Learning data-driven reduced-order models of complex flows

by

Carlos E. Pérez De Jesús

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Chemical and Biological Engineering)

at the

UNIVERSITY OF WISCONSIN - MADISON

2024

Date of final oral examination: May 22, 2024

"Preparemos el terreno

Para sembrar la semilla."

– Celestina Cruz Jiménez, *La Cosecha*

Dedication

A mi familia:

Papi y mami, gracias por ser modelos a seguir, por inculcar en mí valores, y por apoyarme en las buenas y en las malas. Gracias por siempre creer en mí y enseñarme la importancia de la educación. Obtener este grado académico hubiera sido casi imposible sin tenerlos ustedes a mi lado. Alain y Mateo, gracias por ser los mejores hermanos. Alain, gracias por siempre estar ahí para mí. Admiro tu pasión por lo que haces y el apoyo que das a todos los que amas. Mateo, gracias por llenar nuestra familia de tanta felicidad. Te deseo lo mejor mientras vas creciendo y aprendiendo.

Abuela Carmen, abuela Aurea, abuela Tina y abuelo Manolo, gracias por todo el apoyo que me han brindado. Gracias a usted he aprendido la importancia de siempre mantener la fe. Padrino Men, tío Fanchy, tío Javi, tía Diana, tío Tony, tía Katia y madrina Lourdes, gracias por siempre estar ahí mientras crecía y ser una parte tan importante en mi desarrollo como persona. Primos Valerie, Diego, Gaby, Francis, Zohary y Katiria, los mejores momentos los he pasado criándome con ustedes.

Mandy, gracias por llegar a mi vida durante mis años de escuela graduada. Gracias a ti aprendí lo que es amar a una mascota. Escuchar tus maullidos en las mañanas pidiendo comida siempre me llena de felicidad.

To my friends:

Juan and Jean, thanks for being there for me and giving me all the support I needed in my undergraduate days. Juan, I learned so much from you about how to communicate

in english. Thanks to you I always felt confident when I had to write something up. Jean, growing up with you has been the best. I admire how you are always calm and collected and the level of trust that everyone has in you.

Bryce, RJ, Atharva, Ricardo, Alec, Kevin, Eric and Xiaopo, thanks for being amazing friends and helping me throughout my years in graduate school. Bryce, one of the best parts of graduate school was all of the super fun concerts we went to. Thanks for always caring about me and even cooking for me when you saw me stressed about work. RJ, sharing the office with you has been an amazing time. I always looked forward to our daily coffee with Ricardo and Kevin, and our weekend TV and pizza nights. Ricardo, thanks for always helping me when I ask you for help. I have learned so much from you and I wish you the best as a future professor. Alec, getting to brainstorm with you about science has been so fun. Thanks for always helping me understand challenging science concepts. Kevin, thanks for always helping me when I felt overwhelmed. Thanks to you I learned a lot about how to approach science problems. Eric I always admire how thorough you are in your work. I am excited to see all the amazing science problems you will tackle in the future. Xiaopo I got to learn so much from you about coding. Thanks so much for taking your time and explaining to me these concepts.

Thanks to all of the amazing friends I have met in my graduate school years: Will, Lawrence, Gary, Zach, Erin, Christine, Maya, Jack, Kyle, Javiera, Jadiel, Carlos Huang, Kevin Sánchez, Jaron, Nestor and Paolo. Thanks to all of my friends from my undergraduate institution and hometown: Oneil, Alberto, Susana, Jonathan and John. And of course, thanks to all the super smart and supportive group members in the Graham group, past and present: Manish, Jake, Alex, Daniel, Andrew, Ashwin, Charlie, Joysy, and Rafa. I have learned so much from all of you during my time in the group.

To my advisor and committee:

Professor Mike Graham, thanks for taking me in under your guidance and introducing me to this amazing world of research. Your support has been amazing through all this years.

I really appreciate how much you care about your students and this supportive environment has been an important reason of all the work I have accomplished during graduate school. Finally, I would like to thank my committee for following my growth all these years and giving me great feedback on my work.

Abstract

Dynamical models play a crucial role for understanding and solving problems in many engineering applications or natural systems. In turbulence, weather modeling, chemical processes, and other interesting areas in engineering it is desired to find reduced-order models that can make time predictions. The nature of high-dimensionality in fluid systems and recent advances in machine learning have pushed the boundaries of what can be learned when data and physical knowledge of a system is available. The objective of this thesis is to develop deep learning architectures to learn efficient reduced-order models that can faithfully capture the most important features of flows in a low-dimensional representation. We leverage the use of autoencoders to learn low-dimensional representations and dense neural networks to learn an evolution equation on this low-dimensional space. By enforcing symmetry constraints that appear in the Navier-Stokes equations we show how more accurate models for time prediction can be learned, while reducing significantly the dataset. Finally, we present a framework capable of giving estimates of the minimal dimensions needed to represent systems featuring complex dynamics and intricate behavior.

Contents

List of Figures

1

General introduction

Turbulence is notoriously difficult to study and predict due to its nonlinear and chaotic behavior. The flow is "chaotic" in the sense that any small changes in initial conditions of the velocity field will result in trajectories diverging after some amount of time. Turbulent flows can be observed in many industrial and commercial applications, particularly those that involve pumping or transporting fluids. It has been estimated that nearly half of energy consumption in fluid transport in pipes is dissipated by turbulence near the walls. A main issue when flowing a fluid in the turbulent regime (or high velocities) is the increase in drag, giving rise to energy losses as well as CO_2 emissions. As an example, wall-bounded flows account for about 5% of the global carbon emission footprint [31]. Hence, drag reduction is key to reducing energy costs and this can be achieved by implementing control schemes, such as wall actuations that push and pull fluid within the domain to drive the system towards a low drag state. The Navier-Stokes equations (NSE) describes the time evolution of fluids. By tracking the velocities over the domain of interest one can understand the behavior of the system, and subsequently decrease the drag. However, the turbulent nature of the flows requires many grid points (or sensors) to correctly resolve all the relevant spatial and temporal scales of the system. This makes the system high dimensional which leads to challenges in the search of control strategies. Hence, there is a need to find simpler ways to achieve this goal. One way is to build reduced-order models (ROMs), which contain the essential information, and are useful because they can be used to simulate fluids much faster. The ideal ROM has minimal degrees of freedom compared to the full velocity field data, with reduction of up to three orders of magnitude, and the necessary information to evolve in time such that statistics and features of the flow are faithfully captured.

In this work we leverage recent advances in machine learning in order to learn data-driven ROMs. Then, we do the modeling of turbulence from a dynamical system point of view (we learn the right-hand-side of the ordinary differential equation). Most of the focus throughout this thesis is in a two-dimensional flow problem known as Kolmogorov flow which is driven by a sinusoidal force and can exhibit chaotic dynamics. This system is of interest because it shows intermittent behavior, which is common in many flow processes, is challenging to model, and captures the essence of fluid turbulence.

To learn data-driven ROMs we use a combination of different variations of autoencoders (AE) to reduce dimensions and dense neural networks (NN) to evolve in time. We show in this thesis that we are able to learn efficient models that can capture important features of flows. In addition, we also include symmetries of the system that leads to the improving of the model performance. Finally, we present an architecture that automatically estimates the minimal dimensions needed to represent complex systems.

In the following introduction we start by motivating the need to learn reduced-order models. This is followed by an introduction to manifolds. Then, we discuss dimension reduction techniques, neural networks, and finish by motivating the need to understand symmetries in dynamical systems and ways to address them.

1.1 Building Reduced-Order Models

Development of reduced-order models (ROM) for complex flows is an issue of long-standing interest, with applications in improved understanding, as well as control of flow systems. For flows and many other cases, the dynamical system can be written as

$$\frac{dx}{dt} = f(x, t; \beta), \tag{1.1}$$

whose right-hand-side (RHS) is defined as the function f, and β correspond to the parameters of the equations. In the case of the Navier-Stokes Equations (NSE) this could be the density, viscosity, and the length scale. Analytical solutions are only known for simple problems. Then, defining an initial condition and boundary conditions one can solve this "infinte-dimensional" dynamical system using numerical methods. In many flow problems, an issue that arises is the high-dimensionality of the grid (discretization of the equations) that is necessary to correctly represent the state.

Due to this high-dimensionality, it is important to find ROMs for computational speed and to apply control strategies [41]. A successful ROM should capture the important features and dynamics of the true system. In this thesis we use solely data to learn ROMs. The data comes from direct numerical simulations that results from solving the governing equations of motion. Then, the dynamical system for a low-dimensional representation h is given as

$$\frac{dh}{dt} = g(h), \tag{1.2}$$

where $x \mapsto h$. This representation h can be estimated using different approaches. Principal Component Analysis (PCA) provides the best linear representation of a data set, discussed in Section 1.3.1, and gives a set of orthogonal basis vectors that are ordered by the total contribution to the energy of the flow. Projecting onto this basis, and doing a Galerkin approximation results in a classical framework for a ROM. However, in this scenario the equations need to be known, the basis is projected onto the equations to find the model. The major limitation of PCA is that it assumes that the subspace is flat, which is not true for complex chaotic nonlinear systems. Hence, a nonlinear transformation is desired. Popular nonlinear methods for dimension reduction include kernel PCA, diffusion maps, local linear embedding (LLE), isometric feature mapping (Isomap), and t-distributed stochastic neighbor embedding (tSNE) [64]. A drawback of these methods is that they do not provide the function that maps $h \mapsto \hat{x}$.

In this thesis we will consider neural networks (NN) to learn the mapping from $x \mapsto h$. These have shown great success in different flow and dynamical systems as will be shown in further chapters. We specifically want to focus on finding the minimal dimensions needed to capture the data manifold and dynamics.

1.2 Manifolds

For dissipative systems, such as the NSE, it is known that the long-time dynamics will collapse on an manifold $\mathcal{M}$, which has fewer dimensions than the original state space. A depiction of this manifold is shown in Figure 1.1. In fluid mechanics, this manifold is often called an *inertial manifold* [24, 62, 71]. As discussed by Lee [35] this manifold $\mathcal{M}$ of dimension n has some properties that we discuss in this section. The first being that it is a Hausdorff space. This means that for every pair of distinct points $p, q \in \mathcal{M}$ there are disjoint open subsets $U, V \subseteq \mathcal{M}$ such that $p \in U$ and $q \in V$. Hence, the pair is separated but lives in the manifold. The second property is that $\mathcal{M}$ is second-countable which means that there exists a countable basis for the topology of $\mathcal{M}$. The third property is that this manifold is locally Euclidean of dimension n.

This third property implicates that we can find a homeomorphism $\varphi : U \to \widehat{U}$ *locally.* This means that to find a map to the dimension $\mathbb{R}^{d_\mathcal{M}}$ of $\mathcal{M}$ one would need to "cut" the manifold into local representations. A global map of dimension $\mathbb{R}^{d_\mathcal{M}}$ is not guaranteed.

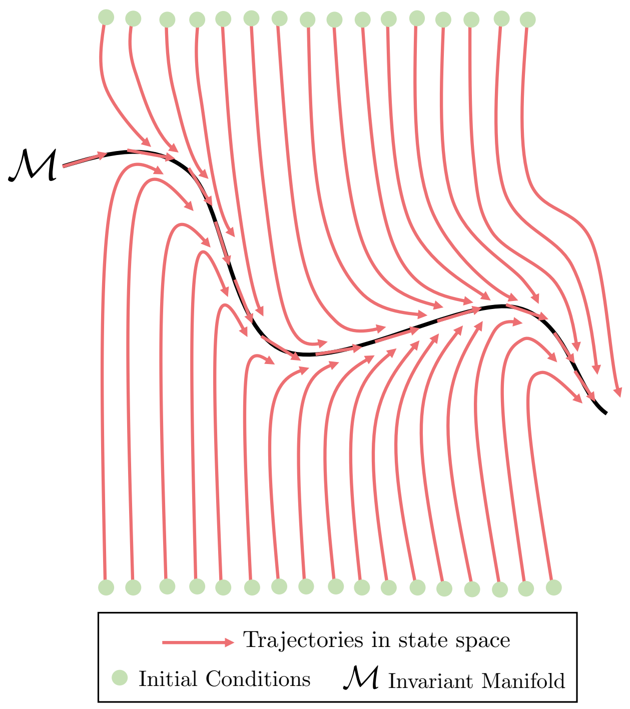


Figure 1.1: Schematic of state space with initial conditions collapsing onto an invariant manifold.

However global map to $\mathbb{R}^{2d_{\mathcal{M}}}$ does exist, giving an upperbound [65].

1.2.1 Coordinate charts

As discussed by Lee [35] we formalize in this section the concept of charts which will motivate our work on finding minimal dimension models. A coordinate chart on $\mathcal{M}$ is a pair (U, φ) where U is an open subset of $\mathcal{M}$ and $\varphi : U \to \widehat{U}$ is a homeomorphism from U to an open subset $\widehat{U}$. Here U is a coordinate domain and φ is a coordinate map. To form the manifold one needs a collection of many charts that will cover the manifold. This collection is called an atlas.

1.3 Dimension Reduction and Neural Networks

1.3.1 Dimension Reduction with Principal Component Analysis

Principal Component Analysis (PCA), also known as proper orthogonal decomposition (POD) and Karhunen-Loéve decomposition [27] seeks a linear transformation such that data is projected into an orthogonal coordinate system. These coordinates are organized by variance and in the case of velocity fields correspond to the energy content. Given N_s data vectors ("snapshots") $x_i \in \mathbb{R}^N$, one can obtain these basis vectors by performing singular value decomposition (SVD) on the data matrix $X = [x_1, x_2, \cdots] \in \mathbb{R}^{N \times N_s}$ such that $X = U\Sigma V^T$. Note, this U and V are different from the one in the previous section. Projecting the data onto the first d_h basis vectors (columns of U) then gives a low-dimensional representation – a projection onto a linear subspace of the full state space.

1.3.2 Neural Network Operations and Autoencoders

In this section we show the operations involved in NNs. Tensors and vectors are bolded for a clearer understanding, however in the following sections of this thesis we will sometimes drop this convention. NNs, are a machine learing model which contain units or neurons, that are connected and form what is called a network. These units are scalar values and weights $\boldsymbol{w} \in \mathbb{R}^m$ connect these through the network as seen in Figure 1.2a where $m = 3$ and $\boldsymbol{w} = (w_{1,1}, w_{2,1}, w_{3,1})$ maps the input neuron x_1 into the vector $\boldsymbol{y} = (y_1, y_2, y_3)$ with the operation $\boldsymbol{y} = \boldsymbol{w}x_1$. Let us now look at the NN in Figure 1.2b. This type of NN is called dense because all the input units are connected with the output units. The output $\boldsymbol{y}$ now takes the form of $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$, where the matrix $\boldsymbol{W} \in \mathbb{R}^{3\times3}$ operates on the input $\boldsymbol{x} \in \mathbb{R}^3$. In this case the weight matrix contains the individual weight vectors that connect the input neurons where $\boldsymbol{W} = [\boldsymbol{w}_1^T \boldsymbol{w}_2^T \boldsymbol{w}_3^T]$.

The values that $\boldsymbol{W}$ takes come from an optimization problem using gradient descent, where the goal is for the NN to achieve a specific task. One of the tasks is to reduce the

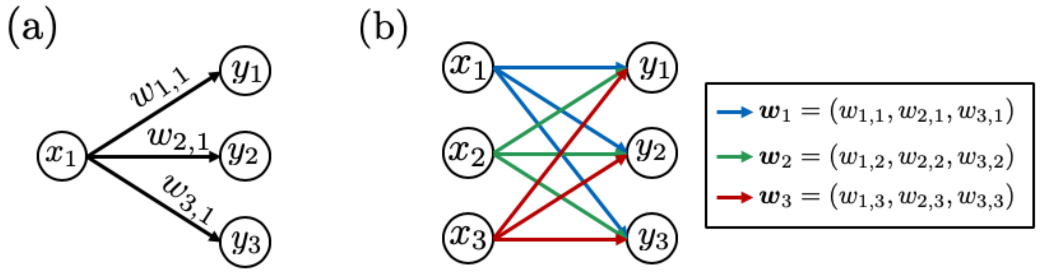

(a)
$w_{1,1}$
$w_{2,1}$
$w_{3,1}$
x_1
y_1
y_2
y_3
(b)
x_1
x_2
x_3
y_1
y_2
y_3
$\boldsymbol{w}_1 = (w_{1,1}, w_{2,1}, w_{3,1})$
$\boldsymbol{w}_2 = (w_{1,2}, w_{2,2}, w_{3,2})$
$\boldsymbol{w}_3 = (w_{1,3}, w_{2,3}, w_{3,3})$

Figure 1.2: (a) Linear NN operation: single neuron input (b) three neurons input

dimensions of an input $\boldsymbol{x}$ to a lower dimensional representation in space $\boldsymbol{h}$ and from this space reconstruct $\hat{\boldsymbol{x}}$ (i.e. $\hat{\boldsymbol{x}} \approx \boldsymbol{x}$). The NN for this case can be seen in Figure 1.3 (for the sake of simplicity, we assume $\boldsymbol{b}_i = \boldsymbol{0}$). The weights then take the form $\boldsymbol{W} = \boldsymbol{W}_2\boldsymbol{W}_1$ and the output is $\hat{\boldsymbol{x}} = \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}$. This type of NN takes the name of an autoencoder (AE) where the mapping $\boldsymbol{x} \mapsto \boldsymbol{h}$ is called the encoder and $\boldsymbol{h} \mapsto \hat{\boldsymbol{x}}$ the decoder. The weights come from optimizing what is called the cost or loss function $\mathcal{L}$. We consider the mean squared error (MSE) where $\mathcal{L} = \frac{1}{k_d \cdot q} \sum_{i=1}^{q} \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2^2$. Here k_d corresponds to the dimension of $\boldsymbol{x}$, in the case of a 32×32 flow field $k_d = 32^2$, and q to the number of number of data. Using $\mathcal{L}$ to optimize a NN can prove to be quite computationally expensive, hence mini-batch Stochastic Gradient Descent (SGD) can be used to optimize $\boldsymbol{W}$ where a random batch of data size $p \ll q$ is used to compute $\mathcal{L}_p = \frac{1}{k_d \cdot p} \sum_{i=1}^{p} \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2^2$ then the weights can be updated by

$$w_{i,j}^{m+1} = w_{i,j}^{m} - \eta \frac{\partial \mathcal{L}_p}{\partial w_{i,j}}, \tag{1.3}$$

where η is a tunning parameter called the learning rate. With backpropagation the derivative $\partial \mathcal{L}_p / \partial w_{i,j}$ is computed, by repeatedly using the chain rule.

Nonlinearities can be introduced in the AE as a function $\sigma(n_i)$ where n_i corresponds to different neurons in the network. A linear operation involves $\sigma(n_i) = n_i$ while common nonlinear functions used in machine learning include the Rectified linear unit (ReLU), which is defined as $\sigma(n_i) = \max(n_i, 0)$, and the tanh function, which takes the form $\sigma(n_i) =$

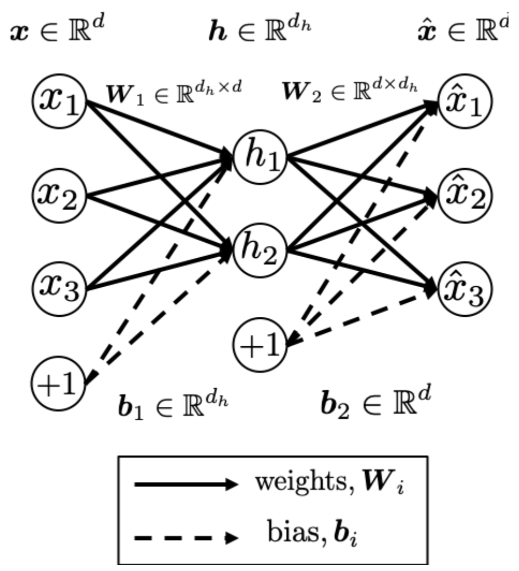

$\boldsymbol{x} \in \mathbb{R}^d$
$\boldsymbol{h} \in \mathbb{R}^{d_h}$
$\hat{\boldsymbol{x}} \in \mathbb{R}^d$
$\boldsymbol{W}_1 \in \mathbb{R}^{d_h \times d}$
$\boldsymbol{W}_2 \in \mathbb{R}^{d \times d_h}$
x_1
x_2
x_3
$+1$
h_1
h_2
$+1$
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\boldsymbol{b}_1 \in \mathbb{R}^{d_h}$
$\boldsymbol{b}_2 \in \mathbb{R}^d$
weights, $\boldsymbol{W}_i$
bias, $\boldsymbol{b}_i$

Figure 1.3: Linear autoencoder structure as depicted by a dense NN

$\tanh(n_i)$. If nonlinearities were to be introduced in Figure 1.3, the mapping from $\boldsymbol{x}$ to $\boldsymbol{h}$ would take the form $\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x})$, where the function σ operates elementwise on $\boldsymbol{W}\boldsymbol{x}$. For the sake of simplicity and to be able to directly connect ideas of linear algebra with NNs, an extra parameter that comes into play which is called the bias $\boldsymbol{b}_i$ was set to zero. This parameter comes into play as an addition to the weight operation on the input and appears as $\boldsymbol{h} = \sigma(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1)$. This parameter is also updated with Equation 1.3. Here, linear networks refer to $\hat{\boldsymbol{x}} = \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}$, and nonlinear networks refer to the inclusion of σ and $\boldsymbol{b}$.

1.3.3 Learning Time Maps

After training an AE we can obtain $h(t)$, to learn the data-driven ROM. Different approaches in literature consist in using long short-term memory (LSTM) and recurrent neural networks (RNN) [20, 46]. However, we know that the NSE are Markovian in nature. This means that we only need the sate at time t to evolve the system to time $t + \tau$. In this section we discuss our approaches to learn the time evolution where $h(t)$ is mapped to $h(t + \tau)$.

Discrete Time Map

After learning a low dimensional representation $h(t)$, we can seek a discrete time map

$$h(t+\tau) = F(h(t)) \tag{1.4}$$

that evolves $h(t)$ from time t to $t+\tau$. The function F can be expressed as a dense NN which is trained with the following loss

$$L_t = \|\tilde{h}(t+\tau) - h(t+\tau)\|^2, \tag{1.5}$$

where $h(t+\tau)$ comes from true data and $\tilde{h}(t+\tau) = F(h(t))$ from the prediction.

Neural Ordinary Differential Equations

Another framework to forecast in time is the the neural ODE (NODE) [14, 38, 40] which are continuous models and has shown to be successful at this task. We use in this thesis a stabilized version used by Linot et. al. [40] where the dynamics on the manifold are described by the equation

$$\frac{dh}{dt} = g_h(h) - Ah. \tag{1.6}$$

Here A is chosen to have a stabilizing effect that keeps solutions from blowing up. The equation is integrated to estimate $h_r(t+\tau)$ as

$$h_r\left(t+\tau\right) = h\left(t\right) + \int_t^{t+\tau} (g_h\left(h(t'); \theta_t\right) - Ah) dt' \tag{1.7}$$

where θ_t are the weights of the NN g_h which are determined by minimizing the loss

$$\mathcal{L}_{\text{NODE}}\left(h; \theta_t\right) = \left\langle \|h(t+\tau) - h_r(t+\tau)\|_2^2 \right\rangle. \tag{1.8}$$

1.4 More Neural Network Frameworks

1.4.1 Dropout

Overfitting is a common problem in networks with large number of parameters. To address this, dropout has been introduced as a technique that helps prevent units from co-adapting [60]. During training random units in the network are dropped, resulting in the training of "thin" networks. As discussed in [60], the feed-forward operation becomes

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p) \\ \widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)} \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^{l} + b_i^{(l+1)} \\ y_i^{(l+1)} &= f\left(z_i^{(l+1)}\right) \end{aligned} \tag{1.9}$$

where $*$ is an element-wise product, l is the layer, and $\mathbf{r}$ is a vector of independent Bernoulli random variables each of which has a probability p of being 1. In the optimization step the active units get updated, only these affect the loss. When testing, all the units are always present and are multiplied by p.

1.4.2 Contrastive Learning

The goal in contrastive learning is to learn representations by maximizing agreement between augmented versions of the same data via a contrastive loss [15]. These augmented versions correspond to a positive pair. In image classification tasks these can be: crops, color distortions, and gaussian blur to mention a few. In the context of an autoencoder, to apply the contrastive loss, h from an encoder can be extracted and combined with a projection head $s(\cdot)$ that maps to the space where the loss in applied, z. The loss function for a positive pair of examples (i, j) is defined as

$$\ell_{i,j} = -\log \frac{\exp\left(\operatorname{sim}\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp\left(\operatorname{sim}\left(z_i, z_k\right)/\tau\right)} \tag{1.10}$$

where $\operatorname{sim}(u, v) = u^\top v / \|u\| \|v\|$, $\mathbf{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function, and τ is a hyperparameter.

1.5 Symmetries in Dynamical Systems

We finalize this introduction by motivating the need to understand symmetries in dynamical systems. The presence of symmetries can prove to play a crucial role when learning models from data. This is in part because these have to spend efforts in learning the different copies of the same data.

Previous work in computer vision has shown that the performance of NN improves when addressing the symmetries of the problem considered, as opposed to data augmentation. Winkels & Cohen applied this concept to pulmonary nodule detection where they used 3D discrete rotation equivariant convolutional neural networks in CT scans [67, 68]. They showed a 10x increase in performance in terms of the data used for training as opposed to using regular CNNs. This means that when the symmetries of the system are known, it desired to address them instead of resorting to data augmentation. In dynamical systems this has shown to also be the case. Linot & Graham addressed the continuous symmetry of the Kuramoto-Sivashinsky equation with improvements in dimension estimates and MSEs and in this thesis (Chapters 2 and 3) we show how this improves models for the NSE.

Addressing symmetries is of importance because any successful attempt to learn ROMs involves dense coverage of the regions that the trajectories explore which are related to the symmetries of the system. Let G be a group of symmetries acting on a dynamical system $f(x)$. The group of symmetries is said to be equivariant given $f(Gx) = Gf(x)$. This means that any symmetry operation applied to the given state will give the same dynamics under a transformation, making these equivalent. In the invariant case $f(Gx) = f(x)$. Hence dynamics will be the same. In the NSE symmetries appear in discrete and continuous forms as we will see in this thesis.

1.5.1 Continuous Symmetry and Phase Aligning

The method of slices [10, 11] will prove to be useful due to the translational invariance nature of the data as will be seen in further chapters. This method consists in taking the Discrete Fourier Transform

$$u(x, \tau) = \sum_{k=-N}^{N} \hat{u}_k(\tau) e^{ikx}, \tag{1.11}$$

and phase shifting by using the phase $\phi =$atan2(Im$(\hat{u}_1)$/Re$(\hat{u}_1)$) where $\hat{u}_1$ corresponds to the first Fourier mode. By shifting the flow fields before training the network, data is translated to a unique location. This is desired because the network does not have to account for trajectories with the same structure in different locations. Another way to do this is to consider a slice template instead of the first Fourier mode. This has been done for pipe flow with great success and more details are given in Section 5.2.

1.5.2 Factoring Out Discrete Symmetries

In the context of the systems we study in this thesis, discrete symmetries can appear in the form of rotations and reflections. Similar to the continuous symmetry, the Fourier modes can be used to factor out the symmetries and map the snapshots to a fundamental space. Budanur & Cvitanovic showed how this can be done for the Kuramoto–Sivashinsky system [7]. In this work they used a polynomial basis from the Fourier coefficients such that the different symmetric versions of the snapshots have the same signs. In Chapter 3 we take a similar approach and use a set of Fourier mode to factor out the symmetries for the NSE.

1.6 Outline of this work

The main goal of this thesis is to develop deep learning architectures to learn efficient data-driven reduced-order models that can faithfully capture the most important features of flows in a low-dimensional representation. We also present a framework capable of giving robust estimates of the minimal dimensions needed to represent complex systems. Our benchmark throughout this thesis is two-dimensional Kolmogorov flow. In the range considered this system is high-dimensional, chaotic, and intermittent which makes it challenging to model.

In the remainder of this thesis we present our work towards modeling these complex systems. In Chapter 2, we present a data-driven framework for minimal-dimensional models that effectively capture the dynamics and properties of the flow. We are able to learn low dimensional models that capture short-time tracking, long-time statistics, and even extreme events with great success. In Chapter 3, we factor out the continuous and discrete symmetries of the system, and build models with improved performance where less data is needed to train the models and equivariance is satisfied. We also use a variation of an autoencoder called implicit rank minimizing autoencoder (IRMAE) that gives estimates of the minimal dimension needed to represent the system. In Chapter 4 the focus is to improve IRMAE by introducing an extension to this framework. We show that this modification improves robustness of dimension estimates.

We conclude this thesis in Chapter 5 where we discuss preliminary results on pipe flow using our presented frameworks and give insights on using different machine learning methods to improve dimension estimates and for addressing symmetries.

2

Data-driven low-dimensional dynamic model of Kolmogorov flow [1]

Reduced order models (ROMs) that capture flow dynamics are of interest for decreasing computational costs for simulation as well as for model-based control approaches. This work presents a data-driven framework for minimal-dimensional models that effectively capture the dynamics and properties of the flow. We apply this to Kolmogorov flow in a regime consisting of chaotic and intermittent behavior, which is common in many flows processes and is challenging to model. The trajectory of the flow travels near relative periodic orbits (RPOs), interspersed with sporadic bursting events corresponding to excursions between the regions containing the RPOs. The first step in development of the models is use of an undercomplete autoencoder to map from the full state data down to a latent space of dramatically lower dimension. Then models of the discrete-time evolution of the dynamics in the latent space are developed. By analyzing the model performance as a function of latent space dimension we can estimate the minimum number of dimensions required to capture the system dynamics. To further reduce the dimension of the dynamical model, we factor out a phase variable in the direction of translational invariance for the flow, leading to separate

[1]The text of this chapter is adapted from the publication by C. E. Pérez De Jesús and M. D. Graham *Physical Review Fluids*, 8, 2023

evolution equations for the pattern and phase dynamics. At a model dimension of five for the pattern dynamics, as opposed to the full state dimension of 1024 (i.e. a 32×32 grid), accurate predictions are found for individual trajectories out to about two Lyapunov times, as well as for long-time statistics. Further small improvements in the results occur as dimension is increased to nine, beyond which the statistics of the model and true system are in very good agreement. The nearly heteroclinic connections between the different RPOs, including the quiescent and bursting time scales, are well captured. We also capture key features of the phase dynamics. Finally, we use the low-dimensional representation to predict future bursting events, finding good success.

2.1 Introduction

Development of reduced order dynamical models for complex flows is an issue of long-standing interest, with applications in improved understanding, as well as control, of flow phenomena. The classical approach for dimension reduction of these systems consists of extracting dominant modes from data via principal component analysis (PCA), also known as proper orthogonal decomposition (POD) and Karhunen-Loéve decomposition [27]. PCA determines a set of basis vectors ordered by their contribution to the total variance (fluctuating kinetic energy) of the flow. Given N_s data vectors (“snapshots") $x_i \in \mathbb{R}^N$, one can obtain these basis vectors by performing singular value decomposition (SVD) on the data matrix $X = [x_1, x_2, \cdots] \in \mathbb{R}^{N \times N_s}$ such that $X = U\Sigma V^T$. Projecting the data onto the first d_h basis vectors (columns of U) then gives a low-dimensional representation – a projection onto a linear subspace of the full state space. To find a reduced order model (ROM), a Galerkin approximation of the Navier-Stokes Equations (NSE) using this basis can be implemented; these have shown some success in capturing the dynamics of coherent structures [3, 50]. Previous research has also used POD as well as a filtered version thereof [58], which are linear reduction techniques, to reduce dimensions and learn a time evolution map from data with the use of neural networks (NNs) [43].

Although PCA provides the best linear representation of a data set in d_h dimensions, in general the long-time dynamics of a general nonlinear dynamical systems are not expected to lie on a linear subspace of the state space. For a primer and more details on data-driven dimension reduction methods for dynamical systems refer to Linot & Graham [38]. For dissipative systems, such as the NSE, it is expected that the long-time dynamics will lie on an invariant manifold $\mathcal{M}$, which can be represented *locally* with Cartesian coordinates, but may have a complex global topology [28]. In fluid mechanics, this manifold is often called an *inertial manifold* [24, 62, 71]. Figure 2.1 schematically illustrates a simple example of this idea. Consider a dynamical system $\dot{x} = F(x)$ for state variable $x \in \mathbb{R}^N$. As time proceeds, general initial conditions in this space evolve toward an invariant manifold $\mathcal{M}$ of dimension $d_{\mathcal{M}}$, which in this example can be described by the equation $q = \Phi(p)$ where $x = p + q$, $p \in \mathbb{R}^{d_{\mathcal{M}}}, q \in \mathbb{R}^{N-d_{\mathcal{M}}}$. Furthermore, if we write the dynamics in terms of p and q as $\dot{p} = f(p, q), \dot{q} = g(p, q)$, then trajectories on $\mathcal{M}$ evolve according to $\dot{p} = f(p, \Phi(p))$: i.e. the long time dynamics are given by a set of ordinary differential equations in $d_{\mathcal{M}}$ dimensions, rather than the N dimensions of the original system. More generally, since $\mathcal{M}$ is invariant under the dynamics, the vector field on $\mathcal{M}$ is always tangent to $\mathcal{M}$, and the dynamics on $\mathcal{M}$ are determined by this vector field. In the present work we do not require that the manifold be represented in this simple form, but rather a more general form $G(x) = 0$. In this example, $G(x) = q - \Phi(p)$.

