AFRL-AFOSR-VA-TR-2022-0420

Distentangling turbulent structure with nonlinear dynamics and machine learning

GRAHAM, MICHAEL
UNIVERSITY OF WISCONSIN SYSTEM
21 N PARK ST STE 6301
MADISON, WI, 53715
USA

07/25/2022
Final Technical Report

DISTRIBUTION A: Distribution approved for public release.

Air Force Research Laboratory
Air Force Office of Scientific Research
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
20220725	Final	START DATE 20180315	END DATE 20220314

4. TITLE AND SUBTITLE
Distentangling turbulent structure with nonlinear dynamics and machine learning

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
	FA9550-18-1-0174	61102F
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
MICHAEL GRAHAM

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSITY OF WISCONSIN SYSTEM
21 N PARK ST STE 6301
MADISON, WI 53715
USA

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Office of Scientific Research 875 N. Randolph St. Room 3112 Arlington, VA 22203	AFRL/AFOSR RTA1	AFRL-AFOSR-VA-TR-2022-0420

12. DISTRIBUTION/AVAILABILITY STATEMENT
A Distribution Unlimited: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The overall objective of the work performed under this grant was to integrate the dynamical systems approach to turbulent flows with ideas and tools from machine learning to develop and apply new approaches for the data-driven modeling and control of complex chaotic flow phenomena. Building on the observation above that very complex high-dimensional dynamics often lie on a surface (manifold) of much lower dimension, a central theme of the work is nonlinear dimension reduction, using machine learning to identify the manifold on which the data lie as well as the dynamical equations for the time-evolution of the system dynamics. Related to the issue of dimension reduction is that of modal decomposition of data -- for example, the classical proper orthogonal decomposition (POD) of flow data generates a set of basis vectors, and a reduced-dimensional representation of the data can be obtained by projection onto a subset of these vectors. (This is a linear dimension reduction process, which always projects data onto a flat surface. Nonlinear approaches like those that we use enable projection onto a curved manifold, and are thus often much more effective at dimension reduction than POD.) A number of specific accomplishments have been achieved. We have developed a new framework, which we call ``Data-driven Manifold Dynamics'' (DManD), that enables development of high-fidelity low-dimensional dynamic models for complex chaotic processes. This has been applied to a model system, the Kuramoto-Sivashinsky equation, a

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	UU	54

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
GREGG ABATE	425-1779

From: Air Force Office of Scientific Research
To: Technical Reports
Subject: [URL Verdict: Neutral][Non-DoD Source] Deliverable Received: FA9550-18-1-0174
Date: Friday, July 1, 2022 12:03:56 PM

OMA Team,
A new Final Performance has been submitted for your attention. Award Number: FA9550-18-1-0174

Supporting Report: https://afosr.gov1.qualtrics.com/WRQualtricsSurveyEngine/File.php?F=F_3MDMETDZUgCh9Kj

- Date Stamp: 12:01:01 , 7/1/2022
- Title: Distentangling turbulent structure with nonlinear dynamics and machine learning
- DISTRIBUTION: Yes- Approved for Public Release (Distro A)
- Principal Investigator: GRAHAM, MICHAEL
  - PI Email: mdgraham@wisc.edu
- Program Officer: ABATE, GREGG
- Report Type: Final Performance
- Reporting Period
  - Start Date: 3/15/2018 12:00:00 AM
  - End Date: 3/14/2022 12:00:00 AM
- Award Number: FA9550-18-1-0174
- Report Due Date: 6/12/2022

ABSTRACT:
The overall objective of the work performed under this grant was to integrate the dynamical systems approach to turbulent flows with ideas and tools from machine learning to develop and apply new approaches for the data-driven modeling and control of complex chaotic flow phenomena. Building on the observation above that very complex high-dimensional dynamics often lie on a surface (manifold) of much lower dimension, a central theme of the work is nonlinear dimension reduction, using machine learning to identify the manifold on which the data lie as well as the dynamical equations for the time-evolution of the system dynamics. Related to the issue of dimension reduction is that of modal decomposition of data -- for example, the classical proper orthogonal decomposition (POD) of flow data generates a set of basis vectors, and a reduced-dimensional representation of the data can be obtained by projection onto a subset of these vectors. (This is a linear dimension reduction process, which always projects data onto a flat surface. Nonlinear approaches like those that we use enable projection onto a curved manifold, and are thus often much more effective at dimension reduction than POD.) A number of specific accomplishments have been achieved. We have developed a new framework, which we call ``Data-driven Manifold Dynamics'' (DManD), that enables development of high-fidelity low-dimensional dynamic models for complex chaotic

processes. This has been applied to a model system, the Kuramoto-Sivashinsky equation, as well as to transitionally-turbulent plane Couette flow. Having in hand an efficient low-order model enables rapid implementation in downstream analyses such as controller design, and we have integrated DManD with reinforcement learning to drive drag reduction in turbulent Couette flow. Additionally, we have developed a new modal decomposition that we call the ``Data-driven wavelet decomposition" (DDWD). This approach integrates key aspects of POD and wavelet analysis, but iIn contrast to traditional wavelet bases, the basis elements at each stage are not simply dilations of given wavelets, but rather are determined stage-by-stage from the data. For data that is not self-similar, neither are the resulting basis elements. Rather, these represent the differing structures at the different stages. In contrast, for self-similar data, the basis vectors at different stages are related to one another by a simple rescaling. Indeed, for data from homogeneous isotropic turbulence we show self-similarity of the learned wavelet basis elements, which in turn reveals the self-similarity of the data.

Final Report: AFOSR grant FA9950-18-1-0174, "Disentangling turbulent structure with nonlinear dynamics and machine learning"

PI: Michael D. Graham, University of Wisconsin-Madison

Unsteady Aerodynamics & Turbulent Flows Program, Gregg Abate, Program Manager
3/15/18-3/14/21 (with no-cost extension to 3/14/22)

Abstract

The overall objective of the work performed under this grant was to integrate the dynamical systems approach to turbulent flows with ideas and tools from machine learning to develop and apply new approaches for the data-driven modeling and control of complex chaotic flow phenomena. Building on the observation above that very complex high-dimensional dynamics often lie on a surface (manifold) of much lower dimension, a central theme of the work is nonlinear dimension reduction, using machine learning to identify the manifold on which the data lie as well as the dynamical equations for the time-evolution of the system dynamics. Related to the issue of dimension reduction is that of modal decomposition of data – for example, the classical proper orthogonal decomposition (POD) of flow data generates a set of basis vectors, and a reduced-dimensional representation of the data can be obtained by projection onto a subset of these vectors. (This is a linear dimension reduction process, which always projects data onto a flat surface. Nonlinear approaches like those that we use enable projection onto a curved manifold, and are thus often much more effective at dimension reduction than POD.) A number of specific accomplishments have been achieved.

We have developed a new framework, which we call "Data-driven Manifold Dynamics" (DManD), that enables development of high-fidelity low-dimensional dynamic models for complex chaotic processes. This has been applied to a model system, the Kuramoto-Sivashinsky equation, as well as to transitionally-turbulent plane Couette flow. Having in hand an efficient low-order model enables rapid implementation in downstream analyses such as controller design, and we have integrated DManD with reinforcement learning to drive drag reduction in turbulent Couette flow.

Additionally, we have developed a new modal decomposition that we call the "Data-driven wavelet decomposition" (DDWD). This approach integrates key aspects of POD and wavelet analysis, but iIn contrast to traditional wavelet bases, the basis elements at each stage are not simply dilations of given wavelets, but rather are determined stage-by-stage from the data. For data that is not self-similar, neither are the resulting basis elements. Rather, these represent the differing structures at the different stages. In contrast, for self-similar data, the basis vectors at different stages are related to one another by a simple rescaling. Indeed, for data from homogeneous isotropic turbulence we show self-similarity of the learned wavelet basis elements, which in turn reveals the self-similarity of the data.

Contents

1 Objectives

Turbulent flow is a complex chaotic process governed by a dynamical system, the Navier-Stokes equations (NSE). Although this set of equations is formally infinite dimensional (as with any partial differential equation), the presence of viscosity damps small scales very rapidly, so that we expect the long time dynamics, even of a turbulent flow, to lie on a finite-dimensional surface (manifold) in state space [53, 24]. A central concept in the dynamics of finite-dimensional dissipative dynamical systems is that of a chaotic attractor. Roughly speaking, an attractor is a trajectory in state space toward which all nearby initial conditions approach asymptotically as time $t \to \infty$, and a chaotic attractor is one on which trajectories that are initially nearby will separate from one another exponentially rapidly. From the dynamical point of view, turbulence is a chaotic attractor of the NSE. In the past couple decades, there has been rapid growth in our understanding of the nonlinear dynamics of turbulent flows especially near transition, in part through the study of exact coherent states (ECS). These are nonturbulent coherent patterns that underlie and organize, at least to some extent, the turbulent dynamics. During the course of the grant, the PI wrote an invited review in *Annual Reviews of Fluid Mechanics* on this topic [43].

Parallel to advances in understanding the nonlinear dynamics of turbulence have come dramatic advances, on several fronts, in the field of machine learning. Availability of large amounts of data from the internet, simulations, and experiemnts, combined with improved algorithms and increased computational power have led to great success in the application of machine learning to many areas, including the sciences and engineering. Furthermore, powerful computing environments such as `tensorflow`, developed by Google, have become widely available, lowering the technical barrier to entry into this field.

The overall objective of the work performed under this grant was to integrate the dynamical systems approach to turbulent flows with ideas and tools from machine learning to develop and apply new approaches for the data-driven modeling and control of complex chaotic flow phenomena. Although the data that we have used so far has mainly come from simulation, we are currently beginning to work with experimental data from collaborators at Georgia Tech, Argonne National Laboratory and Princeton. Building on the observation above that very complex high-dimensional dynamics often lie on a manifold of much lower dimension, a central theme of the work is *nonlinear dimension reduction*, using machine learning to identify

the manifold on which the data lie as well as the dynamical equations for the time-evolution of the system dynamics. Related to the issue of dimension reduction is that of modal decomposition of data – for example, the classical proper orthogonal decomposition (POD) of flow data generates a set of basis vectors, and a reduced-dimensional representation of the data can be obtained by projection onto a subset of these vectors. (This is a linear dimension reduction process, which always projects data onto a flat surface. Nonlinear approaches like those that we use enable projection onto a curved manifold, and are thus often much more effective at dimension reduction than POD.) Specific accomplishments, detailed below, include:

- Nonlinear data-driven reduced-order modeling of complex chaotic dynamics
- Integration of data-driven modeling with reinforcement learning for control of chaotic and turbulent dynamics
- Discovering multiscale and self-similar structure in turbulence with data-driven wavelets.

2 Detailed accomplishments

2.1 Nonlinear data-driven reduced-order modeling of complex chaotic dynamics [69, 70]

2.1.1 Introduction

A common question in many applications is: given a time series of measurements on a system how can a predictive model be generated to estimate future states of the system? For problems like weather forecasting, just knowing the future state of the system is useful. In other problems, like minimizing turbulent drag on an aircraft, models are desirable for creating control policies. These models can sometimes be generated from first principles, but insufficient information on the system often limits the ability to write them explicitly. Even when a model can be written out explicitly, it may be very high-dimensional and computationally expensive to simulate. Thus it is often desirable to generate a low-dimensional model from data.

We consider data sets $\{u(t_1), u(t_2), \ldots, u(t_N)\}$, where $u(t_i) \in \mathbb{R}^d$ comes from measuring the state of a system at a given time t_i. The full state at a given time can be represented by direct measurements (e.g. position and velocity of a pendulum) or by a representation that is diffeomorphic to the state. One such representation is time delay measurements of the system, as shown by Takens [105]. In the case of the pendulum, for example, this could correspond to writing the state as $u(t) = [\theta(t), \theta(t-\tau), \theta(t-2\tau), \theta(t-3\tau)]$, where θ is the angle of the pendulum and τ is the delay time. If u lies on a d-manifold then a time delay representation in $\mathbb{R}^{2d+1}$ is diffeomorphic to u [105].

For many systems of interest the state space is high-dimensional and the dynamics are chaotic. Despite this complex behavior, when these systems are dissipative the long-time dynamics often lie on a smooth invariant finite-dimensional inertial manifold $\mathcal{M} \subset \mathbb{R}^d$ of a much lower dimension ($d_{\mathcal{M}}$) than that of the state space [107]. Two such systems are the Kuramoto-Sivashinsky equation (KSE) [35, 108, 57, 121] and the complex Ginzburg-Landau equation [25]. Similarly, for the Navier-Stokes in two and three spatial dimensions, it has been shown that there is an *approximate* inertial manifold [34, 106]. It is approximate in the sense that there are small variations in many dimensions that can be assumed to be zero and still provide an accurate approximation of the state.

With a mapping to the manifold coordinate system the state can be represented (at least locally) in the manifold coordinates $h(t) \in \mathbb{R}^{d_{\mathcal{M}}}$. That is, mappings χ and $\check{\chi}$ exist such that $h = \chi(u)$ and $u = \check{\chi}(h)$. In the machine learning literature, χ and $\check{\chi}$ correspond to the the encoder and decoder, respectively, of a so-called undercomplete autoencoder structure [69], as we further discuss below. It should be noted that there is no guarantee that $\mathcal{M}$ can be *globally* represented with a cartesian representation in $d_{\mathcal{M}}$ dimensions. Indeed in general this cannot be done, and an "atlas" of overlapping local representations, or "charts", must be used

[63]. The application considered here will not require this more general formalism, but for related work using charts, see [32]. Our aim here is to use data from a spatiotemporally chaotic system to learn the mappings $\chi(u)$ and $\check{\chi}(h)$ back and forth between $\mathbb{R}^d$ and a coordinate system on $\mathcal{M}$ and then to learn the evolution equation for the dynamics in this coordinate system. This will be a minimal-dimensional representation for the dynamics of the system in question. We first introduce some background for the dimension reduction problem and then for the time evolution problem.

Dimension reduction methods Dimension reduction is a challenging and widely-studied problem, and many approaches have been considered. Frequently a linear dimension reduction technique is used (i.e. $\mathcal{M}$ is taken to exist in a linear subspace of $\mathbb{R}^d$). This choice can be rationalized by invocation of Whitney's theorem [116]: any smooth $d_{\mathcal{M}}$-manifold can be embedded into $\mathbb{R}^{2d_{\mathcal{M}}}$. Sauer et al. later refined this result by showing this embedding can be performed by almost every smooth map from a $d_{\mathcal{M}}$-manifold to $\mathbb{R}^n$ for $n > 2d_b$ [100], where d_b is the box-counting dimension of the attractor that lies in the manifold $\mathcal{M}$. The box-counting dimension is one of a family of fractal dimensions that can be computed for a given attractor [48], and must be less than or equal to the manifold dimension. Thus, for almost every smooth map, including linear ones, $h \in \mathbb{R}^{2d_{\mathcal{M}}+1}$ contains the same information as u (i.e. a map exists for reconstructing u from h).

Cunningham and Ghahramani [19] give an overview of linear dimension techniques from a machine learning perspective. Many of these collapse to one of the most common methods – principal component analysis (PCA). (In fluid dynamics, PCA is often referred to as Proper Orthogonal Decomposition (POD).) In PCA the linear transformation that minimizes the reconstruction error or maximizes the variance in the data is found. This transformation comes from projecting data onto the leading left singular vectors $U \in \mathbb{R}^{d\times d_h}$ of the centered snapshot matrix $X = [u(t_0) - \langle u\rangle, ..., u(t_N) - \langle u\rangle]$ (here $\langle . \rangle$ denotes ensemble average). This is equivalent to finding the eigenvectors of the covariance matrix XX^T. Similarly, the eigenvectors of X^TX can be found and related to the eigenvectors of XX^T through the singular value decomposition. This approach is sometimes known as classical scaling [111]. The projection onto these modes is $\tilde{u} = UU^Tu$, where superscript tilde denotes the approximation of the state u. In this projection, $h = U^Tu$ is the low-dimensional representation. Although this projection is the optimal linear transformation for minimizing the reconstruction error, h may still require as many as $2d_{\mathcal{M}} + 1$ dimensions to contain the same information as u.

Finding a *minimal* representation in general requires a nonlinear transformation. Many different techniques exist for nonlinear dimension reduction, often with the goal of preserving some property of the original dataset in the lower dimension. Some popular methods include kernel PCA, diffusion maps, local linear embedding (LLE), isometric feature mapping (Isomap), and t-distributed stochastic neighbor embedding (tSNE) [111]. In kernel PCA the matrix X^TX is replaced with a kernel matrix $K(u_i, u_j)$. This can be viewed as an application of the "kernel trick" on the the covariance matrix between data that has been mapped into an higher-dimensional (often infinite-dimensional) "feature space" [111]. The low-dimensional representation is then computed using the eigenvectors such that $h = [\sum_i^d v_{i,1}K(u_i, u), ..., \sum_i^d v_{i,d_h}K(u_i, u)]$, where $v_{i,k}$ is the ith element of the kth eigenvector of $K(u_i, u_j)$.

Similarly, diffusion maps, LLE, and Isomap can be viewed as extensions of kernel PCA with special kernels [111]. In diffusion maps a Gaussian kernel is applied to the data giving the matrix $A_{ij} = \exp(-||u_i - u_j||/2\varepsilon)$, where ε is a tuning parameter that represents the local connectivity [31]. Then the dimension reduction is performed by computing the eigenvalue decomposition of this matrix (normalized so columns add to one) giving $h_i = [v_{i,2}, ..., v_{i,d_h+1}]$. The first eigenvector is the trivial all-ones vector [31]. In LLE, a linear combination of the k nearest neighbors to a data point are used to minimize $\sum_i ||u_i - \sum_j W_{i,j}u_j||^2$, where $W_{i,j} = 0$ if u_j is not a neighbor of u_i. Then the low-dimensional representation h is calculated to minimize $\sum_i ||h_i - \sum_j W_{i,j}h_j||^2$ using the W calculated in the first step [97]. The solution that minimizes this cost function can be found by computing the eigenvectors of $(I - W)^T(I - W)$, and let-

ting $h_i = [v_{i,1}, ..., v_{i,d_h}]$. For Isomap, the kernel matrix comes from computing the double centered geodesic distances [109, 17] $K(u_i, u_j) = -1/2HD(u_i, u_j)^2H$. Here $D(u_i, u_j)$ is the geodesic distance which is computed in two steps. First, a graph of the k nearest neighbors between points is constructed, weighted by their Euclidean distances, then the distance between points is computed to be the shortest pathway between any points along the graph. After computing D, the matrix is centered with $H = 1/NI - 1 \cdot 1^T$ $(1 = [1, ..., 1]^T)$, and the eigenvalue decomposition of K gives $h_i = [\sqrt{\lambda_1}v_{i,1}, ..., \sqrt{\lambda_{d_h}}v_{i,d_h}]$. Isomap was used to reduce the dimension of many dynamical systems in [6].

The last algorithm we mention here is tSNE [49]. Unlike the previous methods, which use the eigenvalue decompositions to solve the optimization problem, tSNE uses gradient descent. In tSNE, the objective is to match the distribution of the data in the high-dimensional space to the data in the low-dimensional space. This is done by finding h that minimizes the Kullback-Leibler (KL) divergence $\sum_i \sum_j p_{ij} \log p_{ij}/q_{ij}$, where p and q are the distributions of the high- and low-dimensional data respectively. In tSNE these distributions are approximated by $p = \exp(-||u_i - u_j||^2/2\sigma^2)/\sum_{k\neq l} \exp(-||u_k - u_l||^2/2\sigma^2)$ and $q = (1 + ||h_i - h_j||^2)^{-1}/\sum_{k\neq l}(1 + ||h_k - h_l||^2)^{-1}$. There is no guarantee in finding a global minimum or in finding the same solution every time. tSNE is frequently used for visualizing high-dimensional data in two or three dimensions, because it often separates complex data sets out into visually distinct clusters.

A few major drawbacks of these nonlinear dimension reduction techniques are they do not provide the function $\check{\chi}$, which reconstructs u from h, they do not provide a method for adding out-of-sample data, and they do not scale well for large amounts of data. Except for tSNE, the Nyström extension can be used to resolve the out-of-sample problem for these methods [4]. However, using the Nyström extension results in different functions for χ depending on whether the data is in-sample or out-of-sample.

Because of these limitations, instead of using one of the above techniques, we tackle the dimension reduction problem directly, by approximating the mappings to the manifold coordinates χ and back $\check{\chi}$ as NNs. I.e., we use an undercomplete autoencoder [50, 55]. We describe autoencoders in more detail in Section 2.1.2.

Time evolution methods We now turn to developing, from data, dynamical models for the evolution of a state $u(t)$. We consider systems that display deterministic, Markovian dynamics, so if $u(t)$ is the full state (either directly or diffeomorphically through embedding of a sufficient number of time delays), then the dynamics can either be represented by a discrete time map

$$u(t + \tau) = F(u(t)), \tag{1}$$

or an ordinary differential equation (ODE)

$$\frac{du}{dt} = f(u). \tag{2}$$

The learning goal is an accurate representation of F or f, or their lower-dimensional analogues in the case that we apply dimension reduction.

We consider first the discrete-time problem. This is easier because that is the format in which we usually have data. For simple systems that decay to a fixed point or display quasiperiodic dynamics with discrete frequencies, linear discrete-time methods, like dynamic mode decomposition (DMD) [61], can predict dynamics. This type of idea can be extended to nonlinear systems when there is a change of basis which makes the dynamics linear. In Lusch et al. [73] it was shown that an autoencoder can learn this change of basis. However, this approach has not been used to predict the long-time behavior of chaotic systems, which is our interest here. More generally, for every dynamical system, there is a linear, but infinite-dimensional *Koopman operator* that describes the evolution of an arbitrary observable [11]. Our goal is to reduce dimension, so we do not take this approach.

Predicting chaotic dynamics in state space requires a nonlinear model. One successful class of ap-

proaches for doing this includes reservoir networks [88] and recurrent neural networks (RNN) [114, 113]. These methods use discrete time maps, are non-Markovian, and typically increase the dimension of the state space. Reservoir networks work by evolving forward a high-dimensional reservoir state $r(t) \in \mathbb{R}^{d_r}$, and finding a mapping from r to u. This reservoir state is evolved forward in time by some function $r(t+\tau) = G(r(t), W_{\text{in}}u(t))$, where G and W_{in} are chosen a priori. Then, the task is finding the optimal parameters p in $\tilde{u}(t+\tau) = W_{\text{out}}(r(t+\tau); p)$ that minimize $\left\langle ||u(t+\tau) - \tilde{u}(t+\tau)||^2 \right\rangle$. This is non-Markovian because the prediction of the state $u(t+\tau)$ depends on the previous $u(t)$ and $r(t)$. In Pathak et al. [88] a reservoir network was used to predict the chaotic dynamics of the KSE. For data with a state space dimension $d = 64$, they choose a reservoir dimension $d_r = 5000$. That is, here the dimension of the representation has not been reduced, but rather expanded, by two orders of magnitude. An interesting way to address this issue is to realize the reservoir architecture in hardware rather than software [12].

