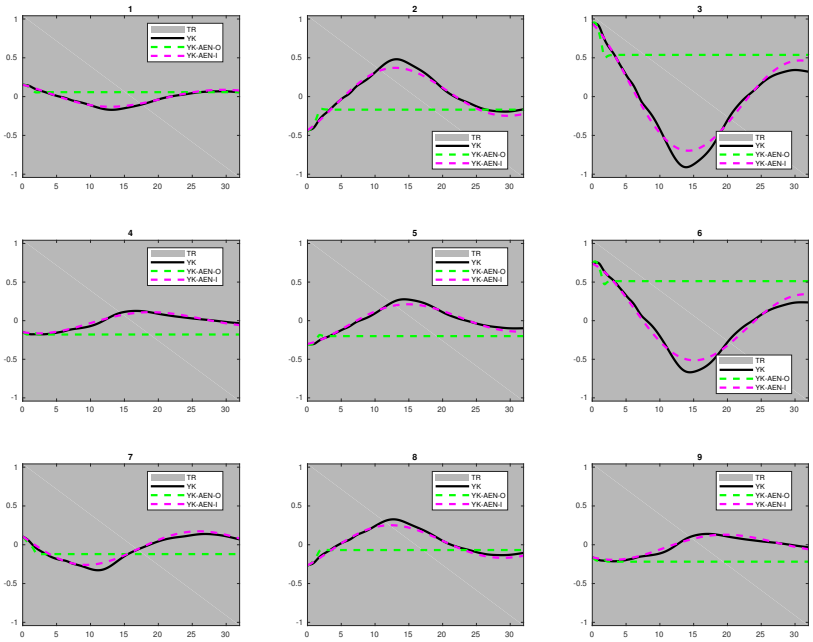


Figure 3.11: Comparison of inner loop vs outer loop predictions in Koopman space of 7 - HL Auto-Encoder (AEN) for Buoyant Boussinesq Mixing Flow

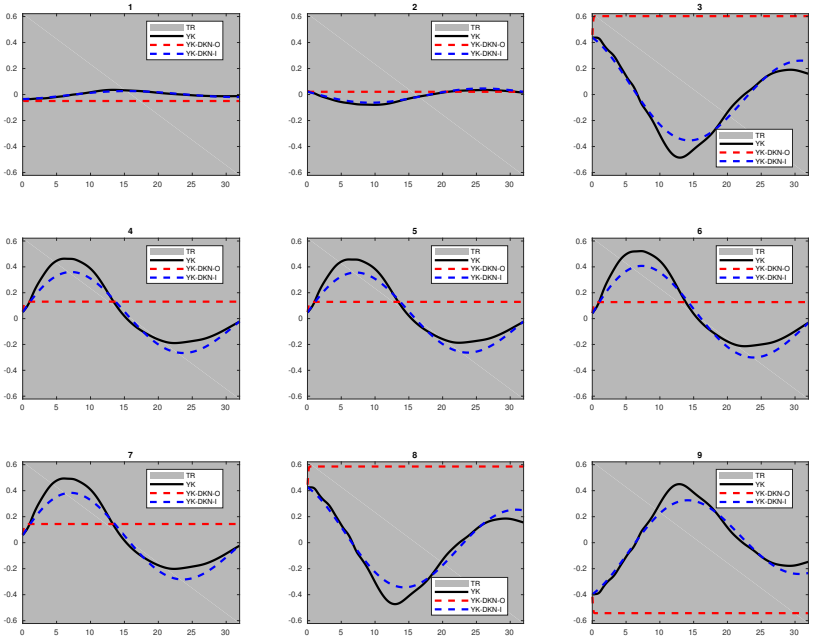


Figure 3.12: Comparison of inner loop vs outer loop predictions in Koopman space of 6 - HL Deep Koopman Network (DKN)for Buoyant Boussinesq Mixing Flow

Train		DMD		EDMD		6-DKN	8-DKN	7-AEN	9-AEN
cycles(BBM)	$\mathcal{E}$	$N=3$	$N=10$	$N=3$	$N=10$	$N=3$	$N=3$	$N=3$	$N=3$
FD: 1 m=1600	$\mathcal{E}_{p-o}$	0.0249	0.013	$8.7\ e^{-3}$	$4.17\ e^{-7}$	0.7847	0.4737	0.5103	0.4729
	$\mathcal{E}_{p-i}$	0.0249	0.013	$8.7\ e^{-3}$	$4.17\ e^{-7}$	$2.5\ e^{-2}$	$9.6\ e^{-3}$	$2.1\ e^{-2}$	$9.7\ e^{-3}$
	$\mathcal{E}_a$	$--$	$--$	$--$	$--$	$9.6\ e^{-7}$	$9.12\ e^{-6}$	$3.5\ e^{-7}$	$1.16\ e^{-6}$
cycles(CW)	$\mathcal{E}$	$N=3$	$N=55$	$N=3$	$N=55$	$N=3$	$N=3$	$N=3$	$N=3$
LC: $16-20$ m=124	$\mathcal{E}_{p-o}$	$7.4\ e^{-3}$	$6.06\ e^{-5}$	$3.51\ e^{-3}$	$--$	$4.5\ e^{-2}$	$1.5\ e^{-2}$	$3.4\ e^{-3}$	$3.0\ e^{-3}$
	$\mathcal{E}_{p-i}$	$7.4\ e^{-3}$	$6.06\ e^{-5}$	$3.51\ e^{-3}$	$--$	$8.3\ e^{-3}$	$7.8\ e^{-3}$	$5.5\ e^{-3}$	$6.9\ e^{-3}$
	$\mathcal{E}_a$	$--$	$--$	$--$	$--$	$7.6\ e^{-6}$	$3.43\ e^{-6}$	$8.6\ e^{-7}$	$9.3\ e^{-8}$
TR-I: $8-20$ m=372	$\mathcal{E}_{p-o}$	0.5129	$4.8\ e^{-2}$	0.776	$8.3\ e^{-5}$	0.1168	0.6	1.074	0.486
	$\mathcal{E}_{p-i}$	0.5129	$4.8\ e^{-2}$	0.776	$8.3\ e^{-5}$	0.521	0.5864	0.5109	0.478
	$\mathcal{E}_a$	$--$	$--$	$--$	$--$	$9.8\ e^{-6}$	$7.35\ e^{-6}$	$1.7\ e^{-7}$	$2.05\ e^{-7}$
TR-II: $4-16$ m=372	$\mathcal{E}_{p-o}$	0.5538	0.7138	0.6831	1.76988	0.4438	0.8078	0.7429	0.727
	$\mathcal{E}_{p-i}$	0.5538	0.7138	0.6831	1.76988	0.4313	0.4437	0.4323	0.4343
	$\mathcal{E}_a$	$--$	$--$	$--$	$--$	$8.6\ e^{-6}$	$1.2\ e^{-5}$	$2.07\ e^{-7}$	$2.4\ e^{-7}$

Table 3.2: Prediction error estimates for layer-wise local (MLOC) and global optimization based Koopman networks for $Re = 100$.

3.2.3 Transient Cylinder Wake Dynamics Predictions

In this section, we have presented the analysis based on predictions of flow over cylinder in the transient regions TR-I and TR-II. In Fig. 3.13, we can see that all the cost function minimize to an order of $1e^{-4}$, while the encoder-decoder error is minimized to an order of $1e^{-6}$. From the cost minimizations in DKN, we can infer that there is a strong coupling between cost of learning the encoder-decoder and Koopman operator. Even though the cost of penalty network was lower, the error of Koopman operator learning was very high, but as the J_{ff} started decreasing and the J_{ckt} started increasing and then both started decreasing, this behavior signals to a strong coupling between cost of learning the encoder-decoder (J_{ckt}) and Koopman operator (J_{ff}). In Fig. 3.14, the predictions obtained from the inner loop does not yield good results, on the contrary to the observations in previous section 3.2.2, the outer loop predictions Fig. 3.15 yield better results than the inner loop predictions. This dichotomy in prediction results between two different flow physics signal more intricate coupling between the encoder-decoder and Koopman operator. By comparing the Koopman operators from the both the Buoyant mixing flow and Transient cylinder wake, we

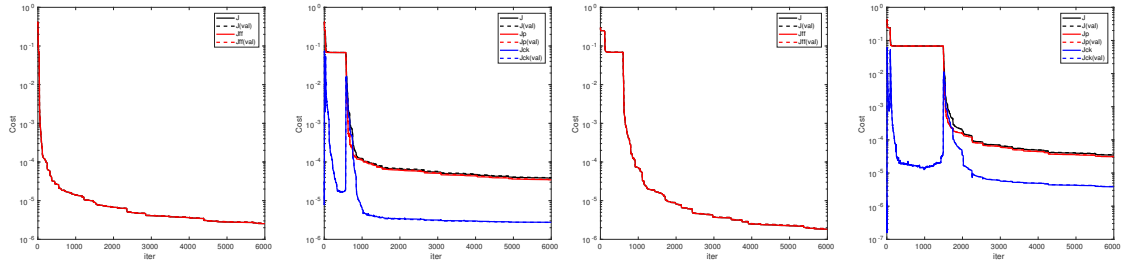


Figure 3.13: Plot of cost function minimization with respect to number of iterations, (top:left) 5 - HL AEN, (top: right) 4 - HL DKN , (bottom:left) 7 - HL AEN and (bottom: right) 6 - HL DKN

can see that there is no correlation. Further, here we would like to highlight that the fact that the magnitudes of Koopman operators found from AEN for transient cylinder are sparse and high, could be a sign of over fitting. Further, when we change the training data to transient region II we see that the two step AEN performs poorly as a result of over fitting(see Fig. 3.17), while the the DKN performs excellently. In Fig. 3.18 the Koopman operators of training region are plotted, we can see that here again we see that the magnitudes of AEN Koopman operator is both sparse and high.

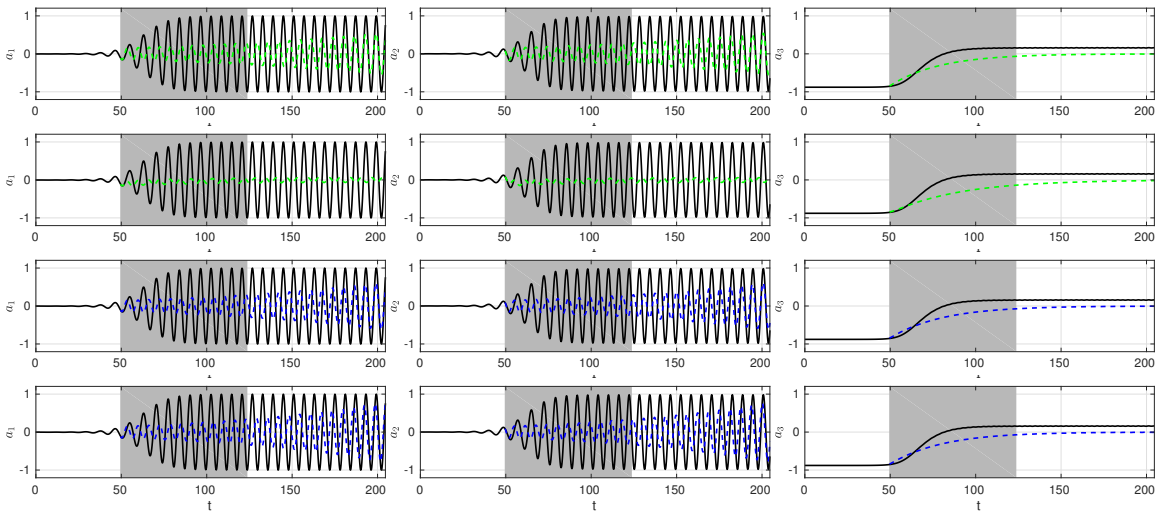


Figure 3.14: Comparison of inner loop predictions based POD weights, where (1^{st} and 2^{nd} row): 5 - HL and 7 - HL Auto-Encoder (AEN) and (3^{rd} and 4^{th} row): 4 - HL and 6 - HL Deep Koopman Network (DKN)

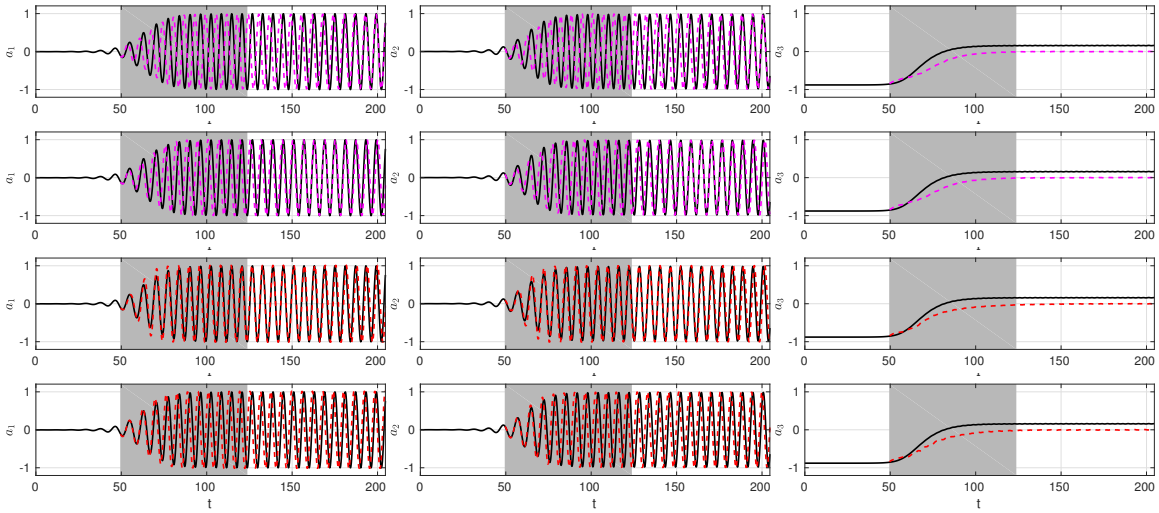


Figure 3.15: Comparison of outer loop predictions of POD weights, where (1^{st} and 2^{nd} row): 5 - HL and 7 - HL Auto-Encoder (AEN) and (3^{rd} and 4^{th} row): 4 - HL and 6 - HL Deep Koopman Network (DKN)

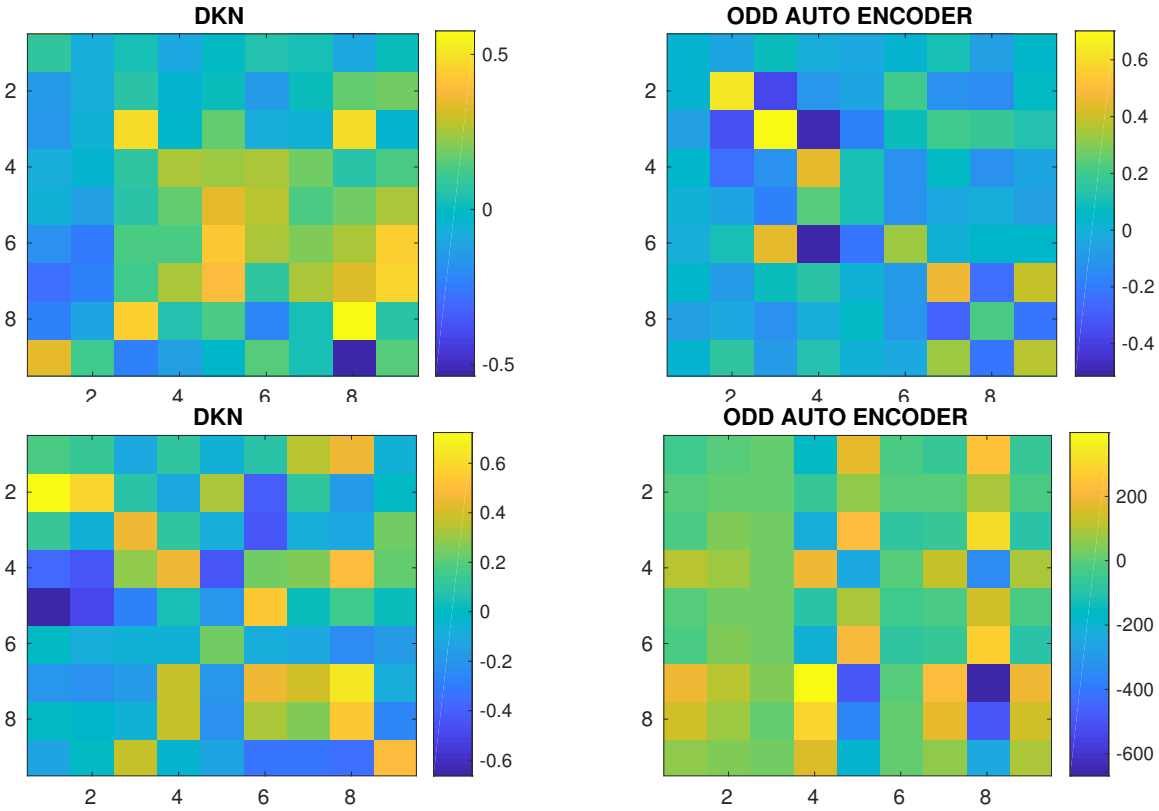


Figure 3.16: Comparison of the Koopman operator obtained from the 6 - HL - DKN and 7 - HL - AEN (first row) Buoyant Boussinesq Mixing Flow and (second row) Cylinder flow

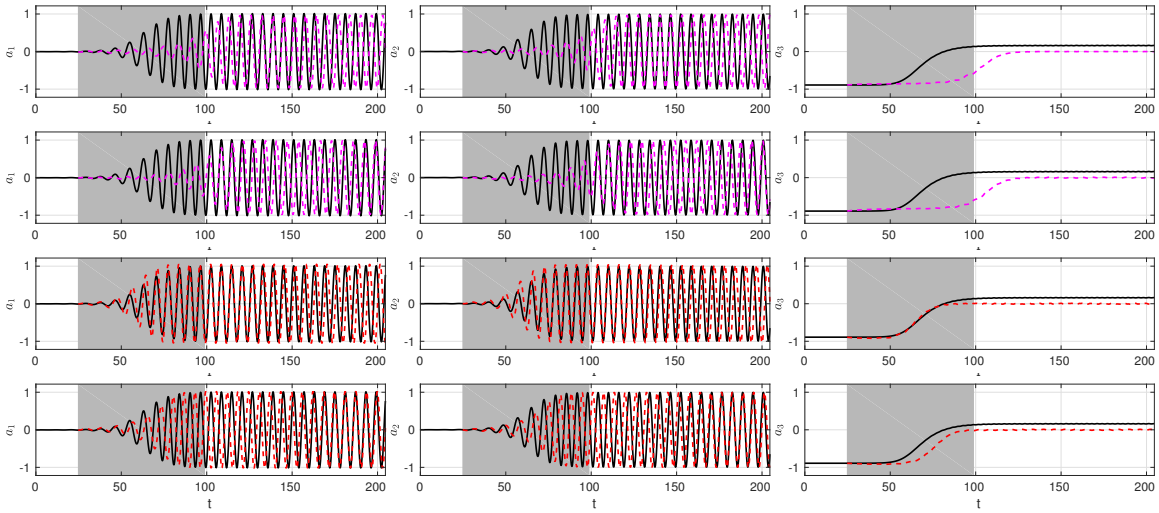


Figure 3.17: Comparison of outer loop predictions of POD weights, where (1^{st} and 2^{nd} row): 5 - HL and 7 - HL Auto-Encoder (AEN) and (3^{rd} and 4^{th} row): 4 - HL and 6 - HL Deep Koopman Network (DKN)

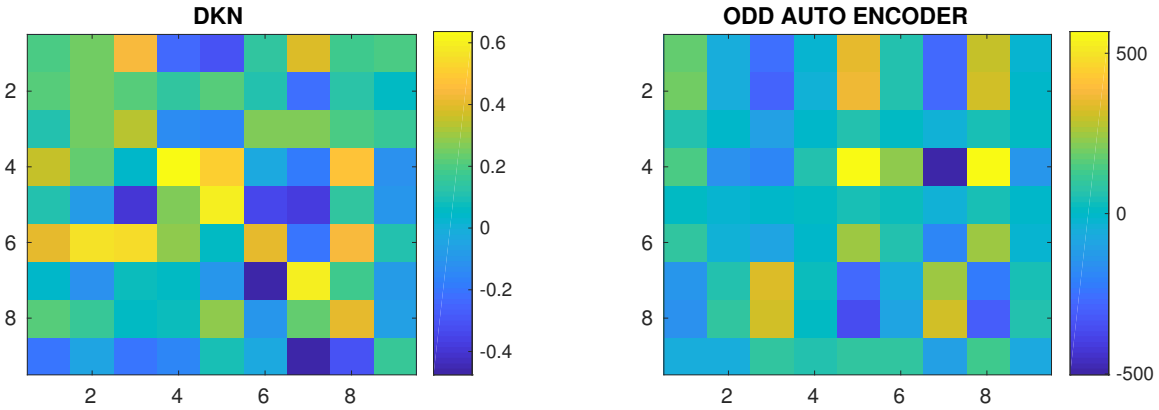


Figure 3.18: Comparison of the Koopman operator obtained from the 6 - HL - DKN and 7 - HL - AEN of Cylinder flow

3.3 Summary

In this section summary of the prediction based analysis of the Deep Koopman Networks is presented. The following are the important observations based on which we have derived preliminary conclusions:

1. Koopman Networks (DKN and AEN) where used as symmetric Markov Linear methods where in both the learning of the encoder-decoder are learnt with the

linear Koopman operator.

2. We have seen from Limit cycle data of cylinder flow and Boussinesq Buoyant mixing flow that symmetric network helps in learning and improves prediction capabilities.

3. Even with the encoder-decoder error driven to atleast $1e^{-6}$, the predictions using inner loop with cylinder flow are still not predicted very well.

