Characterizing multiscale structures and developing hybrid low-order models of chaotic flows

by

Alex Guo

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Chemical and Biological Engineering)

at the

UNIVERSITY OF WISCONSIN - MADISON

2026

Date of final oral examination: December 18, 2025

This dissertation is approved by the following members of the Final Oral Committee:

Michael D. Graham, Professor, Chemical and Biological Engineering

Victor M. Zavala, Professor, Chemical and Biological Engineering

Reid C. Van Lehn, Professor, Chemical and Biological Engineering

Saverio E. Spagnolie, Professor, Mathematics

Dedication

To my family:

Mom and dad: I love you both, and I cannot even imagine how I can ever pay you back for the amount of time and energy you have put into raising me. I would not be where I am today without you both. Thank you mom for all the restless days and nights you spent caring for me and my brothers as we grew up, it is truly an immense task and your unconditional love for us is genuinely admirable. Thank you dad for all the hardships you took on to immigrate to the United States from China all those years ago—I can not begin to express my gratitude for the life and opportunities you have given me. Albert and Alan: we may not have been the closest brothers, but I am still glad to have you in my life.

To my aunt in Taipei: Thank you for all the summers you graciously allowed me to stay at your apartment. You were always generous and caring, and I truly appreciate you being there for me to help me get settled when I interned at Taicang. To my step-grandpa in Taipei: Thank you for all those summers you spent mentoring me at your tutoring center. You provided me guidance not just in terms of excelling in academics, but also in navigating challenges in my personal life. You instilled in me a passion for learning and equipped me with tools to learn effectively. To all my relatives in China, Taiwan, and San Francisco: Thank you all for your warm hospitality; your welcoming reception were what I always looked forward to the most whenever I traveled.

To my friends:

To Jonathan: You are my best friend, and words can't describe how thankful I am to have you in my life; I experienced many personal dilemmas in my life, and you were instrumental in helping me overcome them. To Bhumesh and Brandon: I've known you as roommates since the start of my PhD, and I will miss you both dearly. I will always think back fondly on our movie nights, parties, and staying up late just talking. I truly could not have imagined having better roommates. To Coach Heather, Andy, Leila, Sam, Charlie, and others at MRC: Thank you for all these years together rowing on Lake Mendota. I will greatly miss all our practices, regattas, and social events—you are undoubtedly the nicest group of people I've ever met. To everyone at HMC and teams: thank you for showing me the ropes (literally) with climbing, you were all the chillest people I've ever met and I will always think back to practicing at Summit and crushing it at comps together.

To Carlos, Xiaopo, Eric, RJ, Alec, Kevin, Ricardo, Charlie, Daniel, and Jake: Thank you for all these years of mentorship and friendship, I couldn't have made it this far in my PhD without you. Jake, Alec, and Carlos, I will miss our extensive discussions about research. Charlie: Thank you for the opportunity and your mentorship efforts at LANL. To David: I will miss our disc golf and ultimate frisbee sessions. Carlos, I will miss our talks about climbing—you were the reason I got into outdoor climbing and I've been hooked since. Xiaopo, I'll greatly miss our talks about souls games.

To all my climbing friends: I may have only gotten to know all of you only this year, but I am sincere when I say meeting you all has been transformative for me. Casey, I am so grateful you were so friendly and approached me that day in Summit and introduced me to everyone else. Sam, you are a genuinely kind and strong dude—I'll always trust you with my life to spot/belay me. To Jun, Li, Mia, Spencer, Leo, Iris, David: whether it's trying hard when climbing, or huddling at a campsite eating hotpot, I am truly thankful to have gotten to meet and spend time with all of you. To my girlfriend Lillian: you are the most amazing person I've ever met, and I am honored to have you in my life—I love you.

To my advisor and committee:

To my PhD advisor Mike: I cannot begin to express my gratitude for all your guidance and patience with me over these years. You have helped nurture both my confidence and skills as an academic researcher. Along the road in my PhD, I've experienced a fair amount of setbacks; however, your professional advice and friendly demeanor has always helped set me back on track, and I am indebted to you. To my committee members: Thank you for serving on my committee for my prelim talk, 4th year talk, and my thesis defense. Your counsel and feedback have been instrumental in helping me along my way to a PhD.

Contents

Abstract **viii**

List of Figures **xx**

1 General introduction **1**

1.1 Wall-turbulence 3

1.1.1 Structures in turbulence 5

1.1.2 Attached eddy hypothesis 6

1.1.3 Attached eddy model 9

1.1.4 The -1 power law 11

1.1.5 Dynamical nature of structures in wall-turbulence 14

1.2 Modal-based characterization of structures in data 16

1.2.1 Proper orthogonal decomposition 17

1.2.2 Other modal decompositions 18

1.2.3 Wavelets 19

1.3 Dimension Reduction 22

1.3.1 Manifolds 23

1.3.2 Time-delay embeddings 24

1.3.3 Principal component analysis 25

1.3.4 Autoencoders 25

1.3.5 Implicit rank-minimizing autoencoders (IRMAE) 27

1.4 Time evolution 27
1.4.1 Common methods 28
1.4.2 Neural ordinary differential equations 29
1.4.3 Learning PDE solutions 30
1.5 Outline of this work 33

2 Characterizing 1D structures in turbulent pipe flow **35**
2.1 Introduction 36
2.2 Experimental setup 39
2.3 Data-driven wavelet decomposition 42
2.3.1 Overview and connections to POD and wavelet analysis 43
2.3.2 Pre-processing and modifications for long time series 46
2.4 Results 50
2.4.1 Shape of localized structures at $y^+ = 350$ 51
2.4.2 Self-similarity of localized structures at $y^+ = 350$ 53
2.4.3 Self-similarity of localized structures in wall-normal direction 55
2.4.4 Self-similarity of wavelet projections 58
2.4.5 Energy and spectral scaling laws 61
2.5 Comparison of ranges of self-similarity 65
2.6 Additional details about use and validation of DDWD 67
2.6.1 Details of obtaining an 18-stage DDWD basis 69
2.6.2 DDWD validation using synthetic, self-similar signals 72
2.6.3 Defining the length scale of a wavelet 75
2.6.4 Extending DDWD to 2D and 3D data 75
2.6.5 Relation between wavelet projection and filtering 76
2.7 Conclusions 79

3 Hybrid reduced-order modeling **82**

3.1 Introduction . . . 82

3.2 Methods . . . 87

3.2.1 Data-driven manifold dynamics (DManD) . . . 88

3.2.2 Manifold Galerkin . . . 91

3.2.3 Physics-informed data-driven manifold dynamics (PI-DManD) . . . 93

3.2.4 Architecture and training details . . . 96

3.3 Results . . . 98

3.3.1 Abundant data, exact FOM . . . 99

3.3.2 Scarce data, exact FOM . . . 101

3.3.3 Approximate FOM . . . 104

3.4 Additional information . . . 106

3.4.1 Training and testing speed for methods applied to KSE . . . 106

3.4.2 Long time statistics results for KSE . . . 107

3.4.3 Complex Ginzburg–Landau Equation . . . 109

3.4.4 Integrating Manifold Galerkin (MG) models . . . 119

3.5 Noise robustness . . . 123

3.6 Conclusion . . . 127

4 Reconstructing fields from sparse sensors **130**

4.1 Background on attention mechanism . . . 131

4.2 Introduction . . . 132

4.3 Methodology . . . 133

4.4 Results . . . 135

4.4.1 Non-uniform sampling in time . . . 135

4.4.2 Variable sensor position . . . 135

4.4.3 Chaotic systems . . . 137

4.5 Additional information . . . 138

4.5.1 Time2vec encoding . . . 138
4.5.2 Architecture details . . . 138
4.6 Attention mechanisms for time evolution . . . 140
4.7 Conclusion . . . 141

5 Characterizing 3D structures in turbulent channel flow 143
5.1 Introduction . . . 143
5.2 Methodology . . . 148
5.3 Results . . . 150
5.3.1 DDWD wavelets in x . . . 150
5.3.2 Streaks in viscous layer . . . 151
5.3.3 POD modes from DWT-POD . . . 152
5.3.4 mPOD modes . . . 154
5.3.5 Projections and energy . . . 155
5.4 Conclusion . . . 157

6 Conclusions 159
6.1 General summary . . . 159
6.2 Future work . . . 160
6.2.1 Attached eddy basis . . . 161
6.2.2 Estimating manifold dimension of homogeneous turbulence . . . 164
6.2.3 Extending our reduced-order models . . . 169

References 189

Abstract

Efficient and accurate predictive models are highly sought after in applications such as weather forecasting, chemical processes, and turbulent flows. Wall-bounded turbulence—which is the focus of this work—is a notoriously difficult system to predict due to its chaotic nature and presence of multiscale structures (i.e., eddies or vortices) that must all be resolved in simulations. This thesis contains two main goals. The first is to gain physical insight into wall-bounded turbulence by characterizing its structures. The second goal is to develop and benchmark low-order models of chaotic dynamical systems such as wall-bounded turbulence, which would be instrumental in iterative design and control of these systems due to the many model evaluations required.

We start in Chapter 1 with a background on the theory of physical structures embedded in turbulent flow—their appearance and whether they are self-similar across scales—in addition to background on low-order modeling of dynamics on manifolds. In Chapter 2, we present data-driven wavelets as an ideal modal basis for representing and extracting spatially-localized, multiscale structures from data; we use this basis to discover self-similarity of structures in one-dimensional signals of turbulent pipe flow. In Chapter 3, we develop a physics-informed, data-driven reduced-order model (ROM) for forecasting spatiotemporally chaotic dynamical systems. In Chapter 4, we discuss how to combine time-delay embeddings and attention-based neural networks to reconstruct fields from partial observations of the state, which addresses a real-world challenge where sensors (e.g., those measuring atmospheric data) do not always have full coverage. In Chapter 5, we return to using wavelets to

characterize structures, but this time in three-dimensional turbulent channel flow. Finally, we conclude in Chapter 6 with a general summary of the chapters and an overview of future work towards developing a high-fidelity, low-order model of wall-bounded turbulence.

List of Figures

1.1 Log-law in wall-turbulence. 7

1.2 Sketch of the pre-multiplied PSD of a fluctuating streamwise velocity signal measured close to the wall (see text for description of the scalings and 3 eddy types). Adapted from [148]. 12

1.3 (a) Example wavelets, with Meyer (top) and db2 (bottom). (b) Scale-space decomposition by Fourier modes (top) with corresponding Fourier modes (bottom). (c) Scale-space decomposition by wavelet modes (top) with corresponding wavelet modes (bottom). 19

1.4 Subspaces formed by wavelets periodic in $\mathbb{R}^N$. The full space $\mathbb{R}^N$ is composed of the highlighted subspaces. 21

2.1 Wall-normal positions of the measurement probe. 40

2.2 Premultiplied PSD of u, measured at $y^+ = 350$, without and with filtering at 25 kHz. 41

2.3 Heat map of the time correlation function at all wall-normal positions. $N_\tau = \tau F_s$ is the number of points needed to capture a time lag of τ [s], and $\tau U_b/R$ is the dimensionless eddy turnover time. 42

2.4 Subspaces formed by wavelets periodic in $\mathbb{R}^N$. The full space $\mathbb{R}^N$ is composed of the highlighted subspaces. 44

2.5 Haar wavelets on $\mathbb{R}^8$, following the style of Figure 2.4. The Haar basis for each subspace is shown. 44

2.6 Comparing the 18-stage DDWD basis at $y^+ = 350$ to three known wavelet bases. (a) Superimposed wavelets (only DDWD and Meyer shown for clarity). (b) Stage-8 wavelets. (c) Similarity between DDWD basis and known wavelet bases. 52

2.7 (a) 18-stage DDWD basis at $y^+ = 350$; solid red curves are ψ_{-l} (or ξ_{-18} for stage 18a), and dashed blue curves are $S\psi_{-(l-1)}$ (or $S\xi_{-17}$ for stage 18a), where S essentially dilates its input wavelet by a factor of 2. (b) Self-similarity across adjacent stages (i.e., inner product between the normal and dilated wavelets), where the gray region covers the range of values encountered across 10 DDWD initializations. (c) Self-similarity across stages 1 through 15, where the dashed triangle indicates the region of high self-similarity ($\alpha_{l_a,l_b} \gtrsim 0.95$). 54

2.8 (a) DDWD bases at $y^+ = 350$, $y/R = 0.10$, and $y/R = 0.59$. (b) Similarity across adjacent stages for each basis. (c) Similarity across bases, where b_1 is for the signal closest to the wall and b_2 is for the signal either in region II or III (regions defined in Figure 2.1). 56

2.9 Self-similarity between the reference wavelet at $l_{\text{ref}} = 8$ and $y^+_{\text{ref}} = 350$ and all other wavelets. Dashed horizontal lines separate the wall-normal regions defined in Figure 2.1. Solid white lines are isolines of the streamwise length scale λ_x of the wavelets. 57

2.10 PSDs of u, its stage-l DDWD wavelet projections $P_l u$, and the DDWD wavelets ψ_{-l} at $y^+ = 350$. Note that the PSDs of the wavelets are manually shifted to be at the same height. 59

2.11 (a) Scaled and normalized PSDs of the DDWD wavelet projections in Figure 2.10 for $l \in [3, 13]$. (b) Inner product between the curves in (a), where the dashed triangle indicates the region of high self-similarity ($\gamma_{l_a,l_b} \gtrsim 0.95$). Highly self-similar projections of scales $l \in [6, 11]$ in (a) are colored red based on the triangular region in (b). 59

2.12 Same as Figure 2.9, except with the inner product γ, which quantifies the self-similarity of the PSDs of wavelet projections using the (a) DDWD, (b) Meyer, and (c) db2 basis. Solid white lines are isolines of the streamwise length scale λ_x of the wavelets. 60

2.13 (a) Wavelet energy distribution (WED) of the signal measured at $y^+ = 350$ calculated using the DDWD basis and 3 known wavelet bases (the green and purple curves overlap). (b) Two versions of the DDWD WED from (a), one of which is premultiplied. Note, error bars are standard errors. 63

2.14 (a) Wavelet energy distribution (WED), calculated using the DDWD basis, at all wall-normal positions. This plot is analogous to a premultiplied PSD spectrogram. 64

2.15 Filter v_1 (equivalently, wavelet ψ_{-1}) from the Meyer and stretched Meyer basis in real space (a) and Fourier space (b). 71

2.16 Robustness of DDWD to the strength of the variance penalty parameter λ^2 for synthetic signals composed of (a) Meyer, (b) fk6, and (c) db2 wavelets (b_1 is the corresponding wavelet basis and b_2 is the DDWD basis). Refer to Figure 2.6b for the shapes of these wavelets. 73

2.17 Results of DDWD applied to a synthetic dataset generated from Meyer wavelets using the WED of the Superpipe signal measured at the centerline. Label "normal" indicates that a zero mean was enforced for the DDWD wavelets. Plotted are (a) wavelets at relevant stages and (b) accuracy at each stage with b_1 as the true basis and b_2 as the computed DDWD basis. 74

2.18 Results for Superpipe signal $u(x)$ at $y^+ = 350$. Plotted are the PSD (e.g., for u, plotted is $|\hat{u}|^2$) of various quantities calculated using a stage-4 (a) DDWD and (b) db2 wavelet. See text for the definitions of the quantities. 78

2.19 Results for a synthetic db2 signal constructed with E_l taken at $y^+ = 350$. Plotted are the PSD (e.g., for u, plotted is $|\hat{u}|^2$) of various quantities calculated using a stage-4 (a) db2 (what is found by DDWD) and (b) near-top-hat wavelet (taken from Figure 2.18). See text for the definitions of the quantities. 80

3.1 Data-driven manifold dynamics (DManD) is a data-driven ROM that reduces dimension of the state u with an autoencoder (a) and evolves trajectories with a vector field on the latent space h learned by a neural ODE (b). (c) Physics-informed DManD (PI-DManD) is a hybrid ROM that uses Manifold Galerkin to project the vector field on the full order model (FOM) f_{FOM} onto that of the latent space, learns a neural network approximation of the latent space vector field as g_{mn}, and either uses a corrective neural ODE g_c to fix numerical and FOM approximation errors (PI-DManD-c) or uses g_{mn} as a Bayesian prior for training a DManD model (PI-DManD-B). 90

3.2 KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. (a) Ensemble-averaged short-time tracking error for all methods. (b) Representative trajectories of ground truth and model predictions (contours at $u = \pm 1$). (c) Difference plots between ground truth and model predictions (contours at difference values of ± 1). Representative trajectories and difference plots for other models look similar to those of PI-DManD-c. (d) NODE training convergence for purely data-driven and hybrid methods. 100

3.3 KSE, $L = 22$ (for both data and FOM). Short-time tracking error at one Lyapunov time for scarce data in terms of (a) less data with $N \leq 400,000$ and $\tau = 0.25$ (i.e. shorter time series), (b) widely spaced data with $\tau > 0.25$ and $N = 400,000$, and (c) increasing τ and decreasing N proportionally. Here and elsewhere, error bars show standard error, and all subplots follow the legend in (a). Note the different vertical axis limits for each subplot. Missing PI-DManD-c bars, denoted with asterisk, in (b) and (c) have errors of about 1 (i.e. much larger than the case shown) and are omitted to improve clarity in the vertical axis. 103

3.4 KSE, $N = 400,000$, $\tau = 0.25$. Effects of using an erroneous parameter ($L = 20$) in the FOM to model data generated with $L = 22$. (a) Ground truth trajectory (left) and example trajectory with erroneous parameter (right). (b) Ensemble-averaged short-time tracking error. (c) NODE training convergence for purely data-driven and hybrid methods. 105

3.5 KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. (a) Comparison of joint pdfs. (b) Earth movers distance (EMD) quantifying the difference between pdfs of the ground truth and model predictions, where the dashed red line is the baseline EMD calculated with the pdf of another ground truth trajectory. Joint pdfs for models not shown look similar to joint pdfs of models with similar EMD values. 108

3.6 KSE, $N = 400,000$, $\tau = 0.25$. Effects of using an erroneous parameter ($L = 20$) in the FOM when making model predictions on the data generated with $L = 22$. (a) Comparison of joint pdfs. (b) Earth movers distance (EMD) quantifying the difference between pdfs of the ground truth and model predictions, where the dashed red line is the baseline EMD calculated with the pdf of another ground truth trajectory. Joint pdfs for models not shown look similar to joint pdfs of models with similar EMD values. 109

3.7 Singular value spectra of the covariance of the latent space z for IRMAE-WD applied to CGLE. Red dashed line located at 40th singular value index. . . . 111

3.8 CGLE, $\beta = -2$ (for both data and FOM), $N = 400,000$, $\tau = 0.05$. (a) Ensemble-averaged short-time tracking error for all methods. (b) Absolute value of representative trajectories of ground truth and model predictions (contours at $|A| = 0.4, 0.8$). (c) Difference plots between ground truth and model predictions (contours at difference values of 0.4, 0.8). Representative trajectories and difference plots for other models look similar to those of PI-DManD-c. (d) NODE training convergence for purely data-driven and hybrid methods. 112

3.9 CGLE, $\beta = -2$ (for both data and FOM), $N = 400,000$, $\tau = 0.05$. (a) Comparison of joint pdfs. (b) Earth movers distance (EMD) quantifying the difference between pdfs of the ground truth and model predictions, where the dashed red line is the baseline EMD calculated with the pdf of another ground truth trajectory. Joint pdfs for models not shown look similar to joint pdfs of models with similar EMD values. 114

3.10 CGLE, $\beta = -2$ (for both data and FOM). Short-time tracking error at one Lyapunov time for scarce data in terms of (a) less data with $N \leq 400,000$ and $\tau = 0.05$ (i.e. shorter time series), (b) widely spaced data with $\tau > 0.05$ and $N = 400,000$, and (c) increasing τ and decreasing N proportionally. Error bars are standard errors, and all subplots follow the legend in (a). Note the different vertical axis limits for each subplot. Missing PI-DManD-c bars, denoted with asterisk, in (b) and (c) have errors of about 1 and are omitted to improve clarity in vertical axis. 115

3.11 CGLE, $N = 400,000$, $\tau = 0.05$. Effects of using an erroneous parameter ($\beta = -1.8$ or $\beta = -1$) in the FOM to model data generated with $\beta = -2$. (a) Ground truth trajectory (top-left) and example trajectories for the two erroneous β values (top-right and bottom); note the different colorbar axis limits. Ensemble-averaged short-time tracking error when using $\beta = -1.8$ (b) or $\beta = -1$ (c) in the FOM. 117

3.12 CGLE, N=400,000, $\tau = 0.05$. Effects of using an erroneous parameter ($\beta = -1.8$) in the FOM to model the data generated with $\beta = -2$. (a) Comparison of joint pdfs for $\beta = -1.8$. (b) Earth movers distance (EMD) quantifying the difference between pdfs of the ground truth and model predictions for $\beta = -1.8$. Dashed red line is the baseline EMD calculated with the pdf of another ground truth trajectory. Joint pdfs for models not shown look similar to joint pdfs of models with similar EMD values. 119

3.13 KSE and CGLE, correct parameters for both data and FOM, $N = 400,000$, $\tau = 0.25$ for KSE and $\tau = 0.05$ for CGLE. Comparing short-time tracking of MG model to "autoencoder drift", which is calculated by encoding and decoding an ensemble of initial conditions without time integration. 120

3.14 Ensemble short-time tracking (STT) error for 5 MG models, each from a different autoencoder model. Autoencoder models are trained with different weight initializations and have nearly the same test reconstruction error ϵ_{AE} of 0.22%. "Enc" means the Jacobian of the encoder is used, and "dec" means the pseudoinverse of the Jacobian of the decoder is used. RK4 with step size 0.25 time units is used as the integrator. 121

3.15 KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. Noisy vs. clean signal in space (a)-(c) and in time (d)-(f), where the strength of the noise σ increases from left to right. 124

3.16 KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. Autoencoder reconstruction of the noisy signal vs. the underlying clean signal in space (a)-(c) and in time (d)-(f), where the strength of the noise σ increases from left to right. 125

3.17 KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. Ensemble-averaged short-time tracking error for DManD and PI-DManD-B with varying levels of noise σ. 126

4.1 Architecture details. A time delay embedding of sensor values is concatenated with the positional and temporal encodings and encoded to a compressed latent space. The latent values are passed to the decoder with a query position to reconstruct the field at the current time. 134

4.2 Field reconstruction of flow past a cylinder from sensor measurements that are non-uniform in time. (a) True vorticity field ω, with location of sensor marked with a square. (b) Reconstruction error for methods shown in panels (d)-(f). (c) Reconstruction error across each pair of delay spacings τ_{0j} for method shown in panel (f). Reconstruction fields $\hat{\omega}$ for models trained with (d) no time delay embedding, (e) embedding dimension $m = 3$ with temporal encoding of τ_{0j} via concatenation, and (f) $m = 3$ with temporal encoding of τ_{0j} via t2v are shown in the top panels of each subfigure. Difference in true and reconstructed fields are shown in the respective bottom panels of (d)-(f). In each field image, bounds on the y-axis are $[-9, 9]$ and the x-axis are $[0, 32]$ in dimensionless units. 136

4.3 (a) Time and space averaged reconstruction test error for using a given sensor position as model input; dashed blue circles indicate training positions. (b) Error for each frame in a vortex shedding period for three sensor locations; vertical line corresponds to the snapshot time below for panels (c)-(e), which visualize the reconstruction at the selected locations. Panel (c) is a training position $x \approx 14, y \approx 0$, whereas (d) $y \approx -0.5$ and (e) $y \approx -1$ are test positions. 137

4.4 Model performance of a NODE compared to an attention-based NODE (a-NODE) in terms of ensemble-averaged short-time tracking (left) and long-time statistics quantified through a pdf of the partially-observed state (right). . . 141

5.1 Sketch describing three POD-based decomposition techniques. Shown are (a) DFT-POD, (b) spectral POD (SPOD), and (c) mPOD (multiscale POD). The left panel shows the relevant directions, the middle panel shows the transformed directions after DFT/MRA is performed in the homogeneous/POD direction, and the right panel shows whether the POD modes across different inverse-scale directions are orthogonal under a 1D inner product. 145

5.2 Partitioning of 2D wavelet subspaces, with similar color scheme and notation as Figure 2.4. 148

5.3 Sketch describing (a) DWT-POD and (b) mPOD (multiscale POD). The left panel shows the relevant directions, the middle panel shows the transformed directions after DWT/MRA is performed in the homogeneous/POD direction, and the right panel shows whether the POD modes across different scales are orthogonal under a 1D inner product. 150

5.4 DDWD basis in x found at all wall-normal positions and three specified wall-normal positions. (a) Wavelets for all four cases. (b) Self-similarity across adjacent stages (i.e., inner product between the stage l_x wavelets and the dilated, previous-stage wavelets) for all four cases. 151

5.5 (a) DDWD basis in z found at $y^+ = 7$, with wavelets in red and the dilated, previous-stage wavelets in blue. (b) Adjacent self-similarity for $y^+ = 7$ wavelets and two other wall-normal positions. 152

5.6 (a) Streamwise fluctuating velocity field $u(x, z)$ at $y^+ = 7$. Details scale 3 and below (not shown) are removed from the field to yield the projection onto the stage-4 approximation (b) and projection onto the stage-4 details (c). (d) Zoom-in of (c). 153

5.7 (a) POD modes found using DWT-POD (note, l refers to scale l_x in this figure). (b) Inner product of the $n = 1$ POD mode across scales. 153

5.8 (a) DDWD modes in y in red and dilated, previous-stage wavelets in blue. (b) Adjacent self-similarity. 154

5.9 (a) mPOD modes normalized by the length scale (note stage/scale l should be l_x in this figure). (b) Same as (a) but for POD modes. Center of mass of modes as a function of scale for (c) mPOD and (d) POD modes. 155

5.10 Stage-6 projection of fluctuating streamwise velocity field onto the $n = 1$ mPOD (a) and POD (b) mode. 156

5.11 Energy of modes in fluctuating streamwise velocity field u as a function of modal index n. POD is applied to 1D snapshots of u instead of wavelet projections, mPOD is shown for decreasing scale and increasing n, and mPOD (ordered) is shown for only increasing n. 157

6.1 Attached eddy found with Q-criterion of representative velocity structure of an attached eddy basis (AEB), shown in isometric (a) and side (b) view. (c) Using the AEB as a model for the streamwise Reynolds stress. 163

6.2 Architecture of a convolutional autoencoder. "Conv" denotes convolution layer with kernel size in parentheses and downsample factor for max-pool is in parentheses. 166

6.3 2D HIT without filter (a) and with filter (b). (c) Energy spectrum. (d) Training dynamics of singular value spectra for the filtered vorticity field (IRMAE is learning the Fourier transform here). 167

6.4 Results for unfiltered vorticity field. (a) Relative MSE vs. manifold dimension estimated from IRMAE for 5 models with different weight initializations. (b) Ground truth and representative reconstruction for best model (red square in subplot (a)). (c) Singular value spectra training dynamics for best model. . . 168

1

General introduction

The primary motivation for this thesis is understanding and modeling wall-bounded turbulence, which is defined as turbulent fluid flow occurring near a wall. Wall-bounded turbulence—or wall-turbulence for short—occurs in nearly all flows of practical interest. On the largest scale, atmospheric turbulence can affect weather patterns and pollution levels near the earth's surface. On the finest scales, wall-turbulence can occur in blood flow and cause damage to vessel walls. On intermediate scales, wall-turbulence increases drag when pumping fluids through pipes or ducts, and when operating vehicles that travel through the air or sea—across these cases, the turbulent dissipation occurring near the wall directly contributes to $\sim$ 6% of the world's energy usage in industry and commerce [95]. Even small improvements in turbulent drag reduction can lead to considerable energy savings and lower CO_2 emissions. With an eye towards the holy grail of controlling and reducing turbulent drag, this thesis is divided into two goals: 1) characterizing the underlying physical structures embedded in wall-turbulence, and 2) developing an efficient and accurate time evolution model that can ultimately be applied to predicting wall-turbulence.

For the first goal of characterizing wall-turbulence structures, the aim is to obtain a mechanistic understanding of wall-turbulence. By "structures", we mean the vortices/eddies. Ever since da Vinci's drawing of turbulence from the 1500s, people have long pictured turbulence as an aggregation of chaotic vortices that co-exist at many scales. Apart from the

structures being multiscale, we are interested in their other physical properties (shape and size) and dynamical aspects (where they are located, and how they interact with each other or the wall). In Section 1.1, we provide necessary background on the statistical behavior of turbulence and existing theory for the key features of eddies.

One prominent method for characterizing eddies is to perform a modal decomposition on a wall-turbulence dataset, where the modes then directly represent the structures contained in the dataset. Some techniques use a fixed basis that is independent of the dataset it is applied to. Other techniques are data-driven, in the sense that the modal basis is optimized under some metric (e.g., the variance of the data projected onto the modes is to be maximized). In Section 1.2, we provide background on various modal decomposition techniques and discuss the benefits and drawbacks of each of them for the purpose of unambiguously extracting and characterizing structures in wall-turbulence datasets.

The second goal of this thesis is to develop an efficient and accurate time evolution model for spatiotemporally chaotic dynamical systems. Chaotic dynamical systems include the double pendulum, chemical reaction networks, and turbulence. The defining trait of a chaotic dynamical system is sensitivity to initial conditions—a small change in the initial state leads to solutions that diverge exponentially fast. Since wall-turbulence is both chaotic and multiscale, making accurate time predictions is exceedingly difficult. Traditional approaches involve discretizing and numerically solving the Navier-Stokes equations (NSE); however, performing direct numerical simulations (DNS) of the NSE is computationally intractable for most systems of practical interest due to the large number of grid points needed to resolve the eddies at all length and time scales.

With large training datasets from experiments and simulations becoming more readily available, there is great potential for using machine learning-based models as an alternative to DNS for high-fidelity prediction of turbulence [14]. We provide a survey on data-driven methods for dimension reduction in Section 1.3 and time evolution methods in Section 1.4. By combining dimension reduction with time evolution, one can develop a data-driven

reduced-order model (ROM) that can be evaluated with far fewer degrees of freedom than solving the DNS. Most ROMs are black-box, as in created solely from data, so the second goal of this thesis centers on developing a hybrid ROM that synergistically blends physics (knowledge of the governing equations) with data. Note, the second goal of this thesis will involve developing a hybrid ROM that will only be applied to simpler chaotic dynamical systems, with the application to wall-turbulence being left to future work.

1.1 Wall-turbulence

We first start by defining quantities regarding turbulence in general, not just wall-turbulence. The incompressible Navier Stokes Equations (NSE) that govern fluid flow are

$$\frac{\partial U_i}{\partial t} + \frac{\partial U_i U_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_j^2}\,, \qquad \frac{\partial U_i}{\partial x_i} = 0\,, \tag{1.1}$$

where U_i is the ith velocity component, ρ is the density, ν is the kinematic viscosity, and p is the modified pressure. The Reynolds number, Re, is defined as $\mathrm{Re} = \frac{UL}{\nu}$, where U is the characteristic velocity of the system and L is the characteristic length. The Reynolds number describes the ratio between inertial forces and viscous forces. At a critical Reynolds number, a flow will transition from being laminar to being turbulent wherein structures within the flow will arise that are chaotic and multiscale.

To ensure that the turbulent dynamics are fully captured in a DNS, all structures from the Kolmogorov length scale to the characteristic length scale of the domain must be resolved. One can calculate that the number of mesh points (degrees of freedom) then scales with $\mathrm{Re}^{9/4}$ [214], thus making DNS computationally intractable. While DNS finds use in academic settings, turbulence modeling is used much more often in industry settings. A plethora of turbulence models have been developed that can significantly speed up simulations by trading off accuracy.

One class of turbulence models work with the Reynolds-Averaged Navier-Stokes (RANS)

equations. Let the Reynolds decomposition be defined as $U_i = \overline{U_i} + u_i$, where an overbar represents a time average and u_i is the fluctuating velocity component. Then, the RANS equations can be derived as

$$\frac{\partial \overline{U_i}\,\overline{U_j}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu\frac{\partial^2 \overline{U_i}}{\partial x_j^2} - \frac{\partial \overline{u_i u_j}}{\partial x_j}, \qquad \frac{\partial \overline{U_i}}{\partial x_i} = 0, \tag{1.2}$$

where $\overline{u_i u_j}$ is the Reynolds stress term that falls out of the nonlinear term in the NSE. In order to solve for the mean velocity field $\overline{U_i}$, $\overline{u_i u_j}$ must first be modeled or approximated—this is the closure problem. Although many RANS models have been developed, none of them are able to provide accurate results for all types of geometries and flow conditions. While RANS is much faster to compute than DNS, it can only provide time-independent solutions.

Another class of turbulence modeling is known as large-eddy simulations (LES). In contrast to DNS, LES resolves only the larger length scales while modeling the effects of the smaller length scales that have been filtered away. Consequently, LES provides time-dependent solutions like DNS but at a reduced computational cost. The closure models in the case of LES attempt to model the residual stress tensor term that appears when filtering the NSE. Similar to RANS, LES closure models are not completely general—the development and refinement of them often require decades of extensive domain-specific knowledge before they become viable in industrial applications. As a result, the later parts of this thesis will be focused on developing machine learning alternatives to traditional turbulence modeling.

The rest of this chapter is as follows. Section 1.1.1 discusses structures found in all turbulent flows. Section 1.1.2 provides an overview of the attached eddy hypothesis (AEH), which is the central theory concerning the energetically-dominant structures in wall-turbulence. In Section 1.1.3, we discuss the AEH-inspired attached eddy model (AEM), which can aid in predicting velocity statistics. In Section 1.1.4, we describe the spectral counterpart of the AEM. Lastly, in Section 1.1.5, we discuss the dynamics and interactions between wall-

turbulence structures.

1.1.1 Structures in turbulence

Big whorls have little whorls
Which feed on their velocity,
And little whorls have lesser whorls
And so on to viscosity.

— Lewis F. Richardson, 1920

This century-old rhyme by Richardson [191] provides an observation on the structures ("whorls") existing in all turbulent flows. The rhyme depicts a physical picture of the multitude of length scales in which structures exist. Additionally, the rhyme eludes to the mechanism by which energy is transferred from larger-scale structures to smaller-scale structures. This mechanism for the transfer of energy between structures at each scale is now known today as the "energy cascade". Broadly, the energy cascade states that energy is injected into the large scales, transferred amongst structures at intermediate scales, and ultimately dissipated as heat at the smallest scales.

The turbulent kinetic energy (TKE), given by $\frac{1}{2}\overline{u_i u_i}$, is the mean kinetic energy per unit mass contained in all of the structures or eddies in the flow. The TKE can be expressed as a sum of the energy density of the eddies across all length scales:

$$\frac{1}{2}\overline{u_i u_i} = \int_0^\infty E(k)\,\mathrm{d}k\,, \tag{1.3}$$

where $E(k)$ is the energy spectrum and $k = \frac{2\pi}{\lambda}$ is the wavenumber (inverse length scale), with λ being the wavelength.

In 1941, Kolmogorov quantified the energy cascade using statistical approaches [113]. By using scaling arguments, Kolmogorov derived that within the inertial subrange, which

contains the intermediate range of length scales, the energy spectrum follows the scaling law

$$E(k) \sim k^{-5/3}\,. \tag{1.4}$$

This 5/3 power law was derived for only homogeneous isotropic turbulence (HIT), where the velocity statistics are the same everywhere and in all directions. Kolmogorov's second similarity hypothesis [113], which is used to derive the the 5/3 power law, states that the statistics of the flow for structures in the inertial subrange have a universal form and depend only on the energy dissipation rate. In other words, the eddies in HIT are statistically self-similar.

The 5/3 power law is one of the defining traits of turbulence, and was arguably discovered in part due to first recognizing that turbulence is multiscale. A motif of this that will appear later, and motivation for us, is that physically characterizing the structures can facilitate a mechanistic understanding of wall-turbulence.

1.1.2 Attached eddy hypothesis

Before describing structures in wall-turbulence, we first define nomenclature and regions/layers specific to wall-turbulence. Henceforth, let x, y, z refer to the streamwise, wall-normal, and spanwise directions, respectively. Let U_1, U_2, U_3 refer to the instantaneous streamwise, wall-normal, and spanwise velocity components, respectively. The fluctuating streamwise velocity component is given by the Reynolds decomposition $u_1 = U_1 - \overline{U}_1$, where again overbars denote time averages. Let $\delta_\nu = \nu/u_\tau$ be defined as the viscous length scale, where $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity and τ_w is the wall shear stress.

There are two primary regions, the inner region and the outer region, that are present in all wall-bounded turbulent domains. In the inner region where viscosity is important, the mean velocity profile $\overline{U_1}(y)$ scales only with $y^+ = y/\delta_\nu$—here, we use $^+$ to denote non-dimensionalization with δ_ν (note, we will also interchangeably call δ_ν a "wall unit" length

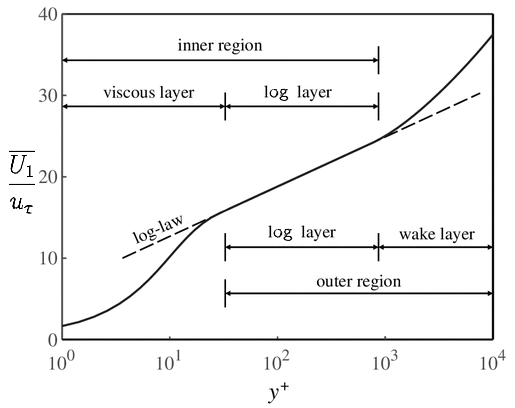


Figure 1.1: Log-law in wall-turbulence.

scale). In the outer region, viscosity is unimportant and the mean velocity scales only with the outer length scale δ_o; depending on the flow geometry, δ_o can be the pipe radius, half-channel height, or boundary layer thickness. The overlap of the inner and outer regions is called the log layer. Below the log layer is the viscous layer where viscosity dominates, and above the log layer is the wake layer where inertia dominates. Within the log layer, the mean streamwise velocity profile follows

$$\frac{\overline{U_1}}{u_\tau} = \frac{1}{\kappa}\log(y^+) + C\,, \tag{1.5}$$

where κ is the von Kármán constant and C is an additive constant. Figure 1.1 shows a depiction of the log-law. The log-law for $\overline{U_1}$ is the most prominent feature of wall-bounded turbulence and appears for all canonical wall-bounded geometries and a wide range of Re [182]. Although the exact values for the extents of the log layer is still disputed [7], the general consensus is that its lower limit is $y^+ = 30$ and its upper limit is $y =\sim 0.3\delta_o$.

Townsend's attached eddy hypothesis (AEH) [219] from 1976 is a statistical model that has been guiding the view of wall-turbulence structures in the outer region for decades. The AEH states that the energetically-dominant eddies randomly distributed in wall-bounded flows are attached to the wall and self-similar. It is important to note that "attached" here can mean physical-attachment to the wall, but it is more so a statement that these eddies

are coherent with the wall and would behave differently if the wall was not there. The AEH states that these attached eddies are only dominant at asymptotically high Re when the viscous layer is infinitely thin such that the attached eddies do, in a sense, extend down to the wall. However, as will be discussed throughout this section, attached eddies are still relevant at non-asymptotically high Re. The result that gives the AEH its credibility is that it can be used to derive the log-law for the second moment of the streamwise velocity U_1 (i.e., the streamwise Reynolds stress $\overline{u_1 u_1}$) within the log layer, similarly to the first moment derived in Equation 1.5.

The outline of the derivation for the AEH is as follows. Let δ be the height of an attached eddy. The underpinning assumption is that eddies are self-similar in geometry, such that the velocity field induced by each eddy scales with δ. By shifting the location of a given eddy $\boldsymbol{x} = (x, y, z)$ around its center position $\boldsymbol{x}_c = (x_c, y_c, 0)$ and scaling by its height, the velocity field contributed by any eddy can be written as

$$U_i(\boldsymbol{x}) = U_o f_i \left(\frac{\boldsymbol{x} - \boldsymbol{x}_c}{\delta} \right) , \tag{1.6}$$

where U_o is a characteristic velocity scale (to be determined) and f_i is the universal function for the velocity field of component i induced by an eddy. Now, let δ_1 be the height of the smallest attached eddy, which is generally regarded to be $\mathcal{O}(100\delta_\nu)$ [1, 217, 257]. Recall δ_o is the outer lengthscale (e.g., pipe radius). The Reynolds stress can then be expressed using the eddy intensity function $I_{ij}(y/\delta)$ as

$$\frac{\overline{u_i u_j}}{U_o^2} = \int_{\delta_1}^{\delta_o} I_{ij} \left(\frac{y}{\delta} \right) p_H(\delta) \, \mathrm{d}\delta, \tag{1.7}$$

where $p_H(\delta)$ is the number density of attached eddies at scale δ (this also needs to be determined), and I_{ij} is given by

$$I_{ij} \left(\frac{y}{\delta} \right) = \int_{-\infty}^{\infty} f_i \left(\frac{\boldsymbol{x}}{\delta} \right) f_j \left(\frac{\boldsymbol{x}}{\delta} \right) \, \mathrm{d} \left(\frac{x}{\delta} \right) \, \mathrm{d} \left(\frac{z}{\delta} \right) . \tag{1.8}$$

By assuming certain forms for $I_{ij}(y/\delta)$ for small y/δ_o and then recognizing that the Reynolds shear stress satisfies $\overline{u_1 u_2} \approx -u_\tau^2$ in the log layer, Townsend found that $U_o = u_\tau$ and the form for $p_H(\delta)$ needs to be $p_H(\delta) = \mathcal{M}/\delta$ with $\mathcal{M}$ being a universal constant (this conclusion about the number density function matches our intuition, where smaller attached eddies are more abundant than larger ones). It was then shown that within the log layer, the streamwise Reynolds stress profile follows

$$\frac{\overline{u_1 u_1}}{u_\tau^2} = -A \log\left(\frac{y}{\delta_o}\right) + B\,, \tag{1.9}$$

where A is the Townsend-Perry constant, and B is an additive constant. This AEH-based derivation of the log-law for $\overline{u_1 u_1}$ is robust to any shape of the representative eddy as long as its eddy intensity functions $I_{ij}(y/\delta)$ satisfy certain asymptotic conditions for $y/\delta_o \to 0$ [148]. Outside of theory, the log-law for $\overline{u_1 u_1}$ has been widely observed in the log layer for experiments and simulations [83, 195]. In contrast to HIT where eddies are only statistically self-similar, here the AEH states that eddies are also physically self-similar. That said, the exact shape of the eddies remains elusive.

1.1.3 Attached eddy model

The attached eddy model (AEM) is a kinematic model developed by [176] that was directly inspired by the statistical model of AEH. The AEM specifies a shape for a statistically-representative attached eddy and then constructs a random distribution of them, with the scale progression of the attached eddies increasing by a factor of 2. Various shapes were chosen, including a Λ-shaped vortex that is similar to the hairpin vortex that was first proposed by [216] in 1956 to be the underlying eddy structure. The AEM assumed the discrete form of the same number density function $p_H = \mathcal{M}/\delta$ used in the AEH. By assuming vorticity was confined to the structure of an attached eddy at scale δ, Biot-Savart law was used to calculate the velocity field induced by the structure of the attached eddy. Lastly,

the velocity field was related to $I_{ij}(y/\delta)$ and the log-law for $\overline{u_1 u_1}$ within the log layer was correctly derived. Note that these results are robust as long as a reasonable shape for the representative attached eddy is chosen, similar to the result for the AEH.

Since the AEM can predict $\overline{u_1 u_1}$ within the log-layer, much work has been done to extend the AEM to predict all Reynolds stress components in both the log-layer and the wake-layer—doing so effectively produces a RANS model that is grounded in the elegant theory of the AEH. In [178], the AEM was extended in two ways: 1) the scale progression of the eddies was made continuous to simplify calculations, and 2) it was modified to also predict the profile for $\overline{U_1}(y)$ in the log layer and the wake layer. To facilitate the second modification, the number density function $p_H(\delta) = \mathcal{M}/\delta$ was multiplied with a weighting function $w(\delta)$ to increase the number of larger-scale attached eddies in the flow. One can view this modification to p_H as artificially creating a higher density of attached eddies that have a height δ above the top of the log layer (i.e., creating packets of attached eddies). The weighting function $w(\delta)$ used to achieve this can vary depending on the type of geometry (e.g., pipe vs. boundary layer flow), and it is necessary to derive the correct profile for $\overline{U_1}(y)$ in the outer region [178]. In a further modification to the AEM, [175] found that $w(\delta)$ was also necessary to derive the correct profiles for some of the Reynolds stresses in the outer region. Furthermore, for favorable or adverse pressure gradient (FPG or APG) in a TBL flow (which is very common in most flow applications), the velocity scale defined by U_o in Equation 1.6 was found to not be equal to u_τ and required another weighting function $T(\delta)$ [179]. In [180], a method to find w and T separately was developed and closure was shown to be possible (but they never proceeded to test if it worked). In [177], the work left by [180] was tested. It was found that for APG/FPG, just having a self-similar hierarchy of attached eddies was not enough. In order to accurately predict the Reynolds stresses, a secondary set of self-similar, detached eddies needed to be correctly specified and considered separately. A shape for the detached eddies (similar to upside-down hairpin vortices) was used, and the refined AEM used in [177] showed that all Reynolds stresses were predicted

fairly accurately in the log-layer and wake-layer. However, with two sets of eddies whose representative shapes needed to be found through trial and error and lack of generalization to variable flow conditions and geometries, no further efforts have been made to make the AEM into a robust RANS model.

1.1.4 The -1 power law

Similar to how a 5/3 power law is inherent with homogeneous turbulence (see Section 1.1.1), there is a -1 power law that is inherent with wall-turbulence. The power spectral density (PSD) describes the energetic contribution of a signal's spectral components as a function of wavenumber (which is inversely proportional to scale). For 1D signals, the PSD is equivalent to an energy spectrum. The PSD of a signal in real space is essentially the magnitude-squared of the signal's Fourier transform [184]. The spectral AEM developed by [178] builds a model of attached eddies in spectral (i.e., Fourier) space instead of physical space. It will be shown that it can be used to derive the log-law for $\overline{u_1 u_1}$ just like the AEM and AEH, but the way it does so is by making a key assumption about the existence of a power law scaling of $\phi_{uu}(k_x) \sim k_x^{-1}$, where ϕ_{uu} is the PSD of a streamwise fluctuating velocity signal $u_1(x)$ and k_x is the streamwise wavenumber.

The spectral AEM relies on dimensional analysis arguments to determine wavenumber scalings under which the PSD will be universal and collapse on each other. The schematic of a premultiplied PSD measured at wall-normal location $y/\delta \to 0$ is shown in Figure 1.2 (the different eddy types will be discussed shortly). $k_x y\, \phi_{uu}(k_x y)/u_\tau^2\, \log(k_x y) \sim k_x^{-5/3}$ The purpose of the premultiplication of the PSD with k_x is to ensure that when a log-scale is used for k_x, equal areas under the curve correspond to equal integrated energies; moreover, as a byproduct, a $\phi_{uu} \sim k_x^{-1}$ scaling law will appear as a plateau region in the premultiplied PSD.

We now show an outline of using the spectral AEM [178] to derive Equation 1.9. The derivation relies on scaling arguments for collapsing a set of premultiplied PSD measured at

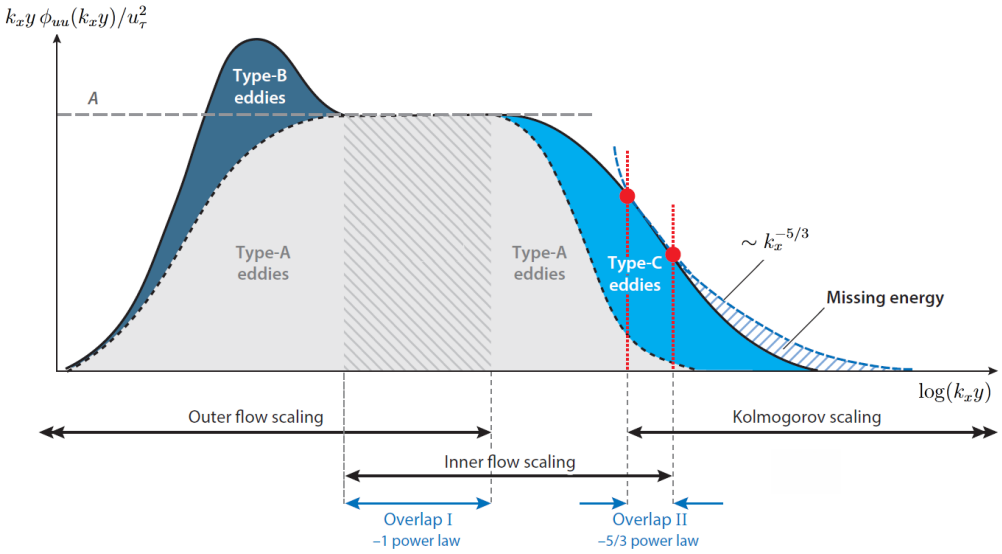


Figure 1.2: Sketch of the pre-multiplied PSD of a fluctuating streamwise velocity signal measured close to the wall (see text for description of the scalings and 3 eddy types). Adapted from [148].

$y/\delta \to 0$. For the outer flow scaling where $k_x \sim \mathcal{O}(1/\delta_o)$, the premultiplied PSD should be independent of y for $y/\delta \to 0$ such that $k_x\phi_{uu}(k_x)/u_\tau^2$ is equal to a universal function $h_1(k_x\delta_o)$. Similarly, for the inner flow scaling where $k_x \sim \mathcal{O}(1/y)$, $k_x\phi_{uu}(k_x)/u_\tau^2$ is equal to a universal function $h_2(k_x y)$. Matching in the overlap I region gives $h_1(k_x\delta_o) = h_2(k_x y) = A$, where A is a constant. This implies that in the overlap region I, $\phi_{uu}(k_x)/u_\tau^2 = A/k_x$—this is known as the $\phi_{uu}(k_x) \sim k_x^{-1}$ law. A similar set of scaling arguments leads to the $\phi_{uu}(k_x) \sim k_x^{-5/3}$ scaling law in the overlap II region. The $k_x^{-5/3}$ scaling law applies to all types of turbulence (e.g., turbulent jet flows) because turbulence energy is always dissipated at the Kolmogorov scale; however, the k_x^{-1} scaling law is exclusive to wall-bounded turbulence because of the consideration of a wall in the formulation here. One can write

$$\frac{\overline{u_1 u_1}}{u_\tau^2} = \int_{-\infty}^{\infty} \frac{k_x\phi_{uu}(k_x)}{u_\tau^2}\,\mathrm{d}(\ln k_x) = \int_{\ln(a/\delta_o)}^{\ln(b/y)} \frac{k_x\phi_{uu}(k_x)}{u_\tau^2}\,\mathrm{d}(\ln k_x) + C(y,\mathrm{Re}) \tag{1.10}$$

$$= -A\ln\left(\frac{y}{\delta_o}\right) + B(y,\mathrm{Re}), \tag{1.11}$$

where a and b are the constants defining the boundaries of the overlap I region in Figure 1.2, $B(y,\mathrm{Re}) = A\ln(a/b) + C(y,\mathrm{Re})$, and $C(y,\mathrm{Re})$ is the energy in the premultiplied PSD

outside the overlap I region that has a slight y and viscous dependence. Equation 1.11 is identical to Equation 1.9. To summarize, the spectral AEM directly shows that a k_x^{-1} scaling law in the premultiplied PSD measured at $y/\delta \to 0$ is needed to derive the log-law for $\overline{u_1 u_1}$.

For the past few decades, there is a tense debate in the field of wall-bounded turbulence surrounding the result of this spectral AEM. The conundrum here is that the assumed plateau region related to the k_x^{-1} scaling law is rarely observed in experiments and simulations [28, 166] but the log-law for $\overline{u_1 u_1}$ has been widely-observed. There have been many reasons proposed for why there is this discrepancy. The primary reason is that there are other types of structures in the flow besides attached eddies. As seen in Figure 1.2, attached eddies are called Type-A eddies and contribute the most to the Reynolds stress (area under the curve). So, the plateau region showing a k_x^{-1} law will only be observable at an asymptotically high friction Reynolds number of $\mathrm{Re}_\tau = \delta_o/\delta_\nu$ of $\mathcal{O}(10^6)$ where there is enough scale separation.

Since Re_τ rarely exceeds $\mathcal{O}(10^4)$ in experiments and simulations, the overlap region I becomes smaller in width and the plateau region starts to become masked by the Type-B eddies in the flow [4] (see Figure 1.2). Type-B eddies are very large, wall-attached structures with height $\sim 0.3\delta_o$ to $1\delta_o$ and streamwise extent $\mathcal{O}(10\delta_o)$ [110], and they are generally referred to as very large scale motions (VLSM). The relationship between VLSMs and attached eddies will be further discussed later. Lastly, the Type-C eddies are the Kolmogorov-scale, wall-detached eddies that are isotropic and viscosity-dependent (these are said to surround the vortex structure of Type-A and Type-B eddies).

It has been recently shown that by adding a y/δ_o dependence for the PSD within the framework of the original spectral AEM, a perfectly flat plateau region is not a necessary condition for a log-law to still appear for $\overline{u_1 u_1}$ [89]. Regardless on whether or not the k_x^{-1} law is a necessary feature of wall-bounded turbulence, the more demanding issue in the pursuit of forming a cohesive picture of the structures in the outer region is to understand the interaction of the Type-A (attached) eddies and Type-B eddies (VLSMs) and how they change with increasing Re_τ.

1.1.5 Dynamical nature of structures in wall-turbulence

We conclude this section with a discussion on the dynamics of the structures found in wall-turbulence. We start with turbulent structures located in the viscous layer (see Figure 1.1). There, the structures were first discovered and characterized in terms of their shape using a structural approach [112]. In the viscous layer, two primary structures exist: 1) streaks, which are spanwise alternating regions of high-speed and low-speed fluid which have a universal/characteristic spacing of $\sim 100\delta_\nu$, and 2) quasi-streamwise vortices. As for the dynamics and sustainment of these structures, [72] proposed the self-sustaining process (SSP). In [72], the insight into the dynamics of the viscous layer was achieved by analyzing DNS data for planar Couette flow in a minimal flow unit (MFU), which is a flow domain just small enough to sustain turbulence. They found that the SSP is a cyclic mechanism that consists of the following three steps: 1) quasi-streamwise vortices act to advect mean shear and generate streaks, 2) streaks breakdown due to unstable, streamwise-varying Fourier modes, and 3) nonlinear interactions of these modes strengthen/regenerate the quasi-streamwise vortices. We see once again, here with the viscous layer, that detailed characterization of turbulent structures is a necessary first step for paving the way to a mechanistic understanding and modeling of turbulent dynamics—we now aim to do so for the log-layer and wake-layer as well.

Unlike for structures in the viscous layer, there is no clear picture for the shape of the outer region structures (Type-A attached eddies or Type-B VLSMs) and for how they are sustained. The general consensus for the shape of Type-A attached eddies is hairpin vortices. As for their dynamics, the general consensus is that attached eddies and VLSMs are generated in a bottom-up process. The idea is that the fluid in the viscous layer are convected upwards to form the smallest attached eddy of height $\mathcal{O}(100\delta_\nu)$. Then, these attached eddies merge with each other to form the next largest attached eddy. This merging continues until the attached eddies reach a height of $\sim 0.3\delta_o$ to $1\delta_o$ and form packets, called large scale motions (LSM), and these packets of attached eddies then merge to form long organized structures

known as VLSMs [217, 257]. However, this bottom-up process is not universally agreed upon. In [49], it was shown that the LSMs and VLSMs in the outer region can be sustained even without any attached eddies being generated above the viscous layer.

A more rigorous study of the sustainment of structures in the outer region rely on analyzing the NSE. Exact coherent states (ECS)—which are nonlinear invariant solutions of the NSE that can be steady states, traveling waves, or traveling waves with a time-periodic modulation—have been shown to represent the streaks and quasi-streamwise vortices of the SSP structures in the viscous layer [231]. Interestingly, there has been evidence that these ECS which correspond to SSP structures may also exist in the outer region and are self-similar [40]. More convincing evidence of the self-similarity of SSP structures within the outer region has been shown by overdamped, large eddy simulations (LES) that can be used to isolate structures at a specific scale [33, 87]; in these simulations, the SSP structures were found to be self-similar throughout the entire outer region and thus consistent with the AEH. Note that the attached eddy in the case here is composed of the dynamically interconnected quasi-streamwise vortices and streaks. On the largest scale, the "attached eddy" is comprised of a streaky motion of streamwise extent $15\delta_o$ and height $\sim 1\delta_o$ that is surrounded by quasi-streamwise vortices of streamwise extent $\sim 3\delta_o$ and height $\sim 1\delta_o$; this aspect ratio of the SSP structures is maintained throughout the outer region all the way down to the viscous layer at $y^+ = 10$. The largest streak component resembles a VLSM and the largest quasi-streamwise vortices component resembles LSMs. Most importantly, the overdamped LES results imply that turbulence can be sustained at each individual scale without any forcing from smaller scales, thus lending more support against the bottom-up process discussed previously.