Similar to reservoir networks, RNNs have a hidden state $h_r \in \mathbb{R}^{d_{h_r}}$ that is evolved forward in time. The hidden state is evolved forward by $h_r(t+\tau) = \sigma_h(u(t), h_r(t); W_h)$, and the future state is estimated from the hidden state $\tilde{u}(t+\tau) = \sigma_u(h_r(t+\tau); W_u)$. The functions σ_h and σ_u take different forms depending upon the type of RNN – two examples are the long short-term memory (LSTM) [51] and the gated recurrent unit [16]. Regardless of architecture, the functions σ_h and σ_u are constructed from NN with parameters W_h and W_u. These parameters come from minimizing the same loss as in reservoir computing $\left\langle ||u(t+\tau) - \tilde{u}(t+\tau)||^2 \right\rangle$. Vlachas et al. [113] provide a comparison between reservoir networks and RNN for chaotic dynamics of the KSE. Both provide predictive capabilities for multiple Lyapunov times, but, as noted, are high-dimensional and non-Markovian. Additionally, these methods typically require evenly spaced data for training, and start-up often requires multiple known states, instead of a single initial condition.

In contrast to these high-dimensional non-Markovian approaches to learning a discrete time mapping we have shown in prior work [69] that a dense NN can approximate $F(u)$. This approach is advantageous because, like the underlying system, it is Markovian – predictions of the next state only depends on the current state. Unlike the previous methods, this approach also drastically reduced the dimension of the state by representing it in terms of the manifold coordinates. For example, for chaotic dynamics of the KSE with a domain size of $L = 22$, we showed that the state $u \in \mathbb{R}^{64}$ could be represented by $h \in \mathbb{R}^8$. We found this coordinate system using an undercomplete autoencoder that corrected on PCA, as described in more detail in Section 2.1.3. Representing the state in this minimal fashion allows for faster prediction, and may provide insight by limiting the system to only the essential degrees of freedom. Iten et al. [56] illustrated the latter point by recovering physical variables from data in some low-dimensional example settings.

Rather than approximating $F(u)$ in Eq. 1, the discrete-time representation, one can, in principle, approximate $f(u)$ in Eq. 2, the continuous-time representation. This is more challenging, because one does not generally have access to data for du/dt. When the time derivative du/dt is known (or estimated from closely spaced data) for all of the states, Gonzalez Garcia et al. [40] showed that a dense NN can provide an accurate functional form of f for the KSE in a regime that exhibits a periodic orbit. In their work they input the state and its derivatives into the NN. A similar data-driven method that requires du/dt is "Sparse Identification of Nonlinear Dynamics" (SINDy) [8]. In this approach, a dictionary of candidate nonlinear functions are selected to represent $f(u)$, and sparse regression is used to identify the dominant terms based on data for both u and du/dt. This idea has been extended for use with autoencoders for dimension reduction, while still requiring time-derivative data, in some cases with invariant manifolds of dimension three or smaller [14]. In a distinct approach, Raissi et al. [92] showed that du/dt can be approximated from closely-spaced data with a multistep time-integration scheme such as the second-order Adams-Bashforth formula, and a NN can be trained to match this approximation.

Now we turn to the more general situation, where one desires a data-driven ODE representation, but does not have data on time derivatives or state data sufficiently closely spaced to accurately approximate them. Now the task of learning $f(u)$ is closely related to the problem of *data assimilation* [3]. In that problem, the general functional form of f is given and the task is to learn a relatively small number of parameters, whereas

in the present case we wish to represent $f(u)$ as an essentially arbitrary function. A framework for doing this, with f given as a neural network, was presented by Chen et al. [15], who dubbed the approach "neural ODEs". To determine the neural network parameters of f, the difference between data and predictions is minimized. To determine the derivatives of f with respect to the parameters, either an adjoint problem can be solved, or an automatic differentiation method used. With a given f, the state evolution at arbitary points in time can be computed as the solution to Eq. 2. We further describe, and apply, this approach in Section 2.1.2.

Neural ODEs have been applied to a number of time series problems. Maulik et al. [76] used neural ODEs and LSTMs for Burgers equation, and showed that both outperform a Galerkin projection approach. Portwood et al. [91] used the neural ODE approach to find an evolution equation for dissipation in decaying isotropic turbulence, showing that it outperformed a heuristic model. Neural ODEs have also been applied to flow around a cylinder in the time-periodic regime, where velocity field simulation data were projected onto 8 PCA modes, and the time evolution of these modes was determined [96].

The present work combines nonlinear dimension reduction using autoencoders with the neural ODE method to model the dynamics of a system displaying spatiotemporal chaos, the Kuramoto-Sivashinsky equation, over a range of parameter values. In Section 2.1.2, we introduce the framework of our methodology. Section 2.1.3 briefly describes the results for the dimension reduction problem alone, then Section 2.1.3 uses the reduced-dimensional descriptions to illustrate performance of the neural ODE approximation for closely spaced data. An important conclusion here is that dimension reduction can improve neural ODE performance relative to predictions in the ambient space, where artifacts arise. Section 2.1.3 examines the role of data spacing on neural ODE performance, showing that even for chaotic systems, good performance on both short-time predictions and long-time statistics can be obtained even from widely spaced data, up to a fairly well-defined limit. Finally, Section 2.1.3 shows comparisons of neural ODE performance with various degrees of dimension reduction, finding a "sweet spot" in terms of performance vs. dimension. We summarize in Section 2.1.4.

2.1.2 Framework

We consider data $u \in \mathbb{R}^d$ that lies on a $d_{\mathcal{M}}$-dimensional manifold $\mathcal{M}$ that is embedded in the ambient space $\mathbb{R}^d$ of the data. With the data, we find a coordinate transformation $\chi : \mathbb{R}^d \to \mathbb{R}^{d_{\mathcal{M}}}$ giving the state in the manifold coordinates, $h \in \mathbb{R}^{d_{\mathcal{M}}}$. We also find the inverse $\check{\chi} : \mathbb{R}^{d_{\mathcal{M}}} \to \mathbb{R}^d$ to reconstruct u from h. Then we describe the dynamics on $\mathcal{M}$ with the differential equation

$$\frac{dh}{dt} = g(h). \tag{3}$$

If we do not do any dimension reduction then Eq. 3 is the same as Eq. 2. Of course, for an arbitrary data set, we do not know $d_{\mathcal{M}}$ a priori, so in Section 2.1.3 we present results with various choices for d_h, where $h \in \mathbb{R}^{d_h}$. Using the mapping to the manifold coordinates, the evolution in the manifold coordinates, and the mapping back, we evolve new initial conditions forward in time. So, our task is to approximate χ, $\check{\chi}$, and g. We also consider the case with no dimension reduction, where we determine $f(u)$, the right hand side (RHS) of the ODE in the ambient space. In general, there can be no expectation that any of these functions have a simple (e.g. polynomial) form, so here we approximate them with NNs, as detailed below.

Figure 1 illustrates this framework on data from a solution of the Lorenz equation [72] that we embedded in four dimensions by mapping the z coordinate of the Lorenz equation to the Archimedean spiral. The change of coordinates for this embedding is given by $[u_1, u_2, u_3, u_4] = [x, y, \alpha z \cos \alpha z, \alpha z \sin \alpha z]$ where x, y, z are the standard variables in the Lorenz equation and $\alpha = 0.2$. In Fig. 1a we show an example of learning the manifold coordinates for this system. The three spatial dimensions for the embedded data are u_1, u_3, and u_4 and the color is u_2. For a trajectory to be chaotic it must lie on at least a three-dimensional manifold, so

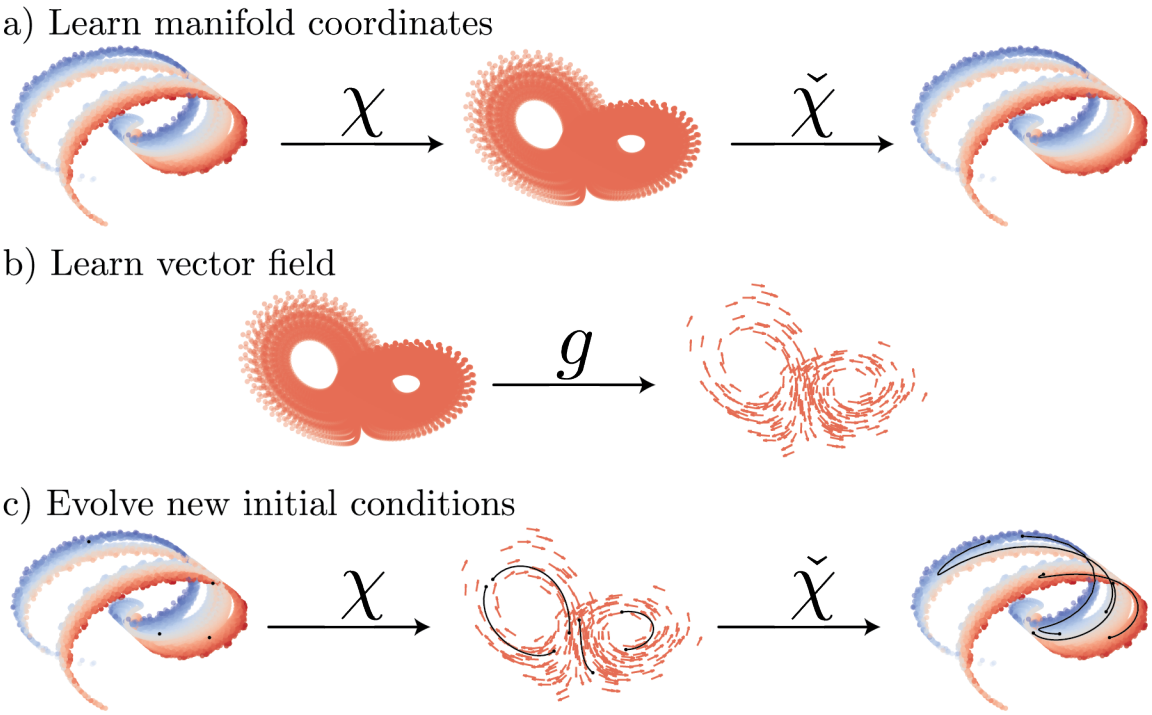

Figure 1: (a) Learning the three-dimensional manifold coordinates of the four-dimensional Lorenz butterfly wrapped on the Archimedean spiral (color is the fourth dimension). (b) Learning the vector field in the manifold coordinates. (c) Transforming new initial conditions (black dots) into the manifold coordinates, evolving them according to the learned vector field g (black curves), and transforming back into the original space.

the minimum embedding dimension for a chaotic trajectory that requires all the steps in this framework is four dimensions. Figure 1b illustrates learning the vector field g in the manifold coordinates. In Fig. 1c we show how, after learning these functions, new initial conditions can be mapped to the manifold coordinates, evolved forward in time, and then mapped back to the ambient space. we denote this method as "Data-driven Manifold Dynamics", or "DManD".

To find χ and $\check{\chi}$, we represent them as NNs and find parameters that minimize a reconstruction loss averaged over a batch of data given by

$$L = \left\langle ||u - \check{\chi}(\chi(u;\theta_1);\theta_2)||^2 \right\rangle, \tag{4}$$

where θ_1 and θ_2 are the weights of χ and $\check{\chi}$, respectively. Here, and elsewhere, we use stochastic gradient descent methods to determine the parameters. Further details are given in Section 2.1.3. This architecture is known as an undercomplete autoencoder, where χ is an encoder and $\check{\chi}$ is a decoder [55]. Autoencoders are widely used for dimension reduction; in the fluid mechanics context they have been used for many examples including flow around an airfoil [84], flow around a flat plate [81], Kolmogorov flow [86], and channel flow [80, 37].

As noted above, to approximate g in Eq. 3, we use the neural ODE approach of Chen et al. [15]. In the neural ODE framework we use g to estimate $\tilde{h}(t_i + \tau)$, an approximation of the reduced state $h(t_i + \tau)$, at

some time $t_i + \tau$ by integrating

$$\tilde{h}(t_i + \tau) = h(t_i) + \int_{t_i}^{t_i+\tau} g(h(t); \theta_3) dt, \tag{5}$$

where θ_3 is the set of weights of the NN. Then g is learned by minimizing the difference between the prediction of the state ($\tilde{h}(t_i + \tau)$) and the known state ($h(t_i + \tau)$), in the manifold coordinates, at that time. Specifically, the loss we minimize is

$$J = \left\langle ||h(t_i + \tau) - \tilde{h}(t_i + \tau)||_1 \right\rangle. \tag{6}$$

Here $||.||_1$ denotes the L_1-norm – of course other norms can be used. By taking this approach we can estimate g from data spaced further apart in time than when g (or more precisely, dh/dt) is estimated directly from finite differences. The difficulty comes in determining the gradient of the loss with respect to the weights of the NN, $\partial J/\partial \theta_3$.

One approach to computation of $\partial J/\partial \theta_3$ is to use automatic differentiation to back-propagate through all of the steps of the time integration scheme used to solve Eq. 5. Another approach involves solving an adjoint problem backwards in time, which is the primary method used in [15], and indeed is the classical approach to optimization problems involving differential equation constraints [3]. A drawback of back-propagating through the solver is that to calculate the gradient, the entire state must be stored at each time step, which can lead to memory issues [15]. However, we consider a chaotic system which puts an implicit constraint on how far apart data can feasibly be sampled (roughly one Lyapunov time) and still yield a good estimate of g. In the present work, we reach this limit before the memory limit of back-propagation becomes an issue. In our trials, the adjoint method yielded results in good agreement with back-propagation, but required longer computation times for training, so we chose to use the back-propagation approach.

Note that we have separated the training problems for determining the manifold coordinates (i.e. the dimension reduction problem) from that for learning the dynamics in those coordinates. In principle, one could have put those problems together, minimizing a single loss that is a linear combination of Eq. 4 and Eq. 6. Indeed, Champion et al. [14] did something similar, though it must be noted that their approach requires data for time derivatives, while ours does not, and also that they knew in advance the dimensions of the manifolds where their data lives, which we do not. And in any case, the invariant manifold dimensions in their examples were three or less. There are two reasons why we treated the two problems separately. The first is conceptual. The question of finding the manifold on which a data set lies is a self-contained problem. It seems natural to address this problem first, and only afterward address the question of how the dynamics evolve on that manifold. The second reason is more practical. Even if one knows a priori the dimension of the manifold, solving simultaneously for the autoencoder and time evolution means simultaneously solving for three neural networks with a very complicated loss landscape. Indeed, when we attempted to train both models together the predictive capabilities were extremely poor. Furthermore, in the problems we address, the manifold dimension is *not* known a priori, so trying to learn a vector field (RHS of an ODE) when one does not even have an idea of how many dimensions that vector field should have may lead to much wasted computer time. Therefore, we take the view that it makes more sense to first get an estimate of how many dimensions are required to represent the data, and only then to estimate the vector field that determines the dynamics.

The data sets we consider in this section[1] are numerical solutions of the 1D Kuramoto-Sivashinsky equation (KSE),

$$\frac{\partial v}{\partial t} = -v\frac{\partial v}{\partial x} - \frac{\partial^2 v}{\partial x^2} - \frac{\partial^4 v}{\partial x^4}, \tag{7}$$

[1] Extension of this approach to turbulent Couette flow and its control are described in Section 2.2.5.

in a domain of length L with periodic boundary conditions. The domain size determines the types of dynamics this system exhibits. For the domain sizes we consider, trajectories exhibit sustained chaos. The dimension of the manifold that contains the global attractor has been computationally approximated using several approaches [119, 23, 69]. We generate high-dimensional state representations ($u \in \mathbb{R}^d$) via a Galerkin projection of v onto Fourier modes. Then, we use an exponential time differencing method [58] to integrate the resulting system of ordinary differential equations forward in time. The code used is available from Cvitanović et al. [20]. The data vectors u that we use are solutions v of the KSE on $d = 64$ equally-spaced grid points in the domain. We performed resolution checks on a grid of $d = 128$ and found, for the domain sizes we consider, trajectories from the same initial condition track one another for multiple Lyapunov times.

2.1.3 Results

Section 2.1.3 briefly describes the dimension reduction (manifold representation results). Section 2.1.3 considers neural ODE predictions, with and without dimension reduction, for closely spaced data. Section 2.1.3 shows the effect of data spacing on the results, and Section 2.1.3 illustrates the effect of the degree of dimension reduction. We summarize in Section 2.1.4.

We consider datasets for three domain sizes, $L = 22$, 44, and 66, where we estimate the manifold dimension to be $d_{\mathcal{M}} = 8$, 18, and 28, as explained in the following section. The relevant timescale for each of these system is the Lyapunov timescale (inverse of the largest Lyapunov exponent), which we previously found to be be very close to the integral time for KSE data [69]. For these domain sizes, the Lyapunov times are $\tau_L = 22.2$, 12.3, and 11.6, respectively [23, 26]. We use 10^5 time units of data for training at each domain size, and vary how frequently we sample this dataset in time. For training the NNs we use Keras [18] for the autoencoders and PyTorch [87] for the neural ODEs. The neural ODE code is a modification on the code used in [15], which includes methods for computing the gradient both with the adjoint method and back-propagation.

Dimension reduction with autoencoders In each section we use autoencoders to approximate the map to the manifold coordinates χ and back $\check{\chi}$. We found in [69] that a useful way to represent the map to the manifold coordinates ($h = \chi(u)$) is as a difference from a linear projection of the data onto the leading d_h PCA modes:

$$h = E(U^T u; \theta_1) + P_{d_h} U^T u. \tag{8}$$

Here E is a NN, U is the full PCA matrix, and P_{d_h} is the projection onto the leading d_h modes. Similarly, we can learn a difference for the decoder ($\tilde{u} = \check{\chi}(h)$)

$$\tilde{u} = UD(h; \theta_2) + U \begin{bmatrix} h \\ 0 \end{bmatrix}, \tag{9}$$

where D is a NN. By taking this approach we simplify the problem such that the NN only needs to correct upon PCA. We refer to this as a hybrid NN (HNN).

In [69] we also took advantage of the translational invariance of the problem. We used the first Fourier mode method-of-slices [10] to phase shift the data. In this method the phase is calculated as

$$\varphi(t) = \text{atan2}\{\text{Im}[a_1(t)], \text{Re}[a_1(t)]\},$$

where a_1 is the projection of the state onto the first Fourier mode. Then, the state is phase shifted at each time using the Fourier shift theorem $\hat{u}_a(t) = a(t)e^{-ik\varphi(t)}$. This approach removes the continuous symmetry, which reduces the dimension of the problem by one. More precisely, it decouples the evolution of the phase φ from the evolution of the pattern $\hat{u}_a$ – the phase evolution depends on $\hat{u}_a$ but not vice versa. This reduction

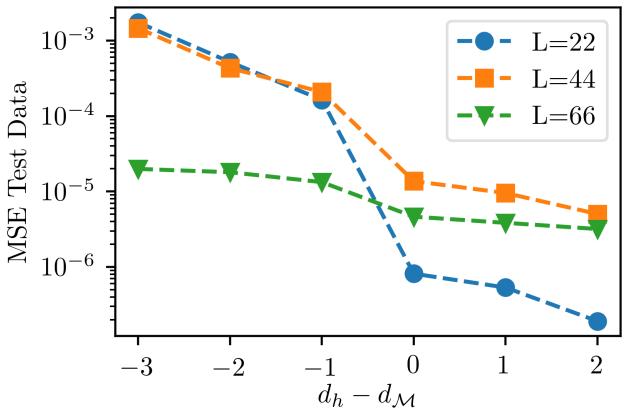

Figure 2: MSE of test data as a function of dimension for domain sizes $L = 22$, 44, and 66. The expected manifold dimensions are $d_{\mathcal{M}} = 8$, 18, and 28 for these domain sizes, respectively.

in dimension improves the performance of the autoencoder, but introduces near-discontinuities in the flow that arise when a_1 approaches zero that must be accounted for by warping time with "in-slice" time [10]. These near-discontinuities add additional difficulties in training the time evolution models, so we did not take this approach here.

In [69] we trained these HNN using an Adam optimizer with a learning rate of 0.001, for multiple dimensions d_h. Figure 2 shows the mean squared reconstruction error (MSE) of a test data set not used in training at various dimensions for each of the domain sizes. When using this hybrid approach we see a significant drop in the MSE at dimensions of $d_h = 8$, 18, and 28 for the domain sizes $L = 22$, 44, and 66, respectively. We take these values of d_h to be the "correct" dimension, $d_{\mathcal{M}}$, in the Sections 2.1.3 and 2.1.3. It is worth noting that finding this drop requires careful hyperparameter tuning. For example, we found using rectified linear units rather than sigmoids resulted in higher MSE for $d_h \gtrsim d_{\mathcal{M}}$ while having little effect on $d_h < d_{\mathcal{M}}$, making the drop in MSE less evident. Due to difficulties in autoencoder training, and due to the increased complexity of the system at larger domain sizes it can be difficult to determine the dimension from the drop in the autoencoder MSE alone. In Section 2.1.3, we further examine the issue of estimating manifold dimension by considering the performance of neural ODE models as a function of dimension – if the estimate of $d_{\mathcal{M}}$ is poor (especially on the low side) then the dynamic model is poor.

The estimated dimension at $L = 22$ of $d_{\mathcal{M}} = 8$ is in agreement with estimates by Ding et al. [23], Vlachas et al. [112], Yang and Radons [118] and Cvitanovic et al [21]. Ding et al. reached this conclusion by considering a finite number of unstable periodic orbits embedded in the long-time attractor and performing Floquet analysis on these solutions. Vlachas et al. also used an autoencoder approach of varying dimension and considering the MSE at different dimensions. In that work, they used different autoencoder architectures (e.g. convolutional autoencoder), with different activation functions (continuously differentiable exponential linear unit), and still found the same drop at $d_h = 8$. Yang and Radons [118] showed that the Lyapunov vectors can be split into two categories by looking for a drop in the Lyapunov exponents. The positive and weakly negative Lyapunov exponents give Lyapunov vectors called "physical modes" that were shown to span a linear space that locally approximates the inertial manifold [118]. This drop in Lyapunov exponents was found to be at $d_h = 8$ in [21] and at $d_h = 9$ in [118].

For larger domains, the dimension estimates reported here are consistent with Yang et al. [119]. In that work they tracked the number of physical modes for larger domain sizes of the KSE. We showed in [69] that our estimates of $d_{\mathcal{M}}$ agree with the predicted dimensions and linear scaling of dimensions shown in Yang et al. [119]. Linear scaling is also seen in Sondak and Protopapas [102], though their estimate of the manifold

Table 1: Architectures of NNs and matrices used in Sections 2.1.3-2.1.3. "Shape" indicates the dimension of each layer, and "activation" the corresponding activation functions (S is the sigmoid activation) [55].

	Function	Shape	Activation
Encoder $L = 22, 44, 66$	χ	$d : d_{\mathcal{M}}$	linear
Decoder $L = 22, 44$	$\check{\chi}$	$d_{\mathcal{M}} : 500 : d$	S:linear
Decoder $L = 66$	$\check{\chi}$	$d_{\mathcal{M}} : 500 : 500 : d$	S:S:linear
ODE	f, g	$d, d_{\mathcal{M}} : 200 : 200 : 200 : d, d_{\mathcal{M}}$	S:S:S:linear
Discrete Map	G	$d_{\mathcal{M}} : 200 : 200 : 200 : d_{\mathcal{M}}$	S:S:S:linear

dimension is higher than ours or any of the other works mentioned here. They approximated dimension by training autoencoders with a high-dimensional latent space, and then considered the variance of each latent variable. Then $d_{\mathcal{M}}$ was taken to be the number of latent variables with a variance above some threshold.

These results guide our selection of dimension in the next two sections. Furthermore, we find that once $d_h \gtrsim d_{\mathcal{M}}$, the same MSE is achieved whether we use the full HNN or simply project on to $d_{\mathcal{M}}$ PCA modes (by setting $E(U^{\mathrm{T}}u) = 0$ in Eq. 8). This observation indicates, for the KSE with periodic boundary conditions, that the leading PCA amplitudes fully parameterize the state. Note that this parametrization is *nonlinear*, because a nonlinear decoder is still needed to reconstruct the full state. So, in the next two sections, χ is the projection onto $d_{\mathcal{M}}$ PCA modes and $\check{\chi}$ is a NN that approximates the remaining PCA coefficients from the leading ones. Table 1 shows the architectures for the autoencoders and for the NNs used for time evolution in Sections 2.1.3 and 2.1.3.