4. Further investigation is required on the outer loop and inner loop predictions for transient cylinder flows. The DKN method with its outer prediction is able to learn the nonlinear dynamics/ transient region of the cylinder flow POD weights, on the other hand a two step process with the same encoder-decoder with inner prediction does not yield promising results.

5. Further investigation is needed to concretely establish the differing trends with different physical nature of the flow.

CHAPTER 4

Deep Koopman Networks: Spectral Analysis

Extraction and analysis of dynamical characteristics from data have two broad varieties that either use the raw data ensemble arranged as a matrix or by learning a linear transition operator when time-resolved data is available. Both these approaches provide very different information of the physics represented by the data. An example of the former are the class of model decomposition techniques that leverage the extent of correlation in the snapshots of data such as the proper orthogonal decomposition or more specifically, spatial POD (Lumley 2007, Holmes 2012, Taira et al. 2017). This approach is also known by other names such as principal component analysis (PCA) and singular value decomposition (SVD). Spatial POD modes represent spatial coherent structures as they are essentially eigenvectors of the spatial two-point correlation tensor of the data and arranged in terms of energy content. Consequently, each spatial POD mode is energetically grouped instead of grouping by temporal scale (frequency) and may contain multiple temporal frequencies. To achieve this frequency-based segregation or decomposition, one can transform the data into spectral space, organize them into frequency bins or groups and then perform a spatial POD analysis. This approach will provide frequency dependent spatial POD modes and is referred to as spectral POD (Towne et al. 2018). The other class of methods leverage dynamical operators by combining spectral analysis with linear time-invariant (LTI) description of the physics. Just as linear operator based control is often preferred on account of the established literature, expertise and track record of success (Kim & Bewley 2007, Rowley & Dawson 2017) for fluid flows, flow analysis has also been heavily linked

to linear system theory (Rowley et al. 2009, Rowley & Dawson 2017, Schmid 2010, Williams et al. 2015, Lu et al. 2018) due to interpretability of the products of spectral analysis.

The success the Koopman approximation framework is tied to two aspects: (i) the ability of the projection or convolutions to the feature space (Rowley & Dawson 2017, Taira et al. 2017, Lu et al. 2018) to accurately map data without loss of information while incorporating the appropriate degree of nonlinearity and (ii) their ability to capture the evolution of the dynamics in the feature space (Lu et al. 2018). In general satisfying (i) enables satisfying (ii). Therefore, one needs to learn the unknown transformation g that is appropriate for a given system and is often modeled from phenomenology or assumed to be spanned by a chosen basis space (POD, Fourier or Gaussian) or both. Invariably, the choice of basis space corresponds also to the method of choice, i.e., DMD algorithms typically use POD basis space and Evolving Gaussian Process (EGP) methods (Csató & Opper 2002, Williams & Rasmussen 1996) leverage Gaussian kernels. Alternately, functional regression based on sparsity promoting l_1 minimization algorithms have also been explored (Brunton et al. 2016). These approaches provide the sparsest basis space to approximate the transformation to the Koopman space as long as one adopts the appropriate library of candidate functions. When the library is insufficiently populated, the alternate strategy is to build nonlinear convolution operators by layering multiple 'elementary' convolutions (Lu et al. 2018). Extended DMD or EDMD (Williams et al. 2015) incorporates a second convolution (over a layer of SVD convolution) by embedding nonlinear functions. This approach builds a hybrid data-functional form of the convolution and is effective if one knows the nature of the nonlinearity *a priori*, but often results in a very high-dimensional function dictionary to approximate the data accurately. The precise nature of the nonlinearity is normally not known *a priori*. The kernel variant of this method, KDMD (Williams et al. 2014) helps with dimensionality reduction,

but is once again limited by the choice of the kernel function. A key limitation of all the above multilayer methods is that each of the layers are treated independently and the map is learned using local criteria, i.e. by direct function evaluation or projection onto a basis space that is optimal with respect to local features. As shown in (Puligilla & Jayaraman 2018*b*), this severely limits the ability of the convolution map to represent the dynamics for a given layer dimension. Feedforward neural networks (FFNNs) provide an intriguing alternative as global optimal convolution architectures and are effective Markovian prediction tools (Puligilla & Jayaraman 2018*b*), but are purely forward maps, i.e they do not use symmetric convolutions which prevents learning of a Markov linear operator. To address this, the authors have developed a modified neural network architecture termed as *Deep Koopman Neural Networks (DKN)* (Puligilla & Jayaraman 2018*a*, Jayaraman & Puligilla 2018) that enforce symmetry and learn the Koopman operator simultaneously by combining two FFNNs trained in sync. An alternative is to leverage deep neural networks (DNNs) to learn multilayer convolution maps from data through the use of symmetric deep auto-encoder networks or AENs (Hinton & Salakhutdinov 2006) that invariably allow for dimensionality reduction and encoding data into a transformed 'feature' space. Both these approaches are similar in using deep learning to compute data-driven embeddings, but the former learns the corresponding Koopman operator approximation simultaneously while learn the mapping. g. In the second approach, the Koopman (Markov linear) operator is learned in a separate step. Both these approaches perform similarly in most cases, but the DKN produced robust long-time predictions that has implications for data-driven modeling.

4.1 Spectral Analysis

In the Koopman framework, the feature space is the observable space and the feature maps are observable functions. The operator theoretic view interprets $\mathcal{K}$ as operating

on the observable space (Williams et al. 2015) $\mathcal{K} : \mathcal{F} \rightarrow \mathcal{F}$. When $\boldsymbol{g}$ and $\boldsymbol{h}$ are identical, then the linear operator $\mathcal{K}$ evolves the Markovian dynamics in Eq. (2.2) as a Koopman evolutionary model given by:

$$\mathcal{K}\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{y}) = g(\boldsymbol{F}(\boldsymbol{x})). \tag{4.1}$$

Since, the Koopman operator has the effect of operating on the functions of state space as shown in Eq. (4.1), it is commonly referred to as a composition operator where $\circ$ represents the composition between $\boldsymbol{g}$ and $\boldsymbol{F}$.

$$\mathcal{K}\boldsymbol{g} = \boldsymbol{g} \circ \boldsymbol{F}. \tag{4.2}$$

Being a linear operator, the products of Koopman spectral analysis such as the eigenfunctions (ϕ_j), eigenmodes/ eigenfunctions (v_j) and eigenvalues (μ_j) can be leveraged to reconstruct the transformation $\boldsymbol{g}(\boldsymbol{x})$ as shown in Eqs. (4.3)-(4.4). This suggests that the transformation $\boldsymbol{g}$ should be spanned by Koopman eigenfunctions $\boldsymbol{\phi}$.

$$\boldsymbol{g}(\boldsymbol{x}) = \sum_{j=1}^{\infty} \phi_j \boldsymbol{v}_j \tag{4.3}$$

$$\boldsymbol{g}(\boldsymbol{y}) = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}) = \sum_{j=1}^{\infty} \phi_j \boldsymbol{v}_j \mu_j \tag{4.4}$$

In this discussion, we consider sequential snapshots of data separated by a constant time-step and denoted by $(\boldsymbol{x}^i, \boldsymbol{x}^{i+1} \dots)$ such that the dynamical system is given by:

$$\boldsymbol{x}^{i+1} = \boldsymbol{F}(\boldsymbol{x}^i) \tag{4.5}$$

The corresponding Koopman representation is:

$$\boldsymbol{g}(\boldsymbol{x}^{i+1}) = \boldsymbol{g}(\boldsymbol{F}(\boldsymbol{x}^i)) = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}^i) \tag{4.6}$$

where, $\boldsymbol{g}$ is some convolution operator yet to be identified (or assumed for a given model). We can split this sequence of snapshots into pairs as $X = (\boldsymbol{x}^1 \dots \boldsymbol{x}^{M-2}, \boldsymbol{x}^{M-1})$ and $Y = (\boldsymbol{x}^2 \dots \boldsymbol{x}^{M-1}, \boldsymbol{x}^M)$ such that $(X, Y) \in \mathbb{R}^{N \times M}$, then the dynamical system in Eq. (4.5) can be represented as

$$Y = \boldsymbol{F}(X) \tag{4.7}$$

and its quasi-linear form is

$$Y \approx \boldsymbol{A}(X)X. \tag{4.8}$$

Here, $\boldsymbol{A}(X) \in \mathbb{R}^{N \times N}$ is the quasi-linear operator describing the evolution of the discrete dynamical system that we are trying to approximate from data. Where N, M represent the dimensions of the instantaneous system state and the number of available snapshots respectively. Typically, $N \gg M$ and for a nonlinear system $\boldsymbol{A}(X)$ evolves with X.

4.1.1 Koopman Modes and Eigenvalues

Once the A matrix is found we can extract the spectral information via the eigenvalue analysis as follows:

$$\boldsymbol{A}\ \boldsymbol{v}_j = \mu_j\ \boldsymbol{v}_j \tag{4.9}$$

where, $\boldsymbol{v}_j$ are the Koopman modes and μ_j Koopman eigenvalues. The growth rate and the discrete frequency content from the Koopman eigenvalues can be extracted via

$$\lambda_j = \frac{ln(\mu_j)}{2\pi\Delta t} \tag{4.10}$$

4.2 Results

In this section, we perform spectral analysis of nonlinear cylinder wake flow with dynamics including a limit-cycle attractor and an unstable equilibrium. It is well

known that beyond the critical Reynolds number,the unstable equilibrium is disturbed by an axi-symmetric perturbation which saturates into a limit-cycle behavior. In this paper we assess the ability of the locally optimal Koopman approximation methods such as DMD & EDMD and globally optimal DKNN & AEN to capture the dominant Koopman spectrum and model structures for the different flow regimes. In this work, we analyze Koopman spectra and modes from different data regimes and assess their impact on long-time limit-cycle predictions. In this chapter we have concentrated our analysis on 6-HL Deep Koopman Network(DKN) and 7 - HL AutoEncoder Network (AEN) for brevity. Further, we have used the MLOC methods (DMD and EDMD) with POD weights that sum to 100% energy for clear comparison. The predictions obtained from such a system are presented in Fig. 4.1, we can see that the predictions from DKN and AEN are comparable to the predictions obtained from full set data.

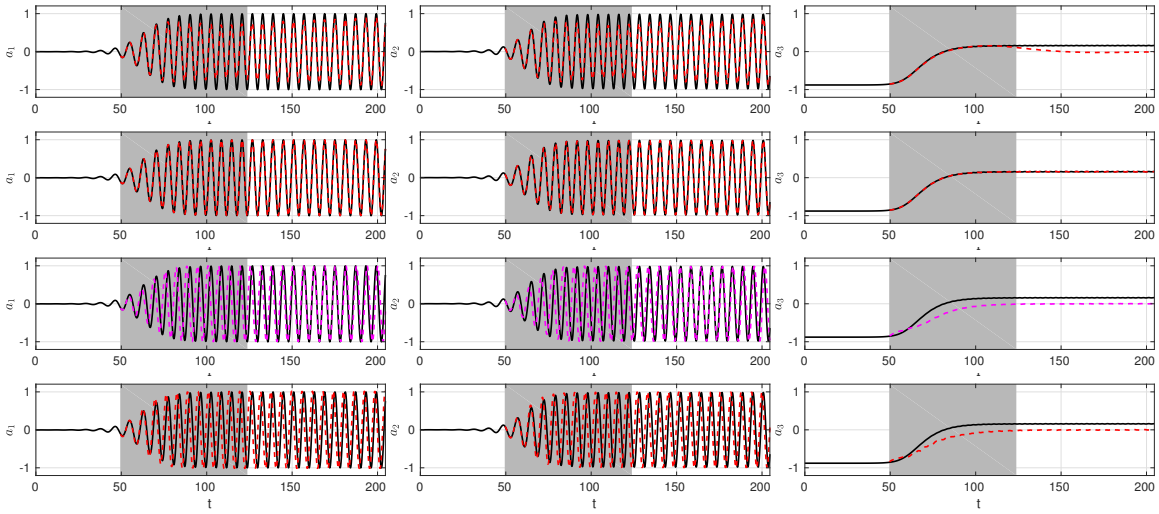


Figure 4.1: Comparison of outer loop and MLOC predictions of POD weights, where (1^{st} and 2^{nd} row): DMD and EDMD with 55 POD weights, (3^{rd} and 4^{th} row): 6 - HL DKN and 7 - HL AEN with 3 POD weights

4.2.1 Limit Cycle Dynamics of Cylinder Flow

Using the Koopman operators used in the previous chapter 3 for predictions of POD weights in the limit cycle region, we will extract the Koopman eigenvalues and Koopman modes. With $\Delta t = 0.2D/u_\infty$, the eigenvalues of the Koopman operator obtained

for all the methods used as part of this study are tabulated in table 4.1. In the table, only the eigenvalues that correspond to the limit cycle are only listed. We can see that, the first three frequencies of the POD weights are captured by DMD, but 4-HL DKN and 4-HL AEN where able to capture the mean mode and the first mode, while other modes did not fall in the vicinity of the higher frequencies. But, it should be noted here that the predictions obtained from Koopman operator accurately predicted the Limit cycle dynamics(see Figs. 3.4 and 3.5). It interesting to see that even though the specific frequency is absent from the eigenvalues spectrum of the Koopman operator, the dynamics are accurately represented. In Fig. 4.2 and 4.3 the Ritz plot of the eigenvalues are plotted along with eigenvalues obtained from DMD(MLOC) method. From the figures, we can see that the the μ_0 the mean mode and μ_1 from both DKN and AEN over lap with μ's of DMD, but all other eigenvalues from the DKN and AEN are damped or decaying. A growth rate vs. frequency plot of the AEN and DKN for limit cycle are plotted in Figs. 4.4, here again we can see that the growth rates and frequencies of λ's are accurately predicted.

	POD	DMD	DKN-4	AEN-4	DKN-6	AEN-6
$\mu(St = 0)$	$1 \pm 0i$	$1.001 \pm 0i$	$1.001 \pm 0i$	$1 \pm 0i$	$1.001 \pm 0i$	$1.001 \pm 0i$
λ_μ	0	0.00079	0.00079	0	0.00079	0.00079
$\mu(St = 0.1655)$	$0.9784 \pm 0.2065i$	$0.9786 \pm 0.2064i$	$0.9786 \pm 0.2059i$	$0.9784 \pm 0.2059i$	$0.9786 \pm 0.206i$	$0.9786 \pm 0.2059i$
λ_μ	$0 \pm 0.1655i$	$0.0001 \pm 0.1654i$	0 ± 0.1652	$-0.0001 \pm 0.1651i$	$0 \pm 0.1651i$	$0 \pm 0.1651i$
$\mu(St = 0.331)$	$0.9147 \pm 0.4041i$	$0.9148 \pm 0.404i$	$--$	$--$	$--$	$0.6078 \pm 0.4056i$
λ_μ	$0 \pm 0.331i$	$0 \pm 0.3309i$	$--$	$--$	$--$	$-0.2497 \pm 0.4683i$
$\mu(St = 04965)$	$0.8115 \pm 0.5843i$	$0.8117 \pm 0.5841i$	$--$	$--$	$--$	$--$
λ_μ	$0 \pm 0.4966i$	$0 \pm 0.4964i$	$--$	$--$	$--$	$--$

Table 4.1: Koopman eigenvalues(μ), growth rate ($Re(\lambda)$) and discrete frequencies ($Im(\lambda)$) obtained from Limit cycle dynamics of cylinder wake flow.

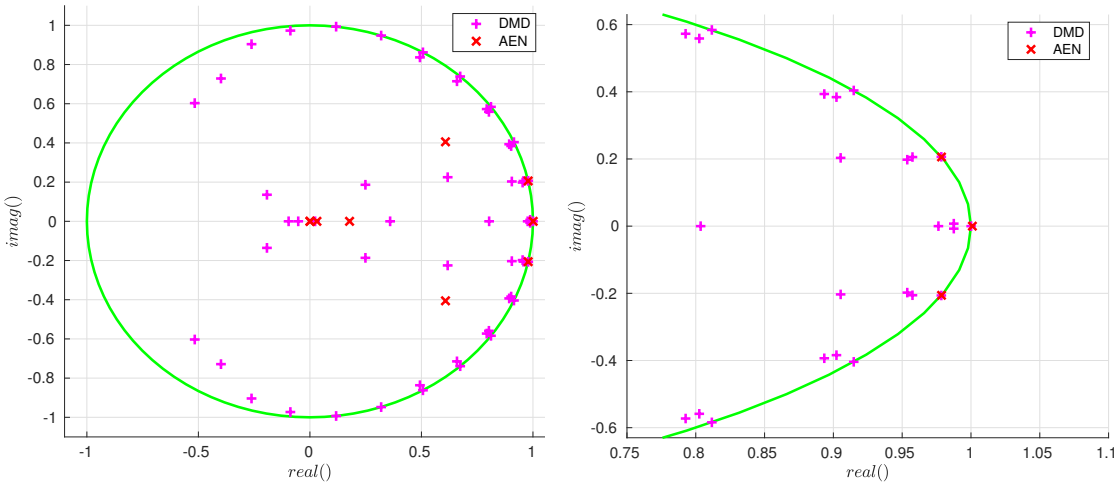


Figure 4.2: Ritz plot of eigenvalues(μ) spectrum obtained from 7 - HL AEN

4.2.2 Identification of Dominant and Allied Structures via Dynamic Mode Decomposition

In this section, identification of the important modal structures of Koopman modes are studied understand the importance of these structures in reconstruction/prediction of the flow field. To do this, we have identified the eigenvalues at certain frequencies from the growth rates obtained from DMD and EDMD. In the Fig. 4.5(1^{st}), we have identified the the frequencies of the periodic modes or coherent structures that are dominant and the allied structures in the vicinity of the periodic modes. For example, the first three modes in the positive x-axis correspond to the mean mode ans its allied structures. Similarly, the next three mode cluster correspond to the first harmonic and its allied structures. In Fig. 4.5, we can see that the eigen spectrum is discrete, we have also looked at the eigen spectrum obtained from EDMD. The growthrates and their respective frequencies are plotted in Fig. 4.5(2^{nd} row), here we can see that the eigen spectrum is continuous. In Figs. 4.6 and 4.7, we have plotted the Koopman modes of the identified growth rates and frequencies. Here, we have plotted the Koopman modes in accordance with the appearance/increase of frequency

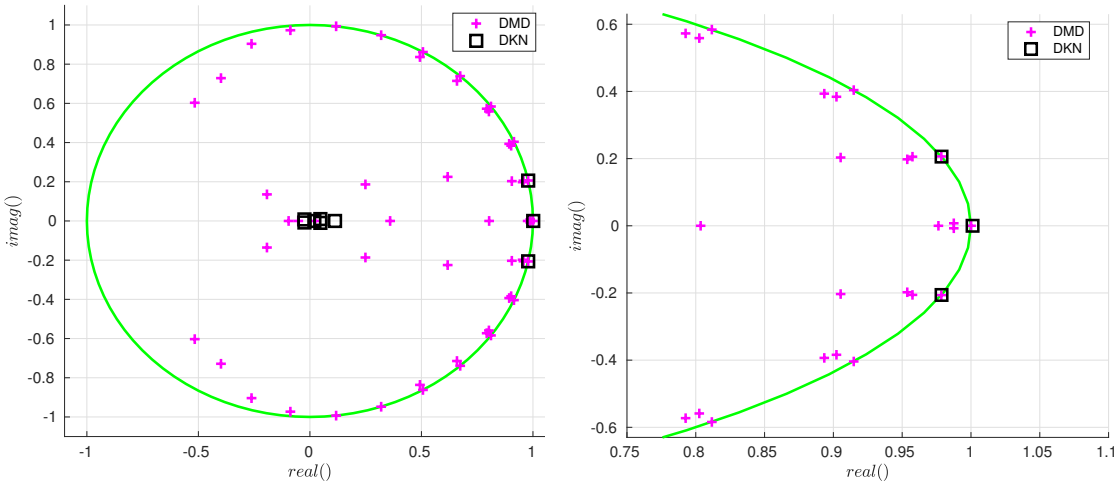


Figure 4.3: Ritz plot of eigenvalues(μ) spectrum obtained from 6 - HL DKN

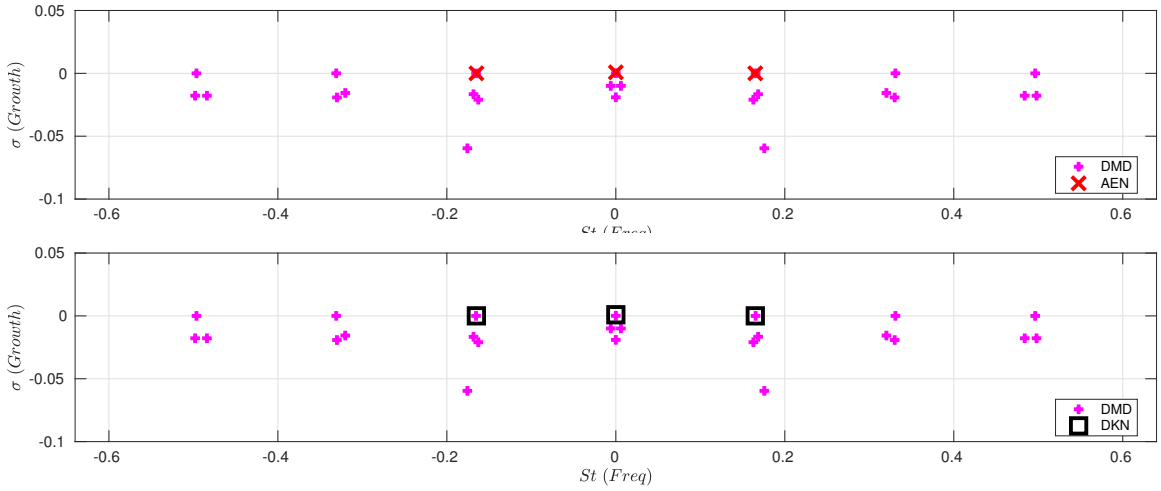


Figure 4.4: Comparison of growth rate and frequencies of (1^{st} row)7 - HL Auto-Encoder(AEN) and (2^{nd} row): 6 - HL Deep Koopman Network (DKN)

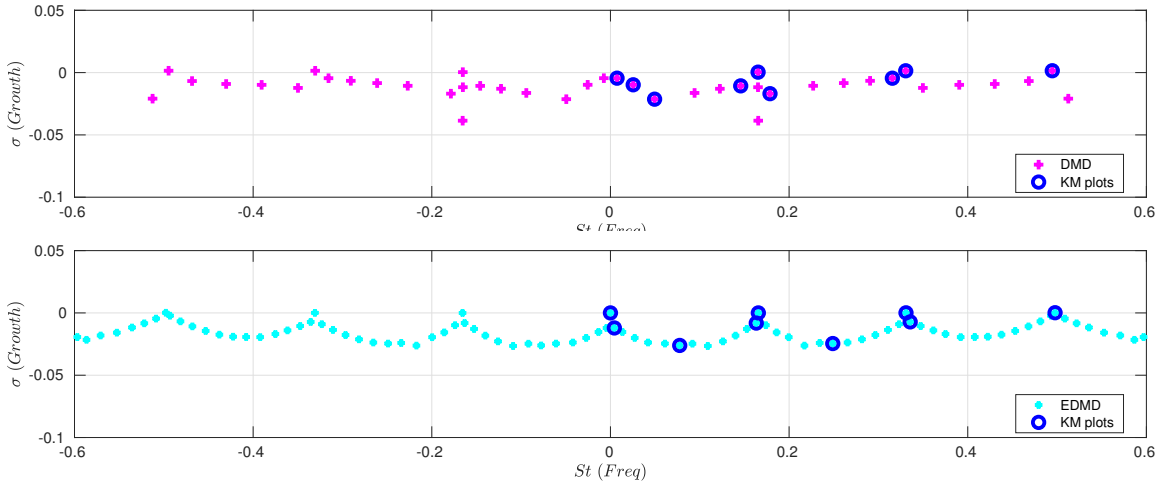


Figure 4.5: Comparison of select growth rate and frequencies of (1^{st} row)DMD and (2^{nd} row): EDMD for understanding the Koopman mode structures.

value. For example, the mean mode for both DMD and EDMD are plotted at first and the followed its allied structures.