In general one can think of breaking up $\mathcal{M}$ into overlapping regions that cover the domain, to find a local representation. These are called charts and are equipped with a coordinate domain and a coordinate map [36]. The strong Whitney's embedding theorem states that any smooth manifold of dimension $d_{\mathcal{M}}$ can be embedded into a Euclidean space of so-called *embedding* dimension $2d_{\mathcal{M}}$ [36, 65]. This means that in the worst case we can expect in principle to be able to find a $2d_{\mathcal{M}}$-dimensional Euclidean space in which the dynamics lie. To find a $d_{\mathcal{M}}$-dimensional Euclidean space one would in general need to develop overlapping

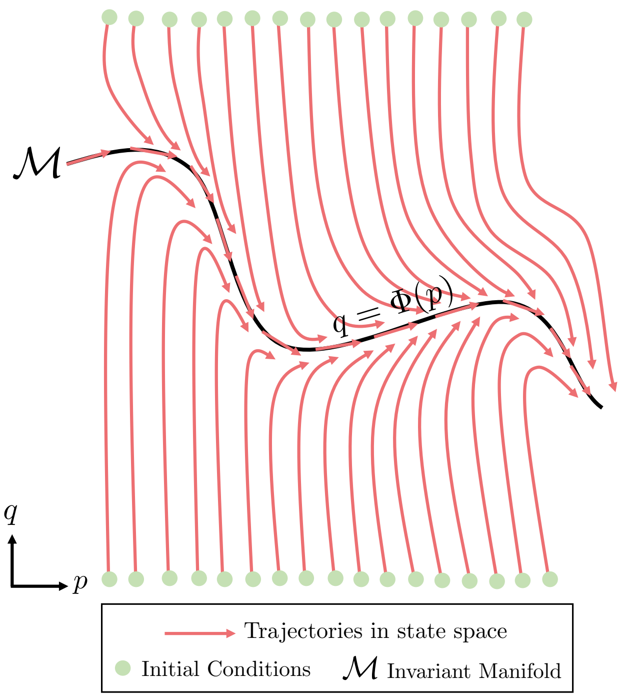


Figure 2.1: Schematic of state space with initial conditions collapsing onto an invariant manifold where the long time dynamics occur.

local representations and evolution equations – this avenue is not pursued in the present work but has been done elsewhere [23]. In this work we aim to find a high-fidelity low-dimensional dynamical model using data from simulations of two-dimensional Kolmogorov flow. In this work, the governing Navier-Stokes Equations will only be used to generate the data – the models will only use this data, not the equations that generated it. Neural networks (NNs) will be used to map between the full state space and the manifold, as well as for the dynamical system model on the manifold.

A number of previous studies have focused on finding data-driven models for fluid flow problems with the use of NNs. Srinivasan *et al.* [59] developed NN models to attempt to predict the time evolution of the Moehlis-Faisst-Eckhardt (MFE) model [46], which is a nine-dimensional model for turbulent shear flows. They used two approaches to finding discrete-time dynamical systems. The first is to simply use a neural network as a discrete-time map, yielding a Markovian representation of the time evolution. The second is to use a long short-term memory (LSTM) network, which yields a non-Markovian evolution equation. Despite the fact that the dynamics are in fact Markovian, the LSTM approach worked better, yielding reasonable agreement with the Reynolds stress profiles. Page *et al.* used deep convolutional autoencoders (CAEs) to learn low-dimensional representations for two-dimensional (in physical space) Kolmogorov flow, showing that these networks retain a wide spectrum of lengthscales and capture meaningful patterns related to the embedded invariant solutions [52]. They considered the case where bursting dynamics is obtained at a Reynolds number of $\mathrm{Re} = 40$ and $n = 4$ wavelengths in the periodic domain. Nakamura *et al.* used CAEs for dimension reduction combined with LSTMs and applied it to minimal turbulent channel flow for $\mathrm{Re}_\tau = 110$ where they showed to capture velocity and Reynolds stress statistics [47]. They studied various degrees of dimension reduction, showing good performance in terms of capturing the statistics; however for drastic dimension reduction they showed how only large vortical structures were captured. Hence, the selection of the minimal dimension to accurately represent the state becomes a challenging task. Reservoir networks have also shown great potential in learning nonlinear models for time evolution. For example, Doan *et al.* trained what they call an Auto-Encoded Reservoir-Computing (AE-RC) framework where the latent space is fed into an Echo State Network (ESN) to model evolution in discrete time [20]. By considering the two-dimensional Kolmogorov flow for $\mathrm{Re} = 30$ and $n = 4$ good performance was obtained when comparing the kinetic energy and dissipation evolution in time. They also showed how the model captures the velocity statistics. However, the nature of the reservoir in the ESN stores past history, making the model non-Markovian.

Although previous research has found data-driven ROMs for fluid flow problems, the focus on these has not been to find the minimal dimension required to capture the data manifold and dynamics. Linot & Graham have addressed this issue for the Kuramoto-Sivashinsky equation (KSE) [37, 38]. They showed that the mean squared error (MSE) of the reconstruction of the snapshots using an AE for the domain size of $L = 22$ exhibited an orders-of-magnitude drop when the dimension of the inertial manifold is reached. Furthermore, modeling the dynamics with a dense NN at this dimension either with a discrete time map [37] or a system of ordinary differential equations (ODE) [38] yields excellent trajectory predictions and long-time statistics. Increasing domain size to $L = 44$ and $L = 66$, which makes the system more chaotic, affects the drops of MSE significantly. However a drop is still seen, and when obtaining the dynamics and calculating long time statistics, good agreement with the true data is obtained. This work, denoted “Data-driven manifold dynamics" (DManD) has been extended to incorporate reinforcement learning control for reduction of dissipation in the KSE, yielding a very effective control policy [73].

We aim to extend this approach to the NSE, specifically to the two-dimensional Kolmogorov flow, where an external forcing drives the dynamics. As Re increases, the trivial state becomes unstable, giving rise to periodic orbits (POs), relative periodic orbits (RPOs) and eventually chaos. Relative periodic orbits correspond to periodic orbits in in a moving reference frame, such that in a fixed frame, the pattern at time $t + T$ is a phase-shifted replica of the pattern at time t. The nature of the weakly turbulent dynamics at a Reynolds number of Re = 14.4, and connections with RPO solutions are the focus of this study. Due to the symmetries of the system the chaotic dynamics travels between unstable RPOs [18] through bursting events [2] that shadow heteroclinic orbits connecting the RPOs. A past study [1] shows that low-dimensional representations can be found with PCA for two-dimensional Kolmogorov flow where in the case of weakly turbulent data, the first two PCA basis in the streamfunction formulation capture most of the energetic content when filtering out the bursting events before the analysis, and including a third basis function captures the bursting information. This point hints at the low-dimensional nature of this system, where a low number of PCA basis functions can energetically represent the data. However, even though the energy can be contained in a low number of basis functions, this does not imply that these will properly capture the dynamics [56]. In [1], development of a model of time-evolution was not considered.

Returning to the aims of the present work, our focus is twofold. We aim to learn a minimal-dimensional high fidelity data-driven model for the long-time dynamics of two-dimensional Kolmogorov flow with the use of an autoencoder (AE), and a discrete-time map, in the form of a dense NN, of the dynamics on the invariant manifold. In this map, the future time prediction only depends on the present state (on the manifold), in keeping with the Markovian nature of the dynamics on the manifold. This approach contrasts with models that use an RNN such as an LSTM, which carry a memory of past states so are not Markovian. It is important to note, however, that the dimension of the invariant manifold is not known a priori, and if we map the data onto a manifold of too low a dimension, then the dynamics on that manifold will not be Markovian. Accordingly, in this work we will carefully assess the performance of our Markovian models as a function of manifold dimension. For our results, the model predictions will be evaluated as a function of dimension, considering short-time trajectories, long-time statistics, quiescent and bursting time distributions, and predictions of bursting events. This paper is structured as follows: in Section 2.2 we present the governing equations together with the symmetries of the system. We also present the dynamics at the two values of Re considered and the connections of the RPOs with the chaotic regime. In Section 2.3 we show the methodology for data-driven dimension reduction and dynamic modeling, which includes the AE architecture and the time map NN. Section 2.4 shows the results, and concluding remarks are given in Section 2.5.

2.2 Kolmogorov flow formulation and dynamics

The two-dimensional Navier-Stokes equations (NSE) with Kolmogorov forcing are

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p = \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \sin(ny)\hat{\boldsymbol{x}} \tag{2.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.2}$$

where $\boldsymbol{u} = [u, v]$ is the velocity vector, p is the pressure, n is the wavenumber of the forcing, and $\hat{\boldsymbol{x}}$ is the unit vector in the x direction. Here $\mathrm{Re} = \frac{\sqrt{\chi}}{v}\left(\frac{L_y}{2\pi}\right)^{3/2}$ where χ is the dimensional forcing amplitude, ν is the kinematic viscosity, and L_y is the size of the domain in the y direction. We consider the periodic domain $[0, 2\pi/\alpha] \times [0, 2\pi]$ with $\alpha = 1$. Vorticity is defined as $\omega = \nabla \times \boldsymbol{u}$. The equations are invariant under several symmetry operations [13], namely a shift (in y)-reflect (in x), a rotation through π, and a continuous translation in x:

$$\mathscr{S} : [u, v, \omega](x, y) \to [-u, v, -\omega]\left(-x, y + \frac{\pi}{n}\right), \tag{2.3}$$

$$\mathscr{R} : [u, v, \omega](x, y) \to [-u, -v, \omega](-x, -y), \tag{2.4}$$

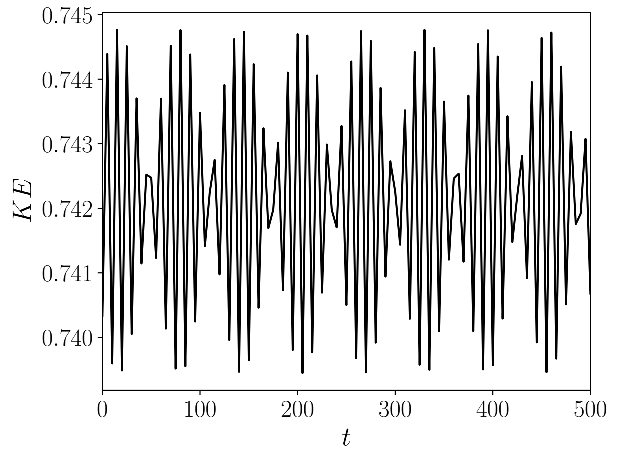
$$\mathscr{T}_l : [u, v, \omega](x, y) \to [u, v, \omega](x + l, y) \quad \text{for } 0 \leqslant l < \frac{2\pi}{\alpha}. \tag{2.5}$$

The total kinetic energy for this system (KE), dissipation rate (D) and power input (I) are

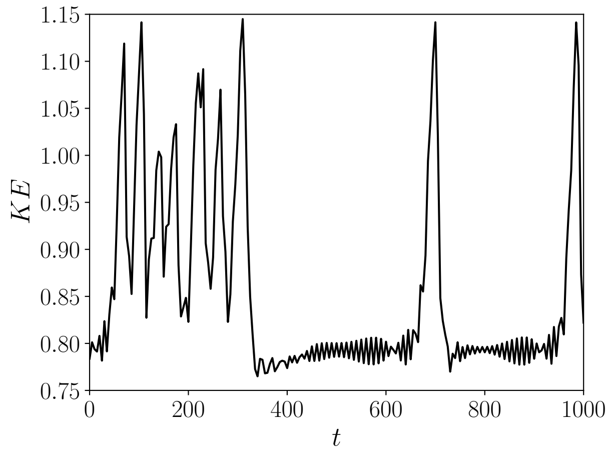
$$KE = \frac{1}{2}\left\langle \boldsymbol{u}^2 \right\rangle_V, D = \frac{1}{\mathrm{Re}}\left\langle |\nabla \boldsymbol{u}|^2 \right\rangle_V, \quad I = \langle u \sin(ny) \rangle_V \tag{2.6}$$

where subscript V corresponds to the average taken over the domain. For the case of $n = 1$ the trivial solution is linearly stable at all Re [30]. It is not until $n = 2$ that the laminar state becomes unstable, with a critical value of $\mathrm{Re}_c = n^{3/2}2^{1/4}$[26, 44, 63].

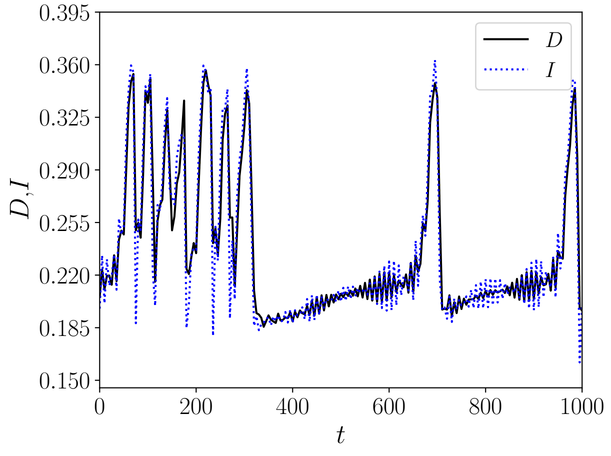
The NSE are evolved numerically in time in the vorticity representation on a $[d_x \times d_y] = [32 \times 32]$ grid following the pseudo-spectral scheme given by Chandler & Kerswell [13], which is based on the code by Bartello & Warn [4]. We show here time series results for the two



(a)



(b)



(c)

Figure 2.2: (a) Time evolution of KE at $\mathrm{Re} = 13.5$. (b) Time evolution of KE at $\mathrm{Re} = 14.4$. (c) Time evolution of D and I at $\mathrm{Re} = 14.4$.

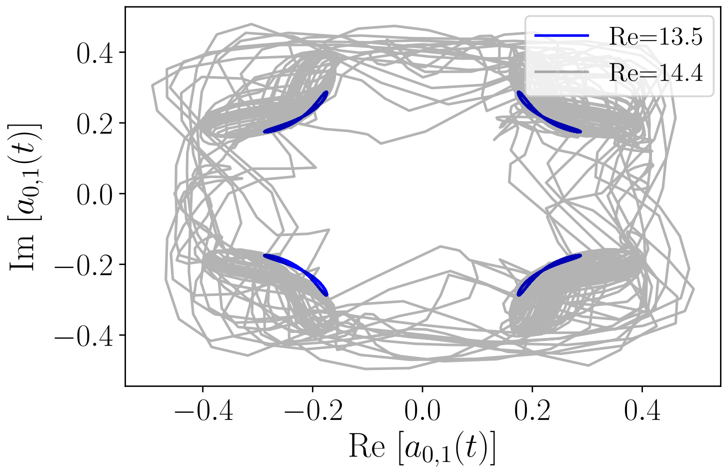


Figure 2.3: Evolution of the real and imaginary components corresponding to the $a_{0,1}(t)$ Fourier mode for $\mathrm{Re} = 13.5$ and $\mathrm{Re} = 14.4$.

dynamical regimes considered in this work, an RPO regime at $\mathrm{Re} = 13.5$ and a chaotic regime at $\mathrm{Re} = 14.4$. Figure 2.2a shows the KE evolution for an RPO obtained at $\mathrm{Re} = 13.5$. Due to the discrete symmetries of the system, there are several RPOs [2], as we further discuss below. Figure 2.2b shows the KE evolution for a trajectory at $\mathrm{Re} = 14.4$. The dynamics are characterized by quiescent intervals where the trajectories are close to RPOs (which are now unstable), punctuated by heteroclinic-like excursions between the RPOs, which are indicated by the intermittent increases of the KE. The RPOs are all related by the symmetries $\mathscr{S}$ and $\mathscr{R}$ [2, 49, 55]. This behavior can also be seen in Figure 2.2c, where the black curve corresponds to the time evolution of D and the blue curve to the time evolution of I. Figure 2.3, shows a state-space projection of a trajectory onto the plane $\mathrm{Re}\,[a_{0,1}(t)] - \mathrm{Im}\,[a_{0,1}(t)]$ where $a(k_x, k_y, t) = a_{k_x,k_y}(t) = \mathcal{F}\{\omega(x, y, t)\}$ is the discrete Fourier transform in x and y. The grey curve corresponds to $\mathrm{Re} = 14.4$ and the different blue curves show four different RPOs related by the shift-reflect symmetry $\mathscr{S}$ at $\mathrm{Re} = 13.5$.

2.3 Data-driven dimension reduction and dynamic modeling

2.3.1 Dimension reduction with autoencoders

To learn a minimal-dimensional model for the two-dimensional Kolmogorov flow we first have to find a low-dimensional nonlinear mapping from the full state to the reduced representation. For this purpose we consider a common machine learning architecture known as an undercomplete autoencoder (AE), whose purpose is to learn a reduced representation of the state such that the reconstruction error with respect to the true data is minimized. The AE consists of an encoder, $\mathcal{E}(\cdot)$, that maps from the full space $\mathbb{R}^N$ to the lower dimensional latent space $h(t) \in \mathbb{R}^{d_h}$ (i.e., coordinates on the manifold $\mathcal{M}$), and a decoder, $\mathcal{D}(\cdot)$, that maps back to the full space. Flattened versions of $\omega(x, y, t)$ are used, which we refer from this point on as $\omega(t)$, so $N = 32 \times 32 = 1024$. We shall see that the latent space dimension d_h will be much smaller than the dimension N of the full spatially-resolved state. The encoder $\mathcal{E}(\omega(t))$ is a coordinate mapping from $\mathbb{R}^N$ to $\mathcal{M}$, and the decoder $\mathcal{D}(h(t))$ is the mapping back from $\mathcal{M}$ to $\mathbb{R}^N$.

We train the AEs with $\omega(t)$ obtained from the evolution of NSE for the original data as well as accounting for the discrete and continuous symmetries. By accounting for the symmetries it is expected that the networks will perform better, by not having to learn the symmetries in the latent space mapping. We account for the continuous symmetry in x, $\mathscr{T}_l$, with the method of slices [10, 11]. The $k_x = 1, k_y = 0$ Fourier mode is used to find the spatial phase: $\phi_x(t) = \operatorname{atan2}\left\{\operatorname{Im}\left[a_{1,0}(t)\right], \operatorname{Re}\left[a_{1,0}(t)\right]\right\}$. This can then be used to phase-align the vorticity snapshots such that this mode is a pure cosine: $\hat{\omega}(x, y, t) = \mathcal{F}^{-1}\left\{\mathcal{F}\{\omega(x, y, t)\}e^{-ik\phi_x(t)}\right\}$. Doing this ensures that the snapshots lie in a reference frame were no translation happens in the x direction. We will learn evolution equations for both $\hat{\omega}(t)$ and $\phi_x(t)$, which we will denote as the pattern dynamics and phase dynamics, respectively. We also consider the shift-reflect (SR) symmetry, $\mathscr{S}$, as well as the rotation through π, $\mathscr{R}$. To account for the SR symmetry the goal is to collapse the phase-aligned snapshots to the same common state. We can define two indicator functions such that the SR subspace is specified. The first one, $I_{Even} = \text{sgn}(\phi_y)$, where $\phi_y(t) = \text{atan}\,2\left\{\text{Im}\left[a_{0,1}(t)\right], \text{Re}\left[a_{0,1}(t)\right]\right\}$ is the spatial phase in y. The second indicator function is $I_{odd} = \text{sgn}(\text{Re}[a_{2,0}(t)])$, the sign of the real part of the second Fourier mode in x. We can then map the vorticity snapshots in such a way that $I_{Even}, I_{Odd} > 0$ by applying SR operations to the state. The rotation symmetry is accounted for, on top of the SR symmetry, by minimizing the l^2-norm of the data with respect to a template snapshot. This is done by applying the discrete operation that rotates and shift-reflects the vorticity snapshots and selecting the snapshot that minimizes the norm. We note that we take a different approach for reducing the symmetries compared to previous research on symmetry-aware AEs [34].

Previous work [37] has shown that training a NN to learn the difference between the data and the projection onto the leading PCA basis vectors improved reconstruction performance compared to learning a latent space directly from the full data. To present the framework, we will use the phase-aligned and flattened vorticity $\hat{\omega}(t)$, since that is what we use for the time-evolution. Below, however, we will present some results where other versions of the data are used – e.g. the data with phase-shifting. The autoencoder aspect of the analysis is identical.

We begin the process by computing the projection of the data onto the first d_h basis vectors, $P_{d_h}U^T\hat{\omega}(t)$. We then seek to learn a d_h-dimensional correction to that projection, $E\left(U^T\hat{\omega}(t)\right)$ – the sum of these is the latent-space representation $h(t)$. In other words, the encoding step learns the deviation from PCA

$$E\left(U^T\hat{\omega}(t)\right) = h(t) - P_{d_h}U^T\hat{\omega}(t). \tag{2.7}$$

We emphasize that this step *is not* simply a projection onto a linear subspace defined by d_h PCA modes– rather it is an approach that learns the deviation of the data from that projection. Similarly the decoding section learns the difference

$$D(h(t)) = U^T\tilde{\hat{\omega}}(t) - \begin{bmatrix} h(t) \\ 0 \end{bmatrix}, \tag{2.8}$$

where $\tilde{\hat{\omega}}(t)$ corresponds to the reconstruction of $\hat{\omega}(t)$. Inserting Equation 2.7 into Equation 2.8 and noting that by definition $\tilde{\hat{\omega}}(t) = U[P_{d_h}U^T\hat{\omega}(t), P_{d-d_h}U^T\hat{\omega}(t)]^T$ we get that the exact solution satisfies $E\left(U^T\hat{\omega}(t)\right) + D_{d_h}((h(t)) = 0$. To satisfy this constraint we add it to the loss function as a penalty to obtain

$$L = \|\hat{\omega}(t) - \tilde{\hat{\omega}}(t)\|^2 + \alpha_L \left\|E(U^T\hat{\omega}(t)) + D_{d_h}(h(t))\right\|^2 \tag{2.9}$$

where $\|\cdot\|$ is the l^2-norm and we select $\alpha_L = 1$. We can now train the AEs by minimizing L via stochastic gradient descent. We train 4 AEs at each of several values of d_h to study the MSE of the reconstruction of $\hat{\omega}(t)$. All models were trained for 300 epochs with an Adam optimizer using Keras. After 300 epochs no further improvement over the test data was observed; see Figure 2.4. The training data consists of long time series from the direct simulations, with initial transients removed. We use a total of 10^5 snapshots separated by $\tau = 5$ time units for Re = 14.4, and 10^4 snapshots separated by $\tau = 5$ for Re = 13.5. We do an 80%/20% split for training and testing respectively. Figure 2.5a shows a summary of the AE and Table 2.1 gives information on the layer dimensions, and activations used in each layer of the encoder and decoder. At each value of d_h, the model with the smallest MSE over a test data set from the phase-aligned data is then selected for the discrete time map. We will show in Section 2.4.1 that factoring out the phase dramatically increases AE performance.

2.3.2 Time evolution via a dense NN

After finding $h(t)$ from the AEs, we seek a discrete-time map

$$h(t+\tau) = F(h(t)) \tag{2.10}$$

that evolves $h(t)$ from time t to $t+\tau$. We fix $\tau = 5$. The function F is also expressed as a dense NN. Here we train 5 NNs for the different d_h cases with the following loss

$$L_t = \|\tilde{h}(t+\tau) - h(t+\tau)\|^2, \tag{2.11}$$

where $h(t+\tau)$ comes from true data and $\tilde{h}(t+\tau) = F(h(t))$ from the prediction, and select the one with the best performance. For the discrete time map we trained for 600 epochs with the use of a learning rate scheduler. In this case we noticed an increase in performance when dropping the learning rate hyperparameter by an order of magnitude after 300 epochs. Figure 2.5b shows a summary the framework just described, and Table 2.1 gives information on the layer dimensions and activations used in each layer.

As discussed previously, the time evolution is done in the phase-aligned space. To complete the dynamical picture we seek a discrete-time map for the phase evolution

$$\Delta\tilde{\phi}_x(t+\tau) = G(h(t)), \tag{2.12}$$

where $\Delta\phi_x(t+\tau) = \phi_x(t+\tau) - \phi_x(t)$. Because of translation equivariance, the actual phase is only unique to within a constant. We train 5 NNs for the the different d_h cases with the following loss

$$L_p = \|\Delta\tilde{\phi}_x(t+\tau) - \Delta\phi_x(t+\tau)\|^2, \tag{2.13}$$

such that $\Delta\tilde{\phi}_x(t+\tau) = G(h(t))$. Figure 2.5c shows a summary of the framework we have described, and Table 2.1 gives information on the layer dimensions and activations used in

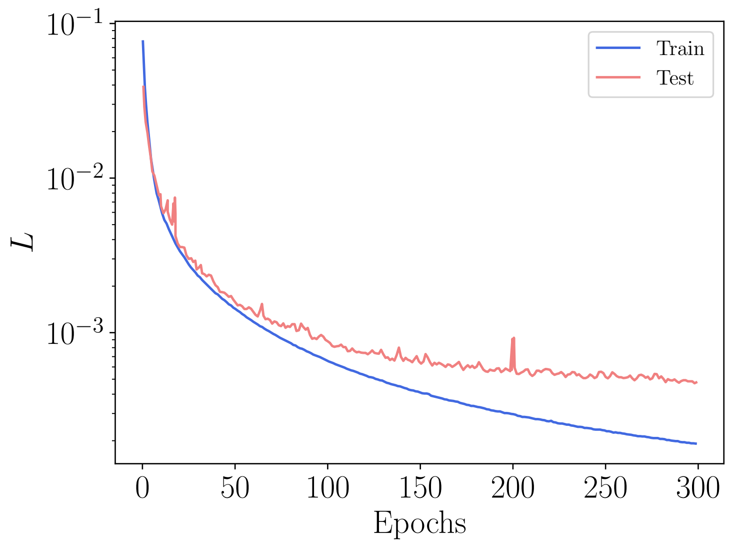


Figure 2.4: Autoencoder loss versus epochs over training and test data sets corresponding to a trial from the case Re $= 14.4$, $d_h = 9$.

each layer.

Table 2.1: Neural network layer dimensions and activations used in each layer. Sigmoid function are denoted 'S'.

	Function	Shape	Activation
Encoder	E	$1024 : 5000 : 1000 : d_h$	S:S:S
Decoder	D	$d_h : 1000 : 5000 : 1024$	S:S:linear
Evolution	F	$d_h : 500 : 500 : d_h$	S:S:linear
Phase Prediction	G	$d_h : 500 : 500 : 500 : 1$	S:S:S:linear

2.4 Results

We present results as follows. First we will show the AE performance for the various d_h and symmetries considered. We then report results for time evolution models, again studying performance as a function of the number of dimensions. Both evolution of the pattern and phase dynamics are considered. We wrap up the results by predicting bursting events based on the low-dimensional representation.

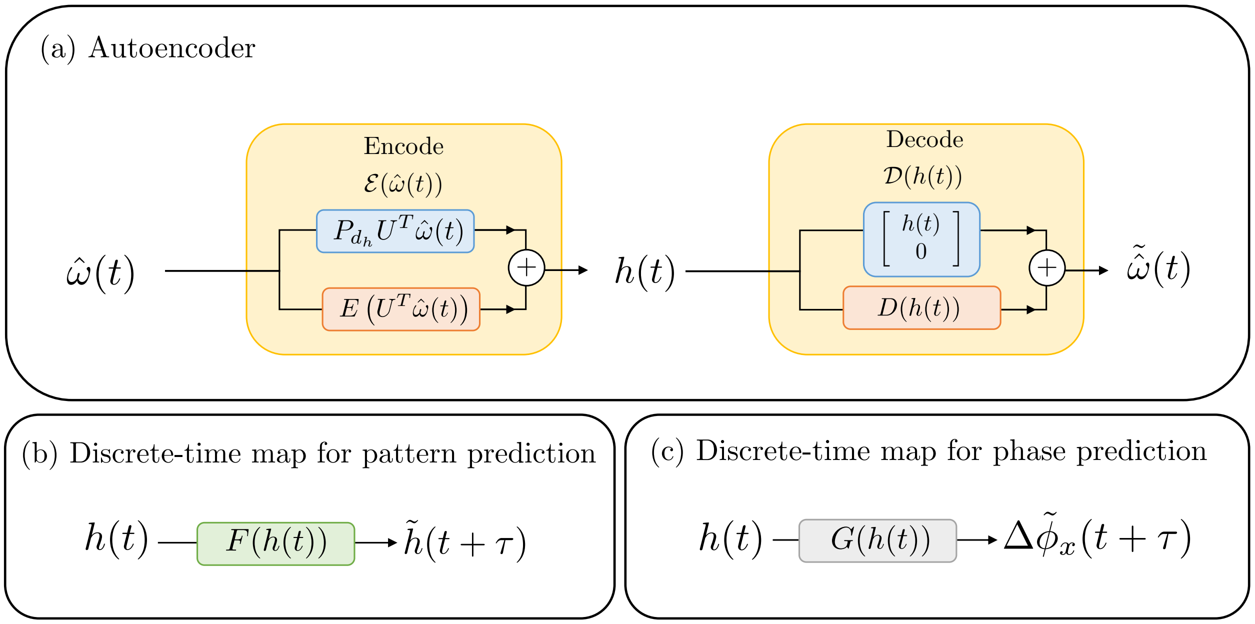
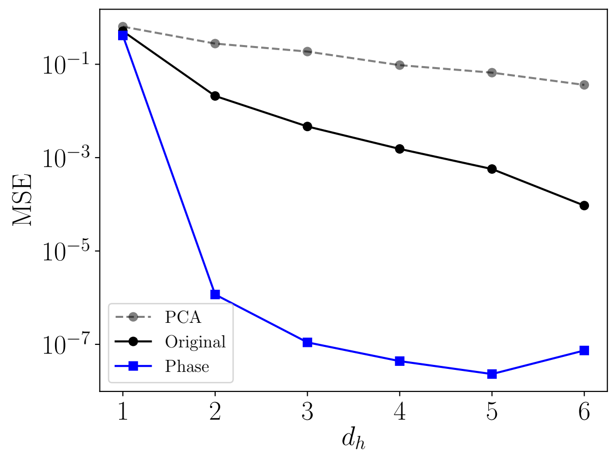

(a) Autoencoder
Encode
$\mathcal{E}(\hat{\omega}(t))$
$P_{d_h}U^T\hat{\omega}(t)$
$E\left(U^T\hat{\omega}(t)\right)$
$\hat{\omega}(t)$
$h(t)$
Decode
$\mathcal{D}(h(t))$
$\begin{bmatrix} h(t) \\ 0 \end{bmatrix}$
$D(h(t))$
$\tilde{\hat{\omega}}(t)$
(b) Discrete-time map for pattern prediction
$h(t)$
$F(h(t))$
$\tilde{h}(t+\tau)$
(c) Discrete-time map for phase prediction
$h(t)$
$G(h(t))$
$\Delta\tilde{\phi}_x(t+\tau)$

Figure 2.5: Neural network frameworks for (a) autoencoder (b) discrete-time map for pattern prediction and (c) discrete-time for phase prediction.

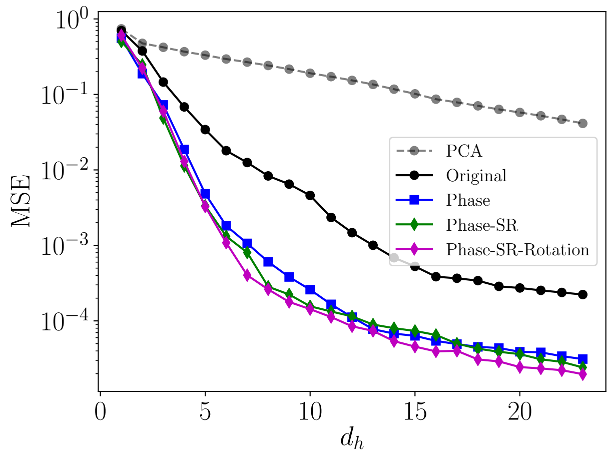
2.4.1 Dimension reduction with autoencoders

We begin by showing results for Re $= 13.5$. In Figure 2.6a we see the MSE versus d_h trend where the grey curve corresponds to the PCA reconstruction for the original data ($\tilde{\omega}(t) = U_{d_h}U_{d_h}^T\omega(t)$), the black curve to the AE with the original data, and the blue curve to the AE with the phase factored out before training. The MSE is calculated over the test data set. Notice that, as expected, the AEs perform better than PCA. This is because of the nonlinearities that are added to the linear optimal latent space found in PCA in combination with the nonlinear decoder. The blue curve exhibits a sharp drop in the MSE at a dimension of $d_h = 2$, which is the correct embedding dimension for a limit cycle. This happens because the phase is accounted for; the dynamics of the system in the phase-aligned reference frame corresponds to a PO and the autoencoder does not have to learn all the possible phases due to the continuous translation in x. The overall embedding dimension is $d_h + 1 = 3$, where 1 corresponds to the phase. Hence we are able to estimate the dimension for this system by looking at the drop in the MSE curve.

We now consider the Re $= 14.4$ case, where the dynamics are chaotic, moving between



(a)



(b)

Figure 2.6: MSE versus dimension d_h over the test data corresponding to (a) Re = 13.5 and (b) Re = 14.4. The PCA curve corresponds to the MSE of the reconstruction for the test data set with respect to the true data $\omega(t)$, with no symmetries factored out, using the truncated U into d_h dimensions such that $\tilde{\omega}(t) = U_{d_h} U_{d_h}^T \omega(t)$; the 'Original', 'Phase', 'Phase-SR', and 'Phase-SR-Rotation' curves correspond to the MSEs of the reconstruction for the test data set with respect to the true data using AEs. In the curve labeled 'Original', no symmetries are factored out and in the other curves the corresponding symmetries in the labels are factored out.

the regions near the now unstable RPOs. In Figure 2.6b we show the same curves as in Figure 2.6a but we also include the green and magenta curves, which in addition factor out the SR and the SR-Rotation symmetries respectively before training the AEs. These are included due to the added complexity of Re = 14.4, where the chaotic trajectory travels in the vicinity of the RPOs related by the symmetry groups previously discussed. A monotonic decrease in MSE can be seen for the different symmetries considered in the blue, green, and magenta curves, but no sharp drop is apparent. Instead we notice that the MSE drops at different rates in different regions. For example, in the blue curve corresponding to the phase aligned data, we see a sharp drop from $d_h = 1 - 6$ followed by a more gradual drop from $d_h = 6 - 13$. In the following sections we couple the dimension-reduction analysis with models for prediction of time evolution for the phase aligned data. We expect that this combination will help us determine how many dimensions are needed to correctly represent the state.