Effect of Dimension Reduction on Time Evolution Now that we have a map to the manifold coordinates and back, the next step is approximating the ODE. Here we train three different types of ODEs to evaluate the impact of mapping the data to the manifold coordinates. We learn the RHS of the ODE in the manifold coordinates: $\dot{h} = g(h)$, in physical space: $\dot{u} = f(u)$, and in the space of the spatial Fourier coefficients: $\dot{\hat{u}} = \hat{f}(\hat{u})$, where $\hat{u} = \mathcal{F}(u)$ and $\mathcal{F}$ is the discrete Fourier transform. For the last two cases there is no dimension reduction, so χ is the identity in the first case and the discrete Fourier transform in the second, with $\check{\chi}$ being, again, the identity in the first case and the discrete inverse Fourier transform in the second.

We train each of the NN with data spaced 0.25 time units in these trials. This timescale is substantially shorter than the Lyapunov time, which decouples the issue of dimension from that of time resolution. Each NN is trained until the loss levels off using an Adam optimizer with a learning rate of 10^{-3} that we drop to 10^{-4} halfway through training. First the autoencoder is trained, then the neural ODE. To avoid spurious results, we train 5 autoencoders then 10 ODEs, and select the combination of autoencoder and ODE that provides the best short-time tracking. Due to random weight initialization and the stochastic nature of the optimization, we train multiple models and use the one that performs best. This is a common practice in neural network training, as noted by Bishop in Section 9.6 of [5]. As mentioned earlier, the models are trained to minimize the loss in Eq. 6. In previous work [69], we added an additional contribution to the loss for the time evolution problem that penalizes predicted states that fall outside the envelope of the joint PDF for energy production and dissipation. In the present work we found excellent performance of the best models without the need for this additional term, because we did not phase-shift the data and thus did not have to work in "in-slice" time.

In Fig. 3 we compare the short- and long-time trajectories between the data at $L = 22$ (Fig. 3a) and the NN models (Fig. 3b-d). For this domain size, the Lyapunov time is $\tau_L = 22.2$. The same initial condition, which is on the attractor, is used for all cases. The first model prediction, shown in Fig. 3b, comes from g (the low-dimensional ODE, with $d_h = 8$), the second model prediction, shown in Fig. 3c, comes from f (the full physical space ODE), and the third model prediction, shown in Fig. 3d, comes from $\hat{f}$ (the full Fourier space ODE). For each of these figures, the first 50 time units (about 2.5 Lyapunov times) are shown in the

left column and 450-500 time units (i.e. after the solution has evolved for more than 20 Lyapunov times) are shown on the right.

All three of the models are able to generate good predictions for times up to about $t = 30$; this is a reasonable quantitative prediction horizon for a data-driven model of chaotic dynamics. After many Lyapunov times, a good model should still yield behavior consistent with the data. Indeed, at long times the reduced model no longer matches the true solution, but continues to exhibit realistic dynamics, as shown in Fig. 3b. In the next section we discuss the long-time behavior of the reduced model in more detail. In contrast, the full state model predictions at long times, shown in Figs. 3c and 3d, develop high wavenumber striations , which eventually pollute the lower wavenumbers, and are thus not faithful to the dynamics. This same behavior appears when modeling the full state evolution for the larger domain sizes $L = 44$ and 66. Training high-dimensional neural ODEs is computationally more expensive, and the predictions from these models are worse.

To diagnose the origin of the spurious high wavenumber behavior, we plot the magnitude of the Fourier modes as a function of time in Fig. 4 over a very long time interval – hundreds of Lyapunov times. In both cases, we observe a very slow linear growth in the high-wavenumber modes. The inset in Fig. 4b shows the time evolution of the RHS of the ODE for the real part of one of the high-wavenumber modes $\hat{f}_{30}$. At long-times $\hat{f}_{30}$ settles to a constant – effectively a bias term. The time-integration of this constant results in the linear growth seen in Fig. 4. We tested multiple activation functions in the NN architecture, and we see growth in the high wavenumbers regardless of whether sigmoids, hyperbolic tangents, or rectified linear units are used.

To further explain the origin of the bias that leads to the linear growth, we first note that in the true system the high wavenumbers are strongly damped due to hyperdiffusivity in the KSE, so data sampled from the attractor has negligible content in these wavenumbers. In particular, with the data given, the model has no information with which to learn that these modes are strongly damped. Furthermore, the prediction horizon in the training of the neural ODE is short (0.25 here), so small errors in high wavenumbers have a negligible effect on the loss. Thus the model does not learn that there should be a strong damping term that would overwhelm any small bias that arises in the RHS due to the stochastic nature of the initialization and training of neural net representations. Indeed, experiments that explicitly introducing a damping term, modifying Eq. 3 to read $dh/dt = g(h;\theta_3) - ah$ where $a > 0$, is sufficient to stabilize models against this slow linear growth. In [68] we show this is the case when the damping is chosen to match the linear term in the KSE. The combination of these factors is why the trajectories of the high-dimensional models leave the attractor, while the trajectories of the reduced-dimensional models, where degrees of freedom that are strongly damped and thus have negligible amplitude on the attractor are absent, do not.

Effect of Temporal Data Spacing on Time Evolution Prediction Now we consider only reduced-dimension ODE models, and address the dependence of model quality on the temporal spacing τ between data points. For $L = 22$, 44, and 66, we select the manifold dimensions mentioned in Section 2.1.3 – $d_{\mathcal{M}} = 8$, 18, and 28, respectively. We judge the model performance based on, first, short-time tracking for $L = 22$ and then long-time statistical agreement for all domain sizes. In these cases autoencoders and ODEs were both trained with τ much larger than the value of 0.25 used in Section 2.1.3, while keeping all other training parameters the same (e.g. architecture, optimizer, dimension). Figure 5 compares a true short-time trajectory (5a) to the model prediction (with $\tau = 10$) of this trajectory (5b) starting from the same initial condition. Qualitatively the space-time plots exhibit similar behavior over around 80 time units. The magnitude of the difference is shown in 5c. Here we see the difference growing at around 40 time units due primarily to a growing spatial phase difference between the true and predicted results. Figure 5d shows the norm of this difference and the minimum value of the norm when shifting $\tilde{u}$ to any position in the domain. The minimum translated difference, in 5d, remains much smaller than the standard Euclidean distance indicating that, in this case, the

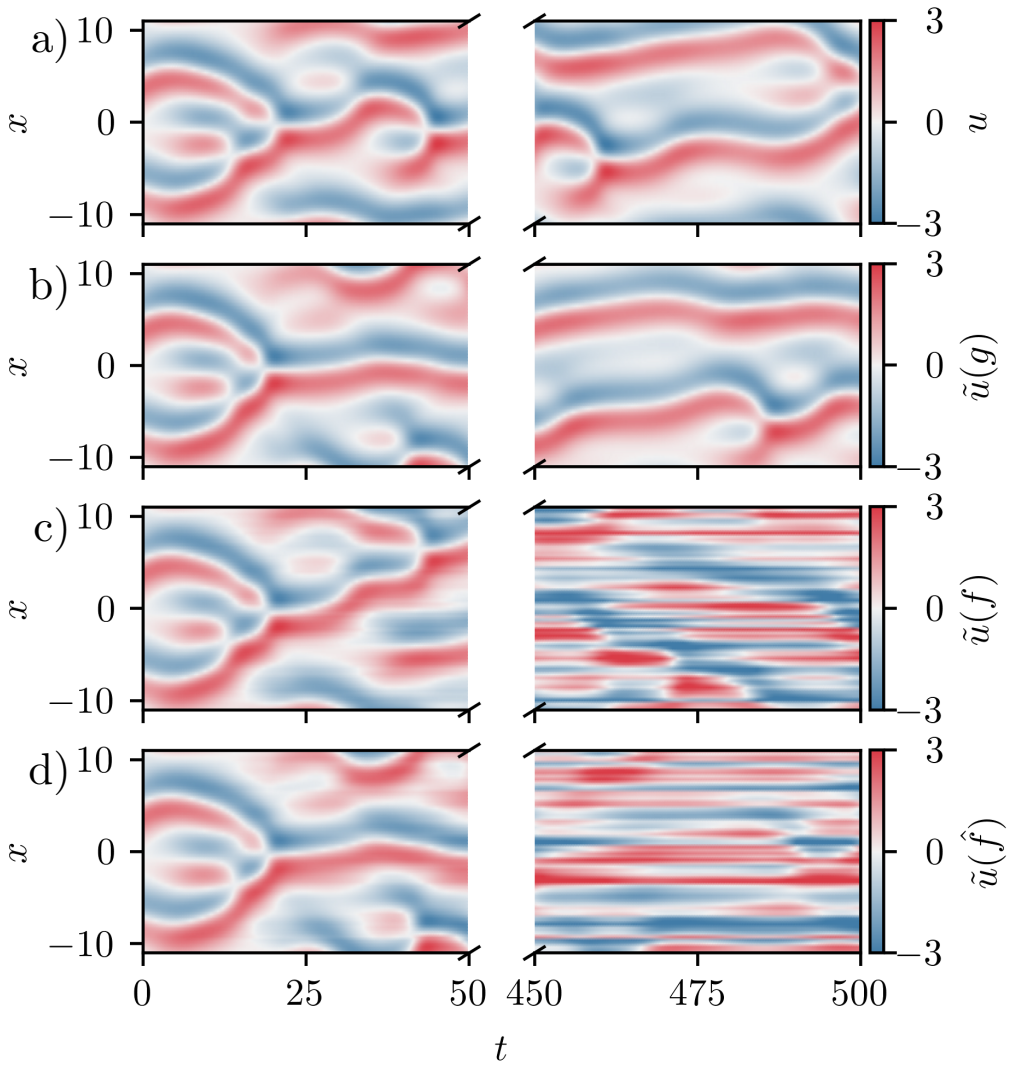

Figure 3: Examples of trajectory predictions for different models, $L = 22, \tau = 0.25$. Left column shows predictions up to ~ 2.3 Lyapunov times, and right shows predictions after ~ 23 Lyapunov times. (a) true trajectory, (b) predicted trajectory when approximating g with $d_{\mathcal{M}} = 8$. (c) predicted trajectory when approximating f. (d) predicted trajectory when approximating $\mathcal{F}(f)$.

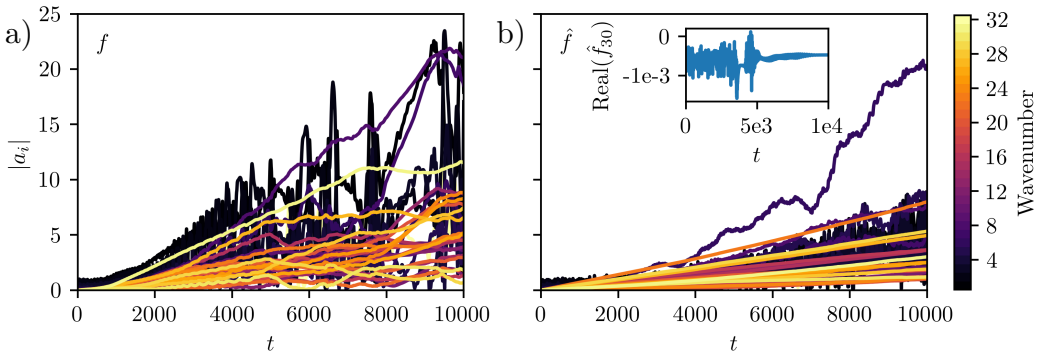

Figure 4: Magnitude of Fourier modes over time when evolving an initial condition with a neural ODE approximation of (a) the full-space dynamics in real space (f) and of (b) the full-space dynamics in Fourier space ($\hat{f}$). $L = 22$. $\tau = 0.25$. The inset in (b) is the evolution of the real part of $\hat{f}_{30}$.

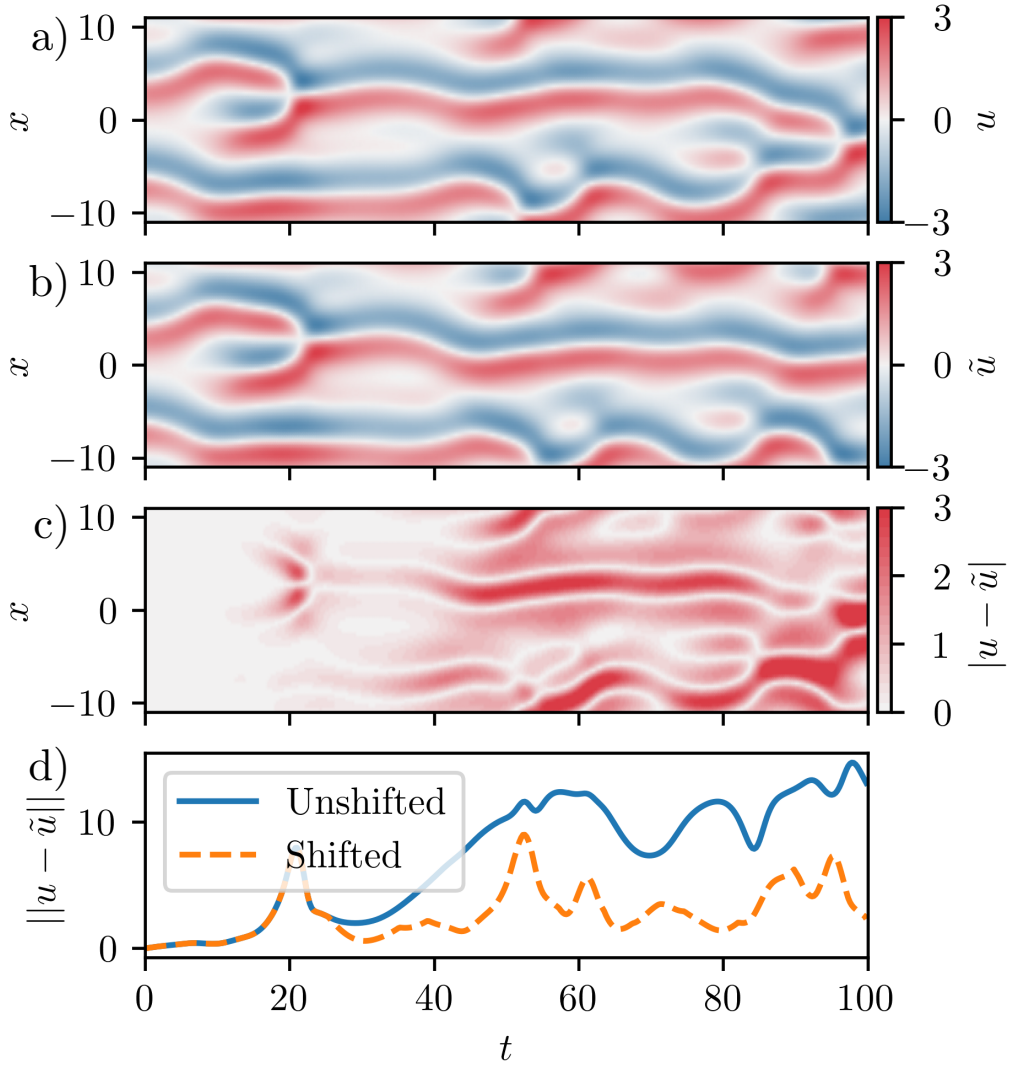

Figure 5: Example of trajectory and predictions for data spaced $\tau = 10$ apart, $L = 22$: (a) true trajectory; (b) predicted trajectory; (c) absolute error between the two trajectories; (d) norm of the error. The norm of the error is shown for the raw data, and for the spatial shift that minimizes this error at each time.

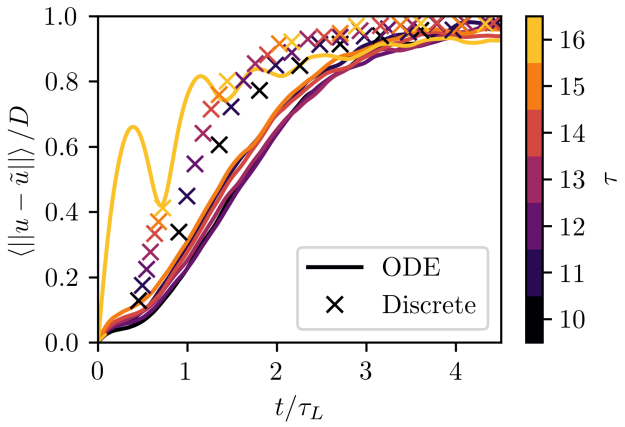

Figure 6: Normalized difference between trajectory and prediction as a function of time for different training data spacing, $L = 22, d_h = 8$.

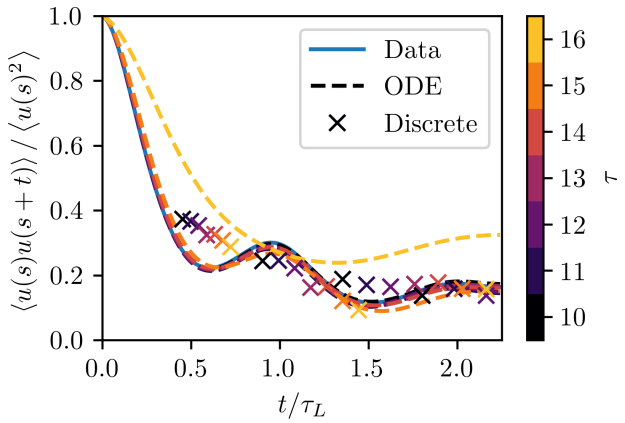

Figure 7: Temporal autocorrelation for models learned with different data spacing, $L = 22, d_h = 8$.

phase difference causes most of the error.

We now consider ensemble average quantities to better understand the reconstruction quality of the models as a function of τ. In addition to the ODE models, we also consider discrete-time maps $h(t_i+\tau) = G(h(t_i))$. Details of the discrete-time map NN architecture is given in Table 1.

Figure 6 shows the root mean squared difference between the exact trajectory and ODE/discrete-time map models as a function of time, averaged over 1000 initial conditions and normalized by the root mean squared difference between random training data, which is denoted with D. In this figure, the ODE models and discrete time maps use data spaced $\tau = 10 - 16$. For $\tau < 10$, there is little improvement in the ODE models' performance. ODE predictions at and below $\tau = 15$ all track well for short times, with the error being halfway to random at around $t \sim 1.5\tau_L$, and diverge at long times, with the error leveling off at around $t \sim 3\tau_L$. Then, performance degrades sharply at $\tau = 16$ (yellow curve). In all cases, the discrete-time map performs worse than the ODE with the same data spacing. Furthermore, although we do not show it here, the performance for discrete-time maps becomes even worse when trying to interpolate between prediction times. This is a significant drawback not seen when using ODEs.

A similar trend appears in the temporal autocorrelation. Figure 7 shows the temporal autocorrelation of u at a given gridpoint averaged over space and 1000 initial conditions. The temporal autocorrelation of the

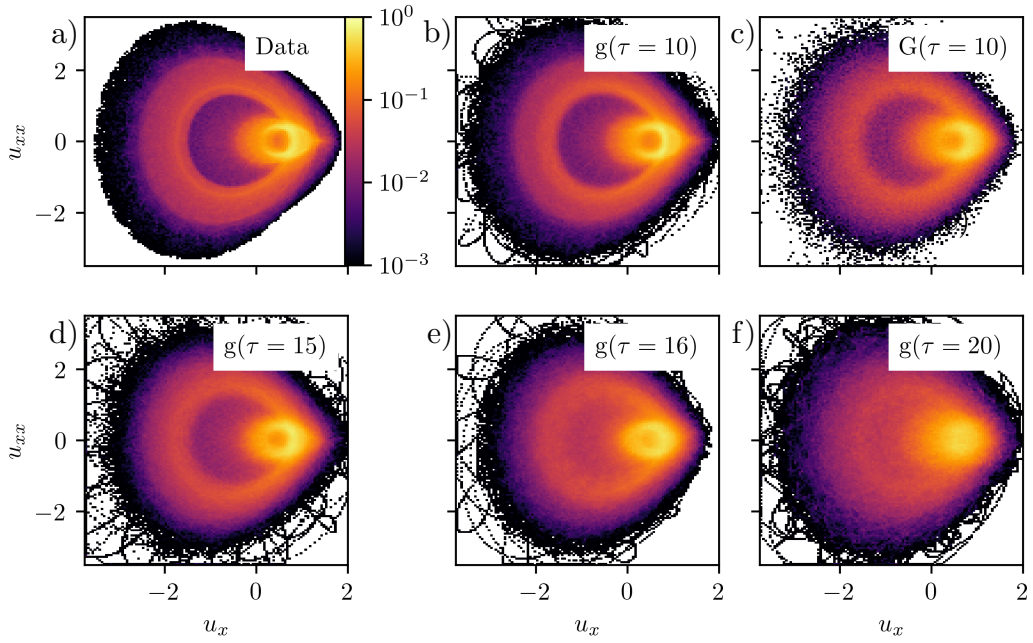

Figure 8: (a) joint PDF at $L = 22, d_h = 8$. (b) and (c) joint PDFs when approximating the ODE and the discrete-time map, respectively, at $\tau = 10$. (d)-(f) joint PDFs when approximating the ODE for $\tau = 15$, 16, and 20.

neural ODE matches the exact temporal autocorrelation well for data spaced up to $\tau = 15$. Then, there is an abrupt deviation from the true solution at $\tau = 16$. Also, in Fig. 7 we show the temporal autocorrelation for discrete-time maps. At $\tau = 10$ discrete-time maps perform worse than all ODEs below $\tau = 15$, and predictions worsen when increasing τ.

The other half of evaluating a model is determining if trajectories stay on the attractor at long times. For this purpose, we consider the long-time joint PDF of the first (u_x) and second (u_{xx}) spatial derivatives of the solution. In the KSE, $\overline{u_x^2}$ is the energy production and $\overline{u_{xx}^2}$ is the dissipation [21], where $\bar{\cdot}$ is the spatial average. We select these quantities because of their relevance to the energy balance, and because two-dimensional joint PDFs are more challenging and important to reconstruct than one-dimensional PDFs. In Fig. 8 the joint PDF for the data is compared to various models. The colormap is a log scale, with low probability excursions in black, and no data in white regions. When $\tau = 10$, the ODE model matches well, with differences primarily in the low probability excursions, while at $\tau = 10$ the discrete-time mapping appears more diffuse in the high probability region and matches poorly. At $\tau = 15$, the joint PDF for the ODE prediction still matches well, degrading once $\tau \geq 16$. This deterioration becomes more evident at $\tau = 20$.

We quantify the difference between true and predicted PDFs by computing the KL divergence

$$D_{KL}(\tilde{P}||P) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{P}(u_x, u_{xx}) \ln \frac{\tilde{P}(u_x, u_{xx})}{P(u_x, u_{xx})} du_x du_{xx} \tag{10}$$

between the true joint PDF, P, and the predicted PDF, $\tilde{P}$. We take the convention of setting the quantity under the integral to 0 when $P = 0$ or $\tilde{P} = 0$ as was done in [13]. The KL divergence appears in Fig. 9 for various τ, where we normalize by the Lyapunov time τ_L. For all domain sizes, the joint PDFs for neural ODE models match the true PDF very well when data is spaced below $\sim$0.7 Lyapunov times. In contrast, good reconstruction of the joint PDF with the discrete-time model, for $L = 22$, requires $\tau/\tau_L \lesssim 0.4$.