The understanding of the interplay between the SSP in the viscous layer (and possibly also in the outer region) and the structures in the outer region remains elusive. Gaining insight into this interplay between all turbulent structures necessitates a detailed characterization of attached eddies and LSMs/VLSMs. Many classes of methods have been applied to determine the shape and degree of self-similarity associated with attached eddies [148]. Inspired by the

hairpin vortex structure first proposed by [216], flow visualization methods, such as particle image velocimetry (PIV), are popular; however, although the embedded flow structures have been observed to be attached to the wall, there is no conclusive evidence on their shape and if they are hairpin vortices or not [42]. One reason for the inconclusiveness of the shape of an attached eddy is that there is no well-agreed upon method to visualize vortices. From PIV data, [98] showed that an attached hairpin vortex that was attached to the wall was present at one moment but disappeared in the next once the threshold chosen for the visualization is crossed. An attached eddy is just a statistically-average representation of the spatiotemporally chaotic motions in the flow, so trying to determine its shape through flow visualization can be misleading. In the next section, we consider a more principled method of determining the shapes of attached eddies.

1.2 Modal-based characterization of structures in data

One promising class of techniques to extract the shape of structures from data is modal analysis. Generally, a modal basis decomposes a vector $u(x)$ into

$$u(x) = \sum_i c_i \phi_i(x) \,, \tag{1.12}$$

which is a sum of the product between the modal coefficients c_i and the basis elements or "modes" ϕ_i over all modal indices i. Provided the basis is orthogonal, one can calculate c_i with (u, ϕ), where $(\cdot, \cdot)$ denotes an inner product.

The choice in the type of basis can affect how well the structures in given dataset can be represented and analyzed. In the context of wall-turbulence, we desire the modes in the basis to represent attached eddies. Therefore, our data is either vorticity fields or the fluctuating velocity fields containing signatures induced by the eddies (this work, as with many others, only considers the latter case); in either case, the chosen basis should have three properties:

1. Learned from data

2. Multiscale

3. Spatially-localized

The first property comes from the fact that we would like the shape of the modes to be discovered from data, as opposed to being set *a priori*. The second property ensures that the modes are multiscale, since both the AEH and AEM provide evidence that attached eddies are multiscale. The third property comes from the fact that attached eddies are likely spatially-localized. Although the AEH does not explicitly require spatial-localization in its formulation, many AEM models do so; furthermore, Lumley has shown that unlocalized structures should not be considered eddies [144, 145, 215]. In the rest of this section, we explore various modal decomposition techniques and discuss their ability to satisfy the three properties listed above.

1.2.1 Proper orthogonal decomposition

The most prominent modal-based method for analyzing coherent structures in turbulence is proper orthogonal decomposition (POD) [207]. POD is known in other research areas as the Karhunen–Loève procedure and also as principal components analysis (PCA). POD is used to discover an orthonormal basis from data—it is well suited for the first property of a modal basis we desire.

The first POD mode ϕ_1 is found with the optimization problem

$$\max_{\phi_1} \left\langle |(u, \phi_1)|^2 \right\rangle \,, \tag{1.13}$$

$$s.t. \quad \|\phi_1\| = 1 \,, \tag{1.14}$$

where $\langle \cdot \rangle$ denotes an ensemble average. POD aims to maximize the variance (energy) of the dataset projected onto the first POD mode with the constraint that the first POD mode has unit norm. In practice, POD is found through an eigendecomposition. Let $Z =$

$[u^{(1)}, u^{(2)}, \ldots, u^{(M)}]$ be the data matrix whose columns contain M snapshots of the data vectors $u^{(j)}$, $j = 1, \ldots, M$, where $u^{(j)} \in \mathbb{R}^N$. The POD modes are then the eigenvectors of the covariance matrix ZZ^T, where ^{T} denotes transpose.

While POD has seen success in applications to nonlinear problems such as turbulence [80], there are limitations to POD since it is a linear process that projects data onto linear spaces. In our case of interest, which is characterizing turbulent structures, POD has the disadvantage that it generally returns modes that are not spatially-localized; in fact, if POD is applied in a statistically-homogeneous direction (e.g., streamwise), only Fourier modes (which have no spatial-localization at all) are returned [80]. Spatial-localization of basis modes is the second property we desire, which POD generally does not provide.

1.2.2 Other modal decompositions

We discuss other notable modal decomposition techniques. The first is dynamic mode decomposition (DMD) [222]. Let $Z' = [u^{(2)}, u^{(3)}, \ldots, u^{(M+1)}]$, which is the same data matrix as Z but with the data vectors shifted by one column (i.e., one time step if the data comes from timeseries). DMD then calculates the linear map $A = Z'Z^\dagger$, where † denotes pseudoinverse. The eigenvectors of A are the DMD modes and the eigenvalues represent the growth/decay rate (or purely oscillation if they lie on the unit circle in the complex plane). Unlike POD, DMD modes are not orthogonal. Instead of being ordered by energy content, DMD modes are frequency ordered. Moreover, if the data is stationary and zero-mean, DMD reduces to a discrete Fourier transform [218]. DMD has ties to the Koopman operator, which will be discussed in detail in Section 1.4.

An operator-based modal decomposition method is resolvent analysis. Resolvent analysis expresses the nonlinear term in the NSE as an input forcing to the resolvent (linearized NSE) to obtain the output velocity response. The linear resolvent modes have been shown to reconstruct known coherent structures and even ECS, as well as producing reconstructed velocity fields containing self-similar, wall-attached structures [88, 152, 162]. One challenge

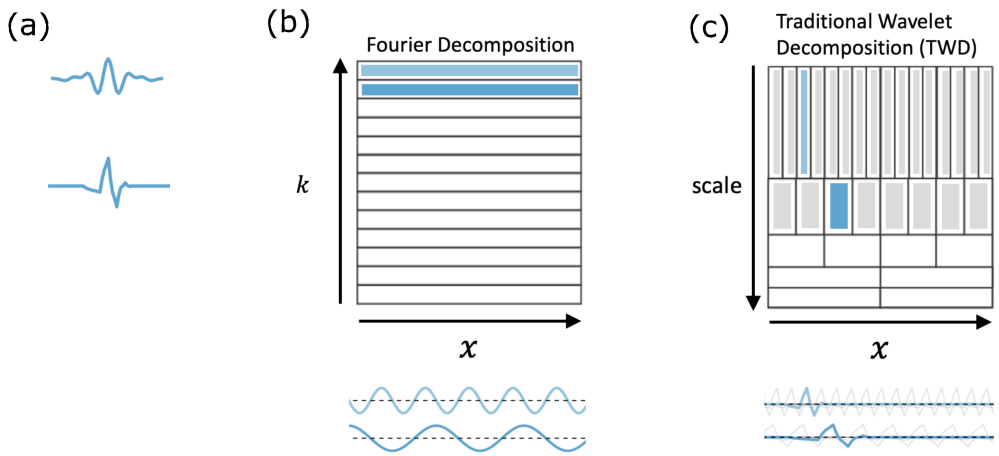


Figure 1.3: (a) Example wavelets, with Meyer (top) and db2 (bottom). (b) Scale-space decomposition by Fourier modes (top) with corresponding Fourier modes (bottom). (c) Scale-space decomposition by wavelet modes (top) with corresponding wavelet modes (bottom).

of resolvent analysis is finding the forcing weight coefficients to represent the nonlinear dynamics. Overall, both DMD modes and resolvent modes are not guaranteed to satisfy any of the three properties listed in the beginning of this section that we desire in our basis.

1.2.3 Wavelets

A wavelet is essentially a wave with finite extent, hence the diminutive suffix "-let". Traditionally, wavelets have a pre-specified shape. Figure 1.3(a) shows two different example wavelets. A wavelet basis merges two aspects of the two most well-known modal bases. While a Fourier basis provides excellent wavenumber (inverse scale) resolution of a signal, there is no spatial resolution/localization since the modes span the whole spatial domain. A Euclidean basis has excellent spatial resolution but no scale resolution. A wavelet basis strikes the balance between the two by using spatially-localized waves at different scales. Figures 1.3(b) and 1.3(c) illustrate these differences between Fourier modes and wavelet modes. As evidenced in Figure 1.3(c), a wavelet basis is composed of basis elements that are translations and dilations of each other.

A wavelet basis satisfies two of the three properties we desire, which are multiscale and spatially-localized. However, the third property of being learned from data is not satisfied.

Traditionally, a wavelet decomposition requires that the shape of the wavelets be specified *a priori* and self-similarity of the wavelets across every scale is then assumed. These two features of a wavelet decomposition—which we refer to as traditional wavelet decomposition (TWD)—are inauspicious for our goal of extracting the shape of structures from data and trying to objectively determine their degree of self-similarity. Chapter 2 of this thesis will explore how to combine the data-driven aspect of POD with TWD to discover self-similarity across scales (if any) instead of having it built-in already.

Wavelets are ideal for representing multiscale structures or features, and they have been used as building blocks for systems such as active matter, images, and networks. For example, wavelets have been used for image compression [55]. Wavelets have also found significant use in turbulence [3, 44–46, 104, 157, 158]. More details on wavelets can be found in [211].

A brief overview of TWD is provided here. Consider a periodic vector in $\mathbb{R}^N$, with N even. The wavelet basis will consist of N vectors that are self-similar across $p = \log_2 N$ stages or scales, and the decomposition to obtain the wavelet coefficients occurs one stage at a time. Let ψ_{-1} be the mother wavelet—this wavelet is specified beforehand and will be used to generate the other $N-1$ basis vectors. From the mother wavelet, one can construct the father wavelet, ξ_{-1}. During the first stage of the decomposition, $\mathbb{R}^N$ is split into two orthogonal subspaces that are each of dimension $N/2$: the approximation subspace V_{-1}, and the detail subspace W_{-1}. V_{-1} is spanned by $N/2$ mutually orthonormal even translates of ξ_{-1}, and W_{-1} is spanned by $N/2$ mutually orthonormal even translates of ψ_{-1}. Projecting a vector in $\mathbb{R}^N$ onto the approximation subspace V_{-1} yields a low-pass filtered version of the vector, and projecting the vector onto the detail subspace W_{-1} yields a high-pass filtered version of the vector (hence the nomenclature of "approximation" and "detail" subspaces); the vector is equal to the superposition of these two projections. This splitting is performed recursively on the approximation subspace, ultimately yielding a decomposition $\mathbb{R}^N = V_{-p} \oplus W_{-p} \oplus \ldots \oplus W_{-1}$ (see Figure 1.4). One may think of this decomposition as the direct sum of a very coarse approximation subspace and progressively finer detail subspaces. At each stage l, V_{-l} and

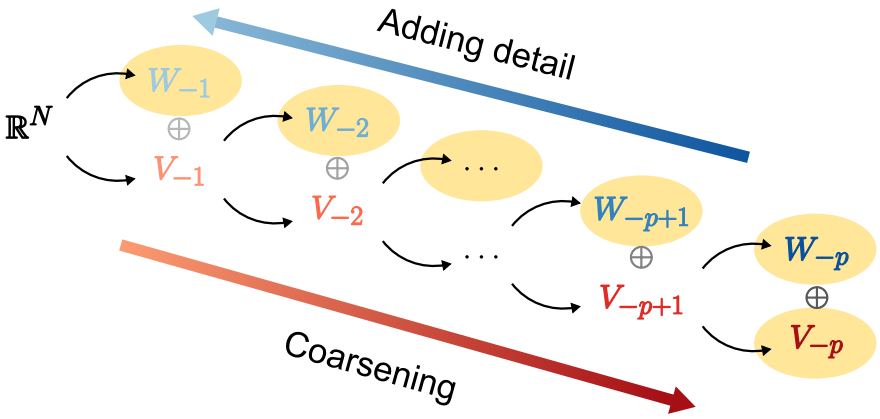


Figure 1.4: Subspaces formed by wavelets periodic in $\mathbb{R}^N$. The full space $\mathbb{R}^N$ is composed of the highlighted subspaces.

W_{-l} are respectively spanned by $N/2^l$ mutually orthonormal translates of ξ_{-l} and ψ_{-l}. The ψ_{-l} wavelet is essentially a factor-of-two dilation of the previous-stage wavelet $\psi_{-(l-1)}$, and similarly for ξ_{-l}.

The orthonormality conditions for the mother wavelet ψ_{-1} and its even translates can be stated in Fourier space as

$$\left|\hat{\psi}_{-1}(n)\right|^2 + \left|\hat{\psi}_{-1}(n+N/2)\right|^2 = 2\,, \quad n = 0, 1, \ldots, N/2-1, \tag{1.15}$$

where $\hat{}$ denotes the discrete Fourier transform (DFT). For a vector $z \in \mathbb{R}^N$, $z(n)$ is its n^{th} component, indexed beginning from zero. Given a ψ_{-1} that satisfies (2.8), the father wavelet ξ_{-1} can be constructed from ψ_{-1}, and the entire resulting wavelet basis is mutually orthonormal. In practice, wavelet bases are constructed using DFT calculations and filter banks. Some discussion on filters will appear in Chapter 2, and interested readers are directed to [55] and [50] for more details. We will equivalently refer to the TWD as a discrete wavelet transform (DWT) throughout this thesis.

We end this section by briefly discussing other types of wavelet transforms. Similar to DFT, there are 2D and 3D DWT; discussion on the 2D and 3D DWT will be left to Chapter 5, as the earlier parts of the thesis will only focus on the 1D DWT. One major class of wavelet transforms is the "continuous wavelet transform" (CWT). The CWT is a misnomer

in that it is used to refer to both to the continuous and discretized transform (i.e., CWT can be applied to discrete vectors). Unlike the DWT, the CWT does not provide a basis. In DWT, there are N total coefficients across $\log_2 N$ scales that correspond to the coordinates of the orthonormal basis; in CWT, there are N coefficients per scale (number of scales can be any integer less than N), so the basis is overcomplete and the CWT wavelets are not orthogonal to each other. DWT is used as a basis for compact representation and sees application mostly in image compression, while CWT is mostly used to analyze timeseries data since there is a higher frequency-resolution of N elements per scale instead of $N/2^l$ elements per scale as with the DWT. The CWT has been used to study scale interactions in which VLSMs modulate the amplitude and frequency of smaller-scale eddies near the wall [5]. As a side note, a coherence metric can be used to classify eddy types. Spectral coherence analysis (SCA) [4, 6] has been used to measure frequency coherence between a temporal signal measured near the wall and a signal in the log-layer to inform on activity of wall-attached eddies; similar to SCA, wavelet coherence analysis (WCA) with the CWT can also be used but with the advantage of time resolution (due to localization of wavelets) in addition to frequency (scale) resolution.

Up to now, we have reviewed topics surrounding characterizing coherent structures in turbulence. For the rest of this Chapter, we will shift our focus to review topics surrounding data-driven methods for dimension reduction and time evolution applied to spatiotemporally chaotic dynamical systems.

1.3 Dimension Reduction

The ability to efficiently and accurately forecast complex systems is highly sought after in engineering applications. Often times, we have partial or full knowledge of the physics of these systems and can perform simulations by discretizing and solving the governing equations. However, directly solving the full-order model (FOM) can be computationally pro-

hibitive for systems exhibiting multiscale, spatiotemporal chaos—the prime example being turbulence. Moreover, the development and refinement of these physics-based models often require decades of extensive domain-specific knowledge before they become viable in industrial applications. In contrast to traditional physics-based models, machine learning-based models are inherently black-box, in that they can be created solely from data. Recently, data-driven methods have seen an upsurge of use in areas such as fluid mechanics [13]. Here, we are interested in developing high-fidelity, reduced-order models (ROMs) of complex systems, which are crucial for iterative design and control of industrially relevant engineering applications. A ROM typically consists of two steps: 1) reduce the dimension or number of degrees of freedom of the system, and 2) evolve the dynamics of the low-order system. Here, we first provide a survey on data-driven methods for dimension reduction before discussing time evolution methods in Section 1.4.

1.3.1 Manifolds

For a wide variety of complex dynamical systems, the state space is nominally very high dimensional. In the case of systems governed by PDEs, the state space is even infinite-dimensional; however, the long-time dynamics of dissipative PDEs are expected to lie on a finite-dimensional invariant manifold [164, 249]. In the case that the invariant manifold contains the global attractor, it is also called an inertial manifold. Let the inertial manifold be denoted by $\mathcal{M}$, which has dimension $d_{\mathcal{M}}$. The manifold has three properties. The first property is that $\mathcal{M}$ is a Hausdorff space; for every pair of distinct points $p, q \in \mathcal{M}$, there are disjoint open subsets $U, V \subseteq \mathcal{M}$ such that $p \in U$ and $q \in V$. The second property is that $\mathcal{M}$ is second-countable, which means there exists a countable basis for the topology of $\mathcal{M}$. The third property is that every point in $\mathcal{M}$ contains a neighborhood that is homeomorphic to a Euclidean space with dimension $\mathbb{R}^{d_{\mathcal{M}}}$.

A global map from the manifold to a Euclidean space of dimension $\mathbb{R}^{d_{\mathcal{M}}}$ is not guaranteed; however, Whitney's embedding theorem [66, 201] states that the embedding dimension

is no more than twice the intrinsic dimension of the manifold, so a global map to $\mathbb{R}^{2d_{\mathcal{M}}}$ is guaranteed. One method of estimating the embedding dimension is the false nearest neighbors algorithm [17, 108]. Since the embedding dimension is often an overestimation of the inertial manifold dimension, a method to determine the latter is preferred. In [243, 244], the inertial manifold dimension of the Kuramoto-Sivashinsky equation (KSE) was estimated by analyzing spectra of Lyapunov vectors. In [39], the inertial manifold dimension of the KSE was estimated using Floquet vectors of unstable periodic orbits. Since these methods require access to the FOM for generating solutions to calculate the Lyapunov/Floquet modes, as well as knowledge of existing periodic orbits for [39], they are not general enough to estimate inertial manifold dimensions from time series data alone. The basis of a ROM is to estimate the dimension of the underlying inertial manifold and find a coordinate mapping from the ambient space onto the manifold.

1.3.2 Time-delay embeddings

In applications of practical interest, one often has limited access or coverage to the entire ambient space (e.g., limited number of buoy sensors covering the sea surface). In these scenarios, one only has partial observables $u_p(t) \in \mathbb{R}^{d_p}$. Whitney's theorem states that there is a diffeomorphic mapping to the manifold coordinates when $d_p \geq 2d_{\mathcal{M}}$ [65]. In the case that $d_p < 2d_{\mathcal{M}}$, one can still recover this diffeomorphism to the manifold by embedding the state and its $m-1$ time delays into the delay coordinate map

$$u_d(t) = [u_p(t), u_p(t-\tau), \ldots, u_p(t-(m-1)\tau)] \in \mathbb{R}^{m \times d_p}, \tag{1.16}$$

where τ is the delay spacing. Takens theorem states that for almost any choice in τ, there exists a diffeomorphic map from u_d to the manifold [201]. The delay spacing is typically estimated by the mutual information (MI) [54]. The methods discussed in the remainder of this section assume that one has full state space observations.

1.3.3 Principal component analysis

Assuming that one knows the inertial manifold dimension, we now examine methods for performing dimension reduction. The most widely used linear dimension reduction technique is principal component analysis (PCA). PCA is known in the fluid mechanics community as proper orthogonal decomposition (POD) [81], which was described in Section 1.2.1. In PCA, data are projected onto a subset of a modal basis ordered by energy content. The advantages of PCA is that there is an interpretable ordering of modes and a simple way to transform the data to and from the truncated basis/representation; however, for data that are more complex or nonlinear, many PCA modes often need to be retained to maintain model fidelity, thus limiting the amount of dimension reduction that can be performed. Additionally, invariant manifolds will often not be linear subspaces, so PCA cannot in general reduce the representation to even the embedding dimension.

There are various alternative methods to PCA that are nonlinear. Nonlinear methods that globally perform dimension reduction include kernel PCA [202] and diffusion maps [31], which are able to find more compact representations than PCA. Some nonlinear dimension reduction techniques use local computations, including locally linear embedding [196], Laplacian eigenmaps [9], isometric feature mapping [213], t-distributed stochastic neighbor embedding (tSNE) [78], and multiscale SVD [139]. The complication of all these nonlinear techniques, however, is that they either do not provide a way to project out-of-sample data onto the manifold representation, and/or they do not provide a coordinate system in which to transform to and from the manifold representation.

1.3.4 Autoencoders

One dimension reduction technique that can transform out-of-sample data points and provide a coordinate system for the reduced representation is the undercomplete autoencoder [60, 79]. An autoencoder uses a deep feedforward neural network to map data u from the state space to a low-dimensional latent space h, and another neural network for the reverse mapping.

The forward mapping is performed with an encoder $\mathcal{E}$ and the reverse mapping is performed with a decoder $\mathcal{D}$. The encoder (and similarly for the decoder) is typically a deep feedforward neural network, which is composed of multiple layers. Each layer multiplies the input vector with a weight matrix, adds on a bias weight vector, and evaluates the final result with a scalar activation function to introduce nonlinearities.

An autoencoder is trained by minimizing the reconstruction error of the autoencoder (AE) with the loss function

$$\mathcal{L}_{\text{AE}} = \langle \|u - \mathcal{D}(\mathcal{E}(u;\theta_{\mathcal{E}});\theta_{\mathcal{D}})\|_2^2 \rangle \,, \tag{1.17}$$

where $\theta_{\mathcal{E}}$ and $\theta_{\mathcal{D}}$ are, respectively, the neural network parameters of the encoder and decoder networks and $\langle \cdot \rangle$ denotes an ensemble average over a batch of training snapshots. The neural network parameters or weights are optimized through stochastic gradient descent. Hyperparameter tuning can consist of changing the learning rates, batch size, number of epochs, and strength of L2 weight regularization (if any). According to the universal approximation theorem [82], a sufficiently expressive neural network is able to approximate any continuous function. Consequently, the latter parts of this thesis related to data-driven reduced-order modeling will leverage this property of neural networks.

Autoencoders have been used to reduce dimension in turbulent flows [161, 168, 208] and other chaotic PDE systems [115, 120]. In addition to dimension reduction, Autoencoders have also been used to estimate the inertial manifold dimension of chaotic systems. In [128, 131], the inertial manifold dimension of the KSE was estimated by incrementally increasing the latent space dimension of the autoencoder until a sharp drop is observed in the state space reconstruction error; however, this methodology requires training multiple autoencoders with different bottleneck dimensions, and the drop is less distinct as the domain size increases [128].

1.3.5 Implicit rank-minimizing autoencoders (IRMAE)

A related technique uses a variant of an autoencoder called implicit rank-minimizing autoencoder (IRMAE) [97, 253]. Let Z be the data matrix with columns containing the snapshots. IRMAE estimates the inertial manifold dimension by employing linear layers in its architecture, where the linear layers contain only the weight matrix multiplication and no bias term or nonlinear activation. It has been shown in [97] that the linear layers act to drive the covariance of the latent space of the data, ZZ^T, to have minimal rank—a sharp drop in the singular value spectra of ZZ^T then gives the rank or estimate of the inertial manifold dimension of the data. Unlike normal autoencoders, one only needs to train a single IRMAE model with a sufficiently large bottleneck dimension, and the dimension estimate is more distinct [253]. However, even with IRMAE, obtaining a robust estimate of the inertial manifold dimension for more complex systems remains challenging, as seen with IRMAE applied to 2D Kolmogorov flow in [185].

With normal autoencoders, the latent space coordinates represent the coordinates on the manifold if the training data lives on the inertial manifold (i.e., transients in time series are removed). With IRMAE, the latent space coordinates are found by projecting the data onto the singular vectors whose singular values are large in magnitude.

1.4 Time evolution

We now turn to discussing various neural network-based methods for time evolution, either for the state space or a latent space found with an autoencoder. The methods we will discuss are trained on time series data of full observations of the state $u \in \mathbb{R}^{d_u}$, where d_u is the dimension of the state space. We assume observations are sampled at discrete time intervals with uniform data spacing τ (though some methods can handle non-uniform spacings).

Let the underlying (ground truth) autonomous dynamical system for the time series data

$u(t)$ be written as

$$\frac{du}{dt} = f(u; \mu), \tag{1.18}$$

where f is a vector field that depends on physical parameters μ (e.g., domain length). Some methods discussed here will try to approximate the function f, while others do not.

1.4.1 Common methods

Here, we discuss common neural network-based forecasting methods that are solely data-driven. Two widely-used data-driven methods for time evolution are recurrent neural networks (RNNs) [226, 227] and reservoir networks [171]. More recently, transformers have been used for time series forecasting [59, 208, 236] due to their ability to learn long-range dependencies. Although these networks have high predictive capabilities, they are inherently non-Markovian techniques. The state space of dissipative dynamical systems may or may not be Markovian; however, if one assumes that there is a coordinate representation where they are, then a Markovian method for time evolution would suffice. Moreover, reservoir networks often have a latent space with dimensionality higher than the original state space, which makes the models not reduced-order.

One dynamical systems-based approach to develop continuous-time models for forecasting is the sparse identification of nonlinear dynamics (SINDy) method [12]. SINDy learns the vector field f in Equation 1.18 as a sparse linear combination of functions derived from a user-defined dictionary of candidate functions. A limitation of SINDy is that specifying a suitable dictionary (especially for complex systems) can be challenging without domain knowledge of the data. Moreover, SINDy generally requires time-derivative data of the state for training, which are either provided or estimated through finite differences. In the latter case, state data would need to be sufficiently small and noise-free; however, variations on SINDY [220, 237] have been shown to relax this requirement. SINDy has also been combined with autoencoders in [20], which allows one to learn the vector field on a reduced-dimensional

latent space.

Another approach for time evolution is with the Koopman operator [15], which is a linear, infinite-dimensional operator for evolving observable functions of a nonlinear, finite dimensional dynamical system. The Koopman operator has ties to dynamic mode decomposition (DMD) [197, 222], which is a data-driven method to find a best-fit linear time evolution operator for the state; however, for general nonlinear problems, DMD provides a poor approximation of the Koopman operator. In extended DMD (EDMD) [240], the observable functions are learned from a prespecified dictionary of observables on the data, which allows for better generalization to nonlinear systems. In EDMD-DL (dictionary learning) [122], the dictionary is learned with a neural network. The advantage of these Koopman-based methods is that the dynamics are evolved with linear functions, which can help with control; however, linear finite-dimensional systems, no matter the dimension, cannot capture the chaotic behavior that is of our interest here.

1.4.2 Neural ordinary differential equations

The last method for data-driven forecasting that we discuss here is neural ordinary differential equations (NODEs) [22]. The NODE is the basis for the work in Chapter 3, and more details will be provided there. Given a state u at time t_i, the time prediction at time τ in the future is given by

$$\hat{u}(t_i + \tau) = u(t_i) + \int_{t_i}^{t_i+\tau} f_d(u(t'); \theta_g)\, \mathrm{d}t', \tag{1.19}$$

where $\hat{u}(t_i + \tau)$ is the NODE prediction of the state at the next sampled time instant, and f_d is the data-driven approximation of f.

The function f_d is typically modeled with a deep feedforward neural network, and its parameters θ_{f_d} are optimized with the loss function

$$\mathcal{L}_{\mathrm{NODE}} = \langle \|\hat{u}(t_i + \tau) - u(t_i + \tau)\|_2^2 \rangle\,. \tag{1.20}$$

The gradient of this loss with respect to the network parameters can be determined either by backpropagating through the ODE solver using automatic differentiation or with a more memory efficient adjoint method [22]. Here, the reconstruction loss for an integration time horizon spanning only a single data spacing interval τ is depicted. While one could consider a multi-step loss with a longer time horizon, doing so typically does not improve the results and only serves to slow down the NODE training. Lastly, unlike discrete-time maps, the NODE learns the continuous-time dynamics. Therefore, one has the option to evaluate a NODE model with an integration time step that is smaller or even larger than the data spacing τ. In recent work [16], it has been shown that EDMD-DL with a projection step onto the state is equivalent to a nonlinear discrete-time flow map (e.g., NODE) in state space—with this projection step, the evolution functions are no longer linear and the main advantage of Koopman-based methods is lost. This equivalence between a NODE and EDMD-DL with projection shown in [16] also demonstrates that a NODE can be structured as a method that learns a set of high-dimensional features (dictionary elements), and then constructs the vector field as a linear combination of them. An advantage of the NODE over SINDy is that instead of prescribing a dictionary for the RHS, the NODE rather learns one. Additionally, unlike SINDy, the data points of the state used for the NODE training can be spaced far apart in time and non-uniformly.

1.4.3 Learning PDE solutions

The data-driven forecasting methods discussed so far have been based on a dynamical systems approach. We now discuss methods that take a PDE approach, which aim to output the solution of a PDE at a given time and spatial coordinate. These methods may or may not be supplemented by the physics (i.e., the governing equations).

A widely used class of methods for combining physics and machine learning for learning solutions to PDEs is physics-informed neural networks (PINNs) [101, 188], which have been applied in areas such as fluid mechanics [189], nano-optics [25], and solid mechanics [70].

PINNs use a physics-based loss term to learn PDE solutions that are physically consistent. The physics-based loss term penalizes the residual of the PDE at collocation points, which acts as a soft constraint to satisfy the PDE. While PINNs may also include a supervised loss on the initial and boundary conditions from data measurements, the solution is mostly influenced by the purely physics-based loss—therefore, PINNs are often regarded as ODE or PDE solvers that are not data-driven. That said, the use of a PINN-like loss term in data-driven frameworks is quite common. One example is in physics-informed deep learning [187], where the data-driven loss is placed on points outside of the initial and boundary conditions.

Since a soft constraint is used in PINNs, conservation laws are not exactly respected. To satisfy specific conservation laws exactly, one can modify the architecture of the neural network; some examples of enforcing these hard constraints include Hamiltonian structure [62] and the divergence-free condition [163]. In [11], architecture-constrained neural networks are used to enforce a set of hard constraints more generally by first linearizing them (which may introduce approximation errors) and including them as fixed layers in the neural network.

Compared to traditional numerical methods, PINNs are effective in solving ill-posed and inverse problems, and they are mesh-free (though collocation points could be considered as a mesh). Recently, [100] demonstrated a machine learning-free method for solving inverse problems that is faster and more accurate than PINNs. One disadvantage of PINNs is that they are trained on a single set of initial conditions, boundary conditions, and PDE parameters—retraining is necessary if any of these change. (Though this limitation is true for most neural network-based methods.) Furthermore, PINNs typically do not have a natural way to generate extrapolated PDE solutions outside the time horizon of the collocation points. This ability to easily extrapolate in time is of interest to us when developing physics-informed ROMs (which we will discuss later).

Another class of neural network-based methods for learning PDE solutions is neural operators (NOs) [125, 142]. Unlike PINNs, NOs are data-driven (with the data generally

coming from a PDE) and do not include physics-based information at collocation points (at least in the original formulation). NOs aim to learn the PDE solution operator parametrized to a family of PDE parameters and initial and boundary conditions, which addresses a key weakness of PINNs. Similar to the universal function approximator property of neural networks [82], there is a universal approximation theorem for operators [24]. While neural networks learn a function to map an independent variable to a dependent variable, NOs learn an operator that maps functions to functions.

The first NO, called DeepONet [142], maps the initial condition or forcing functions to the PDE solutions at desired spatio-temporal coordinates. DeepONet has a dual architecture that contains a branch net that extracts features from the input functions and a trunk net that extracts features from the solution coordinates. In the case of kernel integral-based NOs [126], the features are extracted from the input functions integrated with a kernel function, which enables the NO to be discretization-invariant (i.e., the input functions can be discretized on different meshes). In the Fourier NO (FNO) method [125], the kernel function is restricted to a convolution, thereby allowing faster evaluations of the integral using Fourier transforms. As a result of the FNO learning features based on the larger wavenumber information of the data, it was shown to be capable of zero-shot super-resolution of 2D turbulence data; however, obtaining this resolution-invariant property of the FNO may be more difficult for datasets containing energy spectra that are not as broadband. An extension of DeepONet, called L-DeepONet [114], applies DeepONet to a latent space learned by an autoencoder. Work has also been done on physics-informed NOs [127, 234], which include a PINN-like loss term on the PDE residual.

As a result of being trained on input functions that span a certain range of space-time coordinates, NOs (like PINNs) cannot produce accurate solutions at extrapolated time instances without special modifications to the architecture. To allow extrapolation in time, [125] considered a variant of FNO that processes the input functions only at spatial coordinates while the time component was accounted for with an RNN architecture; however,

it was stated in [125] that the overall network became less expressive and more difficult to train when using this RNN structure.

In general, the main selling point for PINNs and NOs is that they offer a faster alternative to traditional PDE solvers. While PINNs are generally regarded to be slower than traditional solvers [63], NOs are often purported to be faster; however, the systematic review conducted by [151] suggests that most NOs (at least in their current state) are still slower than traditional solvers. In this thesis, we opt to take an approach more grounded in dynamical systems for developing data-driven reduced-order models.

1.5 Outline of this work

As stated in the beginning of this chapter, this thesis is divided into two goals: 1) characterizing the underlying physical structures embedded in wall-turbulence, and 2) developing an efficient and accurate time evolution model that can ultimately be applied to predicting wall-turbulence. We realize the first goal by developing a modal basis that satisfies the properties of being learned from data, spatially-localized, and multiscale. In Chapter 2, we develop such a basis by combining the energetic optimality of POD with traditional wavelet analysis to form a data-driven wavelet decomposition (DDWD) [50]. We adapt and use DDWD to provide unambiguous evidence that one-dimensional (1D) signals of an experimentally-measured turbulent pipe flow contain coherent structures (represented by the discovered wavelets) that are self-similar across a wide range of scales—this result corroborates the AEH.

Towards the second goal, we develop a hybrid ROM that leverages physics-based information to improve on a data-only ROM consisting of an autoencoder and a NODE. In Chapter 3, we apply the hybrid ROM to a chaotic dynamical system that is simpler than the NSE but still exhibits spatiotemporal chaos. We show the hybrid ROM vastly outperforms the data-only ROM in timeseries predictions. The hybrid ROM developed in Chapter 3 serves as a proof-of-concept that physics and data can be blended together to synergistically model

complex, chaotic systems.

The remaining chapters serve as supports for the two goals mentioned before. In Chapter 4, we explore field reconstruction using autoencoders in the case of sparse sensors—in this realistic scenario, one only has partial access to the field of interest and needs to reconstruct the entire field (e.g., reconstructing atmospheric data at all spatial positions from only a sparse number of available sensors measuring quantities such as velocity or temperature). We demonstrate that combining time delay embeddings with attention mechanisms across space-time encodings allows superior field reconstruction of dissipative dynamical systems. Lastly, in Chapter 5, we further adapt and extend DDWD for extracting multiscale structures in three-dimensional (3D) turbulent channel flow. We discover a 3D orthogonal modal basis that is self-similar in all three spatial directions for a wide range of scales.

We conclude this thesis in Chapter 6 with a brief summary of each chapter before discussing future work. First, we provide preliminary results on using convolutional autoencoders to discover the manifold coordinates of 2D HIT. We then show guidelines on using the 3D modal basis developed in Chapter 5 as an "attached eddy basis", which shows potential to vastly reduce the number of degrees of freedom for modeling Reynolds stresses and instantaneous velocity fields, as well as obtaining a picture of a representative attached eddy's appearance. Finally, we discuss approaches for merging both aspects of this thesis (wavelet-based representation of physical structures, and hybrid ROM for forecasting) into a cohesive ROM for wall-turbulence that is both structural- and physics-informed.

2

Characterizing 1D structures in turbulent pipe flow [1]

This study aims to extract and characterize structures in fully developed pipe flow at a friction Reynolds number of $\mathrm{Re}_\tau = 12\,400$. To do so, we employ data-driven wavelet decomposition (DDWD) [50], a method that combines features of proper orthogonal decomposition and wavelet analysis in order to extract energetic and spatially localized structures from data. We apply DDWD to streamwise velocity signals measured separately via a thermal anemometer at 40 wall-normal positions. The resulting localized velocity structures, which we interpret as being reflective of underlying eddies, are self-similar across streamwise extents of 40 wall units to one pipe radius, and across wall-normal positions from $y^+ = 350$ to $y/R = 1$. Notably, the structures are similar in shape to Meyer wavelets. Projections of the data onto the DDWD wavelet subspaces are found to be self-similar as well, but in Fourier space; the bounds of self-similarity are the same as before, except streamwise self-similarity starts at a larger length scale of 450 wall units. The evidence of self-similarity provided in this study lends further support to Townsend's attached eddy hypothesis, although we note that the self-similar structures are detected beyond the log layer and extend to large length

[1] The text of this chapter is adapted from the publication A. Guo, D. Floryan, and M. D. Graham, Self-similar, spatially localized structures in turbulent pipe flow from a data-driven wavelet decomposition, *Journal of Fluid Mechanics*, 971:A9, 9 2023 [68]

scales.

2.1 Introduction

While statistical approaches dominated the first half of the 20th century in understanding the dynamics of turbulence, the increasing availability of experimental and direct numerical simulation (DNS) data has enabled structure-based approaches to yield useful insights. For example, in the viscous sublayer of wall-bounded flows, turbulent structures coherent in space and time were discovered and characterized using hydrogen bubble visualization [112]. These structures consisted of streaks, which are spanwise-alternating regions of high-speed and low-speed fluid, and quasi-streamwise vortices. From this characterization of the sublayer structures, the self-sustaining process (SSP) was developed and is now a widely accepted model for how these structures sustain themselves [72, 96, 230]. It is evident that the detailed characterization of turbulent structures is a necessary first step toward a mechanistic understanding and modeling of wall-bounded turbulence. Structures in wall-bounded turbulence are reviewed by [193], [1], [94], and [152]. Proper orthogonal decomposition (POD), provides one well-known approach to extracting flows structures from data, but has the drawback that for systems with homogeneous directions, the resulting POD structure in those directions is constrained to be purely sinusoidal, precluding direct identification of structure or localization [80]. In the present work, we extend and apply a recently-developed *data-driven wavelet decomposition (DDWD)* [50], which combines aspects of POD and orthogonal wavelet bases, to experimental observations in high-Reynolds number pipe flow.

With an eye toward a structural model of wall-bounded turbulence, Townsend proposed the attached eddy hypothesis (AEH) for asymptotically high-Reynolds number flows [219]. The AEH supposes that the main, energy-containing motion is made up of contributions from attached eddies. Attached eddies are persistent flow patterns in the log layer whose size scales linearly with their distance from the wall; they span the entire range of scales in

the log layer and—critically—are presumed to be geometrically self-similar. The AEH makes specific predictions for how the normal components of the Reynolds stress vary with distance from the wall, which are borne out by experimental [83, 85, 116, 149, 241] and numerical data [121]. Townsend did not commit himself to a particular form for the attached eddies in his statistical AEH model. Inspired by [112], Townsend considered a physical, double-cone model; unfortunately, it was inconsistent with the AEH, and Townsend's modeling efforts ended there.

Inspired by the flow visualization results of [76], [176] specified a shape for the attached eddies, leading to their attached eddy model (AEM). In addition to making the same predictions as the AEH for the normal components of the Reynolds stress, their AEM also yields the log law for the mean flow and predictions for streamwise spectra. They found that all eddy shapes yield similar first- and second-order statistics, but predictions beyond basic statistics are dependent on the shape of the eddy. A key point is that attached eddies, however they look, are spatially localized. More details on the AEM and subsequent refinements are reviewed by [148].

Partly spurred by the success of the AEH, an increasing abundance of investigations have explored whether self-similar coherent motions are present in wall-bounded turbulence. [217] found that the spanwise length scales of conditionally averaged low-speed structures grew linearly with distance from the wall, as attached eddies are supposed to. [36] found that, on average, the spanwise and streamwise length scales of vortical structures grow roughly linearly with distance from the wall. [141] concluded the same for sweep and ejection structures, and [86] concluded the same for structures of positive or negative velocity fluctuations. These studies show that the length scales of various structures grow, on average, with distance from the wall.

Evocative results were published by [37]. By conditionally averaging their three-dimensional particle image velocimetry measurements based on spanwise swirl strength at a chosen distance from the wall, they extracted a structure with the shape of a horseshoe vortex. Re-

peating the analysis at several distances from the wall, they found a similar shape at each distance whose size grew with distance from the wall. The authors suggested that the shape would make a sensible choice of a 'representative eddy.' Although the analysis was qualitative and performed in the outer layer, it provided exciting hints of the presence of self-similar coherent structures.

Recalling that Townsend's eddies constitute the main *energy*-containing motions, [77] used proper orthogonal decomposition (POD) to explore the existence of self-similar eddies in turbulent pipe flow. These results were also highly evocative as the leading (most energetic) POD modes collapsed on top of each other when scaled by their size. The self-similar POD modes spanned a decade in wall-normal length scales, and the largest structures extended into the outer region.

Although POD provides an approach to extract energetic structures from data, it has the drawback that in homogeneous directions, the POD modes are constrained to be Fourier modes, precluding direct identification of structure or localization [80]. Lumley has espoused the view that such unlocalized structures should not be considered eddies [144, 145, 215]. A well-known formalism incorporating localization is that of wavelets. A wavelet basis consists of elements localized in both space and scale. Traditionally, the basis elements are formed by translations and dilations of a pre-specified vector called the mother wavelet; this construction makes wavelets perfectly self-similar across scales. Due to their localization and the space-scale unfolding they produce, wavelets have found much use in turbulence [3, 43–46, 104, 105, 157, 158, 167, 198, 242]. More information about wavelets is available in the brief review article by [211] and in the books by [35], [160], [147], and [55].

In applications, a drawback of wavelets is that there are many possible choices for the mother wavelet. Naturally, one may wonder which wavelet is best suited for a particular application. [103] considered this question in the context of atmospheric turbulence. Motivated by the AEH, they sought to decompose flow variables into contributions from attached (energy-containing) eddies and detached (negligible-energy) eddies. Katul and Vidakovic

used an entropy-minimizing scheme developed by [30] to select the optimal wavelet from a library of known wavelets. We emphasize that the shapes of the wavelets are still pre-specified and that they are perfectly self-similar across different scales. These two features—common in any traditional wavelet analysis—are undesirable if one's goal is to extract structure from data and to determine the degree of self-similarity of the data.

Motivated by the preceding discussion, [50] developed a technique called data-driven wavelet decomposition (DDWD). DDWD integrates the data- and energy-driven nature of POD with the space and scale localization properties of wavelets without imposing self-similarity across scales. [50] applied DDWD to homogeneous isotropic turbulence and found that the structures in the data (represented by the data-driven wavelets) were self-similar for scales corresponding to the inertial subrange, reminiscent of the Richardson cascade. Our goal in this work is to leverage DDWD as a data-driven basis of energetic, spatially localized, multiscale vectors for characterizing the shape and self-similarity of structures in wall-bounded turbulence. We extend and apply the technique to experimental measurements in high-Reynolds number pipe flow. As we will show, we identify strongly self-similar, spatially localized, energetic structures across a range of scales—candidate eddies. We describe the experimental setup and measurements in Section 2.2. We then briefly review DDWD in Section 2.3 and describe extensions to accommodate the presently analyzed experimental measurements. Section 3.3 describes the structures that we identify, and we end by placing our results in the context of the literature in Section 2.5.

2.2 Experimental setup

The analysis in this work is based on data acquired in the Princeton Superpipe facility, a facility capable of creating fully developed pipe flow at high Reynolds numbers. The data were acquired at a Reynolds number of $Re_D = U_b D/\nu = 608\,000$, with a corresponding friction Reynolds number of $Re_\tau = u_\tau R/\nu = 12\,400$, using a cross-wire thermal anemometry

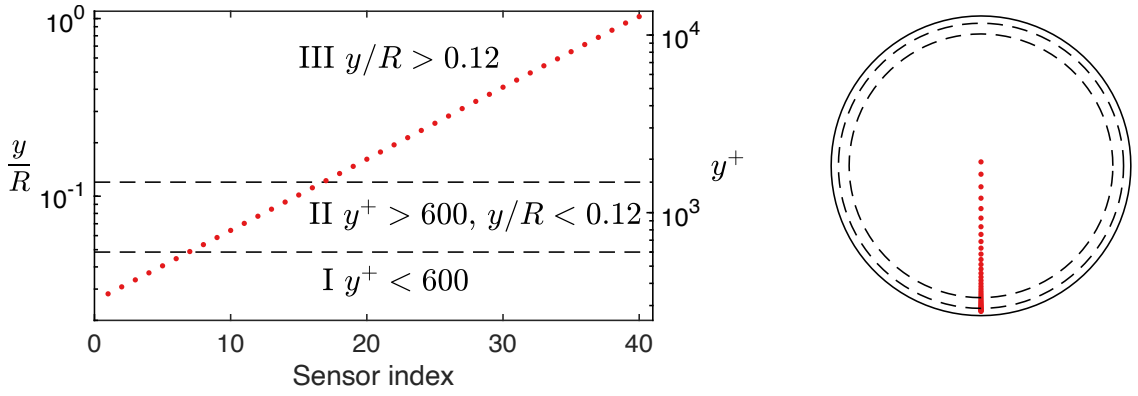


Figure 2.1: Wall-normal positions of the measurement probe.

probe. Here, U_b is the bulk velocity, $D = 2R$ is the pipe diameter, ν is the kinematic viscosity of the fluid, and $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity, where τ_w is the wall shear stress, and ρ is the density of the fluid. The pipe is hydraulically smooth for this operating condition [154]. Details of the facility can be found in [248], and details of the data we use and the measurement probe can be found in [56]; we recapitulate below.

The data consist of long time series of the streamwise velocity component measured at 40 distances from the pipe wall, with two time series taken at each distance. The measurements at different distances from the wall were not simultaneous. The positions y of the measurement probe are shown in Figure 2.1, with a superscript $+$ denoting wall units (non-dimensionalized by ν/u_τ). The measurements were acquired in three regions of the flow: (I) the power-law region, (II) the log-law region, and (III) the wake region. Although the log-law region is classically taken to start at $y^+ = 30$ [183], experiments in the Superpipe suggest that the mean streamwise velocity follows a power law for $50 < y^+ < 600$, with the log law not taking hold until $y^+ = 600$ [153].

Each time series consists of roughly 1.67×10^7 samples, sampled at $F_s = 300$ kHz, and filtered using an analog 8-pole Butterworth filter at 150 kHz. We convert the temporal signals to spatial signals using Taylor's frozen field hypothesis, in which case distances between consecutive samples range from 2.6 wall units at the measurement position closest to the wall to 4.0 wall units at the measurement position furthest from the wall. The measurement

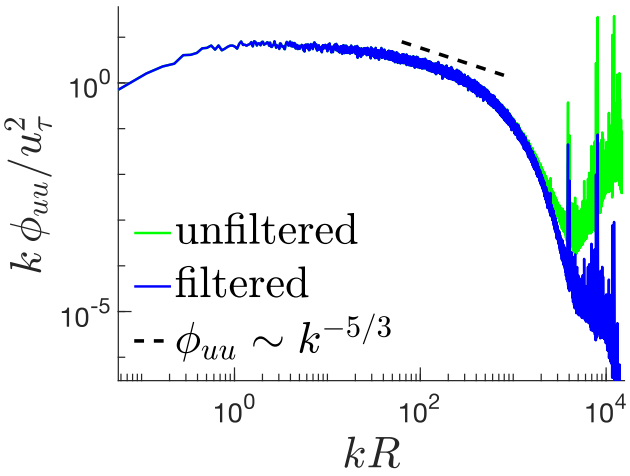


Figure 2.2: Premultiplied PSD of u, measured at $y^+ = 350$, without and with filtering at 25 kHz.

probe had a volume of 42 μm $\times$ 42 μm $\times$ 50 μm, or 8.0 $\times$ 8.0 $\times$ 9.5 wall units^3. The length of the wires in the measurement probe was 60 μm, or 11.4 wall units.

Since the time series were filtered at the Nyquist frequency of 150 kHz during collection, the energy at the largest wavenumbers k corresponding to the measurement noise remains unattenuated. We filter each time series again with a 2-pole Butterworth filter at 25 kHz. Figure 2.2 shows the effect of this filter on the premultiplied power spectral density (PSD) of the signal measured closest to the wall at $y^+ = 350$ (the rest of the time series are affected similarly by the same filter). The PSD, ϕ_{uu}, was estimated using segments that contained 2^{19} elements, overlapped by 50%, and windowed with the Hamming window.

Before describing the method we will use to analyze the structure of the Superpipe data, we estimate the largest coherent structure—or eddy—we can expect to find. The largest eddies in a flow are usually referred to as the very-large-scale motions (VLSMs), and their sizes can be obtained from the PSD of the streamwise velocity. Specifically, the initial peak in the premultiplied PSD is taken to correspond to the VLSM [111]. Selecting this peak can be difficult, however, since producing a good estimate of the PSD involves a tradeoff between using longer data segments (yielding greater resolution in k) and more segments (yielding less variance in the PSD estimate). Even though our time series are rather long, the PSD in Figure 2.2 is the best we can produce without relying on using a moving-average

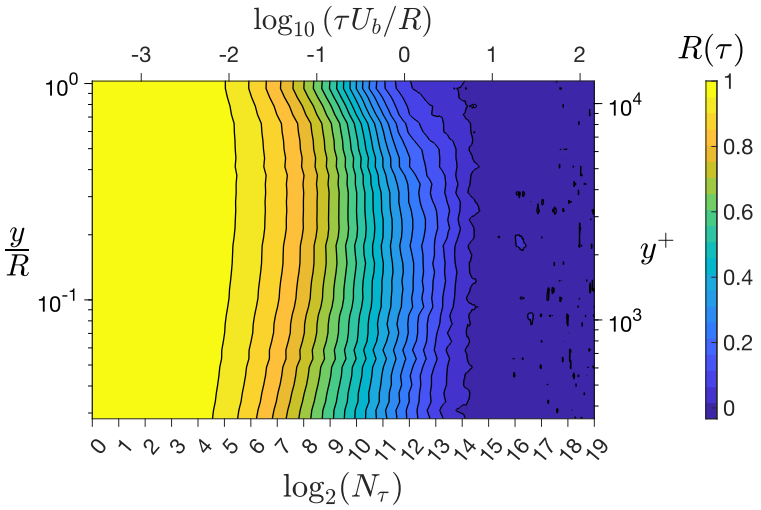


Figure 2.3: Heat map of the time correlation function at all wall-normal positions. $N_\tau = \tau F_s$ is the number of points needed to capture a time lag of τ [s], and $\tau U_b/R$ is the dimensionless eddy turnover time.

filter in k-space. Estimating the location of the initial peak in the premultiplied PSD can be fraught with uncertainty. To avoid the tradeoff involved in estimating PSDs, we opted to use the time correlation function to estimate the largest expected coherent structure. The heatmap of the time correlation function at all 40 wall-normal positions is shown in Figure 2.3, and it suggests all 40 time series are correlated up until 10 eddy turnover times. Since $U_b = 5.4$ m/s and the mean velocity $\overline{U}$ only ranges from 4.2 to 6.4 m/s over the range of wall-normal positions considered here, one can effectively say the structures at all wall-normal positions are correlated in the streamwise direction up until $\mathcal{O}(1R)$ to $\mathcal{O}(10R)$. VLSMs have previously been estimated to be $\mathcal{O}(10R)$ in length [111].

2.3 Data-driven wavelet decomposition

Data-driven wavelet decomposition was introduced by [50], and has its roots in POD and wavelet analysis. First, we provide a conceptual overview of DDWD and its connections to POD and wavelet analysis—for full details, we refer the reader to [50] and to Section 2.6. We then describe the pre-processing steps and modifications needed to analyze the long time series described in Section 2.2.

2.3.1 Overview and connections to POD and wavelet analysis

We consider periodic vectors $z \in \mathbb{R}^N$, with N even, and a dataset $\{z_i \in \mathbb{R}^N\}_{i=1}^M$. Wavelets, POD, and DDWD construct different orthonormal bases for $\mathbb{R}^N$. We interpret the basis elements as structures. Wavelet basis elements are localized structures that are independent of the data being analyzed. POD basis elements, on the other hand, tend to be global structures, and they derive from the data under consideration. Specifically, POD basis elements are constructed via an energy maximization problem and are thus interpreted as energetic structures present in the data. DDWD combines the main features of wavelets and POD, producing basis elements representing energetic, localized structures present in the data.

For our purposes, the most useful way to think about wavelets is by the associated subspaces. A wavelet basis splits $\mathbb{R}^N$ into two orthogonal subspaces of dimension $N/2$: the approximation subspace V_{-1}, and the detail subspace W_{-1}. Subspace V_{-1} is spanned by $N/2$ mutually orthonormal even translates of the father wavelet, ξ_{-1}, and subspace W_{-1} is spanned by $N/2$ mutually orthonormal even translates of the mother wavelet, ψ_{-1}. (By "even", we mean the translates are shifted by an even number of mesh points.) The mother wavelet can be constructed from the father wavelet, and vice versa. Traditionally, projecting a vector in $\mathbb{R}^N$ onto the approximation subspace V_{-1} yields a low-pass filtered version of the vector, and projecting the vector onto the detail subspace W_{-1} yields a high-pass filtered version of the vector (hence the nomenclature of "approximation" and "detail" subspaces). Since V_{-1} and W_{-1} are orthogonal complements, the vector is equal to the superposition of these two projections. The splitting of subspaces is performed recursively on the approximation subspace, ultimately yielding a decomposition $\mathbb{R}^N = V_{-p} \oplus W_{-p} \oplus \ldots \oplus W_{-1}$ for $N = 2^p$, illustrated in Figure 2.4. One may think of this decomposition as the direct sum of a very coarse approximation subspace (V_{-p}) and progressively finer detail subspaces (W_{-p} to W_{-1}). At each stage l, V_{-l} and W_{-l} are respectively spanned by $N/2^l$ mutually orthonormal translates of ξ_{-l} and ψ_{-l}. In a traditional wavelet basis, one specifies the mother wavelet,

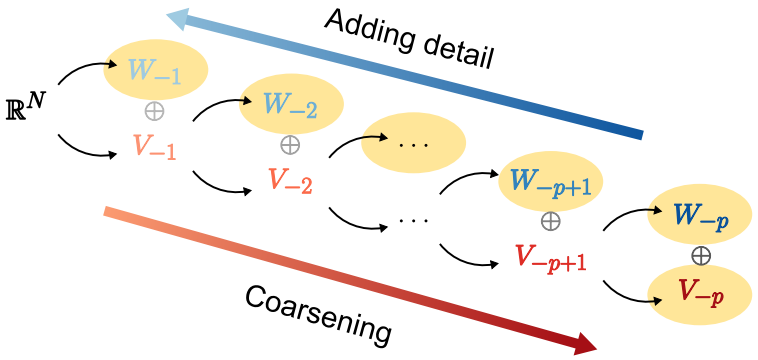


Figure 2.4: Subspaces formed by wavelets periodic in $\mathbb{R}^N$. The full space $\mathbb{R}^N$ is composed of the highlighted subspaces.

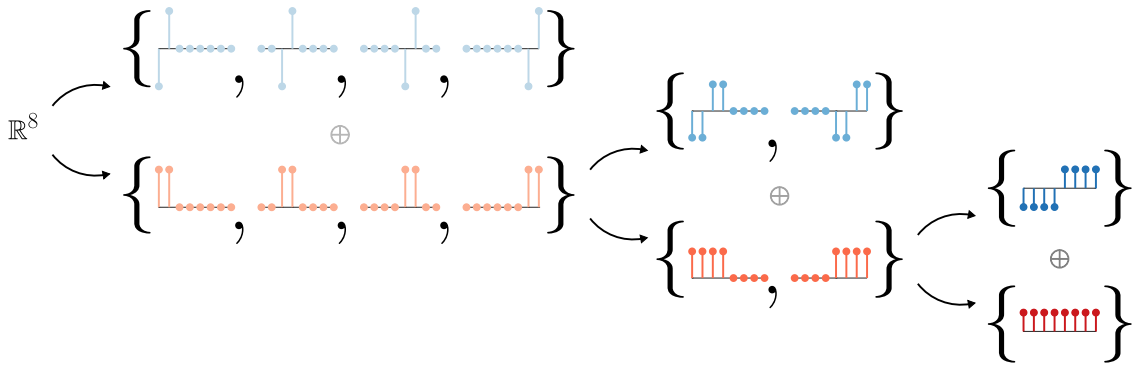


Figure 2.5: Haar wavelets on $\mathbb{R}^8$, following the style of Figure 2.4. The Haar basis for each subspace is shown.

ψ_{-1}, and all other basis elements derive from the mother wavelet. An example—the Haar wavelet basis—is shown in Figure 2.5. Note two salient features of wavelet bases: spatial localization of the basis elements, and self-similarity across stages (the wavelets at coarser stages are essentially simple dilations of the mother wavelet).