The Koopman mode identified, from both DMD and EDMD have similar structures and are representative of the eigenvalues and frequencies. We can see that, although DMD spectrum is not continuous, the Koopman modes have similar structures in the Koopman modes. More detailed analysis is required in terms of identifying the importance of allied structures on the predictions and reconstruction. We have used these structures to make important observations while evaluating the performance of the DKN and AEN methods.

4.3 Spectral Analysis of Transient Dynamics of Cylinder Flow

The eigenvalue spectrum of limit cycle dynamics is rather straight forward and can be easily computed using DMD like methods. The transient dynamics are not captured very well using the conventional DMD, an additional nonlinear mapping is required to learn/capture the dynamics. Here we use EDMD with 55 POD weights and a second order polynomial expansion mapping is used. In Fig. 4.8, the approximate Koopman operators are plotted as image pixels to show the structure of the operator

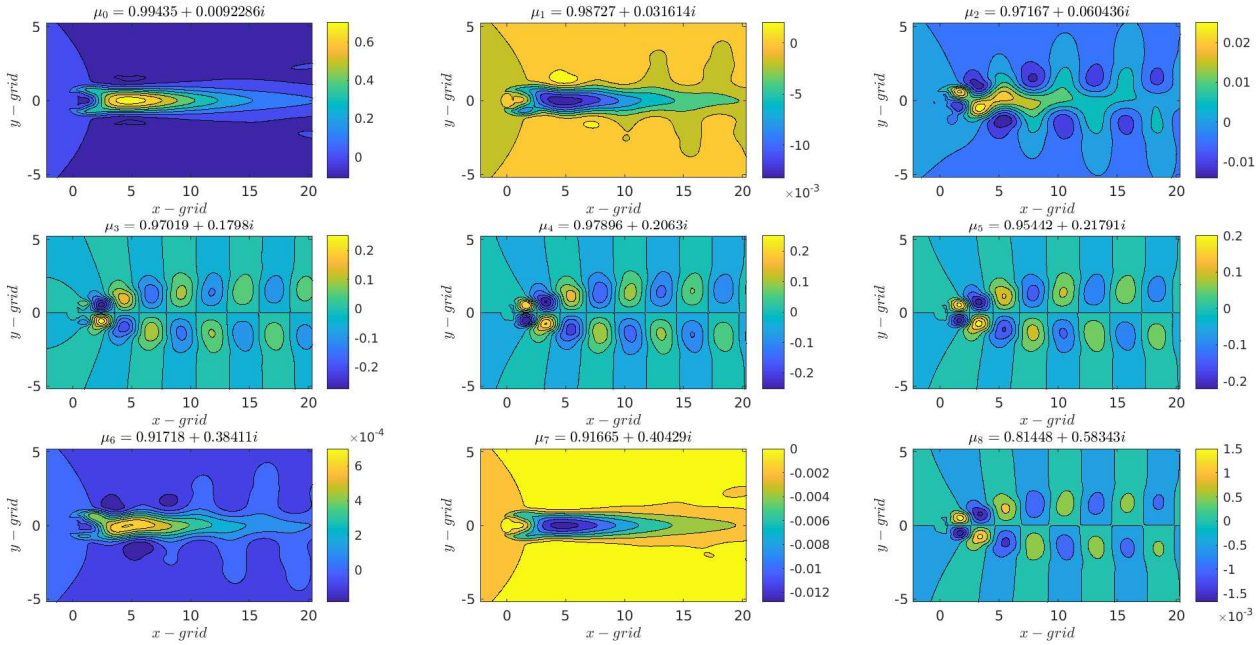


Figure 4.6: Koopman Eigen functions obtained using DMD with 55 POD weights

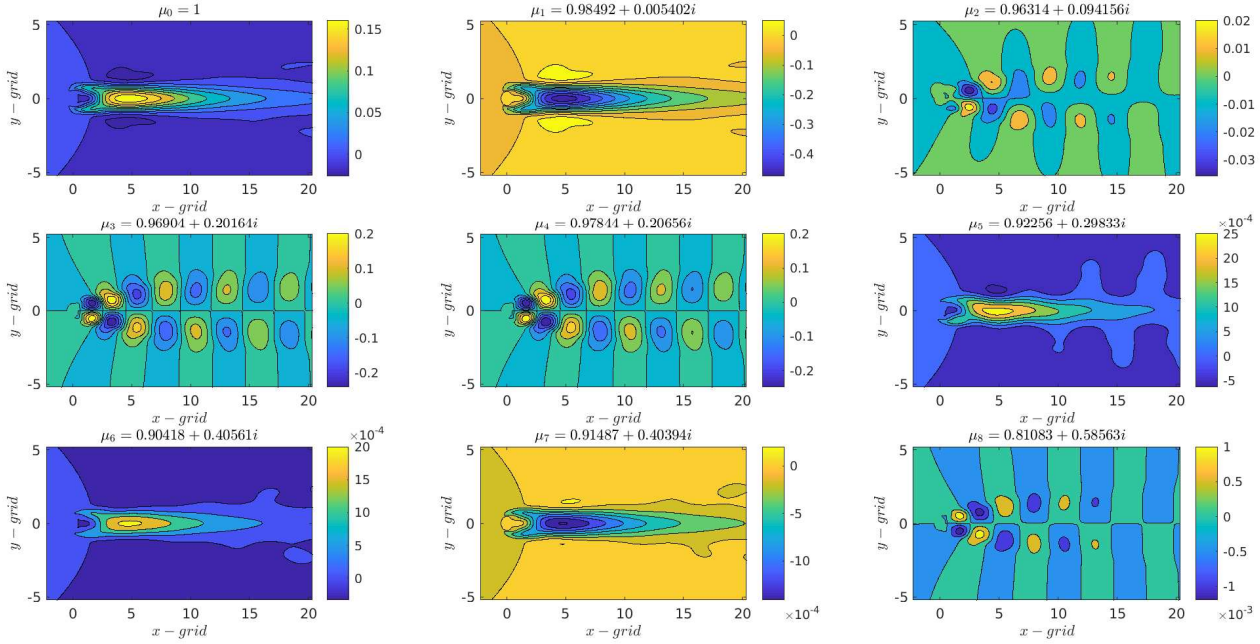


Figure 4.7: Koopman Eigen functions obtained using EDMD with 55 POD weights and 2^{nd} order polynomial mapping

matrix. We can see that, the MLOC methods produce a diagonally dominated matrix and the MGOC methods produce matrices which do not have any clear pattern. But with closer look at the Koopman operator of DKN, we can see it also has diagonally dominated matrix which signifies the features in the Koopman space are correlated. On the other hand the, the Koopman operator is sparse and spiky, which hints might operator be overfitted or the features in the Koopman space are not correlated. All the eigenvalues of the AEN operator lie nearby the eigenvalues μ_0, μ_1, and μ_2 as seen in Fig. 4.9, but the μ_0, μ_1 eigenvalues found from the DKN are overlapping with the eigenvales of DMD and EDMD(see Fig. 4.10). The rest of the eigenvalues from the DKN are decaying and are near by to zero. In case of AEN, the eigenvalues in the vicinity of the μ_0, μ_1, and μ_2 create a combined effect to capture the nonlinear dynamics of the transient region. But it is not very clear from the DKN results, how the transient dynamics are captured. Further investigation is need to assess the quality of the modes obtained from the DKN Koopman operator. In Figs. 4.11, we can see that the DKN and AEN are able to capture the mean eigen values at frequency $St = 0$ and first eigenvalues at $St = \pm 0.1655$. In the table 4.2 the eigenvalues and the frequencies of obtained from all the methods used as part of this study. We can see that the DKN with both 4-HL and 6-HL manages to find the the mean eigenvalue and the first and second eigenvalue accurately. On the other hand, the AEN with 5-HL and 7-HL manages to find the eigen values in the vicinity of the first three most dominating and the mean eigen value.

From the Figs. 4.6 and 4.7, we can see that the dominating modes with higher magnitudes are μ_0 and mu_1 in both DMD and EDMD methods. While μ_2 and μ_3 are two orders less than the dominant eigenfunctions. The structures of the eigenfunctions also signify their nature, for example the eigenfunctions associated with μ_1 and μ_2 are periodic and signify the dominant coherent structure in the flow. While the μ_0 signifies a mean, which indicates the mode doesn't evolve with time. The third mode μ_3 is the

shift mode (Noack et al. 2003). Now by looking at the eigen functions obtained from the DKN and AEN. In Fig. 4.12, we can see that except for the mean mode μ_0 all other modes looks like a combination of shift mode and periodic coherent structure modes. Similarly, in Fig. 4.13, except for the mean mode all the other eigenfunctions indicate periodic coherent structures. Finally, the eigenfunctions from both AEN and DKN have similar magnitude unlike DMD or EDMD where the eigenfunctions μ_0 and μ_1 had higher magnitude which translates to higher dominance in the flow dynamics and reconstruction.

	DMD	EDMD	DKN-4	AEN-4	DKN-6	AEN-6
$\mu(St = 0)$	$0.99 \pm 0.009i$	$1 \pm 0i$	$0.9934 \pm 0i$	$0.9931 \pm 0i$	$0.993 \pm 0i$	$0.995 \pm 0i$
λ_μ	$-0.0080 \pm 0.0072i$	0	-0.0053	-0.0055	-0.0056	-0.0040
$\mu(St = 0.1655)$	$0.979 \pm 0.2063i$	$0.9784 \pm 0.2066i$	$0.9822 \pm 0.1976i$	$0.982 \pm 0.1976i$	$0.982 \pm 0.1982i$	$0.9796 \pm 0.2005i$
λ_μ	$0.0004 \pm 0.1653i$	$0 \pm 0.1656i$	$0.0015 \pm 0.1580i$	$0.0013 \pm 0.1580i$	$0.0014 \pm 0.1585i$	$0.0001 \pm 0.1607i$
$\mu(St = 0.331)$	$0.9167 \pm 0.4043i$	$0.9149 \pm 0.4039i$	$0.2828 \pm 0.2394i$	$0.8445 \pm 0.4453i$	$--$	$0.899 \pm 0.289i$
λ_μ	$0.0015 \pm 0.3305i$	$0.0001 \pm 0.3308i$	$-0.7917 \pm 0.5604i$	$-0.0369 \pm 0.3861i$	$--$	$0.0456 \pm 0.2475i$
$\mu(St = 04965)$	$0.8145 \pm 0.5834i$	$0.8108 \pm 0.5856i$	$--$	$--$	$--$	$0.8247 \pm 0.5495i$
λ_μ	$0.0015 \pm 0.4946i$	$0.0001 \pm 0.4978i$	$--$	$--$	$--$	$-0.0072 \pm 0.4677i$

Table 4.2: Koopman eigenvalues(μ), growth rate ($Re(\lambda)$) and discrete frequencies ($Im(\lambda)$) obtained from Transient region - I.

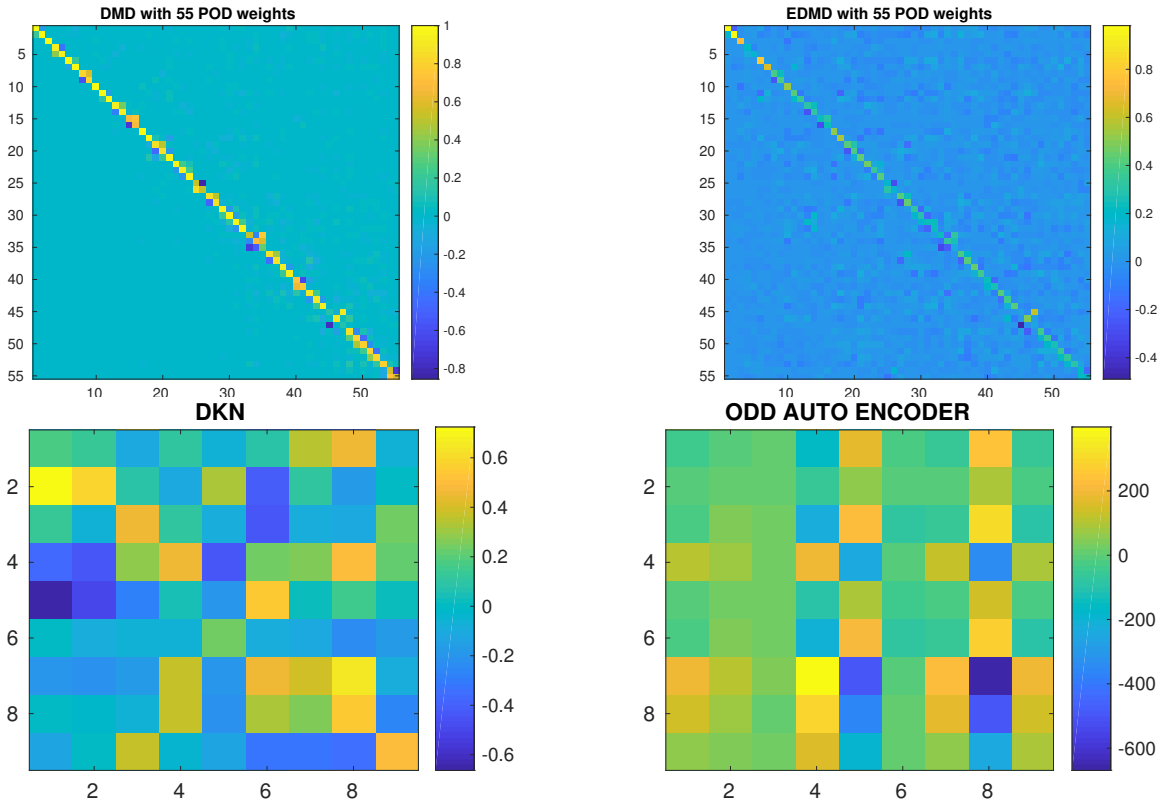


Figure 4.8: Comparison of the Koopman operators from DMD, EDMD, DKN and AEN methods.

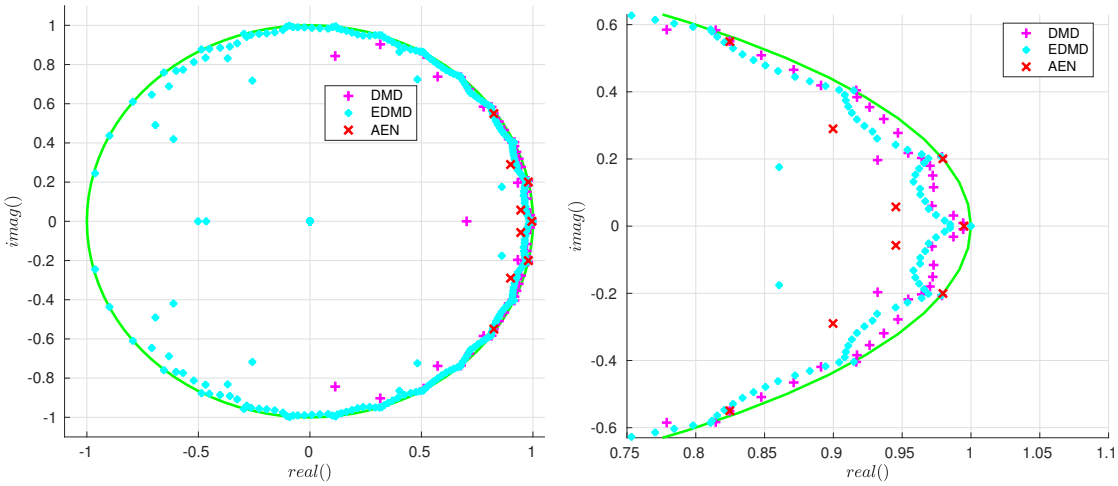


Figure 4.9: Ritz plot of eigenvalues(μ) spectrum obtained from 7 - HL AEN

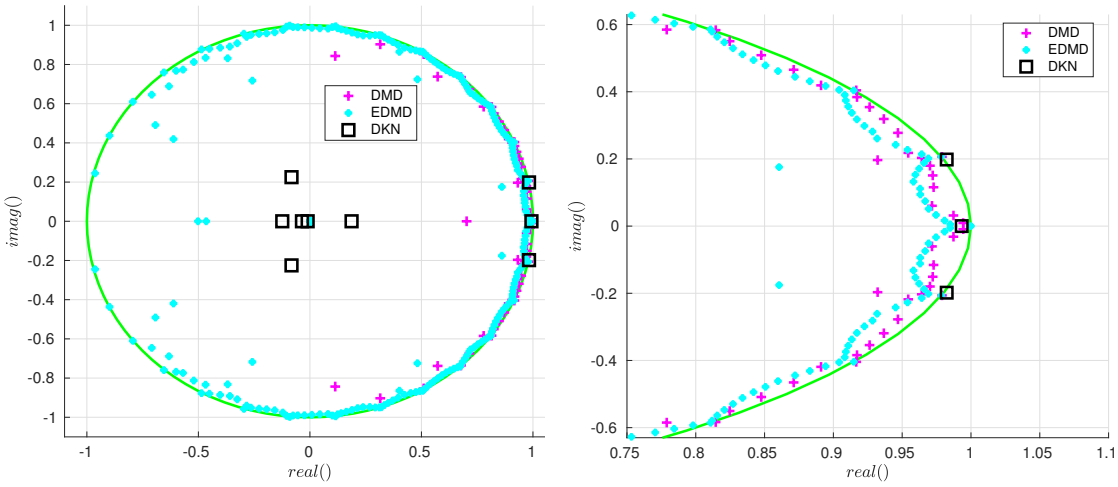


Figure 4.10: Ritz plot of eigenvalues(μ) spectrum obtained from 6 - HL DKN

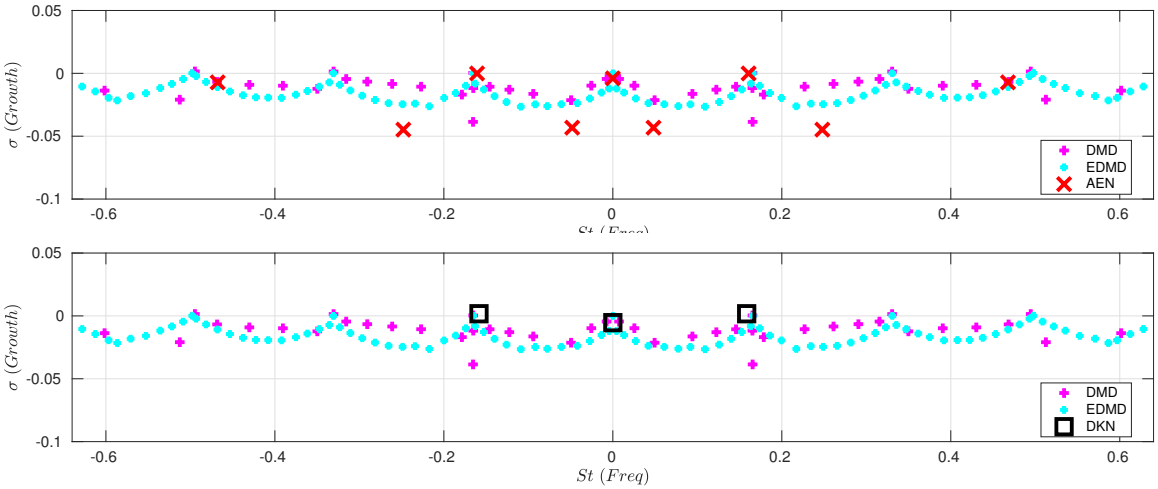


Figure 4.11: Comparison of growth rate and frequencies of (1^{st} row) 7 - HL Auto-Encoder (AEN) and (2^{nd} row): 6 - HL Deep Koopman Network (DKN)