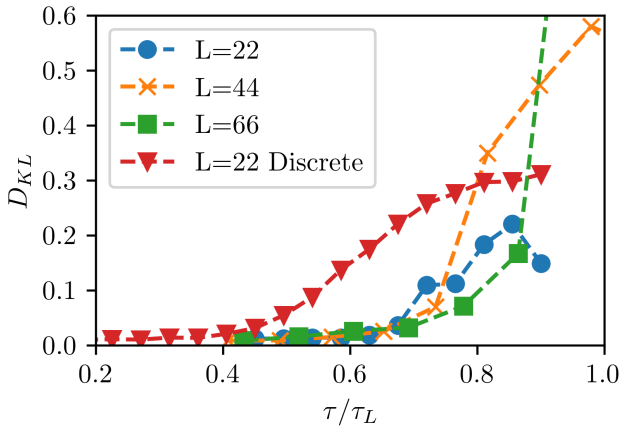

Figure 9: KL divergence of the models trained with different data spacing. Here time is scaled with Lyapunov time for the corresponding domain size. For $L = 22, 44, 66$ we use $d_h = 8, 18, 28$.

Table 2: Architectures of NNs used in Section 2.1.3. Labels are the same as in Table 1.

	Function	Shape	Activation
Encoder $L = 22$	χ	$d : 500 : d_h$	S:linear
Encoder $L = 44, 66$	χ	$d : 500 : 500 : d_h$	S:S:linear
Decoder $L = 22$	$\check{\chi}$	$d_h : 500 : d$	S:linear
Decoder $L = 44, 66$	$\check{\chi}$	$d_h : 500 : 500 : d$	S:S:linear
ODE	g	$d_h : 200 : 200 : 200 : d_h$	S:S:S:linear

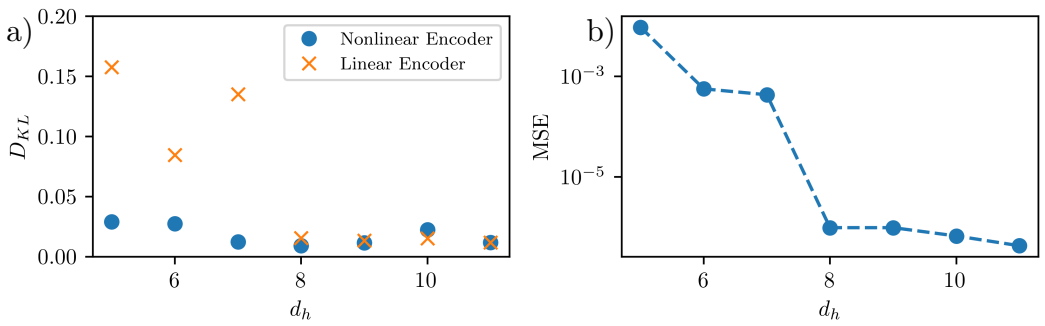

Figure 10: (a) KL divergence of the best autoencoder and neural ODE pair at multiple dimensions for $L = 22$. The nonlinear encoder corresponds to an autoencoder with a NN for encoding, and the linear encoder corresponds to an autoencoder with the projection onto the first d_h Fourier modes for encoding. Both cases use a NN for decoding. (b) mean squared error of reconstruction for the nonlinear autoencoder at each dimension.

Effect of Model Dimension on Time Evolution Predictions In the previous sections, the model dimension was fixed at values assumed to be the correct inertial manifold dimensions based on other studies [69, 119, 23]. Here we vary the dimension and consider the dynamical model performance, which provides more information on the necessary number of dimension. For training we again use 10^5 time units of data spaced apart 0.25 time units and train 5 autoencoders and 15 neural ODEs. The data is normalized by subtracting the mean and dividing by the standard deviation before training and projected onto the full set of PCA modes before going into the autoencoder ($U^T u$). Training was done with an Adam optimizer with learning rate scheduling from 10^{-3} to 10^{-4} halfway through training. For $L = 22$ we trained for 5000 epochs and for $L = 44, 66$ we trained for 500 epochs, due to the computational cost of the larger networks. In all these trials, the loss plateaued at the end of training, but further training in the $L = 44, 66$ cases could lead to mild improvement. The code used for training these autoencoders is available on GitHub [71].

Our first result here is the observation that, for models with a lower dimension than the "correct" one, reasonable approximations of the joint PDF of u_x and u_{xx} can be obtained, but only if a nonlinear encoder is used for dimension reduction. This point is illustrated in Figure 10a, for the case $L = 22$, where we see good reconstruction of the joint PDF when using a nonlinear encoder. Accordingly, the results below all use a nonlinear encoder; architectures are reported in Table 2. However, even though this statistic is reconstructed well at low dimensions, Fig. 10b shows the MSE for the autoencoder, with a nonlinear encoder, still drops significantly at $d_h = 8$. This MSE is on test data normalized using the training data mean and standard deviation. As with the autoencoders used in Sections 2.1.3 and 2.1.3, we again performed manual hyperparameter tuning. Changing the activation functions from sigmoids to rectified linear units and removing the learning rate scheduler both resulted in higher MSE for $d_h \gtrsim d_{\mathcal{M}}$.

Figure 11 shows the ensemble-averaged short-time tracking error of the models with the best tracking as the dimension varies. For all domain sizes the recreation gradually improves and then becomes dimension-independent as dimension increases. This happens because the short-time tracking is directly related to the loss minimized in training the neural ODEs, and h contains the same information as u if d_h is large enough. As mentioned before, for $L = 22$, 44, and 66, respectively, the "correct" manifold dimensions are 8, 18, and 28, and Fig. 11 shows the tracking error becoming dimension-independent near these values.

Now we turn to long-time statistical recreation upon varying dimension. Figure 12 shows the KL divergence from the true joint PDF $P(u_x, u_{xx})$ for all the autoencoder and neural ODE pairs trained for each dimension and domain size. For reference, the dashed lines show D_{KL} when comparing the true joint PDF, used to compute D_{KL} for the models, to a separate joint PDF from the true system, and evolved forward for the same amount of time as the models but from a different initial condition. This gives a baseline for how

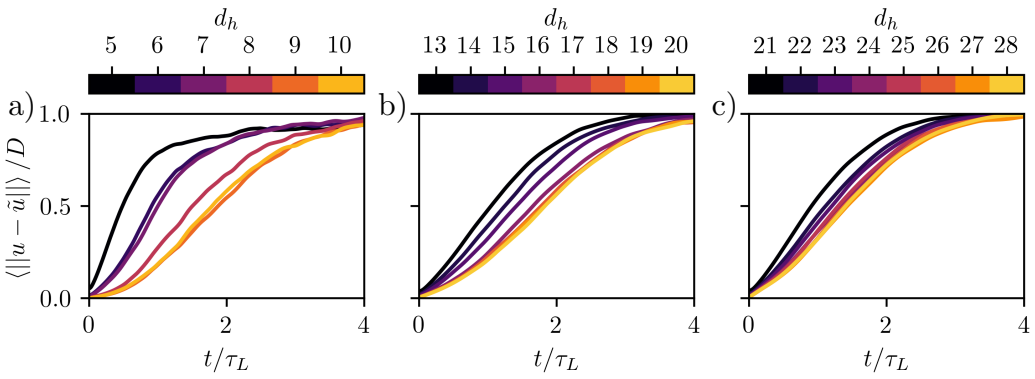

Figure 11: Short-time error at various dimensions. Domain sizes are $L = 22, 44, 66$ for (a)-(c).

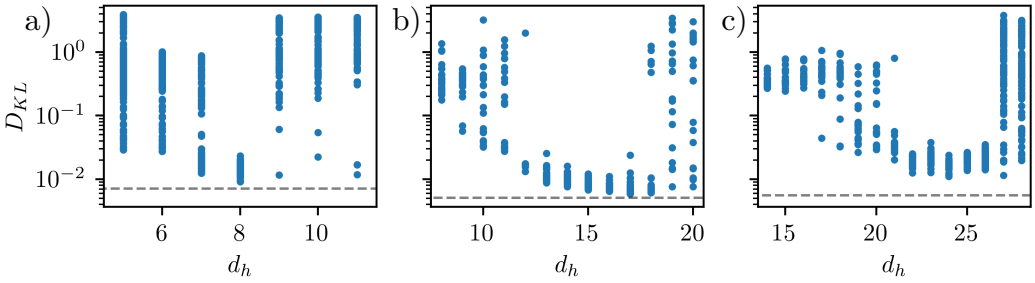

Figure 12: KL divergence for all trained models at different dimensions. Domain sizes are $L = 22, 44, 66$ for (a)-(c). The dotted lines on each plot correspond to the D_{KL} computed by comparing two joint PDFs of the true system.

two empirical joint PDFs from the true system compare. In all the cases, when the dimension is low, the models do a poor job of reconstructing the joint PDF. When we are near the expected manifold dimension i.e. $d_h \approx 8$, 18, and 28, for $L = 22, 44$, and 66, respectively, the typical model performance dramatically improves and the variation in performance between the models decreases, with $D_{KL} \approx 10^{-2}$ for the best-performing models and only a factor of two or three larger for the worst. Furthermore, D_{KL} between true and best-model predictions is only slightly larger than that between the two time intervals for the true system, with particularly close agreement for $L = 22$ and 44. In Fig. 13, we plot the true PDFs and the best model PDFs at $L = 22, 44, 66$ and $d_h = 8, 18, 28$, illustrating the excellent agreement. However, with further increase of dimension, the variation in performance again becomes large, although the best models still perform well. Like the short-time prediction, the joint PDFs of the best models become dimension-independent, but here we see that errors arise, particularly the slow linear growth in high wavenumbers, when training the neural ODE with unnecessary dimensions. Finally, recall that the error at high d_h arises from spurious slow linear growth at high wavenumbers, which, as we saw in Section 2.1.3, arises in the absence of any autoencoder at all.

In summary, Figure 12 highlights that there is a "sweet spot", $d_h \approx d_{\mathcal{M}}$, where we see robust model performance. Below this, too few dimensions are retained so model performance is poor, and above it, spurious slow growth of high wavenumbers pollutes the long-time predictions. So lack of robustness in model performance is an indication that the dimension has been chosen incorrectly – either too low or too high. As noted previously, we also find that stabilizing long-time dynamics is possible with the addition of a damping term.

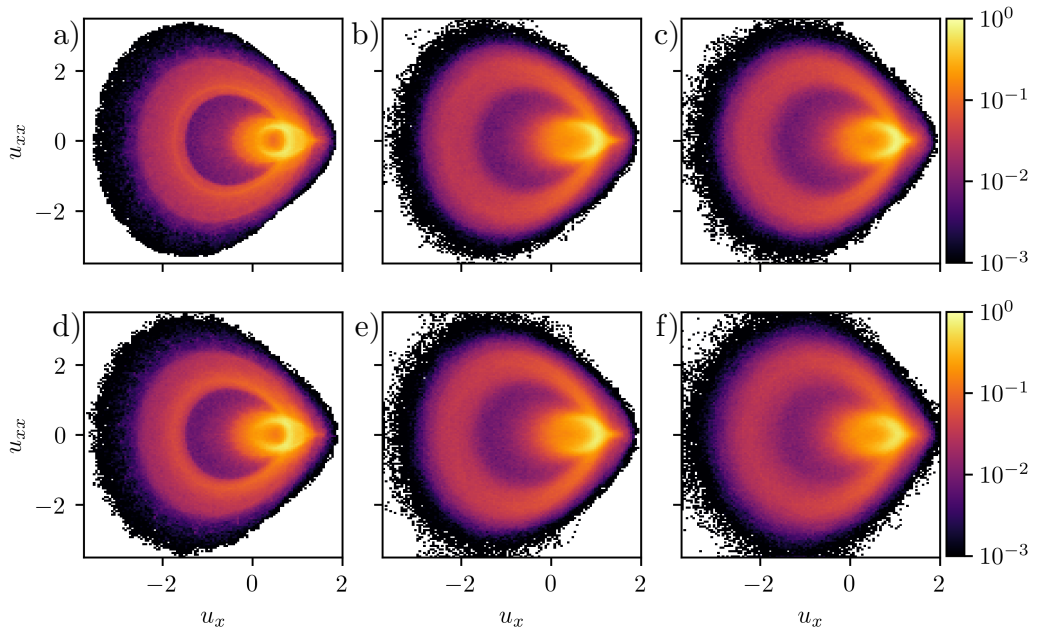

Figure 13: (a)-(c) True joint PDFs for $L = 22, 44, 66$. (d)-(f) joint PDFs from the best models (lowest D_{KL}) for $L = 22, 44, 66$ at $d_h = 8, 18, 28$.

2.1.4 Conclusion

Neural ODEs and undercomplete autoencoders can be used to create accurate data-driven models for time evolution of spatiotemporal chaotic dynamical systems, with dimensions near the expected manifold dimension, which for the present examples is in the approximate range $8 - 28$. Dimension reduction is a vital step in the process. Both the training process and the prediction of trajectories are more expensive with more dimensions, and additional dimensions lead to computational difficulties with the introduction of small spurious bias that leads to slow linear growth of high-wavenumber modes. With dimension reduction, we find that training with data spaced up to ~ 0.7 Lyapunov times gives accurate short-time tracking and long-time statistical reconstruction. Finally, we investigate the impact of varying the dimension and find that model performance improves with dimension, and stops improving once the manifold dimension is reached. However, because of the spurious excitation of high wavenumbers, keeping too many dimensions hurts training, resulting in many poor models. While we have used the Kuramoto-Sivashinsky equation as a model system, this computational approach using data mediated neural ODEs might also be useful in the modeling of PDE systems like atmospheric phenomena [72], in the modeling of infinite-dimensional discrete-time maps [83], or in the modeling of other complex spatially distributed systems [59]. In forthcoming work we will show how this reduced order modeling approach can be integrated into a reinforcement learning framework for control of spatiotemporally chaotic dynamics.

2.2 Integration of data-driven modeling with reinforcement learning for control of chaotic dynamics [122, 123]

2.2.1 Introduction

Design of active control strategies for turbulent drag reduction is a challenging task due to the complex dynamics and the difficulty in devising good control targets. Deep reinforcement learning (RL), an emergent machine learning method capable of learning complex control strategies for high-dimensional systems from

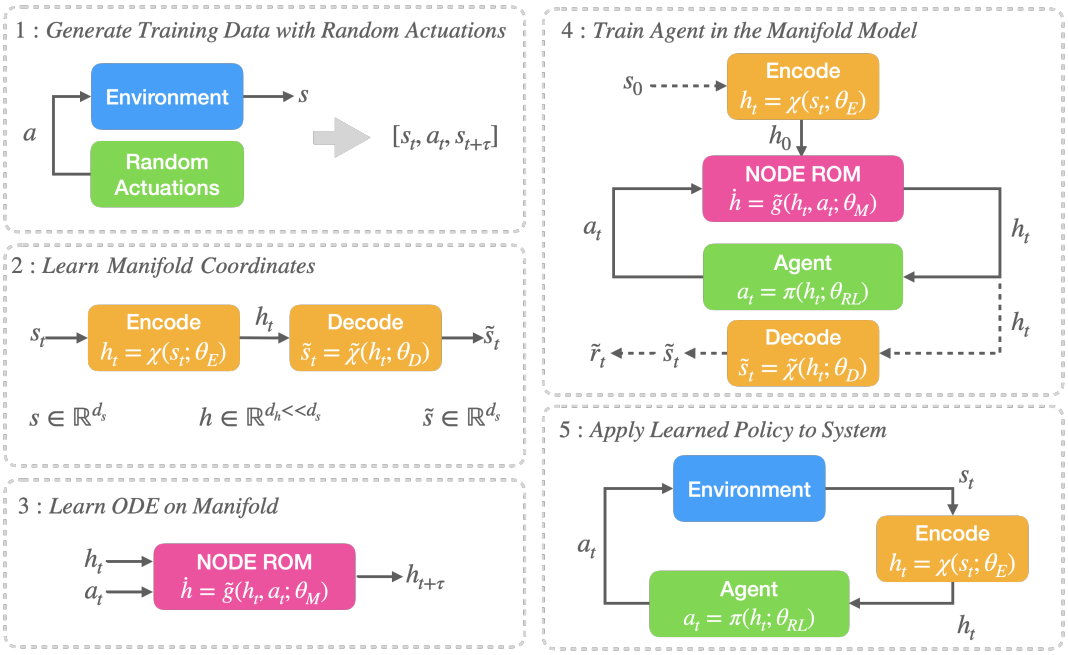

Figure 14: Procedure for learning a DManD model from data, combining it with RL to approximate a control policy, and deploying the approximate policy back to the true system

data, is able to address broad macroscopic control objectives, such as drag minimization, making it promising for flow control application. However, the iterative RL process requires vast amounts of data to be generated from interactions with the target system. For high-dimensional and computationally demanding simulations, such as direct numerical simulations (DNS), this data generation step can be prohibitively expensive.

We mitigate this challenge in a completely data-driven fashion by combining data-driven reduced-order models (ROM) of the flow system with deep RL using the DManD framework described above. We construct our ROMs by combining an undercomplete autoencoder with a neural ordinary differential equation (ODE)–both of which are trained directly from data, as detailed above in Section 2.1. The autoencoder compresses the high-dimensional state representation into a low-dimensional representation while the neural ODE predicts the system dynamics as an ODE in this new representation. This ROM is substituted in place of the true system during RL training to accelerate the learning process. The ROM-based control strategy is then deployed to the flow system for control validation. We denote the entire framework as "DManD-RL".

We first apply our method to the Kuramoto-Sivashinsky equation (KSE), a 1D turbulence proxy that exhibits spatiotemporal chaos, equipped with actuators. We demonstrate that we are capable of learning a ROM of the actuated dynamics and with a control goal analogous to drag reduction, we show that the ROM-based RL strategies perform well in the KSE. We highlight that the RL agent discovers and stabilizes a forced equilibrium solution. Next, we apply our method to a DNS of Couette flow with control in the form of jets that modify the wall-normal velocity at the wall. We find a ROM that captures key short-time behavior of the underlying system using drastically fewer dimensions, and we show RL forces the system to relaminarize far more frequently than the underlying system.

2.2.2 Data-Driven Reduced Order Modeling with DManD

The first step in our procedure is to learn a ROM surrogate of the system we aim to control. We assume we have access to time series data $[s(t_0), s(t_1), ..., s(t_M)]$ of the system and that the dynamics of the system

lie (or approximately lie) on some low-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^d$ (i.e. $s \in \mathcal{M} \subset \mathbb{R}^d$); an assumption that has been affirmed for many dissipative systems [106]. We can find a parameterization $h \in \mathbb{R}^{d_h}$ that is of a much smaller dimension than the ambient space $d_h \ll d$ (when the dimension is minimal we write $d_\mathcal{M}$ in place of d_h). We then formulate the time-evolution of h with an ODE. Once constructed, the ROM forecasts trajectories of h (which we can map to the ambient space s) and uses these trajectories to train the RL agent. Figure 14 (a)-(c) outlines the learning procedure for the ROM. As noted in Section 2.1, we denote this approach as "Data-driven Manifold Dynamics", or DManD.

To find the ROM we first use an undercomplete autoencoder to learn the parameterization of the manifold h. An autoencoder consists of an encoding function $h = \chi(s;\theta_E)$ and a decoding function $s = \check{\chi}(h;\theta_D)$. Both of these functions are constructed by neural networks (NN) with weights $\theta = [\theta_E, \theta_D]$ that are trained simultaneously to minimize the reconstruction loss $L = 1/M \sum_i^M ||s(t_i) - \check{\chi}(\chi(s(t_i)))||^2$. We compute the gradient of this loss with respect to the NN parameters ($dL/d\theta$) and use stochastic gradient descent to update these parameters.

A challenge with this approach is determining the necessary degrees of freedom to fully parameterize the manifold on which the the data lies. Whitney's Embedding Theorem proves as long as the encoding maps to $\mathbb{R}^{2d_\mathcal{M}}$ then h is homeomorphic to s [115]. However, $d_\mathcal{M}$ is unknown a priori, so we must find a way to determine it. We estimate this dimension by tracking the error of the autoencoder as a function of dimension, where we have empirically observed a dramatic improvement in performance once d_h agrees with $d_\mathcal{M}$ [69, 70].

After finding a reduced-dimensional representation h, we next find a dynamical system for h. We chose to learn an ODE for the dynamics of h such that $\dot{h} = g(h, a; \theta_{\text{ODE}})$, where g is a NN and a is the action taken by the controller. The action must be input here because the RL agent iteratively interacts with the system to determine the best control policy. Using this ODE we predict new states by integrating forward in time

$$\tilde{h}(t_i + \tau) = h(t_i) + \int_{t_i}^{t_i+\tau} g(h(t), a; \theta_{\text{ODE}}) dt. \tag{11}$$

We train g by minimizing the difference between the prediction ($\tilde{h}(t_i + \tau)$) and the known state ($h(t_i + \tau)$), in the manifold coordinates, giving the loss $J = \sum_i^M ||h(t_i + \tau) - \tilde{h}(t_i + \tau)||^2$ [15].

2.2.3 Learning a Control Strategy From Data

We use deep RL to learn control strategies from data. The general deep RL framework is a cyclic interactive learning process. During each cycle, the RL agent, the embodiment of the control policy represented by neural-network $a_t = \pi(s_t; \theta_{RL})$, outputs a control action, a_t, given a state observation of the system, s_t. The impact of this action on the system is then quantified by a scalar reward, r_t, which is defined by the control objective. During training, the agent attempts to learn the mapping between s_t and a_t that maximizes the cumulative long time reward and updates accordingly each cycle. In our ROM-based RL framework, we train the agent in the reduced space of the ROM, such that we instead learn $a_t = \pi(h_t; \theta_{RL})$. During agent deployment to the original system for control validation, we simply precede the agent with the encoder $h_t = \chi(s_t; \theta_E)$ to map state observations, s_t, to h_t as this is coordinate system the agent was trained with. Fig. 14 (d)-(e) outlines the RL training and deployment procedure. We comment that our framework can applied with any general deep RL method.

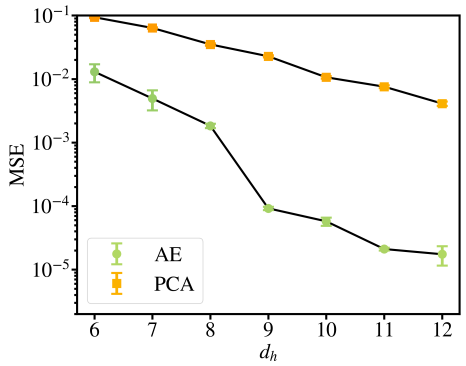

Figure 15: Mean squared error of autoencoders and PCA on test data of the KSE for various dimensions.