POD can be considered in a similar light as wavelet analysis. Letting $\{u_j\}_{j=1}^N$ denote the ordered set of POD basis elements, we have the following decomposition: $\mathbb{R}^N = \text{span}(u_1) \oplus \ldots \oplus \text{span}(u_N)$. That is, POD decomposes $\mathbb{R}^N$ into a direct sum of one-dimensional subspaces. Whereas wavelet subspaces are ordered by scale, POD subspaces are ordered by energy content; whereas wavelet subspaces are determined by the choice of the mother wavelet, POD subspaces are determined from data.

DDWD combines POD and wavelet analysis. Namely, it maintains the subspace structure

of a wavelet basis (shown in Figure 2.4, where each subspace is spanned by mutually orthonormal translates of a single wavelet), but the wavelet at each stage is determined from data by an energy maximization principle. Letting $E(\cdot)$ denote the energy of a dataset contained in a subspace, orthogonality of the subspaces implies that $E(V_{-l}) = E(V_{-l-1}) + E(W_{-l-1})$. That is, whenever a subspace is split, the energy contained in it is also split. Motivated by the observation that large-scale (or coarse) structures tend to be the most energetic ones in fluid flows, 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. An analogous approach is to progressively minimize the energy in the detail subspaces. The design of the subspaces can be framed as an optimization problem. Consider an ensemble of M data vectors forming the dataset $Z = [z_1\ z_2\ \dots\ z_M] \in \mathbb{R}^{N\times M}$. The mathematical formulation for finding the first-stage DDWD wavelet is

$$\begin{aligned} \min_{\psi_{-1}} \quad & \frac{1}{\|Z\|_F^2} \underbrace{\sum_{m=0}^{N/2-1} \|Z^T R^{2m}(\psi_{-1})\|^2}_{ME_1} + \lambda^2 \mathrm{Var}(\psi_{-1}) \\ \text{s.t.} \quad & \psi_{-1}^T R^{2m}(\psi_{-1}) = \delta_{m0}, \quad m = 0, \dots, N/2-1, \end{aligned} \tag{2.1}$$

where R^q is the linear operator that circularly shifts its input by q (e.g., for $\psi_{-1} = [a, b, c, d]^T$, $R^1(\psi_{-1}) = [d, a, b, c]^T$). Above, E_1 denotes the mean energy of the data vectors projected onto W_{-1}, and F denotes the Frobenius norm. The first term in (2.1) is the fraction of the dataset's energy in the finest detail subspace (which typically contains the smallest scale). The circular variance of the wavelet, which is bounded between 0 and 1, is penalized in the second term to encourage the wavelet to be spatially localized. The results are robust to the strength of the penalty parameter λ^2, with the penalty term hardly affecting the wavelets' ability to capture energy [50]. The reason is that the penalty term manipulates the phases of each Fourier component of a wavelet to achieve spatial localization while essentially leaving the magnitudes untouched. Moreover, we find that the shape of the wavelet found by DDWD

is generally robust to the choice of λ^2 across several orders of magnitude (see Section 2.6.2). For the results in this work, we have used $\lambda^2 = 10^{-4}$.

The wavelets at later stages are found recursively. To find ψ_{-2}, we project the dataset onto V_{-1}, replace N by $N/2$ and R^{2m} by R^{4m} in (2.1), and decrease λ^2 by a factor of 4 [50]. In the end, DDWD produces an energetic hierarchy of spatially localized orthonormal structures.

We emphasize two differences between DDWD and traditional wavelet analysis. The first is that the concept of energy is built into DDWD. Whereas the stage of the decomposition induced by a traditional wavelet basis is synonymous with the scale of the wavelets, this is not necessarily the case for a basis produced by DDWD since energy is also taken into account. Scale and energy are often related in fluid flows, however, in which case the energetic hierarchy of structures found by DDWD will also be a hierarchy of scales.

The second difference is that self-similarity across stages is not built into DDWD. It bears repeating that only energy is taken into consideration when finding structures at each stage of the decomposition, and a separate optimization problem is solved at each stage. For data that are not self-similar, neither are the basis elements produced by DDWD. The converse is also true: self-similarity in the data manifests itself in self-similar basis elements. In [50], DDWD applied to data from homogeneous isotropic turbulence produced self-similar basis elements in the range of scales corresponding to the inertial subrange, while when it was applied to white noise and data generated by the Kuramoto–Sivashinsky equation, the basis elements were not self-similar. Determining the degree of self-similarity of structures in the Superpipe data will feature prominently later in this work.

2.3.2 Pre-processing and modifications for long time series

Like POD, DDWD uses an ensemble of vectors (the data). Here, however, we have two long time series of data (for each wall-normal position, which will be analyzed separately). We form an ensemble by breaking the two time series into many shorter segments, as is done

when estimating the PSD. Doing so introduces a number of issues that must be addressed: how long should the shorter segments be; how can border effects be minimized; and how can we efficiently capture very small and very large structures?

Data segmentation

We first address how to segment a long time series into an ensemble of shorter segments of length $N = 2^p$. The segments need to be long enough to capture the length of the largest coherent structure—or eddy—in the flow. Figure 2.3 suggests we need $p \geq 15$ to capture the largest correlated structure; however, the last 2–3 stages of any wavelet basis contain wavelets that are not spatially localized. We set $p = 18$ to ensure the largest spatially localized wavelet represents the largest correlated/coherent structure.

Border effects

Following the classical discrete wavelet transform, DDWD implicitly assumes that signals are periodic. Of course, this is not true for our segmented data. If a segment of data is treated as periodic, there will generally be a discontinuity when the end of the segment is wrapped back around to the beginning. A wavelet decomposition of a signal with a discontinuity generally has high energy in the small-scale wavelets localized around the discontinuity. This border effect could greatly distort the data-driven wavelets found by DDWD.

To circumvent distortions due to border effects, we turn to PSD estimators. PSD estimators based on the discrete Fourier transform also implicitly assume periodic signals, so they are also subject to border effects. To minimize border effects, estimators window the data vectors before taking discrete Fourier transforms of them [184]. Windowing data consists of multiplying the data by a function w that rises from zero to a peak of one and then falls back to zero. We do the same, windowing our data vectors with a 50% Tukey window before performing DDWD on them. [157] similarly windowed his non-periodic data before performing a wavelet analysis. We have tested this procedure on windowed and unwindowed

periodic data, finding that windowing does not affect the shapes of the data-driven wavelets. The type of window used also does not have a significant impact. We double the number of data vectors by overlapping the segments by 50%; this is a common practice to decrease variance when estimating spectra [184] and doing so allows almost every part of the signal to appear in our dataset unattenuated by the 50% Tukey window.

Since windowing data attenuates the signal, PSD estimators compensate for this attenuation in order to calculate accurate estimates of the energy contained within each Fourier mode. For a window $w \in \mathbb{R}^N$, the attenuation caused by the windowing is counteracted by dividing the energy in each Fourier mode by $w^T w/N$, which is bounded between zero and one [184]. Each DDWD subspace contains basis elements that (as a whole) span the entire spatial domain, just like individual Fourier modes. Consequently, we can account for the attenuation from the windowing by dividing the energy in Z and each DDWD subspace by $w^T w/N$. We account for the segment overlapping, which doubles our number of data vectors, by further dividing the energy by 2. The additional factor arises for the following reason. For the long time series u (before segmenting it), its energy is $u^T u$. After we segment it with 50% overlap, we store the resulting segments in a data matrix Z. Each entry of u appears twice in Z due to the 50% overlap (except entries near the beginning and end of u). Then the energy in Z is $\|Z\|_F^2 \approx 2u^T u$. Note that multiplying the data by a constant does not affect the optimization problem in (2.1) because of how the first term in the objective function is normalized. Hereafter, l will denote the stage and E_l will denote the stage-l energy with the factor of $2w^T w/N$ accounted for.

Capturing disparate scales

An important issue is how to efficiently capture very small and very large scales. The very smallest scales in the data will be of a size on the order of one element (2.6 to 4.0 viscous lengths, depending on the wall-normal position) in our data segments, and the very largest scales will be of a size on the order of 2^{18} elements ($55R$ to $85R$). The constrained optimiza-

tion problem in (2.1) is converted to an unconstrained optimization problem (see [50]) and solved using the BFGS algorithm. The optimization problem scales poorly since evaluating the converted objective function once has computational complexity $\mathcal{O}(N^2 \log_2 N + N^2 M)$ and the number of iterations it takes the BFGS algorithm to converge grows with N (although the rate of growth is unclear). This makes the problem intractable for data vectors of length 2^{18}.

We solve this problem by exploiting the iterative nature of a wavelet decomposition and the spatial localization of data-driven wavelets. In the first stage of DDWD, we find the lowest-energy (typically finest-scale) structures. Since these first-stage structures tend to be highly localized, we do not need long data vectors to capture them. Thus, we can perform DDWD on an ensemble of short vectors of length $N_s \ll N$ to capture the smallest-scale structures.

The result of the first stage of DDWD will be wavelets of length N_s, which form part of a basis for $\mathbb{R}^{N_s}$. Ultimately, we need to work our way up to full-length vectors of length N since we seek a basis for $\mathbb{R}^N$. Taking advantage of the spatial localization of the wavelets, we simply pad them with zeros until their length is N. With the full-length first-stage wavelets in hand, we use them to coarsen the full-length data vectors by projecting the data onto the approximation subspace V_{-1} (see Figure 2.4). The projected data have length $N/2$ and are simply coarsened versions of the original data. We proceed iteratively, performing one stage of DDWD at a time on shortened segments of the coarsened data. The technical details are in Section 2.6.1.

Note that once we have our data-driven wavelets, performing a wavelet decomposition with them scales well with the length of the data vectors. It is only the algorithm used to find the data-driven wavelets that scales poorly.

Enforcing zero mean and validating against synthetic data

In traditional wavelet bases, ξ_{-18} from stage $p = 18$ is always a vector of ones and represents the mean component of all elements in $\mathbb{R}^N$. Due to orthogonality, all wavelets ψ_{-l} and their translates must have zero mean. When using DDWD to generate a wavelet basis for a given dataset, one can introduce a hard constraint to enforce zero mean on ψ_{-l} [50]; however, this hard constraint is generally not needed since the energetically minimal ψ_{-l} that DDWD finds also tends to have nearly zero mean. Nonetheless, we choose to enforce zero mean on ψ_{-l} in this work to mitigate a numerical issue that causes the wavelets in stages 13 and up to have artificially high wavenumber content. This numerical issue was found by validating DDWD against synthetic signals composed of known wavelets (see Section 2.6.2 for details). Synthetic data were also used to demonstrate DDWD's ability to uncover many types of wavelets, its robustness to the variance penalty parameter λ^2 across multiple orders of magnitude, the necessity of windowing, and the validity of the modification to DDWD described in Section 2.3.2.

2.4 Results

We apply DDWD to the Superpipe data described in Section 2.2. Starting with the measurements made nearest to the wall ($y^+ = 350$), in Sections 2.4.1 and 2.4.2 we respectively examine the streamwise shape and degree of self-similarity of the localized structures we compute. Extending the analysis to all wall-normal positions, in Section 2.4.3 we discuss the computed structures' streamwise self-similarity and the similarity across different wall-normal positions. In Section 2.4.4 we examine projections of the data onto the computed wavelet subspaces. Lastly, in Section 2.4.5 we derive how spectral power law scalings translate to wavelet space and search for the presence of such scaling laws.

2.4.1 Shape of localized structures at $y^+ = 350$

Figure 2.6a shows the wavelets contained in the 18-stage DDWD basis. For comparison, we have also plotted Meyer wavelets. Each plot corresponds to a different stage of the wavelet decomposition, showing the corresponding wavelet. The plot corresponding to "Stage 18a" shows the basis element for the coarsest approximation subspace, V_{-18}. For the first 15 stages, the range of the horizontal axis is progressively dilated by a factor of two. For ease of comparison, we have shifted the wavelets and, in some cases, reflected them about their horizontal axes (equivalent to multiplying the wavelet coefficients by -1). Although a stage-l wavelet only has translational invariance with respect to shifts by 2^l (i.e., the spacing between its translates), shifting it by any integer is justifiable here since the translational invariance of the data causes the energy contained in the subspace W_{-l} to remain the same no matter the shift value. We observe that the scale and stage of the DWWD wavelets are synonymous. Note that for stages 15 to 18, the wavelets (from all bases) are not spatially localized or self-similar because their scale approaches the length of the data segments.

Figure 2.6b shows the shapes of the stage-8 DDWD wavelet and the Meyer, Fejér-Korovkin 6 (fk6), and Daubechies 2 (db2) wavelets; we chose these wavelets for comparison since they have varying length scales (which we will quantify later) even within the same stage. Figure 2.6c quantifies the similarity of these three known wavelets to the DDWD wavelet with the inner product

$$\alpha_l^{(b_1,b_2)} = \left\langle \psi_{-l}^{(b_1)}, \psi_{-l}^{(b_2)} \right\rangle, \tag{2.2}$$

where $\psi_{-l}^{(b_1)}$ is the stage-l wavelet belonging to basis b_1 and similarly for $\psi_{-l}^{(b_2)}$ (in Figure 2.6c, b_1 is always the DDWD basis while b_2 changes between the three known wavelet bases). The absolute value of the inner product is bounded between 0 and 1. The DDWD wavelets obtained from the Superpipe data are quantitatively most similar to Meyer wavelets. This similarity was also noted when wavelets were computed from data from homogeneous isotropic

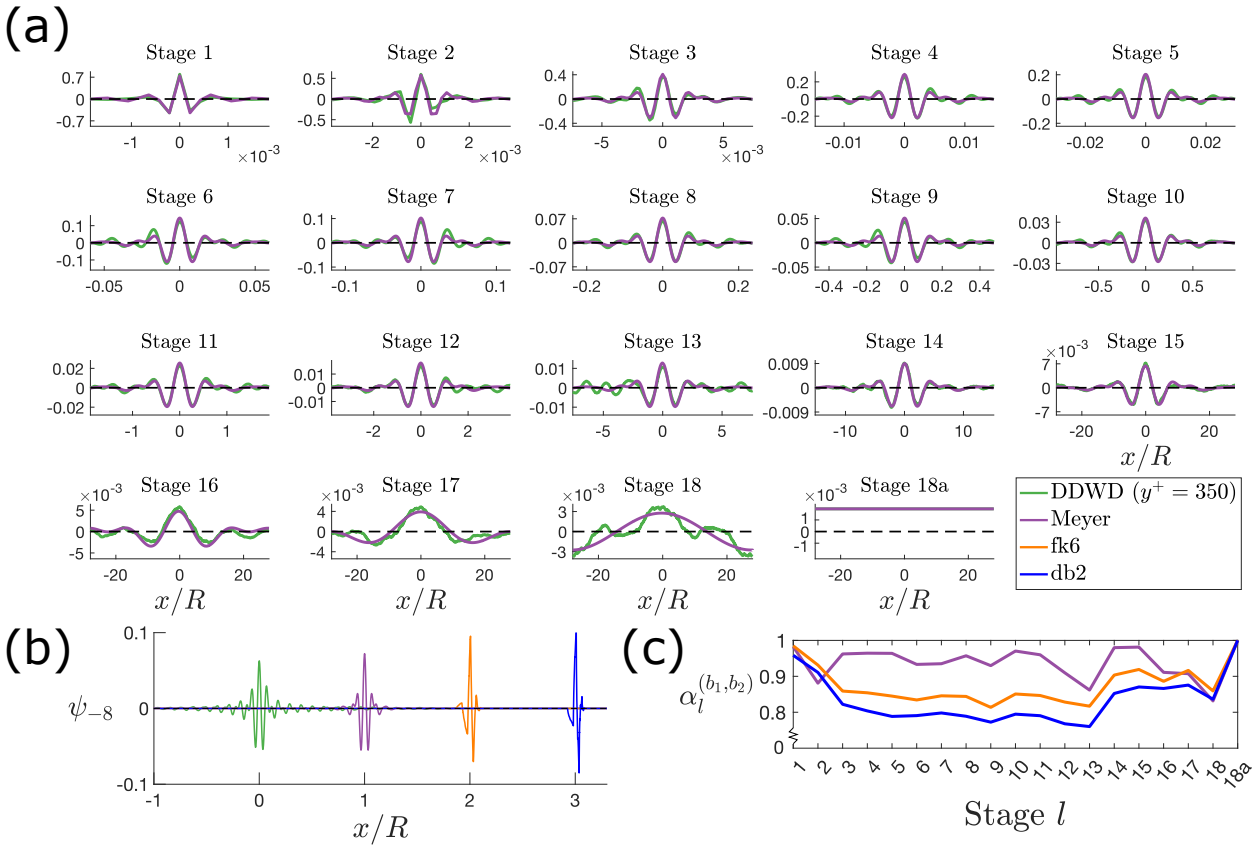


Figure 2.6: Comparing the 18-stage DDWD basis at $y^+ = 350$ to three known wavelet bases. (a) Superimposed wavelets (only DDWD and Meyer shown for clarity). (b) Stage-8 wavelets. (c) Similarity between DDWD basis and known wavelet bases.

turbulence [50]. Two potential reasons that the most energetic wavelets in these two datasets are similar to Meyer wavelets, as opposed to fk6 or db2, is that Meyer wavelets are smoother and have a larger length scale at any given stage. Velocity fields are smooth, so smooth basis functions should give a better representation. As for the larger length scale, that may be a consequence of the combination of smoothness and the orthogonality condition, or may be connected to the fact that there is always a nonlocal aspect to incompressible flows because of the global pressure constraint.

Note that at a given stage, wavelets across different bases can appear very different in shape and width but still have an inner product close to 1 due to having similar Fourier content. In what follows, we consider shapes to be similar when $\alpha_l^{(b_1,b_2)} \gtrsim 0.95$.

2.4.2 Self-similarity of localized structures at $y^+ = 350$

We analyze the similarity of DDWD wavelets across stages, which reflects the streamwise self-similarity of the velocity signal. To do so, we must introduce the scaling operator S. The scaling operator is used when constructing a traditional wavelet basis, with $\psi_{-l} = S\psi_{-(l-1)}$ for all stages. The scaling operator defines our notion of self-similarity: we say that wavelets across a stage are exactly self-similar when $\psi_{-l} = S\psi_{-(l-1)}$, as is the case for traditional wavelets. This operator essentially dilates its input wavelet by a factor of 2, as is evident by comparing the Meyer wavelets from one stage to the next in Figure 2.6a (a precise mathematical definition of S is given in [50]). In Figure 2.7a, the stage-l DDWD wavelets ψ_{-l} are plotted with solid red curves and the dilated, previous-stage wavelets $S\psi_{-(l-1)}$ are plotted with dashed blue curves. The wavelets are apparently strongly self-similar across adjacent stages.

To quantify the degree of self-similarity, we compute the inner product

$$\alpha_{l_a,l_b} = \left\langle \psi_{-l_a}, S^{l_a-l_b}\psi_{-l_b} \right\rangle \ , \ l_a \geq l_b, \tag{2.3}$$

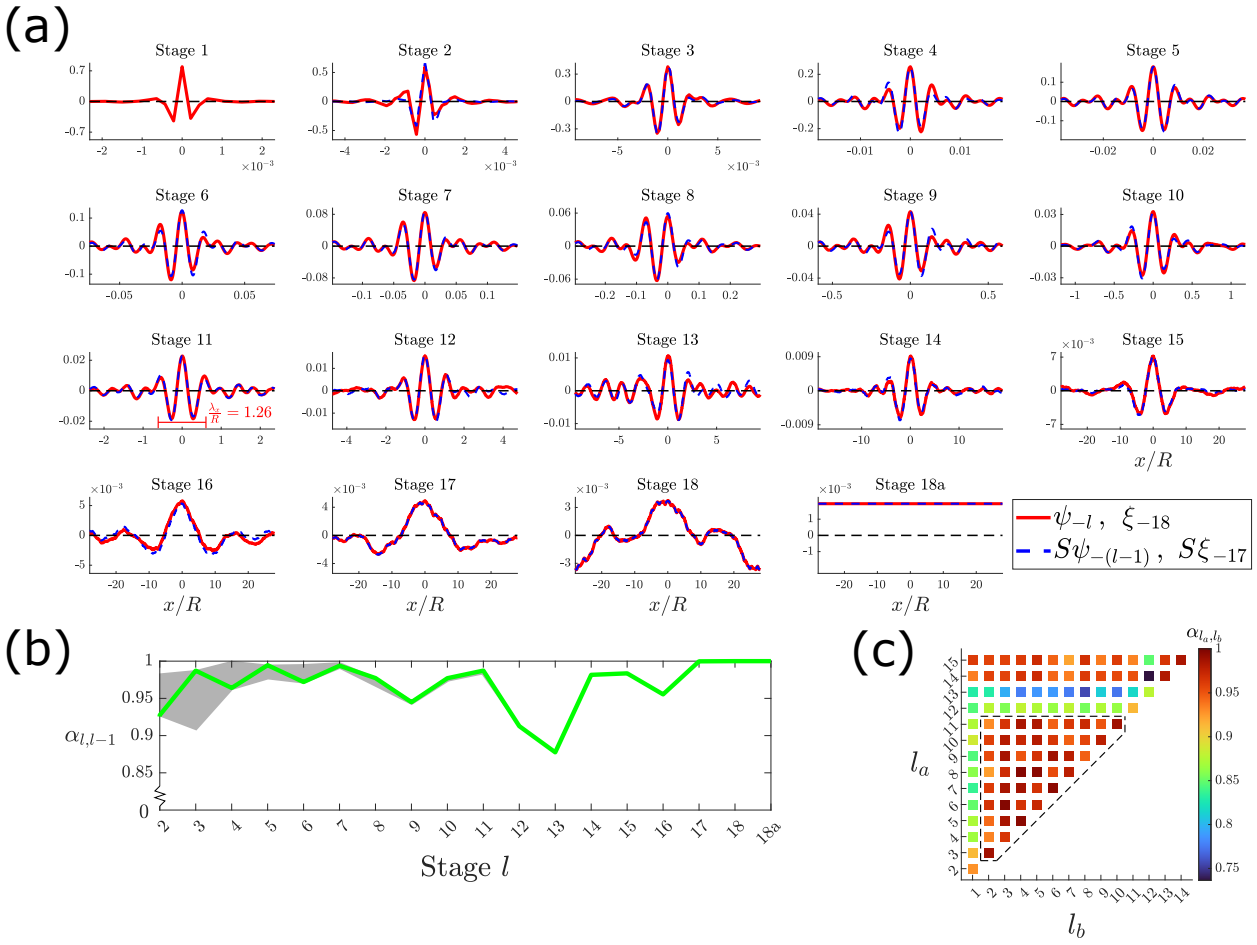


Figure 2.7: (a) 18-stage DDWD basis at $y^+ = 350$; solid red curves are ψ_{-l} (or ξ_{-18} for stage 18a), and dashed blue curves are $S\psi_{-(l-1)}$ (or $S\xi_{-17}$ for stage 18a), where S essentially dilates its input wavelet by a factor of 2. (b) Self-similarity across adjacent stages (i.e., inner product between the normal and dilated wavelets), where the gray region covers the range of values encountered across 10 DDWD initializations. (c) Self-similarity across stages 1 through 15, where the dashed triangle indicates the region of high self-similarity ($\alpha_{l_a,l_b} \gtrsim 0.95$).

analogous to (2.2). Figure 2.7b shows α_{l_a,l_b} for $l_a = l$ and $l_b = l - 1$, which is the inner product between the solid and dashed wavelets appearing in each subplot in Figure 2.7a. A high degree of self-similarity ($\alpha_{l,l-1} \gtrsim 0.95$) across adjacent stages occurs in stages 2 through 11 and then in stages 14 and 15. The drop in self-similarity between stages 11 and 14 is physical—spatially localized structures are no longer self-similar at these stages. However, since the wavelets in the last 2–3 stages (stages 16 and beyond here) of any basis span the whole spatial domain, we disregard them hereafter. Finally, we note (but do not show here) that the drop in self-similarity at stage 11 is robust to decreasing the total number of stages from $p = 18$ to $p = 14$.

Figure 2.7c shows α_{l_a,l_b} for $l_a > l_b, l_a \in [2, 15]$, quantifying the degree of self-similarity across multiple stages. The results in Figure 2.7b appear along the main diagonal in Figure 2.7c. One can see that high self-similarity of ψ_{-l} occurs in stages 2 through 11 across all stages, not just adjacent stages.

Altogether, DDWD reveals that coherent velocity structures (induced by eddies) are highly self-similar in stages 2 through 11. Defining a length scale λ_x for a wavelet as the region centered about the origin that contains 90% of its energy, this range of stages corresponds to $\lambda_x^+ = 20$ through $\lambda_x/R = 1.26$. (Note, our definition for λ_x is justified in Section 2.6.3 and is unrelated to the Fourier wavenumber k.) Not only does DDWD give direct evidence of streamwise self-similarity of localized structures in wall-bounded turbulence, but it also reveals a surprisingly much larger range of self-similarity when compared to many statistical and structural approaches. Section 2.5 includes a more detailed discussion comparing the results obtained using DDWD to those in the literature.

2.4.3 Self-similarity of localized structures in wall-normal direction

Figure 2.8a shows DDWD bases for three wall-normal positions (one for each region defined in Figure 2.1): $y^+ = 350$, $y/R = 0.10$, and $y/R = 0.59$. The wavelets are shown as functions of

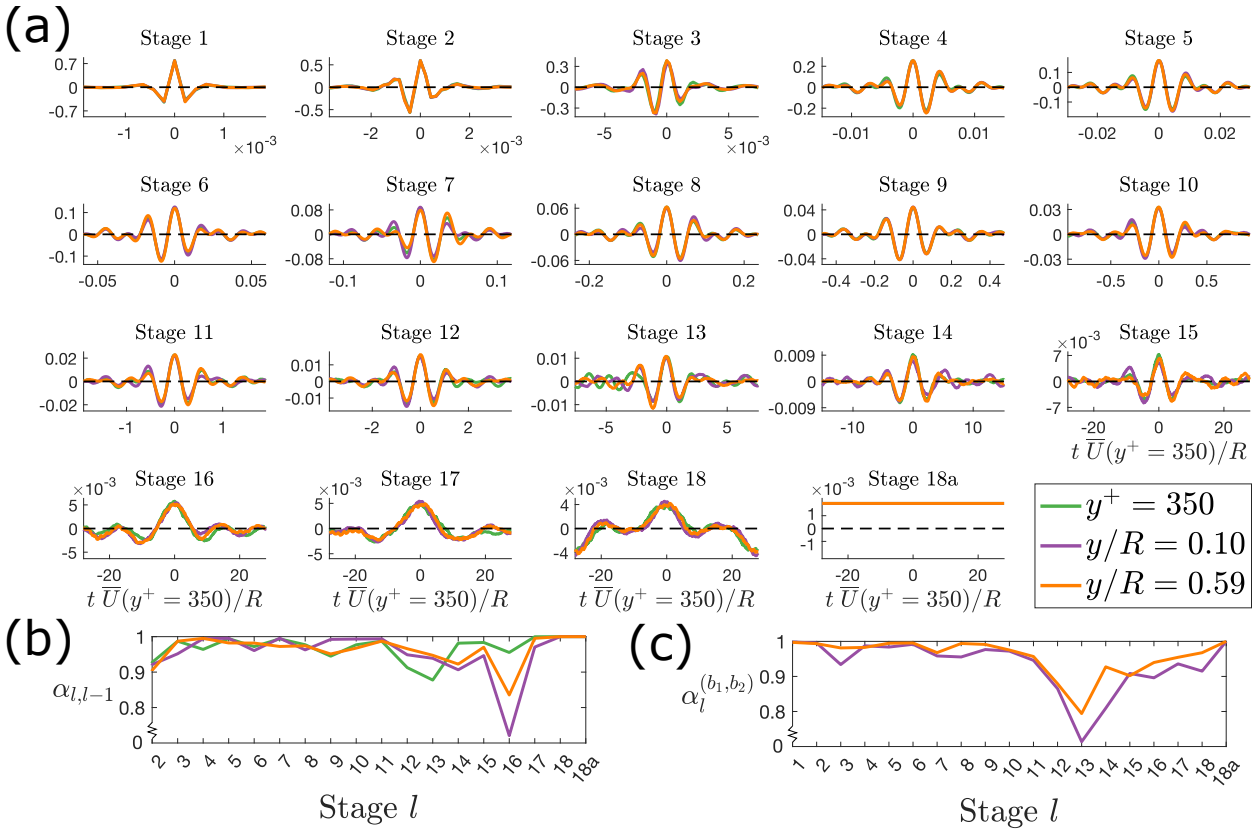


Figure 2.8: (a) DDWD bases at $y^+ = 350$, $y/R = 0.10$, and $y/R = 0.59$. (b) Similarity across adjacent stages for each basis. (c) Similarity across bases, where b_1 is for the signal closest to the wall and b_2 is for the signal either in region II or III (regions defined in Figure 2.1).

time scaled by the eddy turnover time (using the mean streamwise velocity at $y^+ = 350$). We plot in time instead of space to avoid the slight dilation of the wavelets that would occur from using Taylor's hypothesis. For all wall-normal positions, stage is synonymous with scale for the DDWD wavelets. For each wall-normal position, Figure 2.8b quantifies the self-similarity of the wavelets across adjacent stages. Our observation of strong streamwise self-similarity at $y^+ = 350$ extends to the other wall-normal positions in the range of $l \in [2, 11]$. Figure 2.8c quantifies the self-similarity of the DDWD wavelets across wall-normal positions by plotting the inner product in (2.2) computed between the wavelets at $y^+ = 350$ and at the other wall-normal positions. A high degree of wall-normal self-similarity is present for stages $l \in [1, 11]$.

Finally, we quantify the simultaneous self-similarity of the DDWD wavelets in the stream-

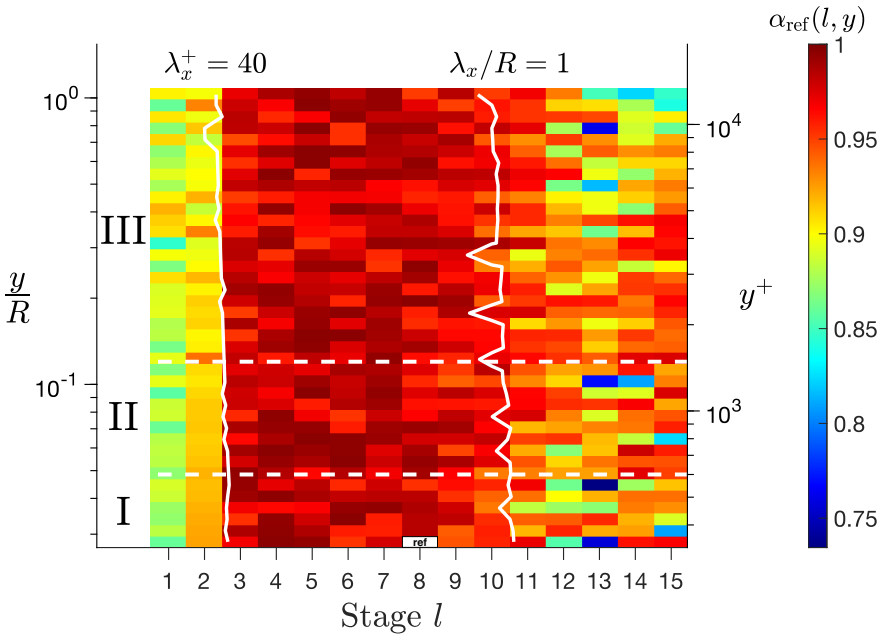


Figure 2.9: Self-similarity between the reference wavelet at $l_{\mathrm{ref}} = 8$ and $y^+_{\mathrm{ref}} = 350$ and all other wavelets. Dashed horizontal lines separate the wall-normal regions defined in Figure 2.1. Solid white lines are isolines of the streamwise length scale λ_x of the wavelets.

wise and wall-normal directions. To do so, we first choose a reference wavelet at stage $l_{\mathrm{ref}} = 8$ and wall-normal position $y^+_{\mathrm{ref}} = 350$. Then, we calculate the inner product between this reference wavelet and all spatially localized wavelets ($l \leq 15$) at all wall-normal positions. We denote this inner product as $\alpha_{\mathrm{ref}}(l, y)$. Note that either the reference wavelet or the wavelet at (l, y) is scaled up to stage $\max(l, l_{\mathrm{ref}})$ using the scaling operator S before computing $\alpha_{\mathrm{ref}}(l, y)$ (which is done in time, not space, to avoid slight dilations resulting from Taylor's hypothesis). Figure 2.9 plots $\alpha_{\mathrm{ref}}(l, y)$. In Section 2.4.2, the DDWD wavelets at $y^+ = 350$ exhibited strong self-similarity for $l \in [2, 11]$; however, based on the criterion $\alpha_{\mathrm{ref}} \gtrsim 0.95$, Figure 2.9 suggests a range of $l \in [3, 10]$ for all wall-normal positions. The slight discrepancy in the range of l is caused by the slight difference between α_{l_a,l_b} and $\alpha_{\mathrm{ref}}(l, y)$; we will use the latter range of $l \in [3, 10]$ henceforth. Figure 2.9 suggests that DDWD wavelets, which form spatially localized, energy-containing motions, are self-similar between streamwise length scales of $\lambda_x^+ = 40$ to $\lambda_x/R = 1$ and between wall-normal length scales of $y^+ = 350$ to $y/R = 1$. Here, the self-similarity of the data has been assessed by the self-similarity of the DDWD

basis. In the next section, a (perhaps) more direct method is explored where we assess the self-similarity of the data projected onto DDWD subspaces of different scales.

2.4.4 Self-similarity of wavelet projections

The stage-l wavelet projection $P_l z_i$ denotes the projection of $z_i \in \mathbb{R}^N$ onto the detail subspace W_{-l} and is given by

$$P_l z_i = \sum_{m=0}^{N/2^l-1} \left\langle R^{2^l m}(\psi_{-l}), z_i \right\rangle R^{2^l m}(\psi_{-l}). \tag{2.4}$$

Henceforth, $P_l u$ denotes the stage-l wavelet projection of the velocity signal u (or rather its segments) onto the DDWD subspaces.

Wavelet projections, just like wavelets themselves, are multiscale. Larger-scale wavelet projections occupy the smaller-k range, and vice versa. For $y^+ = 350$, Figure 2.10 shows the PSDs of u, $P_l u$, and the wavelets themselves, ψ_{-l}. As an aside, Figure 2.10 provides another potential reason why DDWD uncovers Meyer-like wavelets: the lower spatial localization (and thus higher spectral localization) of Meyer-like wavelets allows a narrower range of wavenumbers to be removed from the broadband velocity signal during the energy optimization.

We assess the self-similarity of the wavelet projections in Figure 2.10 by comparing their PSDs. Figure 2.11a shows the PSDs when scaled in k by 2^l in order to account for the dyadic progression of scales; they are also normalized to have unit norm. The self-similarity of $P_l u$ is quantified by γ_{l_a,l_b}, which is the inner product between the scaled and normalized PSDs of scale l_a and l_b; Figure 2.11b plots this inner product. Just as with α, we take $\gamma \gtrsim 0.95$ to indicate a high degree of self-similarity. Unlike DDWD wavelets, the projections are not highly self-similar until a much later stage of $l = 6$.

Similar to the approach used in Section 2.4.3, we calculate the self-similarity of wavelet projections simultaneously in stage and wall-normal position by choosing a reference projection. We choose $l_{\text{ref}} = 8$ at the wall-normal position $y^+_{\text{ref}} = 350$ again, and denote this inner

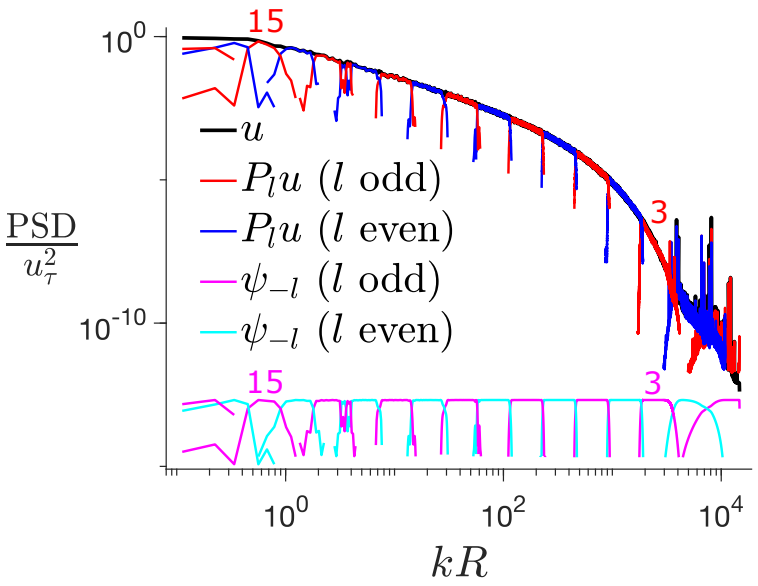


Figure 2.10: PSDs of u, its stage-l DDWD wavelet projections $P_l u$, and the DDWD wavelets ψ_{-l} at $y^+ = 350$. Note that the PSDs of the wavelets are manually shifted to be at the same height.

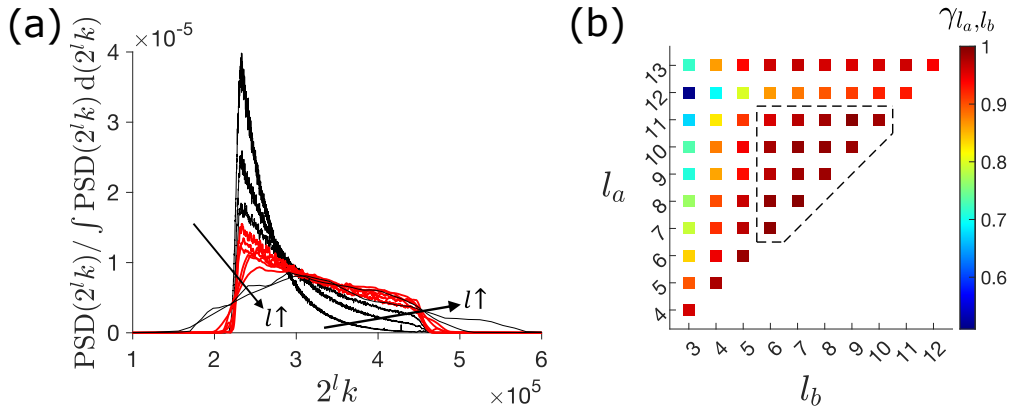


Figure 2.11: (a) Scaled and normalized PSDs of the DDWD wavelet projections in Figure 2.10 for $l \in [3, 13]$. (b) Inner product between the curves in (a), where the dashed triangle indicates the region of high self-similarity ($\gamma_{l_a,l_b} \gtrsim 0.95$). Highly self-similar projections of scales $l \in [6, 11]$ in (a) are colored red based on the triangular region in (b).

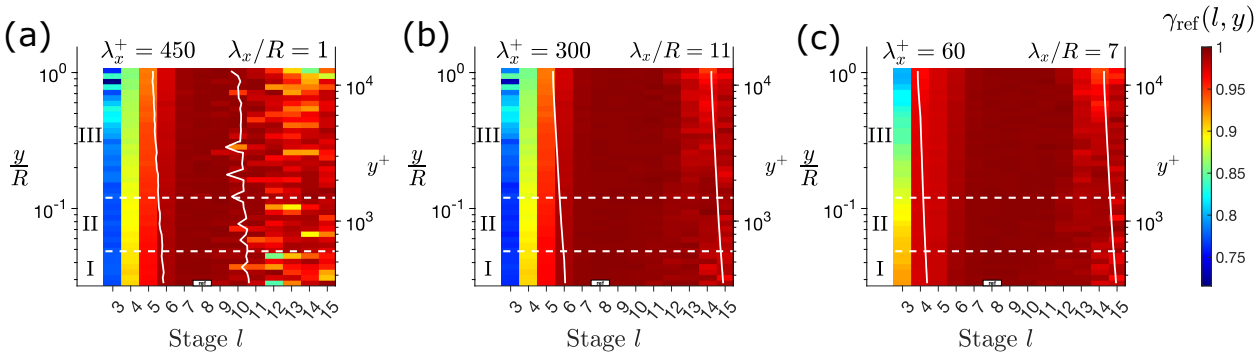


Figure 2.12: Same as Figure 2.9, except with the inner product γ, which quantifies the self-similarity of the PSDs of wavelet projections using the (a) DDWD, (b) Meyer, and (c) db2 basis. Solid white lines are isolines of the streamwise length scale λ_x of the wavelets.

product as $\gamma_{\text{ref}}(l, y)$; note that this inner product is done in frequency, not wavenumber, to avoid slight dilations resulting from Taylor's hypothesis. Figure 2.12 plots $\gamma_{\text{ref}}(l, y)$ for the DDWD, Meyer, and db2 bases. DDWD wavelet projections are self-similar in y from $y^+ = 350$ to $y/R = 1$ and self-similar in x (across stages) up until wavelets of wavelength $\lambda_x/R = 1$, exactly like DDWD wavelets (see Section 2.4.3). The only difference, however, is the lower bound of self-similarity in x—DDWD wavelets are self-similar starting from $\lambda_x^+ = 40$, while the projections are self-similar starting from $\lambda_x^+ = 450$. The reason for the difference is that by comparing the spectra of the projections, we simultaneously account for the shapes of the wavelets and the spectral content of the data, which yields a more stringent criterion for self-similarity. Figures 2.12b and 2.12c show that projecting the data onto traditional wavelet bases (with pre-defined and exactly self-similar shapes across scales) leads to overprediction of γ_{ref} in x by several stages since only spectral content is taken into account: $\gamma_{\text{ref}} \gtrsim 0.95$ for $l \geq 6$ ($\lambda_x^+ \approx 300$) when using Meyer wavelets and for $l \geq 4$ ($\lambda_x^+ \approx 60$) when using db2 wavelets. So even though Meyer wavelets may closely resemble DDWD wavelets (see Figure 2.6), this result shows one advantage to using the DDWD basis rather than a prescribed wavelet basis for performing tasks such as projections.

We now tie the results back to the AEH, which states that eddies in the log layer grow in size with distance from the wall and are self-similar. The energy-optimality principle of DDWD allows the shape of the DDWD wavelets and the DDWD wavelet projections to be

each used as a criterion for measuring the self-similarity of the velocity. Our method defines self-similarity somewhat indirectly—if two quantities (e.g., wavelets) across different stages are strongly similar (i.e., have an inner product value $\gtrsim 0.95$), we say they are self-similar. The critical (and often-made) assumption is that self-similar fluctuating velocity signatures arise from self-similar eddies. In x, DDWD shows that eddies have a high degree of self-similarity from scales of $\lambda_x^+ = 40$ (based on the wavelets) or $\lambda_x^+ = 450$ (based on the wavelet projections) up to $\lambda_x/R = 1$, which is the streamwise extent of LSMs. VLSMs of streamwise extent $\lambda_x/R = \mathcal{O}(10R)$ may also be self-similar, but to a lesser degree. In the wall-normal direction y, our results suggest that eddies are self-similar not just in the log layer but below and well above it.

The self-similarity of the velocity in x may correspond not only with attached eddies, but also with eddies associated with the energy cascade. In the next section, we determine the range of stages that accompany two spectral scaling laws: the $k^{-5/3}$ law related to the energy cascade, and the k^{-1} law related (loosely) to self-similarity of wall-bounded turbulence.

2.4.5 Energy and spectral scaling laws

The log law for the streamwise turbulence intensity, $\overline{uu}$, is a key feature of wall-bounded turbulence that has been widely observed in experiments and simulations [83, 121, 149, 195, 241]. There are three models that have derived this log law. The first is Townsend's AEH [219], which was a statistical model that assumed the contributions to the turbulence intensity from attached eddies were self-similar. The second is the AEM [176], which was a physical model inspired by the AEH that assigns a shape to the attached eddies. The third is the spectral AEM [178], which relied on scaling and dimensional arguments to argue that the PSD of u should exhibit a k^{-1} power law in an intermediate range of wavenumbers; integrating the PSD gives a log law for $\overline{uu}$. So, the spectral AEM shows that a k^{-1} power law in the PSD of u is consistent with the existence of a log law for $\overline{uu}$. Transitively, one could then say that the existence of a k^{-1} power law would be consistent with the existence

of geometrically self-similar eddies proposed by the AEH.

Here, we attempt to search for a k^{-1} scaling in wavelet space instead of k-space. As discussed in Section 2.3.1, the energy contained in a signal is partitioned into the highlighted wavelet subspaces in Figure 2.4. The mean energy of our dataset (segments of u) contained in subspace l is

$$E_l = \frac{1}{M}\sum_{i=1}^{M} \|P_l z_i\|^2 = \frac{1}{M}\sum_{i=1}^{M}\sum_{m=0}^{N/2^l-1} \left|\left\langle R^{2^l m}(\psi_{-l}), z_i \right\rangle\right|^2. \tag{2.5}$$

For a signal of length $N = 2^p$, E_l is defined for stages $l \in [1, p]$ (the detail subspaces) and stage p_a (the final approximation subspace). At stage p_a, ψ_{-l} in (2.5) is replaced by ξ_{-p}. We call the distribution of energy amongst these subspaces the wavelet energy distribution (WED).

The key difference between the PSD and the WED is that the PSD gives the energy per unit wavenumber, while the WED gives the energy contained in a range of wavenumbers. Assuming self-similar wavelets, E_l gives the energy across a range of wavenumbers $\Delta k_l \sim 2^{-l}$ (see Figure 2.10). Therefore a quantity analogous to the PSD, which we call the wavelet power spectrum (WPS), is $\tilde{E}_l = E_l/\Delta k_l \sim 2^l E_l$, as used by [157]. Meneveau's definition of the WPS was carefully normalized to match the energy contained in the continuous signal. The normalization does not change how E_l scales with l, which is our main interest.

We now discuss the equivalence of power laws between the PSD, WED, and WPS. Since the wavelength λ_x of a wavelet is proportional to both 2^l and $1/k$ (k being the central wavenumber of the wavelet), a power law of $\phi_{uu}(k) \sim k^{-\gamma}$ in the PSD will appear either as $\tilde{E}_l \sim (1/\lambda_x)^{-\gamma} \sim (1/2^l)^{-\gamma} \sim 2^{\gamma l}$ or $E_l \sim 2^{(\gamma-1)l}$. The power law scaling for the WED can also be derived by integrating a power law for the PSD across wavenumbers corresponding to adjacent stages:

$$E_l \sim \int_{2^{-l}}^{2^{-(l+1)}} k^{-\gamma}\,\mathrm{d}k = \frac{2^{1-\gamma}-1}{1-\gamma} 2^{(\gamma-1)l}. \tag{2.6}$$

Henceforth, we work with the WED. Therefore $\gamma = 5/3$ and $\gamma = 1$ lead to $E_l \sim 2^{2l/3}$ and

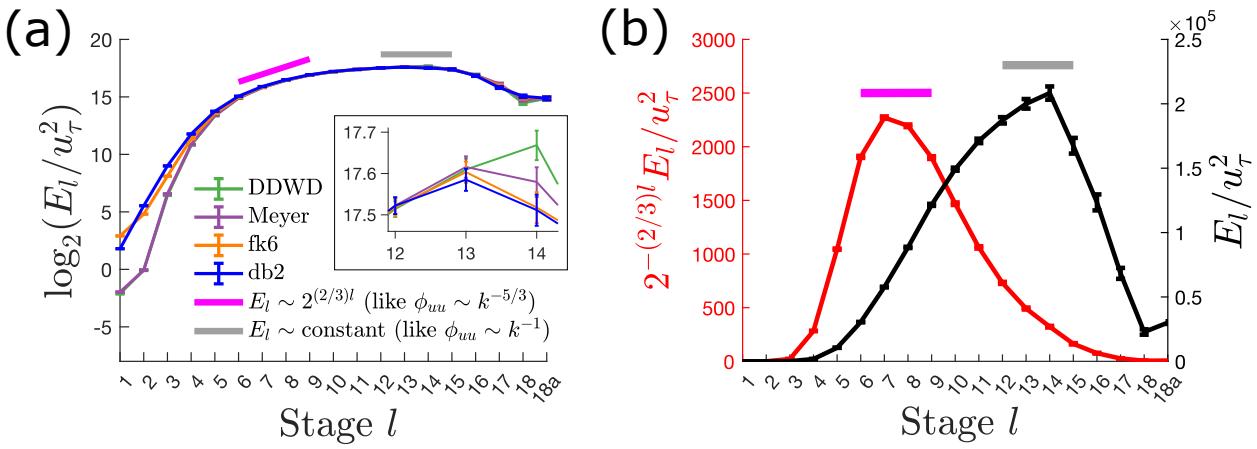


Figure 2.13: (a) Wavelet energy distribution (WED) of the signal measured at $y^+ = 350$ calculated using the DDWD basis and 3 known wavelet bases (the green and purple curves overlap). (b) Two versions of the DDWD WED from (a), one of which is premultiplied. Note, error bars are standard errors.

$E_l \sim 2^0 \sim$ constant, respectively.

As before, we first discuss the WED results for the velocity signal measured closest to the wall. Figure 2.13a shows the WED for this signal when E_l is computed with the DDWD basis and three traditional wavelet bases. As seen in Figure 2.13a and its inset, E_l is lower in the DDWD basis until stage $l = 14$, which is the most energetic (but not the largest) stage. The reader will note that the WED at stage 18a (which corresponds to the squared mean) is non-zero despite u having zero mean. The reason is that, like the PSD, the WED is estimated from an ensemble of segments of u which may themselves have non-zero mean. By appropriately multiplying the WED of the DDWD basis, the two power laws in Figure 2.13a appear as plateau regions in Figure 2.13b. Note that the WED plotted on the right vertical axis in Figure 2.13b is the same as that in Figure 2.13a, except plotted with a linear scale for the vertical axis to better discern whether a plateau region exists. A plateau region corresponding to a $k^{-5/3}$ scaling is not significant, which is common for measurements close to the wall [195, 223]. Nonetheless, one can still say that the inertial subrange occurs around stages 6 through 9, and that the self-similarity results found previously for these stages may also be attributed to the energy cascade and not solely to the energy-containing motions from Townsend's AEH. A plateau region corresponding to a k^{-1} scaling is even less apparent.

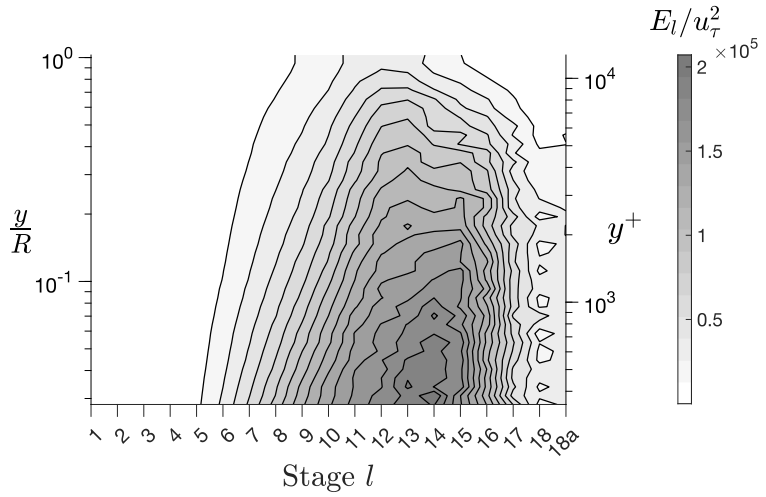


Figure 2.14: (a) Wavelet energy distribution (WED), calculated using the DDWD basis, at all wall-normal positions. This plot is analogous to a premultiplied PSD spectrogram.

Figure 2.14 shows E_l (calculated with the DDWD basis) at all wall-normal positions. This plot is analogous to a premultiplied PSD spectrogram, where a k^{-1} scaling would manifest as a flat region. There is a relatively flat region for $y^+ \lesssim 1000$ and $12 \leq l \leq 15$. In the premultiplied PSD spectrograms of other studies, a flat region appears at a similar location and is referred to as the "outer peak" [83, 84]. In [84], they propose that the outer peak corresponds to the presence of VLSMs, which is consistent with our data since $l \approx 13$ corresponds to $\lambda_x = \mathcal{O}(10R)$ (see Figure 2.7a). The spectral AEM [178] states that the k^{-1} law associated with attached eddies exists at length scales in the overlap of the $\mathcal{O}(y)$ region (stages 6 through 11 here) and the $\mathcal{O}(R)$ region (stages 11 through 15 here)—therefore, although a somewhat flat region appears at $l \approx 13$ in Figure 2.14, it occurs at length scales that are too large to be considered a true k^{-1} law that is associated with attached eddies.

A k^{-1} power law is rarely observed in experiments and simulations [28, 166, 195]. One possible reason for the frequent absence of a k^{-1} power law is that it is only expected to appear when there is enough scale separation in the flow, which occurs at high friction Reynolds numbers. [148] estimate that $\mathrm{Re}_\tau = \mathcal{O}(10^6)$ is required to observe even a small region with k^{-1} scaling, which is not achievable except in the atmospheric boundary layer. Since our Superpipe data has $\mathrm{Re}_\tau = \mathcal{O}(10^4)$, a k^{-1} power law is not expected to appear

clearly. Moreover, [89] showed that even without the presence of a k^{-1} power law, the framework of the spectral AEM can still successfully yield a log law for $\overline{uu}$, which is a key feature of wall-bounded turbulence. Therefore, within the context of the spectral AEM, the existence of a k^{-1} power law may not be a necessary feature of wall-bounded turbulence. As for whether geometric self-similarity is a feature of eddies in wall-bounded turbulence, a consensus has not been reached. In the next section, we compare the self-similarity results found by DDWD to the results of other methods.

2.5 Comparison of ranges of self-similarity

Several turbulent structures have been found and studied in the literature, many of them displaying self-similarity across a range of scales. Here, we compare the ranges of self-similarity for these structures and for the localized structures found herein via DDWD.

Hairpin vortices, first proposed by [216], may be attached eddies. These coherent structures are theorized to be generated in a bottom-up process [1, 217, 257]. At the smallest scale, hairpin vortices of height $\mathcal{O}(100\delta_\nu)$, where δ_ν is a wall unit, are generated in the viscous layer. Then, these hairpin vortices merge with each other in a pairing process to give larger hairpin vortices, which in turn form packets of hairpin vortices (e.g., LSMs and VLSMs) with height $\sim 0.3\delta$ to 1δ, where δ is the turbulent boundary layer (TBL) height. However, this bottom-up generation process is not universally agreed upon. [49] showed that the structure and dynamics of LSMs and VLSMs in the outer region were not affected upon suppressing the generation of hairpin vortices in the viscous layer. Furthermore, [258] used correlation methods to show that the accumulation of streaks in the buffer layer is not necessary for the generation of LSMs. On the other hand, a top-down process was found in which LSMs induced the spanwise drift of streaks.

Another candidate for attached eddies is the structures related to the SSP [72, 231], namely quasi-streamwise vortices and streaks. Although SSP structures are generally con-

sidered to exist only in the buffer layer, [40] found exact coherent states (ECS; see [106] and [61]), corresponding to SSP structures, that exist in the outer region and are self-similar. Further support of SSP structures being self-similar is found in [87] and [33], where overdamped large-eddy simulations (LES) were used to isolate structures at a specific scale. In these simulations, the SSP structures representing attached eddies were found to be self-similar; note that an "attached eddy" in this case is composed of the dynamically interconnected quasi-streamwise vortices and streaks. The largest self-similar attached eddy was found to be comprised of a streaky motion of streamwise extent $15h$ (h is the channel half-height) and height $\sim 0.15h$ that is surrounded by quasi-streamwise vortices of streamwise extent $3 \sim 4h$ and height $0.75h \sim 1h$. This aspect ratio of the self-similar SSP structures is maintained throughout the outer region all the way down to the viscous layer at $\mathcal{O}(10\delta_\nu)$. The largest streak component resembles a VLSM and the largest quasi-streamwise vortices component resembles LSMs. Furthermore, the overdamped LES results imply that turbulence can be sustained at each individual scale without any forcing from smaller scales, thus lending more support against the bottom-up generation of vortices discussed previously.

An operator-based method called resolvent analysis was shown to produce reconstructed velocity fields containing self-similar, wall-attached structures [88, 152, 162]. Resolvent analysis expresses the nonlinear term in the NSE as an input forcing to the resolvent (linearized NSE) to obtain the output velocity response. Results show that the velocity response modes are self-similar, and the reconstructed velocity field contains self-similar structures that are attached to the wall [152]. These structures are confined to the log region and interestingly show a quadratic scaling of streamwise extent with distance from the wall instead of a linear scaling as proposed in the AEH.

A method to isolate coherent structures that are physically attached to the wall is spectral coherence analysis (SCA) [4, 6]. SCA uses the magnitude of the cross-spectrum of a two-point streamwise fluctuating velocity signal (one measured at the wall, and one measured in the log layer) to measure the coherence of structures with the wall at each scale $1/k$. By

arguing that a self-similar hierarchy of attached structures manifests in a certain pattern in the cross-spectrogram, it was shown that a TBL flow contained self-similar attached structures for heights $80\delta_\nu$ to 0.71δ and streamwise wavelengths $\mathcal{O}(1000\delta_\nu)$ to $\sim 10\delta$. These extents were found to be universal for a friction Reynolds number range of $\mathcal{O}(10^3)$ to $\mathcal{O}(10^6)$. The aspect ratio and range of attached-eddy heights found using SCA correspond closely to those of the self-similar SSP structures found using overdamped LES in [87] and [33].