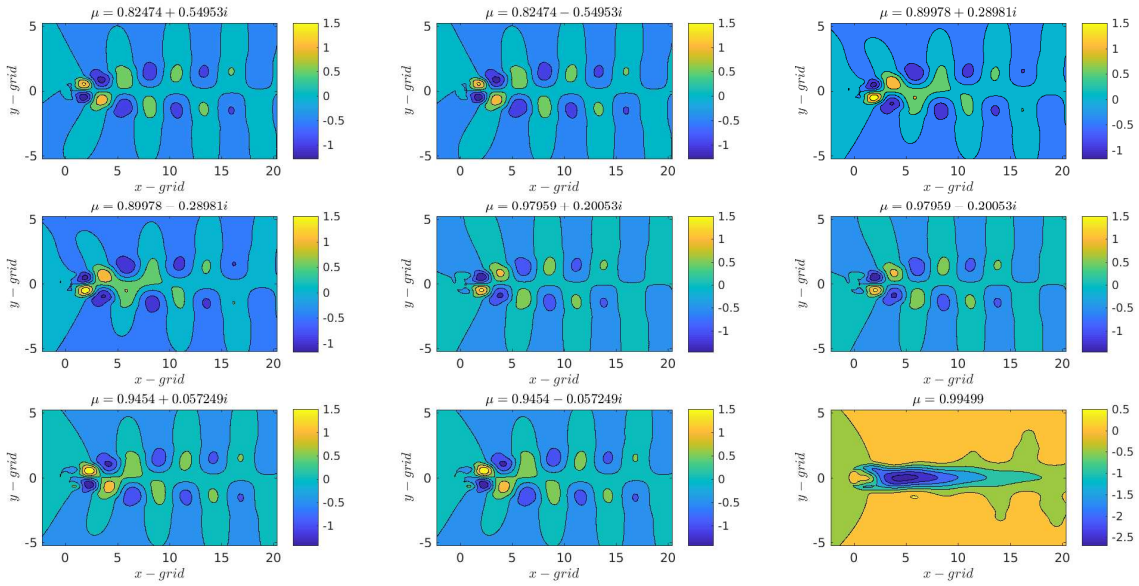


Figure 4.12: Koopman Eigen functions obtained using 7 - HL AutoEncoder Network (AEN), with 3 POD weights

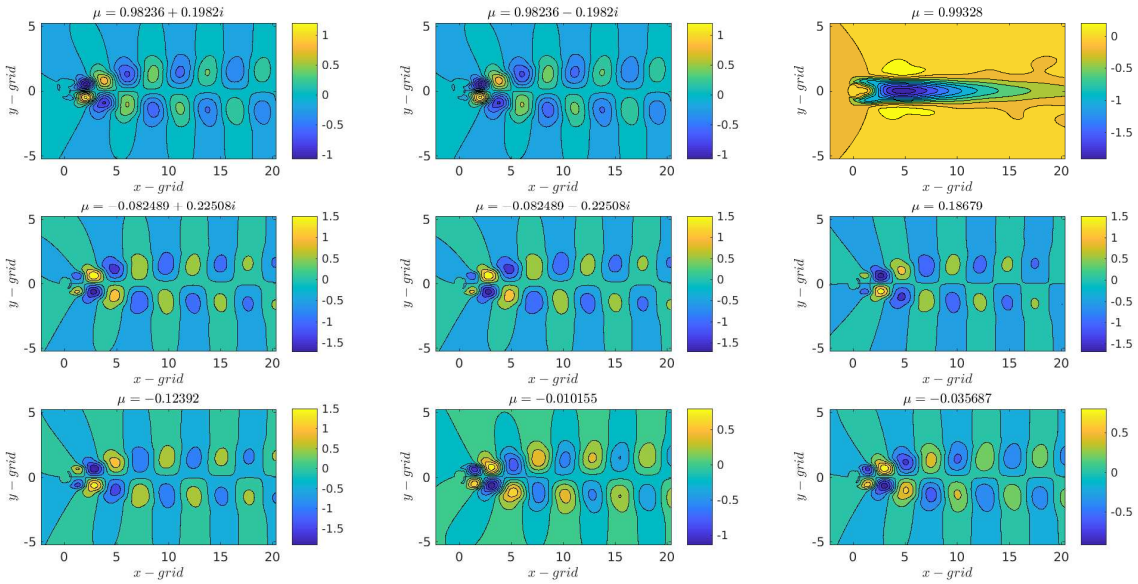


Figure 4.13: Koopman Eigen functions obtained using 6 - HL Deep Koopman Network (DKN), with 3 POD weights

4.3.1 Effect of Input Features vs Hidden Features Increase on Spectral Information

In this section, we have studied the importance of adding data vs spanning the learning space to accurately capture the dynamics and spectral information. Here, we have used 10 POD weights for learning the transient region -I, while using feature growth factor $N_f = 1$. In Fig. 4.14, we can see that the outer prediction obtained using 10 features with same number of hidden features (neurons), the predictions are qualitatively similar. The eigen spectrum obtained Koopman operators of AEN and DKN are plotted in Fig. 4.15 and the growth rates and frequency in Fig. 4.16. We can see that, contrary to DKN and AEN results with 3 POD weights as input features, the spectrum with 10 POD weights captured the first 2 dominant coherent structures and the mean mode. While DKN and AEN with three input features was able to capture only the first harmonic and the mean mode. Further, we can see that the Koopman modes obtained from the 10 POD weight Koopman operator are very

distinct than what was predicted from both DMD and EDMD. Further research is need to provide an full analysis that provides important conclusions.

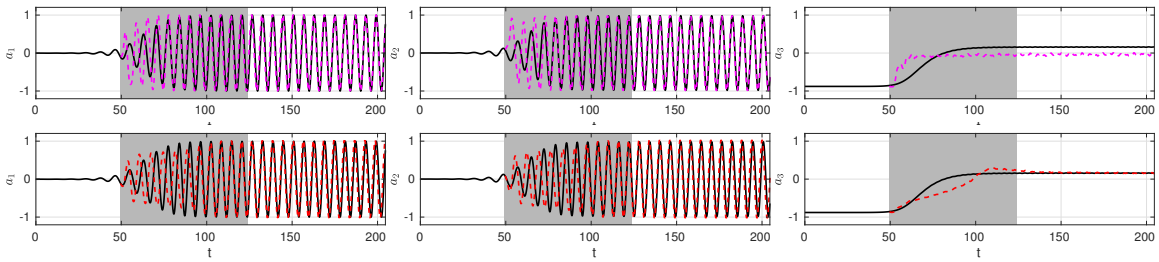


Figure 4.14: Comparison of predictions (1^{st} row)5 - HL Auto-Encoder (AEN) and (2^{nd} row): 4 - HL Deep Koopman Network (DKN)

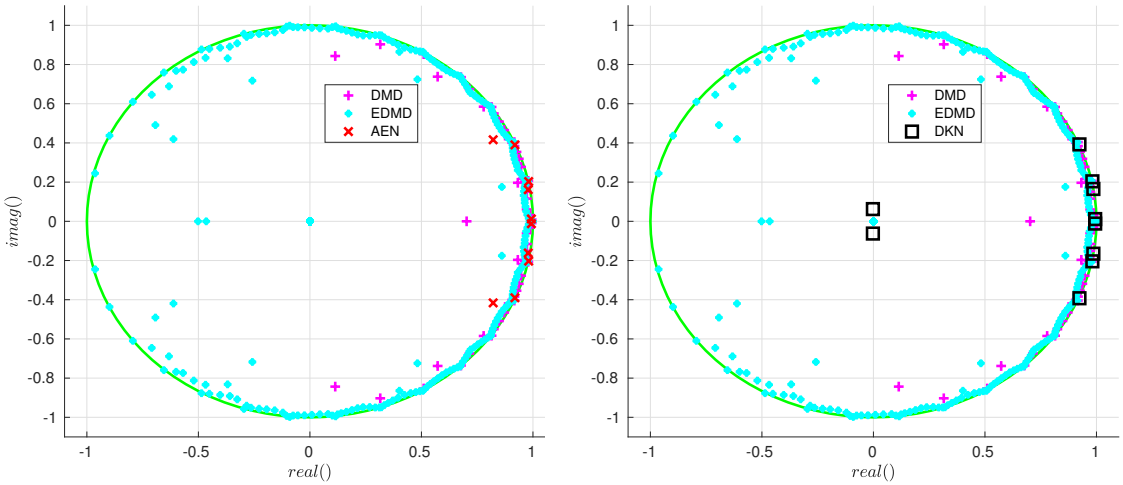


Figure 4.15: Ritz plot of eigenvalues(μ) spectrum obtained from 4 - HL DKN

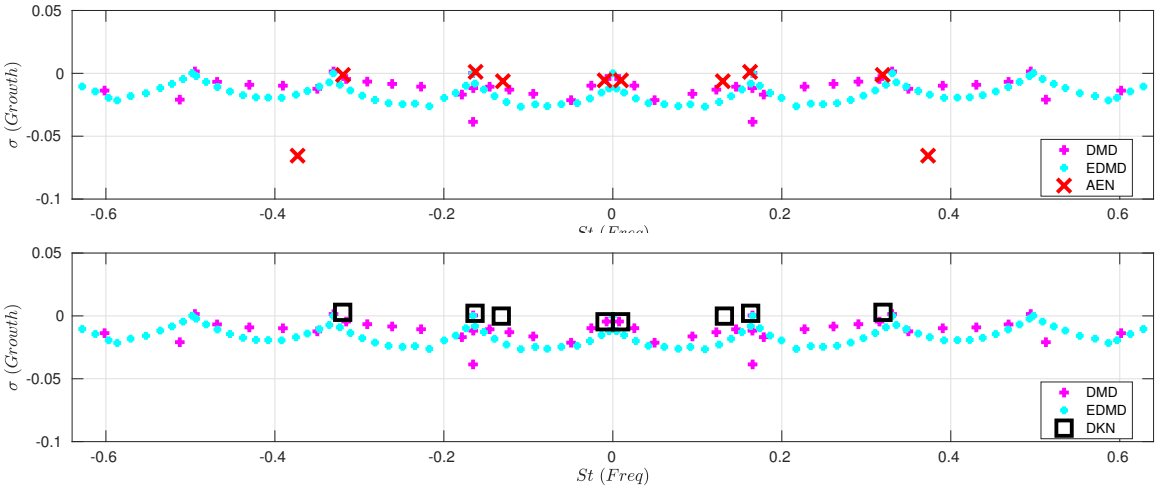


Figure 4.16: Comparison of growth rate and frequencies of (1^{st} row)5 - HL Auto-Encoder (AEN) and (2^{nd} row): 4 - HL Deep Koopman Network (DKN)

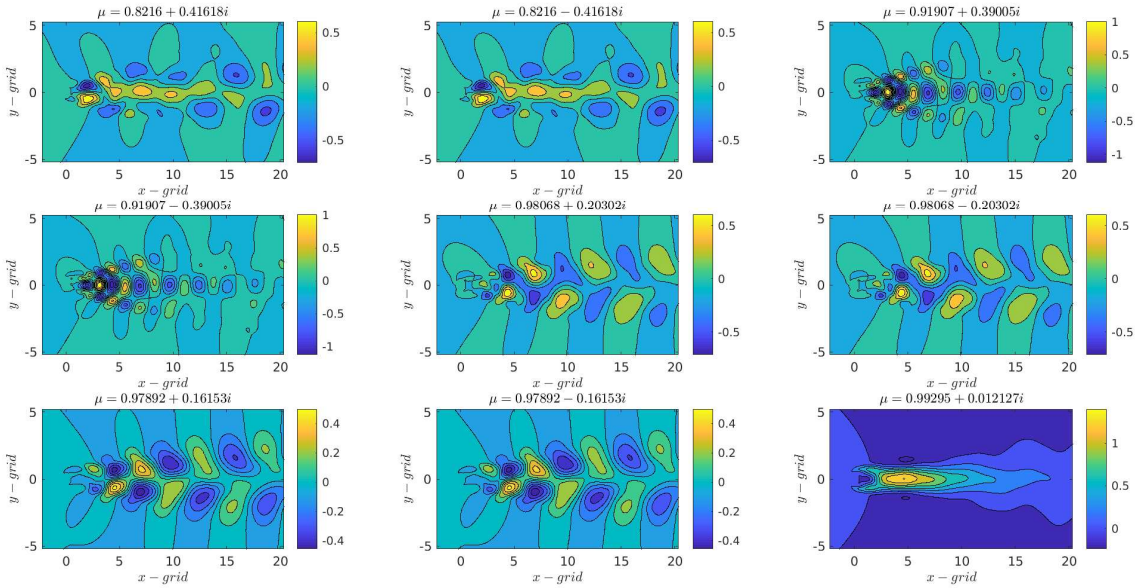


Figure 4.17: Koopman Eigen functions obtained using 5 - HL Autoencoder Network (AEN), with 10 POD weights

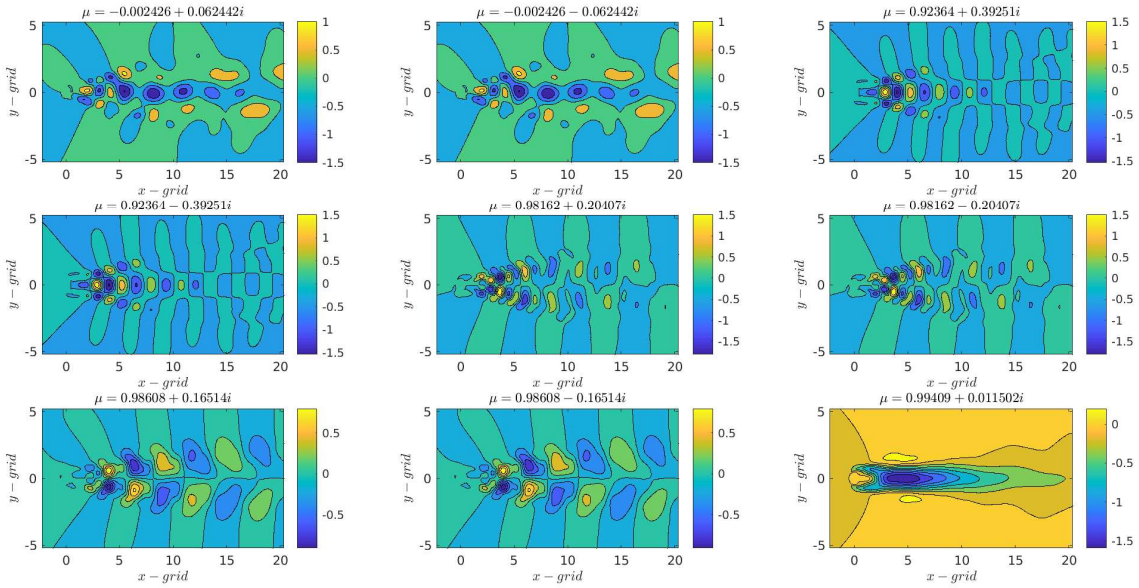


Figure 4.18: Koopman Eigen functions obtained using 4 - HL Deep Koopman Network (AEN), with 10 POD weights

4.3.2 Significance of Accurate Predictions on Spectral Information

In this section, we have presented importance of correlation between accurate predictions and spectral information. In the Fig. 4.19, we can see that the predictions obtained fail to capture the transient dynamics accurately, but they are able to predict the limit cycle dynamics accurately. Using the Koopman operator of these predictions, we can answer if the spectral information captured by this operator can distinguish between the eigenvalues that help in predicting the transient dynamic and limit cycle. Here again we can see from the Fig. 4.20, the growth rates and the frequencies captured are very similar to the ones that were obtained from good predictions results operator. More investigation is needed to concretely assess the importance of the eigenvalue spectrum on the predictions.

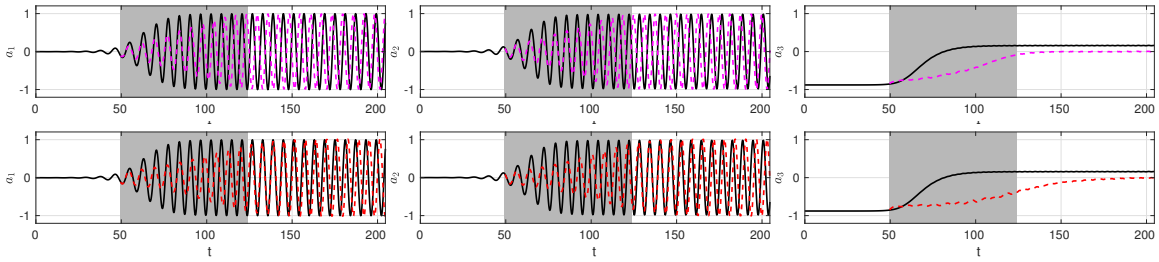


Figure 4.19: Comparison of predictions (1^{st} row) 7 - HL Auto-Encoder (AEN) and (2^{nd} row): 6 - HL Deep Koopman Network (DKN)

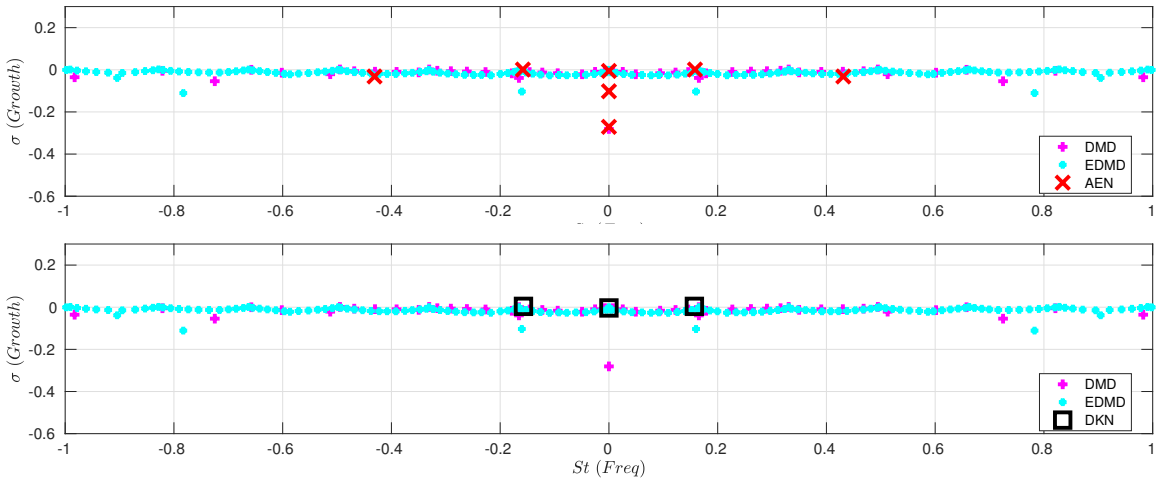


Figure 4.20: Comparison of growth rate and frequencies of (1^{st} row) 7 - HL Auto-Encoder (AEN) and (2^{nd} row): 6 - HL Deep Koopman Network (DKN)

CHAPTER 5

Genetic Algorithm based Global Optimal Convolution

5.1 Introduction

The learning process in artificial neural networks is carried out by backpropagation algorithm Rumelhart et al. (1988) which is widely used and helped in popularizing neural networks. Artificial neural networks have been widely used and deep variant have demonstrated excellent accuracy in image classification and regression. More recently, artificial neural networks have been used in fluid dynamics community as model order reduction methods with main emphasis on extracting underlying physics Raissi et al. (2017). Furthermore, artificial neural networks are used for regression and prediction. Neural networks extract the underlying physics or patterns with the help of activation or transformation functions, which are vital to this process. While back propagation needs gradient information to optimize the weights connecting the neurons. Generally, conventional activation functions used in the machine learning community are not enough to model fluid flows or help in extracting the underlying physics. Methods like proper orthogonal decomposition Schmid (2010), wavelet and Fourier transformation functions have been found optimal to model fluid flows. But these transformation functions don't have derivative information based on which back propagation works. So, there is a need for an optimization technique that potentially replaces back propagation and aid in the usage of physics based transformations which enable efficient modeling. To do this, we have used genetic algorithm which is a global optimizer and do not require gradient information to optimize the system. There have been attempts to use genetic algorithm based methods to optimize neu-

ral networks with limited successYen & Lu (2003), Montana & Davis (1989), while review articles by Yao Yao (1999) and Ding Ding et al. (2013) have provided various possibilities using genetic algorithm in neural networks. This ranges from designing the neural network architecture by optimizing the hyper parameters involved while using back propagation to optimize weights to both designing and optimizing the hyper parameters, weights and number of layers. In this article, we present out initial efforts into using genetic algorithm to train weights using non–dominated sorted genetic algorithm Deb et al. (2002) with conventional transformation functions.

The objectives of this study are as follows

1. Feasibility of genetic algorithm as a substitute for back propagation.
2. Genetic algorithm for designing and optimizing the network simultaneously
3. Develop strategies to help in improve the fine tuning ability of GA to find an optimal solution using multi-objective optimization using aposteriori error.