2.2.4 Example 1: Kuramoto-Sivashinsky Equation

We consider the periodic KSE with external forcing, given by

$$\frac{\partial v}{\partial t} = -v\frac{\partial v}{\partial x} - \frac{\partial^2 v}{\partial x^2} - \frac{\partial^4 v}{\partial x^4} + f(x,t) \tag{12}$$

where f is the control term defined as in [9] and corresponds to 4 evenly spaced Gaussian jets,

$$f(x,t) = \sum_{i=1}^{4} \frac{a_i(t)}{\sqrt{2\pi\sigma_s}} \exp\left(-\frac{(x-X_i)^2}{2\sigma_s^2}\right). \tag{13}$$

We evolve trajectories for a domain size of $L = 22$, which exhibits spatiotemporal chaos with a Lyapunov time of $\sim$22 time units. Spatial discretization is performed with Fourier collocation on a mesh of 64 evenly spaced points and in our formulation the state vector u consists of the solution values at the collocation points. The system is time evolved with a third-order semi-implicit Runge-Kutta scheme [9]. From this simulation we train autoencoders with size 500 sigmoid activated hidden layers (for encoder, decoder) and neural ODEs on 40,000 snapshots of data ($s_t = u(t)$) separated 0.25 time units apart experiencing random jet actuations. Figure 15 shows the performance of autoencoders as we vary the dimension d_h. For this system, the error is sufficiently low at $d_h = 12$, so we use this autoencoder for the mappings χ and $\check{\chi}$. Furthermore, we emphasize that we achieve orders of magnitude improvement in reconstruction performance compared to PCA while using the same number of dimensions. In this representation of the manifold coordinates, we train a neural ODE using the same dataset. In Figure 16 we show the short-time predictions of this model with random actuations match the true system well for around 1-1.5 Lyapunov times. We also comment that in the absence of actuation, i.e. $a_t = 0$, the ROM performs similarly well, indicating that the ROM is able to capture the underlying natural dynamics absent of control inputs despite having been trained with no control-free data.

We now use this ROM as a surrogate model for training the RL agent. In this demonstration we utilize the Deep Deterministic Policy Gradient (DDPG) RL method [67]. As an analogue to energy-saving in the flow control problem, we seek to minimize the dissipation and total power input of the KSE. This optimization is realized by maximizing the reward $r = -D + P_f$, where $D = \langle(\frac{\partial^2 u}{\partial x^2})^2\rangle$ and $P_f = \langle(\frac{\partial u}{\partial x})^2\rangle + \langle uf\rangle$, respectively. Here $\langle\cdot\rangle$ is the spatial average.

The control agent was trained with 1000 episodes of 100 time units long (i.e. 400 transitions per episode),

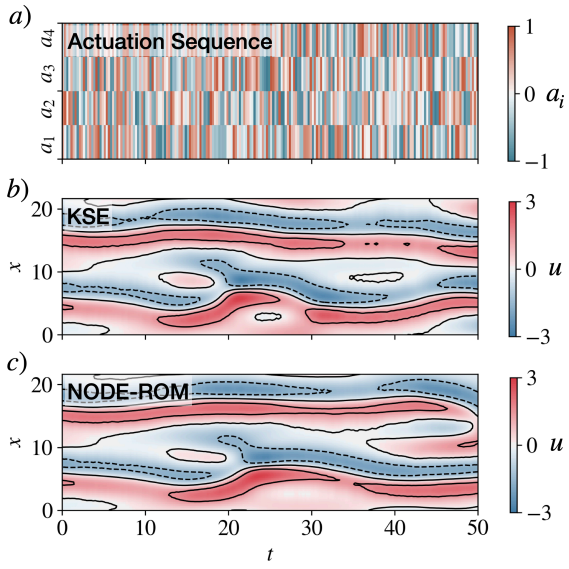

Figure 16: (a) random actuation sequences $a_i(t)$ (b) ground truth KSE trajectory starting from a random initial conditions following actuation sequences in (a), (c) the decoded DManD trajectory following actuation sequence (a) and the same initial condition in (b).

with each episode beginning from a random on-attractor initial condition of the natural, i.e. unforced, KSE. Jet actuations implemented by the control agent, a_t, were maintained constantly from s_t to $s_{t+\tau}$, where $\tau = 0.25$. In this work the DDPG actor and critic networks utilized ReLU activated hidden layers of size 128 and 64, respectively, followed by tanh and linear activations to the outputs of size 4 and 1, respectively.

To assess the performance of our ROM-based policy, the learned control agent was applied to the DManD model, with an example controlled trajectory shown in Fig. 17a. We note that after a brief control transient, the control agent navigates the model dynamics to an equilibrium (steady) state and stabilizes it. The quantities targeted for minimization, D and P_f, estimated from the predicted trajectory $u(t)$, are shown in Figure 17c, revealing that this equilibrium exhibits dissipation much lower than the natural unactuated dynamics. To assess how well this ROM-based control policy transfers to the original KSE (i.e. the true system), the same policy is applied to the true KSE with the same initial condition, as shown in Fig. 17b. We note that the controlled trajectory in the KSE yields not only quantitatively similar transient behavior but also the same low-dissipation equilibrium state as was targeted in the DManD model. The transient behaviors between the two are structurally very similar, although the DManD model displays slightly less strongly damped oscillations as it drives the trajectory to the steady state. The values of D and P_f computed from the true system, shown in Fig. 17d, are nearly identical to that of the DManD model in Fig. 17c.

To demonstrate the robustness of the ROM-based policy, shown in Fig. 17e are the dissipation trajectories of the true KSE beginning from 15 randomly sampled test initial conditions that the ROM-based control agent has not seen before. We highlight that the control agent is able to consistently navigate the system to the same low-dissipation state within $\sim$ 150 time units, with one initial condition requiring $\sim$ 200 time units to converge. Finally we note that although the RL training horizons were only 100 time units long, the control agent is able to generalize to achieve and maintain control well beyond the the horizon it was trained in.

Here we emphasize that the ROM-based policy drives the dynamics to an equilibrium state in both the DManD model and the true KSE, indicating that not only does the DManD model capture this state, but it captures the dynamics leading to it accurately enough such that the RL agent could discover it during training and exploit it in a manner that still translates to the original system. We further emphasize that

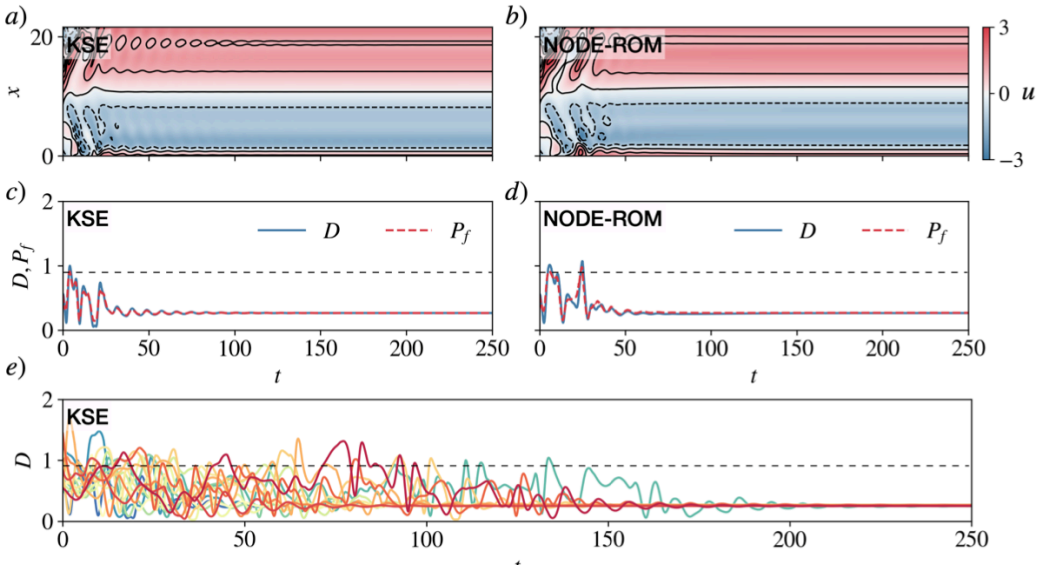

Figure 17: ROM-based RL agent applied to the same initial condition in the a) true KSE and b) data-driven reduced-order model (decoded, $d_h = 12$). The corresponding invariant quantities of dissipation and total input power for the c) true KSE and d) learned reduced-order model. The dashed black line represents the system average of the natural KSE dynamics. e) Controlled dissipation trajectories of the true KSE beginning from 15 randomly sampled test initial conditions.

both the DManD model and agent were never explicitly informed of this low-dissipation state's existence. Finally, we highlight that the discovered low-dissipation state is an unstable state that is stabilized by the control agent. If control is removed, the system returns to the natural chaotic dynamics. The ROM-based RL agent is comparable to that of an RL agent trained directly with original system via direct interactions in both performance and targeting strategy [122].

These observations indicate that the RL policy trained on the model transfers very well to the true system. We attribute this performance to the fact that both the DManD model and RL agent operate in Markovian fashion, i.e. even if the model has slight inaccuracies, so long as the modeled dynamics are reasonably accurate this does not matter once the agent makes its new state observation. Returning to the dynamical significance of the low-dissipation equilibrium state discovered and stabilized by the RL agent, a continuation

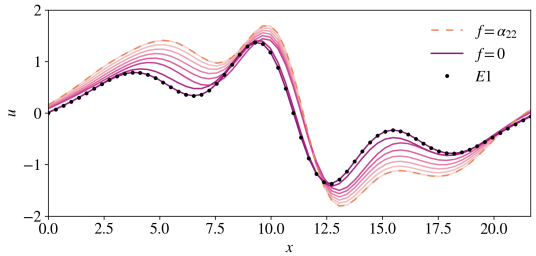

Figure 18: Forcing continuation from the forced equilibrium state (under forcing $f = \alpha_{22}$) discovered by DManD-RL policy (orange, dashed) to the unactuated KSE system (purple). The known equilibrium E1 of the KSE system is also provided (dots).

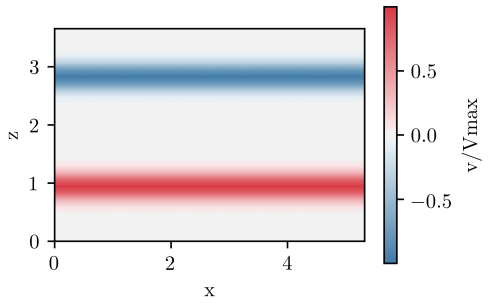

Figure 19: Visualization of the localized wall-normal velocity of the slot-style jets located at $y = -1$.

in mean forcing magnitude was performed. To do so, we Newton-solved for equilibrium solutions to the KSE starting with the discovered equilibrium state while gradually decreasing the magnitude of the mean actuation profile to zero, as was done in [122]. Solutions identified by this continuation in forcing magnitude are shown in Fig. 18, which reveals that equilibrium state captured by the DManD model and discovered by the RL agent is connected to a known existing solution of the KSE known as $E1$ [21]; we obtained a similar result with an RL agent trained on interactions with the full system [122]. A similar observation was made for RL control of 2D bluff body flow [65]. We speculate that in systems with complex nonlinear dynamics, the discovery and stabilization of desirable underlying equilibrium solutions (or other recurrent saddle-point solutions such as unstable periodic orbits) of the system may be a fairly general feature of RL flow control approaches. The nonlinear and exploratory nature of RL algorithms facilitates the discovery of such solutions, and since the dynamics are slow near these solutions, little control action should be required to keep trajectories near them.

2.2.5 Example 2: Turbulent Couette Flow

The second system we consider is Plane Couette flow in a channel with two periodic boundary conditions defined by the nondimensionalized incompressible Navier-Stokes Equations:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla P + \mathrm{Re}^{-1} \nabla^2 \mathbf{u} \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{14}$$

where $\mathbf{u} = [u, v, w]$ is the vector of streamwise, wall-normal, and spanwise velocities, and P is the pressure. The boundary conditions at the wall are $u(\pm 1) = \pm 1$, $\partial v/\partial y(\pm 1) = 0$, $w(\pm 1) = 0$, $v(1) = 0$, and $v(-1) = a_i(t) f(x,z)$. This final boundary condition is the actuation, located only at the bottom wall, which is in the form of two spatially localized slot-style jets that inject or suck fluid in the wall-normal direction with zero net-flux. These slot-jets are oriented along the streamwise direction with Gaussian profiles $f(x,z)$ in the spanwise direction with $V_{\max} = \max(|f(x,z)|) = 1/20$ and $a_i(t) \in [-1, 1]$. In Fig. 19 we show the wall-normal velocity in the case of $a_0 = 1$ and $a_1 = -1$.

Our code was written in Python to speed up communication with the RL code and follows the same algorithms used by [38]. In particular, we convert the field to Fourier space in x and z and Chebyshev space in y with a resolution of $32x35x32$ Fourier-Chebyshev-Fourier modes. Then we use the multistep Adams-Bashforth/Backward-Differentiation 3 method described in [90] for time evolution using a timestep of $\Delta t = 0.02$. This results in a set of implicit Helmholtz equations that we solve at each timestep, and through an influence matrix we compute the pressure required to satisfy $\partial v/\partial y(\pm 1) = 0$. This procedure does not

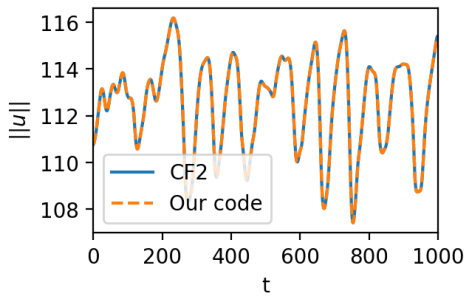

Figure 20: The $L2$ norm of the velocity field, $\mathbf{u}$, of a trajectory produced by Channel Flow 2.0 by Gibson and our python-based Couette DNS code starting from the same initial condition.

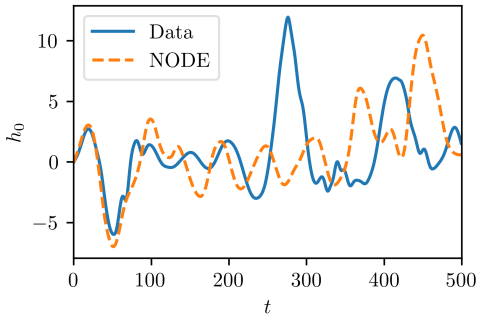

Figure 21: Trajectory of the projection of the Couette flow field onto the first POD mode, and the prediction by the neural ODE.

result in an incompressible velocity field, so we apply a Tau correction to exactly enforce this constraint. The influence matrix and Tau correction methods are described in [60]. Unlike the Gibson et al. code, our code allows us to modify the boundary conditions at the wall by solving for intermediate values in the influence matrix and tau correction methods at each timestep, as opposed to once at the beginning. As illustrated in figure 20, we validated our code by showing that solutions of both codes track nearly exactly for 1000 time units.

We consider Couette flow at a Re $= 400$ and domain size of $[L_x, L_y, L_z] = [7\pi/4, 2, 6\pi/5]$, which are the parameters considered by [38]. For this system we generated 80,000 snapshots of data ($s_t = u(t)$) separated by 1 time unit. We performed proper orthogonal decomposition (POD) for dimension reduction. In future work an autoencoder will be used. After reducing the dimension with POD, we train neural ODEs with this lower-order representation. Figure 21 shows the trajectory of the 1st POD coefficient for the true system and the ROM where h is constructed from the first 10 POD modes. This preliminary result indicates we can achieve qualitatively similar trajectories with very few degrees of freedom using neural ODEs.

With the success of training RL agents with ROMs of the KSE, and having promising preliminary results with low-dimensional ROMs for Couette flow, our next objective is to train a RL agents on Couette flow ROMs utilizing the Soft Actor-Critic (SAC) RL algorithm [45]. However, before this, we first learn a control policy directly from the DNS (i.e. direct interactions) to determine a benchmark control performance and timescale required for training. We train our RL agent for several hundred episodes, with each episode lasting for 300 time units and beginning from a turbulent initial condition that does not naturally laminarize in 300

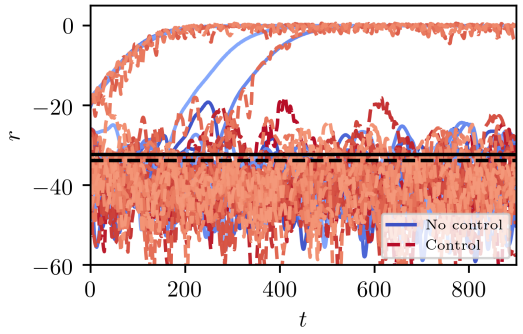

Figure 22: Trajectories beginning from 25 unseen initial conditions evolving under no control (blues) and random slot actuations (reds). The mean drag achieved across the 25 trajectories is shown in black for no control (solid) and random actuation (dashed)

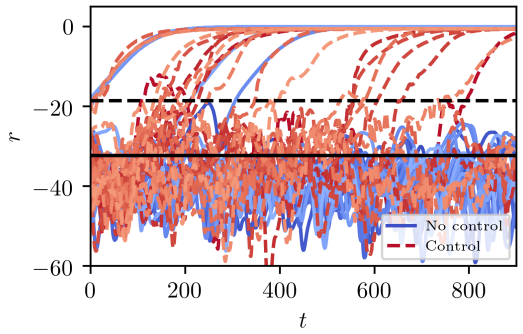

Figure 23: Trajectories beginning from 25 unseen initial conditions evolving under no control (blues) and the RL control agent (reds). The mean drag achieved across the 25 trajectories is shown in black for no control (solid) and the RL agent (dashed).

time units. Actions input by the control agent persist for 5 time units, i.e. the agent makes 60 actions per training episode. The reward the agent attempts to maximize is the negative of the drag (normalized such that laminar is 0), which is given by $r = -\int_0^{L_x}\int_0^{L_z} \partial u/\partial y(-1) + \partial u/\partial y(1) - 2dxdz$. Shown in Fig. 22 is the reward vs. time of trajectories generated by our DNS evolving from 25 new unseen initial conditions under no control and with random slot actuations, i.e. the slot actuations were randomly sampled from a uniform distribution of the allowable control range every 5 time units. We observe that in the presence of random slot actuations, the mean drag across the 25 trajectories over a horizon of 900 time units is worse than in the presence of no control. Furthermore, some instances where the system would naturally laminarize are disrupted by the random slot-jets. Shown in Fig. 23 is again the total drag vs. time of Couette flow generated by our DNS evolving from the same 25 new unseen initial conditions under no control and in the presence of our trained RL agent. We observe that with the learned RL control policy, the mean drag across the 25 trajectories over a horizon of 900 time units is drastically lowered compared to no control. We highlight that the RL agent learns to direct the chaotic turbulent dynamics to the laminar state within this window of 900 time units, which can be seen by the 19 of 25 trajectories laminarizing due to the control agent. This is a stark improvement compared to the 5 of 25 naturally laminarizing trajectories in the presence of no-control

and 4 of 25 in the presence of random slot actuations.

The current accomplishment is the first, to the extent of our knowledge, application of deep RL to learn an active control strategy from a direct numerical simulation of turbulent Couette flow. To train such an agent required 3+ weeks of simulation time on 4 processors. We highlight that our preliminary DManD model is capable of simulating 500 time units of Couette flow in less than the time it takes to simulate 1 time unit by DNS. In forthcoming work we aim to demonstrate the application of training the RL agent via DManD-RL in place of the computationally intensive DNS for efficient determination of control strategies.

2.3 Discovering multiscale and self-similar structure with data-driven wavelets [33]

2.3.1 Introduction

Many important processes are multiscale in nature, meaning that they exhibit structure at multiple scales of time and/or space. In nature, a prominent example is the dynamics of oceans and associated interactions with the atmosphere, which govern the planet's weather and climate systems [62]; much effort is expended in capturing and understanding effects at multiple scales of time and space [64]. In engineering, a prominent example is networks, specifically social media networks. Networks have multiscale structure by virtue of hierarchies of communities of nodes in the networks [1]. Understanding the structure of hierarchical communities in social media networks is crucial to understanding the spread of disinformation (and censorship of information) in these networks [7]. Broadly speaking, identifying and understanding the features present in multiscale processes is crucial to understanding and controlling these processes. Although the application we focus on here will be turbulent fluid flows, the ensuing discussion applies to any multiscale process for which the notions of energy (variance in the statistical context) and localization (a form of sparsity) are relevant.

Turbulence is a canonical multiscale process consisting of localized concentrations of vortex motion that are coherent in space and time and coexist at a wide range of scales. Theoretical arguments indicate that at intermediate scales and far from walls, the structure of a turbulent flow should be self-similar [75, 99]. This notion is qualitatively illustrated in Figure 24, which illustrates a snapshot from a simulation of homogeneous isotropic turbulence (HIT) at several scales [2][89, 66, 120]. As with other multiscale processes, a great challenge in fluid dynamics is to rationally identify and analyze coherent structures from a complex turbulent flow field. While it is often mathematically convenient to analyze signals in the Fourier domain, trigonometric functions are not localized in space, and what one observes at an instant in time in a turbulent flow rarely if ever looks sinusoidal. Alternately, conventional wavelet bases, which are localized and self-similar, can be used for analysis [28]. In both the Fourier and wavelet approaches, the bases for representing the flow are imposed a priori rather than emerging from data.

One of the primary methods of extracting structure from data is principal components analysis (PCA), which in fluid dynamics is typically denoted Proper Orthogonal Decomposition (POD) [52]. (See [104] for other popular modal decomposition methods.) Given an ensemble (often a time series) of data, PCA yields a data-driven orthogonal basis whose elements are optimally ordered by energy content. When applied to velocity field data for a fluid flow, the resulting basis elements may be thought of as the building blocks of that flow, and its application has yielded many structural and dynamical insights [52, 47]. One limitation of PCA is that the basis elements tend not to be localized in space; indeed, for directions in which a field is statistically homogeneous, the PCA basis elements are Fourier modes [52]. In this case, not only do the PCA modes have no localization in space, they also reveal no information about the flow beyond what Fourier decomposition would provide.

A well known formalism that produces bases with spatially localized elements is that of wavelets. The name is quite descriptive: wavelets are localized waves. In particular, wavelet decompositions provide an

[2] http://turbulence.pha.jhu.edu

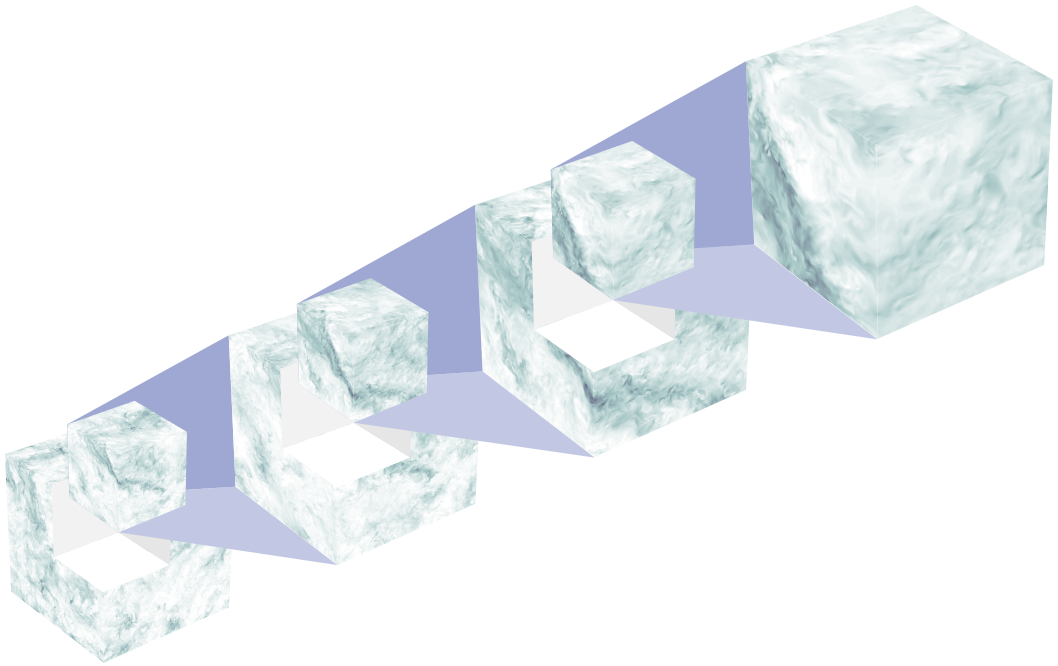

Figure 24: Snapshot of homogeneous isotropic turbulence from the Johns Hopkins Turbulence Database [89, 66, 120], showing the kinetic energy per unit mass, with darker colour corresponding to greater energy.

orthogonal basis whose elements are localized in both space and scale. Traditionally, the basis elements are translations and dilations of a single vector called the mother wavelet [103, 22, 79, 74, 36]. The Supplementary Information provides a concise summary of results relevant to the present work. Traditional wavelet methods (where the mother wavelet is prescribed a priori) have already found use in turbulence precisely because of the space-scale unfolding they produce [2, 27, 28, 29, 30, 82, 98, 117, 78], giving hope that data-driven methods based on wavelets may lead to new insights into turbulence.