Modal decomposition techniques, like DDWD, aim to represent coherent structures with modes. In [77], the radial POD modes of pipe flow were found to be self-similar for heights $150\delta_\nu$ to $0.6R$. In [233], three-dimensional POD modes of a turbulent channel flow were decomposed into individual eddies called "POD eddies;" the wall-attached POD eddies were found to be physically self-similar for heights $\mathcal{O}(100\delta_\nu)$ to $0.4h$. In [73], a wavelet decomposition in time (using db20 wavelets) and POD in space were performed on TBL flow, which yielded modes that were self-similar up to height $\sim 0.5\delta$.

Table 2.1 compares the ranges of self-similarity found by the methods discussed above to the ranges DDWD finds. Compared to the studies above, our DDWD results suggest similar ranges of self-similarity of eddies: streamwise extent starting from $40\delta_\nu$ (based on the wavelets) or $450\delta_\nu$ (based on the wavelet projections) up to $1R$ (and possibly $\mathcal{O}(10R)$), and heights starting from $350\delta_\nu$ to $1R$. Since the closest measurement of the velocity was at 350 wall units, the true lower bound for the self-similarity in the wall-normal direction of the flow experiment may be smaller like in other studies. Finally, we note that while the DDWD wavelet projections give a lower bound of streamwise self-similarity that matches well with other studies, the same cannot be said about DDWD wavelets—one reason could be that DDWD wavelets are uncovering self-similarity of coherent structures related to the energy cascade, rather than the AEH, at these smaller scales.

2.6 Additional details about use and validation of DDWD

Method	x_1/δ_ν	x_2/δ_o	y_1/δ_ν	y_2/δ_o	Re_τ	Type	Data
Flow visualization [1, 217, 257]	varies	varies	$\mathcal{O}(100)$	$\sim$0.3–1	varies	varies	varies
Overdamped LES: streaks [87]	1000	15	10	0.15	1000–2000	channel	sim.
Overdamped LES: vortices [87]	200–300	3–4	50–70	0.75–1	1000–2000	channel	sim.
Spectral coherence analysis [6]	1120	10	80	0.71	2000–1.4×10^6	TBL	varies
POD [77]	n/a	n/a	$\sim$150	$\sim$0.6	1300-2500	pipe	exp.
POD eddies [233]	$\sim$600	$\sim$1.5	$\sim$130	$\sim$0.4	2000	channel	sim.
POD + wavelet transform [73]	$\sim$200	$\sim$2	$\sim$50	$\sim$0.5	235	TBL	exp.
DDWD: wavelets	40	$\sim$1	350	1	12400	pipe	exp.
DDWD: wavelet projections	450	$\sim$1	350	1	12400	pipe	exp.

Table 2.1: Comparison of DDWD to other methods in estimating the range of streamwise self-similarity, $[x_1, x_2]$, and wall-normal self-similarity, $[y_1, y_2]$, of attached eddies in simulation (sim.) and experimental (exp.) datasets of wall-bounded turbulence. δ_ν is a wall-unit, and δ_o is the outer-length scale.

2.6.1 Details of obtaining an 18-stage DDWD basis

As mentioned in Section 2.3.2, one iteration of the optimization problem has a computational complexity of $\mathcal{O}(N^2 \log_2 N + N^2 M)$, and the number of iterations required for the BFGS algorithm to converge grows with N. The optimization problem is intractable for $N = 2^{18}$. To make the problem tractable, we use the fact that wavelets are spatially localized.

The principal idea behind drastically reducing the computation time of DDWD is to find shortened wavelets of length $N_s \ll N$, and then pad them with zeros until they have length N. As long as a wavelet of length N is localized within a region of length less than N_s, it can be accurately represented by a wavelet of length N_s. Reciprocally, a wavelet of length N_s that is localized can be extended to a wavelet of length N by padding it with zeros. Since the spatial extent of a wavelet increases with the stage l roughly as 2^l, our idea will seemingly only work for the early stages of the wavelet hierarchy corresponding to small-scale wavelets. We circumnavigate this issue by working with filters (also called wavelet generators) instead of directly working with wavelets. In what follows, we provide a minimal background on how filters are used to produce a wavelet basis (further details appear in [50]), and how we modify the approach to create an optimization problem that is tractable for large N.

We work in the space $\mathbb{R}^N$. A first-stage wavelet basis for $\mathbb{R}^N$ is an orthonormal basis of the form

$$\{R^{2m} u_1\}_{m=0}^{N/2-1} \cup \{R^{2m} v_1\}_{m=0}^{N/2-1}. \tag{2.7}$$

That is, $\mathbb{R}^N$ is spanned by $N/2$ mutually orthonormal even translates of $u_1, v_1 \in \mathbb{R}^N$. The two sets in (2.7) are respectively bases for the subspaces V_{-1} and W_{-1} that appear in Figure 2.4. The vectors u_1 and v_1 are called filters, with u_1 being equal to the father wavelet ξ_{-1} introduced in Section 2.3.1, and v_1 being equal to the mother wavelet ψ_{-1}. A key point is that wavelets can be constructed from the filters, and working directly with the filters confers computational advantages.

The orthonormality conditions for v_1 and its even translates can be stated in Fourier

space as

$$|\hat{v}_1(n)|^2 + |\hat{v}_1(n+N/2)|^2 = 2\,, \quad n = 0, 1, \ldots, N/2-1, \tag{2.8}$$

where $\hat{}$ denotes the discrete Fourier transform (DFT). For a vector $z \in \mathbb{R}^N$, $z(n)$ is its n^{th} component, indexed beginning from zero. The same conditions hold for u_1, and additional conditions are needed to ensure that the even translates of u_1 and v_1 are mutually orthonormal. Given a v_1 that satisfies (2.8), u_1 can be constructed from v_1 (and vice versa) so that (2.7) is an orthonormal basis.

For $N = 2^p$, one can construct a p^{th}-stage wavelet basis for $\mathbb{R}^N$ corresponding to the p stages of the wavelet hierarchy in Figure 2.4. To do so, we require p sets of filters. Since the dimension of subspaces V_{-l} and W_{-l} is $N/2^l$, the filters become progressively shorter at later stages; specifically, $u_l, v_l \in \mathbb{R}^{N/2^{l-1}}$. The filter v_l at stage l must satisfy the orthonormality constraints in (2.8) (with N replaced by $N/2^{l-1}$), and u_l can be constructed from v_l.

Given the filters, the wavelets at stage l are given by

$$\psi_{-l} = u_1 * U(u_2) * U^2(u_3) * \ldots * U^{l-2}(u_{l-1}) * U^{l-1}(v_l), \tag{2.9}$$

$$\xi_{-l} = u_1 * U(u_2) * U^2(u_3) * \ldots * U^{l-2}(u_{l-1}) * U^{l-1}(u_l), \tag{2.10}$$

where $*$ denotes convolution and U is the upsampling operator. The upsampling operator doubles the length of its input and is defined by $U(z)(n) = z(n/2)$ for n even and 0 for n odd. The $N/2^l$ translates by 2^l of ψ_{-l} and ξ_{-l} span subspaces W_{-l} and V_{-l}, respectively. Since the wavelets are constructed by repeated convolutions of the filters, localized filters yield localized wavelets. For traditional wavelets, the stage-l wavelet has a fixed length N and is localized over a length proportional to 2^l. Conversely, the corresponding filters have varying lengths of $N/2^{l-1}$ but are localized over constant lengths. We will make use of this feature for our modified DDWD algorithm.

In DDWD, we find optimal filters stage by stage. At the first stage, the filters have length N, making the optimization problem intractable for $N = 2^{18}$. At stage l, the filters have

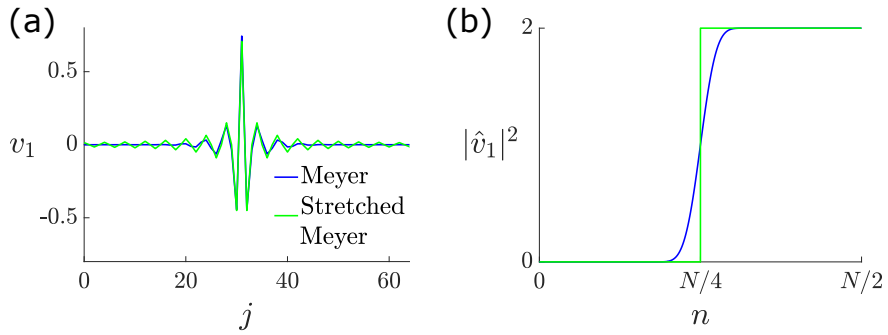


Figure 2.15: Filter v_1 (equivalently, wavelet ψ_{-1}) from the Meyer and stretched Meyer basis in real space (a) and Fourier space (b).

length $N/2^{l-1}$, making the optimization problem easier as we move to later stages. Thus for large N, only the initial stages are intractable, so only the initial stages need to be modified.

In the initial stages, the length over which filters are localized will be $N_s = 2^7$ or less; the reason is as follows. Figure 2.15 shows $v_1 = \psi_{-1}$ and its PSD for a Meyer wavelet in blue. One can confirm orthogonality of the first-stage Meyer basis, since each pair of PSD values located symmetrically around the frequency bin $n = N/4$ satisfies Equation 2.8. Generally, wavelets less spatially-localized are more localized in Fourier space; from this idea, we can construct the "stretched Meyer" wavelet shown in Figure 2.15, which is the most spatially-localized wavelet in Fourier space that still satisfies the orthogonality condition. The stretched Meyer wavelet is computed using Fourier modes with magnitudes as depicted in Figure 2.15b and phases matching those of the normal Meyer wavelet. Figure 2.15a suggests that v_1 from any orthogonal wavelet basis for $\mathbb{R}^N$ will have a spatial extent less than $2^6 = 64$ (or $N_s = 2^7 = 128$ to be safe), provided the phases of $\hat{v}_1$ are manipulated to decrease spatial-localization of v_1 (which is what the variance penalty term in Equation 2.1 does).

We now discuss the modified DDWD algorithm. Let $Z \in \mathbb{R}^{N \times M}$ contain an ensemble of M data vectors of length N. The wavelets (and corresponding filters) at the first stage will be highly localized, permitting us to use data vectors of length $N_s \ll N$ to find them. In this work, we use $N_s = 2^{p_s} = 2^7$. The first step is to resize Z into $Z_s \in \mathbb{R}^{N_s \times MN/N_s}$ so that Z_s contains the shortened data vectors. Standard DDWD is then applied to Z_s to solve for

$v_{1s} \in \mathbb{R}^{N_s}$, which has computational cost $\mathcal{O}(N_s^2 \log N_s + N_s^2(MN/N_s))$ per iteration of the BFGS solver. Moreover, the BFGS solver requires far fewer iterations to converge for a lower-dimensional problem. Then, the filter v_{1s} is padded with zeros until it has length N to give the full-length filter v_1. In general, padding yields a filter v_1 that is slightly non-orthogonal to its even translates. This is easily remedied by multiplying the Fourier coefficient pairs that appear in (2.8) by the unique positive real number that will make condition (2.8) hold, restoring the orthonormality property. Next, v_1 is used to project Z onto the subspace V_{-1} to form $Z_1 \in \mathbb{R}^{N/2 \times M}$. The coarsened data matrix Z_1 is used to calculate the filter at the next stage, v_2, the same way Z is used to calculate v_1. Note that as we move on to the next stage, the dimension of the coarsened dataset is halved, making the optimization problem increasingly tractable. We repeat this process recursively until stage $l = p - p_s + 1$, at which point the coarsened dataset has dimension 2^{p_s} without being shortened. From this stage onward, DDWD without modifications is performed.

In the end, the modified algorithm returns the full p-stage DDWD basis for $\mathbb{R}^N$ but with vastly reduced computing time. The computational savings are enabled by only ever directly using data vectors of maximal length $N_s \ll N$. Crucially, presupposed localization of the wavelets allows us to use shortened data vectors without distorting the resulting wavelets, and the validity of this presupposition can be checked after the fact (we have checked that $N_s = 2^7$ is indeed sufficient).

2.6.2 DDWD validation using synthetic, self-similar signals

To validate the results produced by DDWD, we test whether the algorithm can correctly recover the shape of a wavelet from a synthetic signal that is a superposition of a known wavelet across many scales and locations.

We construct the synthetic signal so that it mimics the velocity measurement at $y^+ = 350$ in the Superpipe. The synthetic signal, u_{synth}, has length 2^{24}. We generate u_{synth} by specifying values for its wavelet coefficients and then perform an inverse discrete wavelet transform. The

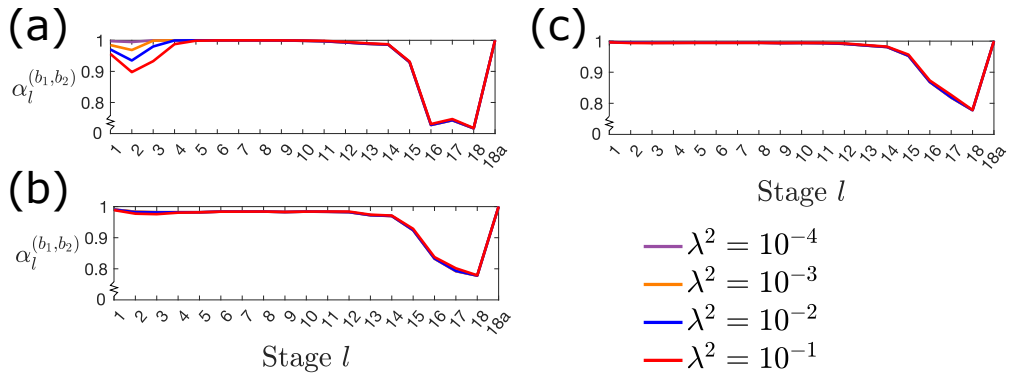


Figure 2.16: Robustness of DDWD to the strength of the variance penalty parameter λ^2 for synthetic signals composed of (a) Meyer, (b) fk6, and (c) db2 wavelets (b_1 is the corresponding wavelet basis and b_2 is the DDWD basis). Refer to Figure 2.6b for the shapes of these wavelets.

coefficients corresponding to each wavelet subspace are sampled from a normal distribution and then normalized such that the energy in each wavelet subspace matches the WED in Figure 2.13a (we choose to match the WED of the Meyer wavelet in that figure). The energy in stages $l \in [19, 24]$ is set equal to zero. We create a dataset of vectors of length 2^{18} from this signal the same way that we created a dataset from the Superpipe velocity signal.

First, we test the robustness of DDWD to the strength of the variance penalty parameter λ^2. To measure the ability of DDWD to uncover the wavelet from which the synthetic dataset is built, we calculate the inner product $\alpha_l^{(b_1,b_2)}$ from (2.2) between the DDWD wavelets and the pre-specified wavelet. As in Section 3.3, $\alpha_l^{(b_1,b_2)} \gtrsim 0.95$ is considered highly accurate. Figure 2.16a shows that DDWD is accurate for stages 1 through 14 across two orders of magnitude of λ^2 when the synthetic dataset is constructed from Meyer wavelets.

Next, we test the accuracy of DDWD when applied to synthetic datasets constructed from fk6 or db2 wavelets. Figures 2.16b and c show that DDWD is able to accurately recover the underlying wavelets. Furthermore, the results are extremely robust to the value of λ^2. In what follows, we fix $\lambda^2 = 10^{-4}$, which is the value that we use when applying DDWD to the Superpipe data.

Finally, we test the robustness of DDWD to the energy distribution of the synthetic signal. The synthetic dataset is constructed from Meyer wavelets and matches the WED

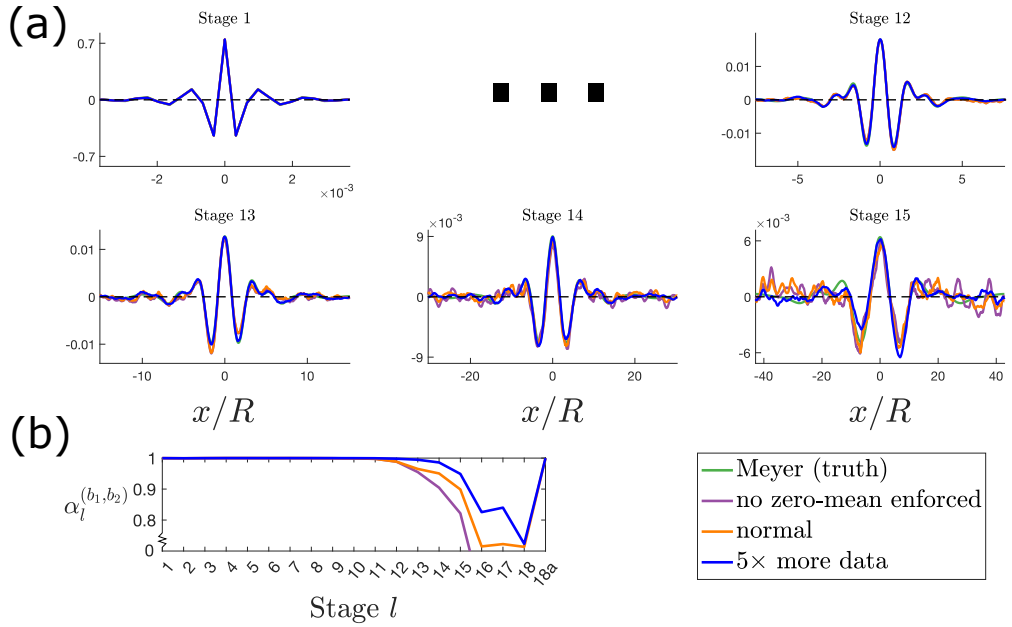


Figure 2.17: Results of DDWD applied to a synthetic dataset generated from Meyer wavelets using the WED of the Superpipe signal measured at the centerline. Label "normal" indicates that a zero mean was enforced for the DDWD wavelets. Plotted are (a) wavelets at relevant stages and (b) accuracy at each stage with b_1 as the true basis and b_2 as the computed DDWD basis.

of the Superpipe velocity at $y/R = 1$ (see Figure 2.14). Figure 2.17 shows that DDWD is accurate for stages 1 through 14 when the wavelets are enforced to have zero mean (labelled "normal"). A zero mean was enforced for the previous tests and for the application to the Superpipe data. When a zero mean is not enforced, the results are less accurate. The accuracy begins to decrease past stage 12, which is where the energy peaks. Similarly, the accuracy in Figure 2.16 begins to decrease past stage 14, which is where the energy peaks for the corresponding dataset. The results produced by DDWD seem to be most reliable in the range of scales where the energy monotonically increases with scale, which is likely related to DDWD minimizing energy scale by scale. As is shown, the results become more accurate as the amount of data increases. Overall, the validation results here indicate that DDWD is accurate for all wall-normal positions from stage 1 through stage 14.

2.6.3 Defining the length scale of a wavelet

There is no consensus for how to measure the length scale of a wavelet, which we denote with λ_x. (Note, our definition for λ_x here is unrelated to the Fourier wavenumber k.) The most common approach is to calculate the center wavenumber of the wavelet, k_c, which is the wavenumber corresponding to either the peak or the first moment of the wavelet's PSD. However, we find that the center wavelength, $\lambda_c = 2\pi/k_c$, is usually too small. For example, the stage-11 wavelet in Figure 2.7a has $\lambda_c \approx 0.6R$, which we find to not be reflective of the length scale of the wavelet since λ_c covers approximately one oscillation while a wavelet consists of several oscillations.

Our method for measuring λ_x involves centering an interval about the center of a wavelet and shrinking the support of the interval from both ends until it contains only 90% of the wavelet's energy; the length of the interval is λ_x. The 90% threshold is not arbitrary. We calculate the wavelet energy distribution (WED), which is the energy in the detail subspaces at each stage, and plot it with stage converted to $2\pi/\lambda_x$—a 90% threshold leads to the best overlap with the premultiplied PSD. (See more details on the WED in Section 2.4.5.) As seen with the stage-11 wavelet in Figure 2.7a, our method for calculating $\lambda_x = 1.26R$ seems to better reflect its length scale than λ_c.

2.6.4 Extending DDWD to 2D and 3D data

While we only work with 1D data in this paper, we briefly discuss how DDWD can be extended to 2D and 3D analyses. In 1D, we maximize the energy in the approximation subspace (contains low-wavenumber information) at each stage. For each stage in 2D, we have two options for the energy maximization. The first option is to maximize the energy in the 2D subspace containing low-wavenumber information in both directions. The second option is to perform 1D DDWD in each direction, then take a tensor product of the 1D bases to get a 2D DDWD basis.

2.6.5 Relation between wavelet projection and filtering

Insert this sentence into Section 2.4.4: DDWD wavelets act similarly, but not equivalently, to band-pass filters (see Section 2.6.5 for how a wavelet projection can deviate non-linearly from a filtering operation). Here, we interpret the properties of the wavelets found by DDWD. The first observation is that DDWD wavelets are less localized in real space when compared to traditional wavelets at the same stage (see Figure 2.6b), which means they are more localized in k space. The second observation is that Figure 2.10 suggests a wavelet projection projection is equivalent to a filtering operation, since the convolution theorem states that the filtering of two signals in real space is the same as element-wise multiplication of them in k-space. These two observations imply that for the Superpipe signal measured closest to the wall, its wavelet projections have minimal energy (area under the curve) when the wavelets are narrow in k space – which seems intuitive. One might wonder if DDWD applied to any broadband signal will return wavelets that are narrow in k space – the answer is no (example shown at end of this section) due to dissimilarites between wavelet projections and filtering.

We will show that performing a wavelet projection of a signal is similar, but not equivalent, to filtering the signal with the wavelet. Consider a signal u partitioned into segments $\{z_i \in \mathbb{R}^N\}_{i=1}^M$. The definition for the wavelet projection onto the detail subspace W_{-l}, repeated from the main text in Section 2.4.4, is

$$P_l z_i = \sum_{m=0}^{N/2^l-1} \underbrace{\left\langle R^{2^l m}(\psi_{-l}), z_i \right\rangle}_{c_l^{(1)}(m)} R^{2^l m}(\psi_{-l}), \tag{2.11}$$

(As stated in the main text, $P_l u$ will be used interchangeably with $P_l z_i$; we also drop the subscript on z_i henceforth for simplicity.)

The result of filtering z with ψ_{-l} (or vice-versa) is the cross correlation $c_l = \psi_{-l} \star z$, where

the jth component of the vector $c_l \in \mathbb{R}^N$ is given by

$$c_l(j) = \left\langle R^j(\psi_{-l}), z \right\rangle . \tag{2.12}$$

Not sure to include or not: As a side note, c_l is referred to in the wavelet community as the Continuous Wavelet Transform (CWT) of z using wavelet ψ_{-l}; the CWT, which is not aptly named since it can refer to either the analog or digital transform, uses (typically) non-orthogonal wavelets at non-dyadic scales in an overcomplete mapping. c_l is, by definition, the result of a filtering operation; further evidence of this fact can be shown in Fourier space as follows. By using the convolution theorem along with the Fast Fourier Transform (FFT), c_l can be quickly computed with

$$c_l = z * \tilde{\psi}_{-l} = \left(\hat{z}\, \hat{\tilde{\psi}}_{-l} \right)^{\vee}, \tag{2.13}$$

where the $*$ denotes convolution, the $\tilde{}$ denotes conjugate reflection, the $\hat{}$ denotes Fourier transform, and the $^{\vee}$ denotes inverse Fourier transform. Figure 2.18a confirms that the cross correlation of z and ψ_{-l} (or convolution of them but with one of them conjugate reflected) in real space is equivalent to element-wise multiplication of them in Fourier space. Figure 2.18a suggests the wavelet projection $P_l u$ and c_l look very similar in k space when ψ_{-4} is a DDWD wavelet (although not shown, $P_l u$ and c_l also look very similar in real space) – $P_l u$ may also be the result of filtering z with ψ_{-4}. But in Figure 2.18b, with ψ_{-4} being a db2 wavelet, one can clearly see nonlinear effects that suggest the wavelet projection isn't exactly analagous to a filtering operation.

We now investigate how the wavelet projection is similar to a filtering operation. Let $c_l^{(1)} \in \mathbb{R}^{N/2^l - 1}$ be the vector whose components are the coordinates of the wavelet basis spanning W_{-1} (see Equation 2.11). We can write $c_l^{(1)} = D_l^{(1)} c_l$, where $D_l^{(1)} : \mathbb{R}^N \to \mathbb{R}^{N/2^l}$ is

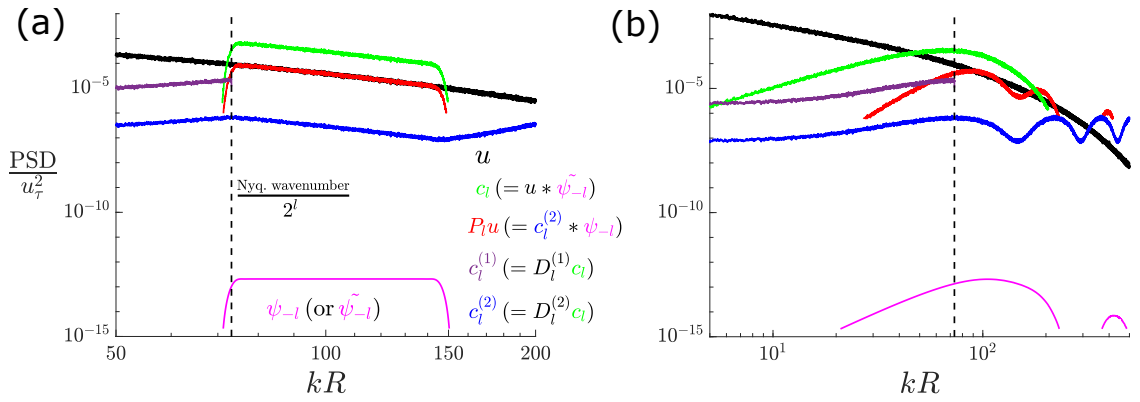


Figure 2.18: Results for Superpipe signal $u(x)$ at $y^+ = 350$. Plotted are the PSD (e.g., for u, plotted is $|\hat{u}|^2$) of various quantities calculated using a stage-4 (a) DDWD and (b) db2 wavelet. See text for the definitions of the quantities.

the first (of two) downsampling operator we define by

$$\left(D_l^{(1)} c_l\right)(m) = c_l(2^l m)\,, \quad m = 0, 1, \ldots, N/2^l - 1. \tag{2.14}$$

For every 2^l components of its input, $D_l^{(1)}$ takes the first and skips the other $2^l - 1$. As will be needed later, we define a second downsampling operator $D_l^{(2)} : \mathbb{R}^N \to \mathbb{R}^N$ that is similar to the first; for every 2^l components of its input, $D_l^{(2)}$ still takes the first component, but now zeros-out the other $2^l - 1$ components. Let $c_l^{(2)} = D_l^{(2)} c_l$, where $D_l^{(2)}$ is formally defined with

$$\left(D_l^{(2)} c_l\right)(j) = \begin{cases} c_l(j) & , \frac{j}{2^l} \in \mathbb{N}_0 \\ 0 & \text{otherwise} \end{cases}, \quad j = 0, 1, \ldots, N-1. \tag{2.15}$$

Let Ψ_{-l} be the square matrix containing $R^j(\psi_{-l})$ in its $(j+1)$th column for $j = 0, 1, \ldots, N-1$. Since Ψ_{-l} is a circulant matrix that is diagonalized by Fourier modes, an alternative form of the DWT projection in Equation 2.11 is

$$P_l z = \Psi_{-l}\, c_l^{(2)} = F^{-1}\, \mathrm{diag}(\hat{\psi_{-l}}) F\, c_l^{(2)} = \left(\hat{\psi_{-l}}\, \hat{c_l^{(2)}}\right)^{\vee} = \psi_{-l} * c_l^{(2)}, \tag{2.16}$$

where F is the DFT matrix. One can see from Figure 2.18 and Equations 2.13 and 2.16 that

c_l results from filtering z with ψ_{-l} (i.e., convolution of z with $\tilde{\psi_{-l}}$), while $P_l u$ results from filtering $\tilde{c_l^{(2)}}$ with ψ_{-l}.

We now investigate how the wavelet projection deviates from a filtering operation. Let f_N be the Nyquist frequency of c_l. When computing $c_l^{(1)} = D_l^{(1)} c_l$, the $f > f_N/2^l$ frequencies become heavily aliased into the $f \leq f_N/2^l$ frequencies (see Figure 2.18). Putting zeros in between the components of $c_l^{(1)}$ to get $c_l^{(2)}$ makes $\left|\hat{c_l^{(1)}}\right|^2$ repeat 2^l times with alternating directions (I don't know how to mathematically show this). Finally, $P_l u$ is obtained by using ψ_{-l} to convolve $c_l^{(2)}$ (or to filter $\tilde{c_l^{(2)}}$). Clearly, computing a wavelet projection there are non-linear effects in addition to a filtering operation when one computes a wavelet projection.

We return to the original question of whether broadband signals will always have wavelet projection with minimal energy (minimal area under the PSD) when using wavelets narrow in k space. are used – the answer is no. To see this, we first make a (lengthy) comparison of wavelet projections to filtering. Consider a synthetic signal made of db2 wavelets, which we generate following the procedure in Section 2.6.2, with a WED mirroring that of the Superpipe signal closest to the wall. DDWD correctly uncovers db2 wavelets as having minimal energy (see Figure 2.16c). Figure 2.19 shows various PSDs but for the synthetic signal (note that $c_l^{(1)}$ is flat in k space due to being sampled from a normal distribution in the procedure described in Section 2.6.2). This result for the broadband synthetic signal suggests a wavelet wider in k space can help minimize the energy of wavelet projections better than one narrower in k space – this counterintuitive result is due to the wavelet projection not being exactly analogous to a filtering operation.

2.7 Conclusions

DDWD was used to extract energetic, spatially localized structures from experimental measurements of the streamwise velocity in a fully developed turbulent pipe flow in the Princeton Superpipe. The structures—which we interpret as being reflective of underlying eddies—were

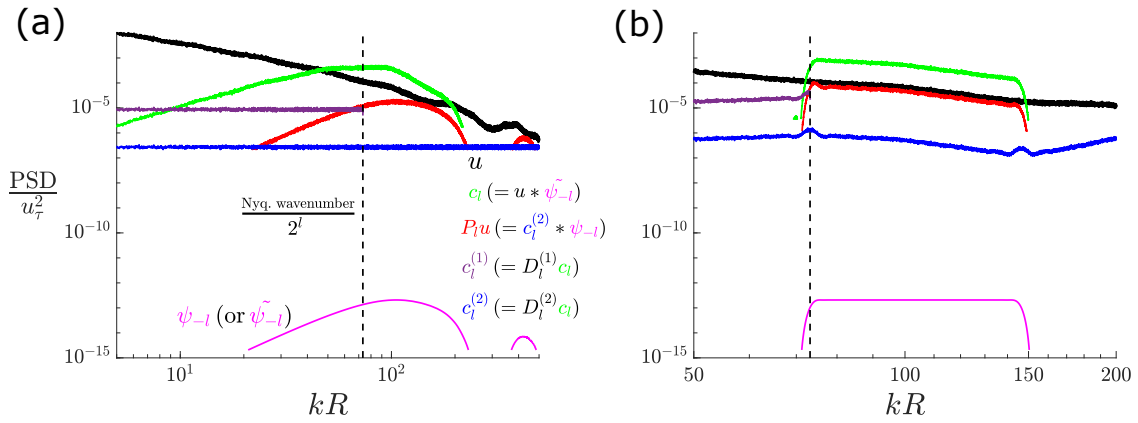


Figure 2.19: Results for a synthetic db2 signal constructed with E_l taken at $y^+ = 350$. Plotted are the PSD (e.g., for u, plotted is $|\hat{u}|^2$) of various quantities calculated using a stage-4 (a) db2 (what is found by DDWD) and (b) near-top-hat wavelet (taken from Figure 2.18). See text for the definitions of the quantities.

found to be self-similar across streamwise extents of 40 wall units to one pipe radius, and across wall-normal positions between $y^+ = 350$ to $y/R = 1$. We found that the structures were similar in shape to Meyer wavelets. An analysis of the spectra of projections of the data onto DDWD subspaces yielded the same bounds for self-similarity except with streamwise self-similarity starting at a larger length scale of 450 wall units. This wavelet projection metric might be more accurate since it considers both the shape of the wavelet (learned from data) and the Fourier content of the data, and since some studies [6, 87] propose the lower bound of streamwise self-similarity starting at $\mathcal{O}(100)$–$\mathcal{O}(1000)$ wall units.

While our bounds for streamwise self-similarity are consistent with the literature, the large range of wall-normal self-similarity seems contrary to the AEH [219], which states that attached eddies are self-similar only in the log layer; however, self-similarity of structures well beyond the log layer has been observed in other studies as well [6, 77, 87].

Characterizing attached eddies is an important first step in gaining mechanistic insight for how they are generated and sustained. In our previous work with HIT data [50], DDWD returned self-similar, Meyer-like wavelets for stages corresponding to the inertial subrange. For the Superpipe data, the inertial subrange corresponds to stages 6 through 9 (450 wall units to $\sim$0.3 pipe radii); however, here we see self-similar, Meyer-like wavelets starting from

stage 3 and ending at stage 10 or 11 (note, wavelet length scales increase by a factor of $\sim$2 with stage). While it is interesting that DDWD returns Meyer-like wavelets for two different types of turbulent flows, two caveats should be made. First, for both datasets, our 1D wavelets can only speak to 1D self-similarity of structures, and thus cannot provide detailed information about 3D structure (e.g., degree of isotropy). Second, for the Superpipe data, DDWD cannot differentiate between velocity structures induced by anisotropic attached eddies and those induced by isotropic eddies corresponding to the energy cascade—especially since the mechanisms governing these two types of eddies are likely coupled and complex.

DDWD wavelets are a natural choice for extracting spatially localized, multiscale velocity structures (induced by eddies) from data. In search for eddies that are physically attached to the wall, future efforts will be directed toward applying DDWD to multi-dimensional data and developing a DDWD analog of the spectral wall-coherence method given in [4, 6].

3

Hybrid reduced-order modeling [1]

While data-driven techniques are powerful tools for reduced-order modeling of systems with chaotic dynamics, great potential remains for leveraging known physics (i.e. a full-order model (FOM)) to improve predictive capability. We develop a hybrid reduced order model (ROM), informed by both data and FOM, for evolving spatiotemporal chaotic dynamics on an invariant manifold whose coordinates are found using an autoencoder. This approach projects the vector field of the FOM onto the invariant manifold; then, this physics-derived vector field is either corrected using dynamic data, or used as a prior that is updated with data. In both cases, the neural ordinary differential equation approach is used. We consider simulated data from the Kuramoto–Sivashinsky and complex Ginzburg–Landau equations. Relative to the data-only approach, for scenarios of abundant data, scarce data, and even an approximate FOM (i.e. erroneous parameter values), the hybrid approach yields substantially improved time-series predictions.

3.1 Introduction

The ability to efficiently and accurately forecast complex systems is highly sought after in many applications. Sometimes we know the full physics of these systems and can perform

[1] The text of this chapter is adapted from the publication A. Guo and M. D. Graham, Blending data and physics for reduced-order modeling of systems with spatiotemporal chaotic dynamics, *arXiv*, 7 2025 [67]

simulations by discretizing and solving the governing equations—this is the case where we have an *exact* "full-order model" (FOM). Since multi-query tasks such as control or design demand the FOM to be evaluated many times, directly solving the FOM is computationally prohibitive for systems exhibiting multiscale, spatiotemporal chaos. Moreover, using the FOM to calculate adjoint models and/or Jacobians (as for downstream tasks such as optimization or finding invariant solutions) is even more computationally intractable than simply time-integrating the FOM. Often times, however, we only have partial knowledge of the physics (e.g. flows of complex fluids)—this is the case where we only have an *approximate* FOM, and the discussion on how fast it is to evaluate/use for high-fidelity modeling becomes moot. For these reasons, here we are interested in developing high-fidelity, computationally efficient reduced-order models (ROMs) of the long-time dynamics of systems with complex spatiotemporal chaotic behavior. In this work, we combine data-driven methods with a physics-based FOM (which can be exact or approximate) to develop a hybrid ROM for forecasting spatiotemporally chaotic systems, demonstrating that using even an approximate FOM can substantially enhance predictive performance relative to an approach using only data.

Perhaps the simplest approach for hybrid data-physics ROMs for systems governed by partial differential equations, is the "proper orthogonal decomposition (POD)-Galerkin" method [81]. POD, also known as principal component analysis (PCA), uses data to discover an orthogonal basis ordered by variance (energy in the case of fluid flow): I.e. the first basis vector points in the direction of maximal variance in the data set, and so on. In POD-Galerkin, the governing equations are projected onto a subset of the POD basis, yielding evolution equations for the POD amplitudes. This approach has been applied to the Navier-Stokes equations in an effort to develop low-dimensional models of turbulent flows [58, 81, 221]. A significant limitation of POD-Galerkin is that it implicitly takes the data to lie on a linear subspace of the full state space (i.e. a flat manifold), which is not generally a good assumption, and may lead to a very large number of POD modes being required to

yield good performance.

Nevertheless, while data from a high-dimensional system such as the Navier-Stokes Equations (NSE) may not lie in a linear subspace, it is expected at long times to lie on a finite-dimensional invariant manifold because of energy dissipation [164, 249]. The foundation of the present work is in using simulation data to find a coordinate representation of this invariant manifold, and then finding the vector field that determines the dynamics on this manifold. For the first step of finding a nonlinear coordinate transformation to an invariant manifold, one can use an undercomplete autoencoder [60, 79]. An autoencoder uses a feedforward neural network to map data from the state space to a low-dimensional latent space, and another neural network for the reverse mapping. AEs have been used to reduce dimension in turbulent flows [161, 168, 208] and other chaotic partial differential equation (PDE) systems [115, 120]. For the second step of learning the dynamics on the manifold coordinates, one can use the neural ordinary differential equations (NODEs) [22] method. Using the time-dependent data, a NODE learns the vector field (right hand side, or RHS) of the dynamical system governing the data as a neural network. The latent space coordinates found using an autoencoder can also be forecasted with recurrent neural networks (RNNs), as demonstrated in [150, 228]); more general approaches for reduced-order, dynamical modeling can be found in [19, 48, 140]. We denote the approach of using a NODE to time-evolve the low-dimensional latent space given by an autoencoder as “data-driven manifold dynamics" (DManD) [128]. Altogether, DManD is a purely data-driven ROM. DManD has been previously used for reduced-order modeling of several chaotic dynamical systems [52, 53, 91, 129, 131, 135, 137, 246, 250, 251]. Incorporating symmetries in DManD substantially aids in training efficiency and predictive performance [92, 173, 186]. Linot and Graham [134] found that with fewer than 20 degrees of freedom, a DManD model of weakly turbulent plane Couette flow quantitatively captured key features of the flow that required 1000 degrees of freedom in a POD-Galerkin approach, despite the DManD approach having no information about the underlying governing equations. The present work illustrates how

to augment DManD with a FOM.

Dimension reduction aside, a widely used method for combining physics and machine learning for learning solutions to PDEs is physics-informed neural networks (PINNs) [101, 188]. PINNs use a physics-based loss term that penalizes the residual of the PDE at collocation points to learn PDE solutions that are physically consistent. While PINNs may also include a supervised loss on the initial and boundary conditions from data measurements, the solution is mostly influenced by the purely physics-based loss—therefore, PINNs are often regarded as ODE or PDE solvers that are not data-driven. Nonetheless, the use of a PINN-like loss term in data-driven frameworks is quite common. One example is in physics-informed deep learning [187], where the data-driven loss is placed on points outside of the initial and boundary conditions. In [127, 234], a PINN-like loss term is used with neural operators [125, 142], which are data-driven techniques for learning PDE solution operators. For the most part, PINNs are not designed to produce accurate solutions outside the time horizon of the collocation points used in training, though some work [109] has been done to address this issue. By contrast, predicting forward in time over an arbitrarily long horizon is the central topic of the present work. Because PINNs are not necessarily data-driven, not reduced-order, and not designed for time-series prediction, we do not consider this approach further.

We now arrive at the physics-based method on which our hybrid ROM described later is based. This method is called Manifold Galerkin (MG) [120]. MG is an accurate FOM-based method that nonlinearly projects the governing equations onto a manifold learned from data in the form of an autoencoder. That is, Manifold Galerkin is similar to POD-Galerkin, just with a nonlinear rather than linear coordinate transformation to a lower-dimensional manifold. The approach of using MG assumes an exact FOM is known. Although the MG model provides the vector field in a reduced-dimensional space (which can then be time-integrated for trajectory predictions), it is often more expensive to integrate than the FOM because the vector field must still be evaluated in the original high-dimensional state space

at each time step. The authors of [120] suggest hyper-reduction approaches (i.e. reducing the mesh size on which the FOM is evaluated) to address this problem of MG, which has been demonstrated in [194]. MG will be discussed in more detail in Section 3.2.2.

To the best of our knowledge, [204] is the only work besides ours that uses MG as a physics-based method to inform a data-driven forecasting model. In that work, the authors developed a hybrid physics-informed NODE (PINODE) method, which applies a soft constraint to the vector field on an autoencoder latent space learned by a NODE; this soft constraint drives the RHS learned by the NODE to the RHS of the MG model, where the former is more efficient to integrate than the latter. Our hybrid ROM (described next) differs from the PINODE method [204] in a few key ways, which are described in Section 3.2.3. Another hybrid method is presented in [172], where the FOM (subject to some approximation error) is directly combined with a reservoir computing network to forecast the state space of a chaotic system; however, since reservoir networks have a hidden state with dimensionality higher than the original state space, this hybrid method is not a ROM. Additionally, reservoir networks are inherently non-Markovian techniques (which is also true for RNNs [226, 227] and transformers [59, 208, 236]) that use past history information to make time series predictions. For data that comes from a memoryless dynamical system such as differential equation, these approaches do not maintain this dynamical systems structure. Our aim is to develop reduced-order models that maintain a differential equation structure.

In this paper, we combine the purely data-driven approach DManD with the physics-based MG method to develop a hybrid ROM framework that we call physics-informed DManD (PI-DManD). We describe our methods for constructing the hybrid ROM in Section 3.2. In Section 3.3, we apply the approach to the prediction of spatiotemporal chaos in the Kuramoto–Sivashinsky equation when varying the amount of training data and the accuracy of the physics-based prior information provided by MG. Section 3.4.3 includes results for the complex Ginzburg–Landau equation (CGLE) [2, 205], another system exhibiting spatiotemporal chaos. In all cases, PI-DManD leads to substantial improvements in predictive

capability relative to the data-only DManD approach.

3.2 Methods

We work with state space data $u \in \mathbb{R}^{d_u}$, where d_u is the state space dimension, sampled with uniform data spacing τ. Given time series data of u generated from a PDE or experiments, we aim to develop a ROM that can be used for time evolution predictions. Besides reducing the number of degrees of freedom in the system, we desire the ROM to give accurate short-time predictions and long-time predictions that capture key statistical features of the true attractor.

Here, we use a single long time series of u as the dataset. Initial transients are removed, so the data lies on an attractor that is embedded in an invariant manifold of the underlying dynamical system. We denote this dataset as $\{u_i\}_{i=1}^{M}$, where $u_i = u(t_i)$ is the snapshot of the state at time instant t_i and M is the number of snapshots. The time series data come from the underlying (ground truth) autonomous dynamical system

$$\frac{du}{dt} = f(u; \mu)\,, \tag{3.1}$$

where f is the true vector field that depends on physical parameters μ (e.g. domain length). Unless the data are from simulations, f and μ are generally not known. Still, one often has some knowledge or model of the underlying physics. Assume that we know a full-order model (FOM) of the true dynamical system in Equation 3.1, written as

$$\frac{du}{dt} = f_{\text{FOM}}(u; \mu_{\text{FOM}})\,, \tag{3.2}$$

where f_{FOM} is the vector field for the model and μ_{FOM} are the physical parameters of the FOM. The field f_{FOM} may only be approximate due to the lack of full first-principles knowledge of the system, while μ_{FOM} may only be approximate if the data are from experiments

and there is uncertainty in the parameters.

In Section 3.2.1, we start by describing a solely data-driven ROM. Then in Section 3.2.2, we discuss how the MG method [120] uses the FOM in Equation 3.2 to create a physics-based time evolution method. Lastly, we present our hybrid ROM in Section 3.2.3, which uses the MG method to incorporate physics knowledge, while also using data to correct for the FOM approximation errors that are not addressed by MG.

3.2.1 Data-driven manifold dynamics (DManD)

We first find a low-dimensional, high-fidelity latent representation of u for which to build the ROM. To do so, we will use an undercomplete autoencoder [78] to discover a coordinate transformation to (and from) a low-dimensional latent space representation $h \in \mathbb{R}^{d_h}$, where $d_h < d_u$. As mentioned in the Introduction, dissipative systems often have long-time dynamics that lie on an invariant manifold with dimension $d_{\mathcal{M}} \ll d_u$. While it is generally difficult to determine $d_{\mathcal{M}}$, there are a variety of methods for estimating it [39, 253] (see Section 3.4.3 for a discussion of one such method and how we applied it to the CGLE). Since we consider data u_i living on an attractor that is embedded in an invariant manifold, an autoencoder can in principle be trained to find a latent space whose components represent the coordinates on this manifold. We note that finding the coordinates on the invariant manifold and learning the dynamics on it can become complicated with the presence of noise, though some work [220] has been done to address this challenge.

An autoencoder consists of two back-to-back neural networks. The encoder $\mathcal{E}$ nonlinearly maps u_i to $h_i = \mathcal{E}(u_i)$. The decoder $\mathcal{D}$ performs the reverse mapping $\tilde{u}_i = \mathcal{D}(h_i)$, where $\tilde{u}_i$ is the reconstructed snapshot. We minimize the reconstruction error of the autoencoder (AE) with the loss function

$$\mathcal{L}_{\text{AE}} = \langle \|u_i - \mathcal{D}(\mathcal{E}(u_i; \theta_{\mathcal{E}}); \theta_{\mathcal{D}})\|_2^2 \rangle_i \,, \tag{3.3}$$

where $\theta_{\mathcal{E}}$ and $\theta_{\mathcal{D}}$ are, respectively, the neural network parameters of the encoder and decoder

networks and $\langle\cdot\rangle_i$ denotes an ensemble average over a batch of training snapshots. Here, and elsewhere, we use stochastic gradient descent to minimize the loss function.

With the autoencoder in hand, the DManD approach to reduced-order modeling learns a dynamical system in the latent space:

$$\frac{dh}{dt} = g(h)\,, \tag{3.4}$$

where g is the true vector field in the manifold coordinates. We train a neural ODE (NODE) [21] to approximate this function. Given a latent state $h(t_i)$, one can calculate

$$\hat{h}_{\mathrm{DManD}}(t_i+\tau) = h(t_i) + \int_{t_i}^{t_i+\tau} g_d(h(t');\theta_{g_d})\,\mathrm{d}t'\,, \tag{3.5}$$

where $\hat{h}_{\mathrm{DManD}}(t_i+\tau)$ is the NODE prediction of the latent state at the next sampled time instant, and g_d is the data-driven approximation of g. We model g_d with a deep feedforward neural network and optimize its parameters θ_{g_d} with the loss function

$$\mathcal{L}_{\mathrm{DManD}} = \langle \|\hat{h}_{\mathrm{DManD}}(t_i+\tau) - h(t_i+\tau)\|_2^2 \rangle_i\,. \tag{3.6}$$

Here, we only consider the reconstruction loss for an integration time horizon spanning a single data spacing interval τ. While one could consider a multi-step loss with a longer time horizon, we find that, for the systems studied here, doing so does not improve the results and slows down training. Nevertheless, the use of a multi-step NODE loss may be helpful in some situations, such as for time-delay embeddings of partially observed data [246]. In the results, we will increase the data spacing τ used in the NODE training. Lastly, we note that it has been recently shown [16] that the NODE approach is equivalent to a Koopman-operator-based method called extended dynamic mode decomposition with dictionary learning (EDMD-DL) [122] with a projection step onto the state.

Overall, this combination of an autoencoder and a NODE to build a ROM is coined

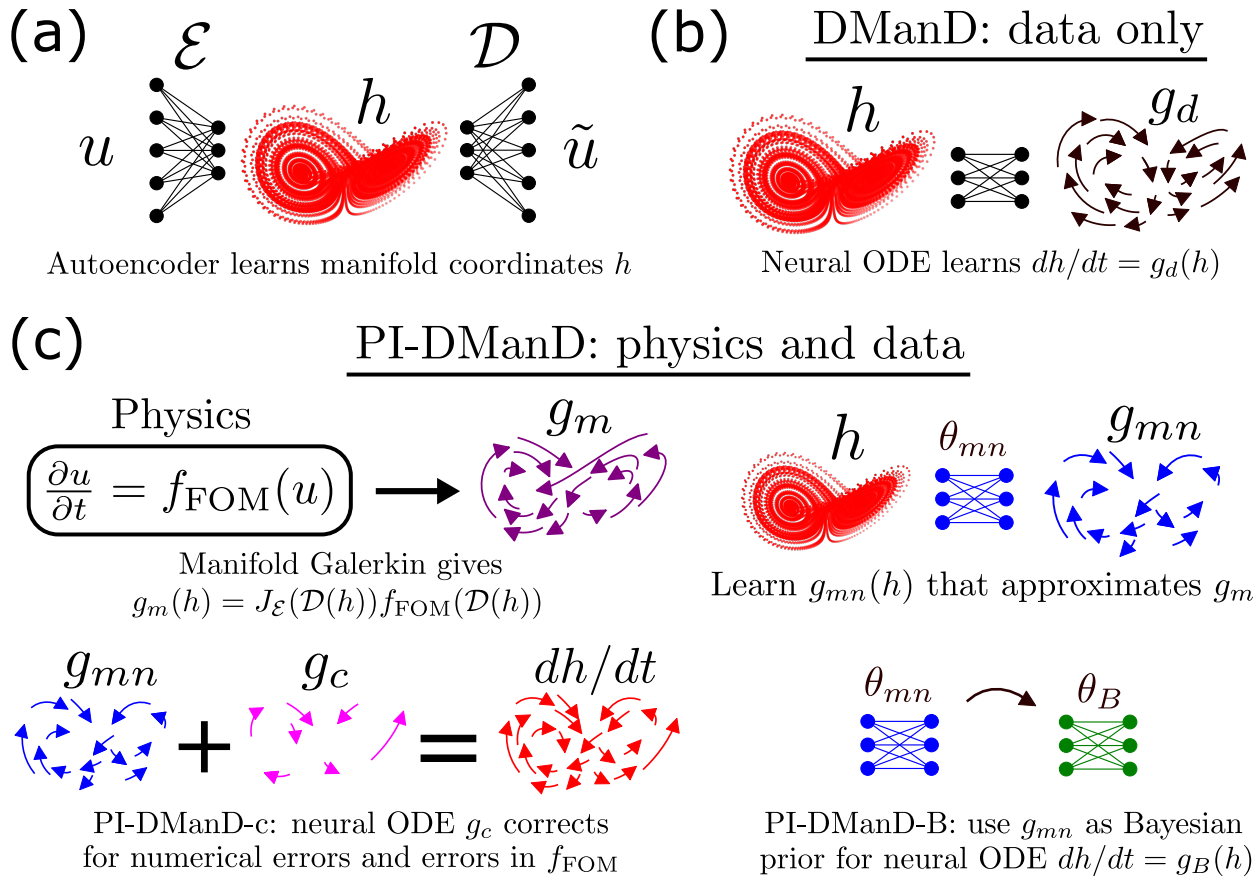


Figure 3.1: Data-driven manifold dynamics (DManD) is a data-driven ROM that reduces dimension of the state u with an autoencoder (a) and evolves trajectories with a vector field on the latent space h learned by a neural ODE (b). (c) Physics-informed DManD (PI-DManD) is a hybrid ROM that uses Manifold Galerkin to project the vector field on the full order model (FOM) f_{FOM} onto that of the latent space, learns a neural network approximation of the latent space vector field as g_{mn}, and either uses a corrective neural ODE g_c to fix numerical and FOM approximation errors (PI-DManD-c) or uses g_{mn} as a Bayesian prior for training a DManD model (PI-DManD-B).

data-driven manifold dynamics (DManD) [128]. A schematic of DManD is depicted in Figure 3.1(a) and 3.1(b). As a side note, NODE models can be stabilized for long-time predictions by following [136], in which the integrand $g_d(h;\theta_{g_d})$ in Equation 3.5 is modified to $g_d(h;\theta_{g_d}) - ah$, where $a > 0$. Although [136] found that adding the linear damping term ah primarily benefits the case where there is no dimension reduction (i.e. a NODE is used to approximate f_{FOM}), we include the damping term nonetheless when learning g_d since doing so also has no negative effects (we use $a = 10^{-3}$ for this paper). The size of the damping term does not affect the performance of the ROM as long as it is a few orders of magnitude smaller than the magnitude

of the data (order 1 here). Lastly, we note that it is possible to simultaneously learn a latent space and a NODE model; however, we find that jointly training an autoencoder and a NODE model often requires more expressive networks, leads to slower training convergence, and does not yield improved performance.

3.2.2 Manifold Galerkin

We now present the formalism for the Manifold Galerkin (MG) method proposed in [120], which directly expresses the vector field f_{FOM} in the manifold coordinates h. Using the chain rule and substituting $u = \mathcal{D}(h; \theta_{\mathcal{D}})$ into the FOM in Equation 3.2 yields

$$\frac{dh}{dt} = \frac{d\mathcal{E}(u)}{du}\frac{du}{dt} = \frac{d\mathcal{E}(u)}{du} f_{\text{FOM}}(\mathcal{D}(h; \theta_{\mathcal{D}}); \mu_{\text{FOM}}) \,, \tag{3.7}$$

Let the Jacobian of the coordinate transformation between u and h be expressed as

$$\frac{\partial \mathcal{E}(u)}{\partial u} = J_{\mathcal{E}}(\mathcal{D}(h; \theta_{\mathcal{D}}); \theta_{\mathcal{E}}) \,. \tag{3.8}$$

Since $\mathcal{E}(u)$ is a neural network, automatic differentiation can be used to easily compute $J_{\mathcal{E}}$. Substituting this expression for $J_{\mathcal{E}}$ into Equation 3.7 yields the final equation for MG:

$$\begin{aligned} \frac{dh}{dt} &= g_m(h; \theta_{\mathcal{E}}, \theta_{\mathcal{D}}, \mu_{\text{FOM}}), \\ g_m &= J_{\mathcal{E}}(\mathcal{D}(h; \theta_{\mathcal{D}}); \theta_{\mathcal{E}})\, f_{\text{FOM}}(\mathcal{D}(h; \theta_{\mathcal{D}}); \mu_{\text{FOM}}) \,. \end{aligned} \tag{3.9}$$

As discussed in Section 3.2.1, the autoencoder is trained on data living on the invariant manifold. As a result, assuming the FOM and its parameters are exact (i.e. $f_{\text{FOM}} = f$ and $\mu_{\text{FOM}} = \mu$), the full vector field is tangent to the invariant manifold coordinates found by the autoencoder. Then, we also have that $g_m(h)$ given by MG is the true vector field on the invariant manifold, expressed in the manifold coordinates h.

Although MG in Equation 3.9 has the appearance of a ROM since it provides the vector

field in the latent space h, it is in fact even slower to solve than the FOM. At each timestep, MG not only retains the expense of computing f_{FOM}, but must also first decode h to u and evaluate a Jacobian (which, for $J_{\mathcal{E}}$, involves another encoding step). Work has been done to accelerate the MG method using hyper-reduction techniques [194], which aim to perform approximate FOM computations that are independent of the ambient dimension d_u. Overall, MG is a physics-based method (but not a ROM) that projects the physics of the FOM onto the manifold coordinates found with a data-driven autoencoder.