5.2 Methodology

In Figs. 2.5 and 5.1 of the neural network used as part of this study. The four layer design is motivated from the previous efforts where in a four layer or higher have found to model the flow more accurately compared to a shallow(one hidden layer) network. Further, this network acts as a precursor to deep neural networks and can help making important observations needed to make genetic algorithm work with such an architecture. Genetic algorithm is widely used as a multi objective optimizer, where in competing multi objective functions are optimized simultaneously. We have posed the training process as a multi objective problem with objective functions to minimize are cost functions shown in Eqn. 5.13 is computed using forward propagation. Training error or apriori error in Eqn. 5.13 which is generally used in backpropagation algorithm to learn has been chosen as objective-1 and additional posteriori error is used

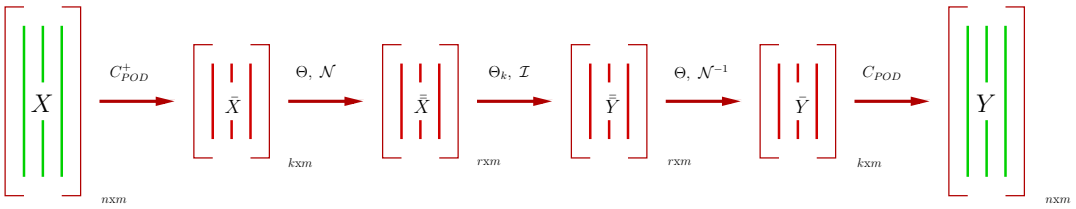


Figure 5.1: A schematic representation of the Modified Feed Forward Neural Network with $\mathcal{N}^{-1}$ used for symmetric architecture.

as objective -2 to understand the effect of learning on regression and performance.

5.2.1 Modified MGOC-1

In this section we discuss the modification with respect to the nonlinear mapping used in a MGOC framework . This framework also follows the same procedure as the MGOC described in section 2.2.3. The following equations clearly describe the procedure for a four layer layer network (without POD convolution layers):

$$Z_1 = \Theta_1 \bar{X}, \tag{5.1}$$

$$\bar{\bar{X}} = \mathcal{N}(Z_1), \tag{5.2}$$

$$Z_2 = \Theta_2 \bar{\bar{X}}, \tag{5.3}$$

$$\bar{\bar{X}} = \mathcal{N}^{-1}(Z_2), \tag{5.4}$$

$$\bar{Y}_p = \Theta_3 \bar{\bar{X}}. \tag{5.5}$$

Here we can see that the inverse of the nonlinear map $\mathcal{N}$ helps in mapping the evolved features from the Koopman space to the original manifold of the POD coefficients. This also helps in symmetry of the framework.

5.2.2 Inverse Mapping Transfer Function: Tansigmoid

Inverse mapping for a typical transfer function tansigmoid is as follows:

$$\mathcal{N} = \frac{e^{-2z} - e^{-2z}}{e^{-2z} + e^{-2z}} \tag{5.6}$$

$$\mathcal{N}^{-1} = \ log\left(\left[\frac{1+z}{1-z}\right]^{\frac{1}{2}}\right) \tag{5.7}$$

Intermediate variable derivative - $\frac{\partial z}{\partial z_i}$ Tan Sigmoid

$$\frac{\partial dz_3}{\partial z_{i3}} = \frac{\partial}{\partial z_i}\left[\frac{1-z_i}{1+z_i}\right]^{-1/2} \tag{5.8}$$

$$= -\frac{1}{2}\left[\frac{1-z_i}{1+z_i}\right]^{-1/2-1} \cdot \left[\frac{(1+z_i)(-1) - (1-z_i)(1)}{(1+z_i)^2}\right] \tag{5.9}$$

$$= -\frac{1}{2}\left[\frac{1-z_i}{1+z_i}\right]^{-3/2} \cdot \left[\frac{-1-z_i-1+z_i}{(1+z_i)^2}\right] \tag{5.10}$$

$$= \left[\frac{1-z_i}{1+z_i}\right]^{-3/2} \cdot \left[\frac{1}{(1+z_i)^2)}\right] \tag{5.11}$$

$$= \frac{1}{(1-z_i)^{3/2} \cdot (1+z_i)^{1/2}} \tag{5.12}$$

$$\mathcal{J}_l = \underbrace{\frac{1}{2m * N}\sum_{i=1}^{m}\sum_{j=1}^{n}(\bar{Y}_p(j,i) - y(j,i))^2}_{\text{Cost function}} \tag{5.13}$$

$$R(\Theta) = \underbrace{\left(\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{p=1}^{P_l-1}\sum_{q=1}^{Q_l}\left(\Theta_{p,q}^{(l)}\right)^2\right)}_{\text{Regularisation term}} \tag{5.14}$$

$$\mathcal{J}_g = \underbrace{\frac{1}{2m * N}\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{\bar{Y}}(j,i) - y(j,i))^2}_{\text{Cost function}} \tag{5.15}$$

5.2.3 Non-dominated Sorting Genetic Algorithm

Evolutionary algorithm used for this study, NSGA–II (Deb et al. 2002) is a popular non domination based genetic algorithm. This method is widely used in multiobjective optimization. The algorithm of the method is detailed in algorithm 4, while the algorithms of important operations done as part of this method like Fast non–dominated sorting and computing crowding distance are detailed in algorithms 5 and 6. The method starts with initialization of the population of individuals with specific chromosome length. Then the populations is sorted to into fronts based on non-domination sorting. These fronts are labeled or fitness ranked from 1 to p based on level of domination/fronts these individuals are in compared to the other individuals in the population. In addition to fitness value these crowding distance of each individual is computed to maintain diversity while creating new generation. Based on this fitness value and crowding distance parents are selected from the population to generate offspring/children using crossover and mutation operators. The current population and current children are sorted again and the best N individuals are progressed into the next generations. This cycle repeats until it satisfies the stopping criteria, which generally is the number of generations.

5.3 Test Bed: Flow over a Cylinder

In this study, direct numerical simulation obtained by solving Navier-Stokes equations over a cylinder are used. In Fig. 5.3 a contour plot of velocity field of the fluid is plotted. The data comprises of velocity field on $\approx 23,000$ data points over 2000 time steps. But, this system can be reduced using proper orthogonal system which provides a low dimensional manifold where the dynamics of such a flow can be modeled and evolved. In Fig. 5.4 the first three modes(low dimensional manifold) and their corresponding weights(dimensions) which govern the dynamics this flow are plotted. By learning these weights and evolving them would enable accurate prediction of the

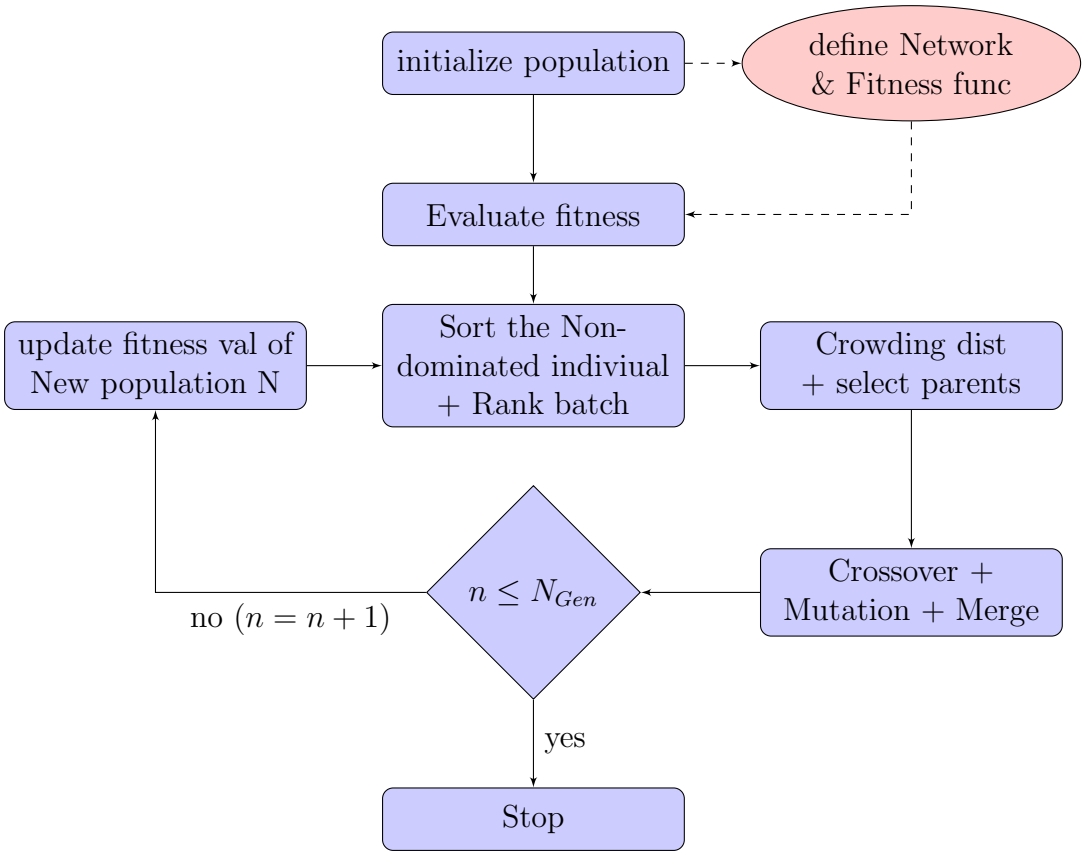


Figure 5.2: Schematic of the NSGA-II algorithm process

Algorithm 4 NSGA 2

procedure INITIALIZE POPULATION(P)
  $R_t = P_t \cup Q_t$
  $\mathcal{F} =$ Fast–non–dominated–sort(R_t)
  $P_{t+1} = \emptyset$ and $i = 1$
  while $|P_{t+1}| + |\mathcal{F}_i| \leq N$ **do**
    Crowding–distance–assignment($\mathcal{F}_i$)
    $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
    $i = i + 1$
    Sort($\mathcal{F}_i, \prec_n$)
    $P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - -|P_{t+1}|)]$
    $Q_{t+1} =$ make–new–pop(P_{t+1})
    $t = t + 1$ ▷ increment the generation counter
  end while
end procedure

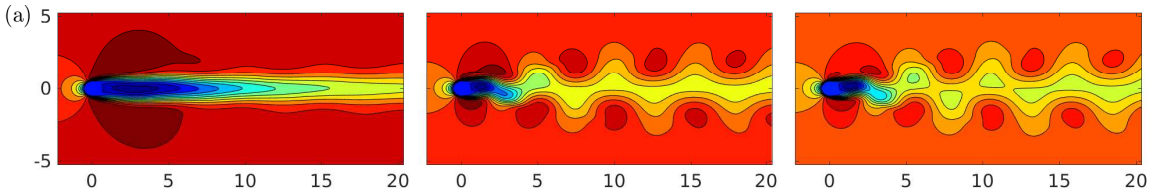


Figure 5.3: Velocity field snapshots of $Re100$ flow field.

fluid flow.

5.4 Results

5.4.1 Modeling Fluid Flow: Multi-Objective Problem

In this study, fluid flow is modeled as a multi objective optimization problem. While the minimization of the local (apriori) error cost function is straight forward, but including the posteriori (global) error cost function as other objectives is to add constraints which aides in improving the predictions. While regularization is also applied and will help in terms of penalizing the weights, the global error is used

Algorithm 5 Fast Non–Dominated Sort

procedure FAST NON–DOMINATED SORT(for each p)
  for $p \in P$ **do**
    $S_p = \emptyset$
    $n_p = 0$
    for $q \in P$ **do**
      if $p \prec q$ **then**
        $S_p = S_p \cup \{q\}$
      else if $(q \prec p)$ **then**
        $n_p = n_p + 1$
      end if
    end for
    if $n_p = 0$ **then**
      $p_{rank} = 1$
      $\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$
    end if
  end for
  $i = 1$ ▷ Initialize the front counter
  while $\mathcal{F}_i \neq \emptyset$ **do**
    $Q = \emptyset$
    for $p \in \mathcal{F}_i$ **do**
      for $q \in S_p$ **do**
        $n_q = n_q - 1$
        if $n_q = 0$ **then** ▷ q ∈ to the next front
          $q_{rank} = i + 1$
          $\mathcal{Q} = \mathcal{Q} \cup \{q\}$
        end if
      end for
    end for
    $i = i + 1$
    $\mathcal{F}_i = Q$
  end while
end procedure

Algorithm 6 Crowding Distance

procedure CROWDING DISTANCE–ASSIGNMENT($\mathcal{I}$)

$l = [\mathcal{I}]$

for $i \in \mathcal{I}$ **do**

Set $\mathcal{I}[i] = 0$

end for

for for each objective m **do**

$\mathcal{I} = sort(\mathcal{I}, m)$

$\mathcal{I}[1]_{dist} = \mathcal{I}[l]_{dist} = \infty$

for $i = 2$ *to* $(l-1)$ **do**

$\mathcal{I}[i]_{dist} = \mathcal{I}[i]_{dist} + (\mathcal{I}[i+1] \cdot m - \mathcal{I}[i-1] \cdot m)/(f_m^{max} - f_m^{min})$

end for

end for

end procedure

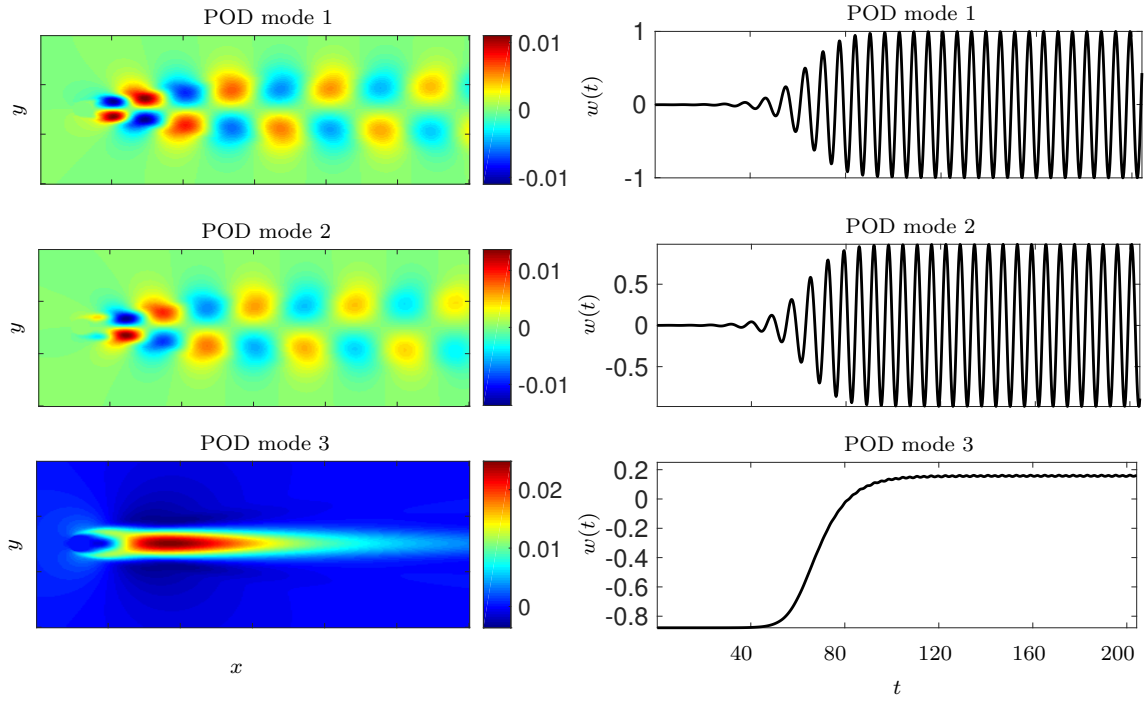


Figure 5.4: POD modes and their corresponding time series weights are plotted

to accurately provide the actual time series error associated with individuals in the population. The choice of global error cost function was motivated from the fact that, with the cost function used here, there is a high probability that the cost of a bad solution individual and a good solution would be same, many such cases have been found through experience. In Fig. 5.5 shows the pareto front of all possible solutions found from the MGOC–GA algorithm, we can see that the prediction in Fig. 5.6 from this pareto front has a good solution even when compared MGOC with backpropagation, whose cost is less. Finally, two objectives are simultaneously optimized and we see that the prediction improves even further which is evident from predictions. Here, the lack of fine tuning final solution ability of NSGA–2 is the caused the satisfactory results when compared to actual solution. In Fig. 5.6, we have plotted randomly chosen pareto front solutions to perform predictions on the transient dynamics. We can see that, the best predictions correspond to the points in pareto front which have both objective functions minimized. Further, we have plotted for comparison the predictions results obtained from DMD, EDMD and FFNN in Fig. 5.7 with same learning space and learning parameters. We can see that GA performed far better then DMD. EDMD and FFNN.

5.5 Summary

In this work, we proposed a derivative free training algorithm for training weights in neural networks. Here, the non dominated sorting genetic algorithm was successfully used as a training algorithm to find optimal set of weights. Here, the optimization problem was cast as a multi-objective problem to improve the accuracy of the predictions and penalize the false positive individuals from the population. It has to be noted here that there is a need an algorithm which fine tunes the weights and filters out a optimal solution based on the competing objective functions. Furthermore, this method can be used to design the network and optimize the weights simultaneously.