A myriad of data-driven methods of structure identification and extraction based on wavelets have been developed (e.g., [39, 42, 44, 77, 85, 95, 94, 93, 101, 110, 124]). Although these methods may yield localized structures, they are limited in that the construction of the resulting basis elements is prescribed in either scale or frequency, and many impose self-similarity on the basis, as is done with traditional wavelets. (The "empirical wavelet transform" of [39] does not have this feature, but relies on the existence of local maxima in the power spectrum of a signal, making it ill-suited to phenomena like turbulence without such local maxima.)

In the present work we develop a method that integrates the data- and energy-driven nature of PCA with the space and scale localization properties of wavelets. As our derivation and illustrative examples will reveal, we impose very little structure in our method, so any structure in the basis may be attributed to the underlying structure of the dataset under consideration. We call the resulting basis a "data-driven wavelet decomposition" (DDWD), and use it to gain insights into the structure of turbulence, though we emphasize that the method is general in its application.

2.3.2 Formulation

Before presenting the DDWD, it will be useful to introduce key features of PCA and conventional wavelet decompositions. Suppose we have a dataset $\{z_i\}_{i=1}^{M} \in \mathbb{R}^N$, each z_i being a sample data vector (e.g., one component of a velocity field uniformly sampled along a line through the flow). We can arrange the dataset into

a matrix $Z \in \mathbb{R}^{N\times M}$ whose columns are the data vectors z_i, normalized so that $\operatorname{tr} ZZ^T = 1$ (the normalization does not change the results of PCA, but is done here because it parallels our formulation of DDWD later). PCA seeks an ordered orthonormal basis $\{\varphi_i\}_{i=1}^N$ such that the energy of the dataset projected onto the first $K \leq N$ basis elements is maximized. One way to state this problem, which parallels our later description of data-driven wavelets, is as follows. We determine the first basis element φ_1 so that the projection of the data onto this element is maximized. This problem can be written

$$\max_{\varphi} \quad \varphi^T ZZ^T \varphi \tag{15}$$

$$\text{s.t.} \quad \varphi^T \varphi = 1. \tag{16}$$

The solution to this problem is the eigenvector of ZZ^T with the largest eigenvalue. The second basis element φ_2 is found by projecting out the component of the data in the φ_1 direction and repeating, yielding that φ_2 is the eigenvector of ZZ^T with the second largest eigenvalue. Basis elements φ_i solve

$$\max_{\varphi} \quad \left\| \varphi^T \left(Z - \textstyle\sum_{j=1}^{i-1} \varphi_j \varphi_j^T Z \right) \right\|_2^2 \tag{17}$$

$$\text{s.t.} \quad \varphi^T \varphi = 1, \quad \varphi^T \varphi_j = 0, \, j = 1, \ldots, i-1. \tag{18}$$

This formulation is recursive, producing a hierarchy of subspaces ordered by how much of the dataset's energy (Frobenius norm) they contain: $\mathbb{R}^N = \operatorname{span}\{\varphi_1\} \oplus \ldots \oplus \operatorname{span}\{\varphi_N\}$. The basis elements φ_i are the eigenvectors of ZZ^T. For statistically homogeneous data in a periodic domain, ZZ^T (more precisely, its expected value) is circulant, in which case the φ_i are simply discrete Fourier modes.

Traditional wavelet decompositions also produce a hierarchy of orthogonal subspaces, but there are important differences from PCA. First, the basis elements are not determined from data, but are selected a priori; there are many standard options [74]. Second, by construction, the decomposition produces a hierarchy of orthogonal subspaces ordered by scale, as shown in Figure 25(a). We consider periodic vectors on $\mathbb{R}^N$, with N even [36]. This space is split into subspaces V_{-1} and W_{-1}, each of dimension $N/2$, and each spanned by the even translates of vectors φ_{-1} (the father wavelet) and ψ_{-1} (the mother wavelet), respectively. Once φ_{-1} is known, ψ_{-1} can be found, and vice versa. The father and mother wavelets, and their even translates, are mutually orthonormal by construction. Subspace V_{-1} is called an approximation subspace because it contains all the low frequencies, and W_{-1} is called a detail subspace because it contains all the high frequencies. Given a signal, its projection onto V_{-1} produces a low-pass filtered version of the signal, and its projection onto W_{-1} produces the detail needed to reconstruct the full signal. We then recursively split the approximation subspaces. For $N = 2^p$ (which we assume throughout), we get a hierarchy of subspaces of progressively coarser scales: $\mathbb{R}^N = W_{-1} \oplus \ldots \oplus W_{-p} \oplus V_{-p}$. For traditional wavelets, the sets of wavelets $\{\varphi_i\}$ and $\{\psi_i\}$ are determined from the father and mother wavelets, respectively, by a rescaling operation that is essentially a simple dilation by a factor of two (see the Supplementary Information of the published paper [33] for more details). This process leads to a hierarchical basis structure of the form shown in Figure 25(a).

The DDWD combines the hierarchical structure of wavelets that is shown in Figure 25(a) with the energetic optimization of PCA. Namely, each time we split a subspace, we design the subsequent subspaces so that the approximation subspace contains as much of the dataset's energy as possible.

The first step of the process is to find the *wavelet generator* u, for which the projection of the data onto this vector and its even translates is maximized. We define V_{-1} as the subspace spanned by these vectors, thus beginning the data-driven construction of a hierarchy with the structure of Figure 25(a). This maximization is subject to (1) the constraint that u and its even translates are mutually orthonormal, and (2) a penalty on

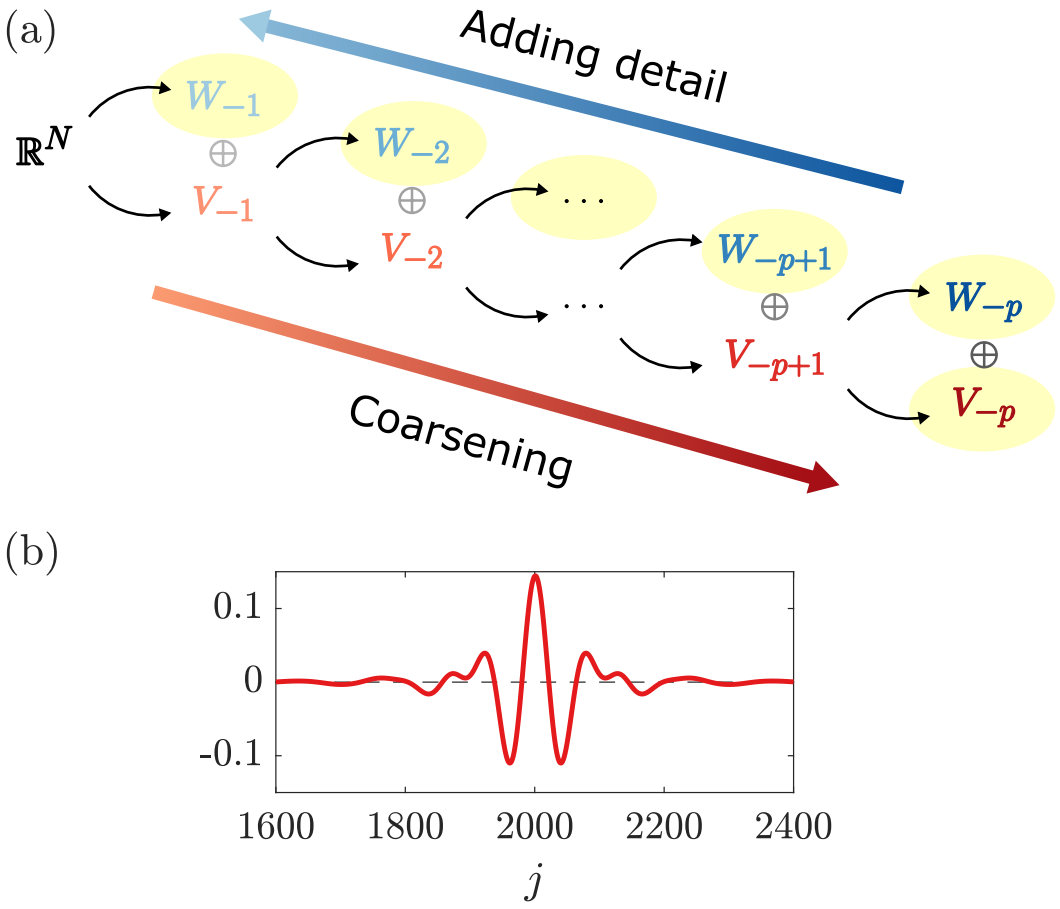

Figure 25: (a) Subspaces from wavelets on $\mathbb{R}^N$. At stage l, approximation subspace V_{-l} is split into detail subspace W_{-l-1} and approximation subspace V_{-l-1}, each half the dimension of V_{-l}. Subspace V_{-l} is spanned by the $N/2^l$ translates by 2^l of φ_{-l}, and W_{-l} is spanned by the $N/2^l$ translates by 2^l of ψ_{-l}. The full space is decomposed into progressively coarser subspaces, $\mathbb{R}^N = W_{-1} \oplus \ldots \oplus W_{-p} \oplus V_{-p}$, or, going the other way, into the addition of progressively finer details. These subspaces are highlighted. In the present work, an ensemble of data is used to define a specific decomposition of this form. (b) Discrete Meyer wavelet for $N = 4096$ and $l = 6$.

the width of u, as measured by its circular variance $\mathrm{Var}(u)$. This problem is stated as

$$\max_u \quad u^T A u - \lambda^2 \mathrm{Var}(u), \quad A = \frac{1}{\|Z\|_F^2} \sum_{k=0}^{N/2-1} R^{-2k} Z Z^T R^{2k} \tag{19}$$

$$\text{s.t.} \qquad u^T R^{2k} u = \delta_{k0}, \; k = 0, \ldots, N/2-1. \tag{20}$$

Here λ measures the penalty on the variance, whose effect on the results we illustrate below, and R is the circular shift operator: e.g., if $u = [a, b, c, d]^T$, then $Ru = [d, a, b, c]^T$. The solution u and its even translates generate the vectors φ_{-1} and ψ_{-1}; the former span V_{-1} and the latter W_{-1}. We then project the data onto V_{-1}, replace N by $N/2$ in the definition of A and the orthonormality constraints, decrease λ by a factor of 2, and repeat, yielding φ_{-2} and ψ_{-2}, and thus the subspaces V_{-2} and W_{-2}. We proceed recursively, finding the subspaces V_{-l} and W_{-l} such that V_{-l} contains the maximal amount of energy of the dataset. In the end, we find an energetic hierarchy of subspaces, optimized stage by stage, whose elements are orthogonal

and localized. In contrast to previous data-driven methods incorporating wavelets, which impose restrictive structure, the only structure we impose is orthogonality, localization, and the hierarchy of Figure 25(a). In the Supplementary Information of the published paper [33], we also draw parallels between the DDWD and convolutional neural networks, and show how the DDWD naturally incorporates pooling and skip connections, two tricks that improve the performance of neural network architectures [41]. Together with its inverse transform, the DDWD is akin to a convolutional autoencoder, but with the additional features of orthogonality of all elements, stage-wise energetic optimality, and the ability to unambiguously extract structure, which make the results interpretable.

We make a point to note that for the DDWD, the stage l of the hierarchy should not be conflated with the concept of scale. For traditional wavelets, stage and scale are interchangeable since whenever a subspace is split, the lower half of frequencies is always pushed to the approximation subspace and the upper half of frequencies is always pushed to the detail subspace. For the DDWD, however, the distribution of frequencies amongst the subspaces is dictated by energetic considerations, which depends on the dataset under consideration. An example below will elucidate this point.

2.3.3 Results

We will demonstrate the DDWD on three datasets with increasingly complex structure to show that the method extracts structure inherent to the data.

Gaussian random data The first dataset we consider consists of Gaussian white noise, which has no structure. By construction, the basis produced by the DDWD is orthonormal, so the change-of-basis transformation is orthogonal. Any orthogonal transformation of Gaussian white noise produces Gaussian white noise. Therefore, in the absence of a variance penalty, applied to Gaussian white noise, the coordinates of the data in the DDWD basis (the wavelet coefficients) will be Gaussian white noise, so all wavelet coefficients will be uncorrelated and have variance equal to that of the input Gaussian white noise. That is, as long as we do not impose a variance penalty, this result implies that for Gaussian white noise there is no optimal set of wavelets, in the sense we have defined. In other words, the DDWD reflects that the dataset has no structure. If we do impose a variance penalty, then the optimal wavelets become discrete delta functions (i.e., the Euclidean basis vectors). The reason for this is simple: all wavelets capture the energy of white noise equally well, but the delta function will be the most localized among them.

The result that all wavelets capture the energy of Gaussian white noise equally well highlights an interesting fact about the DDWD. In Figure 26, we show three sets of wavelets that are computed from a dataset of Gaussian white noise. Figure 26(a) has no variance penalty, Figure 26(b) has a small variance penalty, Figure 26(c) has a large variance penalty, and all wavelets are coloured according to the colour coding used in Figure 25(a). Despite the fact that we have used the structure of Figure 25(a), there is no apparent hierarchy of scales among the left set of wavelets. This highlights what we noted earlier, that the concept of scale is not built into the DDWD; rather, it must be learned from the data. When we add a small variance penalty, wavelets corresponding to finer detail subspaces are more localized, but all wavelets are jagged; this will contrast with our later examples where wavelets corresponding to later stages are smoother, reflecting the inherent structure of the later examples. Note that although the central set of wavelets was computed with non-zero variance penalty, they are not delta functions as we had asserted earlier; this is due to the dataset containing finite samples, and this effect weakens as the number of samples increases or as the variance penalty is increased (as for the right set of wavelets). In Figure 26(c), all of the vectors are discrete delta functions: while this might seem redundant, only certain translates of the discrete delta function are included in each stage; the resulting basis consists of delta functions localized at each mesh point.

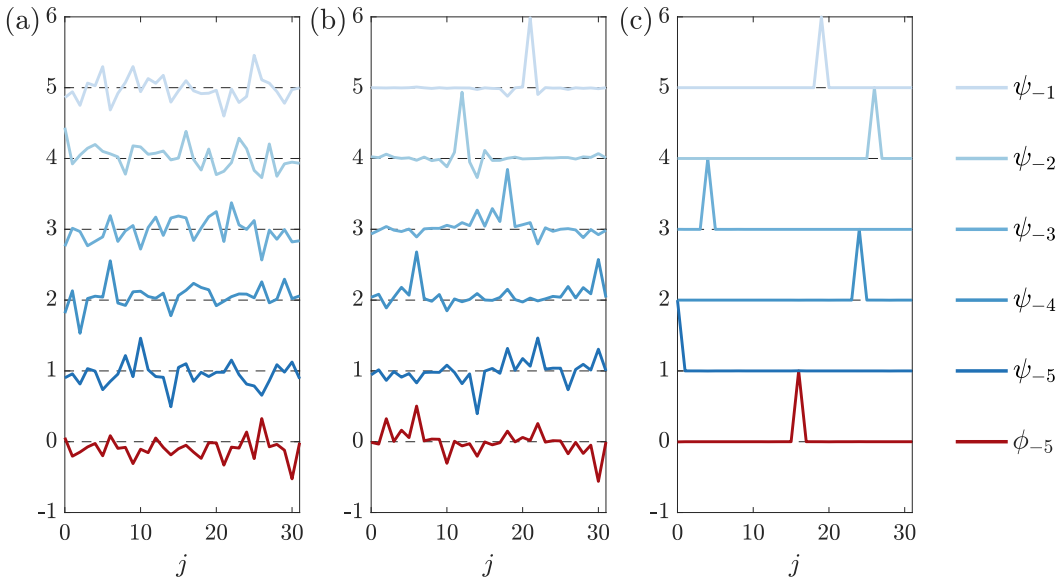

Figure 26: White noise wavelets on $\mathbb{R}^{25}$. Colouring as in Figure 25(a). No variance penalty (a), small variance penalty (b), and large variance penalty (c).

Kuramoto-Sivashinsky chaos The second dataset we consider comes from the Kuramoto-Sivashinsky equation,

$$u_t + uu_x + u_{xx} + vu_{xxxx} = 0 \tag{21}$$

for $0 \leq x \leq 2\pi$, with periodic boundary conditions and $v = (\pi/11)^2$, which yields chaotic dynamics. We compute a numerical solution using a pseudo-spectral method with 64 Fourier modes, and assemble a dataset consisting of 90 001 snapshots taken from a single trajectory. The latter part of the trajectory and the power spectrum in Figure 27 clearly show that the structure is dominated by a single length scale with wavenumber k around 2–3.

We compute the DDWD with a range of variance penalties, showing the result for $\lambda^2 = 0.1$ in Figure 28 (others are shown in the Supplementary Information). We only show one set of wavelets because, no matter the variance penalty, the coarsest subspaces are the same: V_{-6} is spanned by a sine wave with wavenumber $k = 2$ (the most energetic wavenumber), W_{-6} is spanned by a sine wave with wavenumber $k = 3$ (the second most energetic wavenumber), and W_{-5} is spanned by a vector (and its translate) containing only wavenumbers $k = 3$ and 4 ($k = 4$ is the next most energetic wavenumber). The DDWD is thus robust in pushing the dominant (most energetic) length scales of the system to the lowest stages. Moreover, the energy contained in each subspace is also robust to the variance penalty. The first difference between wavelets computed with different variance penalties appears in the subspace W_{-4}, spanned by the four translates of ψ_{-4}. As the variance penalty is increased, the wavenumber $k = 8$ is exchanged for $k = 0$. Energetically, this makes little difference since $k = 8$ is highly damped by the hyperviscous term and contains very little energy, and $k = 0$ contains identically zero energy (for the boundary conditions we use, the spatial mean is constant and can be set to zero). The compositions of the finer detail subspaces do not change qualitatively with variance penalty, with finer detail subspaces containing higher wavenumbers. As the variance penalty is increased, localization in the Fourier domain is exchanged for localization in the spatial domain.

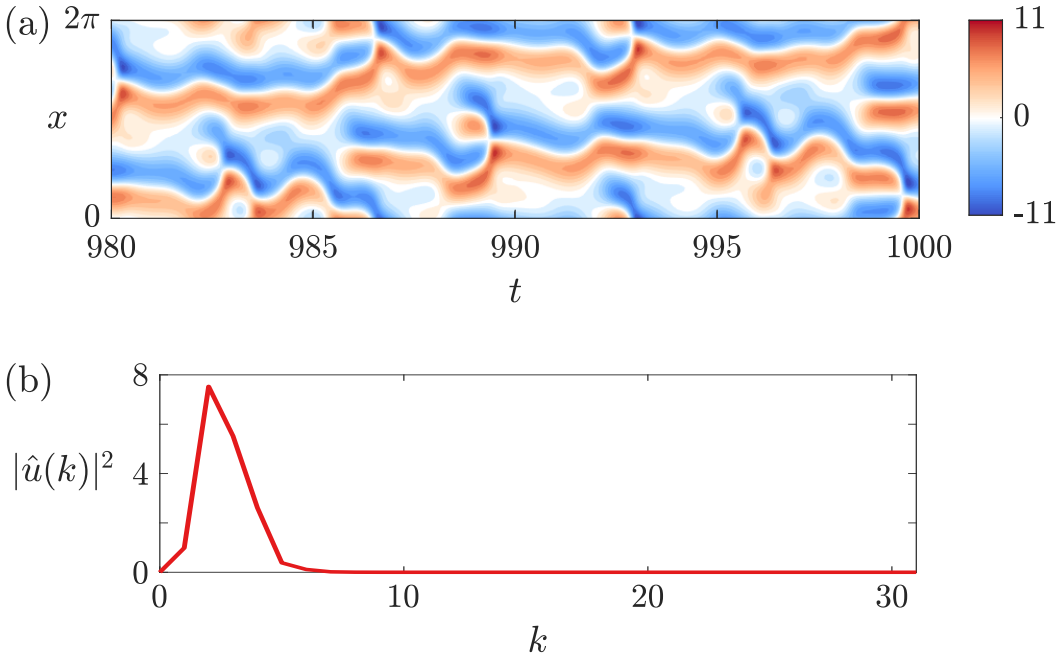

Figure 27: Trajectory (a) and attendant power spectrum (b) of the Kuramoto-Sivashinsky equation.

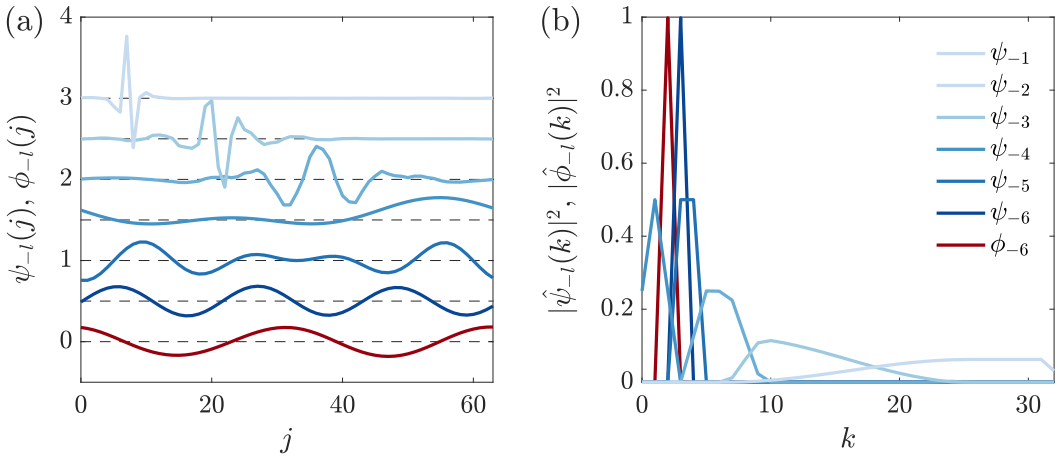

Figure 28: Kuramoto-Sivashinsky wavelets (a), offset from each other by 0.5, and their power spectra (b). Colouring as in Figure 25(a). The variance penalty is $\lambda^2 = 0.1$.

Homogeneous isotropic turbulence The final and primary dataset we consider is of forced homogeneous isotropic turbulence, taken from the Johns Hopkins Turbulence Database [3] [89, 66, 120]. We use a single snapshot from a direct numerical simulation on a 4096^3 periodic grid with a Taylor-scale Reynolds number of 610.57, shown in Figure 24; more details are available in the database's documentation. Our dataset consists of the velocity component aligned with 16384 randomly sampled lines (the "longitudinal velocity") that are parallel to the axes. Each sample is a vector of length $N = 4096$. The power spectrum is broad and has the expected $-5/3$ power law in the inertial subrange, which roughly contains wavenumbers $k \in [2, 60]$.