The run-time expense of MG can be overcome by using a neural network to learn an approximation of the function $g_m(h)$. Since we know the functional form of $f_{\text{FOM}}(u)$, we can use the data points u_i to generate $f_{\text{FOM},i} = f_{\text{FOM}}(u_i)$. If the data come from simulations of the FOM, then $f_{\text{FOM},i}$ is already computed at each time step of the simulation, so we can store it instead of recomputing it. Otherwise, the step to generate $f_{\text{FOM},i}$ from u_i for $i = 1, \ldots, N$ only needs to be performed once as a preprocessing step. We then calculate the MG output data $g_{m,i} = J_{\mathcal{E}}(u_i) f_{\text{FOM},i}$ for all i. We denote the neural network for mapping the inputs $h_i = \mathcal{E}(u_i)$ to the MG outputs $g_{m,i}$ as g_{mn}, and we train it with the loss

$$\mathcal{L}_{\text{MG-N}} = \langle \| g_{mn}(h_i; \theta_{g_{mn}}) - g_{m,i}) \|_2^2 \rangle_i \,. \tag{3.10}$$

Here, only the parameters $\theta_{g_{mn}}$ are learnable (i.e, $g_{m,i}$ are constants and do not depend on the network parameters of the autoencoder). One can substitute $g_m(h)$ in Equation 3.9 with $g_{mn}(h)$ to get

$$\frac{dh}{dt} = g_{mn}(h; \theta_{g_{mn}}) \,. \tag{3.11}$$

Since $g_{mn}(h; \theta_{g_{mn}})$ is a neural network approximation of $g_m(h)$, we call this method MG - neural network (MG-N). As will be shown in Section 3.3, the MG-N model in Equation 3.11 is much faster to integrate than the MG model in Equation 3.9. Note that the dynamic nature of the data set is not used in development of MG/MG-N, so if the autoencoder is accurate, but the FOM is not, these approaches will model an incorrect vector field, albeit

on a correct set of manifold coordinates.

3.2.3 Physics-informed data-driven manifold dynamics (PI-DManD)

Now we describe how we integrate MG-N with DManD to create two physics-informed DManD (PI-DManD) methods. Figure 3.1(c) shows a schematic of these approaches. When using MG-N alone as a ROM, there are three sources of errors: numerical and test error in the autoencoder that influence the training of g_{mn}, test error in $g_{mn}(h)$ itself, and lastly, errors in the FOM. The last type of error grows as the physics-based approximation of the physics $f_{\text{FOM}}(u, \mu_{\text{FOM}})$ deviates further from the true underlying physics $f(u, \mu)$ governing the data. FOM error can occur if $f_{\text{FOM}} \neq f$ and/or $\mu_{\text{FOM}} \neq \mu$; the latter case may occur if there is uncertainty in the physical or PDE parameters (e.g. domain length), which will often arise if the dataset comes from experiments.

In our first PI-DManD approach, denoted PI-DManD-c, we account for all three types of errors by adding a correction term, $g_c(h)$, to the vector field from MG-N to get our final, reduced-order dynamical model in the latent coordinates:

$$\frac{dh}{dt} = g_{mn}(h; \theta_{g_{mn}}) + g_c(h; \theta_{g_c})\,. \tag{3.12}$$

We learn $g_c(h; \theta_{g_c})$ using a NODE approach with loss

$$\mathcal{L}_{\text{PI-DManD-c}} = \langle \|\hat{h}_{\text{PI-DManD-c}}(t_i + \tau) - h(t_i + \tau)\|_2^2 \rangle_i\,, \tag{3.13}$$

where

$$\hat{h}_{\text{PI-DManD-c}}(t_i + \tau) = h(t_i) + \int_{t_i}^{t_i+\tau} [g_{mn}(h(t'); \theta_{g_{mn}}) + g_c(h(t'); \theta_{g_c})]\ \mathrm{d}t'\,. \tag{3.14}$$

Here, only the weights θ_{g_c} are trainable (the MG-N weights $\theta_{g_{mn}}$ are kept frozen). Altogether, PI-DManD-c learns a function $g_{mn} + g_c$ for the latent space dynamics. While MG-N is a

physics-informed ROM that can be heavily biased by the FOM approximation, PI-DManD-c is a hybrid ROM that is less susceptible to this bias, as will be illustrated in the results later. Note, if the physics-based information is accurate, the MG-N vector field g_{mn} will be close to g (the exact vector field in the latent space determined from data) and the difference $g \breve{} g_{mn} = g_c$ will be small; also, in the limit of an exact g_{mn} (and no numerical errors with the autoencoder or MG-N), NODE training with weight decay will lead g_c to vanish. If the physics-based information is not accurate, PI-DManD-c will just learn a larger correction g_c—given that the information encoded in g_{mn} would still be relevant as the NODE for g_c is being trained in this case, PI-DManD-c will not simply reduce to DManD.

The second PI-DManD approach, denoted PI-DManD-B, uses the MG-N model as a prior for training the NODE model. Specifically, we aim to learn a NODE model for the RHS of

$$\frac{dh}{dt} = g_B(h; \theta_{g_B}) \,, \tag{3.15}$$

with the loss

$$\mathcal{L}_{\text{PI-DManD-B}} = \langle \| \hat{h}_{\text{PI-DManD-B}}(t_i + \tau) - h(t_i + \tau) \|_2^2 \rangle_i + \lambda_B \| \theta_{g_B} - \theta_{g_{mn}} \|_2^2 \,, \tag{3.16}$$

where

$$\hat{h}_{\text{PI-DManD-B}}(t_i + \tau) = h(t_i) + \int_{t_i}^{t_i + \tau} g_B(h(t'); \theta_{g_B}) \, \mathrm{d}t' \,. \tag{3.17}$$

The difference here as compared to a DManD model lies in the additional regularization term $\| \theta_{g_B} - \theta_{g_{mn}} \|_2$, which acts to penalize deviations of the NODE weights from the MG-N weights. The prefactor λ_B controls the strength of the regularization. By using this regularization term, the NODE model learns the vector field from data while also being softly constrained to the physics-derived vector field from MG-N—as compared to the correction NODE used in PI-DManD-c, this NODE for g_B is more natural in learning from the data and physics simultaneously, which yields benefits as will be shown later. Another difference is that here

the NODE g_B must have the same architecture as the MG-N network, which is mostly a minor restriction. In addition to the regularization term, we perform a transfer learning step of initializing the weights of g_B to those of MG-N. So, the PI-DManD-B model uses the MG-N model g_{mn} as a physics-based prior.

The loss used by PI-DManD-B can be viewed as being motivated from a Bayesian framework. To see this, consider the regression problem $y = m(x; \theta_m)$, where θ_m are the model parameters. According to Bayes' theorem, we can write the posterior distribution for the parameters θ_M given the data y as

$$p(\theta_m|y) = \frac{p(y|\theta_m)p(\theta_m)}{p(y)}\,. \tag{3.18}$$

Assuming a normal distribution for the likelihood $p(y|\theta_m) \propto \exp(-(y-m)^2/2\sigma_m^2)$ as well as for the prior $p(\theta_m) \propto \exp(-(\theta_m - \theta_p)/2\sigma_p^2)$, maximizing the log likelihood corresponds to minimizing the loss function

$$\mathcal{L} = \frac{1}{2\sigma_m^2}\langle (y-m)^2 \rangle + \frac{1}{2\sigma_p^2}\|\theta_m - \theta_p\|_2^2\,. \tag{3.19}$$

This expression is very similar to Eq. 3.16, motivating the description of PI-DManD-B as a Bayesian approach. Nevertheless, we emphasize that while its loss function can be motivated from Bayes' theorem, the PI-DManD-B model is deterministic and we are not employing a Bayesian inference framework.

It must be noted that PI-DManD bears similarities to the PINODE method of [204]. PI-DManD and PINODE use knowledge of the governing equations to get the physics-derived vector field on the latent space and combines that with a NODE; however, our method differs in several key ways. First, PI-DManD only requires a dataset composed of a time series of data points u_i, while PINODE requires additional data points called "collocation points" that are user-specified. Second, PI-DManD calculates vector field information $f_{\text{FOM},i}$ only from the state u_i (which is the only data we work with), while PINODE calculates $f_{\text{FOM},i}$

only from the user-specified collocation points and not the state data u_i. Third, PI-DManD front-loads the FOM evaluations required by MG into a preprocessing step before training the intermediate MG-N neural network, whereas PINODE performs these generally expensive FOM evaluations during their NODE training—as a result, PI-DManD would generally train faster than PINODE. Fourth, unlike PINODE, we will later consider a case where only an approximate FOM is available for supplying the physics-based information. Lastly, PINODE was only applied to systems with relatively simple long-time dynamics; systems with substantially higher complexity are considered here.

PI-DManD builds a ROM solely from timeseries data of the state u_i (from experiments or simulations) supplemented with information from a FOM (which may or may not be exact). In general, PI-DManD is intrusive in the sense that code modification may be required to calculate the vector field $f_{\text{FOM},i}$ from u_i before proceeding with MG/MG-N. One could consider the case where no FOM is known; then, if the state data is closely spaced and has low noise, one could take numerical derivatives of the state space data to estimate the vector field $f_{\text{FOM},i}$ and proceed with training a PI-DManD model. However, while this approach is non-intrusive, it is not generally applicable because of the limitations just mentioned. In our previous work [133] and this work (see Section 3.3.2), we show that DManD/PI-DManD works well even when the data spacing is half of a Lyapunov time.

3.2.4 Architecture and training details

A summary of the architectures used for all neural networks introduced in Section 3.2 is provided in Table 3.1. Training is performed in PyTorch [170]. The first 80% of the times series is used for training, and the remaining 20% for testing. For all neural networks, we use an AdamW optimizer with a multistep learning rate scheduler that starts the learning rate at 10^{-3} and halves it 7 times until $\approx 10^{-5}$. By default, we use a weight decay of 10^{-4}; however, for the networks trained in the scarce data case in Section 3.3.2 we increase the weight decay strength from 10^{-4} to 10^{-3}–10^{0} (a sweep is done to decide on the value).

Network	Function	Layers	Activations
Encoder	$\mathcal{E}$	d_u:$4d_u$:$2d_u$:d_h	ReLU:ReLU:Linear
Decoder	$\mathcal{D}$	d_h:$2d_u$:$4d_u$:d_u	ReLU:ReLU:Linear
NODE (DManD)	g_d	d_h:256:256:d_h	GeLU:GeLU:Linear
Neural network (MG-N)	g_{mn}	d_h:256:256:d_h	GeLU:GeLU:Linear
NODE (PI-DManD-c)	g_c	d_h:256:256:d_h	GeLU:GeLU:Linear
NODE (PI-DManD-B)	g_B	d_h:256:256:d_h	GeLU:GeLU:Linear

Table 3.1: Neural network architectures for the autoencoder, DManD, MG-N, and the two PI-DManD methods.

For the autoencoder used to learn $\mathcal{E}$ and decoder $\mathcal{D}$ (see architectures in Table 3.1), the dimension of the latent space h needs to be determined. For the KSE system considered in this paper, the dimension of the invariant manifold at $L = 22$ is known to be $d_{\mathcal{M}} = 8$ [39, 128, 132, 253]. For the CGLE system considered in Section 3.4.3, we estimate the dimension of the invariant manifold using an implicit rank-minimizing autoencoder [97, 253] (see Section 3.4.3 for details). For the autoencoder, we use a batch size of 64 and train for 2000 epochs. For the neural network used to learn $g_{mn}(h)$ for MG-N (see architecture in Table 3.1), we use a batch size of 256 and train for 2000 epochs.

The training details for the NODEs g_d, g_c, and g_B (see architectures in Table 3.1) are as follows. We use the 5th order Dormand-Prince (dopri5) adaptive time-step integrator in the `torchdiffeq` package [23] with default tolerances. For all NODE training, we use a batch size of 64 and train for 100,000 *iterations*. While the terms "iterations" and "epochs" are often used interchangeably in the research community with regards to NODE training, the former is the nomenclature used in the original NODE paper [21]. In an iteration, weights are updated by randomly sampling a single batch (size 64 in our case) from the entire dataset and performing a forward and backward pass, while a single epoch performs many iterations until the entire dataset is sampled. The published NODE code in [21] uses iterations, so we do too in our adapted code. For training the NODE model correction g_c for PI-DManD-c in

particular, we use near-zero weight initialization (all other neural network training is with Pytorch's default Xavier weight initialization); this choice is motivated by the fact that g_c would be zero if g_{mn} in Equation 3.12 were exact (i.e. if there were no FOM approximation error). Lastly, for PI-DManD-B, we use $\lambda_B = 10^{-5}$ (see Equation 3.16) for the strength of the regularization parameter. For the systems considered in this paper, we found that the results (even if the FOM is not exact) are robust to the value of λ_B as long as it is less than 10^{-3}; in the limit of $\lambda_B = 0$, where PI-DManD-B is still robust, the method just involves the transfer learning step of initializing the NODE weights to those of the physics-based prior (which suggests that the main advantage of PI-DManD-B comes from the initialization).

3.3 Results

We work with data generated from the Kuramoto–Sivashinsky equation (KSE), which is a 1D PDE exhibiting spatiotemporal chaos and is widely used as a test bed for data-driven methods [128, 172, 244]. The KSE has been used to model physical systems such as the instabilities in flame fronts and thin liquid films [118, 165], and is given by

$$\frac{\partial v}{\partial t} = -v\frac{\partial v}{\partial x} - \frac{\partial^2 v}{\partial x^2} - \frac{\partial^4 v}{\partial x^4}\,. \tag{3.20}$$

We use a domain length of $L = 22$ with periodic boundary conditions, wherein the solution exhibits chaotic dynamics and has a Lyapunov time τ_L of 22.2 time units [39, 41]. For this KSE system, the dimension of the invariant manifold is empirically known to be $d_{\mathcal{M}} = 8$ [39, 128, 132, 253]; therefore, we will use $d_h = 8$ as the dimension for the autoencoder. We generate a long trajectory using an exponential time-differencing RK4 (ETD-RK4) solver [102] in the Fourier domain. We remove the transient and use $M = 500,000$ snapshots of state space vectors u discretized on $d_u = 64$ uniformly spaced grid points, separated by $\tau = 0.25$ time units (though we will vary this spacing later for some cases). We use the first 80% of the dataset for training, so that the number of training snapshots N is 400,000. The

last 100,000 snapshots are for testing (even when we do not use the whole training dataset, as in Section 3.3.2). See Section 3.2.4 for other training details.

We test whether the hybrid ROMs (PI-DManD-c and PI-DManD-B) can outperform the DManD ROM, which is solely data-driven. In Section 3.3.1, we train the methods using the whole time series. In Section 3.3.2, we consider three subcases of using a subset of the time series (note, for one subcase, we will generate and use a new time series dataset with $N = 400,000$ and $\tau > 0.25$). In Section 3.3.3, we consider the case when the parameters used in the FOM do not match the simulation parameters used to generate the data—this models the situation when the FOM is not exact. (In Section 3.4.3, we repeat the tests reported here for the 1D CGLE.)

3.3.1 Abundant data, exact FOM

Here, we use the entire training dataset and the exact physics ($f_{\text{FOM}} = f$) for the MG/MG-N methods. We gauge model performance on how well the model predictions (latent space trajectories that have been decoded back to the full space) track the true solution over a time horizon of $3\tau_L$—any good method should be able to make reasonably accurate predictions over at least τ_L. We calculate the ensemble-averaged, normalized tracking error

$$\epsilon(t) = \left\langle \frac{\|u(t) - \hat{u}(t)\|_2}{\|u(t)\|_2} \right\rangle , \tag{3.21}$$

where $\hat{u}(t)$ is the model prediction trajectory, and the ensemble average $\langle \cdot \rangle$ is calculated using 300 non-overlapping trajectories whose initial conditions are from the testing dataset. In Figure 3.2(a), we plot the tracking error $\epsilon(t)$ for DManD, the physics-informed methods (MG and MG-N), and the two hybrid methods. Here, all non-DManD methods perform similarly and display much smaller error than DManD—at $t = \tau_L$, their error is smaller by a factor of two; this result suggests that including physics aids in model predictions when the provided physics are exact. Figure 3.2(b) shows space-time plots of representative

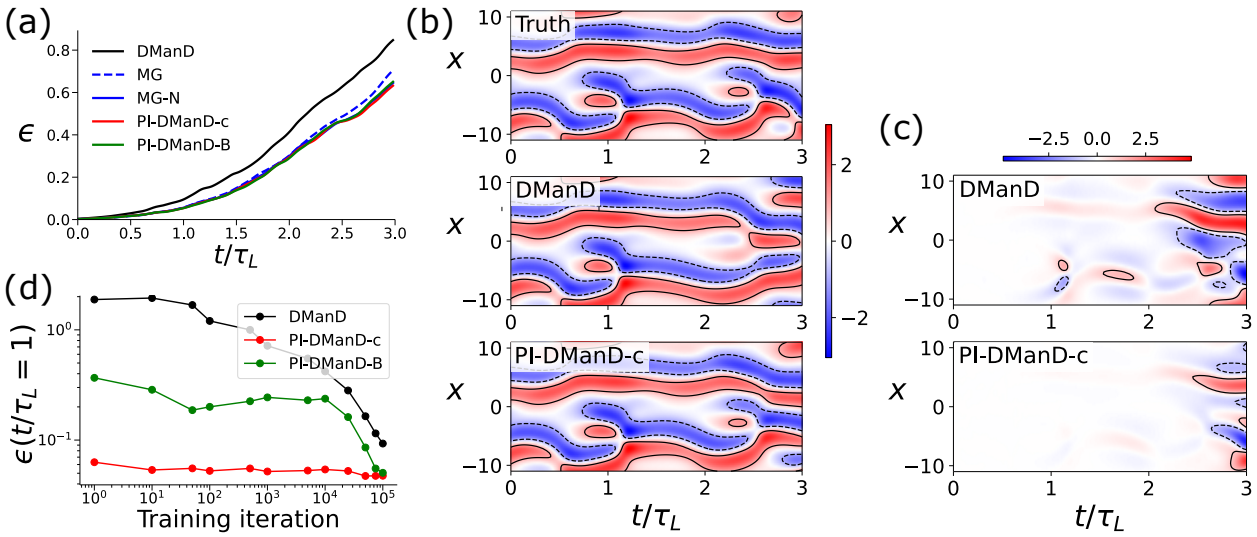


Figure 3.2: KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. (a) Ensemble-averaged short-time tracking error for all methods. (b) Representative trajectories of ground truth and model predictions (contours at $u = \pm 1$). (c) Difference plots between ground truth and model predictions (contours at difference values of ± 1). Representative trajectories and difference plots for other models look similar to those of PI-DManD-c. (d) NODE training convergence for purely data-driven and hybrid methods.

trajectories for ground truth and the DManD and PI-DManD-c model predictions. The differences between the ground truth and model trajectories in Figure 3.2(b) are plotted in Figure 3.2(c), indicating that noticeable differences arise sooner for DManD than for PI-DManD-c. (Representative trajectories and difference plots for other models, not shown, look similar to those of PI-DManD-c due to similar error curves $\epsilon(t)$—see Figure 3.2(a).) In this case where we have a large amount of closely spaced training data and the FOM is exact, all physics-informed and hybrid methods outperform DManD in short-time forecasting.

We now turn to the time used to train and evaluate the ROMs. Figure 3.2(d) shows the evolution of $\epsilon(t = \tau_L)$ during the course of NODE training, for DManD and the two PI-DManD variants. Most notably, the correction vector field g_c learned by PI-DManD-c converges several orders of magnitude faster than the vector field g_d learned by DManD. However, since PI-DManD-c (and PI-DManD-B) are derived from MG-N, the MG-N training time to learn g_{mn} must also be accounted for—in this case, the time to just train MG-N is around 4 times longer than the time to train DManD (see details in Section 3.4.1). While the

combined training time for PI-DManD-c is longer than for DManD, it still has the benefit of better predictive performance. As for the inference time, all models perform comparably (a few seconds) except MG—since MG is not a ROM, it takes 9 to 48 times longer to evolve an initial condition than the other methods (see Tables 3.2 and 3.3 in the Section). Although neural network-based ROMs require a relatively long time to train (a few hours in the case here), they still offer many advantages over using a FOM. First, as discussed in the Introduction, it is often the case (e.g. flows of complex fluids) that there is no exact FOM, where the point about how fast the FOM can be solved is not relevant; in this case, our hybrid ROMs can still be trained with timeseries data supplemented by approximate knowledge of the physics (we demonstrate this case later by using mismatched parameters between data and FOM). Second, the initial training time for these ROMs is easily offset once many evaluations of the ROM (or calculations of its Jacobians or adjoints) is performed. Lastly, we comment that for the 1D PDE systems we consider in this paper (which we are just using as benchmarks), the ROM is not necessarily faster than the FOM, since the amount of dimension reduction from the ambient space dimension to the invariant manifold dimension is moderate—by contrast, for systems such as a chaotic 2D or 3D flow where there is a vast amount of dimension reduction (e.g. ambient space dimension of about 100,000 to manifold dimension 18), the speedup of a ROM would be more noticeable [32, 134, 186**?**].

In Section 3.4.2, we evaluate how well the solution trajectories stay on the true attractor at long times by calculating the joint probability density function (pdf) of some key quantities. There, we show that all five methods captured this long-time statistic equally well.

3.3.2 Scarce data, exact FOM

While still using the exact FOM, we now study the effect of having less data (training data size $N < 400,000$), data spaced further apart ($\tau > 0.25$), or both. We apply this data-scarce training for the autoencoder, the MG-N network, and all NODEs. To prevent overfitting, the weight decay strength in all neural network training is increased here (see training details

in Section 3.2.4).

We first fix the data spacing at $\tau = 0.25$ and vary the training data size from $N = 400,000$ to $100,000$, $25,000$, and $10,000$. In other words, we use a shorter part of our time series data. Figure 3.3(a) shows the ensemble-averaged tracking error ϵ at one Lyapunov time for these four values of N. Above each group of bars in Figure 3.3(a), we report the percent error for the autoencoder (AE) reconstruction

$$\epsilon_{\text{AE}} = \left\langle \frac{\|u_i - \tilde{u}_i\|_2}{\|u_i\|_2} \right\rangle_i \times 100\% \,, \tag{3.22}$$

and the percent error for MG-N in approximating the MG vector field

$$\epsilon_{\text{MG-N}} = \left\langle \frac{\|g_m(h_i) - g_{mn}(h_i)\|_2}{\|g_m(h_i)\|_2} \right\rangle_i \times 100\% \,, \tag{3.23}$$

where $\langle \cdot \rangle_i$ is an ensemble average across all snapshots (indexed by i) in the testing dataset. The base case of $N = 400,000$, which corresponds to Figure 3.2(a), shows that DManD has higher tracking error, by a factor of two, than the other methods. Within the error bars, MG-N and the two hybrid methods perform comparably to DManD for $N < 400,000$. The reason that NODE test error ϵ increases with decreasing N for all methods is due to increasing autoencoder reconstruction error.

Next, we fix $N = 400,000$ and increase τ from 0.25 to 1, 5, and 10. As mentioned earlier, here we need to generate new time series data that cover a longer time interval than the originally used time series data of $N = 400,000$ and $\tau = 0.25$. As seen in Figure 3.3(b), the autoencoder, MG-N, and NODE tracking errors do not vary much with τ. Across the three increased τ values, PI-DManD-B has up to 30% lower error than DManD. Note that the missing PI-DManD-c bar in Figure 3.3(b), replaced by an asterisk, has an error of about unity and is omitted to improve clarity in the vertical axis (and similarly for bar plots elsewhere). Note that a spacing of $\tau = 10$ corresponds to ≈ 0.5 Lyapunov times, and all approaches will ultimately fail as τ becomes comparable to the Lyapunov time; we find that

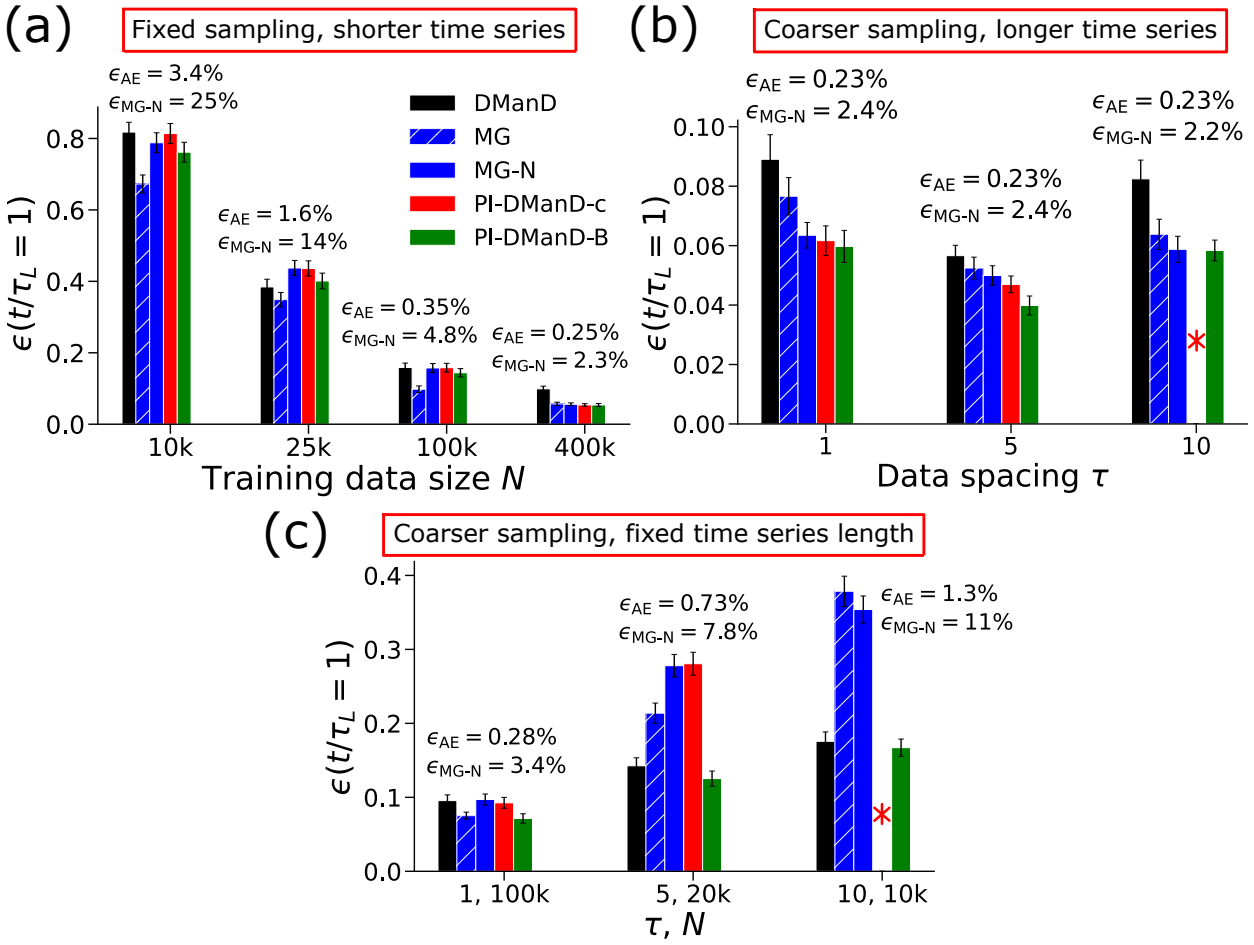


Figure 3.3: KSE, $L = 22$ (for both data and FOM). Short-time tracking error at one Lyapunov time for scarce data in terms of (a) less data with $N \leq 400,000$ and $\tau = 0.25$ (i.e. shorter time series), (b) widely spaced data with $\tau > 0.25$ and $N = 400,000$, and (c) increasing τ and decreasing N proportionally. Here and elsewhere, error bars show standard error, and all subplots follow the legend in (a). Note the different vertical axis limits for each subplot. Missing PI-DManD-c bars, denoted with asterisk, in (b) and (c) have errors of about 1 (i.e. much larger than the case shown) and are omitted to improve clarity in the vertical axis.

PI-DManD-c fails first. Unlike the previously considered scenario with just decreasing N, here we see sizable improvements of PI-DManD-B over DManD.

In the final subcase here, we simultaneously increase τ and decrease N by downsampling the original time series dataset ($N = 400,000$ and $\tau = 0.25$). In Figure 3.3(c), we show the tracking error for downsample factors of 4, 20, and 40; here, only PI-DManD-B has a tracking error that remains comparable to that of DManD (and is slightly lower in the case of $\tau = 1$, $N = 100,000$). Across these three subcases of data-scarce training, PI-DManD-B has equal or lower error than DManD while PI-DManD-c sometimes has a higher error than DManD.

3.3.3 Approximate FOM

An exact FOM is not available for many physical systems, either because of a lack of first-principles knowledge of the governing equations, or uncertainty in the parameter values used in the FOM. Here we test the robustness of our hybrid methods to model error by purposely using an incorrect domain length $L = 20$ in the FOM instead of the correct value of $L = 22$ used to generate the data (i.e. $\mu_{\text{FOM}} \neq \mu$). In other words, the model manifold vector field g_m is generated from f_{FOM} with incorrect parameter values. Figure 3.4(a) shows the space-time plots of example trajectories for $L = 22$ and $L = 20$. With the smaller domain length, the dynamics display intermittent/bursting-like behavior that is very different from that found at $L = 22$. To be clear, we never generate data with $L = 20$ for any training or testing of our models—we only use the $L = 22$ dataset. The parameter $L = 20$ comes into play only when we feed the $L = 22$ dataset into the $L = 20$ FOM.

To make the comparisons in this section, we use abundant training data ($N = 400,000$ and $\tau = 0.25$). Figure 3.4(b) shows the tracking error for all methods. Since DManD does not use any knowledge of the FOM, its short-time tracking performance is comparable to that in Figure 3.2(a) (these two NODE models for DManD only differ by the random weight initialization). As expected, the solely physics-informed MG and MG-N methods have poor

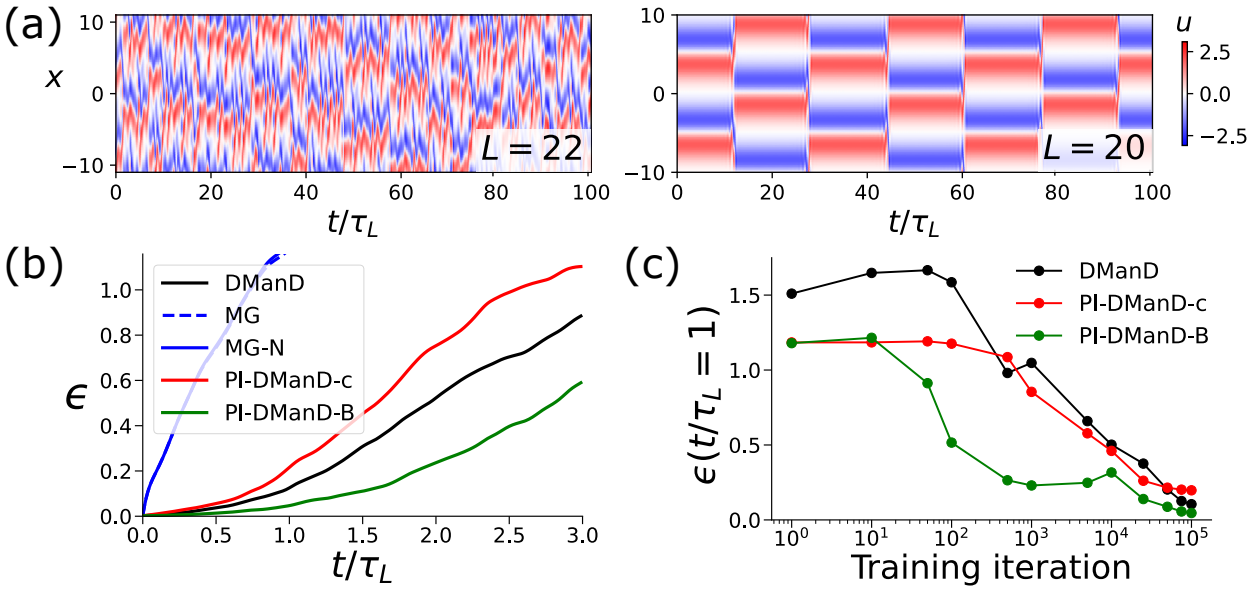


Figure 3.4: KSE, $N = 400,000$, $\tau = 0.25$. Effects of using an erroneous parameter ($L = 20$) in the FOM to model data generated with $L = 22$. (a) Ground truth trajectory (left) and example trajectory with erroneous parameter (right). (b) Ensemble-averaged short-time tracking error. (c) NODE training convergence for purely data-driven and hybrid methods.

short-time tracking due to only trying to learn the vector field in the latent coordinates from the FOM (see Equation 3.9) and not from the data. Since PI-DManD-c uses data to correct for FOM errors, its tracking error is lower than that of MG-N; as shown in Figure 3.4(c), the NODE training convergence for g_c of PI-DManD-c converges slower as compared to the exact FOM case in Figure 3.2, indicating that correcting FOM errors requires more training iterations than only correcting for numerical errors. While PI-DManD-c does not outperform DManD in short-time tracking, PI-DManD-B does. This result indicates that even if the FOM of the physics-based prior is highly erroneous, the hybrid PI-DManD-B ROM can still substantially outperform the solely data-driven DManD ROM in short-time tracking. The mechanism for this result is essentially transfer learning, where the initialization with a physics-based vector field (even if it is from an approximate FOM) is much better than a random guess when training a NODE. As for long-time statistics, both hybrid ROMs perform similarly to DManD (see Section 3.4.2).

3.4 Additional information

3.4.1 Training and testing speed for methods applied to KSE

Here, we examine the cost of training and evaluating each method. As shown in Figure 3.2(d), the NODE g_c learned by the PI-DManD-c model trains orders of magnitude faster than the DManD model—PI-DManD-c requires only about 10 training iterations for convergence, while DManD requires $> 10^4$ iterations (PI-DManD-B also takes a similar number of iterations as DManD). However, the PI-DManD-c model adds g_c to the MG-N model g_{mn} (see Equation 3.12), so the time to obtain an MG-N model needs to be considered in addition to the NODE training time. We define training time for the hybrid ROMs as the time needed to obtain a trained MG-N model plus the time needed to train the NODE. We also define training time for MG-N as the network training time plus the time for preprocessing the MG vector field output data from the data of the vector field on the state space (0.1 hours for KSE). Table 3.2 shows the training time for all methods. One can see that the total time for training either hybrid methods is greater than that of DManD; however, as seen in the last row of Table 3.2, MG-N and the hybrid methods are comparable to DManD in regards to how fast a test trajectory can be evaluated for 100 Lyapunov times. While MG requires no training time to obtain the vector field on the manifold, it is over an order of magnitude slower to evaluate than the other methods—as discussed in Section 3.2.2, the reason is that MG needs to decode h, solve the FOM, and evaluate a Jacobian (which involves an encoding step) for each function evaluation of g_m. So, MG is clearly not a ROM. Although the MG-N and hybrid methods require a longer time to train than DManD, they all take a similar amount of time to evaluate/test and have the added benefit of substantially better short-time tracking of the true solution.

	DManD	MG-N	MG	PI-DManD-c	PI-DManD-B
Train time (hr)	2.2	8.3	n/a	13.1	12.0
Test time (s)	8.3	8.2	102	11.4	7.7

Table 3.2: Comparing computation time for methods applied to KSE. First row is time (in hours) for training. Second row is computation time (in seconds) for testing, where an initial condition is integrated for 100 Lyapunov times. Calculations were run on an Intel Xeon E5-2640 CPU.

3.4.2 Long time statistics results for KSE

Besides the short-time tracking results described in Section 3.3.1, we also gauge model performance based on how well a predicted trajectory stays on the true attractor at long times. Since for KSE at $L = 22$ the invariant manifold for the long-time dynamics is 8-dimensional, we simplify the comparison of the long-time statistics by choosing two relevant quantities and calculating a joint probability distribution function (pdf). The time-evolution of the "energy" $\overline{u^2}/2$, where $\bar{\cdot}$ denotes average over the spatial domain, has a growth ("production") term proportional to $||\overline{u_x^2}||_2$ and a decay ("dissipation") term proportional to $\overline{u_{xx}^2}$, where the subscript x denotes spatial derivative. Based on this observation, we use the joint pdf of u_x and u_{xx} to characterize the attractor. We calculate joint pdfs using the entire testing dataset (100,000 snapshots spaced by $\tau = 0.25$ time units), where model predictions use the first snapshot as an initial condition. Figure 3.5(a) shows the joint pdfs for long-time trajectories corresponding to the ground truth, DManD, and PI-DManD-c. The joint pdfs for PI-DManD-B and the two physics-informed methods are not shown since they are virtually identical to the joint pdfs that are shown. To quantitatively compare the pdfs of the model predictions to the pdf of the ground truth trajectory, we calculate the earth mover's distance (EMD), also known as the Wasserstein-1 distance [181]. Let μ and ν be two discrete distributions calculated from normalized histograms with n_μ and n_ν bins, respectively. The EMD between these two distributions is the minimal cost of transforming one distribution

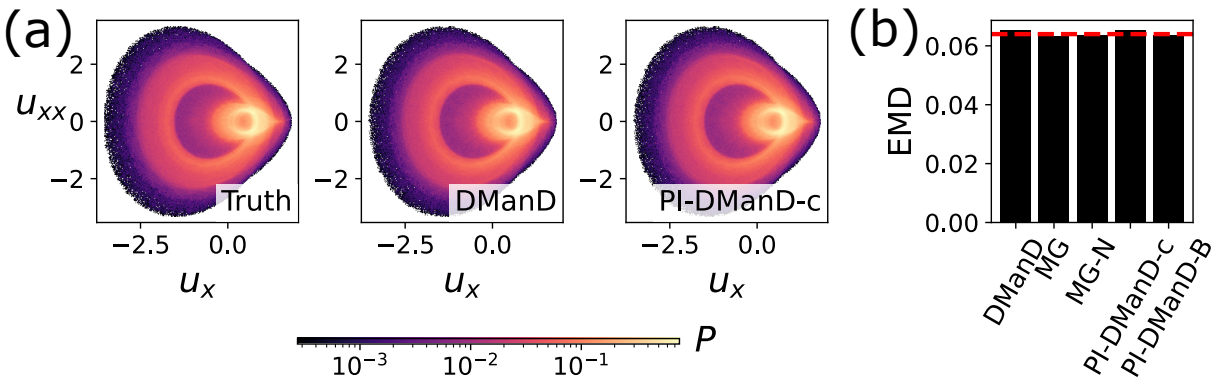


Figure 3.5: KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. (a) Comparison of joint pdfs. (b) Earth movers distance (EMD) quantifying the difference between pdfs of the ground truth and model predictions, where the dashed red line is the baseline EMD calculated with the pdf of another ground truth trajectory. Joint pdfs for models not shown look similar to joint pdfs of models with similar EMD values.

to the other, and is given by

$$\mathrm{EMD}(\mu, \nu) = \min_{\gamma} \sum_{i,j} \gamma_{i,j} \left\| \mu^{(i)} - \nu^{(j)} \right\|_2 \tag{3.24}$$

$$\text{s.t.} \quad \gamma \mathbf{1} = \mu, \quad \gamma^\top \mathbf{1} = \nu, \quad \gamma \geq 0\,, \tag{3.25}$$

where $\mathbf{1}$ is a column vector of ones and $\gamma \in \mathbb{R}^{n_\mu \times n_\nu}$ is a transport plan with $\gamma_{i,j}$ dictating the amount of probability mass to move from bin $\mu^{(i)}$ to bin $\nu^{(j)}$. The EMD represents the minimal work (amount of mass moved multiplied by distance) needed to execute the optimal transport plan. Throughout this paper, the EMD value is calculated with respect to the pdf of the ground truth trajectory in the testing dataset, which we call μ_t.

We first show the long-time statistics reconstruction for the case of abundant data and exact FOM (see Section 3.3.1). In Figure 3.5(b), we plot the EMD between μ_t and the model prediction pdf for all five methods. Note that the dashed red line in Figure 3.5(b) represents the baseline EMD, which is calculated between μ_t and the pdf of a ground truth trajectory in the training dataset. Since all methods have a EMD value similar to this baseline EMD, they perform equally well in capturing the long-time statistics, at least for the joint pdf quantities chosen here.

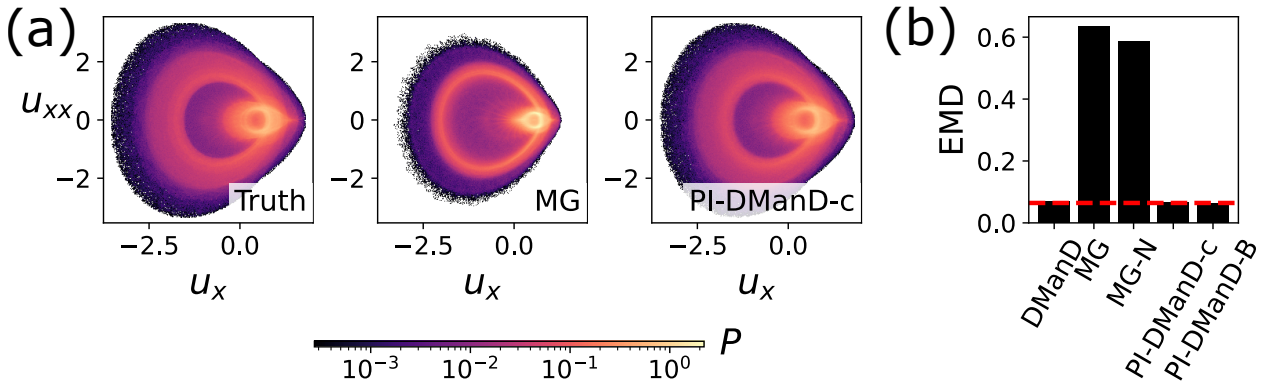


Figure 3.6: KSE, $N = 400,000$, $\tau = 0.25$. Effects of using an erroneous parameter ($L = 20$) in the FOM when making model predictions on the data generated with $L = 22$. (a) Comparison of joint pdfs. (b) Earth movers distance (EMD) quantifying the difference between pdfs of the ground truth and model predictions, where the dashed red line is the baseline EMD calculated with the pdf of another ground truth trajectory. Joint pdfs for models not shown look similar to joint pdfs of models with similar EMD values.

Next, we show the long-time statistics reconstruction for the case of abundant data and inexact FOM (see Section 3.3.3). In Figure 3.6(a), we plot the joint pdfs for the ground truth trajectory and the MG and PI-DManD-c model predictions. In Figure 3.6(b), we plot the EMD value (see Equation 3.25) between pdfs of a model prediction and the ground truth test trajectory; again, the baseline EMD (dashed red line in the figure) is calculated between two ground truth trajectories. (Pdfs for models not shown are similar to shown pdfs if they have similar EMD values.) Figure 3.6 shows that when given an erroneous parameter value for f_{FOM}, MG and MG-N will evolve the $L = 22$ dataset on a completely incorrect attractor. As for the two hybrid methods, they capture the long-time statistics as well as DManD.

3.4.3 Complex Ginzburg–Landau Equation

In addition to the KSE, we have studied PI-DManD for prediction of dynamics of the one-dimensional complex Ginzburg–Landau equation (CGLE)

$$\frac{\partial A}{\partial t} = A + (1 + i\alpha)\frac{\partial^2 A}{\partial x^2} - (1 + i\beta)|A|^2 A \ . \tag{3.26}$$

The CGLE has been used to model reaction-diffusion systems, superfluidity, and liquid crystals [2]. The quantities α and β are parameters. We use a domain length $L = 30$ and periodic boundary conditions. We set $\alpha = 2$ and $\beta = -2$, so when $L = 30$ the dynamics lie in the "defect-mediated turbulence" regime [205]. The solution is computed with an ETD-RK4 solver [102] in the Fourier domain. We used the Bennitin algorithm [10] to determine a Lyapunov time of 2.8 time units. With the initial transient removed, we generate 10^5 snapshots of $A \in \mathbb{C}^{64}$ discretized on $d_u = 64$ uniformly spaced grid points and separated by $\tau = 0.05$ time units (note, τ is five times smaller here for the CGLE compared to the KSE). To work with real numbers in our neural networks, we concatenate the real and imaginary parts of A to form $u \in \mathbb{R}^{128}$. We use u when calculating error quantities.

Determining manifold dimension

Unlike the KSE system considered previously, the dimension of the invariant manifold $d_{\mathcal{M}}$ for data on the attractor for this CGLE system is not known. To estimate it, we use an implicit rank-minimizing autoencoder with weight decay (IRMAE-WD) [97, 253]. IRMAE-WD uses an intermediate latent space $z \in \mathbb{R}^{d_z}$, where d_z acts as an upper bound on the manifold dimension (we use $d_z = 50$ here). The difference between IRMAE-WD and a normal autoencoder is the use of linear layers in the former, which act to drive the covariance matrix zz^T to have minimal rank $d_{\mathcal{M}} < d_z$. IRMAE-WD has been applied to the KSE in [253], where the singular value spectra of zz^T contains a sharp drop-off that spans over 10 orders of magnitude—this clear indication of the rank of zz^T then gives us an estimate of the manifold dimension $d_{\mathcal{M}}$.

We train an IRMAE-WD autoencoder on the CGLE system with weight-decay strength 10^{-5} and four linear layers. Figure 3.7 shows the singular value spectra, where the sharp drop-off covering about 10 orders of magnitude occurring at the 40th singular value index indicates that our data has $d_{\mathcal{M}} = 40$. The IRMAE-WD relative reconstruction error (calculated with Equation 3.22) is $\epsilon_{\text{AE}} = 0.61\%$. Note that we observed similarly sharp drop-offs at singular

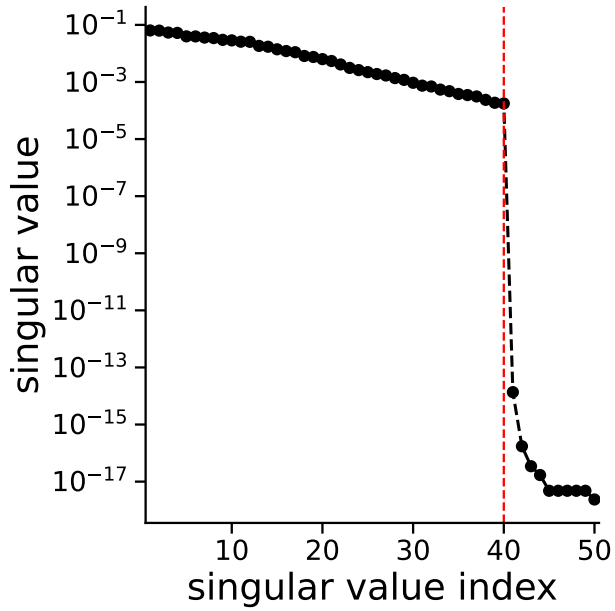


Figure 3.7: Singular value spectra of the covariance of the latent space z for IRMAE-WD applied to CGLE. Red dashed line located at 40th singular value index.

value indices ranging from 33 to 39 for 4 other IRMAE models. Therefore, we choose an estimate that is at the high end (40) of an ensemble of estimates.

Before using the IRMAE-WD in a ROM, we would need to project the intermediate latent space $z \in \mathbb{R}^{50}$ onto a latent space $h \in \mathbb{R}^{40}$ (e.g. with a singular value decomposition); however, doing so would make the encoder $\mathcal{E} : u \to h$ more complicated. Therefore, to simplify the encoder structure whose Jacobian $J_{\mathcal{E}}(h)$ must be computed for MG, we choose to instead train and use a normal autoencoder with latent dimension $d_h = 40$. The relative reconstruction error for this normal autoencoder is $\epsilon_{\text{AE}} = 0.57\%$, which is similar to that of the IRMAE-WD autoencoder. All of the training details—such as neural network architectures and learning rate scheduling—are identical to those for the KSE, which are shown in Table 3.1 and described in the body of Section 3.2.4.

Abundant data, exact FOM

We begin presentation of the results by paralleling the KSE results of Section 3.3.1, using the entire dataset of $N = 400,000$ training snapshots spaced apart by $\tau = 0.05$ time units

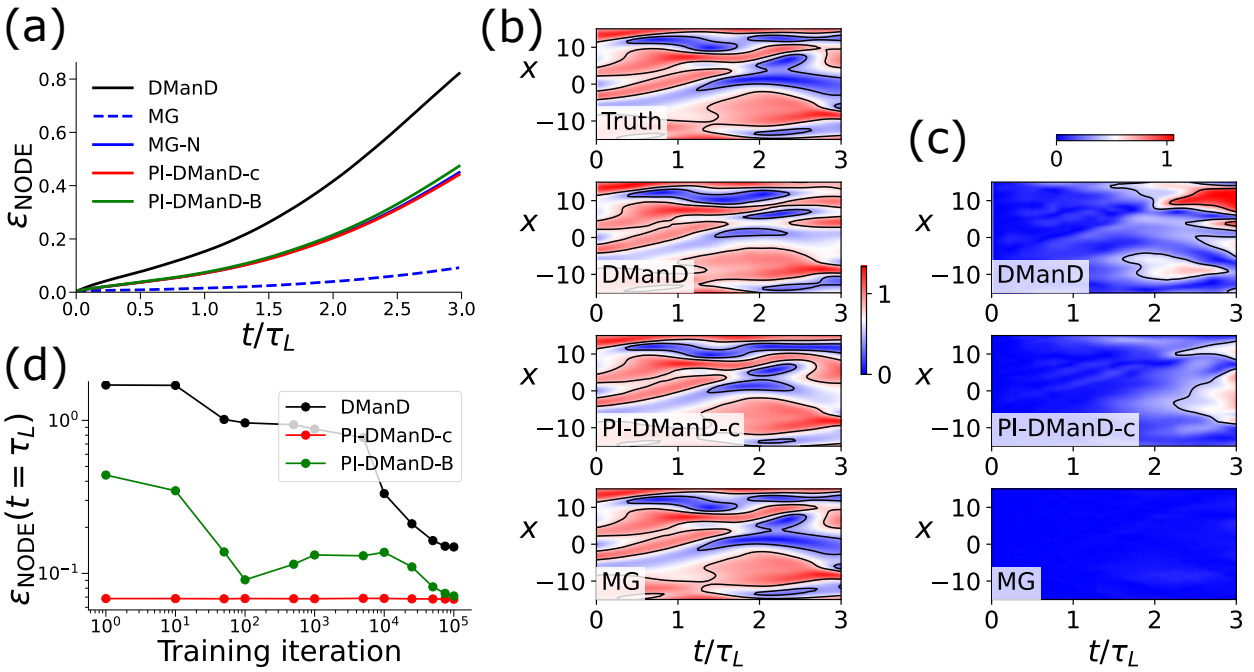


Figure 3.8: CGLE, $\beta = -2$ (for both data and FOM), $N = 400,000$, $\tau = 0.05$. (a) Ensemble-averaged short-time tracking error for all methods. (b) Absolute value of representative trajectories of ground truth and model predictions (contours at $|A| = 0.4, 0.8$). (c) Difference plots between ground truth and model predictions (contours at difference values of 0.4, 0.8). Representative trajectories and difference plots for other models look similar to those of PI-DManD-c. (d) NODE training convergence for purely data-driven and hybrid methods.

and using the same parameters $\alpha = 2$ and $\beta = -2$ for data generation and the FOM. Figure 3.8(a) shows the ensemble-averaged tracking error for all methods. The physics-informed MG-N method (whose curve is directly underneath the red PI-DManD-c curve in Figure 3.8(a)) and the two hybrid methods outperform DManD in short-time tracking; while this result is consistent with the KSE results in Figure 3.2(a), the primary difference here is that MG has dramatically lower short-time tracking error. As shown in the representative space-time plots in Figure 3.8(b) and the difference plots in Figure 3.8(c), MG is tracking the true solution with very small error for 3 Lyapunov times. For KSE, the short-time tracking error at one Lyapunov time is about 0.07 for both MG and MG-N; however, for CGLE, it is about 0.07 for MG-N and about 0.02 for MG. We believe that the gap in the error between MG and MG-N is larger for the CGLE case due to a higher error in approximating g_m with g_{mn} ($\epsilon_{\text{MG-N}}$ is 8% for CGLE and 2.3% for KSE).

	DManD	MG-N	MG	PI-DManD-c	PI-DManD-B
Train time (hr)	2.2	9.3	n/a	13.0	12.3
Test time (s)	4.9	5.0	218.0	6.8	4.5

Table 3.3: Comparing computation time for methods applied to CGLE. First row is time (in hours) for training. Second row is computation time (in seconds) for testing, where an initial condition is integrated for 100 Lyapunov times. Calculations were run on an Intel Xeon E5-2640 CPU.

The superior predictive performance of MG for the CGLE comes at a large cost. In Table 3.3, we show the training and testing time for all methods (analagous to Table 3.2 for the KSE). Here, MG still takes almost two orders of magnitude longer to evolve a trajectory. Figure 3.8(d) shows the NODE training convergence, where (like with the KSE) the NODE g_c learned by the PI-DManD-c model training orders of magnitude faster than the DManD model. However, when accounting for the training cost of the MG-N network g_{mn} (which includes 0.15 hours of preprocessing time), the total cost for training both g_c and g_{mn} for PI-DManD-c is greater than for DManD (see Table 3.3).

We now determine whether long-time trajectories of each method stay on the true CGLE attractor. As with the KSE, we will calculate a joint pdf to use as a simple metric for the long-time statistics. We again performed a kinetic energy ($|A|^2$) balance on the PDE, which yielded a production term $\overline{|A|^2}$ and a dissipation term $\overline{|A_x|^2 + |A|^4}$, where A_x is a spatial derivative; from these terms, we choose the joint pdf quantities as $|A|^2$ and $|A_x|^2$, respectively. The joint pdfs for the ground truth, DManD and PI-DManD-c model predictions are shown in Figure 3.9(a), while the EMD (see Equation 3.25) calculated between all methods and the ground truth trajectory is presented in Figure 3.9(b). As before, the baseline EMD for the dashed red line is calculated between ground truth trajectories with different initial conditions. All methods have a similar EMD to this baseline, so they perform equally well in capturing the long-time statistics of these quantities.

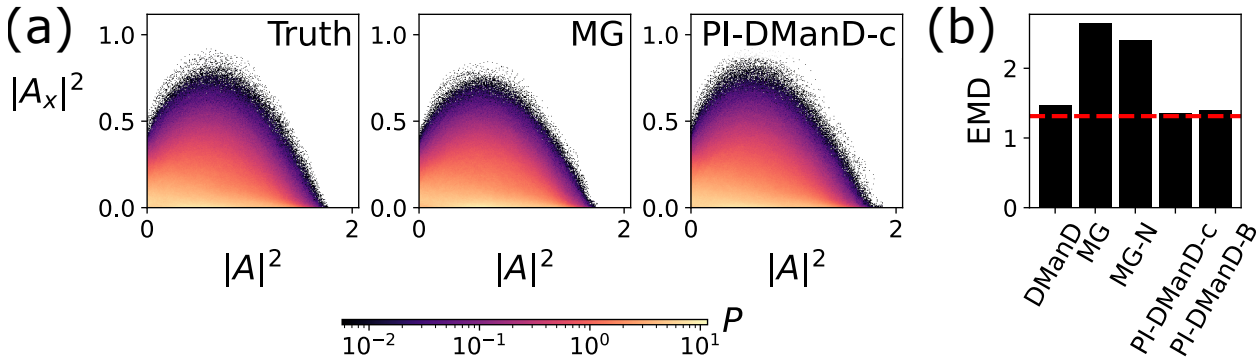


Figure 3.9: CGLE, $\beta = -2$ (for both data and FOM), $N = 400,000$, $\tau = 0.05$. (a) Comparison of joint pdfs. (b) Earth movers distance (EMD) quantifying the difference between pdfs of the ground truth and model predictions, where the dashed red line is the baseline EMD calculated with the pdf of another ground truth trajectory. Joint pdfs for models not shown look similar to joint pdfs of models with similar EMD values.

Scarce data, exact FOM

Here, we parallel the KSE results of Section 3.3.2. As with the KSE, we use the exact FOM and study the effect of having less data (training data size $N < 400,000$), data spaced further apart ($\tau > 0.05$), or both simultaneously.

We first fix the data spacing at $\tau = 0.05$ and vary the training data size from $N = 400,000$ to 100,000, 25,000, and 10,000 (i.e. shorter time series). Figure 3.10(a) shows the ensemble-averaged tracking error ϵ at one Lyapunov time for decreasing N (the case of $N = 400,000$ corresponds to Figure 3.8). Similar to the KSE, here the NODE test error ϵ increases with decreasing N for all methods due to increasing autoencoder reconstruction error ϵ_{AE}. The autoencoder errors ϵ_{AE} and MG-N errors $\epsilon_{\text{MG-N}}$ here are higher than they were for the KSE, but the NODE errors ϵ are about the same for both datasets. Another difference is that for $N = 100,000$, PI-DManD-B (but not PI-DManD-c) is able to achieve error that is substantially lower than that of MG-N, which suggests the hybrid ROM is correcting for MG-N test error.

The second scenario is using a fixed number of training snapshots $N = 400,000$ and increasing the data spacing from $\tau = 0.05$ to 0.2, 1, and 2 (i.e. use time series that are longer and sampled less often as τ increases). Figure 3.10(b) shows the errors for this scenario.