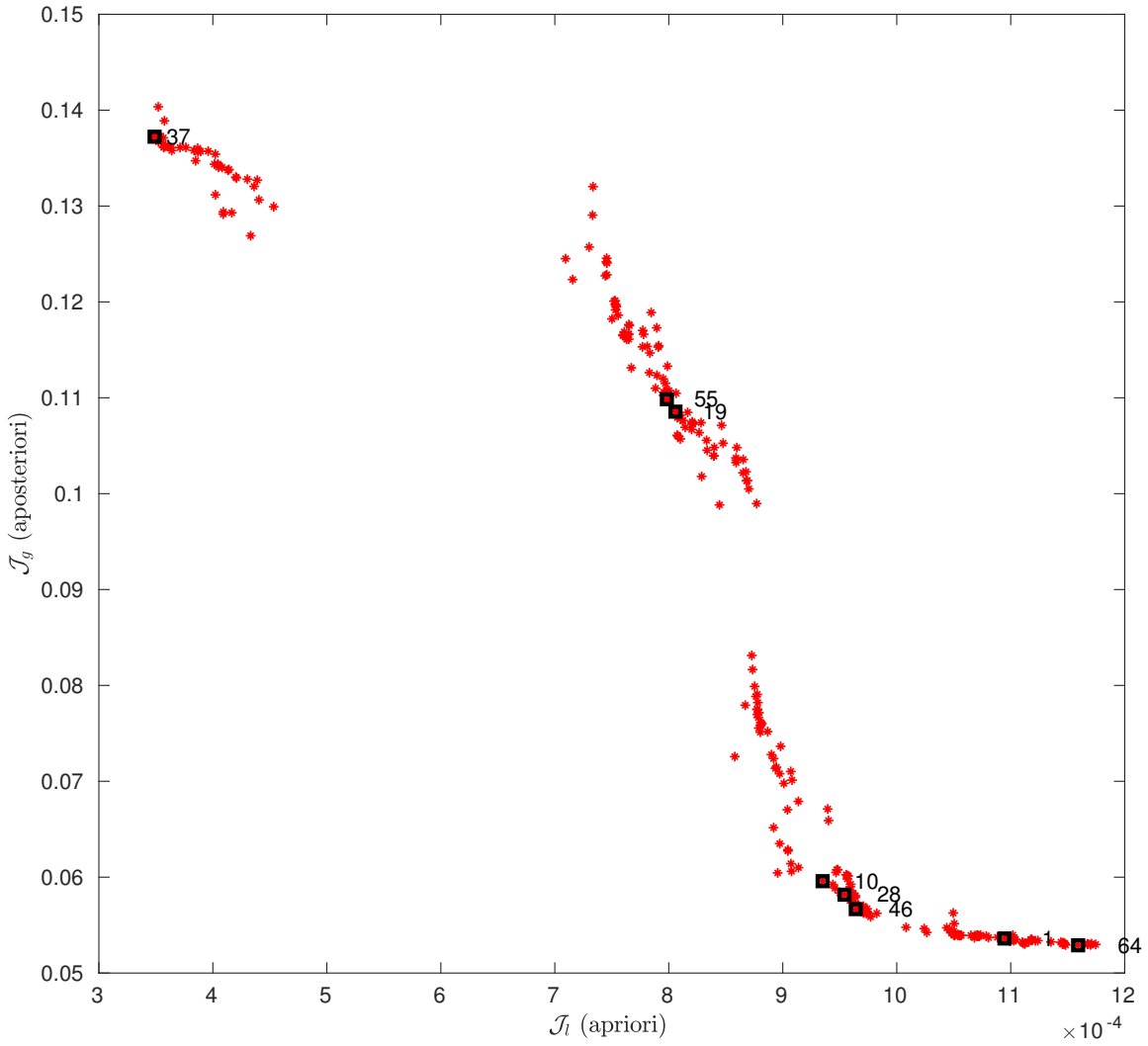


Figure 5.5: Pareto front of apriori error vs. aposteriori error

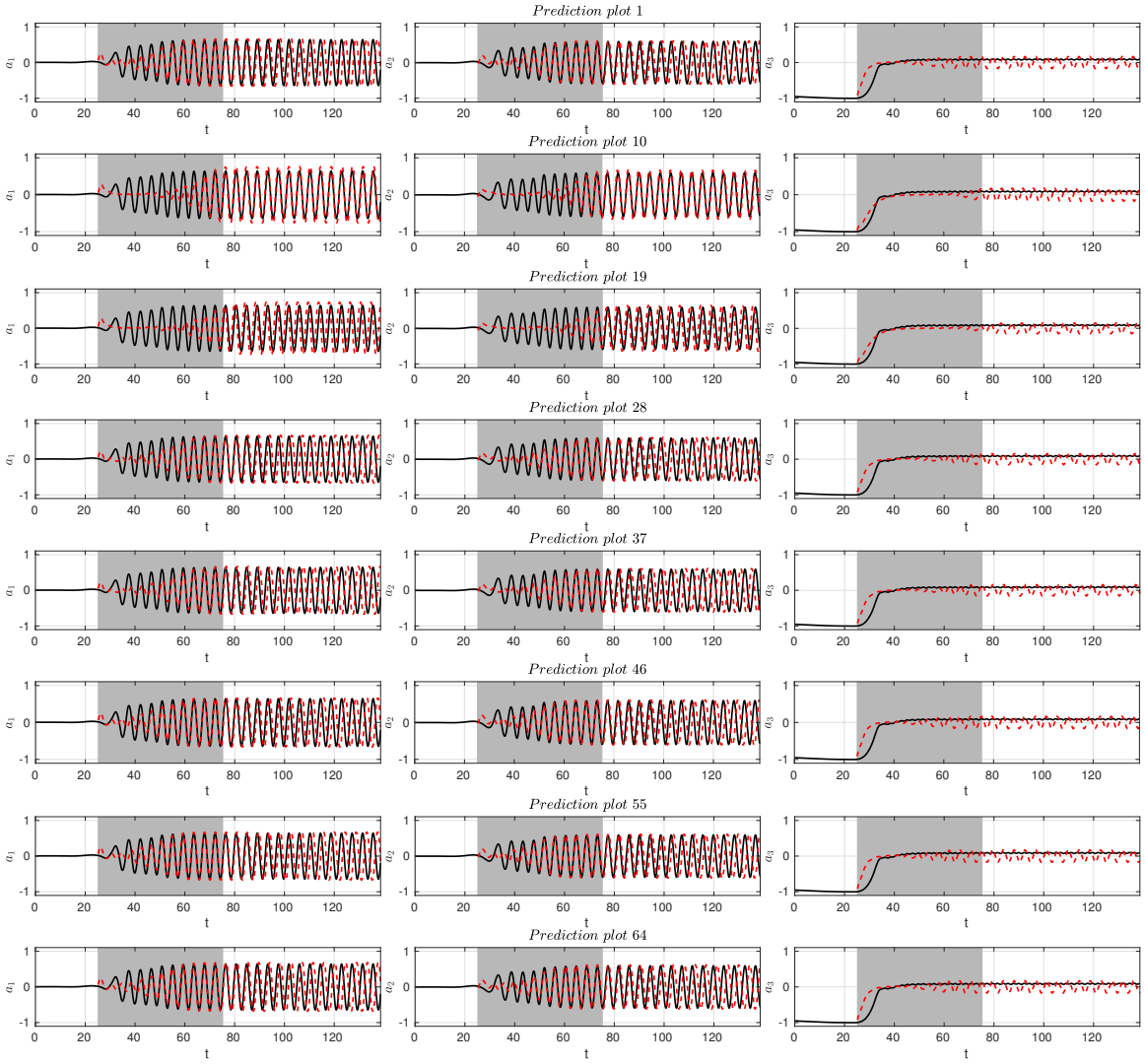


Figure 5.6: Prediction results from the Rank 1 pareto front from NSGA-2

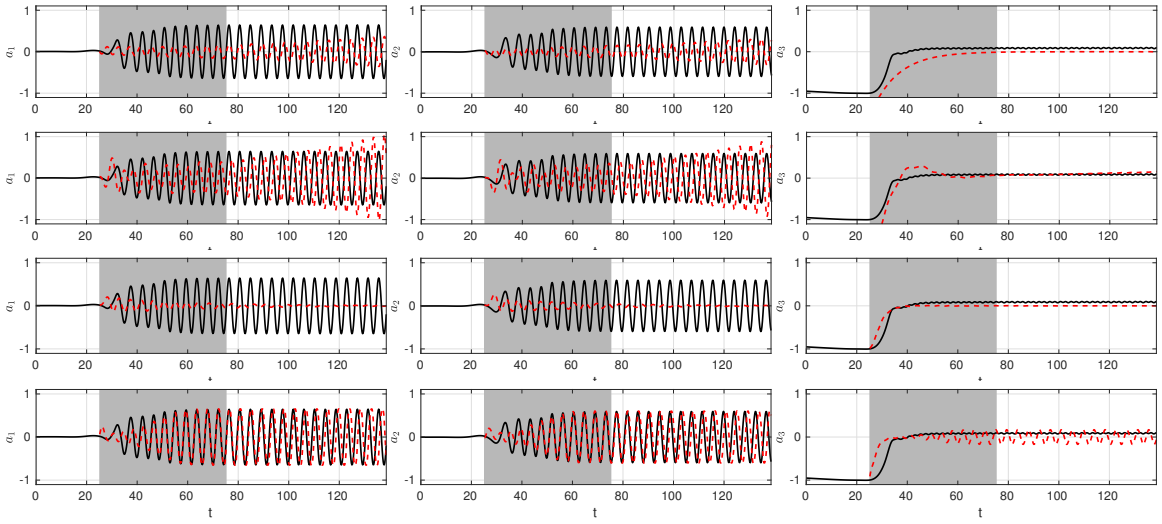


Figure 5.7: Comparison of prediction results between (a) DMD, (b) EDMD, (c) MFFNN-BP and (d) MFFNN-GA (best of the Rank 1 pareto front from NSGA-2).

In addition, this method enables the usage of different sets of activation functions which need not have derivation information. In future, we would like to use this method to first learn low dimensional manifolds and later use them in modeling the fluid flows.

CHAPTER 6

Conclusions and Recommendations

6.1 Summary

In this thesis we explore the role of local (MLOC) versus global optimization (MGOC) of the multilayer convolution maps through the lens of learning parameter dimensionality, symmetric networks, spectral analysis and nonlinear transfer functions on their ability to reconstruct, predict and allow for analysis for the transient, nonlinear dynamics of canonical fluid flows. Fluid flows represent multiscale PDE dynamical systems that often require low-dimensional data-driven representations and evolutionary models for a multitude of applications. The locally optimal multilayer frameworks allow for both backward and forward operations and can support symmetric architectures including Koopman approximation methods such as DMD and EDMD, which also allow for analyzing data. Such approaches allow for spectral analysis of the underlying system in addition to building data-driven models. On the other hand, globally optimal (MGOC) frameworks can support only forward maps due to the choice of gradient-based optimization algorithms being commonly used as the tool of choice for nonlinear parametric regression in machine learning. Consequently, MGOC architectures like FFNN cannot learn the Koopman operator using current methods. Recent work by the authors on leveraging feedback networks Puligilla & Jayaraman (2018*a*) with feedforward networks has allowed us to bypass this limitation to perform Koopman spectral analysis along with building an effective predictive model.

The success of both the MLOC and MGOC architectures are tied to the nonlinearity in the mapping and the size of the learning parameter space in the multilayer

architecture design. We observe that for prediction of limit-cycle dynamics from limit-cycle data where all the different models show reasonable success, FFNN-like MGOC models control the growth of long-time prediction errors better than MLOC models. While prediction errors decrease with increase in dimension of the learning parameters and the appropriateness of the nonlinear transfer function, their combination through a MGOC architecture turned out to be the most effective. As an example of appropriateness of the nonlinear transfer function, we show that tansigmoid functions operate well with MGOC architectures as against MLOC architectures which perform better using polynomial nonlinear features.

To assess the ability of these model architectures to generalize to diverse training data regimes, we considered two different case studies with different training regimes that differ in their limit-cycle content. In order to mimic the availability of only limited resolution data as is commonly the case, we chose to train these models using their low-dimensional representation with only three POD features. Within the context of these restrictions, we observed that for comparable number of learning parameters, the FFNN-like MGOC architectures outperform the corresponding locally optimal MLOC model frameworks by a significant margin in terms of accuracy and robustness. To illustrate this, we show that the 6-MGOC-TS1 architecture with 9 learning parameters produce qualitatively accurate results as against the grossly inaccurate results using DMD/EDMD (4-MLOC and 6-MLOC) models. With increase in $\mathcal{LP}$ dimension, both class of methods converge to the accurate predictions with tangible improvements and slower convergence in the MLOC as compared to the MGOC framework. For example, a 7^{th} polynomial nonlinearity in the MLOC architecture with $O(1.5 \times 10^4)$ parameters produced accurate predictions as against the 6-MGOC-TS3 models with a bias term (traditional FFNN) with just 135 parameters.

The downside of MGOC models is their computational cost and learning time as one needs to solve a nonlinear regression problem, often requiring iterative gradient

search based algorithms which can also impact convergence. A common issue with FFNN/MGOC architectures is the solution being stuck in a local minimum as against a true global minimum for which algorithmic advances continue to be explored. However, this is made up for by the relatively modest increase in $\mathcal{LP}$ dimension needed to improve the learning and prediction performance. The case with limited quantity of information about the limit-cycle dynamics in a different training data regime (TR-II regime) both class of methods find learning and prediction harder. In this case however, the MGOC architectures were able to generate accurate predictions with as little as 135 parameters especially with the inclusion of a bias term whereas the MLOC method with $O(1.5 \times 10^4)$ parameters failed to predict meaningful data. When the inherent system dimension increases at higher Reynolds numbers, i.e. $Re = 1000$, the above trends remain consistent, but the dynamics are harder to predict, especially when the dimension of the training data is not increased. In spite of using data with missing dynamically relevant physics, the FFNN/MGOC models produced stable and qualitatively accurate results.

We also explored data-driven modeling of data-sparse non-stationary buoyant mixing flow phenomena in the context of pure learning-based reconstruction and learning-based prediction. For such systems, the MLOC-based methods turned out to be good at reconstruction (consistent with Lu et al. (2018)), but struggle to learn the dynamics to predict a future state accurately. While both methods struggle with quantitatively accurate prediction of such data, the MGOC frameworks generate better qualitative accuracy in the prediction regime.

MLOC (DMD) and its extensions for a given problem with knowledge of physics learns the nonlinear dynamics excellently but as we change the dynamics, the method perform poorly. Multilayer locally optimal convolutions are limited by the local optimization and we have seen that for a given span of hyper-parameters, MGOC methods perform better in learning the nonlinear dynamics on both the training sets. Further,

the modifications to MGOC has proved to improve the learning in modified MGOC and representation in Deep Koopman Networks. This is evident from the predictions on two different regions of training in transition. Comparison of a two step autoencoder verses the integrated learning in CKNN have shed light on the effect of using locally optimal convolutions in learning. Learning with two step autoencoders renders into a learning exercise in identifying the optimal feature maps. While an integrated approach would utilize the learning capability of back-propagation to learn Koopman subspaces and operator that are coupled and optimized with respect to the dynamical system. Ideally, both the models in true Koopman subspaces should provide similar results. Further, we see that while different physics provides different conclusions and there is a dichotomy of interpretation of results. More research is needed to concretely provide meaningful conclusions. Further, by performing the spectral analysis, we can see that both DKN and AEN methods capture the dynamics accurately but the koopman eigenvalues and eigenvectors are distinct. Finally, we propose a framework with genetic algorithm based MGOC that can enable utilization of data based convolution maps rather than searching for optimal convolutions. There are issues that needs to be addressed in MGOC-GA, like integration of POD like convolutions and finding cost function that accurately represents the learning process, which is future research.

In summary, the strategy of extending the $\mathcal{LP}$ space, learning the model parameters concurrently using a global optimization, constraining symmetry of network and improved regularizations Pan & Duraisamy (2018) can help enhance the efficiency of the learning process, improve robustness and accuracy of the resulting predictions. However, as with machine learning in general, these outcomes are strongly tied to data sufficiency and quality. Consistently, we observe that the MGOC models such as FFNN outperform the suite of MLOC frameworks explored in this thesis, although they are slightly harder to train. We see MLOC models as two-layer shallow learning

architectures requiring large intermediate layer dimensions while the MGOC is its deep learning counterpart that while harder to train can approximate the nonlinear dynamics using a few parameters.

REFERENCES

Bagheri, S. (2013), 'Koopman-mode decomposition of the cylinder wake', *Journal of Fluid Mechanics* **726**, 596–623.

Bengio, Y. (2007), 'On the challenge of learning complex functions', *Progress in Brain Research* **165**, 521–534.

Bengio, Y., Goodfellow, I. J. & Courville, A. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

Bengio, Y., Simard, P. & Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE transactions on neural networks* **5**(2), 157–166.

Benner, P., Gugercin, S. & Willcox, K. (2015), 'A survey of projection-based model reduction methods for parametric dynamical systems', *SIAM review* **57**(4), 483–531.

Berkooz, G., Holmes, P. & Lumley, J. L. (1993), 'The proper orthogonal decomposition in the analysis of turbulent flows', *Annual review of fluid mechanics* **25**(1), 539–575.

Bishop, C., Bishop, C. M. et al. (1995), *Neural networks for pattern recognition*, Oxford university press.

Brunton, S. L., Brunton, B. W., Proctor, J. L. & Kutz, J. N. (2016), 'Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control', *PloS one* **11**(2), e0150171.

Brunton, S. L. & Noack, B. R. (2015), 'Closed-loop turbulence control: progress and challenges', *Applied Mechanics Reviews* **67**(5), 050801.

Brunton, S. L., Proctor, J. L. & Kutz, J. N. (2013), 'Compressive sampling and dynamic mode decomposition', *arXiv preprint arXiv:1312.5186* .

Cantwell, C. D., Moxey, D., Comerford, A., Bolis, A., Rocco, G., Mengaldo, G., De Grazia, D., Yakovlev, S., Lombard, J.-E., Ekelschot, D. et al. (2015), 'Nektar++: An open-source spectral/hp element framework', *Computer Physics Communications* **192**, 205–219.

Cao, Y., Zhu, J., Navon, I. M. & Luo, Z. (2007), 'A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition', *International Journal for Numerical Methods in Fluids* **53**(10), 1571–1583.

Christopher, M. B. (2016), *PATTERN RECOGNITION AND MACHINE LEARNING.*, Springer-Verlag New York.

Csató, L. & Opper, M. (2002), 'Sparse on-line gaussian processes', *Neural computation* **14**(3), 641–668.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002), 'A fast and elitist multiobjective genetic algorithm: Nsga-ii', *IEEE transactions on evolutionary computation* **6**(2), 182–197.

Deem, E. A., Cattafesta, L. N., Yao, H., Hemati, M., Zhang, H. & Rowley, C. W. (2018), Experimental implementation of modal approaches for autonomous reattachment of separated flows, *in* '2018 AIAA Aerospace Sciences Meeting', p. 1052.

Ding, S., Li, H., Su, C., Yu, J. & Jin, F. (2013), 'Evolutionary artificial neural networks: a review', *Artificial Intelligence Review* pp. 1–10.

Edstrand, A. M., Schmid, P. J., Taira, K. & Cattafesta, L. N. (2018), 'A parallel stability analysis of a trailing vortex wake', *Journal of Fluid Mechanics* **837**, 858–895.

Fang, F., Pain, C., Navon, I., Gorman, G., Piggott, M., Allison, P., Farrell, P. & Goddard, A. (2009), 'A pod reduced order unstructured mesh ocean modelling method for moderate reynolds number flows', *Ocean modelling* **28**(1-3), 127–136.

Golub, G. H. & Van Loan, C. F. (2012), *Matrix computations*, Vol. 3, JHU Press.

Gottlieb, S., Shu, C.-W. & Tadmor, E. (2001), 'Strong stability-preserving high-order time discretization methods', *SIAM review* **43**(1), 89–112.

Hinton, G. E. & Salakhutdinov, R. R. (2006), 'Reducing the dimensionality of data with neural networks', *science* **313**(5786), 504–507.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Holmes, P. (2012), *Turbulence, coherent structures, dynamical systems and symmetry*, Cambridge university press.

Hopfield, J. J. (1982), 'Neural networks and physical systems with emergent collective computational abilities', *Proceedings of the national academy of sciences* **79**(8), 2554–2558.

Hornik, K., Stinchcombe, M. & White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural networks* **2**(5), 359–366.

Jayaraman, B. & Puligilla, S. C. (2018), 'Deep koopman multilayer networks for data-driven modeling and spectral analysis of fluid flows', *Manuscript in preparation for Journal fo Fluid Mechanics* pp. 1–23.

Kim, J. & Bewley, T. R. (2007), 'A linear systems approach to flow control', *Annu. Rev. Fluid Mech.* **39**, 383–417.

Koopman, B. O. (1931), 'Hamiltonian systems and transformation in hilbert space', *Proceedings of the National Academy of Sciences* **17**(5), 315–318.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

Lele, S. K. (1992), 'Compact finite difference schemes with spectral-like resolution', *Journal of computational physics* **103**(1), 16–42.

Liu, J.-G., Wang, C. & Johnston, H. (2003), 'A fourth order scheme for incompressible boussinesq equations', *Journal of Scientific Computing* **18**(2), 253–285.

Lu, C. & Jayaraman, B. (2017), Data-driven modeling for nonlinear fluid flows, *in* '23rd AIAA Computational Fluid Dynamics Conference', number 3628, pp. 1–16.

Lu, C., Jayaraman, B., Whitman, J. & Chowdhary, G. (2018), 'Sparse convolution-based markov models for nonlinear fluid flows', *arXiv preprint arXiv:1803.08222* .

Lumley, J. L. (2007), *Stochastic tools in turbulence*, Courier Corporation.

Lusch, B., Kutz, J. N. & Brunton, S. L. (2017), 'Deep learning for universal linear embeddings of nonlinear dynamics', *arXiv preprint arXiv:1712.09707* .

Manohar, K., Brunton, B. W., Kutz, J. N. & Brunton, S. L. (2017), 'Data-driven sparse sensor placement', *arXiv preprint arXiv:1701.07569* .

Mezić, I. (2005), 'Spectral properties of dynamical systems, model reduction and decompositions', *Nonlinear Dynamics* **41**(1), 309–325.

Montana, D. J. & Davis, L. (1989), Training feedforward neural networks using genetic algorithms., *in* 'IJCAI', Vol. 89, pp. 762–767.

Noack, B. R., Afanasiev, K., MORZYŃSKI, M., Tadmor, G. & Thiele, F. (2003), 'A hierarchy of low-dimensional models for the transient and post-transient cylinder wake', *Journal of Fluid Mechanics* **497**, 335–363.

Otto, S. E. & Rowley, C. W. (2017), 'Linearly-recurrent autoencoder networks for learning dynamics', *arXiv preprint arXiv:1712.01378* .

Pan, S. & Duraisamy, K. (2018), 'Long-time predictive modeling of nonlinear dynamical systems using neural networks', *arXiv preprint arXiv:1805.12547* .

Puligilla, S. C. (2018), 'Nonlinear multilayer convolution for data-driven modeling of fluid flow dynamics', *Internal rep* pp. 1–23.

Puligilla, S. C. & Jayaraman, B. (2018*a*), Deep multilayer convolution frameworks for data-driven learning of fluid flow dynamics, *in* '24th AIAA Fluid Dynamics Conference, Aviation Forum', number 3628, pp. 1–22.

Puligilla, S. C. & Jayaraman, B. (2018*b*), 'Neural networks as globally optimal multilayer convolution architectures for learning fluid flows', *arXiv preprint arXiv:1806.08234* .

Puligilla, S. C. & Jayaraman, B. (2018*c*), 'Nonlinear data-driven estimation of transient fluid flows', *Manuscript in preparation* pp. 1–29.

Raissi, M., Perdikaris, P. & Karniadakis, G. E. (2017), 'Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations', *arXiv preprint arXiv:1711.10566* .

Roshko, A. (1954), 'On the development of turbulent wakes from vortex streets', *NACA rep* .

Rowley, C. W. & Dawson, S. T. (2017), 'Model reduction for flow analysis and control', *Annual Review of Fluid Mechanics* **49**, 387–417.

Rowley, C. W., Mezić, I., Bagheri, S., Schlatter, P. & Henningson, D. S. (2009), 'Spectral analysis of nonlinear flows', *Journal of Fluid mechanics* **641**, 115–127.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. et al. (1988), 'Learning representations by back-propagating errors', *Cognitive modeling* **5**(3), 1.

Schmid, P. J. (2010), 'Dynamic mode decomposition of numerical and experimental data', *Journal of Fluid Mechanics.* **656**, 5–28.

Scholkopf, B. & Smola, A. J. (2001), *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT press.

Soltani, R. & Jiang, H. (2016), 'Higher order recurrent neural networks', *arXiv preprint arXiv:1605.00064* .

Taira, K., Brunton, S. L., Dawson, S., Rowley, C. W., Colonius, T., McKeon, B. J., Schmidt, O. T., Gordeyev, S., Theofilis, V. & Ukeiley, L. S. (2017), 'Modal analysis of fluid flows: An overview', *AIAA* **55**(12), 4013–4041.

Towne, A., Schmidt, O. T. & Colonius, T. (2018), 'Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis', *Journal of Fluid Mechanics* **847**, 821–867.

Trefethen, L. N. & Bau III, D. (1997), *Numerical linear algebra*, Vol. 50, Siam.

Weinan, E. & Shu, C.-W. (1998), 'Small-scale structures in boussinesq convection', *Physics of Fluids* .

Williams, C. K. & Rasmussen, C. E. (1996), Gaussian processes for regression, *in* 'Advances in neural information processing systems', pp. 514–520.

Williams, M. O., Kevrekidis, I. G. & Rowley, C. W. (2015), 'A data–driven approximation of the koopman operator: Extending dynamic mode decomposition', *Journal of Nonlinear Science* **25**(6), 1307–1346.

Williams, M. O., Rowley, C. W. & Kevrekidis, I. G. (2014), 'A Kernel-Based Approach to Data-Driven Koopman Spectral Analysis', *ArXiv e-prints* .

Williamson, C. (1989), 'Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low reynolds numbers', *Journal of Fluid Mechanics* **206**, 579–627.

Wu, H., Mardt, A., Pasquali, L. & Noe, F. (2018), 'Deep generative markov state models', *arXiv preprint arXiv:1805.07601* .

Wu, H. & Noé, F. (2017), 'Variational approach for learning markov processes from time series data', *arXiv preprint arXiv:1707.04659* **17**.