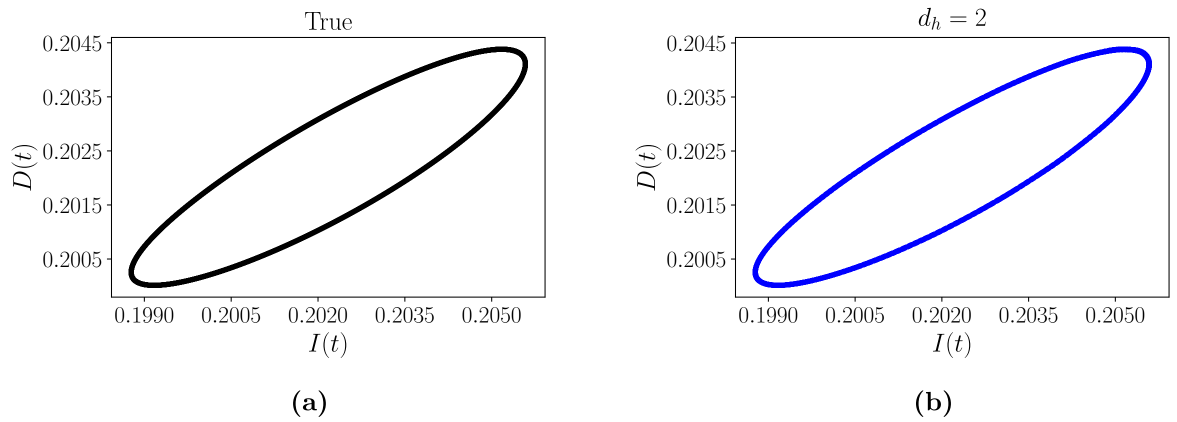


Figure 2.7: Trajectory of $I(t)$ vs $D(t)$ corresponding to $\mathrm{Re} = 13.5$ for (a) true and (b) predicted data corresponding to dimensions $d_h = 2$.

2.4.2 Time evolution as a function of dimension - Short time predictions

The focus of this work is the chaotic dynamics at $\mathrm{Re} = 14.4$. Before considering that case, for completeness we briefly present results for $\mathrm{Re} = 13.5$. In Figure 2.7 we see $D(t)$ versus $I(t)$ for the true and predicted dynamics at $d_h = 2$; they are indistinguishable. At $d_h = 1$, which is not shown, the model fails and the dynamics can not be captured. The reason for this is simple – the embedding dimension for a limit cycle is two.

Now we return to the case of $\mathrm{Re} = 14.4$, focusing first on short-time trajectory predictions. The Lyapunov time t_L for this system is approximately $t_L \approx 20$ [29], hence $t_L \approx 4\tau$. We take initial conditions $h(t) \in \mathbb{R}^{d_h}$ to evolve recurrently with the discrete time map $F(\cdot)$, such that $\tilde{h}(t+\tau) = F(h(t))$, $\tilde{h}(t+2\tau) = F(\tilde{h}(t+\tau))$, $\tilde{h}(t+3\tau) = F(\tilde{h}(t+2\tau))$ and so on. After evolving in time the data is then decoded to get $\tilde{\hat{\omega}}_h(t)$ and compared with $\hat{\omega}(t)$. We consider trajectories with ICs starting in the quiescent as well as in the bursting regions. The nature of the intermittency of the data makes it challenging to assign either bursting or quiescent labels. We consider a window of past and future snapshots and a criterion on $\|\hat{\omega}(t)\|$ to make this decision, using the algorithm described in Algorithm 1.

Doing this we ensure that snapshots that are contained in the bursting events and have a value of $\|\hat{\omega}(t)\|$ similar to quiescent snapshots are correctly classified. We use a threshold on $\|\hat{\omega}(t)\|$ to determine if a check is needed. For the classification strategy any snapshot above a threshold of 60 is classified as bursting with a label of 1, below 60 we enter a loop as shown in Algorithm 1 to determine if it should be classified as bursting or quiescent, where quiescent corresponds to a label of 0. This check is needed to correctly label snapshots that have comparable $\|\hat{\omega}(t)\|$ but are still in the bursting regime. Figure 2.8 shows a short time trajectory where the black line corresponds to $\|\hat{\omega}(t)\|$ and the red to the 0/1 labels. Notice that, as shown in Algorithm 1, some of the data at the beginning and at the end of the time series will not be labeled, there are no past or future snapshots to compare to, and can be removed.

After labeling the data as quiescent or bursting, we then consider the time evolution from ICs of $h(t)$ using the models of various dimensions. We will first show sample trajectories from ICs starting in the two regions, then show the ensemble-averaged prediction error as a function of time. Figure 2.9a shows the KE evolution for an IC starting in the quiescent region. The black curve corresponds to the true data and the colored curves to the different d_h models. At a dimension of $d_h = 3$ the predicted KE diverges quickly with respect to the true KE. In the case of $d_h = 5$ we see that the bursting event is correctly captured, but with a slight lag. However $d_h = 7$ does not capture the bursting in this time frame considered. For $d_h = 9$ the bursting event happens with a significant lag with respect to the true data and $d_h = 11$ captures the event similar to $d_h = 5$. Figure 2.9b shows the KE evolution for an IC starting in the bursting region. The black curve corresponds to the true data and the colored curves to the different d_h models. At a dimension of $d_h = 3$ the KE stays bursting and does not show agreement with the true KE. However $d_h = 5$ shows better agreement and is also capable of closely predicting the end of the bursting event. In the case of $d_h = 7, 9$, and 11 these agree closely with the KE evolution before traveling to the quiescent region.

Algorithm 1 Quiescent/Bursting labeling of vorticity snapshots

```
$W \leftarrow [\hat{\omega}(t_1), \hat{\omega}(t_2) \cdots]$        ▷ Matrix with $N_s$ vorticity snapshots, $W \in \mathbb{R}^{N \times N_s}$
$S$                                                   ▷ Initialize label array $S$
$W_{l2} \leftarrow \|W\|$                  ▷ Calculate $l^2$-norm of snapshots, $W_{l2} \in \mathbb{R}^{N_s}$
$b \leftarrow 10$                          ▷ Number of past snapshots in time to consider
$f \leftarrow 10$                          ▷ Number of future snapshots in time to consider
for $i = b, b+1, \dots N_s - f$ do                                ▷ $i$ is snapshot I.D.
    if $W_{l2}[i] < 60$ then
        $d_p \leftarrow \text{abs}(W_{l2}[i-b:i] - W_{l2}[i])$   ▷ Difference between current and past snapshots
        $b_p \leftarrow \text{sum}(d_p > 5)$          ▷ Sums values that exceed a threshold of 5 (user defined)
        $d_f \leftarrow \text{abs}(W_{l2}[i:i+f] - W_{l2}[i])$ ▷ Difference between current and future snapshots
        $b_f \leftarrow \text{sum}(d_f > 5)$          ▷ Sums values that exceed a threshold of 5 (user defined)
        if $b_p = 0$ or $b_f = 0$ then
            $S[i-b] \leftarrow 0$
        else
            $S[i-b] \leftarrow 1$
        end if
    else
        $S[i-b] \leftarrow 1$
    end if
end for
```

Turning from examples of individual trajectories to ensemble averages, Figure 2.10a shows ensemble averages of the difference between the true and predicted trajectories, separately considering ICs in the bursting and quiescent regions. Solid curves correspond to quiescent ICs and dashed curves to bursting ICs. Starting from $d_h = 3$ we increase up to $d_h = 12$. We selected 10^4 ICs in total where approximately 1/3 of the ICs correspond to bursting. As expected, predictions at $d_h = 3$ diverge quickly from the true dynamics in both quiescent and bursting IC scenarios. With increasing d_h, trajectories track better for both types of ICs. We can also notice that the two darkest curves, corresponding to $d_h = 11, 12$, fall on top of each other in the case of quiescent ICs and the trajectories for the quiescent ICs track almost perfectly for approximately two Lyapunov times for dimensions $d_h = 5$ and higher. In Figure 2.10b we show ensemble averages of the difference between the true and predicted dynamics based on all ICs. The same trend is obtained as discussed for Figure 2.10a with dimensions of $d_h = 9$ and higher in similar agreement, and as expected the errors increase

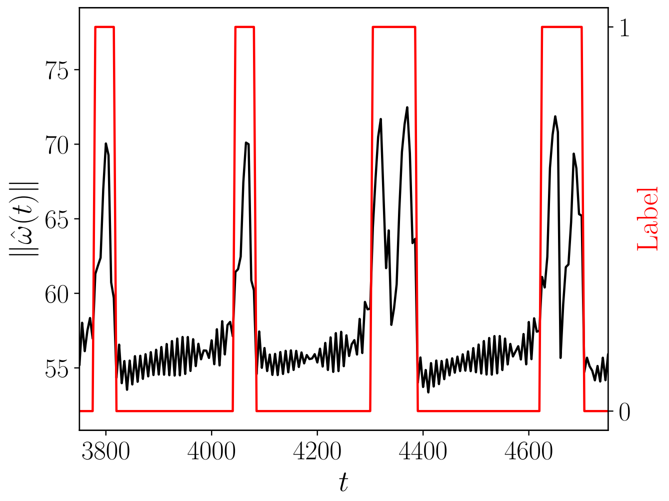


Figure 2.8: Labeling of $\hat{\omega}(t)$ snapshots in a short time series where 1 corresponds to bursting and 0 to quiescent.

for all of the curves due to the divergence of the bursting ICs. We can conclude that models of dimensions $d_h = 5$ and higher are very good at capturing trajectories in the quiescent regions, which happens through the accurate prediction of the oscillatory behavior of the unstable RPO right before a bursting occurs. Prediction from bursting ICs is harder, due to the complex dynamics involved in this region. We also consider , in Figure 2.10c the ensemble averages of the difference between the true and predicted trajectories versus d_h for all ICs with at time instants $t = 0, t_L, 2t_L, 3t_L$. As expected, with increasing t the trajectories deviate from the true data. However we notice that for all of the curves the error decreases with increasing d_h and after $d_h = 9$ little to no improved performance is observed.

2.4.3 Time evolution as a function of dimension - Long time predictions

In this section we present long time statistics for the models and true data at Re $= 14.4$. From ICs on the attractor, we evolve for 2×10^5 time units, yielding to get 4×10^4 snapshots of data. This duration is sufficient to densely sample the quiescent and bursting regions. We note that long time statistics did not change if the IC was in a bursting or quiescent region.

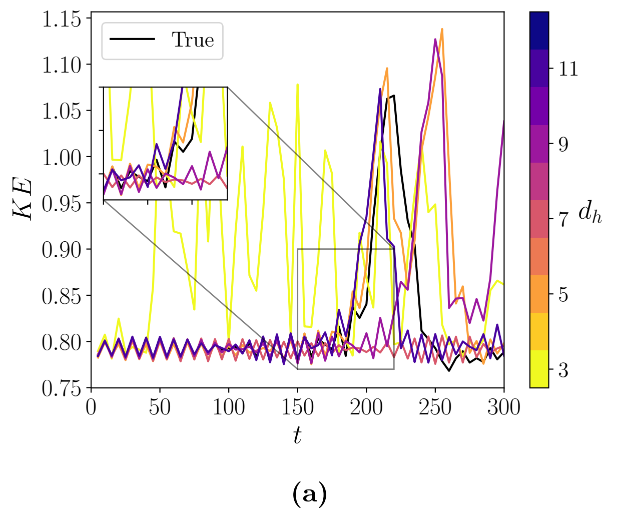
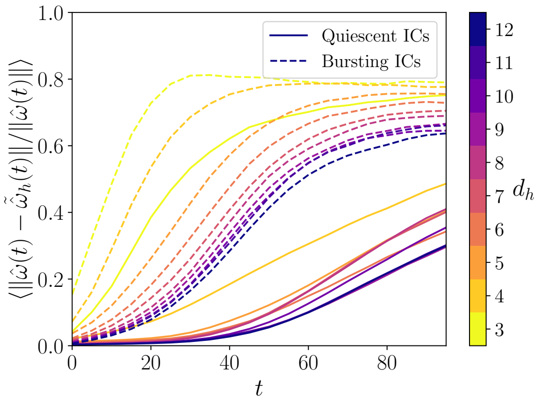


Figure 2.9: Example trajectories of KE at different d_h for (a) a quiescent initial condition and (b) a bursting initial condition, for dimensions $d_h = 3, 5, 7, 9$, and 11.

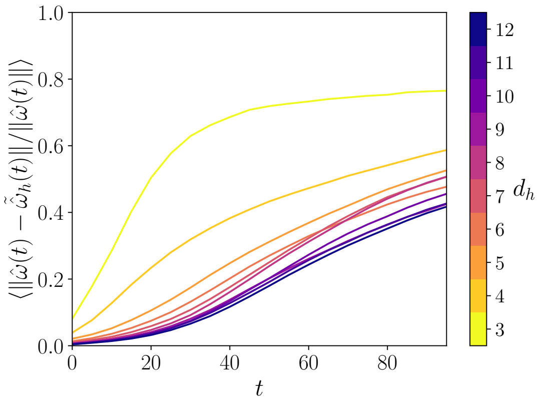
Figure 2.11 shows the joint probability density function (PDF) of I and D for true and predicted data from models with $d_h = 3, 5, 7, 9$, and 11 – note the logarithmic scale, here and below. We notice that at $d_h = 3$ the different areas corresponding to quiescent and bursting regions are populated similarly in terms of the probability intensity compared with the true PDF shown, but the shape of the predicted PDF takes a curved form that is not seen in the true PDF. When we get to $d_h = 5$ the D and I events are captured better, and similarly for increasing dimensions. We also compute the joint PDF of $\mathrm{Re}\,[a_{0,1}]$ and $\mathrm{Im}\,[a_{0,1}]$, shown in Figure 2.12. From this quantity we can observe the heteroclinic-like connections between the unstable RPOs, which correspond to the four ribbon-like regions of high probability. Here we see similar trends as in the joint PDF for I-D: $d_h = 3$ shows poor qualitative reconstruction compared with higher dimensions, and once $d_h \geq 5$, the joint PDFs from the model prediction are virtually indistinguishable from the true PDFs. To further quantify the relationship of the PDFs from the models to the true data, we calculate the Kullback-Leibler (KL) divergence,

$$D_{KL}(\tilde{P}||P) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{P}\{a,b\}\ln\frac{\tilde{P}\{a,b\}}{P\{a,b\}}da\, db, \tag{2.14}$$

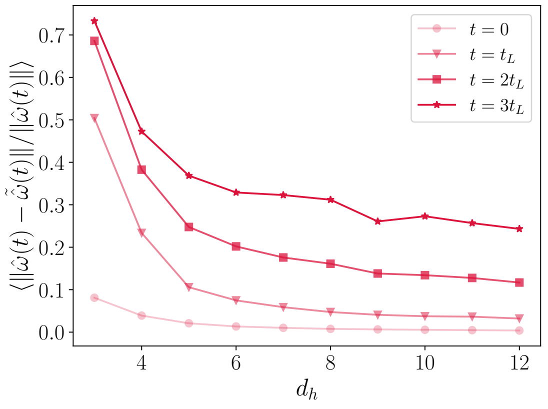
where $\tilde{P}$ corresponds to the predicted PDF and P to the true PDF. Due to the approximation



(a)



(b)

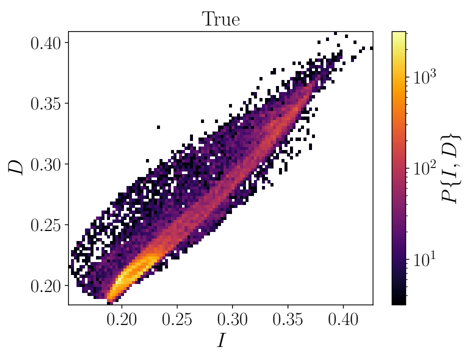


(c)

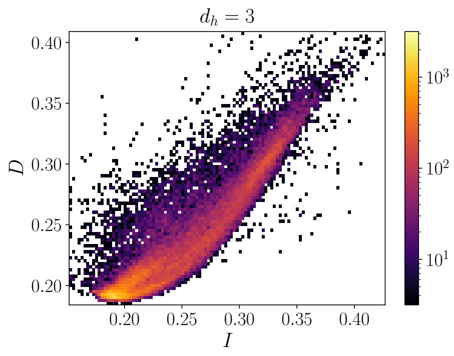
Figure 2.10: Difference between true vorticity evolution and vorticity evolution obtained from the time map F from $h(t)$ where (a) correspond to averages taken over bursting and quiescent ICs and (b) averages over all the data. (c) Difference between true vorticity evolution and vorticity evolution obtained from the time map F from $h(t)$ with varying d_h for increasing t_L. This corresponds to averages over all the data.

of the integral to discrete data we ignore areas where either the true or predicted PDFs are zero. Let us first consider the case $a = I$ and $b = D$. Figure 2.13a shows D_{KL} calculated with varying d_h. The dashed grey line corresponds to D_{KL} calculated over different true data sets. This serves as a baseline for comparison to the predicted PDFs. A significant decrease happens at $d_h = 4$ followed by small decreases at higher dimensions. We see that after $d_h = 5$ no significant information is gained, with errors plateauing at approximately $d_h \geq 7$. We can also look at the case where $a = \text{Re}\,[a_{0,1}]$ and $b = \text{Im}\,[a_{0,1}]$ in Figure 2.13b. We notice that errors of the joint PDF in Figure 2.13b show a similar trend as Figure 2.13a with errors plateauing at approximately $d_h \geq 9$. We can infer from these results that the embedding dimension of this system lies in the range $d_h = 5 - 9$, and furthermore that the data-driven model can reproduce the long-time statistics with very high fidelity.

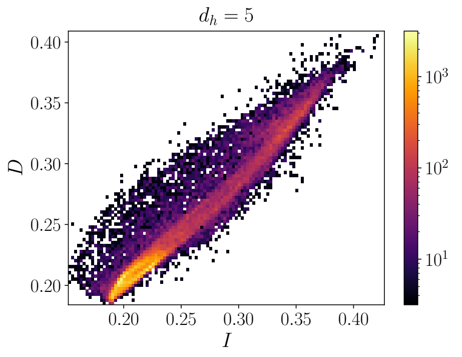
The above PDFs yield no information about the temporal behavior of the system. One temporal feature of significant interest in problems with intermittency is the probability density of the durations of time intervals with different behavior. To address this, we consider the PDFs of time spent in bursting (t_b) and in quiescent (t_q) regions. The labeling method discussed in the previous section is used. For this calculation we take a trajectory of 10^5 snapshots from an arbitrary IC. The PDF for the true data is shown in Figure 2.14a followed by the PDFs that come from the $d_h = 3, 5, 7, 9$, and 11 models in Figures 2.14b - 2.14f. The true data shows that t_q is mostly concentrated between $t \approx 200 - 300$ with a high intensity peak shown at $t = 5$. We attribute this peak to a small fraction of snapshots in the bursting region that get mislabeled as quiescent due to the weakly chaotic nature of the data. We do not expect for this to drastically change our conclusions because the same labeling system is used for the true data and the models. In the case of t_b we notice that these are mostly concentrated between $t \approx 0 - 200$. Looking at both the PDFs and averages of the times we see that $d_h = 3$ fails to correctly capture the shape of the PDF and also underpredicts $\langle t_q \rangle$ and $\langle t_b \rangle$. At $d_h = 5$ we start getting better agreement where we see that the PDFs clearly show the two regions where t_b and t_q are concentrated. In the case of $d_h = 7$ we can see



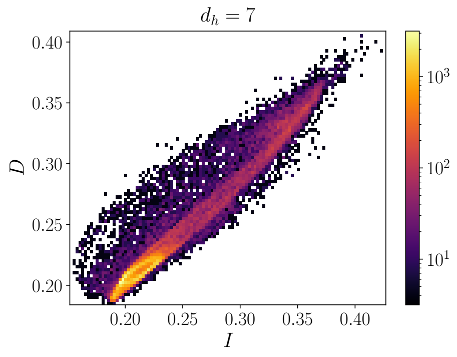
(a)



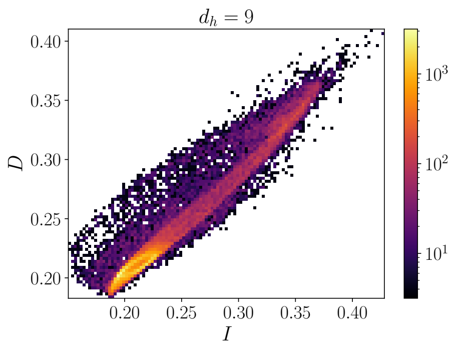
(b)



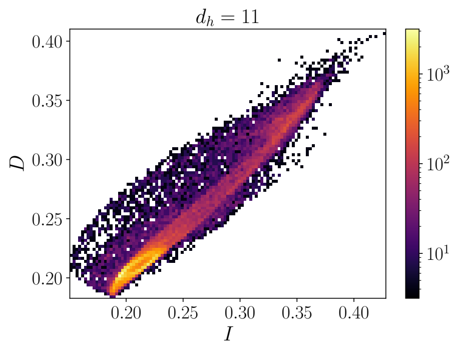
(c)



(d)

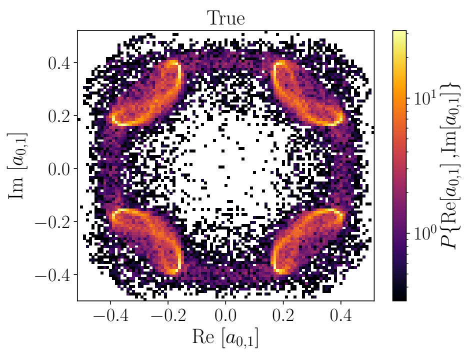


(e)

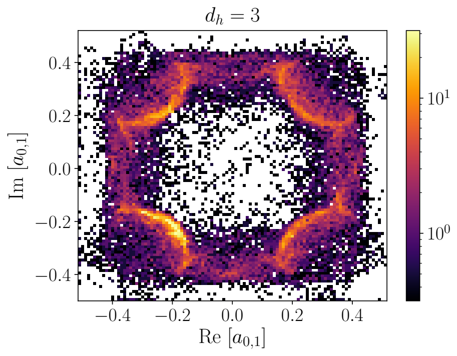


(f)

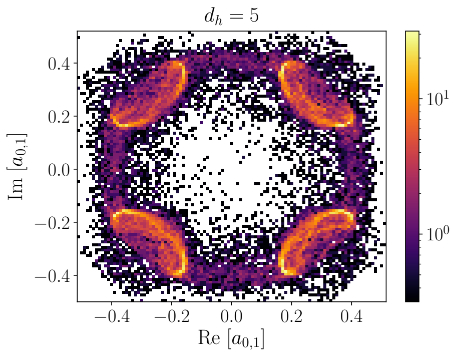
Figure 2.11: Re = 14.4: Joint PDFs of I-D corresponding to Re = 14.4 for (a) true and (b)-(f) predicted data corresponding to dimensions $d_h = 3, 5, 7, 9$, and 11.



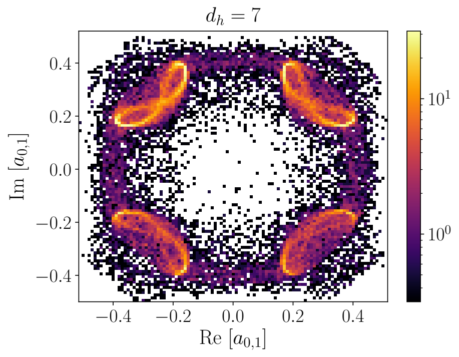
(a)



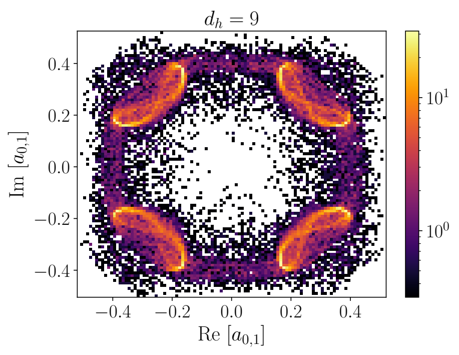
(b)



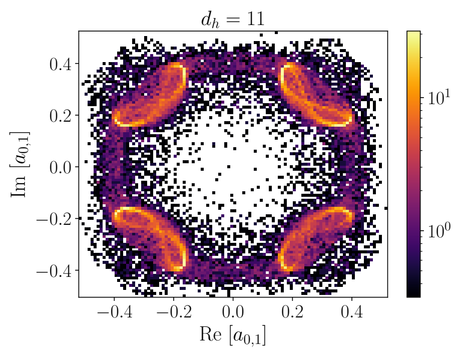
(c)



(d)



(e)



(f)

Figure 2.12: Re $= 14.4$: Joint PDFs of Re $[a_{0,1}(t)]$ − Im $[a_{0,1}(t)]$ corresponding to Re $= 14.4$ for (a) true and (b)-(f) predicted data corresponding to dimensions $d_h = 3, 5, 7, 9$, and 11.

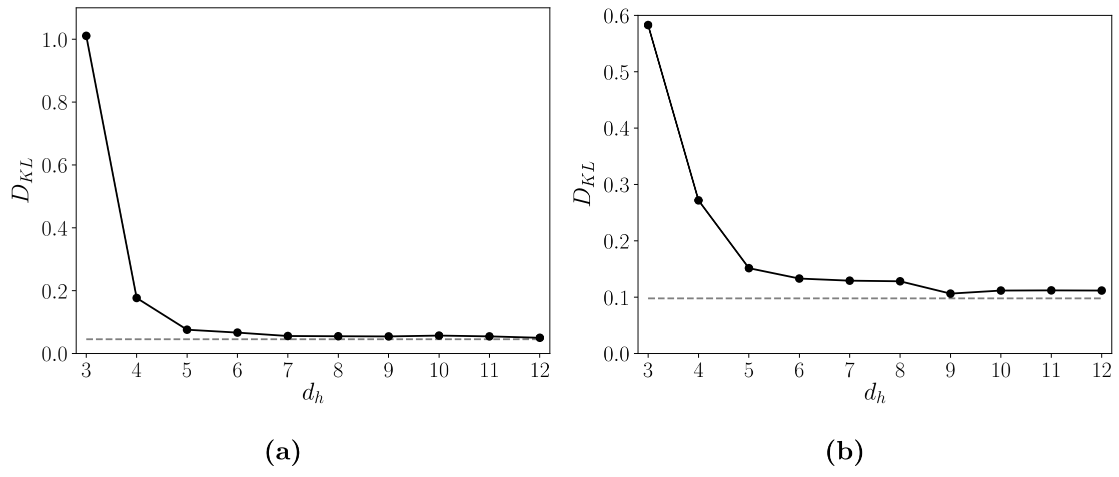
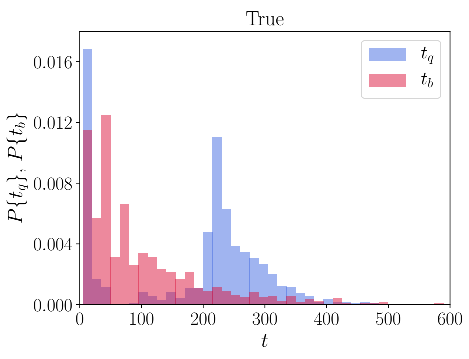


Figure 2.13: Re $= 14.4$: D_{KL} vs dimension d_h for (a) I-D and (b) $\mathrm{Re}\,[a_{0,1}] - \mathrm{Im}\,[a_{0,1}]$ predicted vs true joint PDFs. Dashed grey line corresponds to D_{KL} calculated over true data sets.

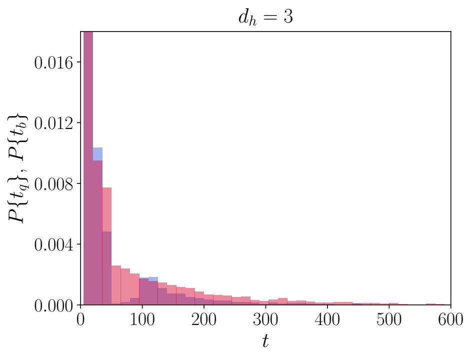
that the quiescent PDF spreads into regions with higher t_q and for $d_h = 9, 11$ these seem to agree better with the true PDF. Figure 2.15 shows D_{KL} with varying d_h for these PDFs. As expected from observing the PDFs we see that D_{KL} decreases up until $d_h = 5$ for both cases. In the case of t_q we see an increase in the error after $d_h = 5$ which agrees with the above observation of the PDF at $d_h = 7$. For t_b, D_{KL} seems to keep slightly decreasing after $d_h = 5$. We also notice that for t_q, D_{KL} reaches a minimum at $d_h = 9$ and for t_b no significant decrease is observed at $d_h \geq 9$. In short, these duration statistics achieve similar agreement at $d_h = 9$, and for the case of t_b errors keep decreasing with increasing d_h. We also calculate the mean of t_q and t_b for the case of $d_h = 9$ and obtain values of $\langle t_q \rangle = 174$ and $\langle t_b \rangle = 97$ which agree closely with the true values of $\langle t_q \rangle = 176$ and $\langle t_b \rangle = 97$.

2.4.4 Phase prediction

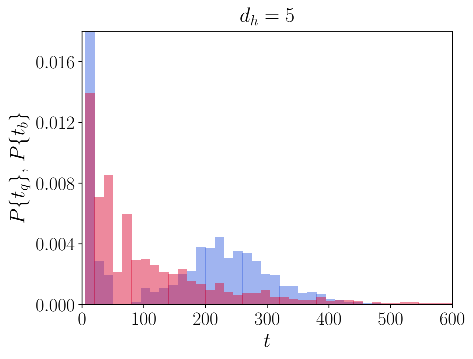
Recall that we gain substantial accuracy in dimension reduction by factoring out the spatial phase $\phi_x(t)$ of the data. Here we complete the dynamical picture of the model predictions at Re $= 14.4$ by illustrating the predictions of phase evolution, as given by the learned phase evolution equation (2.12). Figure 2.16a shows a short time evolution of $\phi_x(t)$ corresponding to the true and predicted data for the $d_h = 3, 5, 7, 9$, and 11 models. The smooth increases and



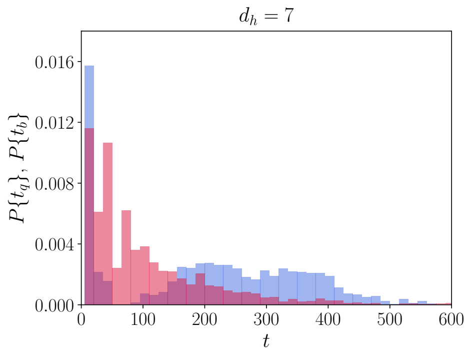
(a)



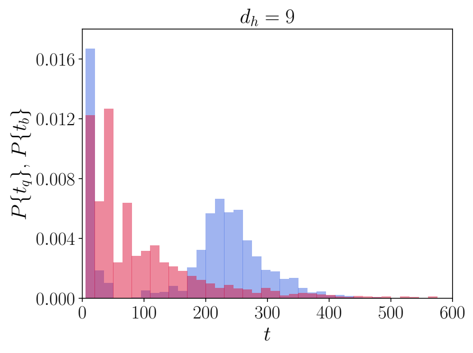
(b)



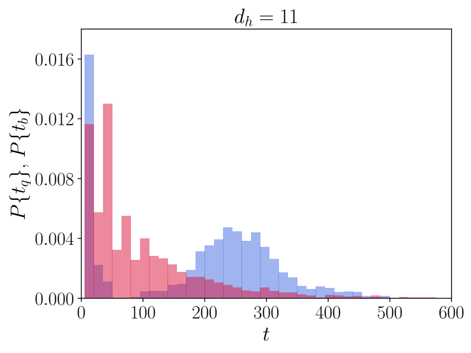
(c)



(d)



(e)



(f)

Figure 2.14: PDFs of t_q and t_b at Re $= 14.4$ for (a) true and (b)-(f) predicted data for dimensions $d_h = 3, 5, 7, 9$, and 11.

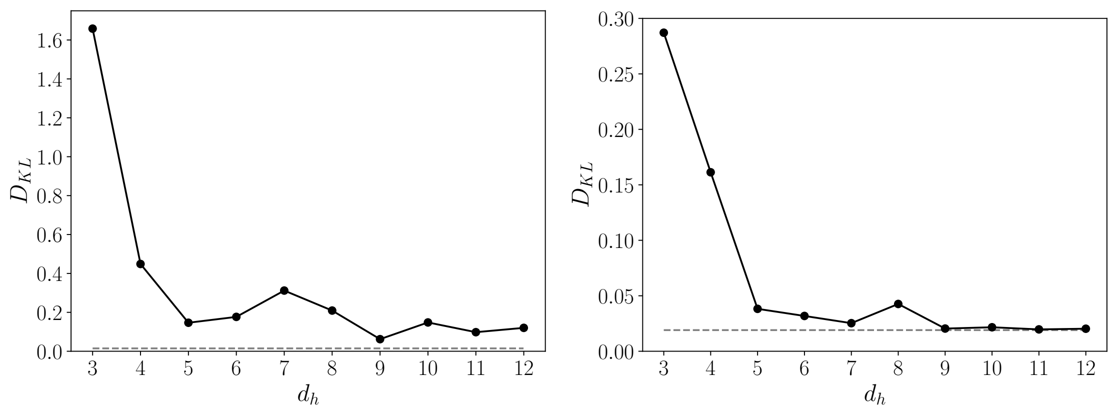
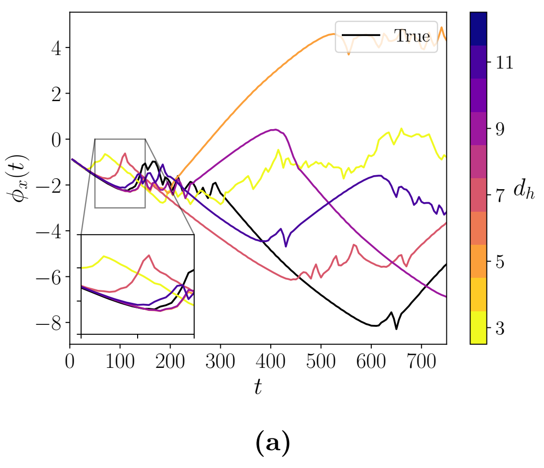


Figure 2.15: Re = 14.4: D_{KL} vs dimension d_h corresponding to PDFs for (a) t_q (b) t_b. Dashed grey line corresponds to D_{KL} calculated over different true data sets.

decreases in Figure 2.16a correspond to trajectories during time intervals where they are near an RPO and thus are traveling in the x-direction. The intervals where the phase flucuates rapidly are the bursts during which the trajectories are moving between the RPO regions. This behavior is well-captured for all of the dimensions shown except for $d_h = 3$. Notice that although the trajectories diverge, for short times we get around two t_L of prediction horizon where the models still capture the correct dynamics, and Figure 2.16a provides a clear visual indications that the loss of predictability occurs during the bursts.