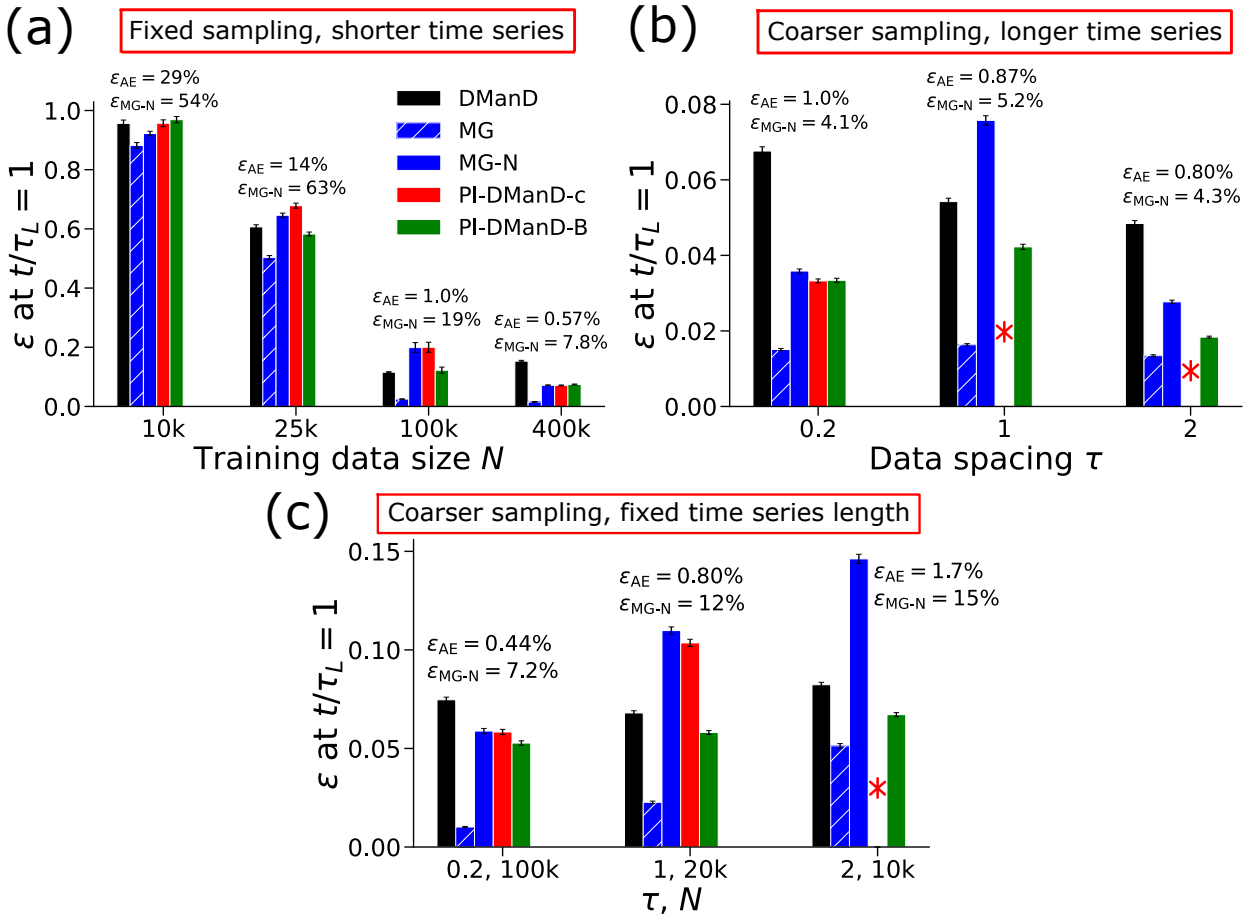


Figure 3.10: CGLE, $\beta = -2$ (for both data and FOM). Short-time tracking error at one Lyapunov time for scarce data in terms of (a) less data with $N \leq 400,000$ and $\tau = 0.05$ (i.e. shorter time series), (b) widely spaced data with $\tau > 0.05$ and $N = 400,000$, and (c) increasing τ and decreasing N proportionally. Error bars are standard errors, and all subplots follow the legend in (a). Note the different vertical axis limits for each subplot. Missing PI-DManD-c bars, denoted with asterisk, in (b) and (c) have errors of about 1 and are omitted to improve clarity in vertical axis.

As with the KSE, here the autoencoder, MG-N, and NODE errors do not vary much with τ. Note that the missing PI-DManD-c bars in Figure 3.10(b), denoted with asterisk, have an error of about 1 and are omitted to improve clarity in the vertical axis. The reason is that $\tau = 1$ time units corresponds to about one third of a Lyapunov time, so PI-DManD-c struggles in short-time tracking in this case and for $\tau = 2$. Across the three increased τ values, both hybrid ROMs have lower error than DManD (except $\tau = 1$ and 2, where only PI-DManD-B does).

Lastly, we study the effect of both increasing the data spacing τ and decreasing the number of training snapshots N (i.e. downsampling the original time series of length $N = 400,000$ and $\tau = 0.05$). In Figure 3.10(c), we show the tracking error for data spacing $\tau = 0.2$, 1, and 2 with N decreasing proportionally. The tracking error ϵ increases for all methods with increasing τ due to the increasing autoencoder error ϵ_{AE}. (As with the KSE, the PI-DManD-c error for $\tau = 2$ and $N = 10,000$ is around 1 and is omitted from Figure 3.10(c) to improve the clarity in the scaling of the vertical axis.) As with the KSE, the tracking errors for MG, MG-N, and PI-DManD-c become larger than that of DManD. For the KSE results in Figure 3.3(c), the PI-DManD-B error remained comparable to DManD error for two out of the three values of τ and was slightly less than the DManD error for the other value of τ; here, however, PI-DManD-B has somewhat lower error than DManD for all three values of τ. As with the KSE, across the three cases of data-scarce training, PI-DManD-B always has equal or lower error than DManD while PI-DManD-c sometimes has a higher error.

Approximate FOM

We now test how well the hybrid methods can perform despite being informed with a FOM containing erroneous parameters. The results here parallel the KSE results in Section 3.3.3. We create a hybrid ROM for evolving the dataset generated with the correct parameters $\alpha = 2$ and $\beta = -2$ when provided a FOM f_{FOM} with $\beta = -1.8$ or $\beta = -1$ (holding

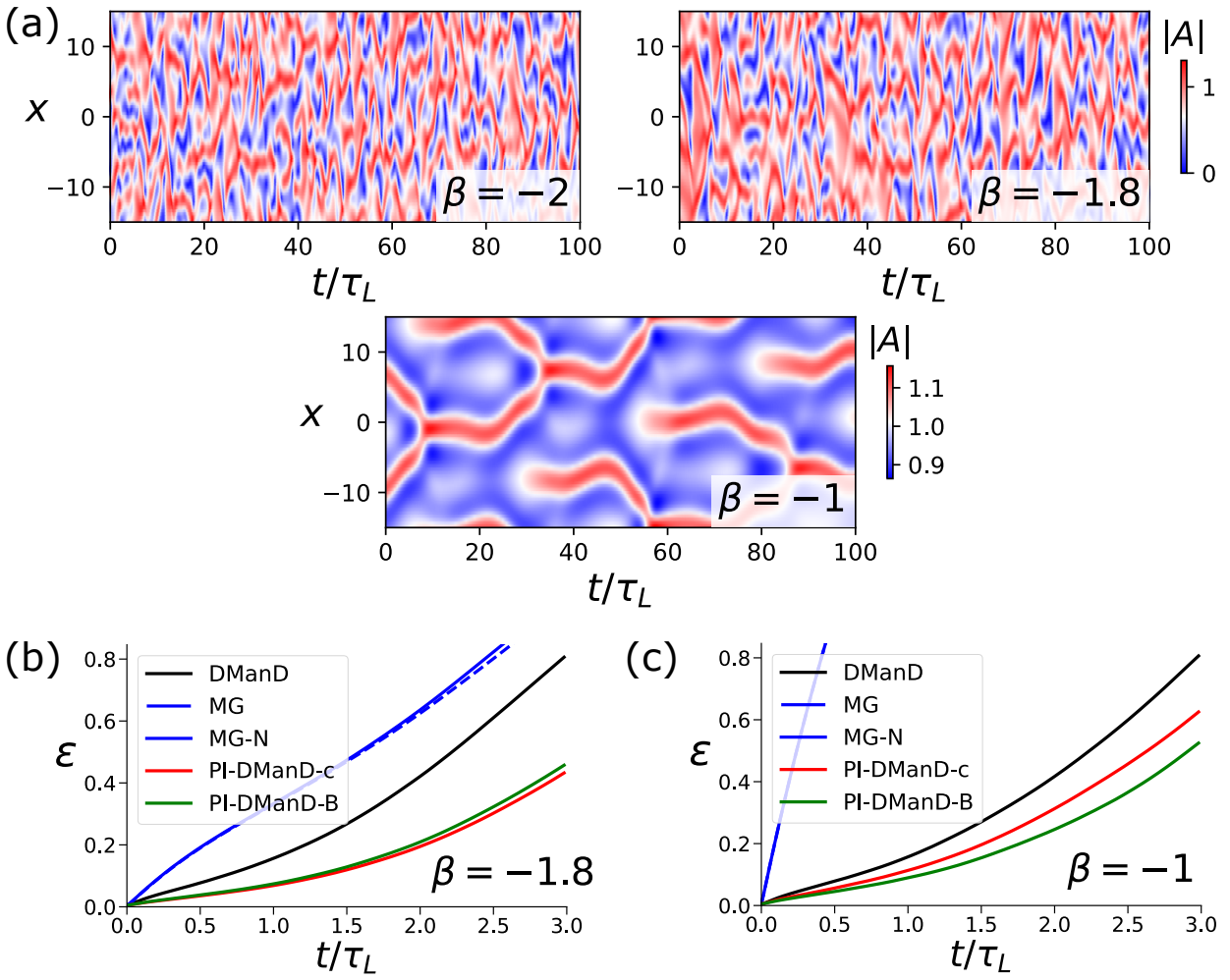


Figure 3.11: CGLE, $N = 400,000$, $\tau = 0.05$. Effects of using an erroneous parameter ($\beta = -1.8$ or $\beta = -1$) in the FOM to model data generated with $\beta = -2$. (a) Ground truth trajectory (top-left) and example trajectories for the two erroneous β values (top-right and bottom); note the different colorbar axis limits. Ensemble-averaged short-time tracking error when using $\beta = -1.8$ (b) or $\beta = -1$ (c) in the FOM.

$\alpha = 2$). For both $\beta = -2$ and $\beta = -1.8$, the dynamics lie in the defect-mediated turbulence regime [205]; however, for $\beta = -1$, the dynamics instead lie in the phase-turbulence regime [205]; example trajectories for these three values of β are shown in Figure 3.11(a). It bears repeating that the two erroneous β values are only used in the $f_{\text{FOM}}(u; \mu_{\text{FOM}})$ term appearing in Equation 3.9 for MG (which also affects g_{mn} learned by MG-N)—no dataset is generated with the erroneous β values.

We first compare short-time tracking performance. Figures 3.11(b) and 3.11(c) show the

tracking error of all methods for $\beta = -1.8$ and $\beta = -1$, respectively. Unsurprisingly, the value of β used in the FOM does not affect the predictive capability of DManD, since it does not use any prior knowledge of the physics. As was the case with KSE $L = 20$ in Figure 4, MG and MG-N here also poorly track the true solution due to being informed with erroneous FOM parameters. (Note, the curves for MG and MG-N in Figure 3.11(c) level off to around 1.5, and the vertical axis limits are shrunk to provide clarity.) As the accuracy of the prior decreases (β strays further away from -2), the tracking error for PI-DManD-c and PI-DManD-B grow slightly but still remain lower than the error for DManD; however, the tracking error for PI-DManD-B is noticeably lower than that of PI-DManD-c for $\beta = -1$, which is in line with the KSE results in Section 3.3.3 where only PI-DManD-B maintained a lower tracking error than DManD. Importantly, across both spatiotemporally chaotic datasets, the results show that including knowledge of the physics (even if inaccurate) helps in creating hybrid ROMs (PI-DManD-B especially) that perform better than solely data-driven ROMs (DManD) in short-time predictions.

Lastly, we investigate the long-time statistics of the model predictions by calculating the joint pdf of quantities $|A|^2$ and $|A_x|^2$. We plot the joint pdfs for the ground truth trajectory and the MG and PI-DManD-c model predictions in Figure 3.12(a) for $\beta = -1.8$. The EMD values (see Equation 3.25) between pdfs of a model prediction and the ground truth test trajectory are provided in Figure 3.12(b). (Again, the baseline EMD represented by the dashed red line in a bar plot is calculated between two ground truth trajectories, and pdfs for models not shown are similar to shown pdfs if they have similar EMD values.) For $\beta = -1$, we report (but not show to reduce redundancy) that the EMD values for MG and MG-N were larger than 40 while the EMD values for the other three methods were as close to the baseline EMD of around 1.3 as for the $\beta = -1.8$ case. When given an erroneous parameter value for the FOM, the two physics-informed MG and MG-N methods evolve the $\beta = -2$ dataset with a completely incorrect vector field, leading to poor reconstruction of long-time statistics—this observation is more pronounced for the case of $\beta = -1$. As for the

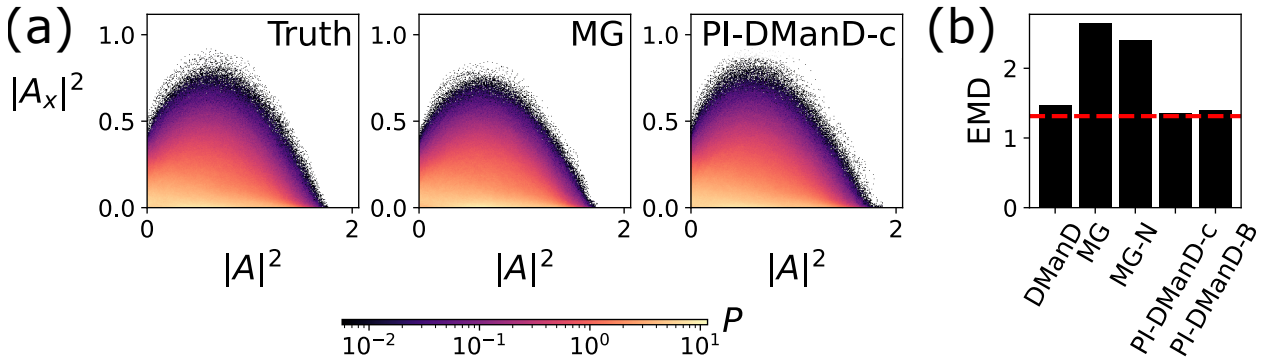


Figure 3.12: CGLE, N=400,000, $\tau = 0.05$. Effects of using an erroneous parameter ($\beta = -1.8$) in the FOM to model the data generated with $\beta = -2$. (a) Comparison of joint pdfs for $\beta = -1.8$. (b) Earth movers distance (EMD) quantifying the difference between pdfs of the ground truth and model predictions for $\beta = -1.8$. Dashed red line is the baseline EMD calculated with the pdf of another ground truth trajectory. Joint pdfs for models not shown look similar to joint pdfs of models with similar EMD values.

two hybrid methods, they capture the long-time statistics of these quantities about as well as DManD.

3.4.4 Integrating Manifold Galerkin (MG) models

The short time tracking (STT) for MG varies sometimes with autoencoder model due to the back-and-forth decoding and encoding (note, none of the other methods have this issue). For KSE, we found that the STT for MG can only be as good as the STT for MG-N; for CGLE, Figure 3.8 shows that the STT for MG can be better than MG-N (while not shown, we do find some MG models trained on different autoencoder models which have worse STT than even DManD). The floor in the MG STT being lower for CGLE could be that the effects of chaos are less apparent when integrating the MG-based ROM of CGLE. However, MG-N and the other hybrid methods (whose STT vary negligibly with autoencoder models) have a STT error of about $\epsilon = 0.07$ at one Lyapunov time for both datasets, which seems to be the floor in the STT for these non-MG-based ROMs. The MLP test errors—ϵ_{MLP} of 1.2% for KSE and 8% for CGLE—reflect this observation. For KSE, the short-time tracking error at one Lyapunov time is about 0.07 for both MG and MG-N; however, for CGLE, it is about 0.07 for MG-N and about 0.02 for MG. We believe that the gap in the error between MG

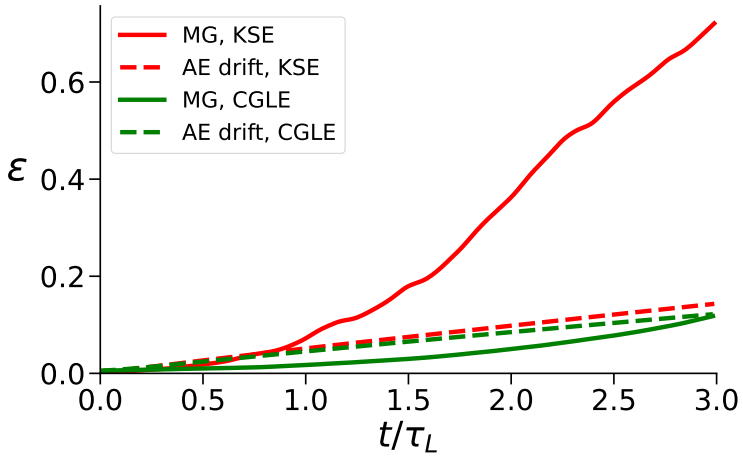


Figure 3.13: KSE and CGLE, correct parameters for both data and FOM, $N = 400,000$, $\tau = 0.25$ for KSE and $\tau = 0.05$ for CGLE. Comparing short-time tracking of MG model to "autoencoder drift", which is calculated by encoding and decoding an ensemble of initial conditions without time integration.

and MG-N is larger for the CGLE case due to a higher error in approximating g_m with g_{mn} ($\epsilon_{\text{MG-N}}$ is 8% for CGLE and 2.3% for KSE).

We plot the MG STT error for KSE and CGLE together in Figure 3.13 along with the "autoencoder drift", which quantifies the build-up of error solely due to encoding and decoding. The integration time-step size is 0.25 time units for KSE and 0.05 for CGLE; "autoencoder drift" does not involve time integration, but is plotted on the same axis by performing a sufficient number of back-and-forth encodes and decodes that corresponds to 3 Lyapunov times. Recall that at each function evaluation of g_m, MG performs a decoding step, solves the FOM, and performs an encoding step to calculate $J_{\mathcal{E}}$; therefore, the autoencoder drift is essentially MG but without the FOM evaluation and time integration. Figure 3.13 indicates that the STT error attributed to autoencoder error is the same for both datasets, but CGLE is less chaotic than KSE since the STT error increases slower.

For MG, the RHS of the full-order model (FOM) f is projected onto the latent space coordinates h using a Jacobian to give the model $g_m(h) = J_{\mathcal{E}}(\mathcal{D}(h))f(\mathcal{D}(h)))$. One can use $J_{\mathcal{D}}^{\dagger}$ instead of $J_{\mathcal{E}}$, but calculating the pseudo-inverse of $J_{\mathcal{D}}$ is more expensive. Moreover, as seen in Figure 3.14, using $J_{\mathcal{D}}^{\dagger}$ can lead to the MG model blowing up. Figure 3.14 also shows

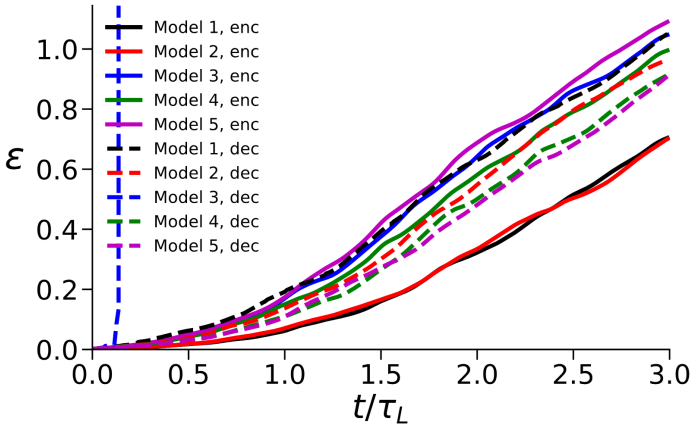


Figure 3.14: Ensemble short-time tracking (STT) error for 5 MG models, each from a different autoencoder model. Autoencoder models are trained with different weight initializations and have nearly the same test reconstruction error ϵ_{AE} of 0.22%. "Enc" means the Jacobian of the encoder is used, and "dec" means the pseudoinverse of the Jacobian of the decoder is used. RK4 with step size 0.25 time units is used as the integrator.

that even when using $J_{\mathcal{E}}$, there is inconsistency with how the model performs—different autoencoder models with nearly the same test reconstruction error lead to MG models that have different short-time tracking (STT) performance. Integrating MG models comes with two challenges, which are described in detail below.

The first challenge is choosing an integrator. The training and testing of all NODE models, as well as the testing of MG-N models, are done with the dopri5 adaptive integrator in `torchdiffeq` [23] with default relative tolerance 10^{-7} and absolute tolerance 10^{-9}. However, we had to use the RK4 integrator with time step size of 0.25 time units for testing the MG models. The reason is that when dopri5 with default tolerances is used to integrate MG, the integrator takes hundreds of adaptive steps per integration time step. The tolerance can be lowered to speed up the computation, but the solution will change based on the chosen tolerance. We confirmed that the RK4 scheme with integration step size of 0.25 time units for KSE and 0.05 time units for CGLE provides converged results (smaller timesteps do not change the MG solution) and more consistent computation time. We also tried stiff integrators, in case the Jacobian is unable to smooth out the stiff directions of the FOM provided by the stiff PDE. However, both the implicit BDF method and LSODA method

(mix of BDF and Adam's) we tried also took a prohibitively long time to integrate.

The second challenge with integrating MG models is the sensitivity of STT error to the latent space coordinates h found by the autoencoder, as seen in Figure 3.14. Multiple autoencoders with different weight initializations—despite them all having similar test reconstruction errors—lead to MG models with varying STT performance (while not shown, the long-time statistics do not vary for the models that do not blow up). This variation in STT performance occurs regardless of which Jacobian we use and which integrator we use. We emphasize that this inconsistency in STT performance only occurs with MG—MG-N and hybrid methods derived from MG-N all provide consistent STT performance (within standard error bars of the error ϵ) across different autoencoder models. The most likely contribution to the variation of STT performance with autoencoder model seen in MG could be the back-and-forth decoding and encoding that MG does at each function evaluation, which leads to a compounding of error that changes depending on the specific set of latent space coordinates generated by the autoencoder. One thing we tried was to train an autoencoder with an additional forecasting penalty term in its loss:

$$\mathcal{L} = \langle \|u(t) - \mathcal{D}(\mathcal{E}(u(t)))\|_2^2 \rangle + \lambda \langle \|u(t+\tau) - \mathcal{D}(h(t+\tau))\|_2^2 \rangle \,, \tag{3.27}$$

where

$$h(t+\tau) = h(t) + \int_t^{t+\tau} g_m(h(t'))dt' \,. \tag{3.28}$$

However, the training loss did not decrease, likely due to the backpropagation having to go through many encodes and decodes and the MG model g_m.

One empirical way we found that made MG somewhat more consistent across different model initializations was to increase the L2 weight regularization strength (we do so for other methods whenever comparisons are made). However, larger L2 regularization leads to somewhat poorer MSE for these other methods. Therefore, while a large L2 regularization were used to generate consistent results for all models for the purpose of comparison, we

recommend using smaller L2 regularization if one is solely trying to obtain a good DManD or PI-DManD-B model. Overall, while the issue with integrating MG models is not well understood, we do know that the model variation in MG does not affect MG-N or either PI-DManD model at all.

3.5 Noise robustness

In our methods, the autoencoder is trained on data living on the invariant manifold. Therefore, if: 1) the FOM is exact, 2) the autoencoder is exact, and 3) the data is noise-free, then the full vector field will be tangent to the manifold, thus allowing the manifold dynamics to be learned. Generally, some or all of these three conditions do not hold. That is why we update the manifold vector field from MG/MG-N with a neural ODE correction based on the dynamics of the data. In this section, we test our methods to the real-world scenario in which data contains observation noise. We expect the physics-based information provided to PI-DManD will aid in evolving the dynamics on the underlying invariant manifold. Critically, all training and testing of our models will be done only with noisy data—there will be no access to clean data whatsoever.

Autoencoders offer an avenue to denoising signals, as the latent space retains only the essential information of the input data. Denoising autoencoders [119, 159, 255] are given noisy input data and trained to reconstruct clean output data; as such, denoising autoencoders require access to clean training data, which is not always available in real-world situations. In [220], enhanced noise robustness is achieved by posing the ODE in the weak-form (which moves the derivatives of the state space onto derivatives of test functions) and then training a SINDy model. Relatedly, [75] trains a SINDy model with noise robustness by geometrically constraining the vector field in the state space (which does not necessarily live on the manifold) onto the manifold through a normal correction term. The challenges in the case of large amounts of noise is finding a mapping to the underlying manifold is more difficult,

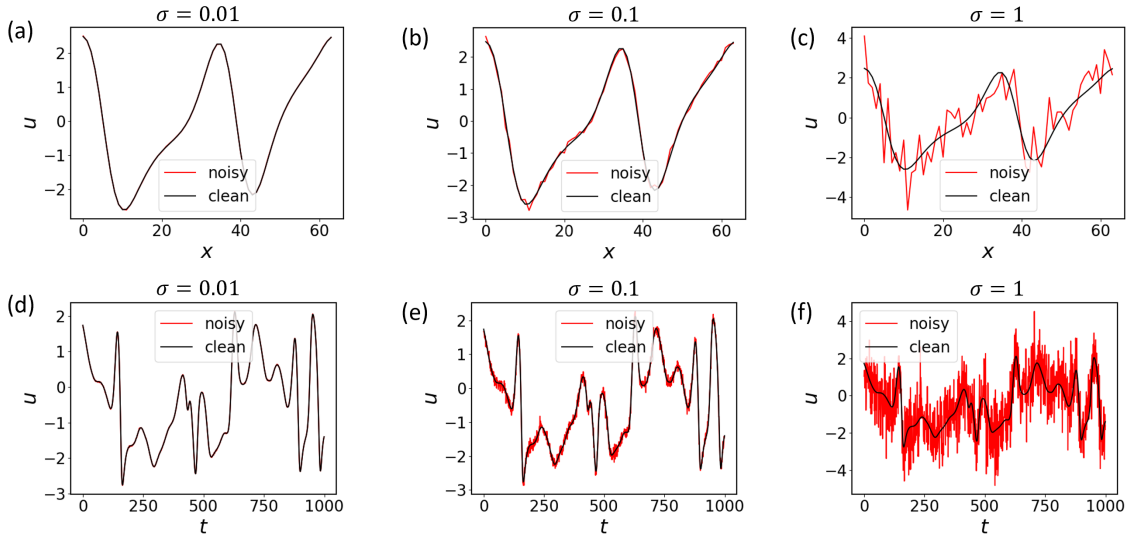


Figure 3.15: KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. Noisy vs. clean signal in space (a)-(c) and in time (d)-(f), where the strength of the noise σ increases from left to right.

as well as staying on the manifold during time integration. Here, we train an autoencoder and assume it gives an accurate enough mapping to the manifold to then train a DManD and PI-DManD-B model.

We work with the KSE in the abundant data case with exact physics/FOM, just like in Section 3.3.1, and all training details are the same as well unless stated otherwise. We add Gaussian white noise, with zero mean and variance σ^2, to the spatial and temporal directions of the entire training and testing dataset. We use values of 0.01, 0.1, and 1 for σ. Figures 3.15(a)-(c) show a snapshot for the three varying levels of noise in space, and similarly in time in Figures 3.15(d)-(f).

Before proceeding to train an autoencoder, we first preprocess the data by filtering it in time. We use a butterworth filter of order 4 with cutoff frequency at the leg or bend in the PSD where the spectrum transitions from decreasing to flat (the higher frequencies to the right of this bend correspond to the noise and not the signal). Figures 3.16(a)-(c) show the autoencoder reconstruction in space, and similarly in time in Figures 3.16(d)-(f). A latent space dimension of $d_h = 8$ was used for all three autoencoders trained here. Figures 3.16(a)-(c) depict the autoencoder error, ϵ_{AE}, which increases with σ. Upon comparing Figure 3.15

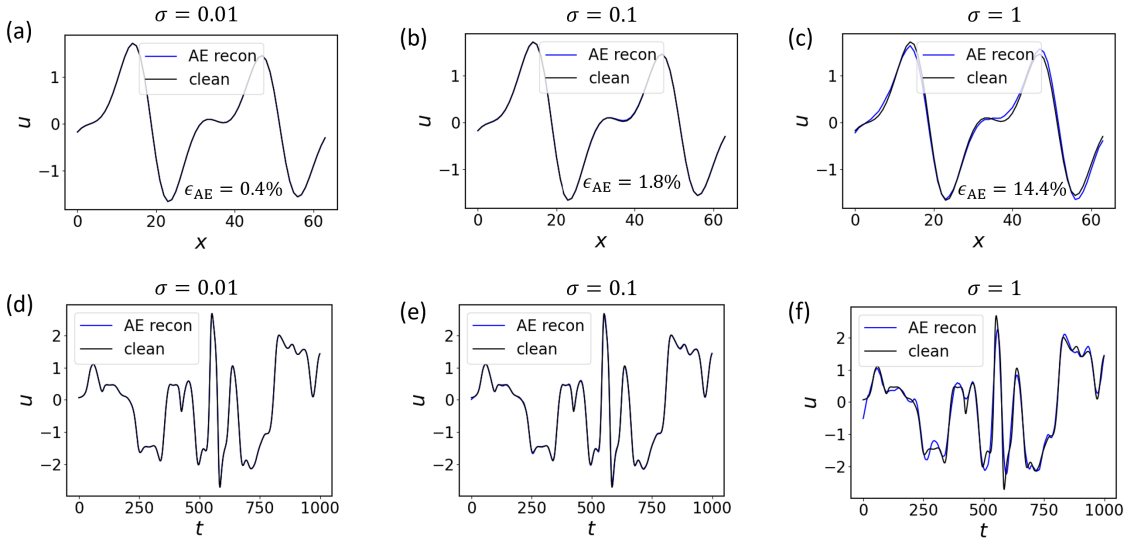


Figure 3.16: KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. Autoencoder reconstruction of the noisy signal vs. the underlying clean signal in space (a)-(c) and in time (d)-(f), where the strength of the noise σ increases from left to right.

with Figure 3.16, one can see that the filtering in time reduces noise in time, while the passing of the data through the autoencoder reduces noise in space.

The autoencoder error is quite high in the case of moderate ($\sigma = 0.1$) and large ($\sigma = 0.1$) noise, and an accurate mapping onto the manifold is not expected for these cases. Nonetheless, we proceed with training time evolution models. We first compute MG vector field data g_m from Equation 3.9. Importantly, we use $\mathcal{D}(h)$ instead of u for the FOM calculation—while the difference between the two quantities is negligible even in the small noise case ($\sigma = 0.01$), the passing of u through the autoencoder is crucial to prevent the compounding of error in g_m. We then proceed to train an MG-N model and all other models. Figure 3.17 shows the tracking error for zero noise and the three noise levels for DManD and PI-DManD-B (MG-N and PI-DManD-c models blow up, so they are not shown). PI-DManD-B performs much better than DManD in the no noise case. For $\sigma = 0.01$ and $\sigma = 0.1$, PI-DManD-B performs better than DManD, especially in the latter case. For $\sigma = 1$, the PI-DManD-B training loss did not decrease, likely as a result of high MG-N test errors associated with the noise level having a similar magnitude as the data itself. Lastly, we note that the filtering in time is important for obtaining good predictions—without

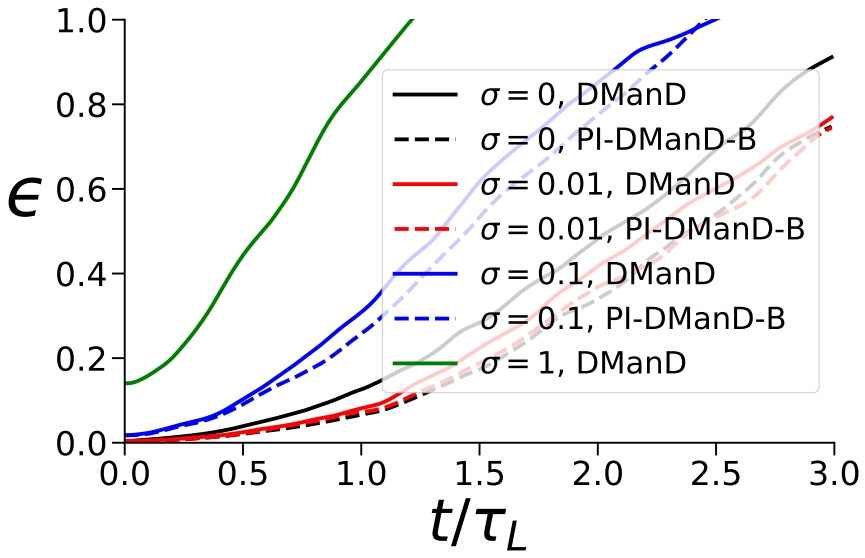


Figure 3.17: KSE, $L = 22$ (for both data and FOM), $N = 400,000$, $\tau = 0.25$. Ensemble-averaged short-time tracking error for DManD and PI-DManD-B with varying levels of noise σ.

filtering, the tracking error at one Lyapunov time was 0.75 for $\sigma = 0.1$ for both DManD and PI-DManD-B.

We end this section with a discussion of future work to obtain models with better noise robustness. One limitation of the methods is in the autoencoder step, where we knew *a priori* that the manifold dimension of the clean data is $d_h = 8$. As such, restricting the latent space dimension to $d_h = 8$ is implicitly allowing our autoencoders to denoise the data in space. A more principled and perhaps effective way to denoise without access to any clean training data is through robust autoencoders [256]. A robust autoencoder has ties to robust PCA in first splitting the dataset X into $X = L + S$, where L is the low-rank data matrix and S contains anomalies (noise or outliers). Then, an optimization problem solves

$$\max_{\theta_{\mathcal{E}},\theta_{\mathcal{D}}} \|L - \mathcal{D}(\mathcal{E}(L;\theta_{\mathcal{E}});\theta_{\mathcal{D}})\|_2 + \lambda\|S\|_1 \,, \tag{3.29}$$

$$s.t. \quad \|X - L - S\|_F = 0 \,, \tag{3.30}$$

where λ controls the amount of noise that is to be isolated in S. One limitation of robust autoencoders is the tuning of λ. We could combine a robust autoencoder with IRMAE, in

which noise is isolated in S and the linear layers in the encoder find the essential coordinates in the noise-free data matrix L. Another principled method for explicitly denoising without access to clean training data (which we can combine with IRMAE) is partitioned autoencoders [210], which isolates the noise into a partitioned subset of the latent space coordinates. Denoising would then involve setting that subset of the latent coefficients to zero and reconstructing just the clean signal. While partitioned autoencoders do not have any tuning of hyperparameters like with robust autoencoders, they do require labeled snapshots of noise-only training data, which may be difficult to generate/identify. Lastly, even we can obtain an effective denoising IRMAE model, there is still the issue that the vector field information calculated with MG (and approximated with MG-N) is easily corrupted by compounding error. One potential remedy for this issue is to introduce a weak-form formulation of the FOM as was done in [220] for SINDy, and also recently done with NODEs in [123].

3.6 Conclusion

In this work, we found that including physics-based prior information substantially improves the predictive capabilities of data-driven ROMs applied to systems with spatiotemporal chaotic dynamics, even if the physics-based information is not completely correct. We built upon previous work on a solely data-driven ROM called DManD [128]—which uses a neural ODE (NODE) approach to learn the dynamics on an invariant manifold whose coordinates are found with an autoencoder—by incorporating ideas from the Manifold Galerkin (MG) method [120]. MG expresses the RHS of the ODE in the latent space coordinates in terms of the RHS of the FOM (i.e. the physics). We approximated the MG vector field with a neural network (which we call MG-N) and combined it with DManD to develop two hybrid ROMs. The first, which we call PI-DManD-c, adds a correction term learned by a NODE to the MG-N vector field in order to account for numerical errors and inherent errors in the FOM.

The second hybrid ROM is a DManD model with an additional weight-decay-like term in the loss that drives the weights of a NODE network toward those of the MG-N network (and a transfer learning step from the latter to the former). This second hybrid ROM can be viewed as using MG-N as a Bayesian prior for training a NODE, and we call it PI-DManD-B (note, while the method is motivated using Bayes' theorem, it is not probabilistic).

We applied DManD and our hybrid ROMs (PI-DManD-c and PI-DManD-B) to the KSE and the 1D CGLE (results for CGLE are in Section 3.4.3), which are two PDE systems with spatiotemporal chaotic dynamics. We summarize the improvement of PI-DManD-B over DManD in making short-time predictions for three case studies in Table 3.4. The first case study is the baseline, where we use exact physics and abundant training data; there, PI-DManD-B (and PI-DManD-c) accurately tracked the true solution for substantially longer—at one Lyapunov time, the ensemble-averaged relative short-time tracking error is 0.1 for DManD and only 0.05 for the hybrid ROMs. The second case study reduced the amount of training data (for three subcases), and the third study involved providing substantially incorrect PDE parameters to the FOM used by MG/MG-N. For these last two studies, PI-DManD-B (but not PI-DManD-C) still provided sizable improvements over DManD, thus demonstrating its robustness to scarce data and FOM errors. While the total training time (MG-N and NODE) for PI-DManD-B is somewhat longer than PI-DManD-c, we recommend PI-DManD-B due to its better short-time tracking performance. Lastly, we trained our methods to mimic the real-world scenario of noisy measurements, where all neural networks were trained with abundant data (corrupted with noise) and exact physics. There, PI-DManD-B provided improved prediction over DManD due to the physics-based information aiding in learning the dynamics on the underlying invariant manifold. Future directions include testing robustness of our hybrid ROMs on complex fluid and flow systems for which only approximate first-principles models are available, and extending our methods to the case where we only have partial observations.

	Exact physics, abundant data	Exact physics, scarce data	Erroneous physics, abundant data
% reduction in relative trajectory error at $t = \tau_L$	**46%–51%** (Figs. 3.2(a), 3.8(a))	**None** (shorter time series, finer sampling; Figs. 3.3(a), 3.10(a)) **22%–62%** (longer time series, coarser sampling; Figs. 3.3(b), 3.10(b)) **6%–29%** (fixed time series length, coarser sampling; Figs. 3.3(c), 3.10(c))	**43%–64%** (Figs. 3.4(b), 3.11(b), 3.11(c))

Table 3.4: Summary of the improvement or reduction in $\epsilon(t = \tau_L)$ (the relative ensemble averaged short-time tracking error at one Lyapunov time) for the hybrid PI-DManD-B method over the solely data-driven DManD method; specifically, the quantities correspond to $(\epsilon_{\text{DManD}} - \epsilon_{\text{PI-DManD-B}})/\epsilon_{\text{DManD}} \times 100\%$. We varied the accuracy of the known physics (i.e. the full order model), the length of the time series (i.e. number of time units covered), and the sampling rate of the data.

4

Reconstructing fields from sparse sensors [1]

The reconstruction of fields from sensor data is a common task in scientific applications. In many cases, the observation is under resolved, and memory must be incorporated to uniquely recover the underlying system. For dissipative systems, dynamical systems theory provides guidance in formulating these data-driven reconstructions, which has been the focus of many optimization and machine learning approaches. Most models are restricted to data sampled at fixed positions and regular time intervals. We introduce a model which overcomes these limitations using attention mechanisms with spatial and temporal encodings. Our model is based on the Senseiver [200], which reconstructs fields from instantaneous sparse sensor measurements. Informed by time delay embedding theorems, we formulate an attention-based model that learns from sensor data at varying spatial position and sampling rates. We evaluate the model on systems exhibiting a limit cycle and spatiotemporal chaos.

[1] The text of this chapter is partially adapted from the publication by A. Guo, G. T. Craven, J. E. Santos, and C. D. Young, Reconstructing dissipative dynamical systems from spatially and temporally sparse sensors, *NeurIPS Workshop ML4PS*, 2024 [69]

4.1 Background on attention mechanism

Before moving to the main parts of this Chapter, we give a brief background on attention mechanisms, which is the backbone of the transformer architecture. Transformers were first developed by [224], and they now see widespread use in natural language processing, computer vision, and time series forecasting [59, 208, 236]. Transformers leverage attention mechanisms to learn long-range dependencies or correlations among a sequence of inputs. Unlike a RNN which only learns dependencies sequentially, attention mechanisms learn these dependencies globally.

Here, we provide an overview of dot-product self attention. Bold uppercase will denote matrices. Let the sequence of inputs be $\mathbf{X} \in \mathbb{R}^{d_s \times d_m}$, where d_s is the sequence length (number of tokens) and d_m is the model dimension. The input sequence $\mathbf{X}$ contains the tokens in its rows. Typically, an encoding is performed so that all tokens have length d_m (e.g., two words of different lengths are encoded into two tokens of the same length). In self-attention, the query and key, and value matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively, by multiplying the input sequence with weight matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$, respectively. The attention matrix $\boldsymbol{\alpha} \in \mathbb{R}^{d_s d_s}$ is calculated with

$$\boldsymbol{\alpha} = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_m}}\right), \tag{4.1}$$

and the attention output is given as

$$\hat{\boldsymbol{X}} = \boldsymbol{\alpha}\mathbf{V}\,. \tag{4.2}$$

The attention matrix component α_{ij} is the softmax, scaled inner product between the ith component of $\mathbf{Q}$ and the jth component of $\mathbf{K}$, and it acts as a weight for the value matrix $\mathbf{V}$. The attention output $\hat{\boldsymbol{X}}$ is usually calculated in multiple pieces or "heads" for parallelization purposes and then added to $\mathbf{X}$, normalized, fed through a feedforward network, and nor-

malized again. In practice, the sequence length of an input can vary; this issue is handled by fixing d_s and then either masking or padding the attention matrix. We have thus far described the input and output to a self-attention block. In the transformer, a self-attention block is typically combined with a cross-attention block to then output probabilities for a variety of tasks found in large language models.

4.2 Introduction

Inspired by success in reconstructing degraded images, neural networks (NNs) have been applied to system reconstruction in many scientific applications [18, 57, 190, 203, 232]. These problems are related to phase space reconstruction via Takens embedding theorem [212], which proves that for dynamics on an attractor, a time delay embedding of a scalar observable has a diffeomorphism to the underlying system. Data-driven applications of these approaches emerged shortly after using local principal components analysis [47] and multilayer perceptrons (MLPs) [192]. Progress in implementation of machine learning algorithms has led to significant advances in the speed and accuracy of reconstruction and forecasting models via reservoir computers (RCs) [143], convolutional NNs [57], Gaussian process regression [99], symbolic regression [8], long-short term memory [239, 259], and neural ODEs (NODEs) [169, 235, 247].

Despite this success, limited availability of data prevents direct application of most models. Data is often noisy, and sensors often drift in space or collect data intermittently. These pose challenges for models trained on clean data available at fixed positions and uniform intervals. The noise issue can be addressed by combining the forecast or reconstruction with model-free data assimilation methods [71]. Several approaches have addressed variable sensor positions and time sampling using NODEs and implicit neural representations [90, 245]. However, these approaches tend to rely on hyperparameter tuning and lack an underlying theoretical motivation. Time delay embeddings have been extended to the case of irregularly

sampled data by 'bundle embeddings' [99, 209], which essentially concatenate the sampling rate into the state representation.

Here, we develop a model for reconstructing the field values of a dynamical system with variable measurement intervals and sensor positions. The approach relies on the bundle embedding theorem [209], which implies the data should reside on a long-time 'inertial manifold' or 'strange attractor' of dimension $d_{\mathcal{M}}$ and embedding dimension $m \geq 2d_{\mathcal{M}}+1$ to ensure the existence of the reconstruction map. Positional and temporal encodings are implemented in an attention-based autoencoder as an extension of the Senseiver model [200]. Compared to the previous model, the time delay embedding improves reconstructions when the number of sensors is lower than the underlying manifold dimension. For low Reynolds number flow past a cylinder—a limit cycle—we demonstrate accurate reconstruction from a single sensor for: 1) a range of time sampling rates and 2) variable sensor positions (both cases done separately). We also compare the model to previous work using MLPs [247] and RCs [143] for reconstructing the Kuramoto-Sivashinsky equation (KSE) in a chaotic parameter regime.

4.3 Methodology

We build upon the attention-based Senseiver model [200] by incorporating time delay embeddings. Let s_i be the sensor measurement recorded at position x_i. For N_s sensor values, we form the time delay embedding $\boldsymbol{s}_d(t) \in \mathbb{R}^{N_s \times m}$, where m is the number of time delays. The matrix component $\boldsymbol{s}_d(i,j,t) = s_i(t-\tau_{ij})$ contains the jth delay of the ith sensor value, where t_{ij} and τ_{ij} are the corresponding sensor measurement time and delay spacing, t is the leading sensor measurement time, and $\tau_{ij} = t - t_{ij}$. The sensor positions $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^{N_s}$ are encoded with sines and cosines [200] to $\boldsymbol{a}_x$, where each physical coordinate value (scalar) is mapped to a vector of length 32. Each component in the delay spacing matrix $\boldsymbol{\tau}$ is encoded with a time2vec (t2v) encoding [107] to get $\boldsymbol{a}_t$ (see Section 4.5.1). This space-time encoding is fed as input along with the sensor value embedding $\boldsymbol{s}_d(t)$ to an attention-based encoder

E that maps to a latent space $\boldsymbol{Z}$; then, an attention-based decoder D maps $\boldsymbol{Z}$ and a query position $\boldsymbol{x}_q$ to get the field value $\hat{s}(t)$. In mathematical terms, the model is as follows:

$$\boldsymbol{a}_x = P_E(\boldsymbol{x}), \tag{4.3}$$

$$\boldsymbol{Z}(t) = E(\boldsymbol{s}_d(t), \boldsymbol{a}_x, \boldsymbol{a}_t) = E(\boldsymbol{s}_d(t), P_E(x_s), \mathrm{t2v}(\boldsymbol{\tau})), \tag{4.4}$$

$$\hat{s}(\boldsymbol{x}_q, t) = D(\boldsymbol{Z}(t), \boldsymbol{a}_q) = D(\boldsymbol{Z}(t), P_E(\boldsymbol{x}_q))\,. \tag{4.5}$$

Figure 6.2 shows a schematic of the model. We query the spatial domain point-by-point to get the fully reconstructed field $\hat{\boldsymbol{s}}$ at the current time t. The loss is $\langle \|s - \hat{s}\|^2 \rangle_t$, where the norm is performed over spatial points and the averaging is done over all training snapshots. In this study we use all spatial positions from the high resolution for training. However, the Senseiver output is pointwise, meaning the model can be trained with any level of downsampling in the ground truth data or choice of random seeding from a full field [64]. Additional details of the architecture is in Section 4.5.2.

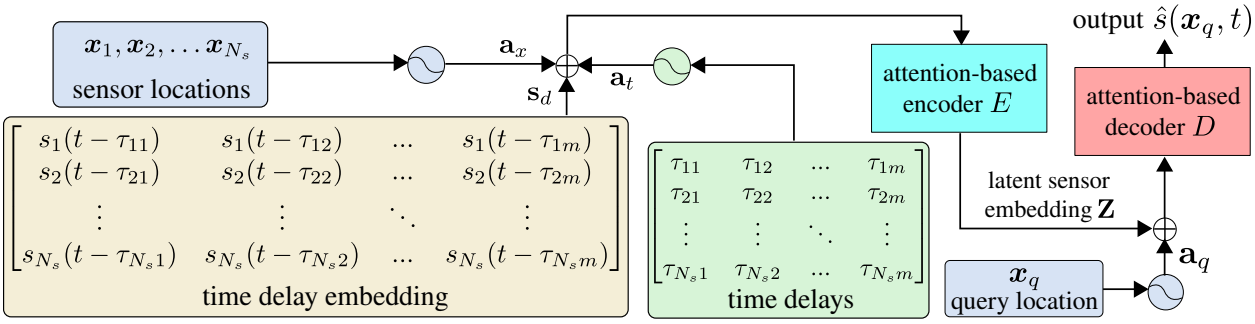


Figure 4.1: Architecture details. A time delay embedding of sensor values is concatenated with the positional and temporal encodings and encoded to a compressed latent space. The latent values are passed to the decoder with a query position to reconstruct the field at the current time.

4.4 Results

4.4.1 Non-uniform sampling in time

We first test the Senseiver on 2D flow past a cylinder at Re = 100, which settles onto a limit cycle with manifold dimension $d_{\mathcal{M}} = 1$. We reconstruct the steady state vorticity field ω from a single sensor with a variable sampling rate. The dataset consists of 5000 snapshots that cover ≈ 4 vortex shedding periods, of which 50 are used for training and the rest used for testing. During training and testing, the snapshots are variably spaced apart by 50 to 500 frames. As seen in Figure 4.2(d), the original Senseiver with no delay embeddings ($m = 1$) fails to capture the limit cycle, producing a mean L2 reconstruction error of $\epsilon = \|\omega - \hat{\omega}\|_2 / \|\omega\|_2 = 0.58$. This is expected, as we cannot prove the reconstruction map exists for $m < 2d_{\mathcal{M}} + 1$. Following Takens theorem, we select the embedding dimension $m = 3$. Concatenating the variable delay spacings τ_{0j} for $j = 1, 2$ to the $m = 3$ embedding of the sensor values yields a lower error of $\epsilon = 0.12$ (see Figure 4.2(e)). Upon using a t2v encoding of τ_{0j}, the error further decreases to $\epsilon = 0.05$ (see Figure 4.2(f)). As seen in Figure 4.2(c), the test error remains low for each pair of delay spacings τ_{0j} when using the method of t2v (while not shown, the heatmap plot of ϵ is similar for the other $m = 3$ method involving concatenation). We interpret the t2v encoding improvement as an increase of expressiveness of the model in predicting with a range of delay spacings rather than a single 'optimal' choice of delay coordinates.

4.4.2 Variable sensor position

We now test the model's robustness to sensor positions not seen during training. We work with uniformly sampled data here, so the positional encoding is still used but the temporal encoding is not. During training and testing, we use one sensor and its 3 time delays (spaced apart by 100 frames) as input to the model. 16 sensor positions in the vortex shedding region with high variance in the sensor values [57] are chosen for training. During testing,

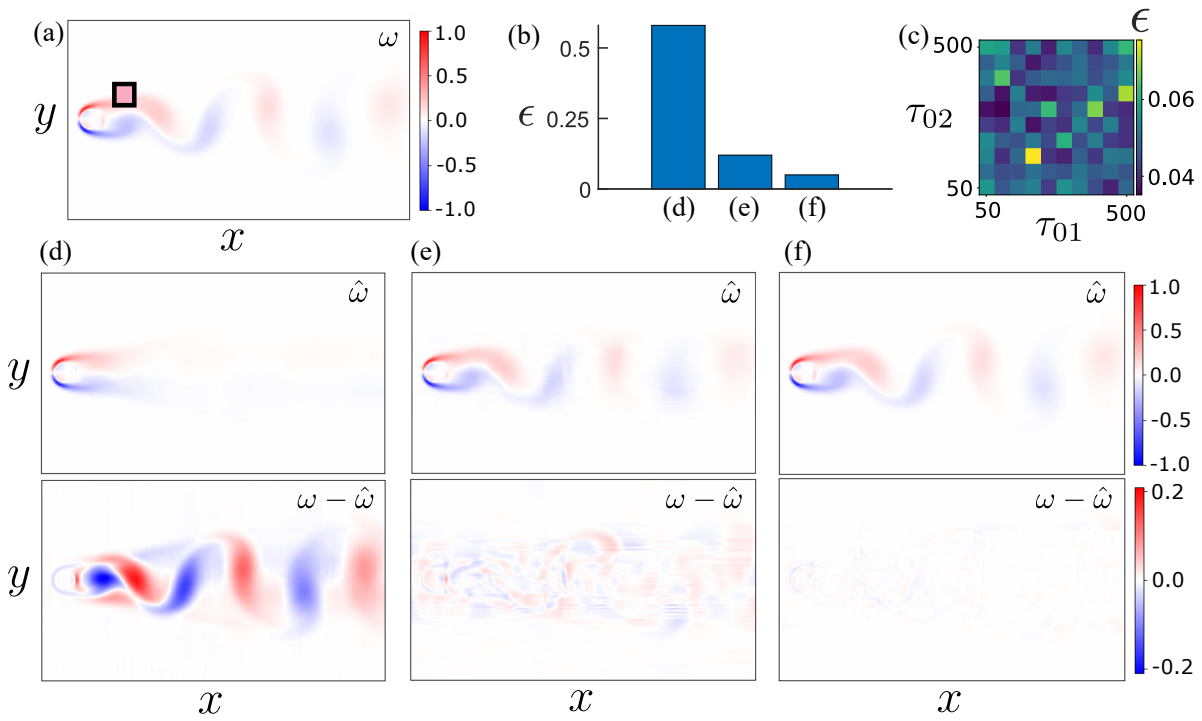


Figure 4.2: Field reconstruction of flow past a cylinder from sensor measurements that are non-uniform in time. (a) True vorticity field ω, with location of sensor marked with a square. (b) Reconstruction error for methods shown in panels (d)-(f). (c) Reconstruction error across each pair of delay spacings τ_{0j} for method shown in panel (f). Reconstruction fields $\hat{\omega}$ for models trained with (d) no time delay embedding, (e) embedding dimension $m = 3$ with temporal encoding of τ_{0j} via concatenation, and (f) $m = 3$ with temporal encoding of τ_{0j} via t2v are shown in the top panels of each subfigure. Difference in true and reconstructed fields are shown in the respective bottom panels of (d)-(f). In each field image, bounds on the y-axis are $[-9, 9]$ and the x-axis are $[0, 32]$ in dimensionless units.

we individually evaluate the model for each and every point in the domain. As seen in Figure 4.3(a), the test error is generally lower for testing locations at or near the 16 training locations (see Figures 4.3(b,c)). The high error at points exterior of the vortex-shedding region is due to the signals exhibiting small variance at all times. While the error is high for most out-of-training points interior of the vortex-shedding region, Figures 4.3(d,e) indicate that the model still reconstructs data on the true attractor but with a phase shift from the ground truth. The phase inaccuracy is likely a result of the chosen delay spacing not being optimal for all sensor positions.

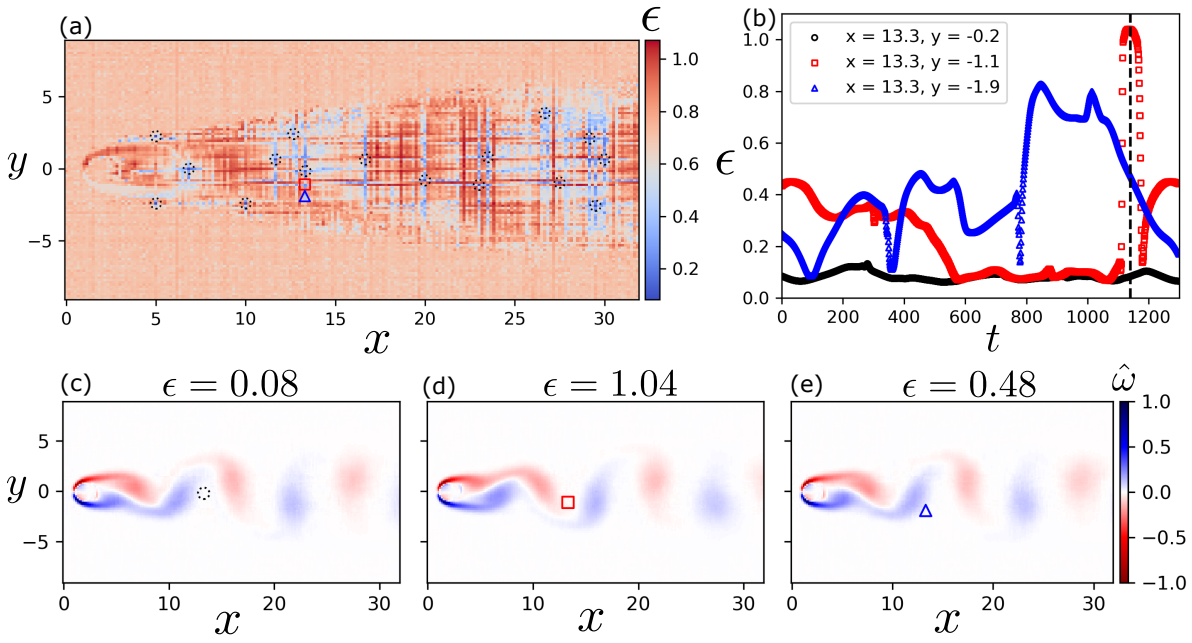


Figure 4.3: (a) Time and space averaged reconstruction test error for using a given sensor position as model input; dashed blue circles indicate training positions. (b) Error for each frame in a vortex shedding period for three sensor locations; vertical line corresponds to the snapshot time below for panels (c)-(e), which visualize the reconstruction at the selected locations. Panel (c) is a training position $x \approx 14, y \approx 0$, whereas (d) $y \approx -0.5$ and (e) $y \approx -1$ are test positions.