Figure 29 shows the DDWD with various variance penalties. While at $\lambda^2 = 10^{-1}$, the wavelets are well-localized only for $l \leq 5$, for $\lambda^2 = 10^0$ and 10^1, localization is observed for $l \leq 8$ and 9, respectively. Moreover, despite the order of magnitude difference in λ^2 between the latter two cases, the wavelets for $4 \leq l \leq 8$ are nearly indistinguishable. Furthermore, with increasing l, the wavelets have increasing scale: the DDWD reveals a hierarchy of scales present in the dataset, a known feature of turbulence. Recall that this feature is not built into the DDWD; rather, the method has extracted the concept of scale hierarchy from the turbulence dataset. In this case, it is appropriate to conflate stage and scale.

It is also worth noting that with increasing variance penalty, the composition of each scale in the Fourier domain becomes smoother and more robust, varying less across different trials. Overall, the composition of the wavelets in the Fourier domain is robust to the variance penalty.

To illustrate the reconstruction of data vectors using the DDWD basis, Figure 30(a) shows one vector from the turbulence dataset and its projections onto the subspaces V_{-l} computed with $\lambda^2 = 10^1$. Lighter colours show more detailed reconstructions and the thin black line shows the original data vector. At the coarsest level of approximation, we essentially reconstruct the spatial mean, and then add progressively finer scale features as we add smaller scale wavelet components. Figures 30(b) and (c), respectively, show the reconstruction errors of the progressively finer projections, and the energy of the entire dataset contained in each stage, for $\lambda^2 = 0, 10^{-1}, 10^0$, and 10^1. The differences in these quantities as λ changes are visibly indistinguishable, indicating robustness of the DDWD with respect to variance penalty.

Most interestingly, we check the wavelets that arise from the HIT data for self-similarity across stages. We present here results for the most localized wavelets, corresponding to $\lambda^2 = 10^1$, and have found that the same conclusions hold for $\lambda^2 = 10^0$. Figures 31(a)–(e) show wavelets ψ_{-l} for $4 \leq l \leq 8$; note the change in horizontal scale from plot to plot. Aside from their horizontal scale, these wavelets are evidently very similar looking. The figure also shows on each plot the rescaled version of the wavelet at the previous level, $S\psi_{-l+1}$, where S essentially dilates a vector by a factor of two and rescales it so that it has unit norm. For ease of comparison, we have shifted the wavelets and in some cases reflected them about their axes. In all cases shown, ψ_{-l} and $S\psi_{-l+1}$ are nearly indistinguishable, indicating strong self-similarity across stages $l = 4$ to $l = 8$. This observation can be quantified: Figure 31(f) shows the inner product $\psi_{-l}^T S\psi_{-l+1}$, whose absolute value is bounded by 0 and 1, for all stages. It is very close to unity for $l > 3$. This strong self-similarity also holds for the lower variance penalty $\lambda^2 = 10^0$, indicating that it is a robust feature derived from the data. Stages 4–8 contain the approximate wavenumbers $k \in [10, 200]$, which coincides with the inertial subrange where self-similarity is expected. (The larger scales are no longer localized, so we draw no significance from the high measure of similarity in those cases.) Interestingly, the wavelets in the self-similar range are quite similar to the discrete Meyer wavelet [74], shown in Figure 25(b), as well as to the Battle-Lemarié wavelet used by Meneveau in his analysis of turbulent flows [78]. Performing Meneveau's analysis with our data-driven wavelets would likely yield similar results, at least in the self-similar range.

It bears repeating that the self-similarity of the wavelets produced by the DDWD is not a result of the method, rather it is a reflection of the system. In the case of the Kuramoto-Sivashinsky system, where we know there is no similarity across scales, there is generally no relation between the data-driven wavelets across scales. For HIT, where self-similarity is hypothesized in a certain range of scales, the data-driven

[3] http://turbulence.pha.jhu.edu

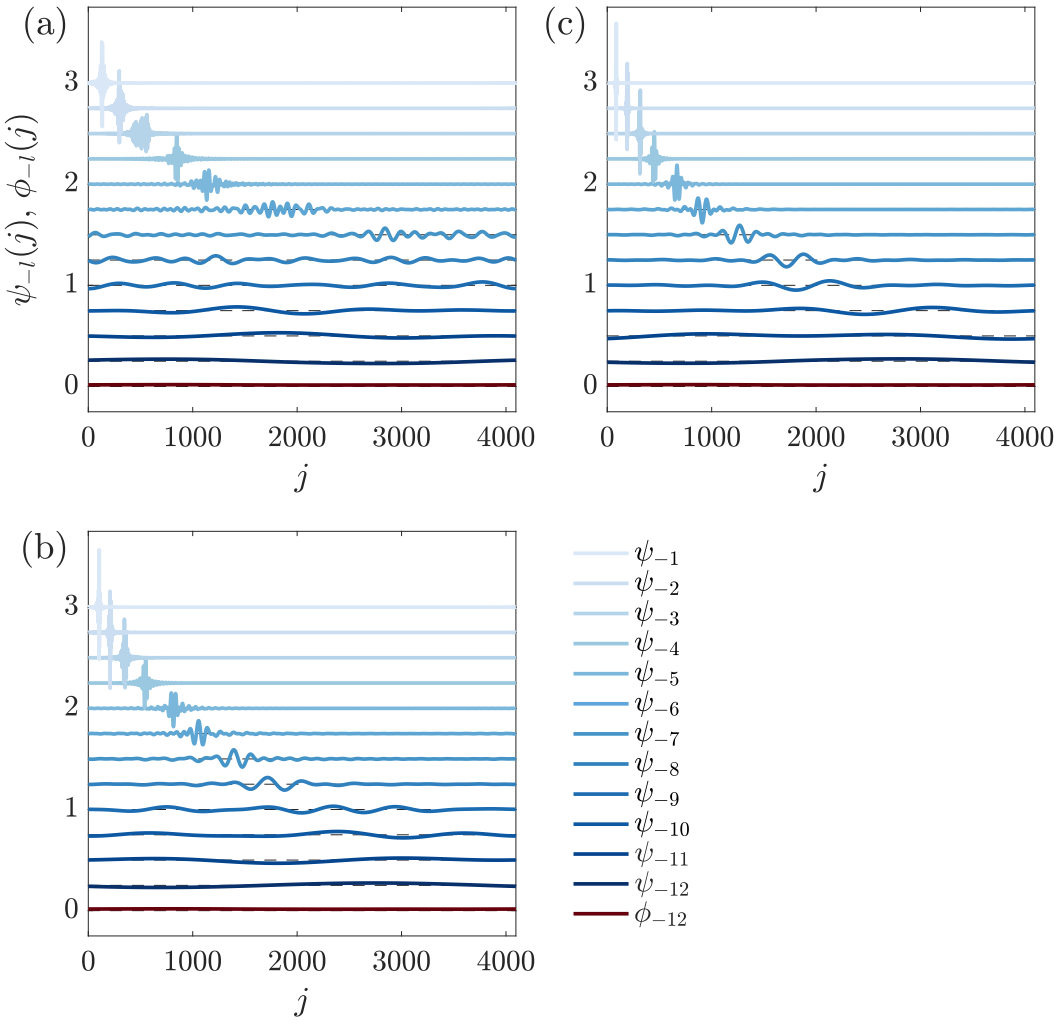

Figure 29: HIT wavelets, vertically offset from each other by 0.25. Colouring as in Figure 25(a). The variance penalties are $\lambda^2 = 10^{-1}$ (a), $\lambda^2 = 10^{0}$ (b), and $\lambda^2 = 10^{1}$ (c).

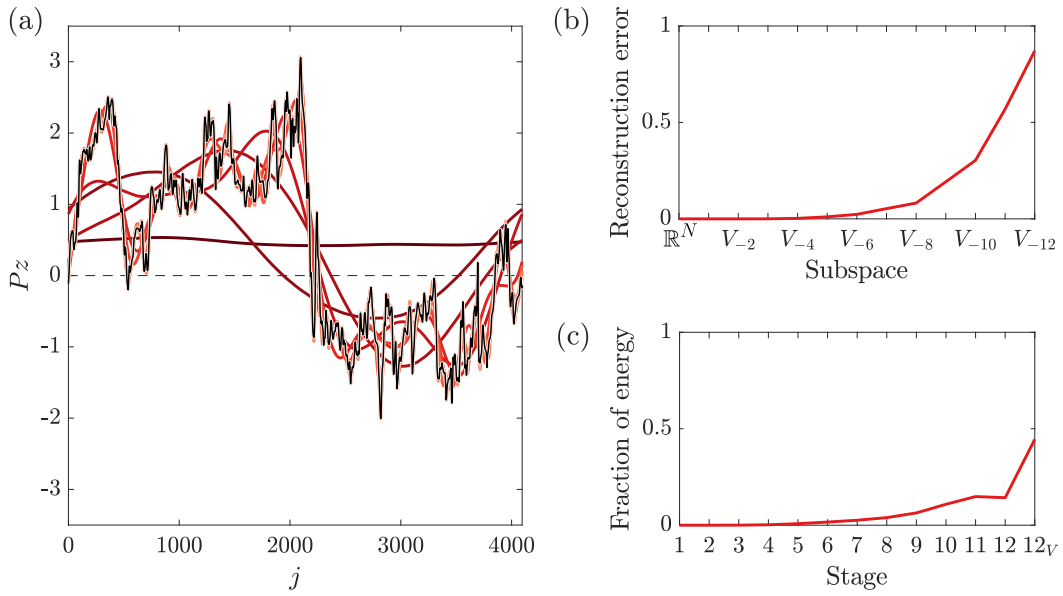

Figure 30: Projection (denoted P) of one vector (denoted z) in the turbulence dataset onto the subspaces V_{-l} computed with $\lambda^2 = 10^1$ (a), with colouring as in Figure 25(a). The thin dashed line shows the origin, and the thin solid line shows the original vector. Also shown are the reconstruction error of each projection (a), and the energy of the dataset contained in each stage for all variance penalties considered (b) ($\lambda^2 = 0, 10^{-1}, 10^0$, and 10^1; only the result for $\lambda^2 = 10^1$ (red) can be seen).

wavelets show self-similarity. Hellström et al. [47] made a somewhat related observation in turbulent pipe flow. They performed PCA on a set of experimentally obtained velocity fields from a cross-section of the pipe, and found that they could rescale the modes so that they overlapped. This observation is consistent with the attached eddy hypothesis about the structure of wall-turbulence [75, 54]. Their modes were global in space, as usually results from PCA; this is particularly true for the azimuthal direction, for which PCA yields Fourier modes due to periodicity. For the HIT data, which is periodic in all three directions, PCA would yield Fourier modes in all three directions, revealing no information about the system that could not be obtained from Fourier decomposition.

2.3.4 Conclusions

We have introduced a method that integrates key aspects of PCA and wavelet analysis to yield a data-driven wavelet decomposition. This method takes an ensemble of data vectors corresponding to field values at a lattice of points in space (or time) and generates a hierarchical orthogonal basis. In contrast to traditional wavelet bases, the basis elements at each stage are not simply dilations of given mother or father wavelets, but rather are determined stage-by-stage from the data. For data that is not self-similar, neither are the resulting basis elements. Rather, these represent the differing structures at the different stages. In contrast, for self-similar data, the basis vectors at different stages are related to one another by a simple rescaling. Indeed, for data from homogeneous isotropic turbulence—a high-dimensional, nonlinear, multiscale process—we show self-similarity of the wavelet basis elements, which in turn reveals the self-similarity of the data, providing direct evidence for a century-old phenomenological picture of turbulence.

Future work on the DDWD will need to extend the methodology to multiple dimensions, different boundary conditions, and unstructured domains. As a start, tensor products can be used to address the first issue, boundary wavelets can be used to address the second issue [74], and wavelets on graphs can be used to

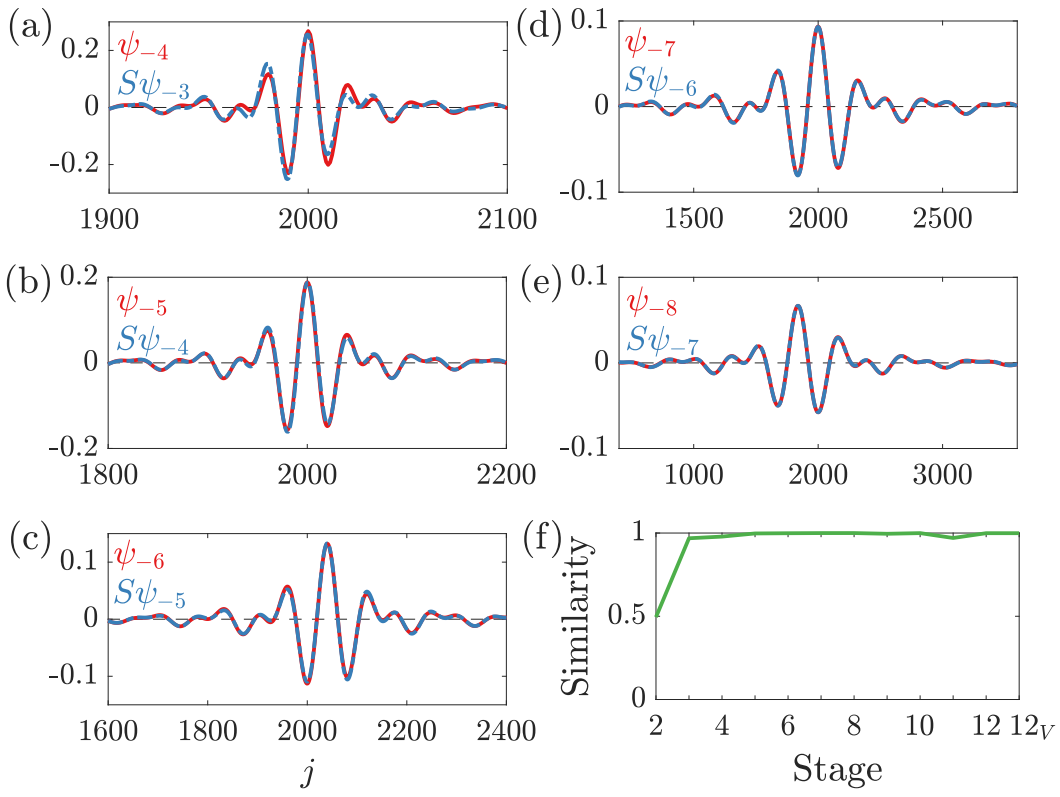

Figure 31: Comparison between computed wavelets ($\lambda^2 = 10^1$) and ones obtained by dilating and rescaling the wavelet from the previous stage for stages $l = 4$ to $l = 8$ (a–e), and the level of similarity across all stages (f).

address the last issue [46]. For incompressible fluid flows, velocity fields are vector-valued and divergence-free; Farge et al. [30] provides a few options to handle this case that may be generalizable to the data-driven case. Attention must also be given to the development of efficient optimization algorithms for computing the basis. Finally, based on the ability of the present method to extract self-similar basis elements from self-similar turbulent flow data, we view it as a potentially important new starting point for identification and characterization of localized hierarchical turbulent structures in a wide variety of fluid flows, as well as other complex multiscale systems. We are particularly interested in applying the DDWD to wall-bounded flows and making connections with the attached eddy model of turbulence.

3 Dissemination

3.1 Publications

1. Graham, M. D., and Floryan, D., "Exact coherent states and the nonlinear dynamics of wall-bounded turbulent flows", *Annual Review of Fluid Mechanics*, 53 227-253 (2021).

2. Zeng, K., Linot, A. J., and Graham M. D., "Data-driven control of spatiotemporal chaos with reduced-order neural ODE-based models and reinforcement learning", submitted (2022). Preprint arxiv:2205.00579v1

3. Linot, A. J., Burby, J. W.,Tang, Q., Balaprakash, P., Graham, M. D., and Maulik, R. "Stabilized Neural Ordinary Differential Equations for Long-Time Forecasting of Dynamical Systems", submitted (2022). Preprint arxiv:2203.15706v1

4. Zeng, K., Linot, A. J. and Graham, M. D. "Learning turbulence control strategies with data-driven reduced-order models and deep reinforcement learning", 12th International Symposium on Turbulence and Shear Flow Phenomena (TSFP12) Osaka, Japan (Online), July 19-22, 2022.

5. Floryan, D. and Graham, M. D., "Charts and atlases for nonlinear data-driven models of dynamics on manifolds", submitted (2021). Preprint arXiv:2108.05928.

6. Linot, A. J. and Graham, M. D., "Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations", *Chaos*, to appear (2022). Preprint arXiv:2109.00060

7. Zeng, K. and Graham, M. D., "Symmetry reduction for deep reinforcement learning active control of chaotic spatiotemporal dynamics", *Phys. Rev. E*, 104, 014210 (2021).

8. Floryan, D. and Graham, M. D., "Discovering multiscale and self-similar structure with data-driven wavelets", *Proceedings of the National Academy of Sciences* 118(1), e2021299118 (2021).

9. Agrawal, R., Ng, H. C.-H., Davies, E. A., Park, J. S., Graham, M. D., Dennis, D. J. C. and Poole, R. J., "Low- and high-drag intermittencies in turbulent channel flows", *Entropy*, 22(10), 112 (2020).

10. Linot, A. J. and Graham, M. D., "Deep learning to discover and predict dynamics on an inertial manifold", *Phys. Rev. E*, 101, 062209 (2020).

11. Whalley, R., Dennis, D. Graham, M. D. and Poole, R.J., "An experimental investigation into spatiotemporal intermittencies in turbulent channel flow close to transition", *Expts. Fluids*, 60, 301 (2019).

3.2 Invited presentations by the PI describing research from this grant

1. Workshop on Challenges and Benchmarks for Quantitative AI in Complex Fluids and Complex Flows, Rome Italy, July 6-8, 2022.
2. Dept. of Physics and Astronomy, Univ. of Edinburgh, May 31, 2022.
3. Julian C. Smith Lectures, School of Chemical and Biomolecular Engineering, Cornell Univ., May 2-3, 2022.
4. Isaac Newton Institute workshop, "Wall-bounded turbulence: beyond current boundaries", Cambridge, UK, March 28-April 1, 2022.
5. Fluid dynamics seminar, Department of Aeronautics, Imperial College, London, UK, March 16, 2022.
6. Clarkson Center for Complex Systems Science, Jan. 28, 2022.
7. Flatiron Institute workshop, "Mechanics of Life", New York, Jan. 11-13, 2022. (Postponed)
8. US-Mexico Symposium on Advances in Polymer Science (MACROMEX 2021), Riviera Maya, Mexico, Nov. 2021.
9. AIChE Annual Meeting, Invited session in memory of Paul Steen, Nov. 2021.
10. Nonlinear dynamics seminar (online), Georgia Tech., Sept. 29 & Oct. 6, 2021. With the flexibility of the online format, we did something a bit different here, in collaboration with the seminar organizer: at the beginning of each lecture, MDG gave an overview, then two group members gave 20 minute talk on their specific research.
11. Penn Institute for Computational Science colloquium, online, Oct. 23, 2020.
12. Workshop on Phase Space Analysis in Complex Systems - From Quantum Dynamics to Turbulent Flows, Dresden, Germany, April 7-9, 2020. Canceled due to COVID-19.
13. Department of Aerospace Engineering and Mechanics, Univ. of Minnesota, March 20, 2020. Postponed due to COVID-19.
14. US-Japan Workshop on Data-Driven Fluid Dynamics, Kobe, Japan, March 12-14, 2020. Postponed due to COVID-19.
15. American Physical Society Annual Meeting, Invited minisymposium in memory of Bruno Eckhardt, March 2, 2020. Virtual session due to COVID-19.
16. Flatiron Institute workshop, "Universality: Turbulence Across Vast Scales", New York, Dec. 2-6, 2019.
17. William R. Schowalter Lecturer, American Institute of Chemical Engineers Annual Meeting, Nov. 2019.
18. AIChE Annual Meeting, Invited session in honor of Yannis Kevrekidis, Nov. 2019.
19. Computing in Engineering Symposium, Univ. of Wisconsin-Madison, Sept. 10, 2019.
20. Workshop on Mathematical Fluids, Materials, and Biology, Univ. of Michigan, June 13-15, 2019.
21. AIChE Annual Meeting, Invited session on "Novel Complex Flows", Oct. 2018.

22. Department of Chemical Engineering and Materials Science Seminar, Univ. of Minnesota, Oct. 23, 2018.

23. IUTAM Symposium on Stochastic dynamical systems approaches to fluid flow transitions, Cornell University, Sept. 12-14, 2018.

3.3 Contributed presentations describing research from this grant

1. A. J. Linot and M. D. Graham. "Modeling Chaotic Spatiotemporal Dynamics with a Minimal Representation Using Neural ODEs." AIChE Annual Meeting, Boston, MA, USA, November 7–11, 2021.

2. A. J. Linot and M. D. Graham. "Learning minimal representations for chaotic dynamics of partial differential equations." Proceedings of the 73rd Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Virtual, November 22–24, 2020.

3. Zeng, K., Linot, A. J., Graham, M. D., "Deep Reinforcement Learning Using Data-Driven Reduced-Order Models Discovers and Stabilizes Low Dissipation Equilibria", 74th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Nov. 21-23, 2021.

4. Zeng, K. and Graham, M. D., " Symmetry Reduction for Deep Reinforcement Learning Active Flow Control", American Institute of Chemical Engineers Annual Meeting, Nov. 7-11, 2021.

5. Pérez De Jesús, C. and Graham, M. D. "Data-driven dynamics of Kolmogorov flow on the inertial manifold", 74th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Nov. 21–23, 2021.

6. Floryan, D. and Graham, M. D. "Charts and atlases for nonlinear data-driven dynamics on a manifold", 74th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Nov. 21–23, 2021.

7. Floryan, D., Guo, A., and Graham, M. D. "A hierarchy of spatially localized, self-similar structures in turbulent pipe flow", 74th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Nov. 21–23, 2021.

8. Floryan, D., Guo, A., and Graham, M. D. "Discovering localized, multidimensional, and multiscale structure with data-driven wavelets", 25th International Congress of Theoretical and Applied Mechanics, Aug. 22–27, 2021.

9. Linot A. J. and Graham, M. D. "Learning minimal representations for chaotic dynamics of partial differential equations", 73nd Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Online, Nov. 22–24, 2020.

10. Linot, A. J. and Graham, M. D., "Deep Learning for Recreating Chaotic Dynamics of Partial Differential Equations with a Minimal Representation", AIChE Annual Meeting, Nov. 2020.

11. Floryan, D. and Graham, M. D. "Discovering multiscale structure using data-driven wavelets", UW-Madison Computing in Engineering Forum, Sept. 2020.

12. Floryan, D. and Graham,M. D. "Revealing self-similar structure with a data-driven wavelet decomposition", 73nd Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Online, Nov. 22–24, 2020.

13. Zeng,K. and Graham, M. D. “Symmetry Reduction for Deep Reinforcement Learning Active Flow Control”, 73nd Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Online, Nov. 22–24, 2020.

14. A. J. Linot and M. D. Graham. ”Nonlinear dimensionality reduction and prediction of chaotic spatiotemporal dynamics in translation-symmetric systems via deep learning.” Proceedings of the 72nd Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Seattle, WA, USA, November 23–26, 2019.

15. A. J. Linot and M. D. Graham. ”Spatially Invariant K-Means Clustering for Analyzing the Turbulent Attractor of Minimal Channel Flow.” AIChE Annual Meeting, Orlando, FL, USA, November 10–15, 2019.

16. A. J. Linot and M. D. Graham. ”Deep learning to discover and predict dynamics on an inertial manifold.” Georgia Tech Nonlinear Science Seminar & Webinar, January 28, 2020.