We now take an approach to quantify how well the model performs with respect to the true data. Taking a look at the drops and increases for $\phi_x(t)$ we can observe that after every burst the trajectory will either travel, essentially randomly, in the positive (increasing ϕ_x) or negative (decreasing ϕ_x) x direction. This behavior is essentially a run and tumble or random walk behavior in the sense that the long periods of positive or negative phase drift correspond to “runs" that are separated by “tumbles" that correspond to the bursts, in which the direction of phase motion is reset. Hence, a natural analysis of quantification for this type of dynamics consists of calculating the mean squared displacement (MSD) of the phase:

$$\mathrm{MSD}(t) = \langle (\phi_x(t) - \phi_x(0))^2 \rangle. \tag{2.15}$$



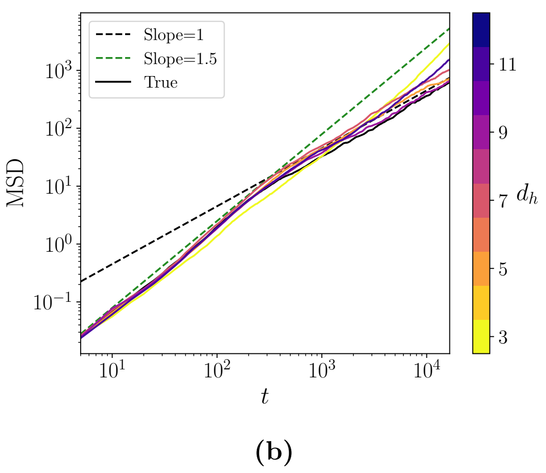


Figure 2.16: (a) Time evolution of ϕ_x corresponding to the true data and models with dimensions $d_h = 3, 5, 7, 9$, and 11. (b) MSD of $\phi_x(t)$ corresponding to true data and models with dimensions $d_h = 3, 5, 7, 9$, and 11.

Figure 2.16b shows the time evolution of MSD of true and predicted data. The black line corresponds to the true data and the black and green dashed lines serve as references with slopes of 1 and 1.5, respectively. The colored lines correspond to models with various dimensions. Looking at the true curve we notice a change from superdiffusive (slope = 1.5) to diffusive (slope = 1) scaling that happens around $t \approx 200$, which corresponds to the mean duration of the quiescent intervals, as discussed above: i.e., to the average time the trajectories travel along the RPOs before bursting. The trajectory then bursts and reorients which is captured by the long time diffusive trend. Looking at the performance of the models we observe that $d_h = 3$ does a good job at capturing the short time scaling, however it is not to able capture the change in slope that is observed in the true data. It is not until $d_h \geq 5$ that the correct behavior at long times is observed – indeed the predictions agree very well with the data, with a slight upward shift at long times corresponding to the slight overprediction of the mean duration of the quiescent periods.

2.4.5 Bursting prediction

Previous research has focused on finding indicators that guide predictions of when a burst will occur. It has been shown for the Kolmogorov flow that before a burst there is a depletion

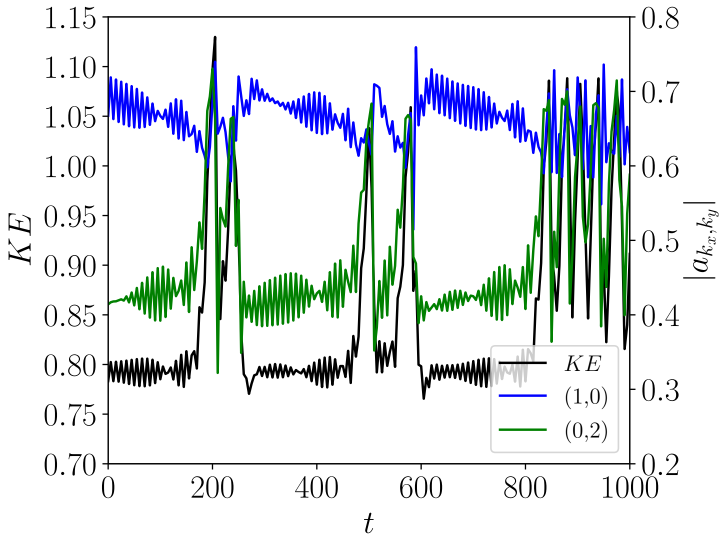


Figure 2.17: Time evolution of KE and amplitudes corresponding to $(1,0)$ and $(0,2)$ Fourier mode for Re $= 14.4$.

of the content in the $(1,0)$ Fourier mode, which then feeds into the forcing mode $(0,n)$ [49]. Figure 2.17 shows how this looks for Re $= 14.4$, $n = 2$. By considering a variational framework and finding solutions to a constrained optimization problem it was also found that examination of these modes can lead to predictions of when a burst will occur [22].

With our framework, natural indicators are the latent variables h, which we will consider here along with some variations, including the indicators used in previous work. To predict bursting events based on a given indicator, we will use a simple binary classifier in the form of a support vector machine (SVM) with a radial basis function kernel [6]. These have shown success in predicting extreme events for problems such as extreme rainfall [48]. With this approach, data at time t is used to learn a function that outputs a binary label of bursting/not bursting at time $t + \tau_b$. For all of the cases considered we use the $d_h = 5, 9$ models, taking a dataset of 5×10^4 snapshots to train the SVM and another 5×10^4 as a test set.

Figure 2.18a shows the percent correct classification of bursting events with varying time τ_b in the future. The black and gray curves corresponds to predicting the events based on the PCA projection of the data, $P_{d_h} U^T \omega$, into the first $d_h = 5$ and 9 coefficients respectively.

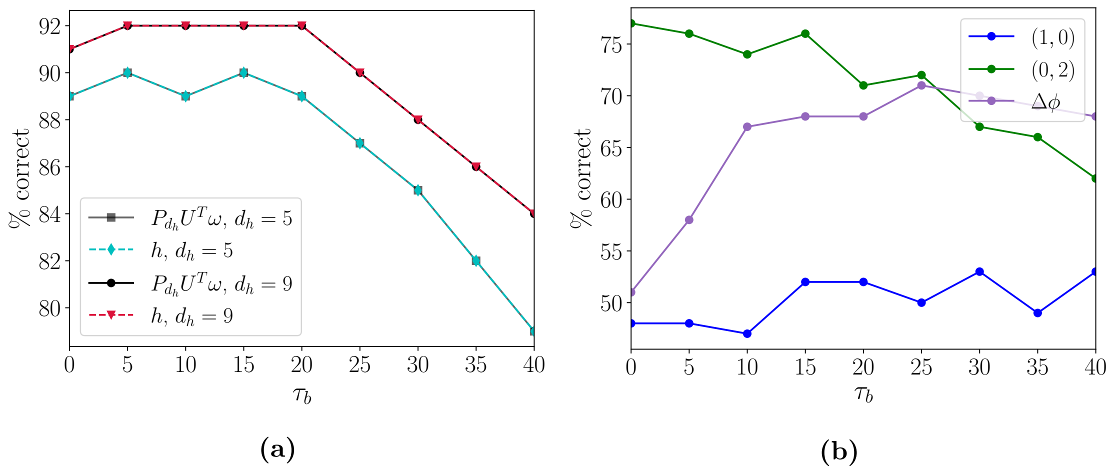


Figure 2.18: Percent of correctly classified bursting events at τ_b forward in time for: (a) $P_{d_h}U^T\omega$ and h at $d_h = 5, 9$, (b) and indicators $\Delta\phi$, $(1,0)$, and $(0,2)$. Note that the vertical scales on (a) and (b) are very different.

The cyan and red curves corresponds to h of dimensions $d_h = 5$ and $d_h = 9$ respectively. We notice that the PCA and h curves fall on top of another and have a high probability of correct classification when considering prediction horizons less than one t_L. For this purpose we see that PCA is enough to predict bursting events. Figure 2.18b shows the percent correct classification of bursting at time τ_b in the future for the previous discussed indicators. None of these work nearly as well as $P_{d_h}U^T\omega$ or h. The blue curve corresponds to $(1,0)$ amplitude of the original true data, the green curve to the forcing $(0,2)$ amplitude, and we also consider $\Delta\phi$ in the purple curve. In the case of $\Delta\phi$ we see some predictability at times longer than one t_L and less than two. This also happens for the case of $(1,0)$, however there seems to be no decrease or increase in the probability of correct classification. We can see from Figure 2.17 that even though there is a depletion in the $(1,0)$ mode preceding bursts, its amplitude does not change dramatically between quiescent and bursting intervals, which may be a reason that it does not provide much predictive power. The amplitude $(0,2)$, which changes more strongly between quiescent and bursting regions, is seen to be the better predictor for bursting events. At small τ_b its predictions outperform $(1,0)$ and $\Delta\phi$, however at times larger than one t_L, $\Delta\phi$ performs better.

2.5 Summary

The nonlinearity of the NSE poses challenges when using ROMs, where the dynamics are expected to evolve on an invariant manifold that will not lie in a linear subspace. Neural networks have proven to be powerful tools for learning efficient ROMs solely from data, however finding and exploiting a *minimal*-dimensional model has not been emphasized. We present a data-driven methodology to learn an estimate of the embedding dimension of the manifold for chaotic Kolmogorov flow and the time evolution on it. An autoencoder is used to find a nonlinear low-dimensional subspace and a dense neural network to evolve it in time.

Our autoencoders are trained on vorticity data from two cases: a case where the dynamics show a relative periodic orbit solution ($\mathrm{Re} = 13.5$), and a case with chaotic dynamics ($\mathrm{Re} = 14.4$). The chaotic regime we consider comes with challenges due to the intermittent behavior observed where the trajectory travels in between quiescent intervals and bursting events. We factor out the rich symmetries of Kolmogorov flow before training of the autoencoders, which dramatically improves reconstruction error of the snapshots. This improves training efficiency by not having to learn a compression of the full state. Specifically, factoring out the translation symmetry decreases the mean-squared reconstruction error by an order of magnitude compared to the case where phase is not factored out, and several orders of magnitude compared to PCA. The phase-aligned low-dimensional subspace is then used for time evolution where the RPO dynamics is learned essentially perfectly at $d_h = 2$ for $\mathrm{Re} = 13.5$ and very good agreement for short and long time statistics is obtained at $d_h = 5$ for $\mathrm{Re} = 14.4$. Further small improvements in the results occur as dimension is increased to nine, beyond which the statistics of the model and true system are in very good agreement. For comparison, the full state space of the numerical simulation data is $N = 1024$.

We also show phase prediction evolution results based on the low-dimensional subspace learned. The time evolution of the true phase exhibits a superdiffusive scaling at short times and a diffusive scaling at long times which we attribute to the traveling near an RPO and the reorientation due to bursting. Finally, using the low-dimensional representation enables accurate prediction of bursting events based on conditions about a Lyapunov time ahead of the event. This work opens new avenues for data-driven ROMs with applications such as control for drag reduction, an example of which is presented for turbulent Couette flow in [41]. One important challenge that remains is more effective treatment of systems with intermittent dynamics like those described here. A recent study [23] has introduced a method that uses the differential topology formalism of charts and atlases to develop *local* manifold representations and dynamical model that can be stitched together to form a global dynamical model. One attractive feature of that formalism is that it enables use of separate representations for regions of state space with very different dynamics, and has already shown in specific cases to provide dramatically improved results for dynamics with intermittency.

3

Building symmetries into data-driven manifold dynamics models for complex flows [1]

Symmetries in a dynamical system provide an opportunity to dramatically improve the performance of data-driven models. For fluid flows, such models are needed for tasks related to design, understanding, prediction, and control. In this work we exploit the symmetries of the Navier-Stokes equations (NSE) and use simulation data to find the manifold where the long-time dynamics live, which has many fewer degrees of freedom than the full state representation, and the evolution equation for the dynamics on that manifold. We call this method “symmetry charting”. The first step is to map to a “fundamental chart”, which is a region in the state space of the flow to which all other regions can be mapped by a symmetry operation. To map to the fundamental chart we identify a set of indicators from the Fourier transform that uniquely identify the symmetries of the system. We then find a low-dimensional coordinate representation of the data in the fundamental chart with the use of an autoencoder. We use a variation called an implicit rank minimizing autoencoder

[1]The text of this chapter is adapted from the prepublication by C. E. Pérez De Jesús, A. J. Linot, and M. D. Graham on arXiv.

with weight decay, which in addition to compressing the dimension of the data, also gives estimates of how many dimensions are needed to represent the data: i.e. the dimension of the invariant manifold of the long-time dynamics. Finally, we learn dynamics on this manifold with the use of neural ordinary differential equations. We apply symmetry charting to two-dimensional Kolmogorov flow in a chaotic bursting regime. This system has a continuous translation symmetry, and discrete rotation and shift-reflect symmetries. With this framework we observe that less data is needed to learn accurate data-driven models, more robust estimates of the manifold dimension are obtained, equivariance of the NSE is satisfied, better short-time tracking with respect to the true data is observed, and long-time statistics are correctly captured.

3.1 Introduction

In recent years, neural networks (NNs) have been implemented to learn data-driven low-dimensional representations and dynamical models of flow problems, with success in systems including the Moehlis-Faisst-Eckhardt (MFE) model [59], Kolmogorov flow [20, 52], and minimal turbulent channel flow [47]. For the most part however, these approaches do not explicitly take advantage of the fact that for dissipative systems like the Navier-Stokes Equations (NSE), the long-time dynamics are expected to lie on an invariant manifold $\mathcal{M}$ (sometimes called an inertial manifold [24, 62, 71]), $d_{\mathcal{M}}$, whose dimension may be much smaller than the nominal number of degrees of freedom N required to specify the state of the system. Ideally, one could identify $d_{\mathcal{M}}$ from data, find a coordinate representation for points on $\mathcal{M}$, and learn the time-evolution on $\mathcal{M}$ in those coordinates. This would be a minimal-dimensional data-driven dynamic model. Linot & Graham explored this approach for chaotic dynamics of the Kuramoto-Sivashinsky equation (KSE) [37, 38]. They showed that the mean-squared error (MSE) of the reconstruction of the snapshots using an autoencoder (AE) for dimension reduction for the domain size of $L = 22$ exhibited an orders-of-magnitude drop when the dimension of the inertial manifold is reached. Furthermore, modeling the dynamics with a dense NN at this dimension either with a discrete-time map [37] or a system of ordinary differential equations (ODE) [38] yielded excellent trajectory predictions and long-time statistics. The approach they introduced is referred to here as *data-driven manifold dynamics* (DManD). For the KSE in larger domains, it was found that simply observing MSE vs. autoencoder bottleneck dimension was not sufficient to determine the manifold dimension and exhaustive tests involving time evolution models vs. dimension were necessary to estimate the manifold dimension. Pérez De Jesús & Graham extended this approach to two-dimensional Kolmogorov flow in a chaotic regime [53]. Here dimension reduction from 1024 dimensions to ≤ 10 was achieved, with very good short-time tracking predictions, long-time statistics, as well as accurate predictions of bursting events. Linot & Graham considered three-dimensional direct numerical simulations of turbulent Couette flow at Re $= 400$ and found accurate data-driven dynamic models with fewer than 20 degrees of freedom [39]. These models were able to capture characteristics of the flows such as streak breakdown and regeneration, short-time tracking, as well as Reynolds stresses and energy balance. They also computed unstable periodic orbits from the models with close resemblance to previously computed orbits from the full system. Relatedly, Zeng et al. [74] exploited advances in autoencoder architecture [32] to yield more precise estimates of $d_{\mathcal{M}}$ for data from high-dimensional chaotic systems. The present work illustrates how symmetries of a flow system can be exploited in the DManD framework to yield highly efficient low-dimensional data-driven models for chaotic flows.

A fundamental notion in the topology of manifolds, which will be useful in exploiting symmetry, is that of charts and atlases [36]. Simply put, a chart is a region of a manifold whose points can be represented in a local Cartesian coordinate system of dimension $d_{\mathcal{M}}$ and which overlaps with neighboring charts, while an atlas is a collection of charts that covers the manifold. This representation of a manifold has several advantages. First, for a manifold with dimension $d_{\mathcal{M}}$, it may not be possible to find a *global* coordinate representation in $d_{\mathcal{M}}$ dimensions: Whitney's embedding theorem states that generic smooth maps for a smooth manifold of dimension $d_{\mathcal{M}}$ can be embedded into a Euclidean space of $2d_{\mathcal{M}}$. Dividing a manifold into an atlas of charts enables *minimal-dimensional* representations locally in $d_{\mathcal{M}}$ dimensions. Second, from the dynamical point of view, dynamics on different parts of a manifold may be very different and a single global representation of a manifold and the dynamics on it may not efficiently capture the dynamics, especially in the data-driven context. These two advantages of an atlas-of-charts representation have been exploited for data-driven modeling. Floryan & Graham developed a method to implement data-driven local representations for dynamical systems such as the quasiperiodic dynamics on a torus, a reaction-diffusion system, and the KSE to learn dynamics on invariant manifolds of minimal dimension [23]. They refer to this method as *Charts and Atlases for Nonlinear Data-Driven Dynamics on Manifolds* – "CANDyMan". Fox et al. then applied this to the MFE model, which displays highly intermittent behavior in the form of quasilaminarization and full relaminarization events, demonstrating more accurate time evolution predictions as global ("single chart") model [25]. In the present work, we use the charts and atlases framework to exploit symmetries.

In the Navier-Stokes equations (NSE), symmetries appear in the form of continuous and discrete symmetry groups. The symmetries of the NSE in physical space are reflected in state space as symmetries of the vector field, and these can generate natural charts of the system. Consider the vector field $\boldsymbol{q}$ shown in Fig. 3.1a where the operation $\mathcal{R}\boldsymbol{q}$ rotates the vector field by $\pi/2$ in the clockwise direction. Trajectories of the ODE $\dot{\boldsymbol{y}} = \boldsymbol{q}$ will reflect this symmetry. Without exploiting symmetry, any model trained on this data will need to represent the whole state space and even so will generally not obey the exact symmetries of the true system due to finite sampling effects. In this work we leverage knowledge of the charts by explicitly considering and factoring out the symmetries. We show a depiction of this in Fig. 3.1b where we know that by applying $\mathcal{R}$ to the data we can map it to a different quadrant, and that by mapping points in quadrants 2, 3, and 4 with $\mathcal{R}$, $\mathcal{R}^2$, and

$\mathcal{R}^3$, respectively, they all end up in quadrant 1. This is shown in Fig. 3.1c where the data collapses on top of each other when mapped to the first quadrant. We will call this region the "fundamental domain" for the state space, and when expanded to overlap with its neighbors, the "fundamental chart".

Some previous work has focused on factoring out symmetries of dynamical systems for dimension reduction in the data-driven context. However, symmetries have not been considered explicitly for data-driven reduced-order models (ROMs). Kneer et al. built symmetries into an AE architecture for dimension reduction and applied it to the Kolmogorov flow system [34]. With the use of branches that receive the different discrete symmetries of the snapshots and spatial transformer networks that manipulate the continuous phase, they were able to map to a fundamental domain by selecting the branch that gives the smallest MSE and backpropagating through it. By doing this, the AE naturally selects a path that leads to the lowest MSE of reconstruction while considering the symmetries of the system. The purpose of this work was not to find the minimal dimension or learn ROMs. Budanur & Cvitanovic formulated a symmetry-reducing scheme, previously applied to the Lorenz equations [45], to the KSE, where a polynomial transformation of the Fourier modes combined with the method-of-slices for factoring out spatial phase leads to a mapping in the fundamental domain [7]. Applying this method to more complicated systems, such as Kolmogorov flow, may be quite challenging. Symmetry reduction has also been successfully used for controlling the KSE with synthetic jets. Zeng & Graham applied discrete symmetry operations to map the state to the fundamental domain, and used this as the input to a reinforcement learning control agent, the performance of which is dramatically improved by exploitation of symmetry [72].

In the present work, we present a framework for learning DManD models in the fundamental chart, as in Fig. 3.1c, where the dynamics of the flow occur, and apply it to chaotic dynamics of the NSE. We refer to this method as "symmetry charting", which results in (at least approximately) minimal-dimensional data-driven models that are equivariant to

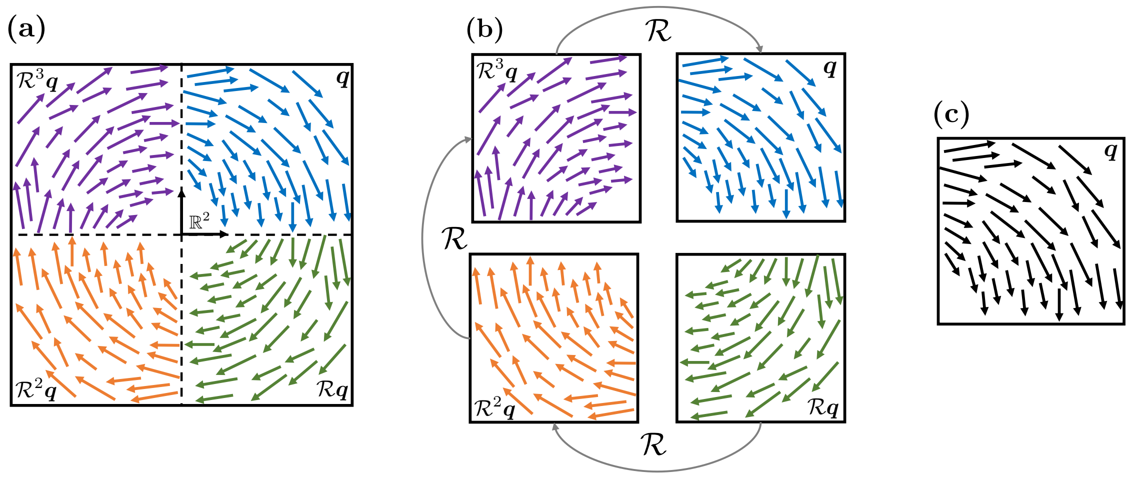

(a)
$\mathcal{R}^3\boldsymbol{q}$
$\boldsymbol{q}$
$\mathbb{R}^2$
$\mathcal{R}^2\boldsymbol{q}$
$\mathcal{R}\boldsymbol{q}$
(b)
$\mathcal{R}$
(c)

Figure 3.1: (a) Vector field $\boldsymbol{q}$ with symmetry group $\mathcal{R}$ in state space. (b) Data in each quadrant can be mapped to another quadrant by operating with $\mathcal{R}$. (c) After mapping everything to the positive quadrant data lies on top of each other.

symmetry operations. In addition the model only need to be learned in one chart because a combination of its symmetric versions will cover the manifold. We first learn minimal-dimensional representations in the fundamental chart followed by an evolution equation for the dynamics on it. To this end we use information about the symmetries of the system combined with NNs to learn the compressed representation and dynamics.

We apply symmetry charting to two-dimensional Kolmogorov flow in a regime where the chaotic dynamics travels between unstable relative periodic orbits (RPOs) [18] through bursting events [2] that shadow heteroclinic orbits connecting the RPOs. RPOs correspond to periodic orbits in a moving reference frame, such that in a fixed frame, the pattern at time $t + T$ is a phase-shifted replica of the pattern at time t. To achieve this goal we map the data to a fundamental chart which is then used to train undercomplete AEs to find low-dimensional representations. To this end, we use Implicit Rank Minimizing autoencoders with weight decay (IRMAE-WD) [74]. This architecture was introduced by Zeng & Graham as a method that estimates $d_{\mathcal{M}}$ without the need to sweep over dimensions and using MSE and statistics as indicators [37, 53]. We give an overview of IRMAE-WD in Section 3.3.2. After learning a low-dimensional representation with IRMAE-WD we learn a time map with the use of the neural ODE (NODE) [14], which we discuss in Section 3.3.3. We then show results in Section 3.4 and finish with concluding remarks in Section 3.5.

3.2 Kolmogorov flow, symmetries, and projections

The two-dimensional NSE with Kolmogorov forcing are

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + \boldsymbol{\nabla} p = \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \sin(ny)\hat{\boldsymbol{x}}, \tag{3.1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{3.2}$$

where flow is in the $x - y$ plane, $\boldsymbol{u} = [u, v]$ is the velocity vector, p is the pressure, n is the wavenumber of the forcing, and $\hat{\boldsymbol{x}}$ is the unit vector in the x direction. Here $\mathrm{Re} = \frac{\sqrt{\chi}}{v}\left(\frac{L_y}{2\pi}\right)^{3/2}$ where χ is the dimensional forcing amplitude, ν is the kinematic viscosity, and L_y is the size of the domain in the y direction. We consider the periodic domain $[0, 2\pi/\alpha] \times [0, 2\pi]$ with $\alpha = 1$. Vorticity is defined as $\omega = \hat{\boldsymbol{z}} \cdot \boldsymbol{\nabla} \times \boldsymbol{u}$, where $\hat{\boldsymbol{z}}$ is the unit vector in the z direction (orthogonal to the flow). The equations are invariant under several symmetry operations [13], namely a shift (in y)-reflect (in x), a rotation through π, and a continuous translation in x:

$$\mathscr{S} : [u, v, \omega](x, y) \to [-u, v, -\omega]\left(-x, y + \frac{\pi}{n}\right), \tag{3.3}$$

$$\mathscr{R} : [u, v, \omega](x, y) \to [-u, -v, \omega](-x, -y), \tag{3.4}$$

$$\mathscr{T}_l : [u, v, \omega](x, y) \to [u, v, \omega](x + l, y). \tag{3.5}$$

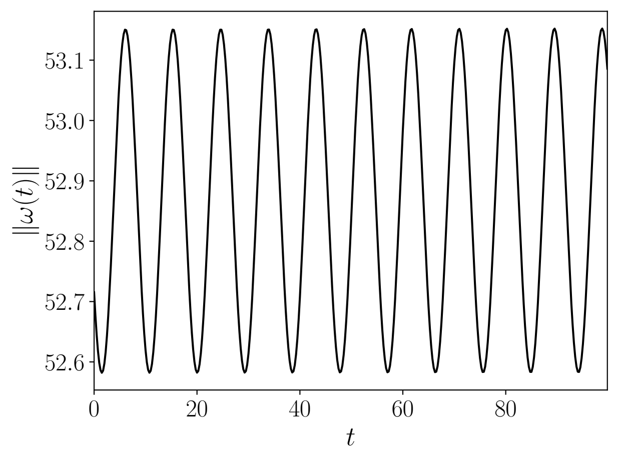
For n wavelengths the state can be shift-reflected $2n-1$ times with the operation $\mathscr{S}$. Together with the rotation $\mathscr{R}$ there will be a total of $4n$ states that are related by symmetries. In the case of $n = 2$ this results in eight regions of state space that are related to one another by symmetries. In the chaotic regime that we consider, trajectories visit each of these regions. Relatedly, velocity fields for the Kolmogorov flow satisfy the *equivarance* property associated with these actions: if $\boldsymbol{u}$ is a solution at a given time, then so are $\mathscr{S}\boldsymbol{u}$, $\mathscr{R}\boldsymbol{u}$, and $\mathscr{T}_l\boldsymbol{u}$. We elaborate on this property in Sec. 3.3.1.

The total kinetic energy for this system (KE), dissipation rate (D) and power input (I) are

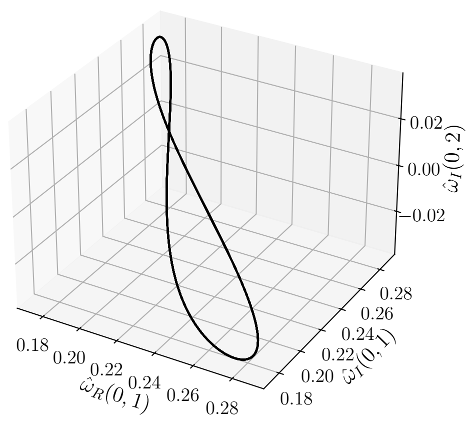
$$KE = \frac{1}{2}\left\langle \boldsymbol{u}^2 \right\rangle_V, D = \frac{1}{\mathrm{Re}}\left\langle |\nabla \boldsymbol{u}|^2 \right\rangle_V, \quad I = \langle u \sin(ny) \rangle_V \tag{3.6}$$

where $\langle \cdot \rangle_V$ corresponds to the average taken over the domain. For the case of $n = 1$ the laminar state is linearly stable at all Re [30]. It is not until $n = 2$ that the laminar state becomes unstable, with a critical value of $\mathrm{Re}_c = n^{3/2}2^{1/4}$[26, 44, 63]. In what follows, we evolve the NSE numerically in the vorticity representation on a $[d_x \times d_y] = [32 \times 32]$ grid following the pseudo-spectral scheme given by Chandler & Kerswell [13], which is based on the code by Bartello & Warn [4].

In Fig. 3.2 we show the time-series evolution of $\|\omega(t)\|$, where $\|\cdot\|$ is the l^2-norm, for an RPO obtained at $\mathrm{Re} = 13.5$, $n = 2$, as well as the state-space projection of the trajectory into the subspace $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1) - \hat{\omega}_I(0,2)$ where $\hat{\omega}(k_x, k_y, t) = \mathcal{F}\{\omega(x, y, t)\} = \hat{\omega}_R(k_x, k_y, t) + i\hat{\omega}_I(k_x, k_y, t)$ is the discrete Fourier transform in x and y, and subscripts R and I correspond to real and imaginary parts. We also consider chaotic Kolmogorov flow with $\mathrm{Re} = 14.4$, $n = 2$. Fig. 3.3 shows the time-series evolution of $\|\omega(t)\|$ as well as the state-space projection of the trajectory into the subspace $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1) - \hat{\omega}_I(0,2)$ for this chaotic regime. Due to the discrete symmetries of the system, there are several RPOs [2]. The dynamics are characterized by quiescent intervals where the trajectories approach the RPOs (which are now unstable), punctuated by fast excursions between the RPOs, which are indicated by the intermittent increases of the $\|\omega(t)\|$ in Fig. 3.3a. Under the projection shown in Fig. 3.3b, we see four RPOs, which initially seems surprising as there are eight discrete symmetries and a continuous symmetry. However, the continuous symmetry is removed under this projection (it will only appear for $k_x \neq 0$), and the discrete symmetry operations of $\mathscr{R}$ and $\mathscr{S}$ will flip the signs of $\hat{\omega}_R(0,1)$, $\hat{\omega}_I(0,1)$, and $\hat{\omega}_I(0,2)$ such that portions of the different RPOs are covered. Specifics of the sign changes will be made clear in Section 3.3.1.

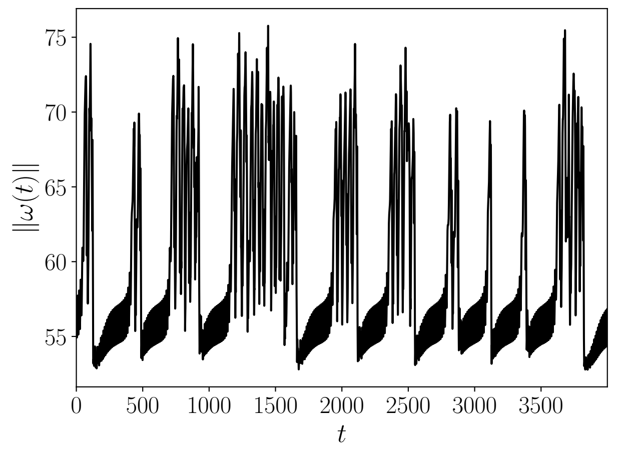


(a)

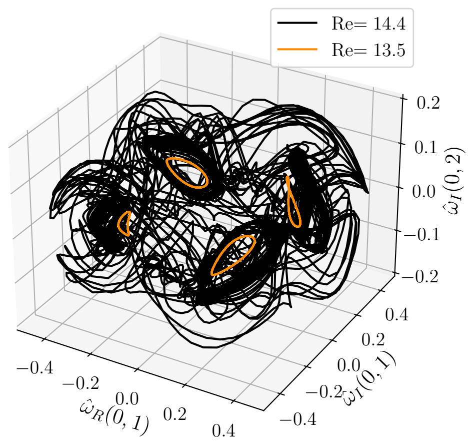


(b)

Figure 3.2: (a) Time evolution of $\|\omega(t)\|$ at Re = 13.5. (b) State-space projection of the trajectory into the subspace $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1) - \hat{\omega}_I(0,2)$ for Re = 13.5.



(a)



(b)

Figure 3.3: (a) Time evolution of $\|\omega(t)\|$ at Re = 14.4. (b) State-space projection of the trajectory into the subspace $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1) - \hat{\omega}_I(0,2)$ for Re = 13.5 and Re = 14.4.

3.3 Data-driven dimension reduction and dynamic modeling

In the following sections, we describe the steps involved in symmetry charting: 1) mapping data to the fundamental space, 2) finding a minimal-dimensional coordinate representation, and 3) evolving the minimal-dimensional state and symmetry indicators forward through time. The parts of this procedure separating it from other data-driven reduced order models are how we map to the fundamental and how we evolve the symmetry indicators through time. Our approach guarantees equivariance under symmetry operations.