4.4.3 Chaotic systems

Lastly, we apply our model to the KSE, which is a 1D partial differential equation exhibiting diverse dynamical behavior. The domain size is $L = 22$ with periodic boundary conditions [34], for which there is sustained chaos on an inertial manifold of dimension $d_{\mathcal{M}} = 8$ [38, 130, 252]. We compare Senseiver to previous work using MLPs with time delays [247] and reservoir computing [143]. For all three methods, the training data size is kept constant at 60,000 with uniform time interval $\Delta t = 0.25$, the test data size is 10^5, and the reconstruction error ϵ is calculated for spatial points not corresponding to sensor positions. We examine the case of 1, 2, 4, and 8 sensors (which are fixed and evenly-spaced) and adjust m so that $N_s m = 16 = 2d_{\mathcal{M}}$ with $\tau = 8\Delta t$. We list ϵ in Table 4.1 for the three methods. For all but the 8-sensor case, our Senseiver model with time delays achieves a lower error than the other two methods. Moreover, our model does so with fewer parameters, which aids in computational efficiency during training and testing.

Our method also generalizes well to non-uniform time spacing (not shown), achieving an error of $\epsilon = 0.12$ for $N_s = 4$ and $m = 4$ delays with spacings ranging from $2\Delta t$ to $16\Delta t$. These results will reported in further detail along with a study of variable sensor positions in a forthcoming full publication.

Table 4.1: Reconstruction error for Kuramoto-Sivashinsky equation (KSE). For all methods except reservoir computing (RC), the number of delays m and sensors N_s are set such that $N_s m = 16$.

Method	$\epsilon\,(N_s = 1)$	$\epsilon\,(N_s = 2)$	$\epsilon\,(N_s = 4)$	$\epsilon\,(N_s = 8)$	parameters
Senseiver	**0.68**	**0.37**	**0.13**	0.05	$\mathbf{2 \times 10^4}$
MLP [Young et al. 2023]	0.70	0.44	0.15	**0.03**	3×10^5
RC [Lu et al. 2017]	0.91	0.68	0.31	0.10	2×10^5

4.5 Additional information

4.5.1 Time2vec encoding

The time2vec encoding [107] maps a scalar τ to a vector

$$\mathrm{t2v}(\tau)[k] = \begin{cases} \omega_k \tau + b_k\,, & \text{if } k = 0 \\ F(\omega_k \tau + b_k)\,, & \text{for } 1 \leq k \leq K-1 \end{cases},$$

where $\mathrm{t2v}(\tau)[k]$ is the kth component of $\mathrm{t2v}(\tau) \in \mathbb{R}^K$, F is an activation function, and ω_k and b_k are learnable parameters. We choose F to be the sine function and K to be 32. The t2v encoding allows both cyclic and non-cyclic information to be encoded.

4.5.2 Architecture details

Here we provide additional details on the attention-based encoder and decoder blocks in Figure 6.2, which are based on the architecture in the original Senseiver model [200]. We

use the scaled dot product attention mechanism [225]

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_m}}\right)\mathbf{V}\,, \tag{4.6}$$

where d_m is the model dimension of query matrix $\mathbf{Q}$, key matrix $\mathbf{K}$, and value matrix $\mathbf{V}$.

Multi-head attention applies a set of N_H linear transformations on $\mathbf{Q},\mathbf{K}$ and $\mathbf{V}$ and computes the attention mechanism on each linear transformation. Then, the outputs from each head are concatenated and passed through a final linear transformation to form the output:

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathbf{O}_1 \oplus \cdots \oplus \mathbf{O}_{N_H}]\,\mathbf{W}_M \tag{4.7}$$

$$\text{where } \mathbf{O}_h = \text{Attention}(\mathbf{Q}\mathbf{W}_h^Q, \mathbf{K}\mathbf{W}_h^K, \mathbf{V}\mathbf{W}_h^V)\,. \tag{4.8}$$

For the time-delayed Senseiver model, the sequence length is $d_s = N_s m$, since each sensor has m delays. The model dimension is $d_m = N_I + 2N_f N_d + K$, where N_I is the number of channels of the sensor measurements (one for the cases considered in this paper), N_f is the number of pairs of sine-cosine frequencies used in the positional encoding for each of the N_d number of spatial directions, and K is the length of the time2vec encoding.

The attention-based encoder processes the input $\mathbf{E}^{(0)} \in \mathbb{R}^{d_s \times d_m}$ with a linear layer to give an intermediate output $\mathbf{E}^{(1)} \in \mathbb{R}^{d_s \times N_c}$, where N_c is the number of latent channels. Next, a multi-head cross attention is performed between a trainable query array $\mathbf{Q}_{\text{in}} \in \mathbb{R}^{N_{Q_{\text{in}}} \times N_c}$ and key and value matrices derived from $\mathbf{E}^{(1)}$ to yield $\mathbf{E}^{(2)}$; here, $N_{Q_{\text{in}}}$ is the latent sequence length that is held constant to allow the attention calculations to be independent of N_s and m. $\mathbf{E}^{(2)}$ is then processed with multi-head self-attention to give the latent space $\mathbf{Z} \in \mathbb{R}^{N_{Q_{\text{in}}} \times N_c}$. The attention-based decoder performs similar steps as the encoder (see [200]), but includes a final MLP layer to reduce the dimension and obtain the predicted measurement $\hat{s}$ at the queried location x_q and current time t.

4.6 Attention mechanisms for time evolution

Here, we describe preliminary results on using attention mechanisms for time evolution of time delay coordinates. The merging of NODEs and transformers or attention mechanisms has seen some interest [27, 93, 254]. In the results here, we will adapt the easy attention method used in [199]. Easy attention foregoes learning the query and key matrices, and it instead directly learns the attention matrix $\boldsymbol{\alpha}$ and uses it in Equation 4.2. The result is not simply a linear MLP composed of matrix $\mathbf{W}_V$ multiplied with $\boldsymbol{\alpha}$, as will be evident below when describing the architecture.

We work with the KSE dataset with all details for the dataset and training as described in Section 4.4.3. The time delay embedding is $u_d(t) \in \mathbb{R}^{N_s m}$, where we consider number of sensors $N_s = 4$ and number of time delays $m = 4$. We aim to learn the RHS of

$$\frac{du_d}{dt} = g(u_d)\,. \tag{4.9}$$

The architecture is as follows. We calculate a t2v encoding of u_d of length 15 and concatenate it with the sensor values u_d to obtain a matrix $\mathbf{R} \in \mathbb{R}^{N_s m \times 16}$. We use multihead attention on $\mathbf{R}$ (see Equation 4.8) with two heads and easy attention (allow $\boldsymbol{\alpha}$ to be learnable). The attention output $\hat{\mathbf{R}} \in \mathbb{R}^{N_s m \times 16}$ is then concatenated with $\mathbf{R}$ (note, self-attention usually adds the two quanties together, but we find concatenation to improve results), which we then flatten and process with a 4-layer MLP to output the approximation of the RHS $g_d \in \mathbb{R}^{N_s m}$. The NODE prediction is then given by

$$\hat{u}_d(t+\tau) = u_d(t) + \int_t^{t+\tau} g_d(u_d(t); \theta_{g_d})\,\mathrm{d}t\,, \tag{4.10}$$

with loss function

$$\mathcal{L} = \langle \|\hat{u}_d(t+\tau) - u_d(t+\tau)\|_2^2 \rangle\,. \tag{4.11}$$

To summarize, we use easy attention to directly learn the dependencies between temporal

encodings of a time delay embedding and process the attention output with an MLP to be used as the RHS approximation. We call this procedure attention-based NODE or a-NODE.

The left side of Figure 4.4 shows the ensemble short-time tracking error for a normal NODE vs. an a-NODE, which are comparable between the two methods. The right side of Figure 4.4 shows the 1D pdf for the partial observable u_p calculated across the four sensor locations (which is allowed due to homogeneity in space), which are also comparable between the two methods. Future work involves understanding and improving the role of easy attention (and other architecture details) in providing enhanced time evolution of time delay embeddings.

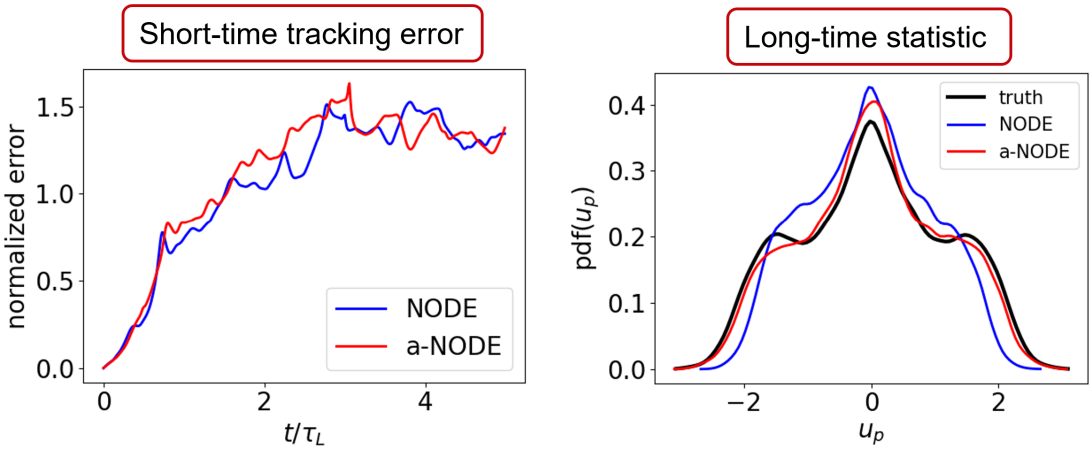


Figure 4.4: Model performance of a NODE compared to an attention-based NODE (a-NODE) in terms of ensemble-averaged short-time tracking (left) and long-time statistics quantified through a pdf of the partially-observed state (right).

4.7 Conclusion

We developed an attention-based model for reconstructing the fields of dissipative dynamical systems from sensor measurements that are spatially sparse and irregularly sampled in time. We expanded on the Senseiver [200] by allowing for time delay embeddings as the input. We first applied our model to flow past a cylinder, where we reconstructed the full vorticity field from just a single sensor with irregular time sampling. The model achieved low reconstruction

error by using a delay embedding dimension of 3 along with a time2vec encoding [107] of the delay spacings to account for the temporal intermittency of the sensor. Next, we tested the model's ability to reconstruct the vorticity field from variable sensor positions not seen during training. Facilitated by the spatial encoding of the sensor positions, the model was able to generalize to out-of-training sensor positions; however, there was some phase error in the reconstruction for test sensor positions located far from the training positions. Lastly, we applied the model to the spatiotemporally chaotic KSE for non-variable sensor positions and uniform time sampling. While our model achieved similar reconstruction performance to MLPs, the attention mechanism used in our model allows for the computational complexity (and thus number of parameters) to scale less strongly with the length of the space-time encoding, which can be quite long and expensive to process with MLPs. We've demonstrated the developed model is capable of reconstructing fields from spatially and temporally sparse sensors that also vary in location. Lastly, we showed preliminary results that demonstrate the potential of using attention mechanisms in NODEs for timeseries forecasting.

5

Characterizing 3D structures in turbulent channel flow

As discussed in Section 1.1, a detailed account on the shape of a representative attached eddy (as well as the range of length scales they are self-similar) remains inconclusive. In Chapter 2, we used DDWD to extract energetic, multiscale structures from 1D fluctuating streamwise velocity signals measured at different positions along the wall-normal direction y. There, the velocity structures—imprinted by attached eddy structures, and modeled with data-driven wavelets—were found to be self-similar in the streamwise direction x for each individual wall-normal position, and the shape of the structures did not vary much across wall-normal positions. In this chapter, we perform a more rigorous characterization of attached eddies by developing a wavelet-based 3D modal decomposition and applying it to 3D turbulent channel flow.

5.1 Introduction

Choosing an appropriate basis for each spatial direction in a 2D or 3D turbulent flow field is crucial. In statistically homogeneous directions, proper orthogonal decomposition (POD) reduces to a Fourier decomposition. In canonical wall-turbulent flows, the streamwise and

spanwise (z) directions are homogeneous, while the wall-normal direction is not. Therefore, POD has often been applied in the wall-normal direction. In [77], a decomposition of the cross-section of a fluctuating streamwise velocity field u in a pipe is performed with Fourier modes in the azimuthal (θ) direction (equivalently the spanwise direction) and POD modes in the radial (r) direction (equivalently the spanwise direction)

$$u(\theta, r, t) = \sum_{n} \sum_{k_\theta} a^{(n,k_\theta)}(t)\, \phi_n^{(k_\theta)}(r)\, \zeta_{k_\theta}(\theta)\,, \tag{5.1}$$

where $\zeta_{k_\theta}(\theta) = \exp(ik_\theta\theta)$ is the Fourier mode of wavenumber k_θ, $\phi_n^{(k_\theta)}(r)$ is the POD mode of modal index n that is also dependent on wavenumber k_θ, and $a_n^{(k_\theta)}(t)$ is the modal coefficient that is a function of time and dependent on the modal indices in both spatial directions. We use a superscript to denote dependence on a modal index. For example, POD modes $\phi_n^{(k_\theta)}(r)$ in [77] are calculated by computing the POD at each azimuthal wavenumber. In other words, the typical approach to performing a Fourier-POD decomposition is to perform the Fourier decomposition (either in 1 or 2 homogeneous directions) and calculate POD for each set of wavenumbers. Since POD modes give a complete basis, $\phi_n^{(k_\theta)}(r)$ are only mutually orthogonal under a 1D inner product if the modes are from the same POD decomposition (i.e., k_θ are the same); two POD modes from different POD decompositions (i.e., k_θ are different) are not orthogonal.

Before discussing other modal decompositions, we first formalize the various directions encountered within the discrete Fourier transform (DFT)-POD decomposition performed in [77]. Figure 5.1(a) shows a depiction of DFT-POD. In the left panel of this figure, we color a "POD direction" with blue, which we define as the direction spanned by the POD modes. We color a "homogeneous direction" with red, which we define as the direction that is typically decomposed with either Fourier or wavelet modes. We color an "ensemble direction" with pink, which we define as the direction in the snapshots needed to form the POD data matrices is taken from (with the example from [77] here, this is the time direction). In the middle

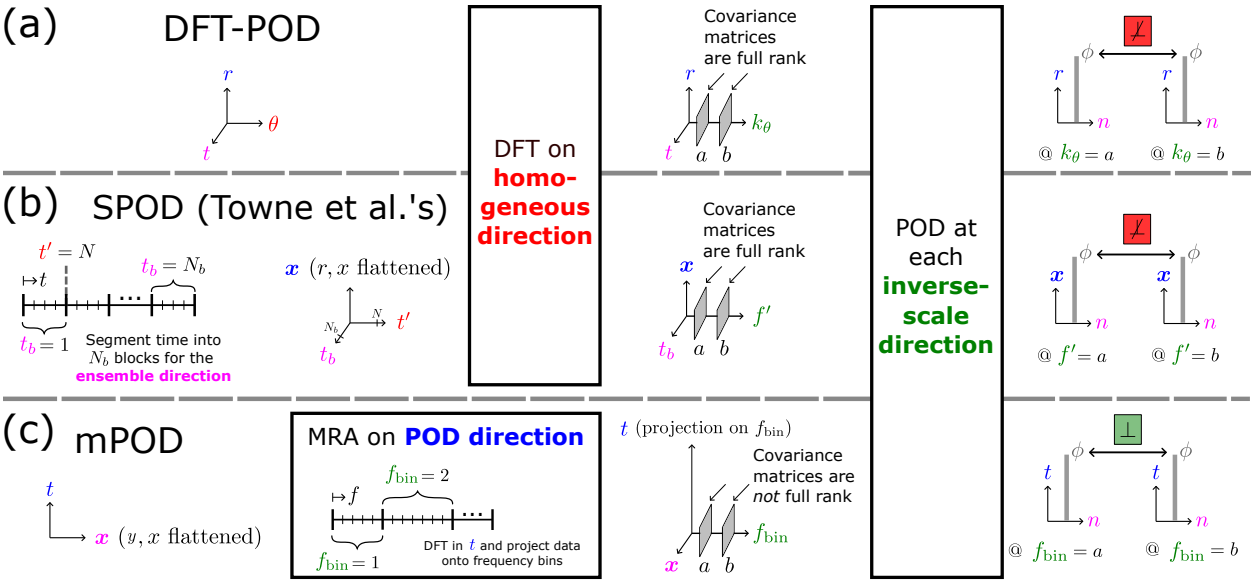


Figure 5.1: Sketch describing three POD-based decomposition techniques. Shown are (a) DFT-POD, (b) spectral POD (SPOD), and (c) mPOD (multiscale POD). The left panel shows the relevant directions, the middle panel shows the transformed directions after DFT/MRA is performed in the homogeneous/POD direction, and the right panel shows whether the POD modes across different inverse-scale directions are orthogonal under a 1D inner product.

panel of Figure 5.1(a), we show the resulting directions once a DFT is performed; there, we color an "inverse-scale direction" with green, which we define as the direction in which the scale changes. Since the covariance of the data matrix is full rank for each k_θ, the right panel of Figure 5.1(a) depicts how there is no 1D orthogonality of the POD modes across different scales. That said, the DFT-POD modes are still mutually orthogonal under a 2D inner product. We would like our modes to represent 3D structures. Therefore, orthogonality under a 3D inner product is sufficient; however, for the purpose of observing self-similarity of modes solely in the wall-normal direction (which was done in [77]), it is also preferable that all modes are 1D mutually orthogonal in each spatial direction too.

The next decomposition technique we discuss is spectral proper orthogonal decomposition (SPOD). There are two seminal papers introducing the SPOD, [218] and [206], but they are about two different techniques of the same name. To avoid confusion, we will refer to the work of Towne et al. in [218] as "SPOD" and the work of Sieber et al. in [206] as "Sieber's SPOD", since we will discuss the former here in detail and the latter in some detail next.

SPOD was applied to the pressure field of turbulent jet flow in [218], and it captured space-time coherence better than POD; it was also shown that SPOD modes are equivalent to an ensemble of DFT/DMD modes and equivalent to resolvent modes under uncorrelated forcing.

The SPOD decomposition of the pressure field p of the jet flow in [218] is

$$p(r,x,t) = \sum_{n} \sum_{f'} a^{(n,f')}\, \phi_n^{(f')}(r,x)\, \zeta_{f'}(t)\,, \tag{5.2}$$

where $\zeta_{f'}(t)$ is a Fourier mode with frequency f'. This equation shows that SPOD is similar to DFT-POD (DFT in time) in that the POD modes (spanning space) form a basis with Fourier modes; however, the key success of SPOD comes from a preprocessing step. The left panel in Figure 5.1(b) shows a depiction of this preprocessing step of SPOD, which is to segment the time direction t to create two directions: a shorter-spanning time direction t', and an ensemble direction t_b containing the blocks. If DFT was done in the original time direction t', there would be many Fourier modes spanning the frequency domain; by performing DFT in a shorter segment, the range of frequencies remains the same but the resolution in Fourier space is decreased—the frequency bins f' are much coarser than what the frequency bins f would be. In other words, SPOD is essentially a short-time Fourier transform in time with POD in space. Figure 5.1(b) shows the remaining steps, which is to perform a DFT in the t' direction and perform a POD decomposition for each coarse frequency bin f'. Similar with DFT-POD, SPOD modes $\phi_n^{(f')}(r,x)$ across different frequency bins f' are not orthogonal to each other under a 1D inner product. Nonetheless, in extracting space-time coherent structures, SPOD is useful in that it balances normal POD (which extracts spatial structures with a lot of spectral mixing in their temporal behavior) and DFT-POD (which extracts spatial structures with too much frequency localization). SPOD has been used successfully find nested coherent structures in elasto-inertial turbulence [117].

The last major decomposition technique we discuss is multiscale proper orthogonal de-

composition (mPOD) [156]. mPOD has origins in Sieber's SPOD [206], where the former filtered/windowed the covariance matrix with kernels of different widths—a small kernel (no filter) led to normal POD, and a large kernel led to DFT. mPOD combines Sieber's SPOD idea of filtering the covariance matrix with multiresolution analyis (MRA). While MRA is a term usually referring to wavelet decompositions, we refer it to as any decomposition in which a signal is split into components of different scale. In fact, a DFT can be viewed as a MRA. In [156], mPOD is applied to the wall-normal velocity v of an impinging jet flow in the $y-x$ plane (jet comes down the y direction and is deflected to the x direction) with decomposition

$$v(y,x,t) = \sum_{n} \sum_{f_{\text{bin}}} a^{(n,f_{\text{bin}})}\, \phi_n^{(f_{\text{bin}})}(y,x)\, \zeta_{f_{\text{bin}}}(t)\,, \tag{5.3}$$

where f_{bin} is the frequency bin. The left panel of Figure 5.1(c) shows the origin of f_{bin}, which is found by performing MRA (projecting the data onto frequency bins of varying size) in time t. These MRA projections are enforced to be orthogonal through Gram-Schmidth orthogonalization. The other panels in Figure 5.1(c) show that because the time t has been turned into both a POD direction and an inverse-scale direction through DFT/MRA, the resulting covariance matrices are not full rank and mPOD modes across different f_{bin} indices are 1D orthogonal—in other words, performing MRA in the POD direction before performing POD produces a multiscale basis that is energetically ordered at each scale. In [156], mPOD is applied in the time direction, which is puzzling since POD (even on projections) should yield Fourier modes yet the mPOD modes for the impinging jet flow were not so localized in the Fourier domain. In [26], mPOD was also applied in the time direction and was used to examine VLSM superposition and modulation of near-wall structures. In this chapter, we aim to decompose a 3D turbulent channel flow dataset with a basis composed of wavelet modes and mPOD or POD modes.

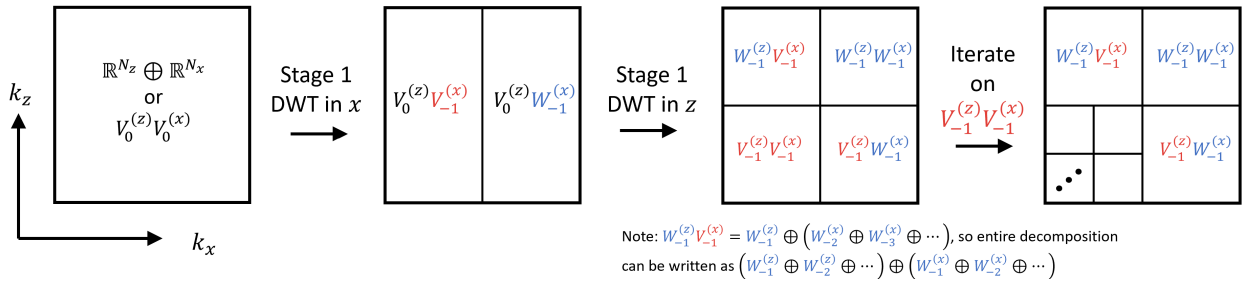


Figure 5.2: Partitioning of 2D wavelet subspaces, with similar color scheme and notation as Figure 2.4.

5.2 Methodology

We apply our methods to a DNS dataset of 3D turbulent channel flow from Johns Hopkins Turbulence Database [124, 174]. The friction Reynolds number is $\mathrm{Re}_\tau = 1000$. Other simulation details can be found on `https://turbulence.pha.jhu.edu/Channel_Flow.aspx`. Let h denote the half-channel height. We gather data on $2^{11} = 2048$ gridpoints in x, $2^9 = 512$ in y, and $2^{11} = 2048$ gridpoints in z (some interpolation is done in z). We generate 250 snapshots separated by about 0.1 eddy turnover times, defined as tU_b/h where U_b is the bulk velocity. Let u, v, w be the streamwise, wall-normal, and spanwise fluctuating velocity components (we will mostly work with u).

The streamwise x and spanwise z directions will be decomposed with a discrete wavelet transform (DWT); see details on DWT and DDWD in Chapter 2. We first discuss the 2D DWT for x-z data. Figure 5.2 shows the iterative partitioning of 2D wavelet subspaces, similar to the 1D wavelet subspaces shown in Figure 2.4. A subspace V_{-l} contains low-wavenumber information, and a subspace W_{-l} contains high-wavenumber information; a superscript with parentheses denotes which direction. At each stage of the 2D DWT, a 1D DWT in one direction is followed by a 1D DWT in the other direction. The decomposition is repeated recursively on the 2D subspace $V_{-l}^{(z)} V_{-l}^{(x)}$. A 2D DWT basis can be formed by following this recursive partitioning, or just performing a 1D tensor basis between the set of subspaces from each direction (see bottom comment in Figure 5.2).

There are three options to perform a 2D DDWD. The first option is the simplest, which is

to apply 1D DDWD unidirectionally and perform a tensor product of each 1D DDWD basis at the end. The second option is to recursively solve the optimization problem at each stage while sequentially handling each direction (i.e., a 1D DDWD problem is solved at each stage in the x direction and then in the z direction). The third option is to recursively solve the optimization problem at each stage and simultaneously handling each direction (i.e., directly maximizing the energy in $V_{-l}^{(z)}V_{-l}^{(x)}$). Through validation results with synthetic data (similar to what was done in Section 2.6.2 in Chapter 2 for details), we find that the first two options work but the third does not (possibly due to implementation errors). In this chapter, we use the first option for finding DDWD wavelets in x and z.

For the wall-normal direction, we have the option of POD or mPOD modes. For simplification, the following formulations and later results will be for y-x slices of data. When using POD modes in y and wavelets in x, the streamwise fluctuating velocity field is decomposed as

$$u(x,y,t) = \sum_{n}\sum_{l_x}\sum_{m_x} a^{(n,l_x,m_x)}(t)\,\phi_n^{(l_x)}(y)\,R^{2^{l_x}m_x}\left[\psi_{-l_x}(x)\right]\,, \tag{5.4}$$

where R is the circular shift operator (see Chapter 2), l_x is the stage, and m_x is the wavelet translate index. Note, $\phi_n^{(l_x)}(y)$ only depends on the modal index l_x and not m_x, since we absorb the m_x index into the ensemble direction (justified by homogeneity in x direction). Figure 5.3(a) shows the decomposition with DWT in x and POD in y, which we will refer to henceforth as "DWT-POD".

The other option for the wall-normal direction is to use mPOD modes. The MRA in mPOD can be done in Fourier space [156] or wavelet space [155]. We choose to do it in wavelet space. One complication is that wavelets are designed for periodic function, while our signals in y are not periodic. To remedy this issue, we zero-pad the signals in y (similar to the windowing procedure done in Chapter 2). Lastly, we have the option of performing a 1D DWT in y as the MRA followed by POD in the projections to obtain POD modes; however, to be consistent with the DWT-POD method, we choose to do a 2D MRA in x and y (we observe that doing a 1D or 2D MRA does not affect the resulting mPOD modes). We then

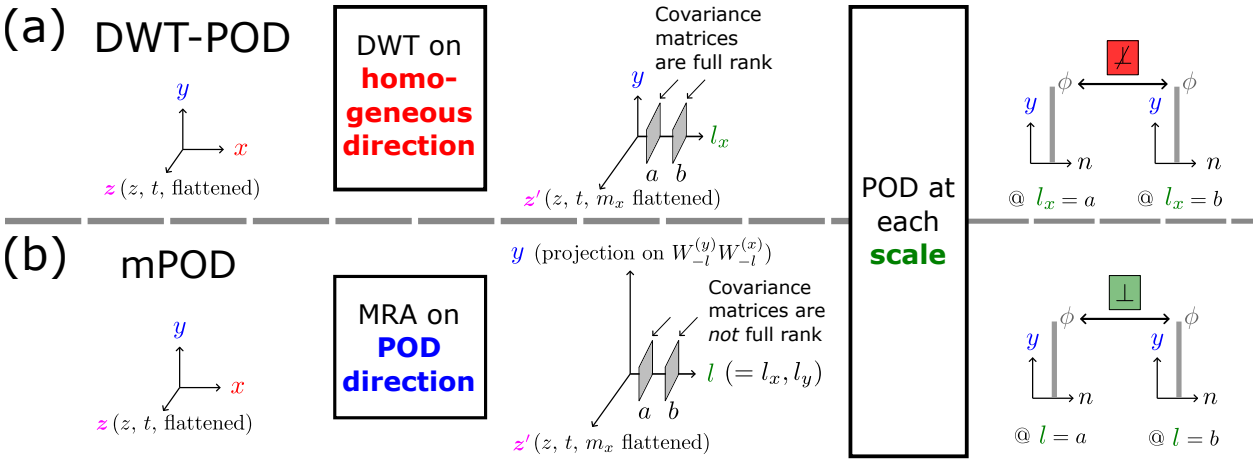


Figure 5.3: Sketch describing (a) DWT-POD and (b) mPOD (multiscale POD). The left panel shows the relevant directions, the middle panel shows the transformed directions after DWT/MRA is performed in the homogeneous/POD direction, and the right panel shows whether the POD modes across different scales are orthogonal under a 1D inner product.

perform 2D wavelet projections onto the diagonal subspaces $W_{-l}^{(y)}W_{-l}^{(x)}$; note, the wavelet projections need only span the y direction and not necessarily the x direction, so projections onto the left-leaning subspaces in Figure 5.2 is also possible (and does not affect the mPOD modes much either). The decomposition of u for mPOD is the same as for DWT-POD in Equation 5.4, except the number of modal indices n for mPOD will decrease with increasing $l = l_x$ since the projections onto larger scale subspaces will have decreasing rank.

5.3 Results

5.3.1 DDWD wavelets in x

First, we show results for applying 1D DDWD to u in the x direction at multiple wall-normal positions, as in Chapter 2. Figure 5.4 shows the wavelets. All notation, self-similarity metrics, and other conventions follow Chapter 2. Note, $\tilde{y}$ is the wall-normal direction when measured from the bottom wall. The DDWD wavelets in x do not change much with y, and they look like Meyer wavelets. Based on the first case where all wall-normal positions are used in the data matrix for DDWD, self-similarity lasts until stage 7 or a wavelet width/wavelength

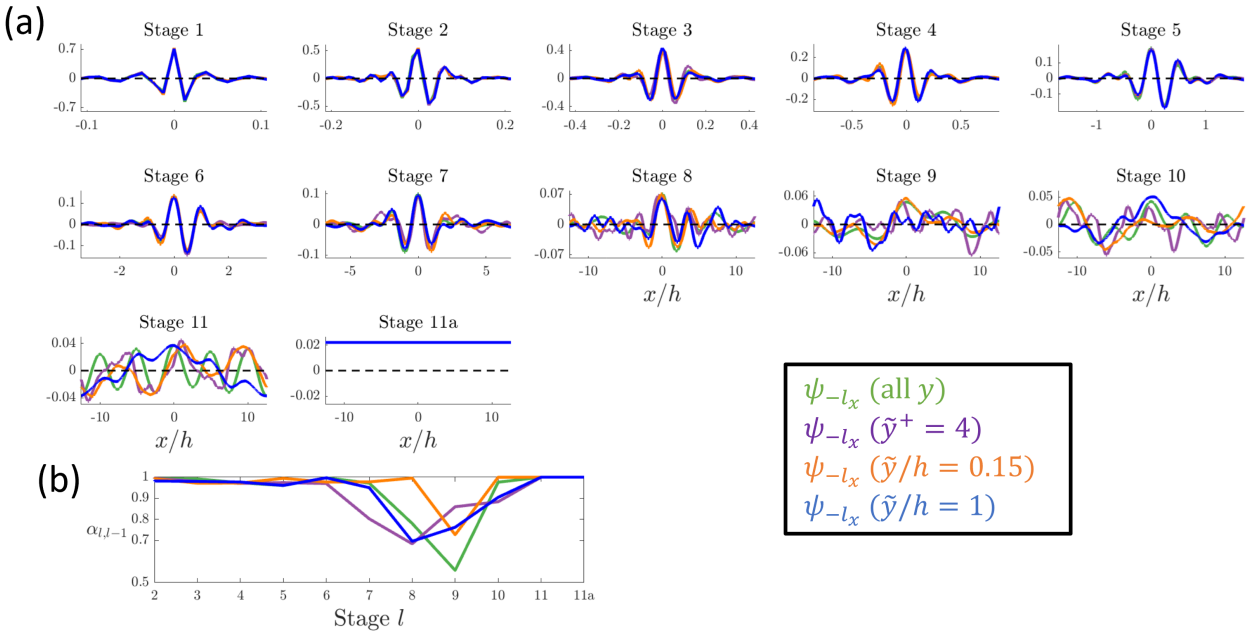


Figure 5.4: DDWD basis in x found at all wall-normal positions and three specified wall-normal positions. (a) Wavelets for all four cases. (b) Self-similarity across adjacent stages (i.e., inner product between the stage l_x wavelets and the dilated, previous-stage wavelets) for all four cases.

of about $6h$, which is consistent with the results of Chapter 2.

5.3.2 Streaks in viscous layer

We turn to applying 2D DDWD to u in the x-z plane at various wall-normal positions. Recall we apply 2D DDWD unidirectionally and using a tensor product to combine the basis from each direction. The 1D DDWD wavelets in x have been shown previously. Figure 5.5(a) shows the 1D DDWD wavelets in z at $y^+ = 7$, which is in the viscous layer where streaks are expected to be found. Figure 5.5(b) shows the adjacent self-similarity metric for these wavelets in green. Figure 5.5(b) also shows the adjacent self-similarity metric for DDWD wavelets calculated other wall-normal positions (these wavelets are not shown, since they look similar to those at $y^+ = 7$).

The drop in self-similarity at stage 4 for DDWD wavelets in z at $y^+ = 7$ may correspond to streaks. Figure 5.6(a) shows the fluctuating velocity field u. Figures 5.6(b) and 5.6(c) show the projection onto the stage-4 approximation and detail subspaces. Figure 5.6(d)

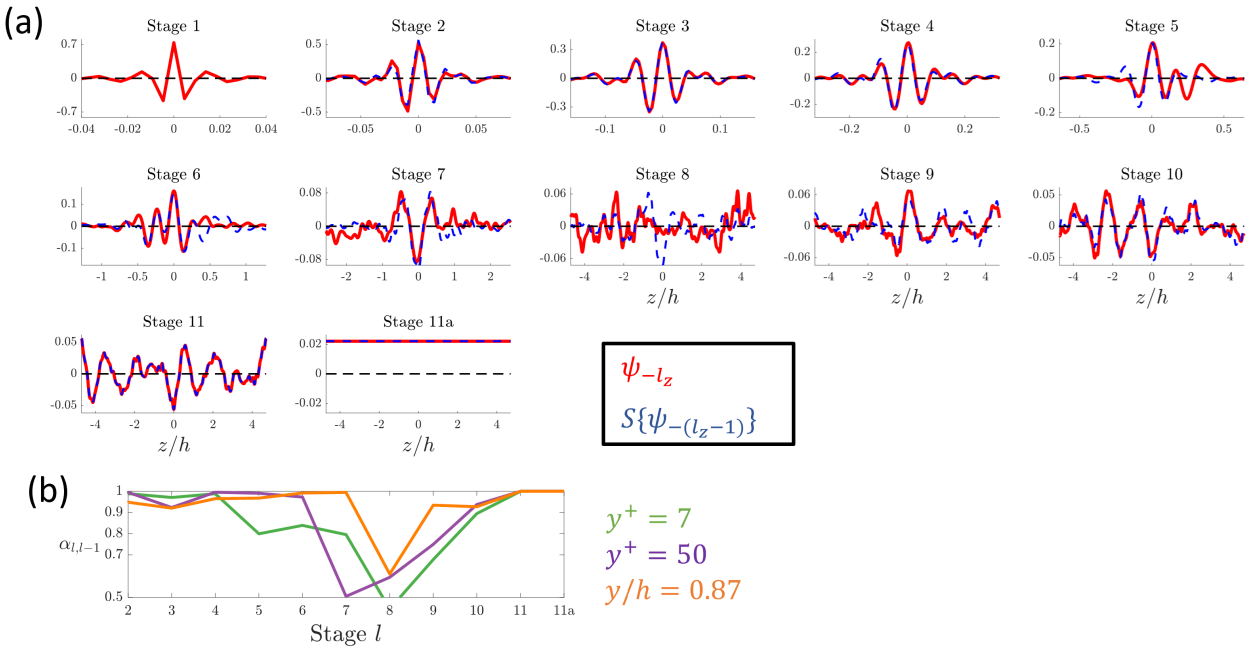


Figure 5.5: (a) DDWD basis in z found at $y^+ = 7$, with wavelets in red and the dilated, previous-stage wavelets in blue. (b) Adjacent self-similarity for $y^+ = 7$ wavelets and two other wall-normal positions.

shows a zoom-in of Figure 5.6(c), showing structures that resemble streaks. When using the friction Reynolds number of 1000 to convert from outer length scales to wall units (by simple division), the streak-like structures have spanwise width $z^+ \approx 100$ and length $x^+ \approx 1000$, which matches literature [112]. The drop in the self-similarity here at stage 4 for $y^+ = 7$ may seem to indicate the lengthscale in which the most energetic structures cease to exist.

5.3.3 POD modes from DWT-POD

To find the modes from DWT-POD, we use the DDWD wavelets in x to perform the DWT and perform POD for each wavelet scale l_x. Figure 5.7(a) shows several of the POD modes for POD index $n = 1$. As expected, since DDWD in x performs a MRA in x, an implicit MRA is also done in y and is reflected in the POD modes becoming larger with larger scale l_x. Figure 5.7(b) shows the magnitude of the inner product between all POD modes of index $n = 1$ across all scales. The diagonal is 1 as expected; however, the off diagonal values of the inner product demonstrate that the POD modes from DWT-POD are not orthogonal across

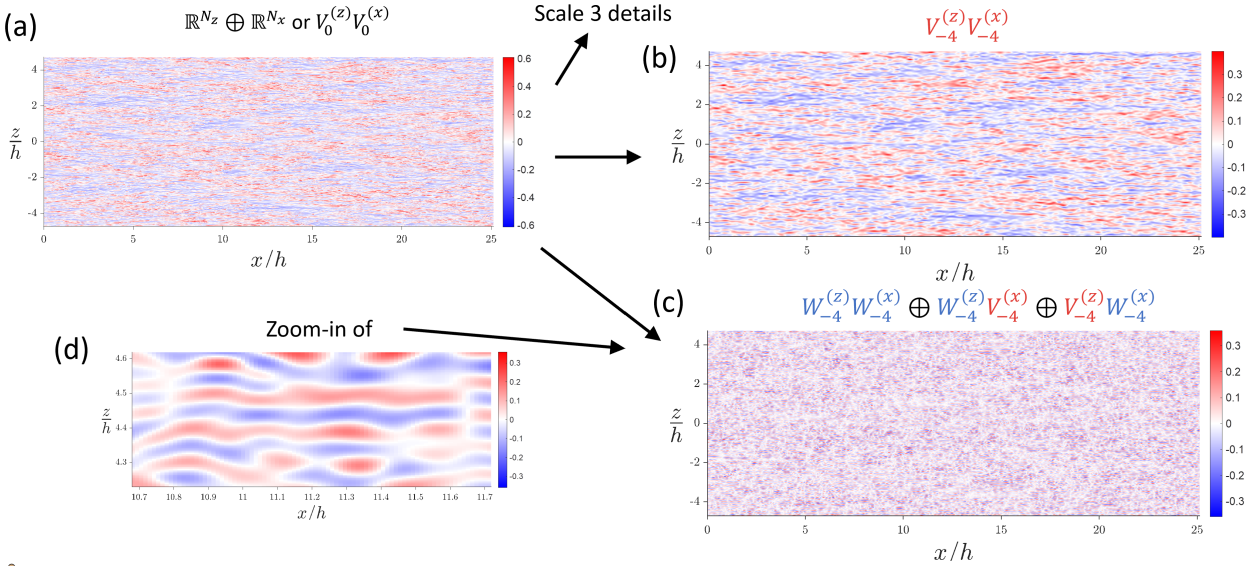


Figure 5.6: (a) Streamwise fluctuating velocity field $u(x, z)$ at $y^+ = 7$. Details scale 3 and below (not shown) are removed from the field to yield the projection onto the stage-4 approximation (b) and projection onto the stage-4 details (c). (d) Zoom-in of (c).

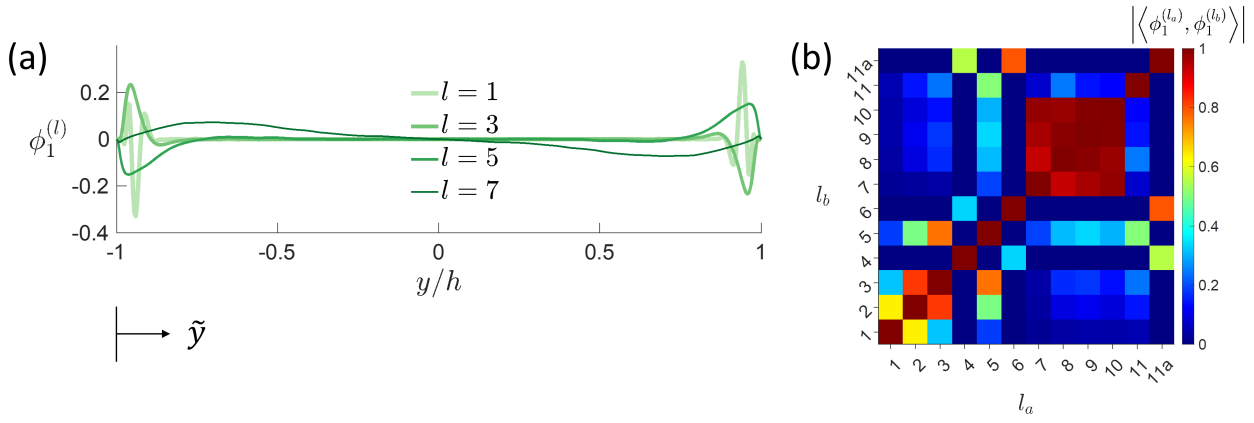


Figure 5.7: (a) POD modes found using DWT-POD (note, l refers to scale l_x in this figure). (b) Inner product of the $n = 1$ POD mode across scales.

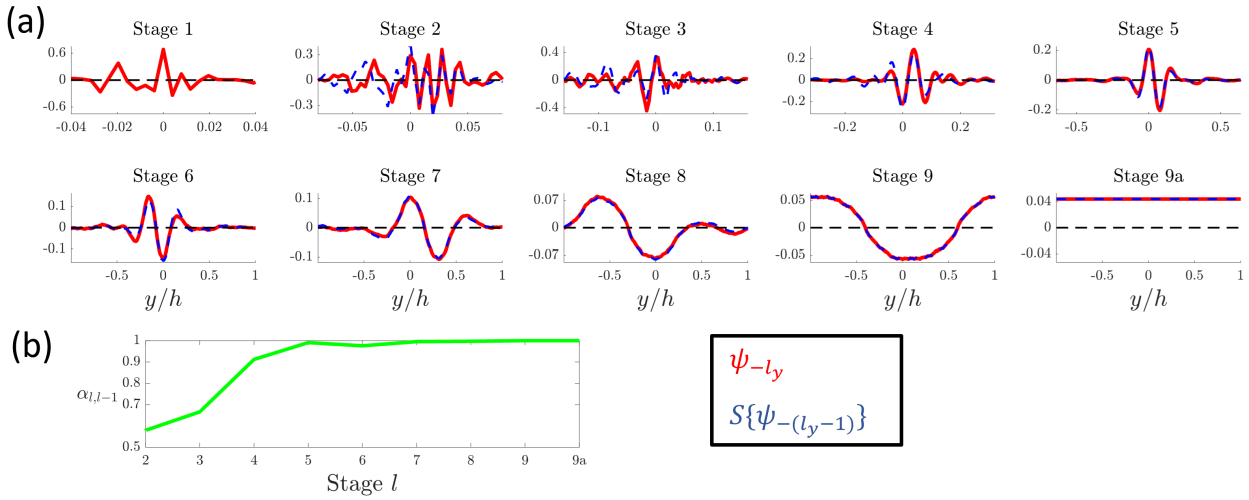


Figure 5.8: (a) DDWD modes in y in red and dilated, previous-stage wavelets in blue. (b) Adjacent self-similarity.

scales.

5.3.4 mPOD modes

We now show mPOD modes. Before performing the MRA in y (2D wavelet projections in x and y), we first find the DDWD wavelets in y. Figure 5.8(a) shows the DDWD modes in y, and Figure 5.8(b) quantifies the adjacent self-similarity. For these DDWD modes, self-similarity does not start until stage 5. However, the purpose of performing DDWD in y is not to obtain modes for representing structures—we just need the basis for performing a 2D wavelet projections for the MRA, and the resulting mPOD modes will be what will represent the structures in y. Moreover, DDWD is designed to extract energetic structures in statistically homogeneous directions (since wavelet translates span the whole domain), while y is a nonhomogeneous direction, so the DDWD results here should not be interpreted too readily.

Figure 5.9(a) shows the mPOD modes as a function of distance from wall $\tilde{y}$ scaled by the length scale 2^{l_x}. The rescaling allows us to visualize the self-similarity of the modes, which appears for stages 3 through 7; this range corresponds to the velocity structures being self-similar from $y^+ = 100$ to $1h$, which is consistent with the results and literature review

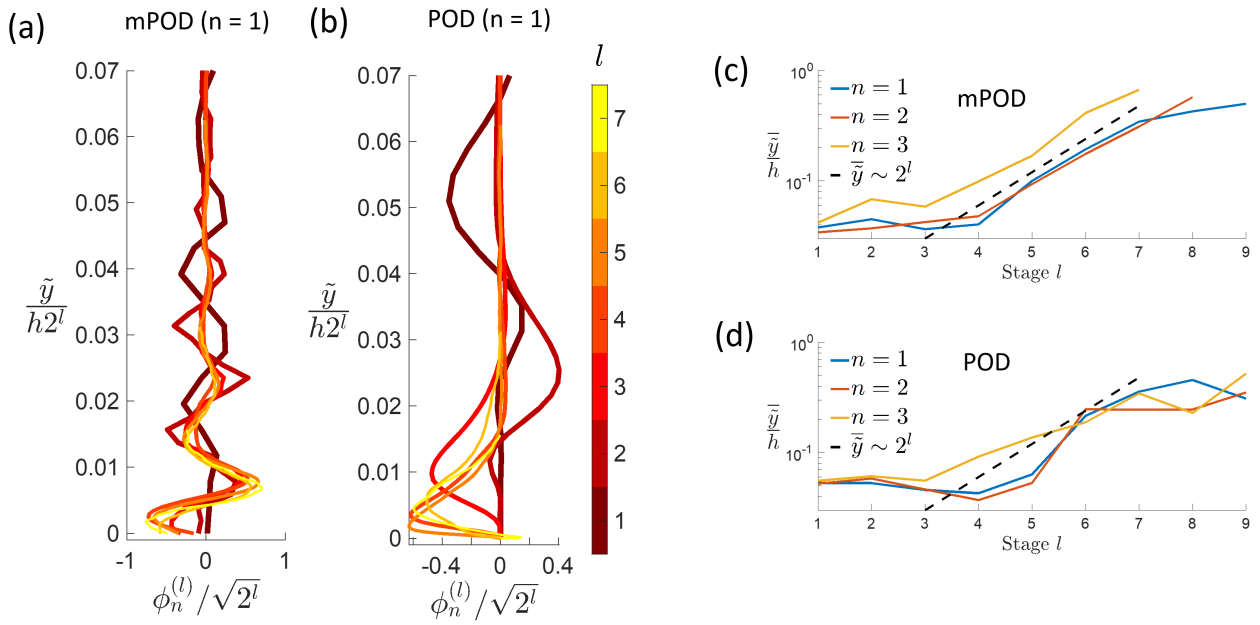


Figure 5.9: (a) mPOD modes normalized by the length scale (note stage/scale l should be l_x in this figure). (b) Same as (a) but for POD modes. Center of mass of modes as a function of scale for (c) mPOD and (d) POD modes.

in Chapter 2. Figure 5.9(b) shows that the POD modes from DWT-POD appear less self-similar. Figures 5.9(c) and 5.9(d) quantify self-similarity for mPOD and POD, respectively, by plotting the center of mass of the POD modes. The center of mass position $\overline{\tilde{y}}$ is calculated by using the POD mode as a probability density function and finding its expected value. Overall, as compared to POD modes, mPOD modes are orthogonal in 1D inner products and display more self-similarity.

5.3.5 Projections and energy

Figure 5.10(a) and 5.10(b) show the stage-6 projection of u onto the $n = 1$ mPOD and POD mode, respectively. For both POD and mPOD, the projections display significant blocking artifacts in both spatial directions. The reason for this is that wavelets and POD/mPOD modes themselves are wavy or blocky in nature so that modes from different modal translates indices can remain orthogonal to each other. One remedy for this blocking artifact could be do reapply 2D POD on the wavelet projections themselves, as done similarly in [74], which

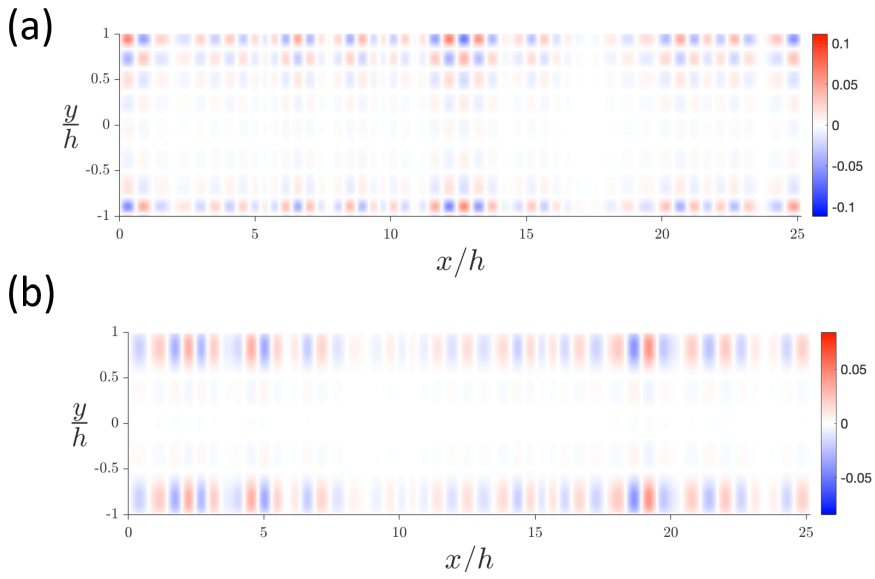


Figure 5.10: Stage-6 projection of fluctuating streamwise velocity field onto the $n = 1$ mPOD (a) and POD (b) mode.

has not been attempted yet here.

Lastly we discuss the energy contained in the modes. The most dominant ($n = 1$) POD modes across all scales contain 30% to 40% of the energy in the fluctuating streamwise velocity field u (the energy is calculated with 2D inner products with the wavelets, since POD modes are not orthogonal in 1D across different scales. The $n = 1$ mPOD modes across all scales contain 50.7% of the energy in u; when including $n = 2$ and $n = 3$, around 80% of the energy is captured. While not shown, the mPOD modes for $n = 2$ and $n = 3$ show a similar range of self-similarity as those shown for $n = 1$.

Figure 5.11 shows the energy λ_n contained in modes used to decompose the wall-normal direction of the u dataset. Note, in the figure, "POD" is not POD from DWT-POD but rather normal POD applied to 1D snapshots of u—as such, the energy in $n = 1$ is the largest it can be out of any orthogonal modal basis. In the figure, the "mPOD" curve is the energy contained in the mPOD modes, with stage/scale decreasing from left to right (within each stage, n increases). In the figure, the "mPOD (ordered)" curve simply orders the energy from highest to lowest in the "mPOD" curve. While mPOD does not give the highest energy optimality as POD, it does provide a multiscale ordering of the modes by wavelet stage.

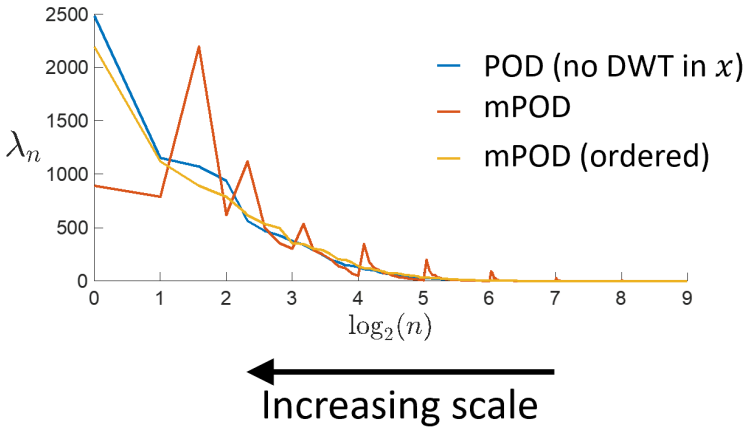


Figure 5.11: Energy of modes in fluctuating streamwise velocity field u as a function of modal index n. POD is applied to 1D snapshots of u instead of wavelet projections, mPOD is shown for decreasing scale and increasing n, and mPOD (ordered) is shown for only increasing n.

Therefore, for the general purpose of modeling energetic, multiscale structures, mPOD has benefits over POD.

We end with some last comments on the analysis performed here. We only treated data in the x-z and y-x planes separately, but doing so does not influence the results—preliminary results (not shown here) with using the entire 3D dataset (downsampled in all directions to avoid memory issues) yield similar results. Nonetheless, work remains to find the ideal optimization problem to simultaneously perform DDWD in x and z and DDWD-based MRA in y for the mPOD modes. Lastly, we focused only on the streamwise component of the fluctuating velocity. Preliminary results (not shown here) show that repeating the analysis for the other two velocity components yields similar levels of self-similarity and energy optimality of the mPOD modes.

5.4 Conclusion

In this chapter, we used wavelet-based modal decomposition methods to characterize structures in the streamwise velocity component of 3D turbulent channel flow. DDWD was applied to the homogeneous streamwise and spanwise directions, where self-similarity in the log- and wake-layer was observed up to about 5 half-channel heights in each direction. In the viscous

layer, wavelet projections were applied to reveal streaks. In the wall-normal direction, we applied two methods: POD at each wavelet stage, and multiscale POD (mPOD) through a wavelet-based projection step. mPOD performs a wavelet projection in the same direction as the POD modes, which offers some benefits. First, orthogonality under a 1D inner product for modes across different wavelet stages/scales was only seen with mPOD modes and not POD modes. Moreover, mPOD modes were more self-similar (100 wall units to 1 channel height) and more energetic (the dominant modes across all scales capture around 15% of the energy of the entire dataset). As for the shape of the modes, the DDWD wavelets in the homogeneous directions are similar in shape to Meyer wavelets, while the mPOD modes are similar in shape to Bessel functions. Overall, the combination of DDWD and mPOD yields a method for extracting multiscale, energetic structures from 3D wall-turbulence data. Future work involves polishing the optimization problem of DDWD/mPOD and repeating the analysis for the wall-normal and spanwise velocity components (the preliminary results for these components, not shown here, are similar to the results for the streamwise component).

6

Conclusions

6.1 General summary

In this thesis, we developed methods to: 1) characterize physical structures in wall-bounded turbulence, and 2) model spatiotemporally chaotic dynamics. The first goal was achieved by employing a modal-based decomposition approach in which modes directly represent the coherent structures in wall-bounded turbulence. The second goal was achieved by using machine learning techniques to construct a reduced-order model (ROM) for making high-fidelity, time series predictions. Specifically, our ROM uses an autoencoder to learn a reduced representation on the underlying inertial manifold, and the vector field on the manifold is then learned with a neural ordinary differential equation (NODE).

In Chapter 2, we applied a data-driven wavelet decomposition (DDWD) to extract structures in experimentally-measured, 1D velocity signals of turbulent pipe flow. DDWD combines the energetic optimality property of proper orthogonal decomposition (POD) with the spatially-localized and multiscale property of wavelets, and is thus well-suited for determining characteristics of attached eddies in wall-bounded turbulence. The DDWD wavelets, which represent the velocity signatures of attached eddies, were similar in shape to Meyer wavelets; moreover, they were self-similar for a large range of lengthscales within the log- and wake-layer, thus lending support to theory on attached eddies.

In Chapter 3, we presented a technique to supplement a data-driven ROM with information on the physics of the system. We applied this hybrid ROM to the Kuramoto–Sivashinsky and complex Ginzburg–Landau equations, which are 1D partial differential equations exhibiting spatiotemporal chaos (similar to the Navier-Stokes equations). We found the hybrid ROM to outperform the solely data-driven ROM in time-series predictions for scenarios of abundant data, scarce data, erroneous physics, and noise.

In Chapter 4, we continued the efforts to improve our ROMs by considering the real-world scenario in which only partial observations of the state is available. We combined the attention mechanism of transformers with time delay embeddings to reconstruct fields from temporally intermittent and spatially sparse sensors. We also presented a guideline on using attention mechanisms in a NODE.