Wu, X., Moin, P., Wallace, J. M., Skarda, J., Lozano-Durán, A. & Hickey, J.-P. (2017), 'Transitional–turbulent spots and turbulent–turbulent spots in boundary layers', *Proceedings of the National Academy of Sciences* p. 201704671.

Yao, X. (1999), 'Evolving artificial neural networks', *Proceedings of the IEEE* **87**(9), 1423–1447.

Yen, G. G. & Lu, H. (2003), 'Hierarchical rank density genetic algorithm for radial-basis function neural network design', *International Journal of Computational Intelligence and Applications* **3**(03), 213–232.

Yu, R., Zheng, S. & Liu, Y. (n.d.), Learning chaotic dynamics using tensor recurrent neural networks, *in* 'Proceedings of the ICML 17 Workshop on Deep Structured Prediction, Sydney, Australia, PMLR 70, 2017'.

CHAPTER 7

Appendix

7.1 Effect of Bias on Predictions

The results presented in the main sections of this article were based on FFNN-like MGOC architectures devoid of the bias term. It is well known from machine learning literature Hornik et al. (1989) that the presence of the bias term contributes significantly to the universal function approximation characteristic of FFNNs provided sufficient $\mathcal{LP}$s are used to capture the dynamics. The lack of a bias parameter impacts the predictions of the shift mode for the transient cylinder wake dynamics (section 2.3.5). In contrast, the modes with zero mean were predicted accurately. The bias term helps in quantitative translation (shift) of the learned dynamics into higher or lower values as the case may be. In this additional discussion we provide predictions obtained with a non zero bias term. In Figs. 7.1 and 7.3, we have plotted the predictions obtained from 6-MGOC-TS1, -TS3, TS9 with TR-I for $Re = 100$ and $Re = 1000$ respectively. And, in Figs. 7.2 and 7.4, the predictions obtained from 6-MGOC-TS3 and -TS9 with the TR-II data. In these cases, we see that the shift mode (third POD feature) is predicted accurately with the bias term. Alternatively, one can preprocess the input data such that their mean values are zero.

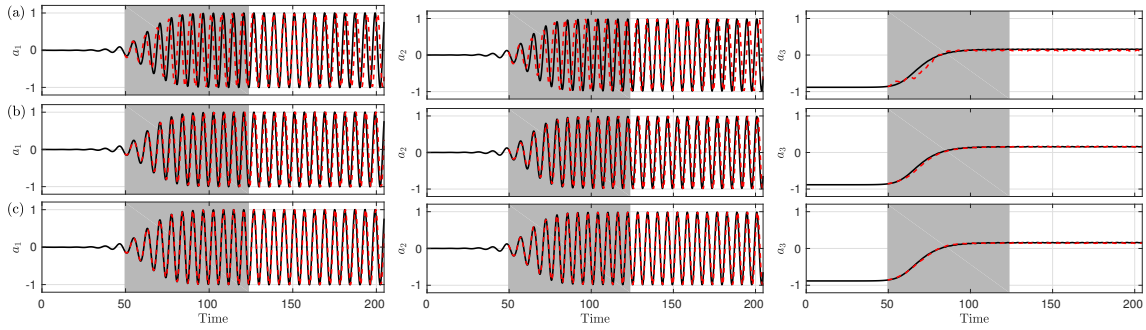


Figure 7.1: Times series of predicted $Re100$ POD features obtained from (a) 6 - MGOC -TS1 (b) 6 - MGOC - TS3 and (c) 6 - MGOC - TS9 compared with their respective original coefficients for TR-I region.

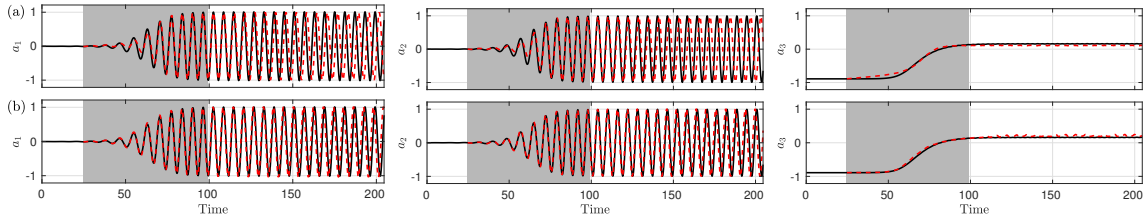


Figure 7.2: Times series of predicted $Re100$ POD features obtained from (a) 6 - MGOC - TS3 and (b) 6 - MGOC - TS9 compared with their respective original coefficients for TR-II region.

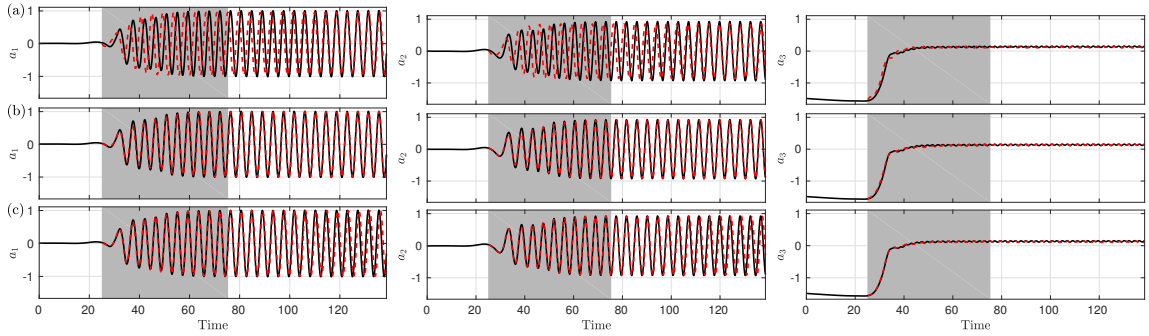


Figure 7.3: Times series of predicted $Re1000$ POD features obtained from (a) 6 - MGOC - TS1 (b) 6 - MGOC - TS3 and (c) 6 - MGOC - TS9 compared with their respective original coefficients in the TR-I.

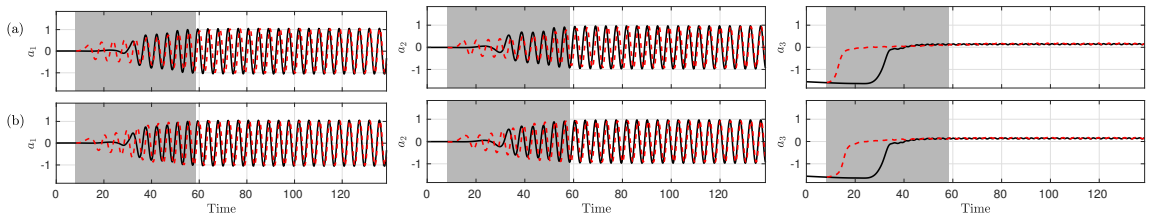


Figure 7.4: Times series of predicted $Re1000$ POD features obtained from (a) 6 - MGOC - TS3 and (b) 6 - MGOC - TS9 compared with their respective original coefficients in the TR-II.

7.2 Spectral Analysis of the Transient 2D Buoyant Boussinesq Mixing Flow

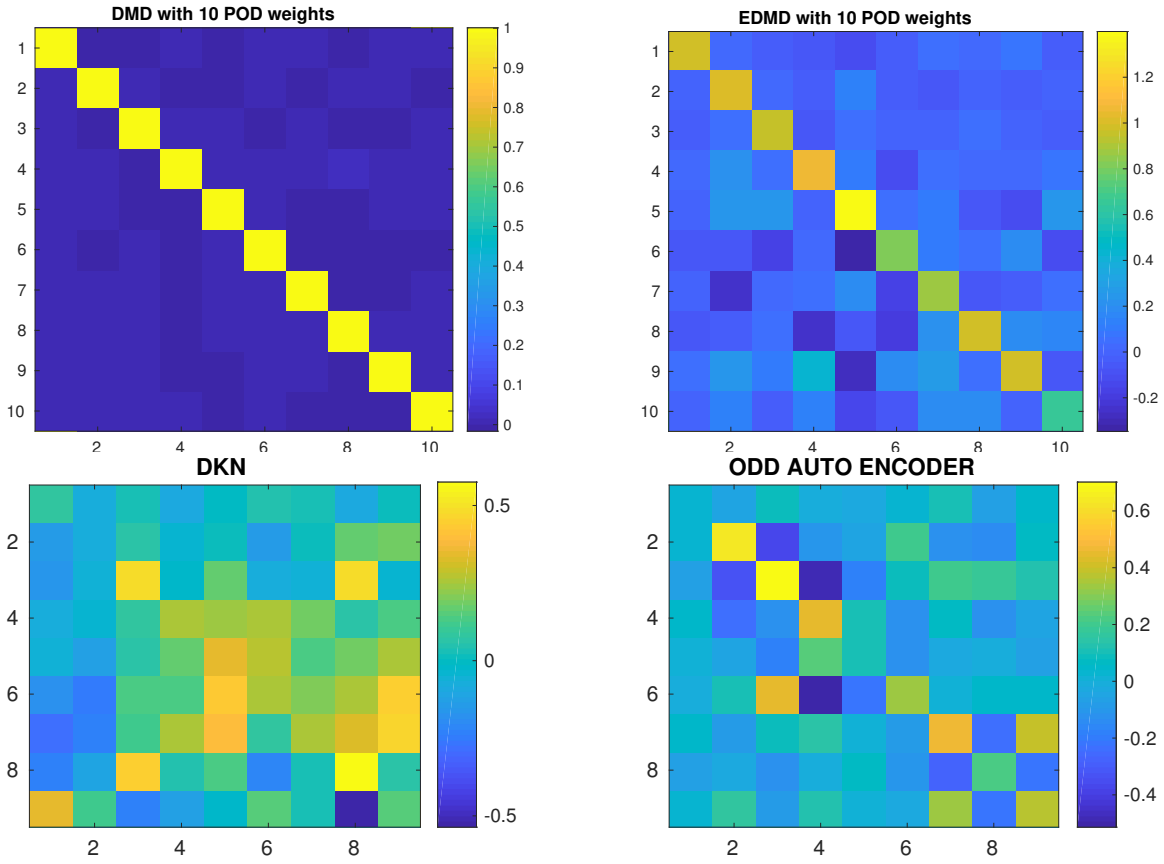


Figure 7.5: Comparison of Koopman operators obtained, where (1^{st} row): from DMD and EDMD and (2^{nd} row): 6 - HL Deep Koopman Network (DKN) and Auto-Encoder Network for 2-D Boussinesq Bouyant mixing flow

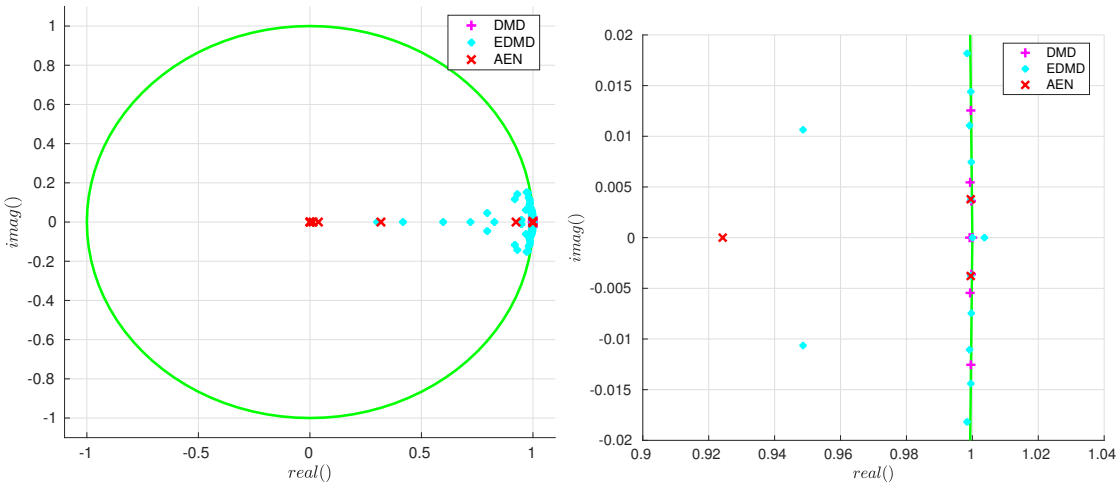


Figure 7.6: Ritz plot of eigenvalues(μ) spectrum obtained from 7 - HL AEN for 2-D Boussinesq Bouyant mixing flow

	DMD	EDMD	DKN-4	AEN-4	DKN-6	AEN-6
$\mu(St=0)$	$1 \pm 0i$	$1 \pm 0i$	$1 \pm 0i$	$0.997 \pm 0i$	$0.2627 \pm 0i$	$0.9242 \pm 0i$
λ_μ	0	0	0	-0.0239	-10.6375	-0.6273
$\mu(St=0.03)$	$0.99 \pm 0.0036i$	$0.99 \pm 0.0037i$	$0.9996 \pm 0.00383i$	$0.9995 \pm 0.00382i$	$0.9996 \pm 0.0037i$	$0.9995 \pm 0.00378i$
λ_μ	$-0.0799 \pm 0.0289i$	$-0.0799 \pm 0.0297i$	$-0.0031 \pm 0.0305i$	$-0.0039 \pm 0.0304i$	$-0.0031 \pm 0.0295i$	$-0.0039 \pm 0.0301i$
$\mu(St=0.06)$	$0.99 \pm 0.0054i$	$0.99 \pm 0.007456i$	$--$	$0.9989 \pm 0.0091i$	$--$	$--$
λ_μ	$-0.0799 \pm 0.0434i$	$-0.0798 \pm 0.0599i$	$--$	$-0.0084 \pm 0.0725i$	$--$	$--$
$\mu(St=0.09)$	$0.99 \pm 0.0125i$	$0.9995 \pm 0.144i$	$--$	$--$	$--$	$--$
λ_μ	$-0.0793 \pm 0.1005i$	$0.0778 \pm 1.1387i$	$--$	$--$	$--$	$--$

Table 7.1: Koopman eigenvalues(μ), growth rate ($Re(\lambda)$) and discrete frequencies ($Im(\lambda)$) obtained from 2-D Boussinesq Bouyant mixing flow.

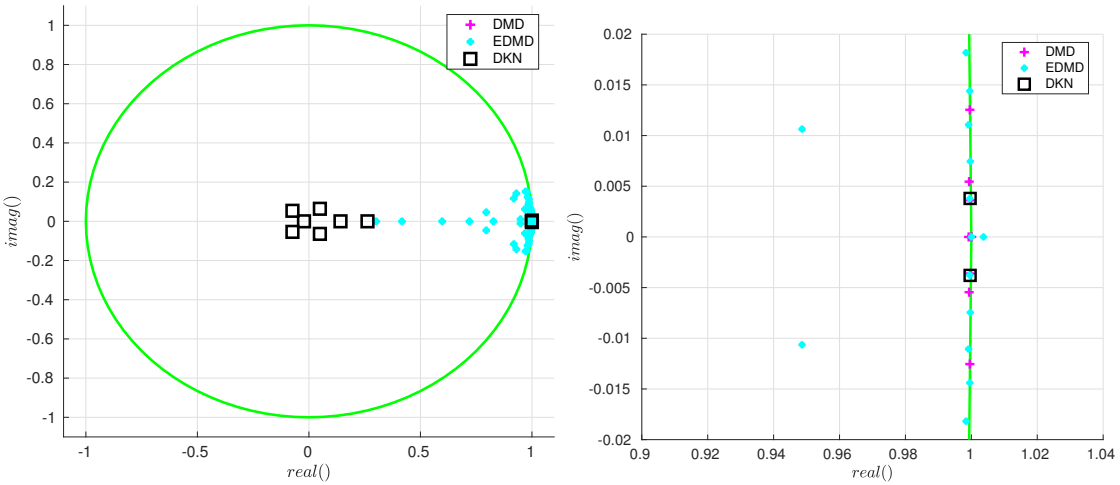


Figure 7.7: Ritz plot of eigenvalues(μ) spectrum obtained from 6 - HL DKN for 2-D Boussinesq Bouyant mixing flow

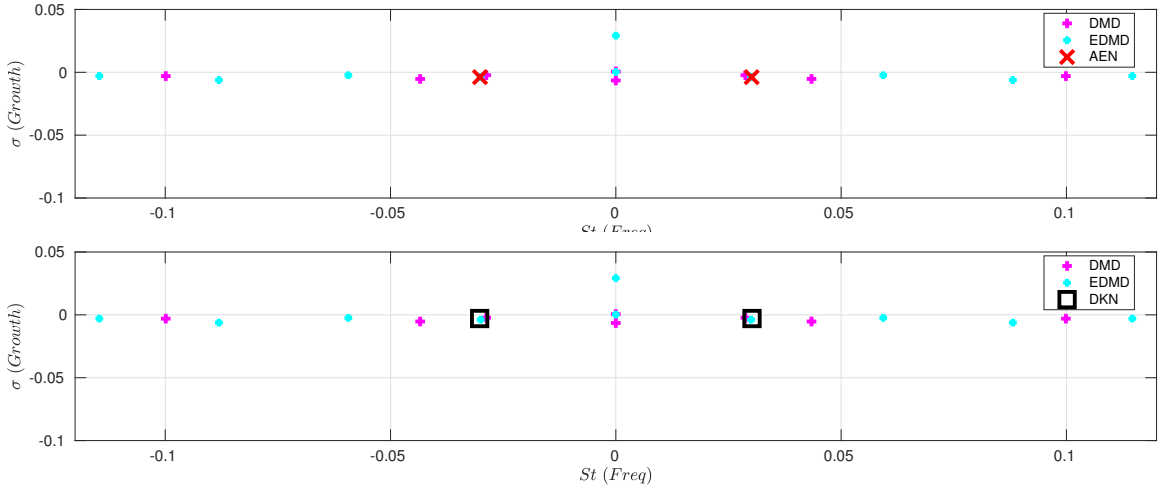


Figure 7.8: Comparison of growth rate and frequencies of (1^{st} row)7 - HL Auto-Encoder(AEN) and (2^{nd} row): 6 - HL Deep Koopman Network (DKN) for 2-D Boussinesq Bouyant mixing flow

7.3 Tutorial on Inhouse Deep Koopman Network Code and Backpropagation Algorithm

In this tutorial, a detailed description and usage of in-house MATLAB code used for generating FFNN, MFFNN, DKN and AEN results will be explained. The following are the important parameters needed to setup the code for any of the aforementioned architectures.

```
1  mcase      %%  Model case
2  Ncyc_s     %%  cycles #(num) starting for training
3  Ncyc_f     %%  cycles #(num) final for training
4  Ncyc_p     %%  cycles #(num) to predict from Ncyc_s
5  P          %%  #(num) of hidden layers
6  N          %%  #(num) of input features/POD weights
7  N_fac      %%  feature factor(# of neurons = N_fac * N)
8  cfunc      %%  transfer function in the encoder or FFNN or MFFNN
9  kfunc      %%  transfer function on the output layer
10 ifunc      %%  transfer function in the decoder or FFNN or MFFNN
11 FAO        %%  Flag for TF or inverse TF in the encoder
12 LAO        %%  Flag for TF or inverse TF in the decoder
13 CKT        %%  Flag for FFNN, DKN or AEN
14 BLP        %%  Weightage of cost functions in DKN
15 LBD        %%  logspace start, end, divisions for Regularization
16 opt_alg    %%  Flag for optimization method
17 param_ac   %%  relevant parameter value for Transfer function
18 incond_f   %%  Weights initialization(\Theta)
19 scaling    %%  Flag for scaling the input data
20 IMGFMT     %%  Output figures image format
21 num_iters  %%  # of iterations(epochs)
```

Further the following are the important distinctions between different architecture setups. For example

```
cfunc     %% FFNN, MFFNN, DKN, AEN : 'tansigmoid' or other TF ...
    (encoder part)
kfunc     %% FFNN, MFFNN, DKN, AEN : 'linear'
ifunc     %% FFNN, DKN, AEN : 'tansigmoid'; and  MFFNN: 'log'
FAO       %% FFNN, MFFNN, DKN, AEN : 'conv'
LAO       %% FFNN, DKN, AEN : 'conv'; and  MFFNN: inv_conv
CKT       %% FFNN or MFFNN : 0; DKN: 1; AEN: 54.
BLP       %% FFNN, MFFNN, AEN : 1.0; and generally for DKN : [1-2)
```

COnstructing diffeent architectures using *Tfunc* and *INV* for example: for a six hidden layer networks:

1. FFNN or DKN:

```
Tfunc = [repmat(cfunc,P/2,1);repmat(ifunc,P/2,1);kfunc];
INV = [repmat(Aconv(1),(P/2),1); repmat(Aconv(2),(P/2),1); ...
    Aconv(1)];
```

for 6 - HL with tansigmoid TF yields:

```
Tfunc ...
    ={'tansig','tansig','tansig','tansig','tansig','tansig','lin'}
INV = {'conv','conv','conv','conv','conv','conv','conv'}
```

2. AEN:

```
Tfunc = ...
    [repmat(cfunc,(P-1)/2,1);kfunc;repmat(ifunc,(P-1)/2,1);kfunc];
INV = [repmat(Aconv(1),(P-1)/2,1); Aconv(1); ...
    repmat(Aconv(2),(P-1)/2,1); Aconv(1)];
```

```
Tfunc ...
    ={'tansig','tansig','tansig','lin','tansig','tansig','tansig'
,'lin'}
INV = {'conv','conv','conv','conv','conv','conv','conv','conv'}
```

3. MFFNN:

```
Tfunc = [repmat(cfunc,P/2,1);repmat(ifunc,P/2,1);kfunc];
INV = [repmat(Aconv(1),(P/2),1); repmat(Aconv(2),(P/2),1); ...
    Aconv(1)];
```

for 6 - HL with tansigmoid TF yields:

```
Tfunc ={'tansig','tansig','tansig','log','log','log','lin'}
INV = ...
    {'conv','conv','conv','inv_conv','inv_conv','inv_conv','conv'}
```

7.3.1 Weights: Random Initialization

Zero initialization (symmetry). Not a good idea all the time (It might never converge). After each update, parameter corresponding to the inputs going into each of two hidden limits are identical.