3.3.1 Map to fundamental domain

First we discuss a method to map trajectories of Kolmogorov flow to the fundamental domain where symmetries are factored out [7]. The first step is to identify a set of indicators that are related to the group of symmetries. In order to find these indicators we consider the effect of these symmetry operations in Fourier space. After Fourier transforming the equations and simplifying these are the actions of the symmetry operations on the Fourier coefficients:

$$\mathscr{S} : \hat{\omega}(k_x, k_y) \to -\hat{\omega}(-k_x, k_y) e^{i k_y \pi / n}, \tag{3.7}$$

$$\mathscr{R} : \hat{\omega}(k_x, k_y) \to \bar{\hat{\omega}}(k_x, k_y), \tag{3.8}$$

$$\mathscr{T}_l : \hat{\omega}(k_x, k_y) \to \hat{\omega}(k_x, k_y) e^{-i k_x l}, \tag{3.9}$$

where $\bar{\cdot}$ denotes the complex conjugate.

Now, we will show how these symmetry operations act on the Fourier coefficients in specific ways that we can exploit to classify any given state as lying within a specific region or domain of state space. We refer to this classification of the continuous symmetry as the "phase" and the discrete symmetry as the "indicator". To compute the phase and remove the continuous symmetry $\mathscr{T}_l$, we use the First Fourier mode method-of-slices [10, 11]. This method involves computing the spatial phase ϕ_x of the $k_x = 1$ and $k_y = 0$ Fourier mode: $\phi_x(t) = \operatorname{atan2}\{\hat{\omega}_I(1,0), \hat{\omega}_R(1,0)\}$. Then, to phase-align the data, we shift the vorticity snapshots such that this mode is a pure cosine: $\omega_l(x,y,t) = \mathcal{F}^{-1}\left\{\mathcal{F}\{\omega(x,y,t)\}e^{-ik\phi_x(t)}\right\}$. Doing this ensures that the snapshots lie in a reference frame where no translation happens in the x direction. From Eq. 3.9 we see that this operation only modifies Fourier coefficients with $k_x \neq 0$. Thus, if we use $k_x = 0$ coefficients to determine the discrete indicator it will be independent of the phase.

For the shift-reflect operation, we consider the $(k_x = 0, k_y = 1)$ mode. In Eq. 3.7 the shift-reflect operation modifies this Fourier coefficient in this manner: $\hat{\omega}_R(0,1) + i\hat{\omega}_I(0,1) \to \hat{\omega}_I(0,1) - i\hat{\omega}_R(0,1)$. This shows that the shift-reflect operation maps between different quadrants of the $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1)$ plane. Next, we consider the rotation operation on the $(k_x = 0, k_y = 2)$ mode. In Eq. 3.8 we see this operation simply flips the sign of the imaginary component $\hat{\omega}_I(0,2) \to -\hat{\omega}_I(0,2)$. Thus, applying the eight possible discrete symmetry operations (including the identity operation) to a vorticity field will map us to eight unique points in the $\hat{\omega}_R(0,1)$, $\hat{\omega}_I(0,1)$, $\hat{\omega}_I(0,2)$ space. As such, we may classify each octant with an $\mathcal{I}$ between $0-7$. This means that any snapshot can be mapped to a fundamental domain and be uniquely identified by $\mathcal{I}$. In Fig. 3.4 we show the indicators for Re $= 13.5$. The colored curves correspond to the sections of the different RPOs that are related by discrete symmetries. Notice that each section of the different RPOs can be uniquely identified by $\mathcal{I}$. Table 3.1 shows how any snapshot in any octant can be mapped to the $\mathcal{I} = 0$ octant by applying the corresponding discrete symmetry operations. These same indicators extend to the chaotic case Re $= 14.4$ and in the rest of this work we refer to the fundamental domain as the space corresponding to $\mathcal{I} = 0$. Fig. 3.5 shows the state-space projection of a chaotic trajectory at Re $= 14.4$ into the region $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1) - \hat{\omega}_I(0,2)$, where the different colors give the indicator for the chart in which that data point lies. More generally, we note that any point in the state space of the system can represented by a point in the fundamental domain along with an indicator $\mathcal{I}$ and phase ϕ_x.

This representation is satisfactory for static data, but not for modeling trajectories represented in the fundamental domain, because when a trajectory exits one region, its corresponding trajectory in the fundamental domain leaves there and instantaneously enters at a different point: i.e. the time evolution is discontinuous in this representation.

To address this issue we use the concept of charts and atlases that is fundamental to the study of manifolds [36], in a way that is closely related to that of Floryan & Graham [23]. An atlas for a manifold is a collection of patches, each of which must overlap with its neighbors, that are called charts. In each chart, a local coordinate representation can be found, and for each pair of overlapping charts, a transition map takes coordinates in one chart to those in the neighboring one – with this coordinate tranformation there are no discontinuities. To use this formalism in the present case requires us to expand the fundamental domain so that it overlaps with its neighbors, so the fundamental chart is simply the fundamental domain, comprised of “interior points" plus the overlap region (“exterior points"), and there are seven other charts, each generated by the symmetry operation acting on the fundamental chart including the overlap regions. We choose a coordinate representation for all charts that corresponds to that for the fundamental chart, and for a trajectory that leaves the fundamental domain with e.g. $\mathcal{I} = 4$ and reenters with $\mathcal{I} = 6$, the transition map is the function that moves the point in the fundamental chart from the “exit" of the $\mathcal{I} = 4$ chart to the “entrance" of the $\mathcal{I} = 6$ chart.

In Fig. 3.5a we show the state space projection of a trajectory, identifying interior points with solid markers and exterior points with open markers. We isolate the region close to an RPO in Fig. 3.5b to make this more clear. Solid markers correspond to data points that lie in the interior of a chart. Open markers correpond to the data points in the connecting chart that will be included together with the interior data. Many different approaches can be taken to identify points to include in the exterior. Here we simply identified all points in a small volume around each octant as exterior points. We did this by finding the dimensions of a box surrounding the octant (by calculating the maximum absolute value in the $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1) - \hat{\omega}_I(0,2)$ directions) and then increasing the size of this box by expanding each direction 20%. Any point that lies in the volume between the octant and this expanded domain is an exterior point. We varied the percent increase to 5%, 10%, and 30% and observed no change in the results.

Now that we have identified these exterior points we can continue with the procedure of mapping the data to the fundamental chart. To do this, we apply the appropriate discrete operations to the interior and exterior points of each chart to map the state such that the interior points fall in $\mathcal{I} = 0$ ($\hat{\omega}_R(0,1) > 0$, $\hat{\omega}_I(0,1) > 0$, $\hat{\omega}_I(0,2) > 0$). Fig. 3.6a shows the colored clusters mapped onto the positive octant, dark blue cluster. Figures 3.6b and 3.6c show projections in the planes $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1)$ and $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,2)$ respectively of data mapped to the positive octant. The dashed lines correspond to the boundaries between the different octants. We see clearly that mapping the data to the fundamental chart leads to a much higher density of data points and thus to better sampling than if we considered the whole state space without accounting for the symmetries. Following the mapping to the fundamental chart we phase-align the snapshots and refer to this data as $\tilde{\omega}$ from this point on.

Table 3.1: Indicators $\mathcal{I}$ and the corresponding operations required to map to $\mathcal{I} = 0$

$\mathcal{I}$	Discrete operations
0	None
1	$\mathscr{R}\omega$
2	$\mathscr{S}^3\omega$
3	$\mathscr{S}^3\mathscr{R}\omega$
4	$\mathscr{S}^2\mathscr{R}\omega$
5	$\mathscr{S}\mathscr{R}\omega$
6	$\mathscr{S}^2\omega$
7	$\mathscr{S}\omega$

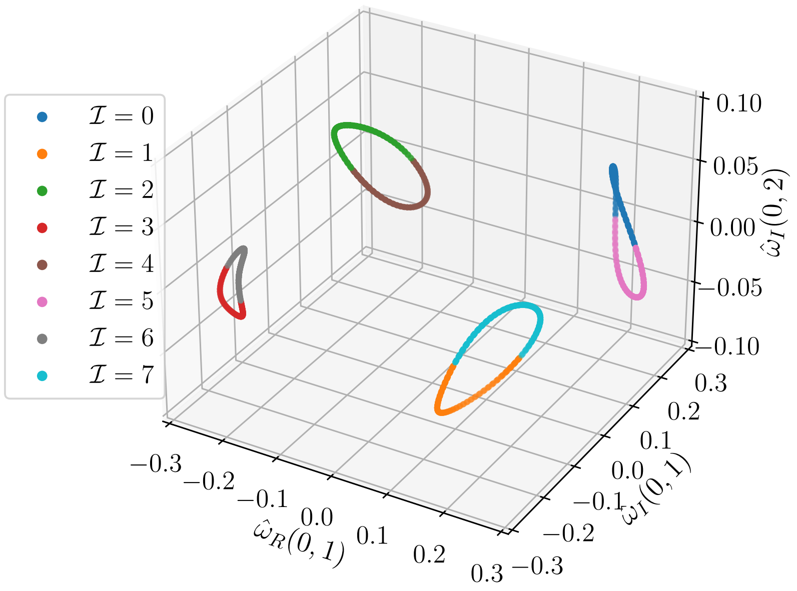


Figure 3.4: State-space projection of the different RPOs into the plane $\hat{\omega}_R(0,1)$–$\hat{\omega}_I(0,1)$–$\hat{\omega}_I(0,2)$ for Re = 13.5. The different colors are for different discrete symmetry indicators which are shown in the legend.

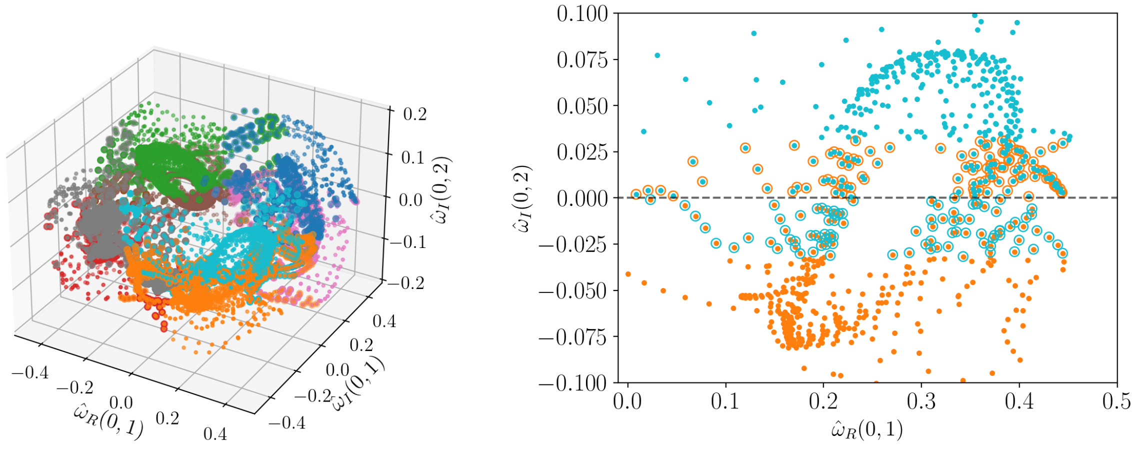
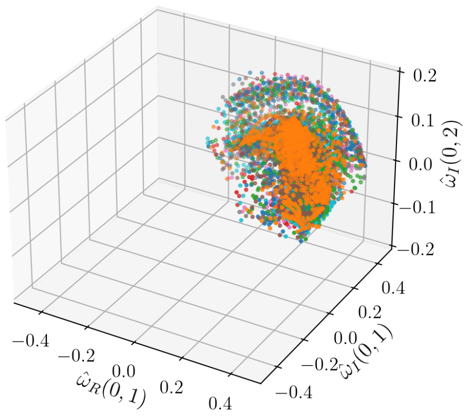
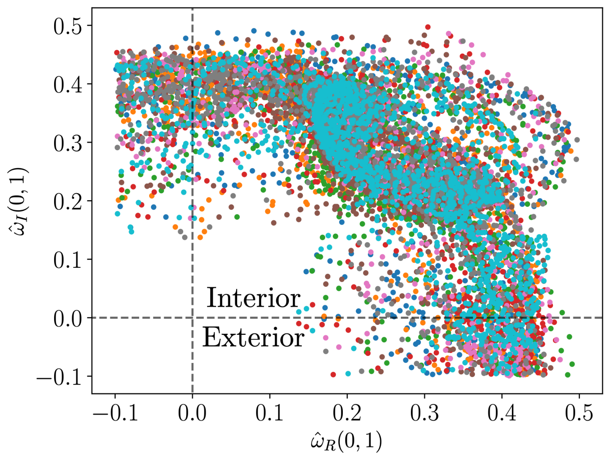


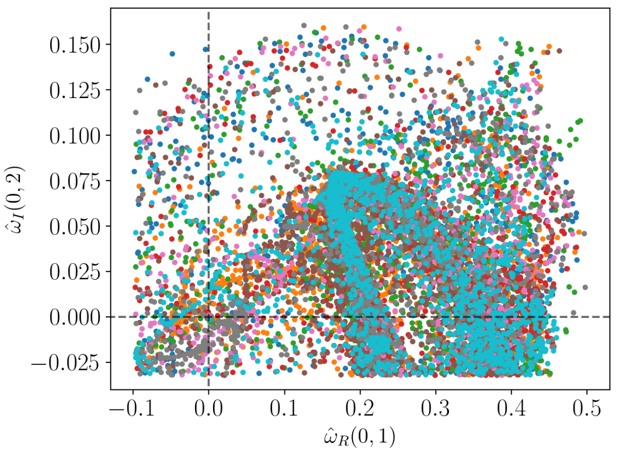
Figure 3.5: (a) State-space projection of a trajectory into the subspace $\hat{\omega}_R(0,1)$–$\hat{\omega}_I(0,1)$–$\hat{\omega}_I(0,2)$ for Re = 14.4. Colors correspond to the discrete symmetry indicator. (b) Zoom into the region near the RPO corresponding to $\mathcal{I} = 1, 7$. Solid markers correspond to interior points and open markers to exterior points – e.g. the blue solid markers within an open orange marker are in the interior region of chart 7 and exterior region of chart 1.



(a)



(b)



(c)

Figure 3.6: (a) Snapshots mapped to the fundamental chart ($\hat{\omega}_R(0,1) > 0$, $\hat{\omega}_I(0,1) > 0$, $\hat{\omega}_I(0,2) > 0$). (b) and (c) are the state-space projection of the trajectory into the planes $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1)$ and $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,2)$, respectively. The dotted lines give the boundary between interior and exterior points.

3.3.2 Finding a manifold coordinate representation with IRMAE-WD

In the previous section we discussed how to obtain the fundamental chart. We have defined a natural subdivision of the state space, where the invariant manifold of the dynamics is represented by an atlas of charts that are related by symmetry operations. This means that we do not need to learn the full invariant manifold, only the piece in the fundamental chart. To find a low-dimensional representation of the fundamental chart we use a variant of a common machine-learning architecture known as an undercomplete autoencoder (AE), whose purpose is to learn a reduced representation of the state such that the reconstruction error with respect to the true data is minimized. We flatten the vorticity field $\tilde{\omega}$ such that $N = 32 \times 32 = 1024$. The AE consists of an encoder and a decoder. The encoder maps from the full space $\tilde{\omega} \in \mathbb{R}^N$ to the lower-dimensional latent space $h(t) \in \mathbb{R}^{d_h}$, where ideally $d_h = d_{\mathcal{M}} \ll N$, and the decoder maps back to the full space $\tilde{\omega}_r$.

Previous works have focused on tracking MSE and dynamic model performance as d_h varies to find good low-dimensional representations [37, 39, 53]. This is a tedious trial-and-error process that in general does not yield a precise estimate of $d_{\mathcal{M}}$. Other works have learned compressed representations of flow problems [20, 47, 52]. However, performance over a systematic range of d_h is not examined in these cases. A more systematic alternative would be highly desirable. In recent work, Zeng et al. [74] showed that a straightforward variation on a standard autoencoder can yield robust and precise estimates of $d_{\mathcal{M}}$ for a range of systems, as well as an orthogonal manifold coordinate system. The architecture they study is called an Implicit Rank Minimizing Autoencoder with weigh decay (IRMAE-WD), and involves inserting a series of linear layers between the encoder and decoder and adding an L_2 regularization on the neural network weights in the loss. The effect of these additions is an AE for which the standard gradient descent algorithm for learning NN weights drives the rank of the covariance of the data in the latent representation to a minimum while maintaining representational capability. Applying this to the KSE and other systems resulted in the rank being equal to the dimension of the inertial manifold $d_{\mathcal{M}}$. IRMAE was originally presented by Jing et al. [32] to learn low-rank representations for image-based classification and generative problems.

Fig. 3.7a shows the IRMAE-WD framework. The encoder, denoted by $E\left(\tilde{\omega};\theta_E\right)$ reduces the dimension from N to d_z. We then include a linear network $\mathcal{W}\left(\cdot;\theta_W\right)$ between the encoder and the decoder which consists of several linear layers (matrix multiplications). Finally, the decoder $\tilde{\omega}_r = D\left(z;\theta_D\right)$ maps back to the full space. An L_2 ("weight decay") regularization to the weights is also added, with prefactor λ. The loss function for this architecture is

$$\mathcal{L}\left(\tilde{\omega};\theta_E,\theta_W,\theta_D\right) = \left\langle \|\tilde{\omega} - D\left(\mathcal{W}\left(E\left(\tilde{\omega};\theta_E\right);\theta_W\right);\theta_D\right)\|_2^2 \right\rangle + \frac{\lambda}{2}\|\theta\|_2^2. \tag{3.10}$$

where $\langle\cdot\rangle$ is the average over a training batch, θ_E the weights of the encoder, θ_D the weights of the decoder, and θ_W the weights of the linear network. Zeng et al. [74] found dimension estimates for the KSE to be robust to the number of linear layers, λ, and d_z. However, as we will show, there is more variability when selecting these parameters for Kolmogorov flow. Nevertheless, we will see that IRMAE-WD yields a robust estimate of the upper bound of $d_{\mathcal{M}}$ that proves very useful.

After training, we can perform SVD on the covariance matrix of the encoded data matrix Z to obtain the singular vectors U and singular values S as shown in Fig. 3.7b. Projecting z onto $\hat{U}^T$ gives an orthogonal representation $h^+ = U^T z$ as illustrated in Fig. 3.7c. Then, by choosing only the singular values above some very small threshold (typically $\gtrsim 6$ orders of magnitude smaller than the leading singular values), we may project down to fewer dimensions by projecting onto the corresponding singular vectors U, denoted $\hat{U}$ to yield the low-rank manifold representation $h = \hat{U}z$ (Fig. 3.7d). We refer to Table 3.2 for details on the architecture.

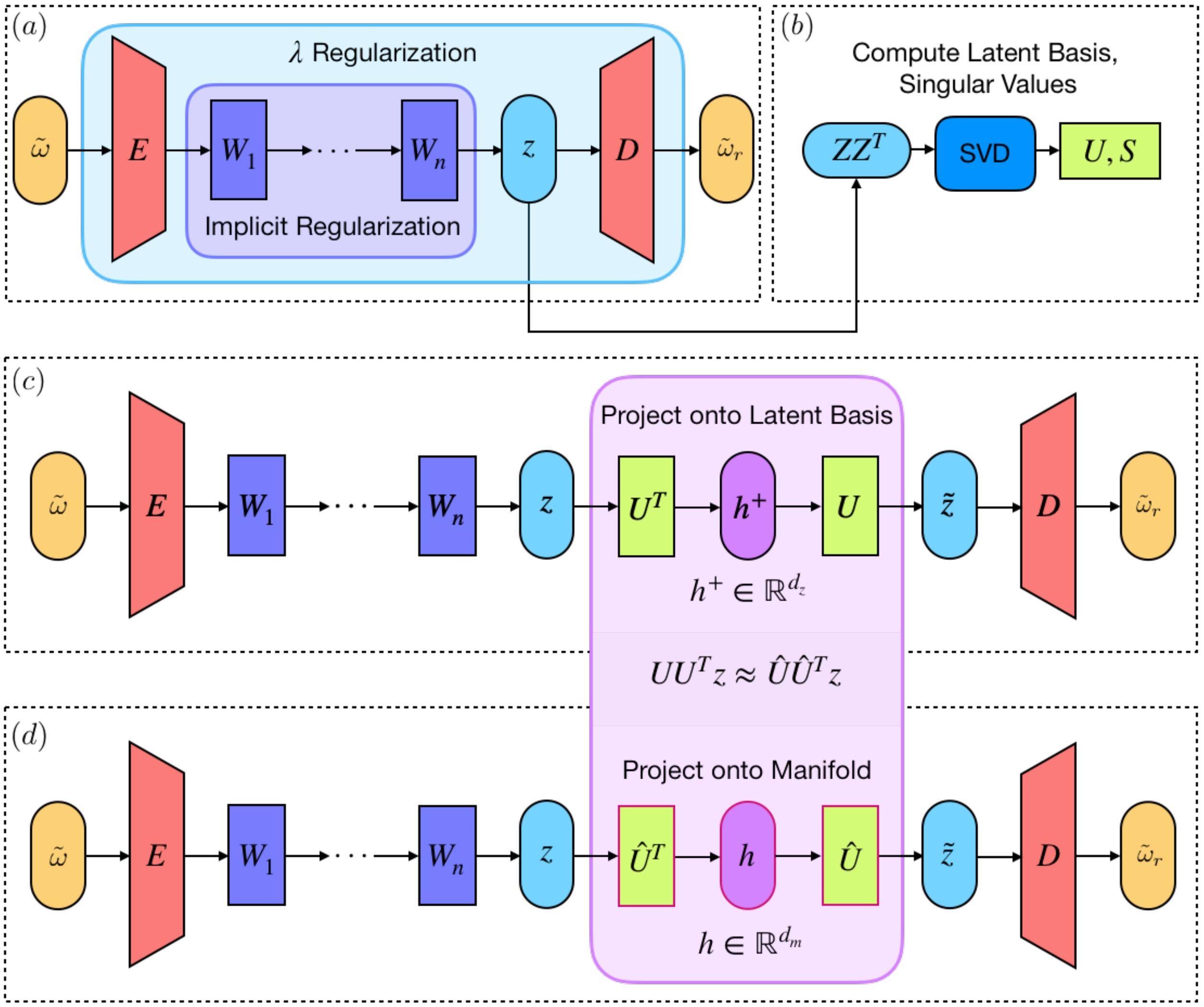


Figure 3.7: Implicit Rank Minimizing autoencoder with weight decay (IRMAE-WD) framework: a) network architecture with regularization mechanisms, b) singular value decomposition of the covariance of the learned latent data representation Z, c) projection of latent variables onto manifold coordinates d) isolated projection of latent variables onto manifold coordinates. Image reproduced with permission from [74].

3.3.3 Time evolution of pattern with neural ODEs

In the previous two steps we first mapped our data to the fundamental chart, allowing us to represent the state with $\tilde{\omega}$, $\mathcal{I}$, and ϕ_x. Then we reduced the dimension of $\tilde{\omega}$ by mapping it to the manifold coordinates h. Next, we need to find a method to evolve h, $\mathcal{I}$, and ϕ through time. To forecast $h(t)$ in time we use the neural ODE (NODE) framework[14, 38, 40]. We use a stabilized version proposed by Linot et. al. [40] where the dynamics on the manifold in the fundamental chart are described by the equation

$$\frac{dh}{dt} = g_h(h) - Ah. \tag{3.11}$$

Here A is chosen to have a stabilizing effect that keeps solutions from blowing up. We can define this parameter as $A = \text{diag}(\kappa[\text{std}(h)])$ where $\kappa = 0.1$ is multiplied by the element-wise standard deviation of h. The value of κ can be changed, but we see good prediction for the selected value. The equation is integrated to estimate $h_r(t+\tau)$ as

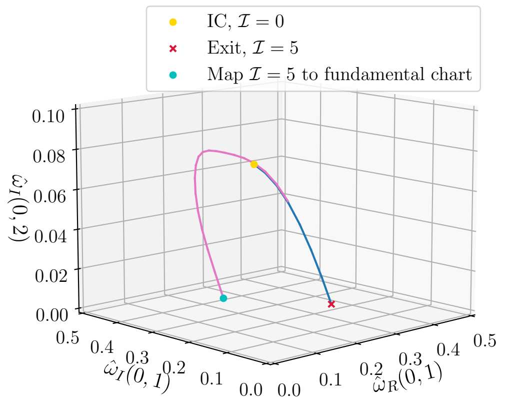
$$h_r\left(t+\tau\right) = h\left(t\right) + \int_t^{t+\tau} (g_h\left(h(t'); \theta_t\right) - Ah) dt' \tag{3.12}$$

where θ_t are the weights of the NN g_h, refer to Table 3.2, which are determined by minimizing the loss

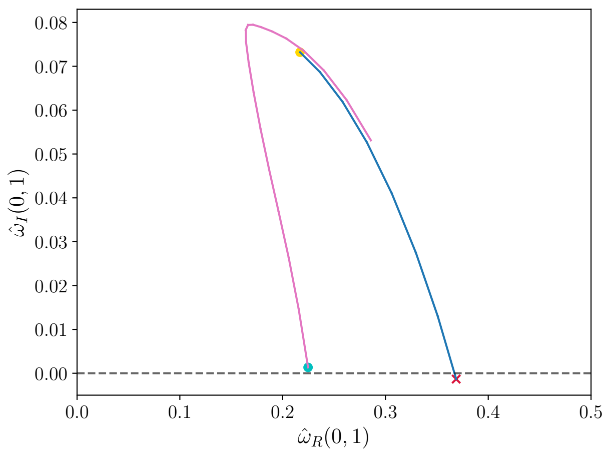
$$\mathcal{L}_{\text{NODE}}\left(h; \theta_t\right) = \left\langle \|h(t+\tau) - h_r(t+\tau)\|_2^2 \right\rangle. \tag{3.13}$$

To train the NODE model we first gather $\omega(t)$ and $\omega(t+\tau)$. We then map to the fundamental chart in pairs. As an example, if a snapshot $\omega(t)$ lies in $\mathcal{I} = 5$ we apply the corresponding discrete operations such that $h(t)$ falls in the interior of the fundamental chart $\mathcal{I} = 0$. The same discrete operations, which are $\mathscr{S}\mathscr{R}$ (refer to Table 3.1), are applied to $\omega(t+\tau)$. This means that $h(t+\tau)$ does not need to fall in the interior. We select $\tau = 0.5$ which is a small enough time such that the exterior region is covered by the autoencoder.

Now that we trained the NODE models we want to evolve trajectories in time. To do



(a)



(b)

Figure 3.8: (a) Initial condition (IC) starting at $\mathcal{I}=0$ is evolved and exits into the chart $\mathcal{I}=5$. This exit is mapped back to $\mathcal{I}=0$ to keep evolving in the fundamental chart. (b) Two-dimensional projection showing the exit from the fundamental chart. Here we selected an IC near the RPO, which is the reason why this trajectory nearly closes on itself.

this we need to track the pattern h and the indicator $\mathcal{I}$ at every time. We first show an exit and entry in the state space representation in Fig. 3.8. The initial condition (IC) starts in $\mathcal{I}=0$ and exits through the bottom part of the domain. Looking at Fig. 3.4 we see that this corresponds to a transition from $\mathcal{I}=0$ to $\mathcal{I}=5$. Hence, to keep evolving in the fundamental chart we need to apply the corresponding discrete operations to map to $\mathcal{I}=0$ and keep track of the new indicator $\mathcal{I}=5$. Notice that we need to keep track of the new indicator to map back to the full space at the end. This can generalize to a longer trajectory in which case the indicator changes depending on where the trajectory leaves the fundamental chart. The transitions between indicators corresponding to the different charts are shown in Fig. 3.9. Here we see a graph representation of the connections between the 8 symmetry charts for a trajectory of 10^5 snapshots. The intensity of each connection is related to the probability of the trajectory transitioning to another chart. The four darker lines correspond to the shadowing of the RPOs and the lighter lines are related to the bursting events. Now that we know how the indicators change we can evolve initial conditions with the NODE models to obtain predicted trajectories in h.

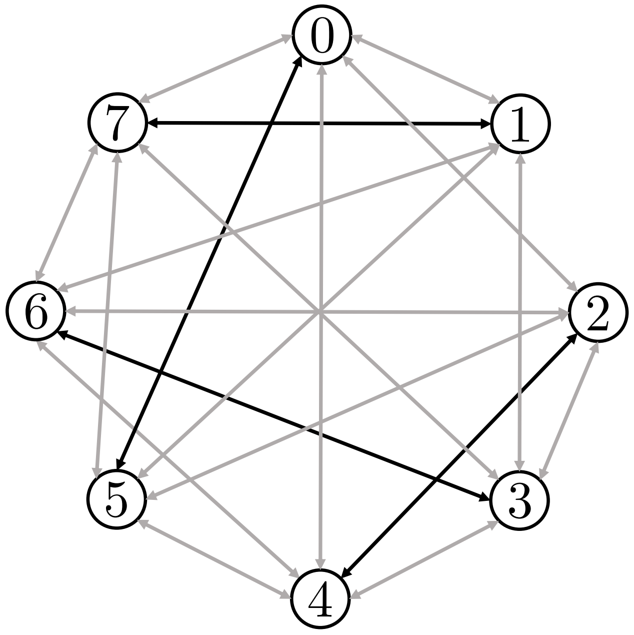


Figure 3.9: Graph representation of connections between the symmetry subspaces where pairs {2,4}, {0,5}, {6,3}, {7,1} correspond to the top and bottom sections of the octants where each unstable RPO lies. The intensity of each connection is related to the probability of the trajectory transitioning to another chart. The probability of staying in the same chart (not included in this depiction) is $\sim 94\%$, . Darker lines correspond to forward and reverse probabilities of $\sim 5\%$. Lighter lines correspond to forward and reverse probabilities of $\leq 1\%$.

Here we discuss the detailed operations for evolving trajectories in the fundamental chart with the NODE models. Fig. 3.10a summarizes the methodology for evolving in time. An initial condition in the fundamental space is first encoded and projected with $\hat{U}$ to obtain $h(t)$. Then the NODE maps a trajectory forward T time units to yield $h_{\text{temp}}(t+T)$ (note the indicator does not change in this step). We select $T = \tau$. After this, we perform the appropriate symmetry operations, detailed below, to find $h_r(t+T)$ and update the new indicator $\mathcal{I}(t+T)$. This is repeated to continue evolving forward in time. In Fig. 3.10b we present the method for performing these symmetry operations.

There are two main steps in applying the symmetry operations: 1) determining $\mathcal{I}$ from h_{temp}, and 2) getting the values of $h_r(t+T)$ and $\mathcal{I}(t+T)$. Step 1 is simply a classification problem where we wish to find a function that classifies new values of h_{temp} as either lying within or outside the fundamental domain. In Floryan & Graham [23] this was performed by identifying the nearest neighbor of the training data in the manifold coordinates and classifying h_{temp} with the same label. However, any classification technique can work – for example, support vector machines (SVM) [6] also worked well at this task. In the case of our specific problem, we found the fastest method was to map h_{temp} to the ambient space ω_{temp} at every step and calculate $\hat{\omega}_R(0,1)$, $\hat{\omega}_I(0,1)$, $\hat{\omega}_I(0,2)$ to verify if h_{temp} lies in the fundamental space. Step 2 involves updating the manifold coordinates to get $h_r(t+T)$. If the indicator changes, symmetries are factored out with $\mathcal{I}(t+T)$ and the snapshot is encoded to get the new $h_r(t+T)$. If the indicator stays the same $h_r(t+T) = h_{\text{temp}}(t+T)$ and $\mathcal{I}(t+T) = \mathcal{I}(t)$.