3.4 Outreach

The PI and his research group are active in UW-Madison campus outreach. The PI in conjunction with the Institute for Chemical Education at UW has developed a new an undergraduate special topics course, CBE 562, “Chemical Engineers in the Community”. This is a service-learning course that connects Chemical Engineering students with their local community through the development and implementation of hands-on inquiry based engineering lessons for middle school level after-school science programs. Students will learn how to communicate complex science and engineering topics to learners at all levels, learn about how to work with a middle school students from diverse backgrounds, and gain a basic understanding of how students learn. Although the PI spearheaded this, a number of other faculty members and their research groups are already participating or have committed to do so. The idea is that the department can generate both a broad repertoire of activities and a high level of student and faculty involvement.

Because of COVID, the last time this program ran was F18. (It’s on the books again for F22.) Students developed demos and projects on the Bernoulli effect, liquid-liquid phase transitions, non-Newtonian flows, surface tension, buoyancy, viscosity and mixing. They participated in Engineer’s Day, a campus public outreach event, and also participated in the Wisconsin Science Festival on the UW-Madison campus. Figure 32 shows K-12 students learning about surface tension and viscosity with UW undergraduates.

In addition, the PI’s research group participates in the UW College of Engineering’s semiannual Engineering Expo. This is a semiannual event that brings K-12 kids to the campus for demostrations and activities involving engineering. The group members put together demos and activities on the topic of fluid dynamics.

4 Impacts

4.1 Development of the principal disciplines of the project

- The work performed here was the first to show that a data-driven reduced-order modeling framework could yield quantitative short-time predictions and long-time statistics for a dynamical system whose ambient state space might have dimension of 10^5 and whose long-term dynamics lie on a manifold of dimension $10-100$, and perhaps higher. Prior demonstrations involved very simple systems with ambient space dimensions of $\sim 10-100$ and manifold dimension of three (e.g. the Lorenz model).

- We have demonstrated that with this highly efficient framework, a nonlinear reinforcement learning control algorithm for turbulent plane Couette flow can now be trained using the reduced order model in a matter of hours, when using the full DNS would take months.

Figure 32: K-12 students and UW-Madison undergraduates from the PI's course "Chemical Engineers in the Community" studying interfacial tension (front) and viscosity (rear) at the Wisconsin Science Festival, Oct. 2018.

- In very recent work motivated by the approach pioneered in this grant, we have developed a new algorithm that enables very precise determination of the dimension of the manifold where a high-dimensional data set lie, and efficient representation of that manifold. Previously, no reliable method had been available, so all current approaches to reduced order modeling for high-dimensional systems require guesswork and iteration to find a good dimension for the representation. This new approach will thus supersede all current nonlinear dimension reduction approaches. Furthermore, now that we have a method to determine the exact dimension of the manifold where the dynamics lie, we can systematically develop and understand approximate representations with fewer dimensions, which may tie in with, for example, improved subbrid models for complex turbulent flows. In a real turbulent flow, even the reduced-order model with exactly the right number of dimensions may too high-dimensional for incorporation into a controller design algorithm, but the new methods open ways to systematically eliminating degrees of freedom rather than doing so in an ad hoc fashion.
- The data-driven wavelet decomposition provides a new kind of modal decomposition that reveals multiscale structure in flows in ways that more traditional decompositions, which are not explicitly hierarchical do not.
- With the above tools in hand, we are poised to apply them to a range of experimental data sets. We are currently working with Mike Schatz and Roman Grigoriev at Georgia Tech, Romit Maulik at Argonne and Markus Hultmark at Princeton on various flow data sets.

4.2 Other disciplines

While our work has all been done in the context of fluid dynamics, the approaches that we have developed are applicable and will be of interest for any problem involving data involving many degrees of freedom and scales.

4.3 Development of human resources

This grant has supported much of the graduate education of one student, Alec Linot, and has contributed to the education of several others, due the highly collaborative nature of the PI's group. Alec and the other students will enter the engineering research workforce with a powerful combination of educational experiences in fluid dynamics, scientific computing, machine learning and applied mathematics that will be very beneficial to the country's intellectual infrastructure.

4.4 Teaching and educational experiences

The principal impact of this grant on teaching and educational experience has arisen as the PI has become increasingly knowledgeable about the field of machine learning. The PI teaches the first-semester mathematical modeling course for graduates students in his department, which primarily focuses on linear algebra, ordinary differential equations, and probability. Because many of the foundations of machine learning lie in linear algebra, the PI has begun using examples and problems motivated by machine learning in the linear algebra portion of the course.

References

[1] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, *Link communities reveal multiscale complexity in networks*, Nature, 466 (2010), pp. 761–764.

[2] F. Argoul, A. Arneodo, G. Grasseau, Y. Gagne, E. J. Hopfinger, and U. Frisch, *Wavelet analysis of turbulence reveals the multifractal nature of the Richardson cascade*, Nature, 338 (1989), pp. 51–53.

[3] M. Asch, M. Bocquet, and M. Nodet, *Data Assimilation: Methods, Algorithms, and Applications*, SIAM, 2016.

[4] Y. Bengio, J. F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet, *Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering*, Advances in Neural Information Processing Systems, (2004).

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Inc., USA, 1995.

[6] E. Bollt, *Attractor modeling and empirical nonlinear model reduction of dissipative dynamical systems*, International Journal of Bifurcation and Chaos, 17 (2007), pp. 1199–1219.

[7] S. Bradshaw and P. N. Howard, *Challenging truth and trust: A global inventory of organized social media manipulation*, tech. rep., University of Oxford, 2018.

[8] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the National Academy of Sciences, 113 (2016), pp. 3932–3937.

[9] M. A. Bucci, O. Semeraro, A. Allauzen, G. Wisniewski, L. Cordier, and L. Mathelin, *Control of chaotic systems by deep reinforcement learning*, Proc. R. Soc. A., 475 (2019).

[10] 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 (2015), pp. 1–5.

[11] M. Budišić, R. Mohr, and I. Mezić, *Applied koopmanism*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 22 (2012), p. 047510.

[12] D. Canaday, A. Griffith, and D. J. Gauthier, *Rapid time series prediction with a hardware-based reservoir computer*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 28 (2018), p. 123119.

[13] F. Cazáis and A. Lhéritier, *Beyond two-sample-tests: Localizing data discrepancies in high-dimensional spaces*, in 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2015, pp. 1–10.

[14] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, *Data-driven discovery of coordinates and governing equations*, Proceedings of the National Academy of Sciences, 116 (2019), pp. 22445–22451.

[15] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, *Neural ordinary differential equations*, arXiv preprint arXiv:1806.07366, (2019).

[16] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder–decoder approaches*, in Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, Oct. 2014, Association for Computational Linguistics, pp. 103–111.

[17] H. Choi and S. Choi, *Robust kernel isomap*, Pattern Recognition, 40 (2007), pp. 853–862.

[18] F. Chollet et al., *Keras*. https://keras.io, 2015.

[19] J. P. Cunningham and Z. Ghahramani, *Linear dimensionality reduction: Survey, insights, and generalizations*, Journal of Machine Learning Research, 16 (2015), pp. 2859–2900.

[20] P. Cvitanović, R. Artuso, R. Mainieri, G. Tanner, and G. Vattay, *Chaos: Classical and Quantum*, Niels Bohr Inst., Copenhagen, 2016.

[21] P. Cvitanović, R. L. Davidchack, and E. Siminos, *On the State Space Geometry of the Kuramoto–Sivashinsky Flow in a Periodic Domain*, SIAM Journal on Applied Dynamical Systems, 9 (2010), pp. 1–33.

[22] I. Daubechies, *Ten lectures on wavelets*, SIAM, 1992.

[23] X. Ding, H. Chaté, P. Cvitanović, E. Siminos, and K. A. Takeuchi, *Estimating the Dimension of an Inertial Manifold from Unstable Periodic Orbits*, Physical Review Letters, 117 (2016), pp. 1–5.

[24] C. R. Doering and J. D. Gibbon, *Applied analysis of the Navier-Stokes equations*, Cambridge University Press, Cambridge, 1 ed., 1995.

[25] C. R. Doering, J. D. Gibbon, D. D. Holm, and B. Nicolaenko, *Low-dimensional behaviour in the complex Ginzburg-Landau equation*, Nonlinearity, 1 (1988), pp. 279–309.

[26] R. A. Edson, J. E. Bunder, T. W. Mattner, and A. J. Roberts, *Lyapunov exponents of the Kuramoto–Sivashinsky PDE*, The ANZIAM Journal, 61 (2019), pp. 270–285.

[27] R. Everson, L. Sirovich, and K. R. Sreenivasan, *Wavelet analysis of the turbulent jet*, Physics Letters A, 145 (1990), pp. 314–322.

[28] M. Farge, *Wavelet transforms and their applications to turbulence*, Annual Review of Fluid Mechanics, 24 (1992), pp. 395–458.

[29] M. Farge, G. Pellegrino, and K. Schneider, *Coherent vortex extraction in 3D turbulent flows using orthogonal wavelets*, Physical Review Letters, 87 (2001), p. 054501.

[30] M. Farge, K. Schneider, G. Pellegrino, A. A. Wray, and R. S. Rogallo, *Coherent vortex extraction in three-dimensional homogeneous turbulence: Comparison between CVS-wavelet and POD-Fourier decompositions*, Physics of Fluids, 15 (2003), pp. 2886–2896.

[31] A. L. Ferguson, A. Z. Panagiotopoulos, I. G. Kevrekidis, and P. G. Debenedetti, *Nonlinear dimensionality reduction in molecular simulation: The diffusion map approach*, Chemical Physics Letters, 509 (2011), pp. 1–11.

[32] D. Floryan and M. D. Graham, *Charts and atlases for nonlinear data-driven models of dynamics on manifolds*, arXiv preprint arXiv:2108.05928, (2021).

[33] D. Floryan and M. D. Graham, *Discovering multiscale and self-similar structure with data-driven wavelets*, PNAS, 118 (2021), p. e2021299118.

[34] 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 - Modélisation Mathématique et Analyse Numérique, 22 (1988), pp. 93–118.

[35] C. Foias, B. Nicolaenko, G. R. Sell, and R. Temam, *Inertial manifold for the Kuramoto-Sivashinsky equation and an estimate of their lowest dimension*, J. Math. Pure Appl., 67 (1988), pp. 197–226.

[36] M. W. Frazier, *An introduction to wavelets through linear algebra*, Springer Science & Business Media, 2006.

[37] K. Fukami, T. Nakamura, and K. Fukagata, *Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data*, Physics of Fluids, 32 (2020), p. 095110.

[38] J. F. Gibson, J. Halcrow, and P. Cvitanović, *Visualizing the geometry of state space in plane Couette flow*, Journal of Fluid Mechanics, 611 (2008), pp. 107–130.

[39] J. Gilles, *Empirical wavelet transform*, IEEE Transactions on Signal Processing, 61 (2013), pp. 3999–4010.

[40] R. Gonzalez-Garcia, R. Rico-Martinez, and I. G. Kevrekidis, *Identification of distributed parameter systems: A neural net based approach*, Computers & Chemical EngineeringComputers & Chemical Engineering, 22 (1998), pp. S965–S968.

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT Press, 2016.

[42] R. A. Gopinath, J. E. Odegard, and C. S. Burrus, *Optimal wavelet representation of signals and the wavelet sampling theorem*, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 41 (1994), pp. 262–277.

[43] M. D. Graham and D. Floryan, *Exact coherent states and the nonlinear dynamics of wall-bounded turbulent flows*, Annu Rev Fluid Mech, to appear (2020).

[44] U. Grasemann and R. Miikkulainen, *Evolving wavelets using a coevolutionary genetic algorithm and lifting*, in Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 969–980.

[45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, 2018.

[46] D. K. Hammond, P. Vandergheynst, and R. Gribonval, *Wavelets on graphs via spectral graph theory*, Applied and Computational Harmonic Analysis, 30 (2011), pp. 129–150.

[47] L. H. O. Hellström, I. Marusic, and A. J. Smits, *Self-similarity of the large-scale motions in turbulent pipe flow*, J. Fluid Mech., 792 (2016), pp. R1–12.

[48] H. Hentschel and I. Procaccia, *The infinite number of generalized dimensions of fractals and strange attractors*, Physica D: Nonlinear Phenomena, 8 (1983), pp. 435–444.

[49] G. Hinton and S. Roweis, *Stochastic neighbor embedding*, Advances in neural information processing systems, 15 (2003), pp. 833–840.

[50] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science, 313 (2006), pp. 504–507.

[51] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, Neural Computation, 9 (1997), pp. 1735–1780.

[52] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge University Press, Cambridge, 2 ed., 2012.

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

[54] Y. Hwang, *Statistical structure of self-sustaining attached eddies in turbulent channel flow*, J. Fluid Mech., 767 (2015), pp. 254–289.

[55] A. C. Ian Goodfellow, Yoshua Bengio, *The Deep Learning Book*, vol. 521, 2017.

[56] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, *Discovering Physical Concepts with Neural Networks*, Physical Review Letters, 124 (2020), p. 10508.

[57] M. S. Jolly, R. Rosa, and R. Temam, *Evaluating the dimension of an inertial manifold for the Kuramoto-Sivashinsky equation*, Advances in Differential Equations, 5 (2000), pp. 31–66.

[58] A.-K. Kassam and L. N. Trefethen, *Fourth-order time-stepping for stiff PDEs*, SIAM Journal on Scientific Computing, 26 (2005), pp. 1214–1233.

[59] P. G. Kevrekidis, J. Cuevas-Maraver, Y. Drossinos, Z. Rapti, and G. A. Kevrekidis, *Reaction-diffusion spatial modeling of covid-19: Greece and andalusia as case examples*, Phys. Rev. E, 104 (2021), p. 024412.

[60] L. Kleiser and U. Schumann, *Treatment of Incompressibility and Boundary Conditions in 3-D Numerical Spectral Simulations of Plane Channel Flows*, Vieweg+Teubner Verlag, Wiesbaden, 1980, pp. 165–173.

[61] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.

[62] W. K.-M. Lau and D. E. Waliser, *Intraseasonal variability in the atmosphere-ocean climate system*, Springer Science & Business Media, 2011.

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

[64] P. F. J. Lermusiaux, J. Schröter, S. Danilov, M. Iskandarani, N. Pinardi, and J. J. Westerink, *Multiscale modeling of coastal, shelf, and global ocean dynamics*, 2013.

[65] J. Li and M. Zhang, *Reinforcement-learning-based control of confined cylinder wakes with stability analyses*, Journal of Fluid Mechanics, 932 (2022), p. A44.

[66] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, *A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence*, Journal of Turbulence, (2008), p. N31.

[67] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous control with deep reinforcement learning*, 2016.

[68] 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*, arXiv preprint arXiv:2203.15706, (2022).

[69] A. J. Linot and M. D. Graham, *Deep learning to discover and predict dynamics on an inertial manifold*, Phys. Rev. E, 101 (2020), p. 062209.

[70] A. J. Linot and M. D. Graham, *Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations*, arXiv preprint arXiv:2109.00060 (Chaos, to appear), (2021).

[71] A. J. Linot and M. D. Graham, . https://github.com/alinot5/KSE_ROM_NODE, 2022.

[72] E. N. Lorenz, *Deterministic Nonperiodic Flow*, Journal of the Atmospheric Sciences, 20 (1963), pp. 130–141.

[73] B. Lusch, J. N. Kutz, and S. L. Brunton, *Deep learning for universal linear embeddings of nonlinear dynamics*, Nature Communications, (2018).

[74] S. Mallat, *A wavelet tour of signal processing*, Elsevier, 1999.

[75] I. Marusic and J. P. Monty, *Attached Eddy Model of Wall Turbulence*, Annu Rev Fluid Mech, 51 (2019), pp. 49–74.

[76] R. Maulik, A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, and D. Livescu, *Time-series learning of latent-space dynamics for reduced-order model closure*, Physica D: Nonlinear Phenomena, 405 (2020), p. 132368.

[77] M. A. Mendez, M. Balabane, and J.-M. Buchlin, *Multi-scale proper orthogonal decomposition of complex fluid flows*, Journal of Fluid Mechanics, 870 (2019), pp. 988–1036.

[78] C. Meneveau, *Analysis of turbulence in the orthonormal wavelet representation*, Journal of Fluid Mechanics, 232 (1991), pp. 469–520.

[79] Y. Meyer, *Wavelets and Operators: Volume 1*, vol. 37, Cambridge University Press, 1992.

[80] M. Milano and P. Koumoutsakos, *Neural network modeling for near wall turbulent flow*, Journal of Computational Physics, 182 (2002), pp. 1–26.

[81] N. J. Nair and A. Goza, *Leveraging reduced-order models for state estimation using deep learning*, Journal of Fluid Mechanics, 897 (2020), pp. 1–13.

[82] N. Okamoto, K. Yoshimatsu, K. Schneider, M. Farge, and Y. Kaneda, *Coherent vortices in high resolution direct numerical simulation of homogeneous isotropic turbulence: A wavelet viewpoint*, Physics of Fluids, 19 (2007), p. 115109.

[83] A. Y. Okulov, *Structured light entities, chaos and nonlocal maps*, Chaos, Solitons and Fractals, 133 (2020), p. 109638.

[84] N. Omata and S. Shirayama, *A novel method of low-dimensional representation for temporal behavior of flow fields using deep autoencoder*, AIP Advances, 9 (2019), p. 015006.

[85] B. Ophir, M. Lustig, and M. Elad, *Multi-scale dictionary learning using wavelets*, IEEE Journal of Selected Topics in Signal Processing, 5 (2011), pp. 1014–1024.

[86] J. Page, M. P. Brenner, and R. R. Kerswell, *Revealing the state space of turbulence using machine learning*, Phys. Rev. Fluids, 6 (2021), p. 034402.

[87] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *Pytorch: An imperative style, high-performance deep learning library*, in Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[88] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, *Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach*, Physical Review Letters, 120 (2018), p. 24102.

[89] E. Perlman, R. Burns, Y. Li, and C. Meneveau, *Data exploration of turbulence simulations using a database cluster*, in Proceedings of the 2007 ACM/IEEE conference on Supercomputing, 2007, pp. 1–11.

[90] R. Peyret, *Spectral methods for incompressible viscous flow*, Springer, 2011.

[91] G. D. Portwood, P. P. Mitra, M. D. Ribeiro, T. M. Nguyen, B. T. Nadiga, J. A. Saenz, M. Chertkov, A. Garg, A. Anandkumar, A. Dengel, R. Baraniuk, and D. P. Schmidt, *Turbulence forecasting via neural ODE*, arXiv preprint arXiv:1911.05180, (2019).

[92] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems*, arXiv preprint arXiv:1801.01236, (2018), pp. 1–19.

[93] D. Recoskie and R. Mann, *Gradient-based filter design for the dual-tree wavelet transform*, arXiv preprint arXiv:1806.01793, (2018).

[94] ——, *Learning filters for the 2D wavelet transform*, in 2018 15th Conference on Computer and Robot Vision (CRV), IEEE, 2018, pp. 198–205.

[95] ——, *Learning sparse wavelet representations*, arXiv preprint arXiv:1802.02961, (2018).

[96] C. J. G. Rojas, A. Dengel, and M. D. Ribeiro, *Reduced-order model for fluid flows via neural ordinary differential equations*, arXiv preprint arXiv:2102.02248, (2021).

[97] S. T. Roweis and L. K. Saul, *Nonlinear dimensionality reduction by locally linear embedding*, Science, 290 (2000), pp. 2323–2326.

[98] J. Ruppert-Felsot, M. Farge, and P. Petitjeans, *Wavelet tools to study intermittency: application to vortex bursting*, Journal of Fluid Mechanics, 636 (2009), pp. 427–453.

[99] P. Sagaut and C. Cambon, *Homogeneous Turbulence Dynamics*, Springer International Publishing, Cham, 2018.

[100] T. Sauer, J. A. Yorke, and M. Casdagli, *Embedology*, Journal of Statistical Physics, 65 (1991), pp. 579–616.

[101] A. Søgaard, *Learning optimal wavelet bases using a neural network approach*, arXiv preprint arXiv:1706.03041, (2017).

[102] D. Sondak and P. Protopapas, *Learning a reduced basis of dynamical systems using an autoencoder*, Phys. Rev. E, 104 (2021), p. 034202.

[103] G. Strang, *Wavelets and dilation equations - A brief introduction*, Siam Rev, 31 (1989), pp. 614–627.

[104] K. Taira, S. L. Brunton, S. T. M. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, *Modal analysis of fluid flows: An overview*, AIAA Journal, 55 (2017), pp. 4013–4041.

[105] F. Takens, *Detecting strange attractors in turbulence*, in Dynamical Systems and Turbulence, Warwick 1980, D. Rand and L.-S. Young, eds., Berlin, Heidelberg, 1981, Springer Berlin Heidelberg, pp. 366–381.

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

[107] ——, *Inertial manifolds*, The Mathematical Intelligencer, 12 (1990), pp. 68–74.

[108] R. Temam and X. Wang, *Estimates on the lowest dimension of inertial manifolds for the Kuramoto-Sivasbinsky equation in the general case*, Differential and Integral Equations, 7 (1994), pp. 1095–1108.

[109] J. B. Tenenbaum, V. de Silva, and J. C. Langford, *A global geometric framework for nonlinear dimensionality reduction*, Science, 290 (2000), pp. 2319–2323.

[110] A. H. Tewfik, D. Sinha, and P. Jorgensen, *On the optimal choice of a wavelet for signal representation*, IEEE Transactions on Information Theory, 38 (1992), pp. 747–765.

[111] L. J. P. Van Der Maaten, E. O. Postma, and H. J. Van Den Herik, *Dimensionality Reduction: A Comparative Review*, Journal of Machine Learning Research, 10 (2009), pp. 1–41.

[112] P. Vlachas, G. Arampatzis, C. Uhler, and P. Koumoutsakos, *Learning the effective dynamics of complex multiscale systems*, arXiv preprint arXiv:2006.13431, (2021).

[113] P. Vlachas, J. Pathak, B. Hunt, T. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, *Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics*, Neural Networks, 126 (2020), pp. 191–217.

[114] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, *Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 474 (2018-05), p. 20170844.

[115] H. Whitney, *Differentiable Manifolds*, Annals of Mathematics, 37 (1936), pp. 645–680.

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

[117] M. Yamada and K. Ohkitani, *An identification of energy cascade in turbulence by orthonormal wavelet analysis*, Progress of Theoretical Physics, 86 (1991), pp. 799–815.

[118] H.-l. Yang and G. Radons, *Geometry of inertial manifolds probed via a lyapunov projection method*, Phys. Rev. Lett., 108 (2012), p. 154101.

[119] 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 (2009), pp. 1–4.

[120] P. K. Yeung, D. A. Donzis, and K. R. Sreenivasan, *Dissipation, enstrophy and pressure statistics in turbulence simulations at high Reynolds numbers*, Journal of Fluid Mechanics, 700 (2012), pp. 5–15.

[121] S. Zelik, *Inertial manifolds and finite-dimensional reduction for dissipative PDEs*, Proceedings of the Royal Society of Edinburgh Section A: Mathematics, 144 (2013), pp. 1245–1327.

[122] K. Zeng and M. D. Graham, *Symmetry reduction for deep reinforcement learning active control of chaotic spatiotemporal dynamics*, Phys. Rev. E, 104 (2021), p. 014210.

[123] 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, (2022).

[124] Y. Zhuang and J. S. Baras, *Optimal wavelet basis selection for signal representation*, in Wavelet Applications, vol. 2242, International Society for Optics and Photonics, 1994, pp. 200–211.