Random initialization (symmetry breaking). Initialize each Θ_{ij}^{L} to a random value

in $[-\epsilon, \epsilon] \implies -\epsilon \leq \Theta_{i.j}^{l} \leq \epsilon$.

$$\Theta_l = (rand(M, N) * 2 * \epsilon) - \epsilon \tag{7.1}$$

7.3.2 Gradient Checking: Validating Backpropagation Algorithm

Approximating gradient and checking, using Central difference to approximating,

$$\frac{d}{d\theta_i} J(\theta) = \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon} \tag{7.2}$$

the above gradient help validate the backpropagation algorithm coded. Note:

- Implement Backdrop to compute D_{vec} (unrolled D^1, D^2, D^3)
- Implement numerical gradient check to compute grad approx.
- Make sure they produce similarities

7.3.3 P - HL Network for FFNN and DKN

1. Forward propagation

$$\begin{aligned} z_1 &= \theta_0 a_0 = \theta_0 X \\ a_1 &= g_1(z_1) \end{aligned} \tag{7.3}$$

$$\begin{aligned} z_p &= \Theta_{p-1} a_{p-1} \\ a_p &= g_p(z_p) \end{aligned} \tag{7.4}$$

$$a_{p+1} = g_{p+1}(z_{p+1}) \rightarrow [\hat{Y}] \tag{7.5}$$

Here the data is divided into training data and validation data.

```
1
2  %% ============== FEED FORWARD PROPAGATION=================
3  a0 = X;
4  Z(:,:,1)  = Theta0*a0;
5  Z_VAL(:,:,1)  = Theta0*Xval;
6  for i = 1:P-1
7  A(:,:,i)  = sigmoid(inv_var(Z(:,:,i),Tfunc{i},INV{i}),Tfunc{i});
8  Z(:,:,i+1) = THETA(:,:,i)*A(:,:,i);
9
10 A_VAL(:,:,i)  = ...
       sigmoid(inv_var(Z_VAL(:,:,i),Tfunc{i},INV{i}),Tfunc{i});
11 Z_VAL(:,:,i+1) = THETA(:,:,i)*A_VAL(:,:,i);
12
13 REG = THETA(:,:,i).*THETA(:,:,i);
14 REGN = REGN + sum(REG(:));
15 end
16 A(:,:,P) = sigmoid(inv_var(Z(:,:,P),Tfunc{P},INV{P}),Tfunc{P});
17 Zpp1 = Thetap*A(:,:,P);
18 HT  = sigmoid(inv_var(Zpp1,Tfunc{P+1},INV{P+1}),Tfunc{P+1});
19
20 A_VAL(:,:,P) = ...
       sigmoid(inv_var(Z_VAL(:,:,P),Tfunc{P},INV{P}),Tfunc{P});
21 Zpp1_VAL = Thetap*A_VAL(:,:,P);
22 HT_VAL  = ...
       sigmoid(inv_var(Zpp1_VAL,Tfunc{P+1},INV{P+1}),Tfunc{P+1});
```

2. Penalty network for symmetry (koopman)

$$\begin{aligned} cz_{k1} &= \theta_0 Y \\ ca_{k1} &= g_1(cz_k 1) \end{aligned} \tag{7.6}$$

$$cz_{kl} = \Theta_{kl-1} ca_{kl-1}$$
$$ca_{kl} = g_{kl}(cz_{kl}) \tag{7.7}$$

$$ca_{pkl} = g_{pkl}(cz_{pkl}) \rightarrow [\hat{Y}] \tag{7.8}$$

where $Pkl = P/2$.

```
%% ==========KOOPMAN THEORY CONSTRAINED FORWARD ...
    PROPAGATION=======
if CKT == 1
    ZCKT(:,:,1) = Theta0*y;
    ZCKT_VAL(:,:,1) = Theta0*yval;
    for ck = 1:Pkl-1
        ACKT(:,:,ck) = sigmoid(ZCKT(:,:,ck),Tfunc{ck});
        ZCKT(:,:,ck+1) = THETA(:,:,ck)*ACKT(:,:,ck);

        ACKT_VAL(:,:,ck) = sigmoid(ZCKT_VAL(:,:,ck),Tfunc{ck});
        ZCKT_VAL(:,:,ck+1) = THETA(:,:,ck)*ACKT_VAL(:,:,ck);
    end
    ACKT(:,:,Pkl) = sigmoid(ZCKT(:,:,Pkl),Tfunc{Pkl});

    ACKT_VAL(:,:,Pkl) = sigmoid(ZCKT_VAL(:,:,Pkl),Tfunc{Pkl});
end
```

3. Cost for FFNNN and DKN

$$J = \frac{1}{2m} \sum (\hat{Y} - Y)^2 \tag{7.9}$$

4. Cost with Penalty network (DKN)

$$J = \frac{1}{2m} \sum (\hat{Y} - Y)^2 + \frac{1}{2m} \sum (z_{pkl+1} - ca_{pkl})^2 \tag{7.10}$$

```
%%======================== COST FUNCTIONS ==============
JER = 0.0;
JCKT = 0.0;
yshift =sum((y - mean(y,2)).^2,2);
RDEN = zeros(size(y));
for b = 1:bat
     JER = JER + (((HT(:,b) - y(:,b))'*(HT(:,b) - y(:,b))));

if CKT == 1
JCKT  = JCKT + (ACKT(:,b,Pkl)  - ...
    Z(:,b,Pkl+1))'*(ACKT(:,b,Pkl)  - Z(:,b,Pkl+1));
end
     RDEN(:,b) =  ((HT(:,b) - y(:,b)).^2)./yshift(:);
end
JER   = JER./(2.0*bat);
JCKT  = JCKT./(2.0*bat);
RPI   = sum(ones(length(y(:,1)),1) - ...
    sum(RDEN,2))/length(y(:,1));

J = BLP.*JER  +(2 - BLP).*JCKT + REGN;
```

5. Back Propagation for 6 Hidden layer Network

$$\frac{dJ}{d\theta_6} = \underbrace{\frac{1}{m}(a_7 - Y)\frac{dg_7}{d\hat{z}_7} \cdot \frac{d\hat{z}_7}{dz_7}}_{\delta_7} \cdot a_6 + 0 \tag{7.11}$$

$$\frac{dJ}{d\theta_5} = \underbrace{\theta_6\delta_7\frac{dg_6}{d\hat{z}_6} \cdot \frac{d\hat{z}_6}{dz_6}}_{\delta_6} \cdot a_5 + 0 \tag{7.12}$$

$$\frac{dJ}{d\theta_4} = \underbrace{\theta_5\delta_6\frac{dg_5}{d\hat{z}_5} \cdot \frac{d\hat{z}_5}{dz_5}}_{\delta_5} \cdot a_4 + 0 \tag{7.13}$$

$$\frac{dJ}{d\theta_3} = \underbrace{\theta_4\delta_5\frac{dg_4}{d\hat{z}_4}\cdot\frac{d\hat{z}_4}{dz_4}}_{\delta_4}\cdot a_3 + \underbrace{\frac{1}{m}(z_4 - a_{3CKT})a_3}_{\delta_{CBP4}} \tag{7.14}$$

$$\frac{dJ}{d\theta_2} = \underbrace{\theta_3\delta_4\frac{dg_3}{d\hat{z}_3}\cdot\frac{d\hat{z}_3}{dz_3}}_{\delta_3}\cdot a_2 + \underbrace{\theta_3\delta_{CBP4}\frac{dg_3}{d\hat{z}_3}\frac{d\hat{z}_3}{dz_3}}_{\delta_{CBP3}}\cdot a_2 + \underbrace{(-\frac{1}{m})(z_4 - a_{3CKT})\frac{dg_3}{dcz_3})}_{\delta_{EBP3}}\cdot a_{2CKT} \tag{7.15}$$

$$\frac{dJ}{d\theta_1} = \underbrace{\theta_2\delta_3\frac{dg_2}{d\hat{z}_2}\cdot\frac{d\hat{z}_2}{dz_2}}_{\delta_2}\cdot a_1 + \underbrace{\theta_2\delta_{CBP3}\frac{dg_2}{d\hat{z}_2}\frac{d\hat{z}_2}{dz_2}}_{\delta_{CBP2}}\cdot a_1 + \underbrace{\theta_2\delta_{EBP3}\frac{dg_2}{dcz_2}}_{\delta_{EBP2}}\cdot a_{1CKT} \tag{7.16}$$

$$\frac{dJ}{d\theta_0} = \underbrace{\theta_1\delta_2\frac{dg_1}{d\hat{z}_1}\cdot\frac{d\hat{z}_1}{dz_1}}_{\delta_1}\cdot a_0 + \underbrace{\theta_1\delta_{CBP2}\frac{dg_1}{d\hat{z}_1}\frac{d\hat{z}_1}{dz_1}}_{\delta_{CBP1}}\cdot a_0 + \underbrace{\theta_1\delta_{EBP2}\frac{dg_1}{dcz_1}}_{\delta_{EBP1}}\cdot a_{0CKT} \tag{7.17}$$

```
%% == BACKPROPAGATION ================================
% %=== INITIALIZING FOR GENERAL NN=================
DELTA = zeros(Nhl,bat,P);
DJDT = zeros(Nhl,Nhl,P-1);
∆pp1 = zeros(ols,bat);
dJdTp = 0.0;
dJdT0 = 0.0;

% %======= INITIALIZING FOR CONSTRAINED NN===========
DELTA_CBP = zeros(Nhl,bat,Pkl +1);
DELTA_EBP = zeros(Nhl,bat,Pkl);
DJDT_CKT = zeros(Nhl,Nhl,Pkl);
dJdT0_CKT = 0.0;


for j = 1:bat
    ∆pp1(:,j) = (HT(:,j) - y(:,j)).*
     sigmoidGrad(inv_var(Zpp1(:,j),Tfunc{P+1},INV{P+1}), ...
         Tfunc{P+1})....
```

```
19          .*inv_var_grad(Zpp1(:,j),Tfunc{P+1},INV{P+1});
20
21      DELTA(:,j,P) = Thetap'*∆pp1(:,j).* ...
            sigmoidGrad(inv_var(Z(:,j,P),Tfunc{P},INV{P}),Tfunc{P})....
22          .*inv_var_grad(Z(:,j,P),Tfunc{P},INV{P});
23
24      for lay = 1:P-1
25          DELTA(:,j,P-lay) = THETA(:,:,P-lay)'*DELTA(:,j,P+1-lay)...
26              .*sigmoidGrad(inv_var(Z(:,j,P-lay),Tfunc{P-lay},INV{P-lay}), ...
                    Tfunc{P-lay})....
27              .*inv_var_grad(Z(:,j,P-lay),Tfunc{P-lay},INV{P-lay});
28
29          DJDT(:,:,P-lay) =  DJDT(:,:,P-lay) + ...
                DELTA(:,j,P+1-lay)*A(:,j,P-lay)';
30
31      end
32          dJdTp = dJdTp + ∆pp1(:,j)*A(:,j,P)';
33          dJdT0 = dJdT0 + DELTA(:,j,1)*a0(:,j)';
34
35  %%=============================================
36      if CKT == 1
37          DELTA_CBP(:,j,Pkl+1) = -(ACKT(:,j,Pkl) - Z(:,j,Pkl+1));
38
39          DELTA_EBP(:,j,Pkl) = (ACKT(:,j,Pkl) - Z(:,j,Pkl+1))...
40              .*sigmoidGrad(ZCKT(:,j,Pkl),Tfunc{Pkl});
41
42          for clay = 1:Pkl
43          DELTA_CBP(:,j,(Pkl +1 - clay)) = THETA(:,:,(Pkl + 1 - ...
                clay))'*DELTA_CBP(:,j, (Pkl+2 - clay))...
44              .*sigmoidGrad(Z(:,j,Pkl+1 - clay),Tfunc{Pkl+1 - clay});
45
46          DJDT_CKT(:,:,Pkl+1 -clay) = DJDT_CKT(:,:,Pkl+1 -clay) + ...
```

```
            DELTA_CBP(:,j,Pkl+2 - clay)*A(:,j,Pkl+1-clay)';
47
48              if ((Pkl + 1-clay) < Pkl)
49              DELTA_EBP(:,j,Pkl +1 -clay) = THETA(:,:,Pkl+1 - ...
                    clay)'*DELTA_EBP(:,j,Pkl+2 - clay)...
50              .*sigmoidGrad(ZCKT(:,j,Pkl+1-clay),Tfunc{Pkl+1-clay});
51
52              DJDT_CKT(:,:,Pkl+1-clay) = DJDT_CKT(:,:,Pkl+1-clay) + ...
                    DELTA_EBP(:,j,Pkl+2-clay)*ACKT(:,j,Pkl+1-clay)';
53
54              end
55          end
56          dJdT0_CKT =  dJdT0_CKT + (DELTA_EBP(:,j,1)*y(:,j)') + ...
                (DELTA_CBP(:,j,1)*a0(:,j)');
57
58      elseif CKT == 0 || CKT == 54
59      %%% NO CKT APPLIED
60      else
61      error('Provide CKT to either zero or one')
62      end
63  end
64  %%======= COMPUTING THE FINAL GRADIENTS =============
65
66  Theta0_grad = BLP.*dJdT0./bat + (lambda/bat).*Theta0 + ((2-BLP).* ...
        dJdT0_CKT)./bat ;
67  Thetap_grad = BLP.*dJdTp./bat + (lambda/bat).*Thetap;
68
69  for lay = 1:P-1
70      THETA_GRAD(:,:,lay) = BLP.*DJDT(:,:,lay)./bat + ...
            (lambda/bat).*THETA(:,:,lay);
71      if lay ≤ Pkl && CKT == 1
72      THETA_GRAD(:,:,lay) = THETA_GRAD(:,:,lay) + ...
```

```
          ((2-BLP).*DJDT_CKT(:,:,lay))./bat;
    end

end
%% === UNROLL GRADIENTS ========================
 GRAD = [Theta0_grad(:) ; THETA_GRAD(:); Thetap_grad(:)];
```

7.3.4 Post Processing Codes

Predictions using initial condition $X(0)$

```
function p = predict_CKNN(Theta0,THETA,Thetap, X,Tfunc,INV)
P = length(THETA(1,1,:))+1;
a0 = X;
Z = Theta0*a0;
for i = 1:P-1
A  = sigmoid(inv_var(Z,Tfunc{i},INV{i}),Tfunc{i});
Z = THETA(:,:,i)*A;
end
A = sigmoid(inv_var(Z,Tfunc{P},INV{P}),Tfunc{P});
Zp = Thetap*A;
p  = sigmoid(inv_var(Zp,Tfunc{P+1},INV{P+1}),Tfunc{P+1});
end
```

Encoder:

```
function [Xout] = ENCODER_CKNN(Theta0,THETA,Xin,Tfunc,INV,CKT)
P   = (length(THETA(1,1,:))+1);
%%==== FOR CKNN ============================
if mod(P,2) == 0 && CKT == 1
Pkl = (length(THETA(1,1,:))+1)/2;
```

```
Z = Theta0*Xin;
if Pkl ≥ 2
    for i = 1:Pkl-1
        A = sigmoid(inv_var(Z,Tfunc{i},INV{i}),Tfunc{i});
        Z = THETA(:,:,i)*A;
    end
end
Xout = sigmoid(inv_var(Z,Tfunc{Pkl},INV{Pkl}),Tfunc{Pkl});

%%====== FOR AUTOENCODER ====================================
elseif mod(P,2) == 1 && CKT == 54
Pkl = (length(THETA(1,1,:)))/2;
Z = Theta0*Xin;
if Pkl ≥ 2
    for i = 1:Pkl
        A = sigmoid(inv_var(Z,Tfunc{i},INV{i}),Tfunc{i});
        Z = THETA(:,:,i)*A;
    end
end
Xout = sigmoid(inv_var(Z,Tfunc{Pkl+1},INV{Pkl+1}),Tfunc{Pkl+1});
end


end
```

Decoder:

```
function[Xout] = DECODER_CKNN(THETA,Thetap,Xin,Tfunc,INV,CKT)
P    = (length(THETA(1,1,:))+1);
%%==== FOR CKNN ============================
if mod(P,2) == 0 && CKT == 1

```

```
Pkl = (length(THETA(1,1,:))+1)/2;
Amid = Xin;
if Pkl ≥ 2
    for i = Pkl+1:P-1
    Zmid = THETA(:,:,i)*Amid;
    Amid = sigmoid(inv_var(Zmid,Tfunc{i+1},INV{i+1}),Tfunc{i+1});
    end
end
Zmid = Thetap*Amid;
Xout = sigmoid(inv_var(Zmid,Tfunc{P+1},INV{P+1}),Tfunc{P+1});

%%====== FOR AUTOENCODER ====================================
elseif mod(P,2) == 1 && CKT == 54

Pkl = (length(THETA(1,1,:)))/2;
Amid = Xin;
if Pkl ≥ 2
    for i = Pkl+1:P-1
    Zmid = THETA(:,:,i)*Amid;
    Amid = sigmoid(inv_var(Zmid,Tfunc{i+1},INV{i+1}),Tfunc{i+1});
    end
end
Zmid = Thetap*Amid;
Xout = sigmoid(inv_var(Zmid,Tfunc{P+1},INV{P+1}),Tfunc{P+1});
end



end
```

```
%%%=================== WRITE THE STATS TO A FILE ...
```

```
    ========================
save (['THETA_FINAL_' num2str(ger) ...
    '.mat'],'Theta0','THETA','Thetap');

if CKT == 0
save(['YPRED_FINAL_' num2str(ger) ...
    '.mat'],'X','y','yt','Tpred','Ypred','Ypred_n')
elseif CKT == 1
save(['YPRED_FINAL_' num2str(ger) ...
    '.mat'],'X','y','yt','Tpred','Ypred','Yp_inner','Yp_lin','Ypred_n')
save(['YK_PRED_FINAL_' num2str(ger) ...
    '.mat'],'YK','YK_pred','YK_pred_lin','EKP')
THETA_KOOPMAN = THETA(:,:,int8((P)/2));
save(['EIG_KOOPMAN' num2str(ger) ...
    '.mat'],'U_K','D_K','V_K','THETA_KOOPMAN','THETA_LIN')
clear THETA_KOOPMAN
elseif CKT == 54
save(['YPRED_FINAL_' num2str(ger) ...
    '.mat'],'X','y','yt','Tpred','Ypred','Yp_inner','Ypred_n')
save(['YK_PRED_FINAL_' num2str(ger) '.mat'],'YK','YK_pred','EKP')
save(['EIG_KOOPMAN' num2str(ger) ...
    '.mat'],'U_K','D_K','V_K','THETA_LIN')
end


fprintf('\n Pred Error on Training set: %5.5g\n',T_error );
fprintf('\n Pred Error on Non-Training set:%5.5g\n',P_error );
fprintf('\n Pred Error on Non-Training set+noisy input: ...
    %5.5g\n',P_error_n );
fprintf('\n%8s\t\t%8s\t\t%8s\n','Mean','Range','Variance');
fprintf('\n%8.5f\t\t%8.5f\t\t%8.5f %15s ...
    \n',TMean_error,TRange_error,TVariance_error,'TrainSet' );
```

```
24 fprintf('\n%8.5f\t\t%8.5f\t\t%8.5f %15s ...
       \n',YMean_error,YRange_error,YVariance_error,'nonTrainSet' );
25
26
27 if CKT == 1 || CKT == 54
28     pred_stat = [ger,lambda,BPerror,J,JER, ...
           JCKT,REGN,T_error,P_error,P_error_n,E_AN,YKP_error, ...
           YMean_error,...
29                 YRange_error, ...
                       YVariance_error,norm(THETA_ER),PRED_ER, ...
                       double(rng_seed)];
30 else
31     pred_stat = [ger,lambda,BPerror,J,JER, ...
           JCKT,REGN,T_error,P_error,P_error_n,YMean_error,...
32                 YRange_error, ...
                       YVariance_error,norm(THETA_ER),PRED_ER, ...
                       double(rng_seed)];
33 end
34
35 stats_plot(ger,:) = pred_stat;
```

VITA

Shivakanth Chary Puligilla

Candidate for the Degree of

Masters of Science

Thesis: DEEP MULTILAYER CONVOLUTION FRAMEWORKS FOR DATA-DRIVEN LEARNING OF NONLINEAR DYNAMICS IN FLUID FLOWS

Major Field: Mechanical and Aerospace Engineering

Biographical:

Personal Data: Born in Hyderabad, India on 23^{rd} September of 1988.

Education:

1. Received a Bachelor of Technology in Aeronautical Engineering from Jawaharlal Nehru Technological University, Hyderabad, India in May 2010.
2. Received a Master of Science in Aerospace Engineering from Indian Institute of Science, Bengalore, India in July 2017.
3. Completed the requirements for the degree of Master of Science with a major in Mechanical and Aerospace Engineering at Oklahoma State University in July, 2018.

Experience: Graduate research and Teaching assistant for two years at MAE, OKSTATE.

Google Scholar: `https://scholar.google.com/citations?user=YiOa4jkAAAAJ&hl=en`

Linkedin: `https://www.linkedin.com/in/shivakanth-chary-702a0b38/`

Professional Affiliations: Student member: American Institute of Aeronautics and Astronautics (AIAA).
Student member: American Physical Society (APS).
Student member: Society for Industrial and Applied Mathematics (SIAM).