3.3.4 Time evolution of phase with neural ODEs

We also wish to time-evolve the phase $\phi_x(t)$. The dynamics of the phase only depend upon the fundamental-chart vorticity pattern h and the indicator $\mathcal{I}$, so we can describe them with

$$\frac{d\phi_x}{dt} = g_{\mathcal{I}}(\mathcal{I}) g_\phi(h;\theta_\phi). \tag{3.14}$$

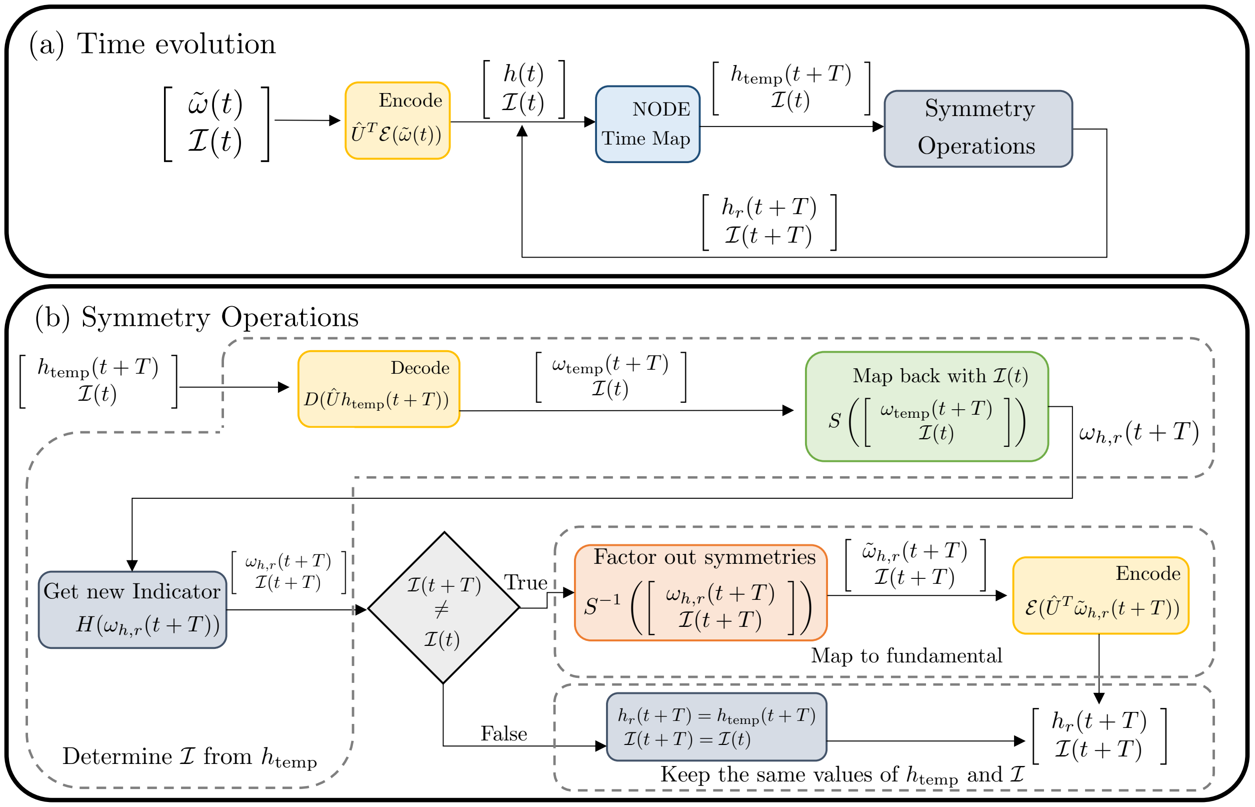

(a) Time evolution
$\begin{bmatrix}\tilde{\omega}(t)\\ \mathcal{I}(t)\end{bmatrix}$
Encode
$\hat{U}^T\mathcal{E}(\tilde{\omega}(t))$
$\begin{bmatrix}h(t)\\ \mathcal{I}(t)\end{bmatrix}$
NODE Time Map
$\begin{bmatrix}h_{\text{temp}}(t+T)\\ \mathcal{I}(t)\end{bmatrix}$
Symmetry Operations
$\begin{bmatrix}h_r(t+T)\\ \mathcal{I}(t+T)\end{bmatrix}$
(b) Symmetry Operations
$\begin{bmatrix}h_{\text{temp}}(t+T)\\ \mathcal{I}(t)\end{bmatrix}$
Decode
$D(\hat{U}h_{\text{temp}}(t+T))$
$\begin{bmatrix}\omega_{\text{temp}}(t+T)\\ \mathcal{I}(t)\end{bmatrix}$
Map back with $\mathcal{I}(t)$
$S\left(\begin{bmatrix}\omega_{\text{temp}}(t+T)\\ \mathcal{I}(t)\end{bmatrix}\right)$
$\omega_{h,r}(t+T)$
Get new Indicator
$H(\omega_{h,r}(t+T))$
$\begin{bmatrix}\omega_{h,r}(t+T)\\ \mathcal{I}(t+T)\end{bmatrix}$
$\mathcal{I}(t+T) \neq \mathcal{I}(t)$
True
Factor out symmetries
$S^{-1}\left(\begin{bmatrix}\omega_{h,r}(t+T)\\ \mathcal{I}(t+T)\end{bmatrix}\right)$
$\begin{bmatrix}\tilde{\omega}_{h,r}(t+T)\\ \mathcal{I}(t+T)\end{bmatrix}$
Encode
$\mathcal{E}(\hat{U}^T\tilde{\omega}_{h,r}(t+T))$
Map to fundamental
False
$h_r(t+T) = h_{\text{temp}}(t+T)$
$\mathcal{I}(t+T) = \mathcal{I}(t)$
$\begin{bmatrix}h_r(t+T)\\ \mathcal{I}(t+T)\end{bmatrix}$
Keep the same values of h_{temp} and $\mathcal{I}$
Determine $\mathcal{I}$ from h_{temp}

Figure 3.10: (a) Time evolution of $h(t)$ with NODE and (b) symmetry operation check to map back to fundamental space if needed.

The term $g_{\mathcal{I}}(\mathcal{I})$ equals 1 if an even number of discrete operations map h back to the fundamental space, or -1 if an odd number of discrete operations map h back to the fundamental space. To understand how the signs of Equation 3.14 change consider the effect of discrete symmetry operations on the phase calculation $\phi_x = \operatorname{atan2}\{\hat{\omega}_I(1,0), \hat{\omega}_R(1,0)\}$. Applying a discrete symmetry operation on a snapshot changes the phase variable to either

$$\phi_{x,\mathscr{S}} = \mathscr{S}\phi_x = \operatorname{atan2}\{\hat{\omega}_I(1,0), -\hat{\omega}_R(1,0)\} = \pi - \phi_x, \tag{3.15}$$

or

$$\phi_{x,\mathscr{R}} = \mathscr{R}\phi_x = \operatorname{atan2}\{-\hat{\omega}_I(1,0), \hat{\omega}_R(1,0)\} = -\phi_x. \tag{3.16}$$

Then, by simply taking the time derivative, we see that $d\phi_{x,\mathscr{S}}/dt = -d\phi_x/dt$ and $d\phi_{x,\mathscr{R}}/dt = -d\phi_x/dt$. Hence the operation of a discrete symmetry (rotation and shift-reflect) changes the sign. Now, we train g_ϕ by fixing the NODE g_h to evolve h forward in time, and use Equation 3.14 to make predictions $\phi_{x,r}$. We update the parameters of g_ϕ to minimize the loss

$$\mathcal{L}_{\phi,\mathrm{NODE}}(\phi_x;\theta_\phi) = \left\langle \|\phi_x(t+\tau) - \phi_{x,r}(t+\tau)\|_2^2 \right\rangle, \tag{3.17}$$

and refer to Table 3.2 for details on the NODE architecture.

Table 3.2: Neural network layer dimensions and activations used in each layer. Sigmoid functions are denoted 'S'. Number of linear layers W_i can be varied.

	Function	Shape	Activation
Encoder	E	1024 : 2048 : 256 : 40	ReLU:ReLU:Linear
Decoder	D	40 : 256 : 2048 : 1024	ReLU:ReLU:Linear
Linear Net	$\mathcal{W} = W_1 W_2 ...$	40 : 40	Linear
Pattern NODE	g_h	d_m : 500 : 500 : 500 : d_m	S:S:S:Linear
Phase NODE	g_ϕ	d_m : 500 : 500 : 500 : 1	S:S:S:Linear

3.4 Results

We present results for the chaotic case Re $= 14.4$, whose trajectories sample all eight charts introduced above. First we illustrate the effect of training data size on the autoencoder performance when considering the fundamental chart data, as opposed to using the original and phase-aligned data. These results are shown in Section 3.4.1. We then use IRMAE-WD to estimate the minimal dimension needed to represent the data. We summarize these results in Section 3.4.1. The time evolution model performance results are shown in Section 3.4.2. Here we confirm the equivariance of the model, (Section 3.4.2) then show the performance for short-time tracking, long-time statistics, and phase evolution (Sections 3.4.2, 3.4.2, and 3.4.2 respectively).

3.4.1 Dimension reduction with IRMAE-WD

To train our IRMAE-WD models we minimize $\mathcal{L}$ via stochastic gradient descent. Following Zeng et al. [74], we use the AdamW optimizer, which decouples weight decay from the adaptive gradient update and helps avoid the issue of weights with larger gradient amplitudes being regularized disproportionately, as observed in Adam [42]. All models were trained for a total of 1000 epochs and with a learning rate of 10^{-3}. We consider 10^5 snapshots of the original data (ω), the phase-aligned data (ω_l), and the data mapped to the fundamental space ($\tilde{\omega}$) separated by $\tau = 0.5$ time units.

Before training the AEs, the mean is subtracted and the data is divided by its standard deviation. Three models are trained for each case of varying number of linear layers $L = 0, 4, 6, 8, 10$ and weight decay values of $\lambda = 10^{-4}, 10^{-6}, 10^{-8}$ for a total of 45 models for the ω, ω_l, and $\tilde{\omega}$ cases. For all of the networks discussed in this work, we use a 80/20 split of the data for training and testing respectively. We select $d_z = 40$, which is significantly higher than the dimension expected based on our previous work [53].

Effect of training data size on performance

A major benefit of mapping our data to the fundamental chart is that it results in eightfold denser sampling within that chart, as shown in Fig. 3.6. We also see that all the data is in a much smaller part of state space, and only that part of state space needs to be represented. As such, in this section, we test if this increased density allows us to use less data for training AEs. We factor out the symmetries of our system as discussed in Section 3.3.1 and train IRMAE-WD models for the case of $L = 4$ and $\lambda = 10^{-4}$. In Fig. 3.11 we show the test MSE of the IRMAE-WD models as we vary the amount of data for three cases: the original data (ω), the phase-aligned data (ω_l), and the data mapped to the fundamental space ($\tilde{\omega}$). In all cases, we use the same $20,000$ test snapshots to calculate the MSE, train 3 models at each training data size, and reduce to a dimension of $d_z = 40$. Notice that here we do not project to a lower dimension; that will be done below. The purpose here is not to map to the minimal dimension, but to study the effect of data size on training.

As can be seen in Fig. 3.11, removing phase results in an order of magnitude improvement in the MSE over the original data, and mapping to the fundamental results in nearly another order of magnitude improvement in the MSE over the phase-aligned data. Thus, this drastic improvement in performance allows us to use far less data. For example, 800 snapshots in the fundamental space performs nearly as well as $80,000$ snapshots in the original space. Similarly, $8,000$ snapshots in the fundamental case outperforms $80,000$ snapshots in the phase-aligned case. Now that we have shown the effect of the number of snapshots on performance we use $80,000$ snapshots for the remaining studies.

Effect on dimension estimates with varying linear layers and weight decay

Here we study the effects on the dimension estimates when varying L and λ for the original, phase-aligned, and fundamental case. As discussed in Section 3.3.2 we perform SVD on the covariance matrix of the encoded data ZZ^T to obtain U and truncate to obtain $h = \hat{U}^T z$. Zeng et al. [74] showed that structuring an autoencoder with linear layers and using weight

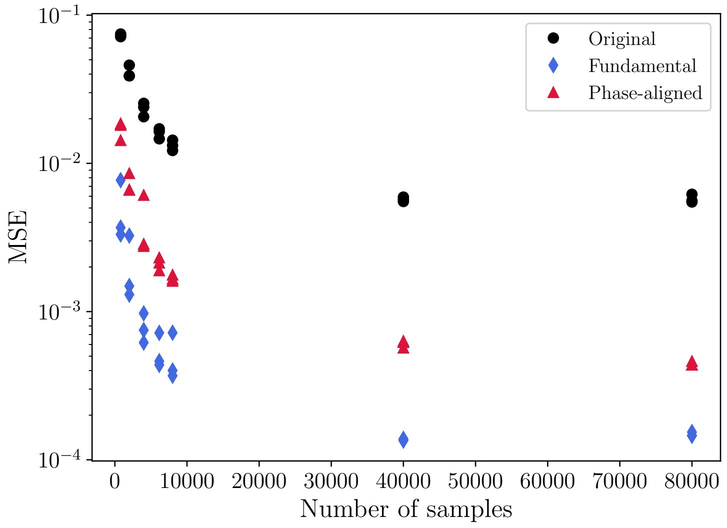


Figure 3.11: MSE vs number of data size used for training IRMAE-WD corresponding to a test data of 20,000 samples for Re = 14.4. The parameters used are $L = 4$ and $\lambda = 10^{-4}$ and three trials are considered for each case. The data is reduced to a dimension of $d_z = 40$.

decay causes the latent space to become low-dimensional through training. In Fig. 3.12, we show the evolution of these singular values through training. As the model trains for longer times, the trailing singular values tend towards zero. These can be truncated without any loss of information. For most cases, this drop is drastic ($\sim$ 10 orders of magnitude) and a threshold can be defined to select how many singular values to keep (i.e. to select the number of dimensions).

In Fig. 3.13 we plot the estimate of dimension for the original (black), phase-aligned (red), and fundamental chart (blue) data as we vary the number of linear layers L and the weight decay parameter λ. For each case, there are two clear clusters – the cluster with higher dimension corresponds to $L = 0$ and the other cluster has linear layers, $L > 0$.This happens because in the absence of linear layers there is no mechanism to drive the rank to a minimal value. When we factor out symmetries the dimension estimates become less spread with ranges of $d_h = 7 - 10$ for fundamental, $d_h = 8 - 13$ for phase-aligned, and $d_h = 11 - 18$ for original. This narrowing of the distribution likely happens due to the dense coverage of the state space in the fundamental chart, which better captures the shape of the manifold. The dimension estimate range from the fundamental chart is in good agreement with our

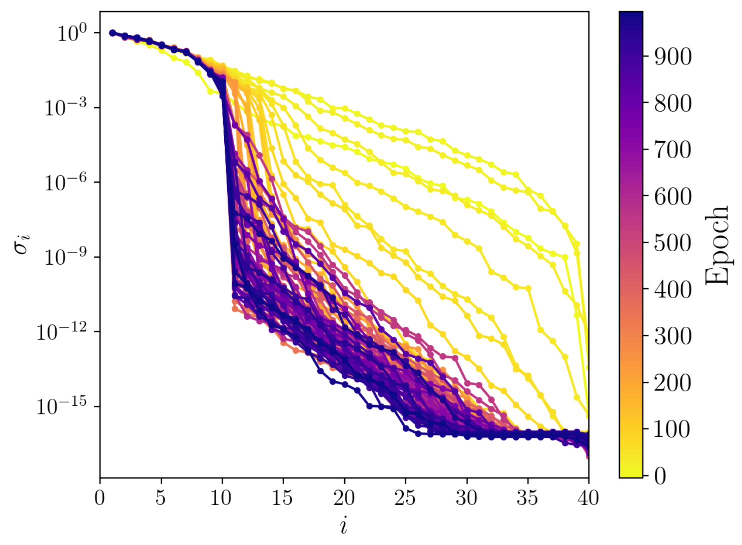


Figure 3.12: Evolution of singular values of the covariance matrix of the encoded test data ZZ^T during training of an IRMAE-WD model with $L = 4$ and $\lambda = 10^{-4}$. Here the drop happens at $d_h = 10$.

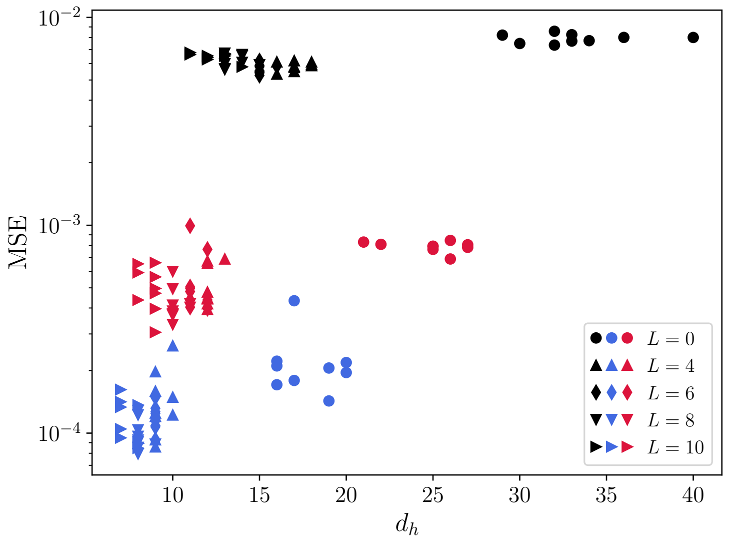


Figure 3.13: MSE vs dimension d_h given by the spectral gap of the singular values for the original (black), phase-aligned (red), and fundamental chart (blue) cases. Each case contains three trials of combinations of parameters $L = 0, 4, 6, 8, 10$ and $\lambda = 10^{-4}, 10^{-6}, 10^{-8}$.

previous observations of $d_h = 9$ in [53]. In the following analysis we select a conservative estimate of the dimension which appears to be $d_h = 10$ from the fundamental chart data.

3.4.2 Time evolution

To learn our NODE models we first train g_h by minimizing Equation 3.13 ($\mathcal{L}_{\text{NODE}}$) and then we fix g_h and train g_ϕ by minimizing Equation 3.14 ($\mathcal{L}_{\phi,\text{NODE}}$) via the Adam optimizer. We train for a total of 40000 epochs and a learning rate scheduler that drops from 10^{-3} to 10^{-4} at epoch number 13334 (1/3 into training) and from 10^{-4} to 10^{-5} at epoch number 26667 (2/3 into training). As previously discussed we use $\tau = 0.5$ which ensures the trajectory spends some time steps in an overlap region as it moves from chart to chart, so we can learn the dynamics there.

Equivariance

The results we obtain with this framework should be equivariant with respect to initial conditions. This means that after we apply any of the symmetry operations described above to an initial condition, the resulting trajectory from the original initial condition and the new initial condition must be equivalent up to this symmetry operation. Here we show that our methodology retains equivariance. We select the IRMAE-WD models with the lowest MSEs at a dimension of $d_h = 10$, which is a conservative dimension estimates for the fundamental case, for both the phase-aligned and fundamental model. We then sample 1000 initial conditions separated by 10 time units such that we cover different regions of state space. For every initial condition, we apply all the discrete symmetry operations in the original Fourier space, mapping the data into every octant. Then we evolve these initial conditions forward 1000 time units with the DManD model, with and without symmetry charting. To test for equivariance, we compute the trajectory error TE between predicted

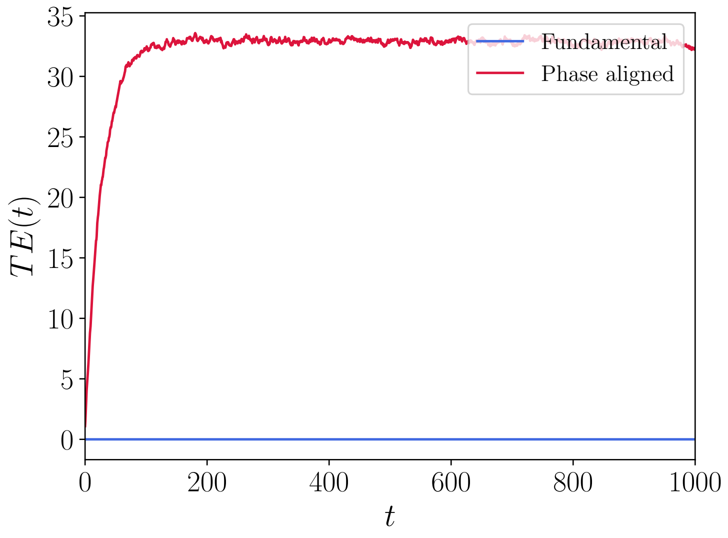


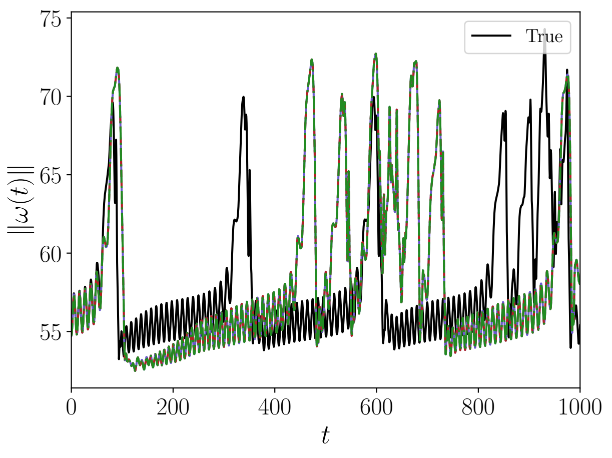
Figure 3.14: $TE(t)$ vs t for Fundamental and Phase-aligned case where $TE(t)$ corresponds to the error calculation between predicted NODE trajectories, from a model with $d_h = 10$, from initial conditions that are related by symmetry transformations.

trajectories as follows

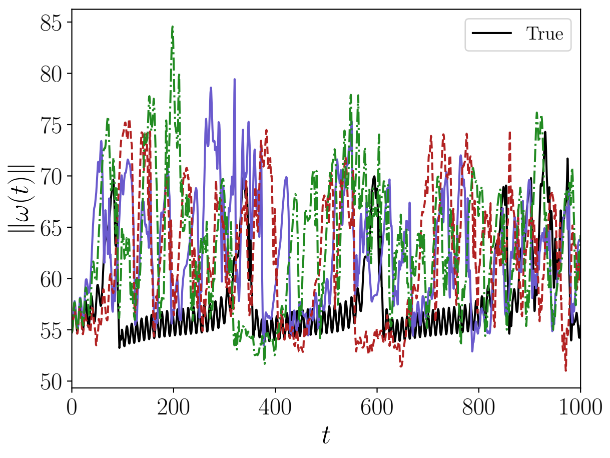
$$TE(t) = \frac{1}{100 \times 7} \sum_{i=1}^{100} \sum_{j=1}^{7} \|\tilde{\omega}_{i,j}(t) - \tilde{\omega}_{i,j=0}(t)\|, \tag{3.18}$$

where i corresponds to the trajectory number, and j to the initial chart (i.e. $j = 0$ corresponds to the original initial condition of the ith trajectory). Fig. 3.14 presents this error through time for our symmetry charting method (Fundamental) and for models with only phase-alignment. As expected, the time integration from the model trained in the fundamental space satisfies equivariance perfectly with a $TE = 0$. The phase-aligned curve does not satisfy equivariance, and we see that trajectories diverge fairly quickly.

A more severe consequence of not enforcing the discrete symmetries can be seen in trajectory predictions. Fig. 3.15 depicts $\|\omega(t)\|$ vs t for the true data (shown in black) and the time integration of initial conditions starting in different charts (red – $\mathcal{I} = 0$, blue – $\mathcal{I} = 1$, and green – $\mathcal{I} = 2$) for both fundamental (a) and phase-aligned cases (b). In all cases, the predicted trajectories diverge from the true trajectory after some time – this is expected for a chaotic system, as explore further in Section 3.4.2. However, the symmetry-charted



(a)



(b)

Figure 3.15: $\|\omega(t)\|$ vs time for initial conditions integrated with the (a) Fundamental and (b) Phase aligned NODE model for $d_h = 10$. Colors correspond to initial conditions starting in different fundamental domains (red – $\mathcal{I} = 0$, blue – $\mathcal{I} = 1$, and green – $\mathcal{I} = 2$). In (a), all the different-colored trajectories coincide.

predictions in Fig. 3.15a exhibit excellent short-time tracking and captures the bursting event that happens around $t = 100$. At longer times, the prediction is not quantitatively accurate but still captures the alternation between quiescent and bursting intervals observed in the true data. By contrast, the predictions for the phase-aligned model, Fig. 3.15b, deviate quickly from the true data, and furthermore do not even exhibit intermittency between quiescent and bursting dynamics – they stay in a bursting regime. Thus the models that do not account for the discrete symmetries do not capture the dynamics correctly, even at a qualitative level. These results reinforce the major advantage of properly accounting for symmetries, as the symmetry charting method does.

Short-time predictions

In this section, we focus on short-time trajectory predictions. The Lyapunov time t_L for this system is approximately $t_L \approx 20$ [29]. We take initial conditions of $h(t)$ and evolve for 100 time units. These are then decoded and compared with the true vorticity snapshots. We consider trajectories with initial conditions starting in the quiescent as well as in the bursting regions. The dynamics at $\mathrm{Re} = 14.4$ are characterized by quiescent intervals where

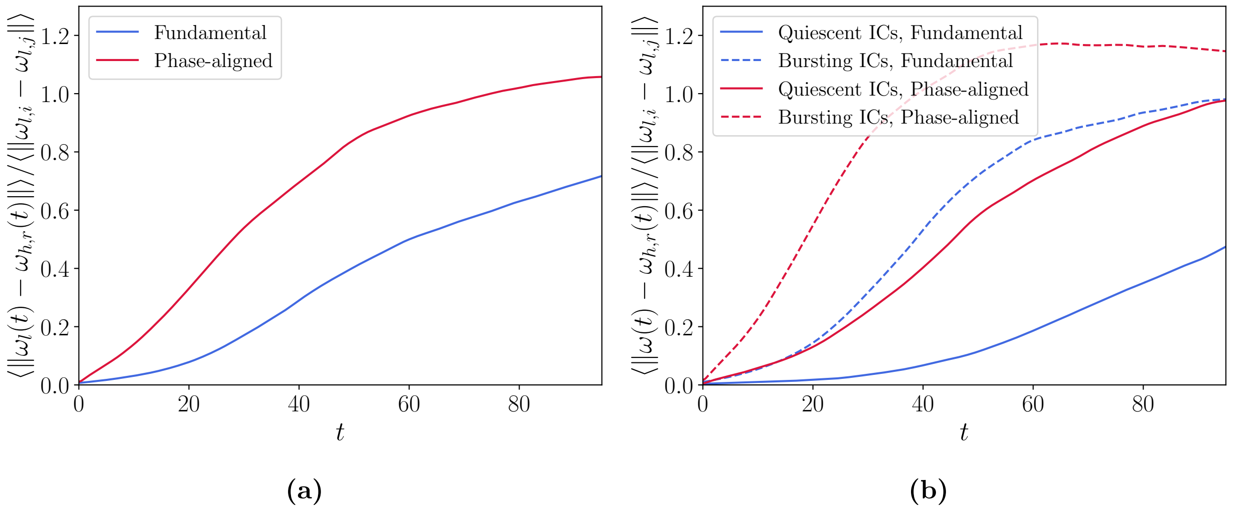


Figure 3.16: Difference between true vorticity evolution and vorticity evolution obtained from the Fundamental and Phase aligned NODEs from $h(t)$ of $d_h = 10$, where (a) corresponds to averages over all initial conditions, and (b) corresponds to averages taken over bursting and quiescent initial conditions.

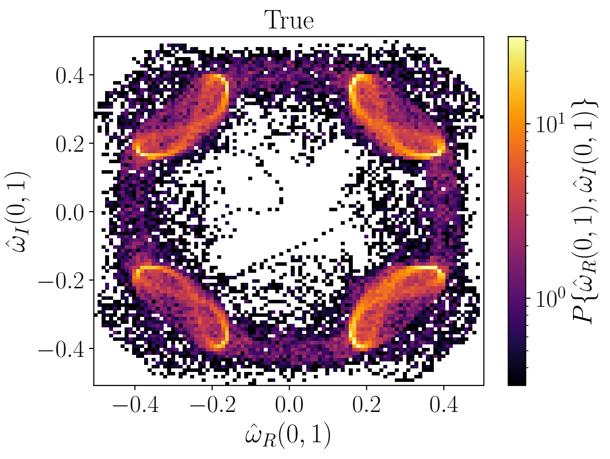
the trajectories are close to RPOs (which are now unstable), punctuated by heteroclinic-like excursions (bursting) between the RPOs, which are indicated by the intermittent increases of $\|\omega(t)\|$ as observed in Fig. 3.3a. The nature of the intermittency of the data makes it challenging to assign either bursting or quiescent labels. To split the initial conditions as quiescent or bursting we use the algorithm discussed in [53].

We first show in Fig. 3.16a the ensemble-averaged prediction error as a function of time for 10^4 initial conditions. We use the same models as in the previous section which corresponds to $d_h = 10$ for both fundamental and phase-aligned case. The comparison is done with true phase-aligned data, so after obtaining the prediction from the fundamental case we use the indicators to include the symmetries. The error is normalized with random differences of the true data, where i and j correspond to different snapshots. With this normalization, when the curves approach 1 this means that on average the distance between the model and the true data is the same as if we selected random points from the true data. The DManD models using symmetry charting significantly outperform the phase-aligned DManD models. This agrees with Fig. 3.15 as discussed above. Similar improvement is observed in Fig. 3.16b, and can be attributed to the organized (near-RPO) nature of the dynamics in the quiescent region. Also, the dynamics spend more time in this area, so there is more data for the autoencoder to train on.

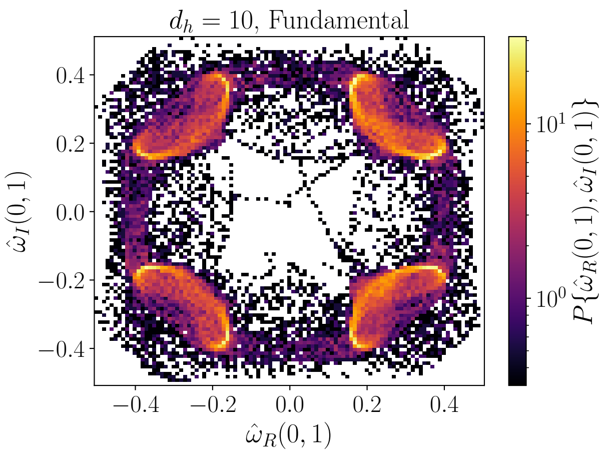
Long-time predictions

Now that we demonstrated the short-time predictive capabilities of the model, we next turn to the ability of the model to reconstruct the long-time statistics of the attractor. For this comparison, we sample 2×10^4 time units of data every 0.5 time units for the DNS and the DManD models. Fig. 3.17 shows the joint probability density function (PDF) of $\mathrm{Re}[a_{0,1}(t)]$ and $\mathrm{Im}[a_{0,1}(t)]$ for true and predicted data from the models with $d_h = 10$ for the fundamental and phase-aligned case. The true joint PDF (Fig. 3.17a) and the symmetry charting fundamental joint PDF (Fig. 3.17b) are in excellent agreement. These PDFs both show a strong, equal preference for trajectories to shadow the four unstable RPOs, and lower probabilities between them where the bursting transitions of the RPO regions occur. In contrast, the joint PDF for the phase-aligned model (Fig. 3.17c) shows that the model samples the space before falling onto an unphysical stable RPO (bright closed curve) near one of the true system's unstable RPOs. Clearly, in this case, the phase-aligned model fails to capture the system's true dynamics.

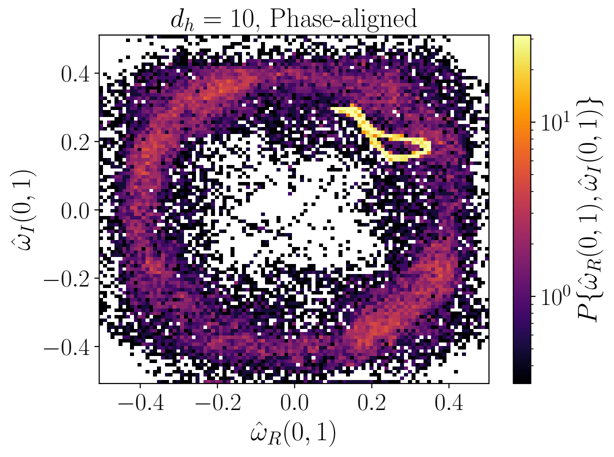
Another important quantity to consider is the ability of the models to capture the energy balance of the system. In Fig. 3.18 we show the joint PDF of I and D for the DNS and the same models. Again, the model trained in the fundamental space closely matches the joint PDF of the true data. The phase-aligned model both underestimates the energy associated with the high probability RPOs, and overestimates the energy associated with the low probability high power input and dissipation events.



(a)

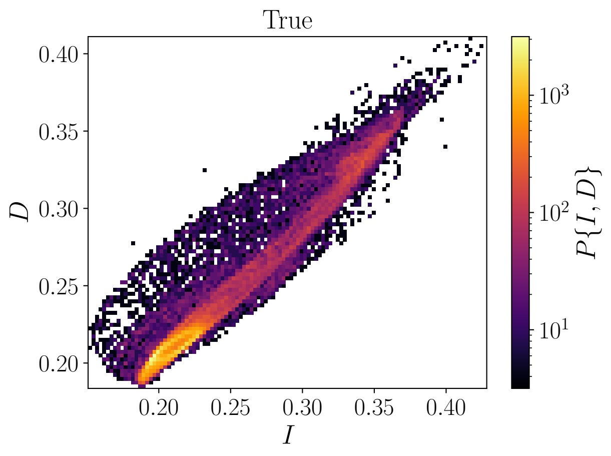


(b)

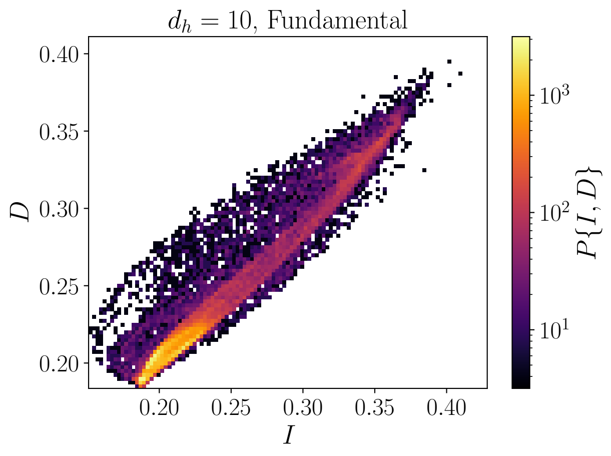


(c)

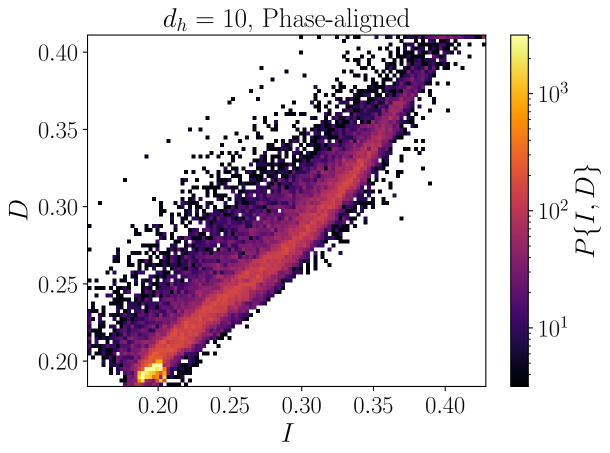
Figure 3.17: Joint PDFs of $\hat{\omega}_R(0,1)$-$\hat{\omega}_I(0,1)$ of (a) true, and predicted data corresponding to dimension $d_h = 10$ from the (b) Fundamental and (c) Phase-aligned models.



(a)



(b)



(c)

Figure 3.18: Joint PDFs of I-D of (a) true, and predicted data corresponding to dimension $d_h = 10$ from the (b) Fundamental and (c) Phase-aligned models.

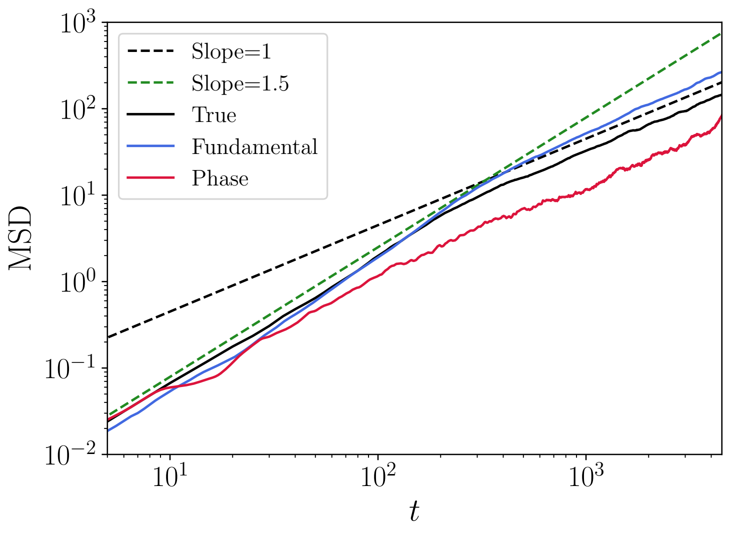


Figure 3.19: MSD of $\phi_x(t)$ corresponding to models with dimension $d_h = 10$.

Phase variable prediction

To complete the dynamical picture we predict the phase evolution as given by Equation 3.14. We compare the models to the true data by calculating the mean squared displacement (MSD) of the phase,

$$\text{MSD}(t) = \langle(\phi_x(t) - \phi_x(0))^2\rangle \tag{3.19}$$

as was done in our previous work [53]. Due to the bursting region, where the direction of phase evolution is essentially randomly reset, at long times the phase ϕ_x exhibits random-walk behavior, which the MSD reflects. We take 420 initial conditions separated by 15 time units and use the models of $d_h = 10$ to predict $\phi_x(t)$ and calculate the MSD. This is done for the fundamental and phase-aligned models.

Fig. 3.19 shows the evolution of MSD of true and predicted data. Here the black solid line corresponds to the true data, and the black and green dashed lines serve as references with slopes of 1 and 1.5 respectively. There is a change from superdiffusive ($MSD \sim t^{1.5}$) to diffusive ($MSD \sim t$) scaling around $t \approx 200$, which corresponds to the mean duration of the quiescent intervals: i.e., to the average time the trajectories travel along the RPOs before bursting. The fundamental chart model accurately captures both the change in slope and the timing of this change in slope. However, the curve of the phase-aligned model fails to match the true MSD at long times. This happens because the phase-aligned model is unable to predict $h(t)$ correctly at long times as observed in Fig. 3.15b, hence resulting in poor predictions for $\phi_x(t)$.

3.5 Summary

Symmetries appear naturally in dynamical systems and we show here that correctly accounting for them dramatically improves the performance of data-driven models for time evolution on an invariant manifold. In this work, we introduce a method that we call symmetry charting and apply it to Kolmogorov flow in a chaotic regime of $\mathrm{Re} = 14.4$. This symmetry charting method factors out symmetries so we can train ROMs in a fundamental chart and ensure equivariant trajectories – we are essentially learning coordinates and dynamics on one region (chart) of the invariant manifold for the long-time dynamics rather than having to learn these for the entire manifold. To do so, we first factor out the symmetries by identifying a set of indicators that differentiate the set of discrete symmetries for the system. Here, Fourier coefficients serve this purpose; we found that the signs of specific Fourier modes could uniquely identify all discrete symmetry operations. We then use IRMAE-WD, an autoencoder architecture that tends to drive the rank of the latent space covariance of the data to a minimum, to find a low-dimensional representation of the data. This method overcomes the need to sweep over latent space dimensions. We observe that factoring out symmetries improves the MSE of the reconstruction as well as the dimension estimates and robustness to IRMAE-WD hyperparameters. When considering the original data (i.e. no symmetries factored out) and the phase-aligned data (only continuous symmetry factored out) the MSE is higher and the range of dimension estimates is wider. We also note that the dimension estimate range in the fundamental chart agrees well with the dimension found in our previous work for the same system [53]. We then train a NODE with latent space of dimension

$d_h = 10$ for both phase-aligned and fundamental space data. This dimension is the upper bound of the dimension estimate obtained for the fundamental chart data. The resulting models using symmetry charting to map to the fundamental space accurately reconstructed the DNS at both short and long times. In contrast, the phase-aligned model quickly landed on an unphysical stable RPO leading to poor reconstruction of the dynamics.

The approach described here is essentially a version of the “CANDyMan" (Charts and Atlases for Nonlinear Data-driven Dynamics on Manifolds) approach described in [23, 25]. As shown on those studies, learning charts and atlases to develop *local* manifold representations and dynamical models can improve performance for systems with complex dynamics. There, however, the charts were found by clustering; here that step is bypassed by using a priori knowledge, about the system, namely its symmetries. The methodology presented in this work can be applied to other dynamical systems with rich symmetries by identifying the corresponding indicators. Future directions for symmetry charting and CANDyMan include applications such as control in systems with symmetry (cf. [72]) as well as development of hierarchical methods where the fundamental chart identified by symmetry can be further subdivided using a cluster algorithm or a chart autoencoder [57].

4

Improving robustness of dimension estimates with implicit rank minimizing autoencoders

The manifold dimension is not known for many complex systems and its estimating is a nontrivial task. This manifold will usually have many fewer degrees of freedom than the full state representation. For fluid flows, such minimal dimensional representations are needed to creat invariant-manifold-based data-driven models which can result in decreased computational cost for simulations and be used for model-based control approaches. In this work we use a variation of implicit rank minimizing autoencoders with weight decay (IRMAE-WD) to improve robustness of dimension estimates for a library of dynamical systems. We introduce a set of paths, or branches, that the autoencoder can take in the forward pass step. These branches vary by the number of linear layers used after the encoding part. We call this architecture IRMAE-WD-B, where B stands for branching. With IRMAE-WD-B we avoid the need to define a number of linear layers, resulting in less models trained, and observe that dimension estimates become more consistent for complex dynamical systems.

4.1 Introduction

The Navier-Stokes Equations (NSE) are dissipative PDEs, so it is expected that the long-time dynamics will lie on an invariant manifold $\mathcal{M}$, which can be represented *locally* with Cartesian coordinates, but may have a complex global topology [28]. In dissipative PDEs, this manifold is often called an *inertial manifold* [24, 62, 71]. This manifold is of many fewer degrees of freedom $d_{\mathcal{M}}$ and will be embeded on an ambient space $\mathbb{R}^N$ where often $d_{\mathcal{M}} \ll N$. Then for the representation, we only need at least $d_{\mathcal{M}}$ degrees of freedom. However, this is in the local sense, meaning that no global representation of dimension $d_{\mathcal{M}}$ is available. A representation of dimension $d_{\mathcal{M}}$ can be found by "cutting" the space into overlapping charts which together form at atlas that covers the space [23]. Alternatively a global representation with *embedding dimension* $d_e \leq 2d_{\mathcal{M}}$ can be guaranteed by Whitney's theorem [36]. In this work we aim to estimate manifold dimensions for different dynamical systems with the use of autoencoders (AE) given access to only data. We will show how the addition of paths in the AE with different amounts of linear layers improves robustness of dimension estimates.

Different methods to estimate manifold dimensions have been introduced in literature. For the Kuramoto-Sivashinsky equation (KSE) the manifold dimension has been estimated using covariant Lyapunov vectors and monitoring the drop of the Lyapunov spectrum [69, 70]. A Floquet mode approach has also been used for this system to estimate the dimension, which involves knowledge of unstable periodic orbits [19]. These methods require access to the governing equations and high-precision solutions which is generally not available for the NSE. On the other hand, data can also be used to estimate manifold dimensions. This is the case of linear projection methods such as Principal Component Analysis (PCA) and variants of it [5, 33, 75]. PCA determines a set of orthogonal basis vectors ordered by their contribution to the total variance (fluctuating kinetic energy) of the flow. Given N_s data vectors (“snapshots") $x_i \in \mathbb{R}^N$, one can obtain these basis vectors by performing singular value decomposition (SVD) on the data matrix $X = [x_1, x_2, \cdots] \in \mathbb{R}^{N \times N_s}$ such that $X = U\Sigma V^T$. Projecting the data onto the first d_h basis vectors (columns of U) then gives a low-dimensional representation – a projection onto a linear subspace of the full state space. However, these low-dimensional representations can overestimate the dimension of the manifold where at least $d_{\mathcal{M}} + 1$ dimensions will be required to represent the state.

Nonlinear methods are natural extensions to correctly capture these manifold dimensions. For more details on different methods, we refer to Zeng et al. [74]. Here we focus on the use of AEs to learn these low-dimensional representations. Neural networks (NNs) have been used extensively to learn data-driven minimal dimensional models that can capture important features of dynamical systems. Linot & Graham showed that for the KSE, tracking the mean-squared error (MSE) of the reconstruction of the snapshots using an AE for the domain size of $L = 22$ exhibited several orders-of-magnitude drop when the dimension of the inertial manifold is reached. Furthermore, modeling the dynamics with a dense NN at this dimension either with a discrete time map [37] or a system of ordinary differential equations (ODE) [38] yields excellent trajectory predictions and long-time statistics. However by increasing the domain size to $L = 44$ and $L = 66$, which makes the system more chaotic, they observed that this MSE drop is affected significantly. Pérez De Jesús & Graham extended this approach to two-dimensional Kolmogorov Flow in a chaotic regime where they noticed that this drop in MSE does not appear [53]. Nevertheless, modeling the dynamics at dimensions where the mean-squared error (MSE) stopped decreasing ($d_h = 5$ as opposed to $N = 1024$) showed that short and long time statistics can be captured as well as prediction of bursting events. Similarly Linot & Graham found low-dimensional representations of minimal flow unit turbulent planar Couette flow at $\mathrm{Re} = 400$ where the MSEs leveled off at a dimension of $d_h = 18$ [39]. They then learned minimal-dimensional models showing that these models were able to capture characteristics of the flows such as streak breakdown and regeneration, short-time tracking, as well as Reynolds stresses and energy balance. They also computed unstable periodic orbits from the models with close resemblance to previously computed orbits from the full system. The studies described above show how minimal-dimensional models can be successfully learned with AEs. However, the question of estimating the inertial manifold dimension still remains open. Recently, a variation of a standard AE was introduced by Jing et al. [32] to learn low-rank representations for image-based classification and generative problems. Zeng et al. [74] showed that this architecture can yield robust and precise estimates of $d_{\mathcal{M}}$, as well as an orthogonal manifold coordinate system. The architecture they study is called an Implicit Rank Minimizing Autoencoder with weight decay (IRMAE-WD), and involves inserting a series of linear layers between the encoder and decoder and adding an L_2 regularization on the neural network weights in the loss. The effect of these additions is an AE for which the standard gradient descent algorithm for learning NN weights drives the rank of the covariance of the data in the latent representation to a minimum while maintaining representational capability. Applying this to the KSE and other systems resulted in the rank being equal to the dimension of the inertial manifold $d_{\mathcal{M}}$. Pérez De Jesús et. al. [54] applied IRMAE-WD to Kolmogorov flow in a fundamental representation, where symmetries are factored out, and observed that the dimension estimates lie in a range of $d_h = 7 - 10$. As opposed to the KSE, they observed that the dimension estimates change when varying the number of linear layers and weight decay parameters.

In this work we explore an extension to IRMAE-WD that improves robustness of dimension estimates for more complex systems. These are more complicated in the sense that they are either more chaotic, embedded in a higher dimension, or both. We show that this is achieved with the addition of branches with different numbers of linear layers. We call this architecture IRMAE-WD-B, where B stand for branching. In the following sections we explain this framework and apply it to a library of dynamical systems.

4.2 Formulation

In this section we present IRMAE-WD-B, an extension of IRMAE-WD introduced by Zeng et al. [74]. A schematic representation of the framework is shown in Figure 4.1. The encoder, denoted by $E\left(\tilde{\omega};\theta_E\right)$ reduces the dimension from N to d_z. We then include a linear network

$\mathcal{W}_i\left(\cdot;\theta_W\right)$ between the encoder and the different decoders which consists of several linear layers (matrix multiplications). We note that the forward pass propagates through different branches of varying number of linear layers, which differs from IRMAE-WD. This bypasses the step of defining a specific number of linear layer. Finally, the decoders $\tilde{\omega}_{r,i} = D_i\left(z_i;\theta_{D_i}\right)$ map back to the full space. An L_2 (“weight decay") regularization to the weights is also added, with prefactor λ. The loss function for this architecture is

$$\mathcal{L}_i\left(\tilde{\omega};\theta_E,\theta_W,\theta_{D_i}\right) = \left\langle \|\tilde{\omega} - D_i\left(\mathcal{W}_i\left(E\left(\tilde{\omega};\theta_E\right);\theta_W\right);\theta_{D_i}\right)\|_2^2\right\rangle + \frac{\lambda}{2}\|\theta\|_2^2. \tag{4.1}$$

where $\langle\cdot\rangle$ is the average over a training batch, θ_E the weights of the encoder, θ_{D_i} the weights of the decoder i, and θ_W the weights of the linear network which contains the linear layers used in the branch. Notice that this architecture will output a number of losses defined by the number of linear layers. We can train this network by selecting a branch and backpropagating through it to update the weights. To select the branch we first calculate the loss for each of the paths. This is then weighted by calculating $w_i = e^{-\mathcal{L}_i}$ and normalizing by the sum of all of the paths $\hat{w}_i = w_i/\sum w_i$. We can then define the ranges $\hat{w}_{1,\text{ range}} = [0,\hat{w}_1)$, $\hat{w}_{2,\text{ range}} = [\hat{w}_1,\hat{w}_1+\hat{w}_2)$, and so on for all of the paths. By generating a random number between zero and one at every epoch during training we can select with higher probability the path that has the lowest loss, while also giving it the opportunity to explore other paths. To further improve the model at the end, we train using the lowest MSE in the last 10% of the total number of epochs.

After training, we can perform SVD on the covariance matrix of the encoded data matrix Z_i to obtain the singular vectors U_i and singular values S_i, as shown in Fig. 4.1b. Then, by choosing only the singular values above some very small threshold (typically $\gtrsim 6$ orders of magnitude smaller than the leading singular values), we may project down to fewer dimensions by projecting onto the corresponding singular vectors U, denoted $\hat{U}$ to yield the low-rank manifold representation. We refer to Table 4.1 for details on the architecture.

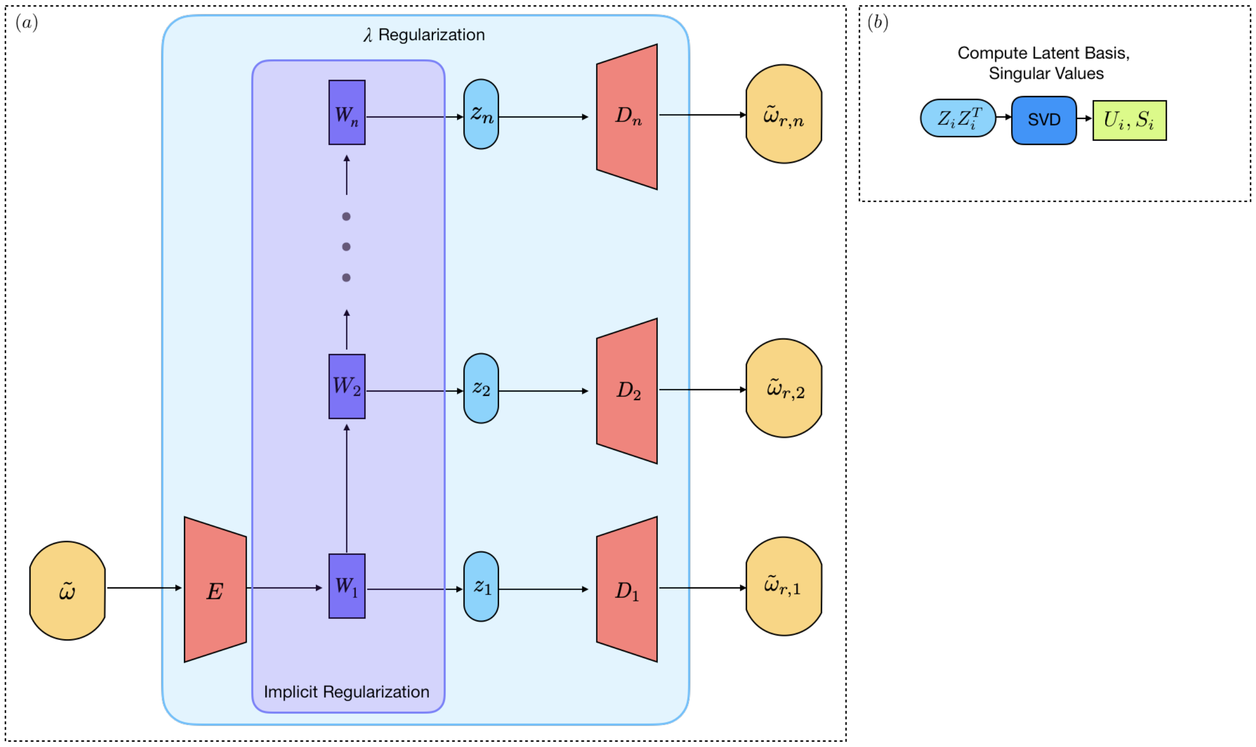


Figure 4.1: Implicit Rank Minimizing autoencoder with weight decay and branching (IRMAE-WD-B.) framework: a) network architecture with regularization mechanisms, b) singular value decomposition of the covariance of the learned latent data representation Z_i.

4.3 Results

4.3.1 Dimension estimates for a library of systems

We test IRMAE-WD-B on a library of systems in this section. First we show a comparison between IRMAE-WD and IRMAE-WD-B applied to the case of a one-dimensional arc embedded in an ambient space of $N = 1000$. Then, we increase complexity, and present results for KSE, Kolmogorov flow, and finally we test IRMAE-WD-B in combination with k-means clustering to learn local representations.

Table 4.1: Here we list the architecture and parameters utilized in the studies of this paper. For brevity, the decoders, $\mathcal{D}$, of each architecture is simply mirrors of the encoder, $\mathcal{E}$. Each network has a total of 10 linear layers with shape $d_z \times d_z$ between the encoder and decoder. Learning rates were set to 10^{-3} and with mini-batches of 128.

Dataset	$\mathcal{E}$	Activation	d_z
Arc	1000/2000/500/20	ReLU/ReLU/lin	20
KSE $L = 22$	64/512/256/20	ReLU/ReLU/lin	20
Kolmogorov flow	1024/2048/256/40	ReLU/ReLU/lin	40

One-dimensional arc embedded in higher dimensional space

In the case of the NSE, turbulent trajectories track solutions that appear in the form of traveling waves (TW) and relative periodic orbits (RPO) which are known as exact coherent states (ECS) [18]. Hence, these will be embedded in high dimensional spaces which can be in the order of $\sim 100,000$ degrees of freedom. Inspired by this we build a toy problem consisting of a one dimensional arc embedded in $N = 1000$. This resembles a portion of a periodic orbit, hence we can represent and find a mapping to a dimension of $d_h = 1$. To embed this we first generate data as shown in Figure 4.2 which will have a dimension of $X_{\text{arc}} \in \mathbb{R}^{2\times N_s}$. To embed in a dimension of $N = 1000$ we generate a random matrix of size $X_{\text{rand}} \in \mathbb{R}^{2\times 1000}$. By multiplying these matrices we can get the data used for training $X = X_{\text{rand}}^T X_{\text{arc}} \in \mathbb{R}^{1000\times N_s}$. We first want to investigate if IRMAE-WD is able to capture the drop in the singular values at a dimension of $d_h = 1$. To do this we vary the number

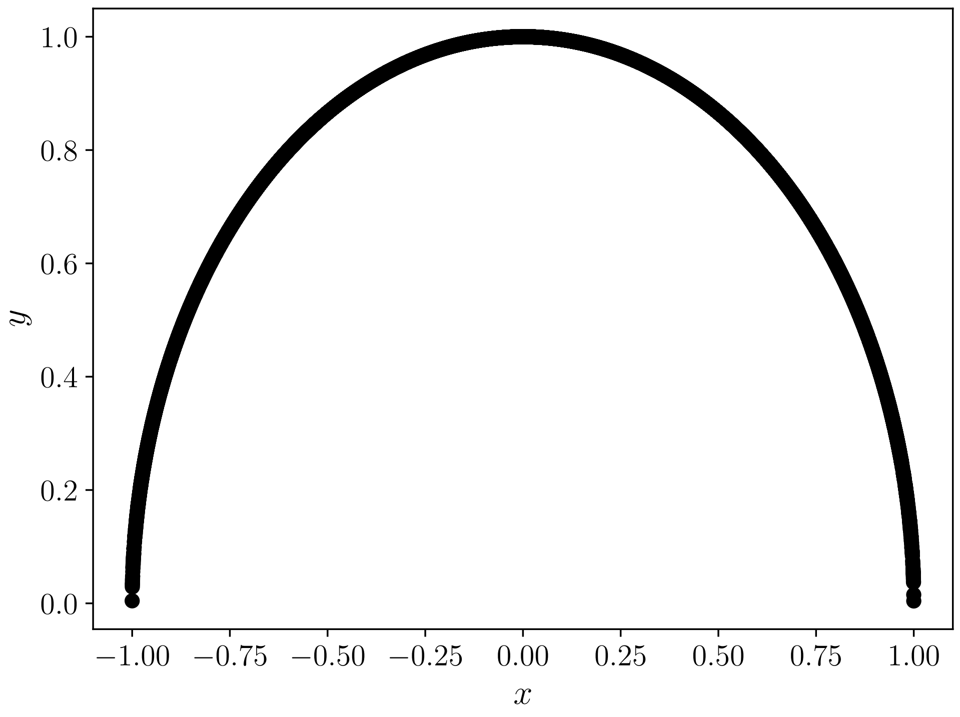
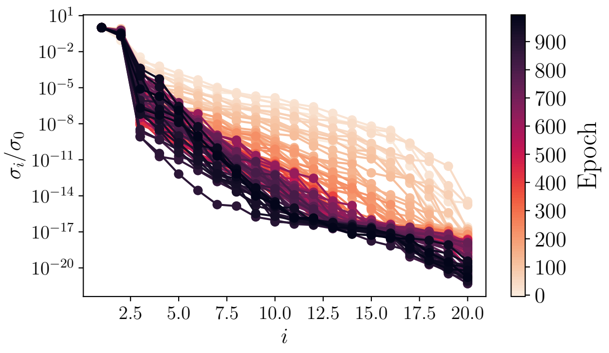


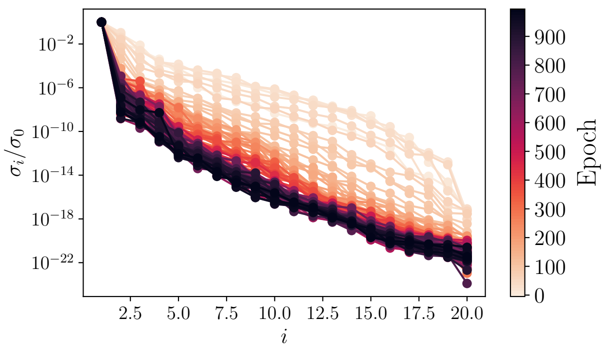
Figure 4.2: Visualization of arc data used for training and testing.

of linear layers and use a value of $\lambda = 10^{-6}$ for the weigh decay parameter. Specifics of the NN architecture is given in Table 4.1. We consider the cases of $L = 4, 6, 8$, which fall in the ranges studied by Zeng et al. [74] and train three models for each. Results for $L = 6$ are shown in Figure 4.3. Here we see that only one of the trials is able to capture the drop at $d_h = 1$. For the case of $L = 4$ (not shown) all of the drops happen at $d_h = 2$ and for $L = 8$ (not shown) only one of the trials captures the drop of $d_h = 1$ similar to $L = 6$.

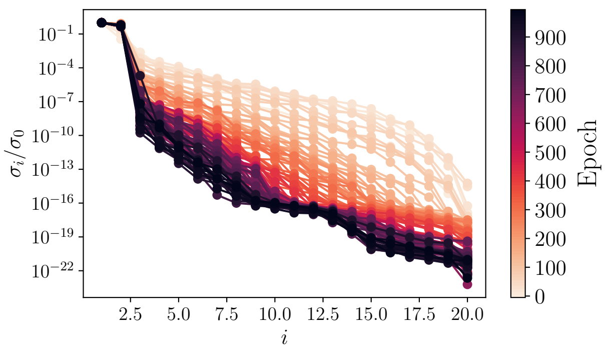
When the aim is to find the minimal-dimensional model, which we know should be of dimension $d_h = 1$, this poses a problem. With this we motivate a new modification in the architecture presented in Section 4.2. We specifically ask the question if there is a correct number of linear layers needed to correctly estimate the dimension. We select a total of $L = 10$ which is in the range considered by Zeng et al. [74]. This means that the architecture will be able to explore linear layers starting from $L = 1$ through $L = 10$. We train a total of five models with the architecture discussed in Section 4.2. In Figure 4.4 we show results of MSE versus dimension estimate from the models trained with IRMAE-WD and IRMAE-



(a)



(b)



(c)

Figure 4.3: Evolution of singular values of the covariance matrix of the encoded test data ZZ^T during training of an IRMAE-WD model with $L = 6$ and $\lambda = 10^{-6}$ for three trials. Here the drops happen at (a) $d_h = 2$, (b) $d_h = 1$, and (c) $d_h = 2$.

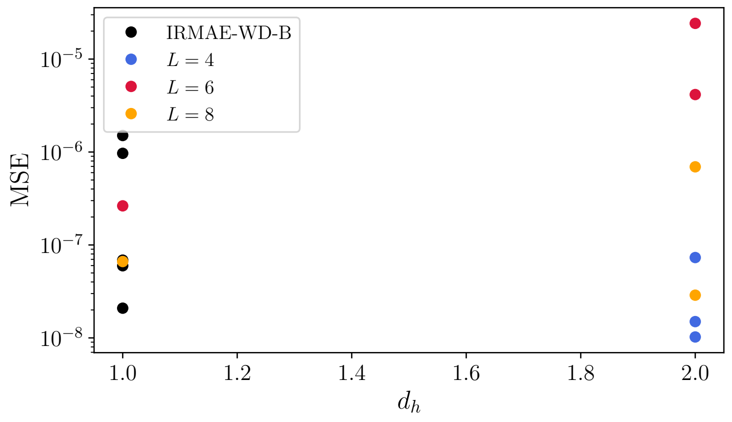


Figure 4.4: Arc: MSE vs dimension d_h given by the spectral gap of the singular values for $L = 4, 6, 8$ and $\lambda = 10^{-6}$ and our proposed architecture. Each case of varying L contains three trials and for our architecture we consider five trials.

WD-B. The latter is able to correctly capture $d_h = 1$ and performs better. In addition we only have to train five models and we avoid changing L which is done automatically. Interestingly, the number of L selected at the end for the five cases is different with values of $L = 5, 5, 6, 3, 2$. This means that there is not a specific number of linear layers that can correctly predict the dimension and there might be another factor involved. The success of this architecture in this simple system motivates us to extend this to more complex problems which are shown in the following sections.

Kuramoto-Sivashinsky equation (KSE)

The one-dimensional KSE takes takes the form

$$\frac{\partial v}{\partial t} = -v\frac{\partial v}{\partial x} - \frac{\partial^2 v}{\partial x^2} - \frac{\partial^4 v}{\partial x^4}, \tag{4.2}$$

Similar to Zeng et al. [74] we test our architecture on the KSE equation $L = 22$ with 40,000 snapshots sampled on 64 mesh points. Details of the architecture is given in Table 4.1. For this system it is known that the inertial manifold dimension is $d_h = 8$ [19, 37, 61, 70]. We see that this is the case for the five trials that we test with comparable MSEs. An example of one of the trials is shown in Figure 4.5 which has an MSE over the test data of 5×10^{-4}.

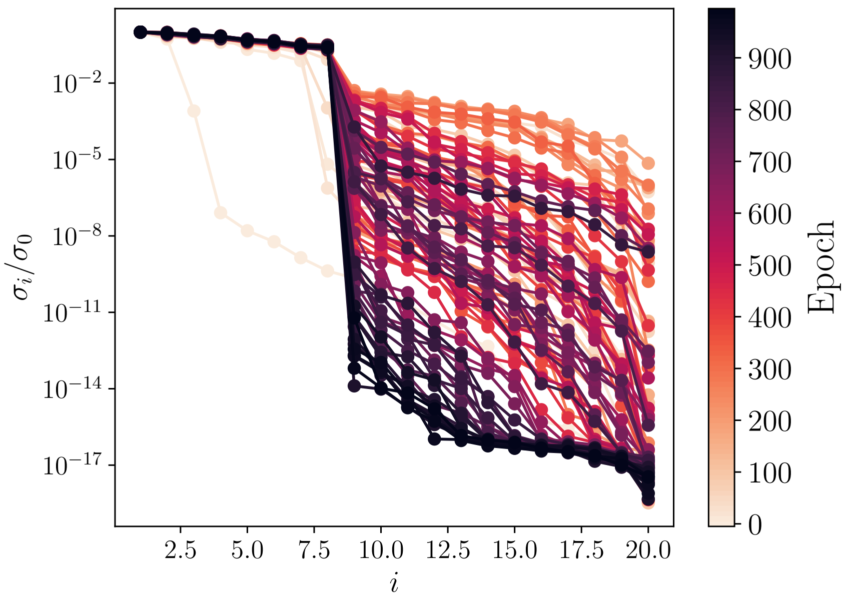


Figure 4.5: Evolution of singular values of the covariance matrix of the encoded test data ZZ^T during training of our model for the KSE with a final linear layer of $L = 9$ and $\lambda = 10^{-6}$. Here the drop happens at $d_h = 8$ as expected.

This shows again the robustness of our method where we are able to estimate the correct dimension and we avoid training many models with different amounts of linear layers.

Kolmogorov Flow - Re $= 14.4$

The two-dimensional NSE with Kolmogorov forcing are

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + \boldsymbol{\nabla} p = \frac{1}{\text{Re}} \nabla^2 \boldsymbol{u} + \sin(ny)\hat{\boldsymbol{x}}, \tag{4.3}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{4.4}$$

where flow is in the $x - y$ plane, $\boldsymbol{u} = [u, v]$ is the velocity vector, p is the pressure, n is the wavenumber of the forcing, and $\hat{\boldsymbol{x}}$ is the unit vector in the x direction. Here $\text{Re} = \frac{\sqrt{\chi}}{v}\left(\frac{L_y}{2\pi}\right)^{3/2}$ where χ is the dimensional forcing amplitude, ν is the kinematic viscosity, and L_y is the size of the domain in the y direction. We consider the periodic domain $[0, 2\pi/\alpha] \times [0, 2\pi]$ with $\alpha = 1$. Vorticity is defined as $\omega = \hat{\boldsymbol{z}} \cdot \boldsymbol{\nabla} \times \boldsymbol{u}$, where $\hat{\boldsymbol{z}}$ is the unit vector in the z direction (orthogonal to the flow).

In the case of Kolmogorov flow the manifold dimension in the chaotic regime of $\mathrm{Re} = 14.4$, $n = 2$ is not known. However, recent articles have found minimal dimensional models where dynamics can be predicted [53, 54]. Pérez De Jesús et al. [54] used IRMAE-WD to get dimension estimates for this system in the fundamental representation where symmetries are factored out. They observed that accounting for the symmetries drastically improved the dimension estimates for the system, with ranges of $d_h = 7 - 10$ for data with symmetries factored out versus $d_h = 39 - 40$ for the original data. They were also able to learn dynamical models with neural ordinary differential equations with great success at a dimension of $d_h = 10$. Pérez De Jesús & Graham were also able to find great dynamical models at a dimensions of $d_h = 5 - 9$ using autoencoders and tracking the performance of the MSE [53]. However, sweeping over dimensions was necessary for this analysis as well as training models for each of the dimensions. We now want to test our method in this data for the case of factored out symmetries. For more details on the symmetries and how we account for them refer to [54]. In Figure 4.6 we show results of MSE versus dimension estimate from the models trained with IRMAE-WD and our method. We can see that our method consistently estimates a dimension of $d_h = 8$ as opposed to varying the number of linear layers. The performance of the models trained seem to be slightly affected based on the MSE, however it is still in the same order of magnitude.

4.3.2 Hierarchical clustering

As discussed in Section 4.1 a system in the inertial manifold representation $d_{\mathcal{M}}$ can be seen as overlapping charts which together form an atlas. An example of this can be any dynamical system that lies in a circle, which is the case of a limit cycle. Reducing the dimensions of the global system to find a minimal representation results in $2d_{\mathcal{M}} = 2$. One can only uniquely represent this system with $d_{\mathcal{M}} = 1$ in the local sense. This is the case for the arc data showed in Section 4.3.

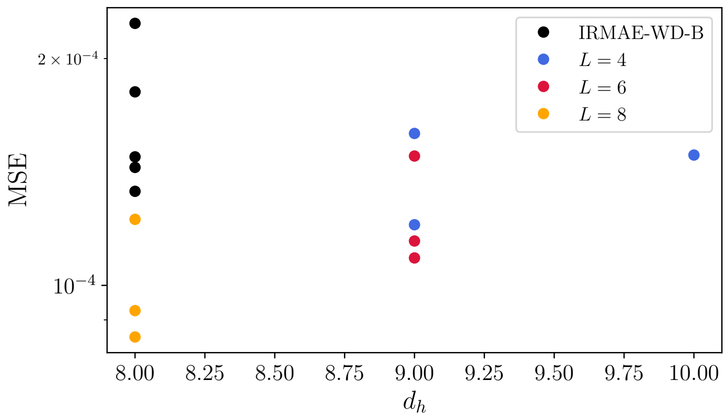


Figure 4.6: Kolmogorov Flow, Re $= 14.4$: MSE vs dimension d_h given by the spectral gap of the singular values for $L = 4, 6, 8$ and $\lambda = 10^{-6}$ and our proposed architecture. Each case of varying L contains three trials and for our architecture we consider five trials.

For the arc case, data is generated such that $y = [0, 1]$ for $x = [-1, 1]$. Hence the other half corresponds to $y = [-1, 0)$ and by stitching these two together one can represent the full circle. Similarly, more complicated systems can be charted into many local representations that together capture the full system. However, how to chart these systems might not be as trivial as for the circle case. Floryan & Graham developed a method to implement data-driven local representations for dynamical systems such as the quasiperiodic dynamics on a torus, a reaction-diffusion system, and the KSE to learn dynamics on invariant manifolds of minimal dimension [23]. They refer to this method as *Charts and Atlases for Nonlinear Data-Driven Dynamics on Manifolds* – “CANDyMan". Pérez De Jesús et al. [54] took an approach inspired by CANDyMan where a single chart is learned for Kolmogorov flow which comes from the mapping of the symmetries to a fundamental chart.

In this section we extend the idea presented by Pérez De Jesús et al. [54] where we further split the fundamental space into local charts similar to Floryan & Graham [23]. The motivation comes from the complexity of chaotic systems governed by the NSE where the dynamics can lie in the vicinity of many exact coherent states [18]. In the case of Kolmogorov flow, Re $= 14.4$ due to the discrete symmetries of the system, there are several RPOs [2]. The dynamics are characterized by quiescent intervals where the trajectories approach the

RPOs (which are now unstable), punctuated by fast excursions. There are four RPOs related by symmetries which means that at least four clusters are needed to capture each RPO. By mapping to the fundamental chart we bypass this step and ideally less clusters are needed to find the inertial manifold dimension.

Figure 4.7a displays k-means clustering with a total of five clusters to the fundamental chart. Here we see the mapping of the data to the $\hat{\omega}_R(0,1) - \hat{\omega}_I(0,1) - \hat{\omega}_I(0,2)$ projection which comes from taking the Fourier transform of the data. Cluster three and four capture the area near the RPO while the other clusters capture the regions corresponding to the heteroclinic and homoclinic like connections between the different RPOS. Figure 4.7b shows the time evolution of $\langle \|\omega(t)\|^2 \rangle$ corresponding to the five clusters. We now look at the MSEs versus dimension estimates for all of the clusters. Figure 4.8 shows the MSEs versus dimension estimates for the different clusters. As in previous sections, we compare our method with the original IRMAE-WD. The black markers correspond to our method and the blue markers correspond to original IRMAE-WD. Different from the previous cases we see that the dimension estimates are not as robust. Instead our method now estimates a range of dimensions $d_h = 7 - 9$. As opposed to original IRMAE-WD, $d_h = 7 - 13$, the range of dimensions our method estimates is smaller. We also observe that most of the estimates happen at $d_h = 7$.

4.4 Summary

In this work we introduce IRMAE-WD-B, an extension to IRMAE-WD as presented by Zeng et al. [74]. The primary enhancement involves integrating branches into the AE architecture to mitigate the framework's dependency on the number of linear layers in the encoder. This adjustment addresses a limitation present in the original IRMAE-WD, where the parameter for the number of linear layers is predetermined. Thus, this new architecture reduces the amount of models to be trained as it explores different paths. In this study, we observed

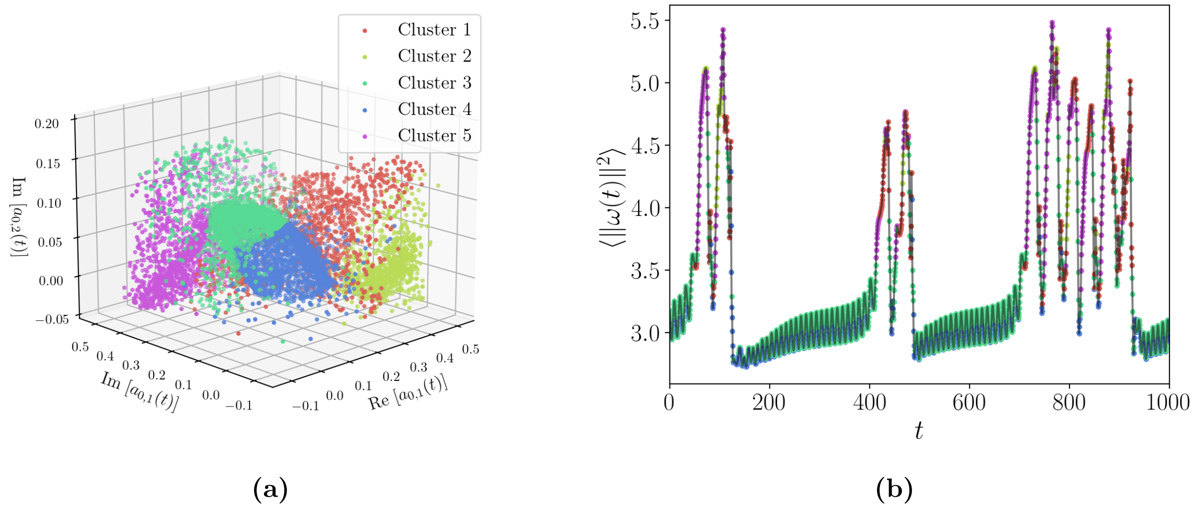


Figure 4.7: (a) State-space projection of a trajectory into the subspace $\hat{\omega}_R(0,1)-\hat{\omega}_I(0,1)-\hat{\omega}_I(0,2)$ for Re $= 14.4$. Colors correspond to the different clusters. (b) $\langle\|\omega(t)\|^2\rangle$ vs time with colors corresponding to different clusters.

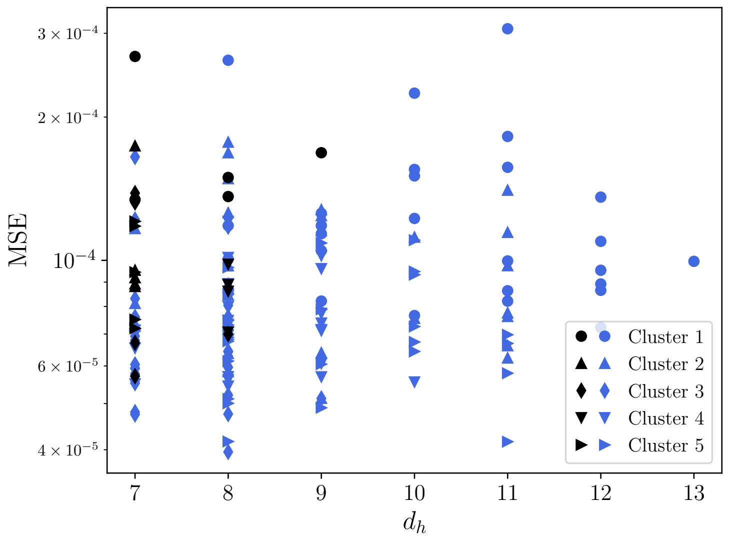


Figure 4.8: Kolmogorov Flow, Re $= 14.4$, k-means: MSE vs dimension d_h given by the spectral gap of the singular values for. Black markers correspond to IRMAE-WD-B models and blue markers correspond to IRMAE-WD models with valying number of linear layers $L = 4, 6, 8$.

that IRMAE-WD-B does not adhere to a specific pathway for the same system; rather, it demonstrates an alternation of varying numbers of linear layers across all systems examined. From this outcome, we suspect that there is a level of randomness that is helping improve the dimension estimate. Moving forward we are interested in learning models for forecasting in these minimal-dimensional representations. Code and sample data that support the findings of this study are openly available at `https://github.com/mdgrahamwisc/IRMAE_WD_B`.

5

Conclusions

5.1 General summary

In this thesis, we developed data-driven reduced-order models for the Navier-Stokes equations. We mainly focused on the two-dimensional Kolmogorov flow problem in a chaotic regime which contains the essence characteristics of fluid turbulence. This is motivated by the high-dimensionality of the systems, and subsequently the need to find efficient high fidelity reduced order models. These can further be deployed for controls to reduce drag in the case of turbulent flows and for predictive purposes. Motivated by the recent advances in machine learning we use autoencoders to learn low-dimensional representations of the data and combine with neural ordinary differential equations and dense neural networks to predict in time. We also develop a methodology which includes the physics of the system, and an extension to IRMAE-WD, which improves dimension estimates for complex data.

In Chapter 2 we presented a data-driven methodology to learn an estimate of the embedding dimension of the manifold for chaotic Kolmogorov flow and the time evolution on it. An autoencoder is used to find a nonlinear low-dimensional subspace and a dense neural network to evolve it in time. By analyzing the model performance as a function of latent space dimension we can estimate the minimum number of dimensions required to capture the system dynamics. We then calculated long and short time statistics based on the data-driven reduced-order models showing great success compared with true data.

In Chapter 3 we introduced a method that we call symmetry charting and apply it to Kolmogorov flow in a chaotic regime. This symmetry charting method factors out symmetries so we can train reduced-order models in a fundamental chart and ensure equivariant trajectories – where we are essentially learning coordinates and dynamics on one region of the invariant manifold for the long-time dynamics rather than having to learn these for the entire manifold. We show great success when factoring out symmetries compared with models that are trained with original data.

In Chapter 4 we presented an extension to IRMAE-WD which was introduced by Zeng et al. [74] to automatically estimate the underlying dimensionality of a data set, and applied it to a library of dynamical systems. The architecture is extended such that it can explore various paths that contain different numbers of linear layers. This is shown to improve robustness of dimension estimates for the Navier-Stokes equations, which is a hard problem, and we give insights on estimating inertial manifold dimensions for these complicated systems.

In the following sections we discuss extensions of our work to a more complicated system and potential avenues. We show how the symmetry charting method can be applied to pipe flow with good preliminary results and how different regularizations and machine learning methods can be used for robust dimension estimates and forecasting. We also discuss some final thoughts on hierarchical clustering and discuss preliminary results on using IRMAE-WD-B on other complex cases.

5.2 Future work

5.2.1 Symmetry charting applied to pipe flow

In our previous work we focused on two-dimensional Kolmogorov flow which exhibits chaotic dynamics at Re $= 14.4$ and $n = 2$. However, turbulence appears in three dimensions in industrial settings and nature. Hence, it is important to extend what we have learned to more realistic scenarios. In this section we focus our attention to pipe flow. As discussed by [17, 66] the governing equations are

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{U} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + 32 \frac{\beta}{\mathrm{Re}} \hat{z} + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u}, \tag{5.1}$$

$$\nabla \cdot \boldsymbol{u} = 0. \tag{5.2}$$

The Reynolds number is defined as $\mathrm{Re} = UD/\nu$, where U, D, and ν are mean velocity of the flow, diameter of the pipe, and kinematic viscosity. In this formulation $\boldsymbol{u}$ corresponds to the deviation from the Hagen-Poiseuille flow equilibrium $\boldsymbol{U}(r) = 2\left(1 - (2r)^2\right) \hat{\boldsymbol{z}}$. Pressure is p and $\beta = \beta(t)$ is the fractional pressure gradient needed to maintain a constant mass flux. The computational domain is $\Omega = [1/2, 2\pi/m, \pi/\alpha] \equiv (r, \theta, z) \in [0, 1/2] \times [0, 2\pi/m] \times [0, \pi/\alpha]$, where $L = \pi/\alpha$ is the length of the pipe, and m is a parameter to account for shift-invariance in the azimuthal direction. For this study $\alpha = 1.7$ which has been used in previous works [12, 66]. We show in Figure 5.1 a snapshot of the magnitude of the velocity components for the case of $m = 4$ ('shift-and-reflect' invariant space), $\mathrm{Re} = 2500$ which corresponds to a fourth of the domain in the θ direction, the section of the pipe is copied to get the figure shown. This minimal computational cell has been the focus of previous work to learn a library of invariant solutions [12].

In this section we focus on the case where $m = 1$ (naturally periodic pipe flow). In many systems and practical applications one has access to the full domain, and not the invariant space as in the case of $m = 4$. Hence, it is important to address the symmetries in pipe flow to build accurate data-driven reduced order models.

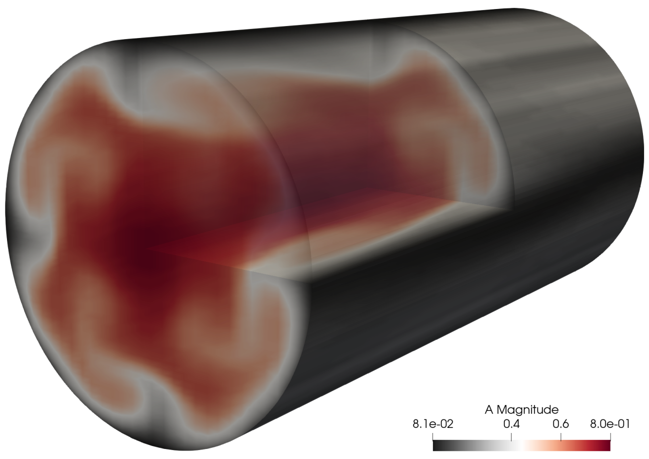


Figure 5.1: Magnitude of velocities of pipe flow for $m = 4$, $\mathrm{Re} = 2500$.

Pipe flow symmetries

The equations for pipe flow are invariant under several symmetry operations [66], namely a reflection along the $\theta = 0$ azimuthal angle, and continuous translations in θ and z:

$$g(\phi, \ell)[u, v, w, p](r, \theta, z) = [u, v, w, p](r, \theta - \phi, z - \ell), \tag{5.3}$$

$$\sigma[u, v, w, p](r, \theta, z) = [u, -v, w, p](r, -\theta, z). \tag{5.4}$$

Similar to our previous work [54] where we developed the symmetry charting method, we are interested in factoring out these symmetries to map to a fundamental chart. To apply this method we first want to understand how the Fourier coefficients change under the symmetry operations. This will provide a set of indicators that can be used to map to the fundamental chart. After Fourier transforming the equations in the $\theta - z$ directions and simplifying, these are the actions of the symmetry operations on the Fourier coefficients:

$$g(\phi, \ell)[\hat{u}, \hat{v}, \hat{w}](r, k_\theta, k_z) \to [\hat{u}, \hat{v}, \hat{w}](r, k_\theta, k_z) e^{-ik_\theta \phi} e^{-ik_z \ell}, \tag{5.5}$$

$$\sigma[\hat{u}, \hat{v}, \hat{w}](r, k_\theta, k_z) \to [\hat{u}, -\hat{v}, \hat{w}](r, -k_\theta, k_z). \tag{5.6}$$

Map to fundamental chart

To remove the continuous symmetry we use the method of slices as presented by Budanur & Hof [8, 9]. A slice template is defined as

$$\boldsymbol{u}'_z(r, \theta, z) = J_0(\alpha r) \cos(2\pi z/L), \tag{5.7}$$

where J_0 is the Bessel function of the first kind which vanishes at the pipe wall, $J_0(\alpha/2) = 0$. This template can be used to phase-align the snapshots as

$$\boldsymbol{u}_z(t) = g_z\left(L\phi_z/2\pi\right) \boldsymbol{u}(t), \tag{5.8}$$

where

$$\phi_z(t) = \arg\left[\langle \boldsymbol{u}(t), \boldsymbol{u}'_z \rangle + i \left\langle \boldsymbol{u}(t), g_z(-L/4)\boldsymbol{u}'_z \right\rangle\right]. \tag{5.9}$$

Similarly for the θ direction

$$\boldsymbol{u}'_\theta(r, \theta, z) = J_0(\alpha r) \cos(\theta), \tag{5.10}$$

$$\boldsymbol{u}_\theta(t) = g_\theta\left(\phi_\theta\right) \boldsymbol{u}(t), \tag{5.11}$$

$$\phi_\theta(t) = \arg\left[\langle \boldsymbol{u}(t), \boldsymbol{u}'_\theta \rangle + i \left\langle \boldsymbol{u}(t), g_\theta(-\pi/2)\boldsymbol{u}'_\theta \right\rangle\right]. \tag{5.12}$$

To factor out the discrete symmetry we can consider the Fourier mode $k_\theta = 1$, $k_z = 0$ as an indicator. This modifies the Fourier coefficient as $\hat{v}_R(r, 1, 0) + i\hat{v}_I(r, 1, 0) \to -\hat{v}_R(r, -1, 0) -$

$i\hat{v}_I(r,-1,0)$ for the $\hat{v}$ component of velocity. We can also pick other components of k_θ, k_z. Setting the sign of $\hat{v}_R(r,1,0)$ to be positive maps the snapshots to a fundamental subspace.

Autoencoder preliminary results

For this system we consider a grid of size $(N_r, N_\theta, N_z) = (64, 60, 48)$. Together with the three components of velocity the state size is $N = 552,960$. Instead of feeding the large full velocity field in the AE we take the approach discussed by Linot & Graham [39] and Duggleby et al. [21] where POD is performed to do an initial linear reduction. In this step, we reduce the dimensions from $N = 552,960$ to $N = 4,080$ which contains 98% of the energy. This linear reduction is crucial to train the AEs. Then, we train an IRMAE-WD model to reduce the dimensions to $d_h = 50$ with $L = 4$ linear layers and a weight decay value of $\lambda = 10^{-6}$. We show results of the reconstruction of the first 10^3 POD modes in Figure 5.2. Notice that factoring out the symmetries as discussed in the previous section vastly improves the reconstruction of the POD modes. Moving forward in this work we want to train NODEs for time integration in the fundamental chart. We expect that these NODEs will outperform the models that do not have symmetries factored out. Ideally these will show improved performance in short-time tracking and long-time statistics.

5.2.2 Regularizing using dropout

In Chapter 4 we introduced a methodology that improves robustness of dimension estimates for a library of dynamical systems. We achieved this with addition of paths with different number of linear layers. A natural follow-up to this work is the exploration of dropout. This is analogous to the method we proposed in the sense that instead of pathing through different branches, random connections are dropped during training [60]. This is equivalent to many models trained and has shown to help with overfitting. We have tested this on the case of the one-dimensional arc embedded in 1000 dimensions with preliminary results showing success of drops at a dimension of $d_h = 1$. Further tests in this area include varying

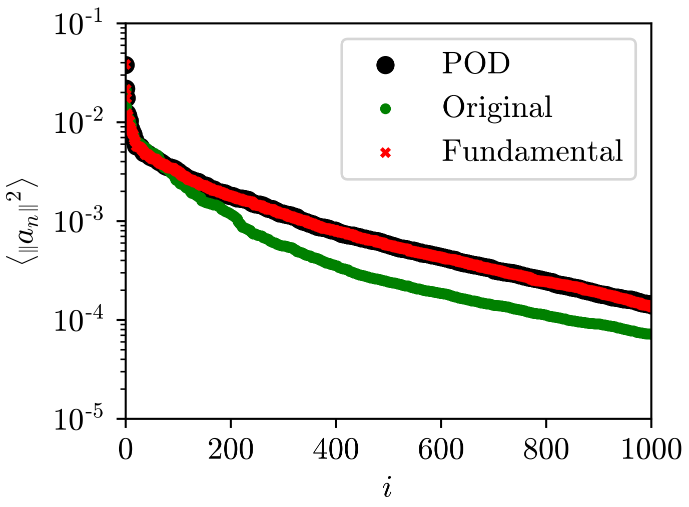


Figure 5.2: Reconstruction of POD modes for the case $m = 1$, Re $= 2500$ from models trained with IRMAE-WD on original data and data mapped to the fundamental chart. These are compared with the real POD modes.

the probability of zeroed elements in the architecture, varying number of linear layers or combine with IRMAE-WD-B, and extending to more complicated systems.

5.2.3 Contrastive learning with symmetries

Throughout this thesis we have shown how accounting for symmetries improves forecasting of reduced-order models. Specifically, short-time tracking is greatly affected when factoring out continuous and discrete symmetries. This was shown in Chapters 2 and 3 for two-dimensional Kolmogorov flow in a chaotic regime. A possible way to address symmetries and learn more efficient models is to use contrastive learning. In contrastive learning positive representations, which in this case corresponds to data in the same symmetry subspace, are mapped such that these are similar and negative representations, from a different symmetry subspace, are mapped such that these are more orthogonal [15, 16, 51]. This will result in latent representations that are separated based on the symmetries of the system. Similar to our previous work one can then train models for time evolution. An added benefit of using contrastive learning is that one can also address noisy versions of the snapshots, which is common in experimental data.

5.2.4 Forecasting and hierarchical clustering

We introduced in Chapter 4 the idea of hierarchical clustering which is inspired by CANDyMan presented in [23] and our symmetry charting method discussed in Chapter 3. With hierarchical clustering one can find local representations of minimal dimension in a fundamental chart where symmetries are factored out. Inspired by our previous works, we can learn models for time integration by tracking the indicator related to the symmetry and the local chart. The fundamental representation will typically contain data with different dynamical behavior. In the case of Kolmogorov flow there can be organized oscillating regions, quiescent, and bursting regions of increased energy. As in CANDyMan, k-means can be used to "cut" the space. However, different approaches can be taken depending on the type of data like isolating the quiescent and bursting regions in Kolmogorov flow.

5.2.5 IRMAE-WD-B in more complex cases

In Chapter 4 we introduced our framework to improve robustness of dimension estimates. We also tested this architecture, with preliminary results, on more complex cases including Kolmogorov flow at Re $= 20$, $n = 2$ and plane Couette flow (fluid confined between two plates) at Re $= 400$. In Kolmogorov flow ($N = 1024$) most of the dimension estimates are close to $d_h = 20$, however we did see some models that gave dimensions close to $d_h = 10$. This might be due to the added complexity of the system. For Couette flow ($N = 502$, POD modes) the estimates are close to $d_h = 30$ with one of the models predicting $d_h = 36$. In this case only one of the continuous symmetries was factored out. We would expect the dimensions to be more robust when factoring all the symmetries, as is the case for Kolmogorov flow. An interesting avenue is to systematically consider the symmetries of more complicated systems before performing IRMAE-WD-B which could improve results.

References

[1] D. Armbruster, R. Heiland, E. J. Kostelich, and B. Nicolaenko. Phase-space analysis of bursting behavior in Kolmogorov flow. *Physica D: Nonlinear Phenomena*, 58(1-4): 392–401, 1992.

[2] D. Armbruster, B. Nicolaenko, N. Smaoui, and P. Chossat. Symmetries and dynamics for 2-D Navier-Stokes flow. *Physica D: Nonlinear Phenomena*, 95(1):81–93, 1996.

[3] N. Aubry, P. Holmes, J. L. Lumley, and E. Stone. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *Journal of Fluid Mechanics*, 192: 115–173, 1988.

[4] P. Bartello and T. Warn. Self-similarity of decaying two-dimensional turbulence. *Journal of Fluid Mechanics*, 326:357–372, 1996.

[5] C. Bishop. Bayesian PCA. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1998.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[7] N. B. Budanur and P. Cvitanović. Unstable Manifolds of Relative Periodic Orbits in the Symmetry-Reduced State Space of the Kuramoto–Sivashinsky System. *Journal of Statistical Physics*, 167(3-4):636–655, 2017.

[8] N. B. Budanur and B. Hof. Heteroclinic path to spatially localized chaos in pipe flow. *Journal of Fluid Mechanics*, 827:R1, 2017.

[9] N. B. Budanur and B. Hof. Complexity of the laminar-turbulent boundary in pipe flow. *Physical Review Fluids*, 3(5):054401, 2018.

[10] N. B. Budanur, D. Borrero-Echeverry, and P. Cvitanović. Periodic orbit analysis of a system with continuous symmetry—A tutorial. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(7):073112, 2015.

[11] N. B. Budanur, P. Cvitanović, R. L. Davidchack, and E. Siminos. Reduction of SO (2) symmetry for spatially extended dynamical systems. *Physical review letters*, 114(8): 084102, 2015.

[12] N. B. Budanur, K. Y. Short, M. Farazmand, A. P. Willis, and P. Cvitanović. Relative periodic orbits form the backbone of turbulent pipe flow. *Journal of Fluid Mechanics*, 833:274–301, 2017.

[13] G. J. Chandler and R. R. Kerswell. Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow. *Journal of Fluid Mechanics*, 722:554–595, 2013.

[14] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. *Advances in neural information processing systems*, 31, 2018.

[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[16] C. Y. Chuang, J. Robinson, Y. C. Lin, A. Torralba, and S. Jegelka. Debiased contrastive learning. *Advances in neural information processing systems*, 33:8765–8775, 2020.

[17] C. R. Constante-Amores and M. D. Graham. Data-driven state-space and Koopman operator models of coherent state dynamics on invariant manifolds. *Journal of Fluid Mechanics*, 984:R9, 2024.

[18] C. J. Crowley, J. L. Pughe-Sanford, W. Toler, M. C. Krygier, R. O. Grigoriev, and M. F. Schatz. Turbulence tracks recurrent solutions. *Proceedings of the National Academy of Sciences*, 119(34):e2120665119, 2022.

[19] X. Ding, H. Chaté, P. Cvitanović, E. Siminos, and K. Takeuchi. Estimating the dimension of an inertial manifold from unstable periodic orbits. *Physical review letters*, 117 (2):024101, 2016.

[20] N. A. K. Doan, W. Polifke, and L. Magri. Auto-encoded reservoir computing for turbulence learning. In *International Conference on Computational Science*, pages 344–351. Springer, 2021.

[21] A. Duggleby, K. S. Ball, M. R. Paul, and P. F. Fischer. Dynamical eigenfunction decomposition of turbulent pipe flow. *Journal of Turbulence*, (8):N43, 2007.

[22] M. Farazmand and T. P. Sapsis. A variational approach to probing extreme events in turbulent dynamical systems. *Science advances*, 3(9):e1701533, 2017.

[23] D. Floryan and M. D. Graham. Data-driven discovery of intrinsic dynamics. *Nature Machine Intelligence*, 4(12):1113–1120, 2022.

[24] C. Foias, O. Manley, and R. Temam. Modelling of the interaction of small and large eddies in two dimensional turbulent flows. *ESAIM: Mathematical Modelling and Numerical Analysis*, 22(1):93–118, 1988.

[25] A. J. Fox, C. R. Constante-Amores, and M. D. Graham. Predicting extreme events in a data-driven model of turbulent shear flow using an atlas of charts. *Physical Review Fluids*, 8(9):094401, 2023.

[26] J. Green. Two-dimensional turbulence near the viscous limit. *Journal of Fluid Mechanics*, 62(2):273–287, 1974.

[27] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, coherent structures, dynamical systems and symmetry.* Cambridge university press, 2012.

[28] E. Hopf. A mathematical example displaying features of turbulence. *Communications on Pure and Applied Mathematics*, 1(4):303–322, 1948.

[29] M. Inubushi, M. U. Kobayashi, S. I. Takehiro, and M. Yamada. Covariant Lyapunov analysis of chaotic Kolmogorov flows. *Physical Review E*, 85(1):016331, 2012.

[30] V. Iudovich. Example of the generation of a secondary stationary or periodic flow when there is loss of stability of the laminar flow of a viscous incompressible fluid. *Journal of Applied Mathematics and Mechanics*, 29(3):527–544, 1965.

[31] J. Jiménez and A. Lozano-Durán. Coherent structures in wall-bounded turbulence. *Journal of Fluid Mechanics*, 842, 2018.

[32] L. Jing, J. Zbontar, et al. Implicit rank-minimizing autoencoder. *Advances in Neural Information Processing Systems*, 33:14736–14746, 2020.

[33] I. Jolliffe. *Principal Component Analysis.* Springer Verlag, 1986.

[34] S. Kneer, T. Sayadi, D. Sipp, P. Schmid, and G. Rigas. Symmetry-Aware Autoencoders: s-PCA and s-nlPCA. *arXiv preprint arXiv:2111.02893*, 2021.

[35] J. Lee. *Introduction to Smooth Manifolds.* Graduate Texts in Mathematics. Springer, 2003.

[36] J. M. Lee. Smooth manifolds. In *Introduction to smooth manifolds*, pages 1–31. Springer, 2013.

[37] A. J. Linot and M. D. Graham. Deep learning to discover and predict dynamics on an inertial manifold. *Physical Review E*, 101(6):062209, 2020.

[38] A. J. Linot and M. D. Graham. Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(7):073110, 2022.

[39] A. J. Linot and M. D. Graham. Dynamics of a data-driven low-dimensional model of turbulent minimal Couette flow. *Journal of Fluid Mechanics*, 973:A42, 2023.

[40] A. J. Linot, J. W. Burby, Q. Tang, P. Balaprakash, M. D. Graham, and R. Maulik. Stabilized neural ordinary differential equations for long-time forecasting of dynamical systems. *Journal of Computational Physics*, 474:111838, 2023.

[41] A. J. Linot, K. Zeng, and M. D. Graham. Turbulence control in plane Couette flow using low-dimensional neural ODE-based models and deep reinforcement learning. *International Journal of Heat and Fluid Flow*, 101:109139, 2023.

[42] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[43] H. F. Lui and W. R. Wolf. Construction of reduced-order models for fluid flows using deep feedforward neural networks. *Journal of Fluid Mechanics*, 872:963–994, 2019.

[44] L. Meshalkin and I. G. Sinai. Investigation of the stability of a stationary solution of a system of equations for the plane movement of an incompressible viscous liquid. *Journal of Applied Mathematics and Mechanics*, 25(6):1700–1705, 1961.

[45] R. Miranda and E. Stone. The proto-Lorenz system. *Physics Letters A*, 178(1-2): 105–113, 1993.

[46] J. Moehlis, H. Faisst, and B. Eckhardt. A low-dimensional model for turbulent shear flows. *New Journal of Physics*, 6(1):56, 2004.

[47] T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, and K. Fukagata. Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Physics of Fluids*, 33(2):025116, 2021.

[48] M. A. Nayak and S. Ghosh. Prediction of extreme rainfall event using weather pattern recognition and support vector machine classifier. *Theoretical and applied climatology*, 114(3):583–603, 2013.

[49] B. Nicolaenko and Z. S. She. Symmetry-breaking homoclinic chaos in Kolmogorov flows. In *Nonlinear world*. 1990.

[50] B. R. Noack and H. Eckelmann. A low-dimensional Galerkin method for the three-dimensional flow around a circular cylinder. *Physics of Fluids*, 6(1):124–143, 1994.

[51] A. V. D. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[52] J. Page, M. P. Brenner, and R. R. Kerswell. Revealing the state space of turbulence using machine learning. *Physical Review Fluids*, 6(3):034402, 2021.

[53] C. E. Pérez De Jesús and M. D. Graham. Data-driven low-dimensional dynamic model of Kolmogorov flow. *Physical Review Fluids*, 8(4):044402, 2023.

[54] C. E. Pérez De Jesús, A. J. Linot, and M. D. Graham. Building symmetries into data-driven manifold dynamics models for complex flows. *arXiv preprint arXiv:2312.10235*, 2023.

[55] N. Platt, L. Sirovich, and N. Fitzmaurice. An investigation of chaotic Kolmogorov flows. *Physics of Fluids A: Fluid Dynamics*, 3(4):681–696, 1991.

[56] C. W. Rowley and S. T. Dawson. Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49:387–417, 2017.

[57] S. Schonscheck, J. Chen, and R. Lai. Chart Auto-Encoders for Manifold Structured Data. *arXiv*, 2019.

[58] M. Sieber, C. O. Paschereit, and K. Oberleithner. Spectral proper orthogonal decomposition. *Journal of Fluid Mechanics*, 792:798–828, 2016.

[59] P. A. Srinivasan, L. Guastoni, H. Azizpour, P. Schlatter, and R. Vinuesa. Predictions of turbulent shear flows using deep neural networks. *Physical Review Fluids*, 4(5):054603, 2019.

[60] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[61] K. A. Takeuchi, H. L. Yang, F. Ginelli, G. Radons, and H. Chaté. Hyperbolic decoupling of tangent space and effective dimension of dissipative systems. *Physical Review E*, 84 (4):046214, 2011.

[62] R. Temam. Do inertial manifolds apply to turbulence? *Physica D: Nonlinear Phenomena*, 37(1-3):146–152, 1989.

[63] A. Thess. Instabilities in two-dimensional spatially periodic flows. Part I: Kolmogorov flow. *Physics of Fluids A: Fluid Dynamics*, 4(7):1385–1395, 1992.

[64] L. Van Der Maaten, E. O. Postma, H. J. van den Herik, et al. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(66-71):13, 2009.

[65] H. Whitney. The self-intersections of a smooth n-manifold in 2n-space. *Annals of Mathematics*, 45(2):220–246, 1944.

[66] A. P. Willis, P. Cvitanović, and M. Avila. Revealing the state space of turbulent pipe flow by symmetry reduction. *Journal of Fluid Mechanics*, 721:514–540, 2013.

[67] M. Winkels and T. S. Cohen. 3D G-CNNs for pulmonary nodule detection. *arXiv preprint arXiv:1804.04656*, 2018.

[68] M. Winkels and T. S. Cohen. Pulmonary nodule detection in CT scans with equivariant CNNs. *Medical image analysis*, 55:15–26, 2019.

[69] H. L. Yang and G. Radons. Geometry of inertial manifolds probed via a Lyapunov projection method. *Physical review letters*, 108(15):154101, 2012.

[70] H. L. Yang, K. A. Takeuchi, F. Ginelli, H. Chaté, and G. Radons. Hyperbolicity and the effective dimension of spatially extended dissipative systems. *Physical review letters*, 102(7):074102, 2009.

[71] S. Zelik. Attractors. Then and now. *arXiv preprint arXiv:2208.12101*, 2022.

[72] K. Zeng and M. D. Graham. Symmetry reduction for deep reinforcement learning active control of chaotic spatiotemporal dynamics. *Physical Review E*, 104(1):014210, 2021.

[73] K. Zeng, A. J. Linot, and M. D. Graham. Data-driven control of spatiotemporal chaos with reduced-order neural ODE-based models and reinforcement learning. *arXiv preprint ArXiv:2205.00579, to appear in Royal Society Proceedings A*, 2022.

[74] K. Zeng, C. E. P. De Jesús, A. J. Fox, and M. D. Graham. Autoencoders for discovering manifold dimension and coordinates in data from complex dynamical systems. *Machine Learning: Science and Technology*, 5(2):025053, 2024.

[75] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):262 – 286, 2004.