In Chapter 5, we returned to wavelet-based characterization of structures in a simulation of 3D turbulent channel flow. We adapted a technique called multiscale POD (mPOD) to perform a multiresolution analysis with DDWD wavelet projections. As compared to DDWD, mPOD is more suitable for extracting energetic, multiscale structures in the nonhomogeneous wall-normal direction. Unlike POD modes, mPOD modes remain orthogonal under 1D inner products, which aids in the analysis of energy content and possibly in modeling efforts. The discovered 3D DDWD-mPOD basis was also self-similar, further corroborating the results of Chapter 2.

6.2 Future work

Here, we elaborate on avenues of future work. In Section 6.2.1, we start by providing preliminary results for using the modal basis in Chapter 5 to create a kinematic model of wall-turbulence. In Section 6.2.2, we present preliminary results on using IRMAE to estimate the manifold dimension of 2D homogeneous isotropic turbulence (HIT). We end this thesis in Section 6.2.3 by discussing extensions that can be made to our hybrid ROMs

and applications to more complex systems.

6.2.1 Attached eddy basis

Here, we aim to develop a kinematic model of wall-turbulence, similar to the attached eddy model (AEM) discussed in Section 1.1.3. The AEM constructs a model by populating a flow domain with multiscale vortex structures (which induce velocity signatures) that have the underpinning assumption of self-similarity. Since the 3D DDWD/mPOD basis in Chapter 5 was discovered to be self-similar across a large range of scales, it is effectively an attached eddy basis (AEB).

We begin by transforming the DDWD/mPOD decomposition from Chapter 5 into a form consistent with the AEH. We rewrite the AEH in Equation 1.6 into

$$u(x,y,z) = \sum_j U_j \, f\left(\frac{x-x_c}{\delta}, \frac{y}{\delta}, \frac{z-z_c}{\delta}\right), \tag{6.1}$$

where U_j is the velocity scale of eddy j (not universal, since we do not restrict our focus to the log-layer as was done in the AEH), $\boldsymbol{x}_c = (x_c, y_c, 0)$ is the center position of an eddy, and f is the universal function for the velocity field induced by the representative eddy. We focus on the fluctuating streamwise component of the velocity for now. The DDWD/mPOD decomposition can be written

$$\begin{aligned} u(x,y,z) = \sum_n \sum_l \phi_n^{(l)}(y) \sum_{m_x} \sum_{m_z} \Big[& a_{HH}^{(n,l,m_x,m_z)} \, \psi_{-l}\left(x - 2^l m_x\right) \, \psi_{-l}\left(z - 2^l m_z\right) \\ & + a_{LH}^{(n,l,m_x,m_z)} \, \xi_{-l}\left(x - 2^l m_x\right) \, \psi_{-l}\left(z - 2^l m_z\right) \\ & + a_{HL}^{(n,l,m_x,m_z)} \, \psi_{-l}\left(x - 2^l m_x\right) \, \xi_{-l}\left(z - 2^l m_z\right) \Big], \end{aligned} \tag{6.2}$$

where we avoid a double sum over l_x and l_z indices by representing the high-wavenumber-in-x—high wavenumber-in-z (HH) subspace along with the off-diagonal (LH and HL) subspaces at stage l with detail wavelets ψ_{-l} and approximation wavelets ξ_{-l}. To convert this equation

into the AEH form in Equation 6.1, we make two assumptions: 1) the modes are self-similar, and 2) the $n = 1$ mPOD mode is dominant; in Chapter 5, the modes were self-similar from stages 3 through 7 and the total energy in the $n = 1$ modes was around 50%, so these assumptions are not far off. Then, we can set the lengthscale of the eddy δ to 2^l and eliminate the sums over n and l to get

$$\begin{aligned} u(x,y,z) \approx \sum_j \Bigg[& U_{j,HH} \underbrace{\psi_{-l}\left(\frac{x-2^l m_x}{2^l}\right) \phi_1\left(\frac{y}{2^l}\right) \psi_{-l}\left(\frac{z-2^l m_z}{2^l}\right)}_{f_{HH}} \\ & U_{j,LH} \underbrace{\xi_{-l}\left(\frac{x-2^l m_x}{2^l}\right) \phi_1\left(\frac{y}{2^l}\right) \psi_{-l}\left(\frac{z-2^l m_z}{2^l}\right)}_{f_{LH}} \\ & U_{j,HL} \underbrace{\psi_{-l}\left(\frac{x-2^l m_x}{2^l}\right) \phi_1\left(\frac{y}{2^l}\right) \xi_{-l}\left(\frac{z-2^l m_z}{2^l}\right)}_{f_{HL}} \Bigg], \end{aligned} \tag{6.3}$$

which matches the AEH-based decomposition in Equation 6.1 when using three universal functions f (one for each of the three pairs of wavelets). Repeating this analysis for the other two velocity components means that a total of only nine universal functions are needed to represent the three-component, 3D velocity field. If the velocity scales (such as $U_{j,HH}$) follow a universal probability density function, then we have an AEB that can model the instantaneous velocity field with a representative velocity structure containing only nine degrees of freedom.

The first natural step is to obtain a view of the shape of the attached eddy (vortex) that gives way to the velocity structure given by the AEB. To do so, we use the turbulent channel flow dataset from Chapter 5, downsample it by a factor of four in all directions (to reduce memory costs), and apply a decomposition with Meyer wavelets in x and z (since the DDWD wavelets found in Chapter 5 were Meyer-like) and the MRA done in y. We take a stage-4 basis element and use regression to calculate the RMSE of the modal coefficients $\sqrt{\left(a_{--}^{(n,l,m_x,m_z)}\right)^2}$ for each subspace; the ratios were 2.1 for HL vs. HH and 0.8 for LH vs. HH. These ratios were used to determine the ratio of contributions from each universal function. Now that the

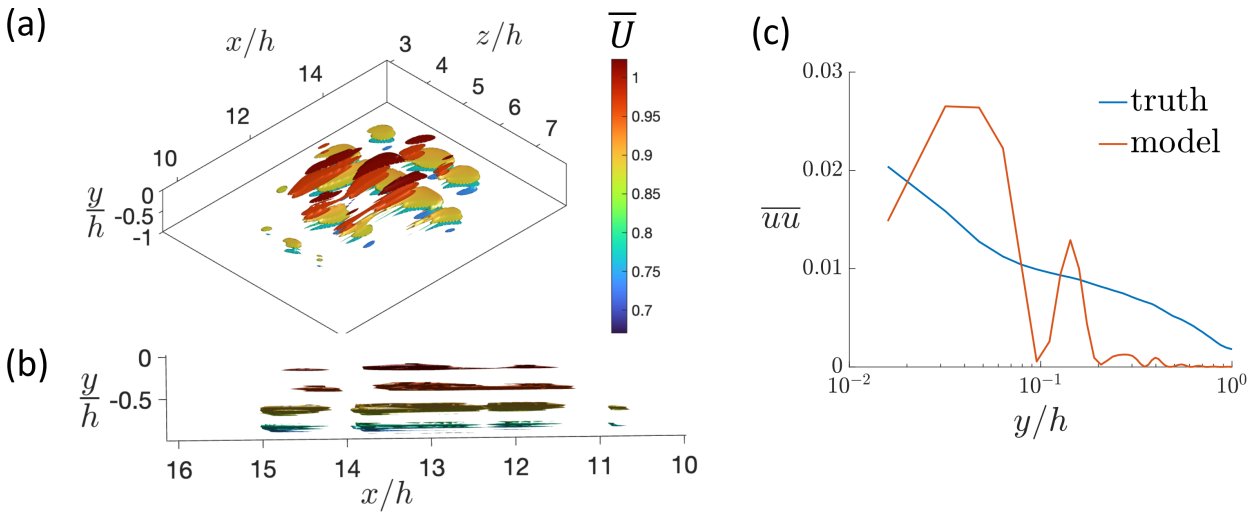


Figure 6.1: Attached eddy found with Q-criterion of representative velocity structure of an attached eddy basis (AEB), shown in isometric (a) and side (b) view. (c) Using the AEB as a model for the streamwise Reynolds stress.

representative velocity structure is distilled into a single function, we apply Q-criterion to find the corresponding attached eddy. Figure 6.1(a) shows the representative attached eddy found with the AEB, with a side view depicted in Figure 6.1(b). The attached eddy displays many oscillations in all three spatial directions, which results from the representative velocity structure containing many oscillations itself.

Next, we attempt to use the AEB as a AEM for obtaining Reynolds stresses and/or instantaneous velocity fields. Like with the AEM, we will randomly populate the field with the representative velocity structure at different sizes and locations. First, we assume the number density of attached eddies at scale $\delta = 2^l$ scales as $p_H(\delta) \sim 1/\delta$. Next, we assume the same ratios of the RMSE of modal coefficients as calculated before. Finally, we assume a fixed wavelet energy distribution (WED) for the HH subspace that increases exponentially with stage l. Figure 6.1(c) shows that the model does not capture even the log-law in the streamwise Reynolds stress; therefore, the instantaneous velocity fields (not shown) from the model are even worse off.

To improve on the AEB, a detailed account of the energy distribution for the modal coefficients (absorbed into the velocity scales $U_{j,\_}$) across all subspaces needs to be deter-

mined. Similar to the WED found in Chapter 2, these distributions may follow power laws. Moreover, the number density of attached eddies may not be as simple as $p_H \sim 1/\delta$, since many AEM models modify this function for the wake layer (see Section 1.1.3). Lastly, Figure 6.1 shows that the attached eddy and velocity structure derived from the AEB suffer from blocking artifacts. One idea is to use POD modes instead of mPOD modes in the y direction; however, as seen in Chapter 5, the oscillations still exist in POD modes and may still introduce blocking artifacts in y and will do nothing to remedy the blocking artifacts in the other spatial directions.

We end this section by discussing an alternative to a traditional discrete wavelet transform (DWT) for 2D/3D transforms. The DWT, as seen throughout this thesis, is designed to represent spatially-localized or point singularities. The oscillations in the tail of the wavelets lead to blocking artifacts too, which are more apparent in 2D/3D than 1D. One alternative is to use curvelets [146], which are a variation of wavelets that are designed to represent curves in 2D or surfaces in 3D. So, as opposed to using many point-wise wavelets to represent a curved turbulent structure, curvelets can do so with fewer modes, as was demonstrated in [146]. One downside of curvelets is that they do not form a basis—the redundancy factor (which measures the overcompleteness of the basis) in 2D is 7.2, which is not as serious as say the continuous wavelet transform—and adapting curvelets to be orthogonal is an open problem that remains challenging. So, in the context of representing structures in turbulence, there is the tradeoff that curvelets give more efficient representations but also lack orthogonality.

6.2.2 Estimating manifold dimension of homogeneous turbulence

Here, we apply the IRMAE method [253] to estimate the manifold dimension of 2D homogeneous isotropic turbulence (HIT). Much work has been done to apply IRMAE and DManD to complex systems, such as 3D turbulent Couette flow in a minimal domain [134], 3D turbulent pipe flow [32] in a minimal domain, and 2D Kolmogorov flow [92, 173, 186].

In these systems, the number of degrees of freedom on the inertial manifold is $\mathcal{O}(10)$. In 3D wall-turbulent flows of practical interest, the flow domain is larger and the Re is high—the number of degrees of freedom on the inertial manifold is expected to be at least $\mathcal{O}(100)$ or $\mathcal{O}(1000)$. In this section, we will show preliminary results for using a convolutional autoencoder (CAE) variant on IRMAE to estimate the inertial manifold dimension of 2D HIT at high Re.

The 2D incompressible vorticity equation is given by

$$\frac{\partial \omega}{\partial t} = -\mathbf{u} \cdot \nabla \omega + \frac{1}{\mathrm{Re}} \nabla^2 \omega \,, \tag{6.4}$$

where the vorticity is related to the velocity by $\omega = \nabla \times \mathbf{u}$. We consider a domain $(x, y) \in [0, 2\pi] \times [0, 2\pi]$ with periodic boundary conditions. For the simulation, we modify the equation according to [229] by adding two damping terms and a forcing term:

$$\frac{\partial \omega}{\partial t} = -\mathbf{u} \cdot \nabla \omega + \frac{1}{\mathrm{Re}} \nabla^2 \omega + (-1)^{m_1+1} \nu \Delta^{m^1} \omega + (-1)^{m_2+1} \alpha \Delta^{-m_2} \omega + f \,, \tag{6.5}$$

where the $(-1)^{m_1+1} \nu \Delta^{m^1} \omega$ is the hyper-viscosity term, $(-1)^{m_2+1} \alpha \Delta^{-m_2} \omega$ is the hyper-drag term, and f is the forcing term. The forcing term is necessary to maintain a statistically stationary flow, otherwise the flow would decay. Following [229], we apply a random forcing at radial wavenumber k_f; at every timestep, we reset the magnitude of the Fourier component at k_f to be a fixed value while leaving the phase alone, which effectively provides a constant forcing at a specific scale but dispersed in space. 2D HIT exhibits an inverse energy cascade, where the injected energy at k_f moves to larger scales following a $-5/3$ power law—therefore, the hyper-drag term is needed to remove kinetic energy at small wavenumbers to prevent large-scale structures from dominating the flow. The inertial range of 2D HIT, where energy flows from wavenumber k_f to higher wavenumbers associated with dissipation, follows a -3 power law. The hyper-viscosity term, along with the hyper-drag term, are used to establish a wide inertial range. We use $m_1 = 8$ and $m_2 = 1$. The simulation is performed with

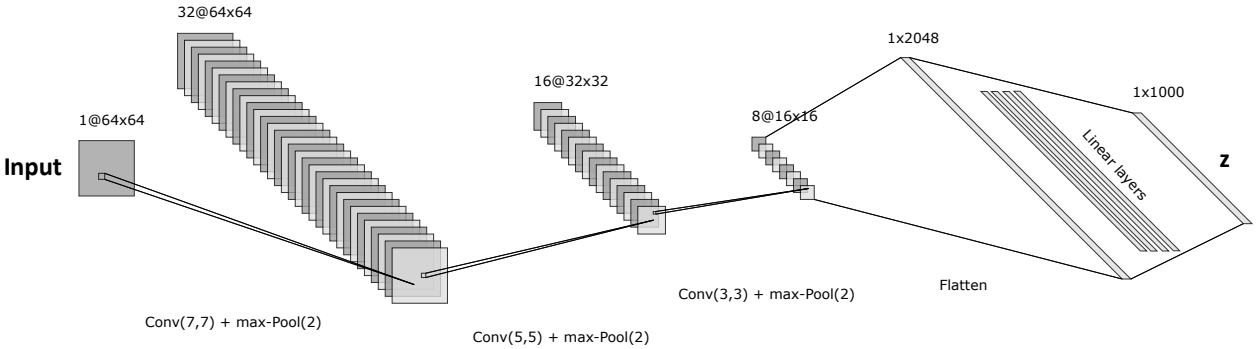


Figure 6.2: Architecture of a convolutional autoencoder. "Conv" denotes convolution layer with kernel size in parentheses and downsample factor for max-pool is in parentheses.

code written by Daniel Floryan. A pseudo-spectral method is used to discretize in space, and the time stepping is done with the Crank-Nicolson method (for the hyper-viscosity and hyper-drag terms) and Heun's method (for the convection and forcing terms).

We now discuss the neural network architecture we use for IRMAE. As compared to feedforward neural networks, convolutional neural networks (CNNs) are better suited to extract features from large images due to convolutional filters enabling the sharing of weights. We employ a convolutional autoencoder (CAE) to discover the inertial manifold coordinates. Figure 6.2 shows the CAE architecture that we will use (see caption for details). We use circular padding after each convolution layer, and all layers use GeLU activation. We use a learning rate of 10^{-3}, batch size of 64, 4 linear layers, and a weight decay strength of 10^{-4}. We train with an AdamW optimizer, with no learning rate scheduler, for 100 epochs. For training, we generate a single timeseries of ω consisting of 50,000 snapshots separated by 10^{-3} time units (simulation step size is 10^{-4} time units).

We now show results for a simulation generated with $N = 64$ grid points in each direction, a forcing wavenumber of $k_f = 4$, and $\mathrm{Re} = 1000$. Figure 6.3(a) shows a snapshot of ω, while Figure 6.3(b) shows a low-pass filtered version of it. Figure 6.3(c) shows the radial energy spectrum $E(k)$, where k is the radial wavenumber, depicting that the simulation produces the expected power law in the inertial range. The inverse energy cascade and $-5/3$ power law are shrouded due to the small forcing wavenumber. We first show IRMAE results for

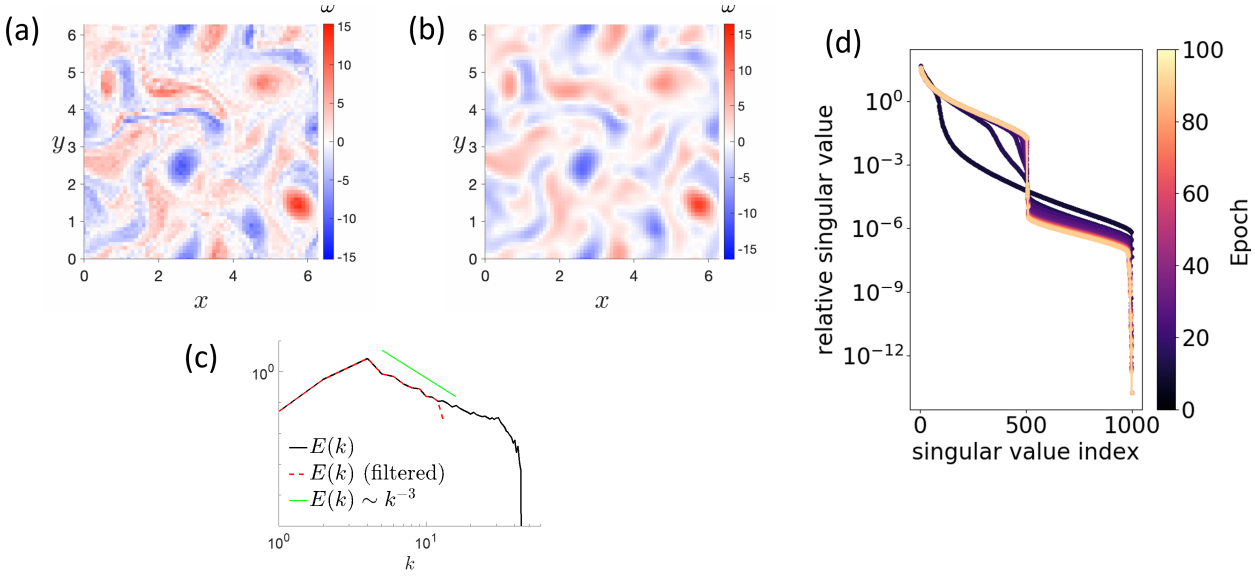


Figure 6.3: 2D HIT without filter (a) and with filter (b). (c) Energy spectrum. (d) Training dynamics of singular value spectra for the filtered vorticity field (IRMAE is learning the Fourier transform here).

the filtered vorticity field. Figure 6.3(d) shows the training dynamics of the singular value spectra. At the end of training, there is a relatively sharp drop in the spectra at around 500; however, this result is misleading. Since the filtered data contains 16 wavenumber modes in each direction, each with a real and imaginary component, there are 512 degrees of freedom in Fourier space. Consequently, the CAE is essentially finding the Fourier components of the data and not the inertial manifold coordinates.

We now show results for applying IRMAE to the unfiltered results in Figure 6.3(a). Figure 6.4(a) shows a plot of relative MSE vs. estimated dimension from 5 models (same training, just with different random weight initializations). IRMAE is not robust, since the dimension estimate is varying a lot and there is no minimum in the MSE vs. dimension curve. For the best performing model, Figure 6.4(b) shows a comparison of a representative snapshot to the ground truth and Figure 6.4(c) shows the training dynamics of IRMAE. The MSE is relatively high, and there is no clear drop in the singular value spectra (as with other models too).

IRMAE does not give a robust dimension estimate of 2D HIT, as evidenced by Figure

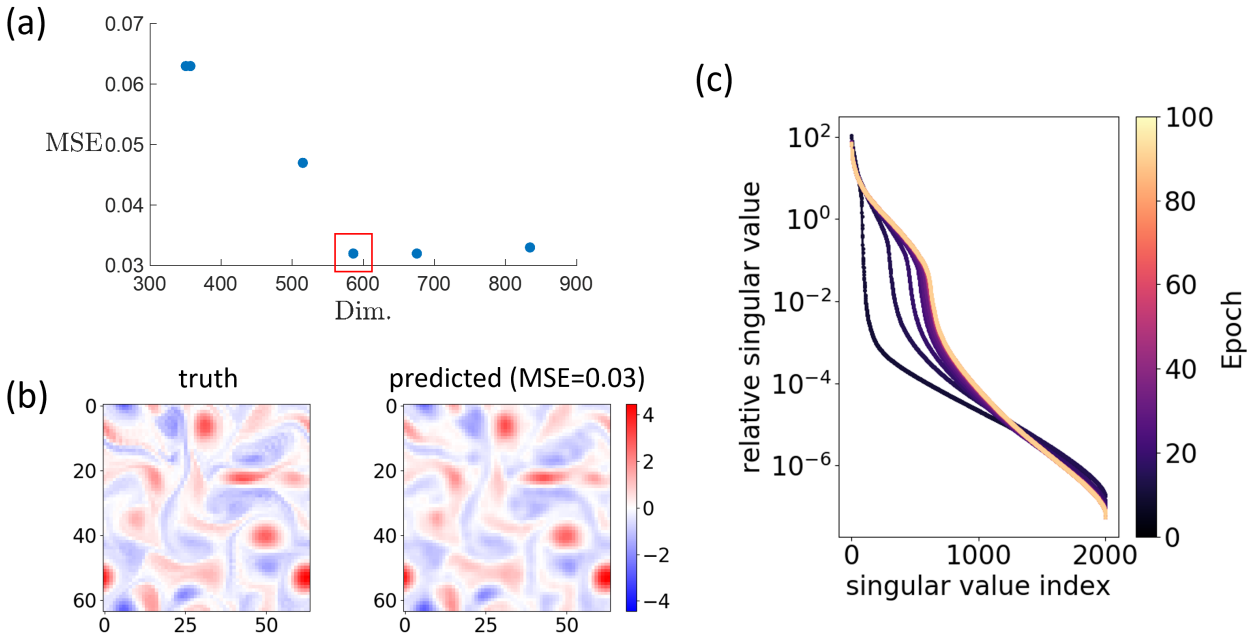


Figure 6.4: Results for unfiltered vorticity field. (a) Relative MSE vs. manifold dimension estimated from IRMAE for 5 models with different weight initializations. (b) Ground truth and representative reconstruction for best model (red square in subplot (a)). (c) Singular value spectra training dynamics for best model.

6.4. We tried various tweaks, but nothing helped this issue. We tried increasing the number of grid points N, making the network more expressive, varying the number of linear layers and/or weight decay strength, increasing the time series length, using an ensemble of time series, and different weight initializations for the linear layers. Overall, our network seems expressive enough, since the reconstruction error for 2D HIT is typically on the order or 0.05 [238]. In recent work applying IRMAE to 2D Kolmogorov flow [29], robust dimension estimates of around 50 to 480 were calculated for flows with Re ranging from 25 to 400. Since our data is at Re of 1000 and uses a forcing function that is more complex, the inertial manifold dimension is likely close to 1000.

We showed preliminary results on using IRMAE with a CAE architecture to estimate the inertial manifold dimension of 2D HIT. The dimension estimates were not robust, and work that remains to improve the robustness include trying dense blocks in the CAE as was done in [29], trying lower Re simulations, and trying a simpler forcing function that is deterministic.

6.2.3 Extending our reduced-order models

For this last section of the thesis, we discuss approaches for extending our hybrid ROM developed in Chapter 3 to more complex systems and challenging situations. The primary system of interest on which to apply PI-DManD is 3D channel flow, where the supplied information on the Navier-Stokes equations is suspected to aid in time evolution. First, however, DManD needs to be refined. The critical assumption for our dynamical systems-based approach for ROMs is that we can find the coordinates on the underlying manifold. However, with complex systems, the curse of dimensionality prohibits the estimation of the manifold dimension and the ability to obtain an accurate autoencoder model, as demonstrated in Section 6.2.2.

Linear dimension reduction as a preprocessing step to neural network training is one way to combat the curse of dimensionality. For 3D turbulent flows at low Re [32, 134], linear dimension reduction with POD proved useful for training DManD models. For higher Re flows, however, the number of POD modes that need to be retained to capture most of the energy of the original dataset increases substantially and the curse of dimensionality is not mediated. To achieve a more efficient linear dimension reduction, one can use the mPOD/DDWD modes of the AEB (as discussed in Chapter 5 and Section 6.2.1), since they directly represent the physical structures in the turbulent flow. The merging of the AEB as a linear dimension reduction step with the hybrid ROM can lead to a ROM that is physics-informed, data-driven, and structure-informed.

Another way to improve the training of DManD/PI-DManD models in complex systems is to utilize wavelets as convolutional filters in a CAE or CNN architecture. Just as POD is a linear autoencoder, a wavelet transform is a linear CAE [51]. Wavelet modes act as a convolution filter over a signal to produce wavelet coefficients. As such, wavelets are filters that are interpretable with fixed weights, while typical convolutional filters have high predictive power with weights that need to be learned. A wavelet scattering network (WSN) strikes the balance between the two by employing wavelets with fixed weights for the convolution and then applying nonlinear activation functions on the convolution. Combining the ideas

of the WSN with the AEB offers another option to develop a hybrid ROM that is informed by the turbulent structures in the data.

Lastly, we examine methods to extend our PI-DManD model to be more applicable to real-world scenarios. First, we desire our model to be robust to noise; as discussed in Chapter 3, PI-DManD is already robust to noise but can be improved upon with a robust autoencoder and/or a weak-formulation of the NODE [123]. Second, we desire our PI-DManD model to be applicable to the scenario where full coverage of the ambient space is not available (e.g., a wall-turbulent flow with only shear stress or pressure measurements at the wall). In Chapter 4, we demonstrated the use of time delays to recover a diffeomorphism to the inertial manifold. We can train a neural network to map from time delay embeddings to the manifold coordinates, and another neural network for the reverse mapping. Then, a physics-based MG model on the manifold coordinates and a time evolution model (possibly containing attention mechanisms) on the time delayed partial observables can be combined into a hybrid ROM. Lastly, we aim to further test PI-DManD on handling the scenario where the known physics are only approximate. One such scenario is with complex/non-rheometric flows of complex fluids, where accurate first-principles constitutive models do not exist. We aim to apply PI-DManD first to the case of rigid rod suspensions (governed by orientation probability distribution function) and then systems governed by stochastic differential equations. Lastly, we plan to integrate all of these extensions to PI-DManD discussed here with learning control policies to reduce turbulent drag, as was done with reinforcement learning in minimal turbulent Couette flow in [138].

This thesis explored the ideas of: 1) using wavelets to represent coherent structures in wall-bounded turbulence, and 2) informing data-driven low-order models with knowledge of the physics governing the system. We believe that merging these two ideas and refining them will pave the way for ultimately being able to mechanistically understand, model, and control Newtonian/non-Newtonian turbulent flows.

References

[1] R. J. Adrian. Hairpin vortex organization in wall turbulence. *Physics of fluids*, 19(4): 041301, 2007.

[2] I. S. Aranson and L. Kramer. The world of the complex Ginzburg-Landau equation. *Reviews of Modern Physics*, 74(1):99–143, 2002.

[3] 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(6210):51–53, 1989.

[4] W. J. Baars and I. Marusic. Data-driven decomposition of the streamwise turbulence kinetic energy in boundary layers. Part 1. Energy spectra. *Journal of Fluid Mechanics*, 882:A251–A2540, jan 2020.

[5] W. J. Baars, K. M. Talluru, N. Hutchins, and I. Marusic. Wavelet analysis of wall turbulence to study large-scale modulation of small scales. *Experiments in Fluids*, 56 (10):188–, oct 2015.

[6] W. J. Baars, N. Hutchins, and I. Marusic. Self-similarity of wall-attached turbulence in boundary layers. *Journal of Fluid Mechanics*, 823:R2, 7 2017.

[7] S. C. Bailey, M. Vallikivi, M. Hultmark, and A. J. Smits. Estimating the value of von Kármán's constant in turbulent pipe flow. *Journal of Fluid Mechanics*, 749:79–98, jun 2014.

[8] J. Bakarji, K. Champion, J. Nathan Kutz, and S. L. Brunton. Discovering governing equations from partial measurements with deep delay autoencoders. *Proceedings of the Royal Society A*, 479(2276):20230422, 2023.

[9] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, jun 2003.

[10] G. Benettin, L. Galgani, A. Giorgilli, and J. M. Strelcyn. Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; A method for computing all of them. Part 2: Numerical application. *Meccanica*, 15(1):21–30, mar 1980.

[11] T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine. Enforcing Analytic Constraints in Neural Networks Emulating Physical Systems. *Physical Review Letters*, 126(9):098302, 03 2021.

[12] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 113(15):3932–3937, 2016.

[13] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine Learning for Fluid Mechanics. *Annual Review of Fluid Mechanics*, 52(1):1–32, 2019.

[14] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine Learning for Fluid Mechanics, 2020.

[15] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz. Modern Koopman Theory for Dynamical Systems. *SIAM Review*, 64(2):229–340, may 2022.

[16] J. Buzhardt, C. R. Constante-Amores, and M. D. Graham. On the relationship between Koopman operator approximations and neural ordinary differential equations for data-driven time-evolution predictions. *Chaos*, 35(4), apr 2025.

[17] L. Cao. Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D: Nonlinear Phenomena*, 110(1-2):43–50, dec 1997.

[18] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91: 045002, Dec 2019.

[19] M. Cenedese, J. Axås, B. Bäuerlein, K. Avila, and G. Haller. Data-driven modeling and prediction of non-linearizable dynamics via spectral submanifolds. *Nature Communications*, 13(1):872, feb 2022.

[20] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *ArXiv*, stat.OT, 03 2019.

[21] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

[22] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. *arXiv preprint*, 2019.

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

[24] T. Chen and H. Chen. Universal Approximation to Nonlinear Operators by Neural Networks with Arbitrary Activation Functions and Its Application to Dynamical Systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, jul 1995.

[25] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, 28(8): 11618, jan 2020.

[26] C. Chi, D. Thévenin, A. J. Smits, S. Wolligandt, and H. Theisel. Identification and analysis of very-large-scale turbulent motions using multiscale proper orthogonal decomposition. *Physical Review Fluids*, 7(8):084603, aug 2022.

[27] J. T. Chien and Y. H. Chen. Learning Continuous-Time Dynamics With Attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1906–1918, feb 2023.

[28] D. Chung, I. Marusic, J. P. Monty, M. Vallikivi, and A. J. Smits. On the universality of inertial energy in the log layer of turbulent boundary layer and pipe flows. *Experiments in Fluids*, 56(7):141, jul 2015.

[29] A. Cleary and J. Page. Characterizing the Reynolds number dependence of the chaotic attractor in two-dimensional turbulence with dimension-minimizing autoencoders. *Physical Review E*, 112(5):055105, nov 2025.

[30] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38(2):713–718, 1992.

[31] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102 21:7426–31, 2005.

[32] R. Constante-Amores, A. J. Linot, M. D. Graham, C. R. Constante-Amores, A. J. Linot, and M. D. Graham. Dynamics of a data-driven low-dimensional model of turbulent minimal pipe flow. *Journal of Fluid Mechanics*, 1020:A58, oct 2025.

[33] C. Cossu and Y. Hwang. Self-sustaining processes at all scales in wall-bounded turbulent shear flows. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2089), mar 2017.

[34] P. Cvitanovic, R. Artuso, R. Mainieri, G. Tanner, G. Vattay, N. Whelan, and A. Wirzba. *Chaos: classical and quantum*. Niels Bohr Institute, Copenhagen, 2016.

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

[36] J. C. del Alamo, J. Jimenez, P. Zandonade, and R. D. Moser. Self-similar vortex clusters in the turbulent logarithmic region. *Journal of Fluid Mechanics*, 561:329–358, 2006.

[37] D. J. C. Dennis and T. B. Nickels. Experimental measurement of large-scale three-dimensional structures in a turbulent boundary layer. part 1. vortex packets. *Journal of Fluid Mechanics*, 673:180–217, 2011.

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

[39] 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(2):024101, jul 2016.

[40] B. Eckhardt and S. Zammert. Small scale exact coherent structures at large Reynolds numbers in plane Couette flow. *Nonlinearity*, 31(2):R66–R77, feb 2018.

[41] R. A. Edson, J. E. Bunder, T. W. Mattner, and A. J. Roberts. LYAPUNOV EXPONENTS of the KURAMOTO-SIVASHINSKY PDE. *ANZIAM Journal*, 61(3):270–285, jul 2019.

[42] G. Eitel-Amor, R. órlú, P. Schlatter, and O. Flores. Hairpin vortices in turbulent boundary layers. *Physics of Fluids*, 27(2):025108, feb 2015.

[43] R. Everson, L. Sirovich, and K. R. Sreenivasan. Wavelet analysis of the turbulent jet. *Physics Letters A*, 145(6-7):314–322, 1990.

[44] M. Farge. Wavelet transforms and their applications to turbulence. *Annual Review of Fluid Mechanics*, 24(1):395–458, 1992.

[45] M. Farge, G. Pellegrino, and K. Schneider. Coherent vortex extraction in 3D turbulent flows using orthogonal wavelets. *Physical Review Letters*, 87(5):054501, 2001.

[46] 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(10):2886–2896, 2003.

[47] J. D. Farmer and J. J. Sidorowich. Predicting chaotic time series. *Physical review letters*, 59(8):845, 1987.

[48] 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(1-3):1–11, 2011.

[49] O. Flores and J. Jiménez. Effect of wall-boundary disturbances on turbulent channel flows. *Journal of Fluid Mechanics*, 566:357–376, nov 2006.

[50] D. Floryan and M. D. Graham. Discovering multiscale and self-similar structure with data-driven wavelets. *Proceedings of the National Academy of Sciences*, 118: e2021299118, 1 2021.

[51] D. Floryan and M. D. Graham. Discovering multiscale and self-similar structure with data-driven wavelets. *Proceedings of the National Academy of Sciences of the United States of America*, 118(1):e2021299118, jan 2021.

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

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

[54] A. M. Fraser, H. L. Swinney, A. M. Fraser, and H. L. Swinney. Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33(2):1134–1140, 1986.

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

[56] M. K. Fu, Y. Fan, and M. Hultmark. Design and validation of a nanoscale cross-wire probe (X-NSTAP). *Experiments in Fluids*, 60(6):1–14, 2019.

[57] K. Fukami, R. Maulik, N. Ramachandra, K. Fukagata, and K. Taira. Global field reconstruction from sparse sensors with Voronoi tessellation-assisted deep learning. *Nature Machine Intelligence 2021 3:11*, 3(11):945–951, 10 2021.

[58] J. F. Gibson. *Dynamical systems models of wall-bounded, shear-flow turbulence*. PhD thesis, Cornell University, Jan. 2002.

[59] W. Gilpin. Model scale versus domain knowledge in statistical forecasting of chaotic systems. *Physical Review Research*, 5(4):043252, oct 2023.

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

[61] M. D. Graham and D. Floryan. Exact coherent states and the nonlinear dynamics of wall-bounded turbulent flows. *Annual Review of Fluid Mechanics*, 53:227–253, 2021.

[62] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[63] T. G. Grossmann, U. J. Komorowska, J. Latz, and C. B. Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics (Institute of Mathematics and Its Applications)*, 89(1):143–174, jun 2024.

[64] A. Güemes, C. Sanmiguel Vila, and S. Discetti. Super-resolution generative adversarial networks of randomly-seeded fields. *Nature Machine Intelligence*, 4(12):1165–1173, 2022.

[65] V. Guillemin and A. Pollack. Differential Topology (AMS Chelsea Publishing). page 222, 2010.

[66] V. Guillemin and A. Pollaek. Differential topology. 1974.

[67] A. Guo and M. D. Graham. Blending data and physics for reduced-order modeling of systems with spatiotemporal chaotic dynamics. *arXiv*, 7 2025.

[68] A. Guo, D. Floryan, and M. D. Graham. Self-similar, spatially localized structures in turbulent pipe flow from a data-driven wavelet decomposition. *Journal of Fluid Mechanics*, 971:A9, 9 2023.

[69] A. Guo, G. T. Craven, J. E. Santos, and C. D. Young. Reconstructing dissipative dynamical systems from spatially and temporally sparse sensors. In *NeurIPS 2024 Workshop, Machine Learning and the Physical Sciences*, 2024.

[70] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, jun 2021.

[71] F. Hamilton, T. Berry, and T. Sauer. Ensemble kalman filtering without a model. *Physical Review X*, 6(1):011021, 2016.

[72] J. M. Hamilton, J. Kim, and F. Waleffe. Regeneration mechanisms of near-wall turbulence structures. *Journal of Fluid Mechanics*, 287:317–348, 3 1995.

[73] G. He, J. Wang, and A. Rinoshika. Orthogonal wavelet multiresolution analysis of the turbulent boundary layer measured with two-dimensional time-resolved particle image velocimetry. *Physical Review E*, 99(5):053105, may 2019.

[74] G. He, J. Wang, and A. Rinoshika. Orthogonal wavelet multiresolution analysis of the turbulent boundary layer measured with two-dimensional time-resolved particle image velocimetry. *Physical review. E*, 99(5-1), may 2019.

[75] H. He and Y. Li. Learning Vector Fields of Differential Equations on Manifolds with Curvature-Adaptive Geometrically Constrained SINDy. *Communications in Nonlinear Science and Numerical Simulation*, page 109406, oct 2025.

[76] M. R. Head and P. Bandyopadhyay. New aspects of turbulent boundary-layer structure. *Journal of Fluid Mechanics*, 107:297–338, 1981.

[77] L. H. Hellström, I. Marusic, and A. J. Smits. Self-similarity of the large-scale motions in turbulent pipe flow. *Journal of Fluid Mechanics*, 792:R1, mar 2016.

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

[79] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

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

[81] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge Monographs on Mechanics. Cambridge University Press, 2 edition, 2012.

[82] K. HORNIK, M. STINCHCOMBE, and H. WHITE. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[83] M. Hultmark, M. Vallikivi, S. C. Bailey, and A. J. Smits. Turbulent pipe flow at extreme Reynolds numbers. *Physical Review Letters*, 108, 2 2012.

[84] N. Hutchins and I. Marusic. Evidence of very long meandering features in the logarithmic region of turbulent boundary layers. *Journal of Fluid Mechanics*, 579:1–28, 2007.

[85] N. Hutchins, K. Chauhan, I. Marusic, J. Monty, and J. Klewicki. Towards reconciling the large-scale structure of turbulent boundary layers in the atmosphere and laboratory. *Boundary-Layer Meteorology*, 145(2):273–306, 2012.

[86] J. Hwang and H. J. Sung. Wall-attached structures of velocity fluctuations in a turbulent boundary layer. *Journal of Fluid Mechanics*, 856:958–983, 2018.

[87] Y. Hwang. Statistical structure of self-sustaining attached eddies in turbulent channel flow. *Journal of Fluid Mechanics*, 767:254–289, mar 2015.

[88] Y. Hwang and C. Cossu. Linear non-normal energy amplification of harmonic and stochastic forcing in the turbulent channel flow. *Journal of Fluid Mechanics*, 664: 51–73, dec 2010.

[89] Y. Hwang, N. Hutchins, and I. Marusic. The logarithmic variance of streamwise velocity and conundrum in wall turbulence. *Journal of Fluid Mechanics*, 933:A8, feb 2022.

[90] V. Iakovlev, M. Heinonen, and H. Lähdesmäki. Learning space-time continuous latent neural pdes from partially observed states. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[92] C. E. P. D. Jesús, A. J. Linot, and M. D. Graham. Building symmetries into data-driven manifold dynamics models for complex flows. *arXiv*, 2023.

[93] S. Y. Jhin, M. Jo, T. Kong, J. Jeon, and N. Park. ACE-NODE: Attentive Co-Evolving Neural Ordinary Differential Equations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 736–745, New York, NY, USA, aug 2021. ACM.

[94] J. Jiménez. Cascades in wall-bounded turbulence. *Annual Review of Fluid Mechanics*, 44:27–45, 2012.

[95] J. Jiménez. Near-wall turbulence. *Physics of Fluids*, 25(10):101302, oct 2013.

[96] J. Jiménez and A. Pinelli. The autonomous cycle of near-wall turbulence. *Journal of Fluid Mechanics*, 389:335–359, 1999.

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

[98] Y. Jodai and G. E. Elsinga. Experimental observation of hairpin auto-generation events in a turbulent boundary layer. *Journal of Fluid Mechanics*, 795:611–633, may 2016.

[99] B. Johnson and S. B. Munch. An empirical dynamic modeling framework for missing or irregular samples. *Ecological Modelling*, 468:109948, 2022.

[100] P. Karnakov, S. Litvinov, and P. Koumoutsakos. Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks. *PNAS Nexus*, 3, 12 2023.

[101] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[102] A. K. Kassam and L. N. Trefethen. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, jan 2005.

[103] G. Katul and B. Vidakovic. The partitioning of attached and detached eddy motion in the atmospheric surface layer using Lorentz wavelet filtering. *Boundary-Layer Meteorology*, 77(2):153–172, 1996.

[104] G. Katul and B. Vidakovic. Identification of low-dimensional energy containing/flux transporting eddy motion in the atmospheric surface layer using wavelet thresholding methods. *Journal of the Atmospheric Sciences*, 55(3):377–389, 1998.

[105] G. G. Katul, M. B. Parlange, and C. R. Chu. Intermittency, local isotropy, and non-gaussian statistics in atmospheric surface layer turbulence. *Physics of Fluids*, 6(7): 2480–2492, 1994.

[106] G. Kawahara, M. Uhlmann, and L. van Veen. The significance of simple invariant solutions in turbulent flows. *Annual Review of Fluid Mechanics*, 44:203–225, 2012.

[107] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker. Time2vec: Learning a vector representation of time. *arXiv*, 2019.

[108] M. B. Kennel, R. Brown, and H. D. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, 45(6):3403–3411, mar 1992.

[109] J. Kim, K. Lee, D. Lee, S. Y. Jhin, and N. Park. Dpm: A novel training method for physics-informed neural networks in extrapolation. *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, 9B:8146–8154, 12 2020.

[110] K. C. Kim and R. J. Adrian. Very large-scale motion in the outer layer. *Physics of Fluids*, 11(2):417–422, 1999.

[111] K. C. Kim and R. J. Adrian. Very large-scale motion in the outer layer. *Physics of Fluids*, 11(2):417–422, 1999.

[112] S. J. Kline, W. C. Reynolds, F. A. Schraub, and P. W. Runstadler. The structure of turbulent boundary layers. *Journal of Fluid Mechanics*, 30(4):741–773, dec 1967.

[113] A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Proceedings of the USSR Academy of Sciences*, 1941.

[114] K. Kontolati, S. Goswami, G. E. Karniadakis, and M. D. Shields. Learning nonlinear operators in latent spaces for real-time predictions of complex dynamics in physical systems. *Nature Communications*, 15(1):5101, 2024.

[115] E. D. Koronaki, N. Evangelou, C. P. Martin-Linares, E. S. Titi, and I. G. Kevrekidis. Nonlinear dimensionality reduction then and now: AIMs for dissipative PDEs in the ML era. *Journal of Computational Physics*, 506:112910, 2024.

[116] V. Kulandaivelu. *Evolution of zero pressure gradient turbulent boundary layers from different initial conditions*. PhD thesis, University of Melbourne, 2012.

[117] M. Kumar, M. D. Graham, M. Kumar, and M. D. Graham. Nested travelling wave structures in elastoinertial turbulence. *Journal of Fluid Mechanics*, 993:A8, sep 2024.

[118] Y. Kuramoto. Diffusion-Induced Chaos in Reaction Systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 1978.

[119] H. Lee, A. B. Rajat, R. Andrew, and Y. Ng. Efficient sparse coding algorithms. *Advances in Neural Information Processing Systems*, 19, 2006.

[120] K. Lee and K. T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404: 108973, 2020.

[121] M. Lee and R. D. Moser. Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. *Journal of Fluid Mechanics*, 774:395–415, 2015.

[122] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27 (10):103111, 07 2017.

[123] X. Li, J. Harlim, D. Chakraborty, and R. Maulik. A Weak Penalty Neural ODE for Learning Chaotic Dynamics from Noisy Time Series. nov 2025.

[124] 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*, 9:1–29, apr 2008.

[125] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *ICLR 2021 Workshop on Integration of Deep Neural Models and Differential Equations*, 10 2020.

[126] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural Operator: Graph Kernel Network for Partial Differential Equations. *ArXiv*, cs.LG, 03 2020.

[127] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-Informed Neural Operator for Learning Partial Differential Equations. *ACM / IMS Journal of Data Science*, 1(3):1–27, sep 2024.

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

[129] A. J. Linot and M. D. Graham. Deep learning to discover and predict dynamics on an inertial manifold. *Physical Review E*, 101(6):062209, 06 2020.

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

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

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

[133] A. J. Linot and M. D. Graham. Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations. *Chaos*, 32(7):073110, 07 2022.

[134] A. J. Linot and M. D. Graham. Dynamics of a data-driven low-dimensional model of turbulent minimal Couette flow. *arXiv*, 2023.

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

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

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

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

[139] A. V. Little, Y. M. Jung, and M. Maggioni. Multiscale estimation of intrinsic dimensionality of data sets. In *AAAI Fall Symposium - Technical Report*, volume FS-09-04, pages 26–33, 2009.

[140] A. Liu, J. Axås, and G. Haller. Data-driven modeling and forecasting of chaotic dynamics on inertial manifolds constructed as spectral submanifolds. *Chaos*, 34(3): 033140, 03 2024.

[141] A. Lozano-Durán and J. Jiménez. Time-resolved evolution of coherent structures in turbulent channels: characterization of eddies and cascades. *Journal of Fluid Mechanics*, 759:432–471, 2014.

[142] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, mar 2021.

[143] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(4):041102, 2017.

[144] J. L. Lumley. *Stochastic Tools in Turbulence*, chapter 3.14, pages 80–82. Academic Press, 1970.

[145] J. L. Lumley. Coherent structures in turbulence. In *Transition and Turbulence*, pages 215–242. Elsevier, 1981.

[146] J. Ma, M. Y. Hussaini, O. V. Vasilyev, and F. X. Le Dimet. Multiscale geometric analysis of turbulence by curvelets. *Physics of Fluids*, 21(7), jul 2009.

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

[148] I. Marusic and J. P. Monty. Attached Eddy Model of Wall Turbulence. *Annual Review of Fluid Mechanics*, 51(1):49–74, jan 2019.

[149] I. Marusic, J. P. Monty, M. Hultmark, and A. J. Smits. On the logarithmic region in wall turbulence. *Journal of Fluid Mechanics*, 716, 2013.

[150] R. Maulik, B. Lusch, and P. Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3), mar 2021.

[151] N. McGreivy and A. Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6(10):1256–1269, sep 2024.

[152] B. J. McKeon. The engine behind (wall) turbulence: Perspectives on scale interactions. *Journal of Fluid Mechanics*, 817, apr 2017.

[153] B. J. McKeon, J. Li, W. Jiang, J. F. Morrison, and A. J. Smits. Further observations on the mean velocity distribution in fully developed pipe flow. *Journal of Fluid Mechanics*, 501:135–147, 2004.

[154] B. J. McKeon, C. J. Swanson, M. V. Zagarola, R. J. Donnelly, and A. J. Smits. Friction factors for smooth pipe flow. *Journal of Fluid Mechanics*, 511:41–44, 2004.

[155] M. A. Mendez, M. Balabane, and J. M. Buchlin. Multi-scale proper orthogonal decomposition (mPOD). *AIP Conference Proceedings*, 1978(1), jul 2018.

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

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

[158] C. Meneveau. Dual spectra and mixed energy cascade of turbulence in the wavelet representation. *Physical Review Letters*, 66(11):1450, 1991.

[159] L. Meng, S. Ding, and Y. Xue. Research on denoising sparse autoencoder. *International Journal of Machine Learning and Cybernetics*, 8(5):1719–1729, oct 2017.

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

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

[162] R. Moarref, A. S. Sharma, J. A. Tropp, and B. J. McKeon. Model-based scaling of the streamwise energy density in high-Reynolds-number turbulent channels. *Journal of Fluid Mechanics*, 734:275–316, nov 2013.

[163] A. T. Mohan, N. Lubbers, M. Chertkov, and D. Livescu. Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence. *Physical Review Fluids*, 8(1):014604, jan 2023.

[164] R. E. Mortensen. Infinite-Dimensional Dynamical Systems in Mechanics and Physics (Roger Temam). *SIAM Review*, 33(1):156–158, 1991.

[165] A. A. Nepomnyashchii. Stability of wavy conditions in a film flowing down an inclined plane. *Fluid Dynamics*, 9:354–359, 1975.

[166] T. B. Nickels, I. Marusic, S. Hafez, and M. S. Chong. Evidence of the k1-1 law in a high-Reynolds-number turbulent boundary layer. *Physical Review Letters*, 95(7): 074501, aug 2005.

[167] 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(11):115109, 2007.

[168] N. Omata and S. Shirayama. A novel method of low-dimensional representation for temporal behavior of flow fields using deep autoencoder. *AIP Advances*, 9(1):15006, jan 2019.

[169] S. Ouala, D. Nguyen, L. Drumetz, B. Chapron, A. Pascual, F. Collard, L. Gaultier, and R. Fablet. Learning latent dynamics for partially observed chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(10):103121, 2020.

[170] 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*, pages 8024–8035. Curran Associates, Inc., 2019.

[171] 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(2):24102, 2018.

[172] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos*, 28(4), apr 2018.

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

[174] E. Perlman, R. Burns, Y. Li, and C. Meneveau. Data exploration of turbulence simulations using a database cluster. *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, SC'07*, 2007.

[175] A. E. Perry. Turbulence Modelling Using Coherent Structures in Wakes, Plane Mixing Layers and Wall Turbulence. In *Perspectives in Turbulence Studies*, pages 115–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[176] A. E. Perry and M. S. Chong. On the mechanism of wall turbulence. *Journal of Fluid Mechanics*, 119:173–217, 1982.

[177] A. E. Perry and I. Marusic. A wall-wake model for the turbulence structure of boundary layers. Part 1. Extension of the attached eddy hypothesis. *Journal of Fluid Mechanics*, 298:361–388, sep 1995.

[178] A. E. Perry, S. Henbest, and M. S. Chong. A theoretical and experimental study of wall turbulence. *Journal of Fluid Mechanics*, 165(-1):163–199, apr 1986.

[179] A. E. Perry, J. D. Li, and I. Marusic. Towards a closure scheme for turbulent boundary layers using the attached eddy hypothesis. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 336(1640):67–79, jul 1991.

[180] A. E. Perry, I. Marusic, and J. D. Li. Wall turbulence closure based on classical similarity laws and the attached eddy hypothesis. *Physics of Fluids*, 6(2):1024–1035, feb 1994.

[181] G. Peyré and M. Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11:1–257, 3 2018.

[182] S. B. Pope. *Turbulent Flows*. Cambridge University Press, aug 2000.

[183] S. B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[184] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

[185] Pérez-De-Jesús, A. J. Linot, and M. D. Graham. Building symmetries into data-driven manifold dynamics models for complex flows, 2023.

[186] C. E. Pérez De Jesús, A. J. Linot, and M. D. Graham. Building symmetries into data-driven manifold dynamics models for complex flows. *Physical Review Fluids*, 2025.

[187] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. nov 2017.

[188] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686 – 707, 2019.

[189] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, feb 2020.

[190] S. Ravishankar, J. C. Ye, and J. A. Fessler. Image reconstruction: From sparsity to data-adaptive methods and machine learning. *Proceedings of the IEEE*, 108(1):86–109, 2020.

[191] L. F. Richardson. *Weather prediction by numerical process*. Cambridge University Press, 1922.

[192] R. Rico-Martinez, K. Krischer, I. Kevrekidis, M. Kube, and J. Hudson. Discrete-vs. continuous-time nonlinear signal processing of cu electrodissolution data. *Chemical Engineering Communications*, 118(1):25–48, 1992.

[193] S. K. Robinson. Coherent motions in the turbulent boundary layer. *Annual Review of Fluid Mechanics*, 23(1):601–639, 1991.

[194] F. Romor, G. Stabile, and G. Rozza. Non-linear Manifold Reduced-Order Models with Convolutional Autoencoders and Reduced Over-Collocation Method. *Journal of Scientific Computing*, 94(3):1–39, mar 2023.

[195] B. J. Rosenberg, M. Hultmark, M. Vallikivi, S. C. Bailey, and A. J. Smits. Turbulence spectra in smooth- and rough-wall pipe flow at extreme Reynolds numbers. *Journal of Fluid Mechanics*, 731:46–63, 9 2013.

[196] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[197] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. *J. Fluid Mech.*, 641:115, Dec. 2009.

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

[199] M. Sanchis-Agudo, Y. Wang, R. Arnau, L. Guastoni, J. Lim, K. Duraisamy, and R. Vinuesa. Easy attention: A simple attention mechanism for temporal predictions with transformers. *APL Computational Physics*, 1(1):016104, aug 2023.

[200] J. E. Santos, Z. R. Fox, A. Mohan, D. O'Malley, H. Viswanathan, and N. Lubbers. Development of the senseiver for efficient field reconstruction from sparse observations. *Nature Machine Intelligence*, 5(11):1317–1325, 2023.

[201] T. Sauer, J. A. Yorke, and M. Casdagli. Embedology. *Journal of Statistical Physics*, 65(3-4):579–616, nov 1991.

[202] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 07 1998.

[203] D. A. Serino, M. L. Klasky, B. T. Nadiga, X. Xu, and T. Wilcox. Reconstructing richtmyer-meshkov instabilities from noisy radiographs using low dimensional features and attention-based neural networks. *arXiv preprint arXiv:2408.00985*, 2024.

[204] A. Sholokhov, Y. Liu, H. Mansour, and S. Nabi. Physics-informed neural ode (pinode): embedding physics into models using collocation points. *Scientific Reports*, 13:10166, 6 2023.

[205] B. I. Shraiman, A. Pumir, W. van Saarloos, P. C. Hohenberg, H. Chaté, and M. Holen. Spatiotemporal chaos in the one-dimensional complex Ginzburg-Landau equation. *Physica D: Nonlinear Phenomena*, 57(3-4):241–248, aug 1992.

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

[207] T. R. Smith, J. Moehlis, and P. Holmes. Low-dimensional modelling of turbulence using the proper orthogonal decomposition: A tutorial. *Nonlinear Dynamics*, 41(1-3): 275–307, 2005.

[208] A. Solera-Rico, C. S. Vila, M. Gómez-López, Y. Wang, A. Almashjary, S. T. M. Dawson, and R. Vinuesa. -Variational autoencoders and transformers for reduced-order modelling of fluid flows. *Nature Communications*, 15(1):1361, 2024.

[209] J. Stark, D. S. Broomhead, M. E. Davies, and J. Huke. Takens embedding theorems for forced and stochastic systems. *Nonlinear Analysis: Theory, Methods & Applications*, 30(8):5303–5314, 1997.

[210] D. Stowell and R. E. Turner. Denoising without access to clean data using a partitioned autoencoder. sep 2015.

[211] G. Strang. Wavelets and dilation equations: A brief introduction. *SIAM Review*, 31 (4):614–627, 1989.

[212] F. Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

[213] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[214] H. Tennekes and J. L. Lumley. *A First Course in Turbulence*. The MIT Press, 1972.

[215] H. Tennekes and J. L. Lumley. *A First Course in Turbulence*, chapter 8.2, pages 258–259. MIT Press, 1972.

[216] T. Theodorsen. Mechanism of turbulence. *Proceedings of the Second Midwestern Conference on Fluid Mechanics*, pages 1–18, 1952.

[217] C. D. Tomkins and R. J. Adrian. Spanwise structure and scale growth in turbulent boundary layers. *Journal of Fluid Mechanics*, 490:37–74, sep 2003.

[218] A. Towne, O. T. Schmidt, and T. Colonius. Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. *Journal of Fluid Mechanics*, 847:821–867, jul 2018.

[219] A. A. Townsend. *The structure of turbulent shear flow*. Cambridge University Press, 1976.

[220] A. Tran, X. He, D. A. Messenger, Y. Choi, and D. M. Bortz. Weak-form latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 427, apr 2024.

[221] R. S. Troy, M. Jeff, and H. Philip. Low-Dimensional Modelling of Turbulence Using the Proper Orthogonal Decomposition: A Tutorial. Technical Report 1024, 2005.

[222] J. H. Tu. Dynamic Mode Decomposition: Theory and Applications. (September): 1–30, 2013.

[223] M. Vallikivi, B. Ganapathisubramani, and A. J. Smits. Spectral scaling in boundary layers and pipes at very high Reynolds numbers. *Journal of Fluid Mechanics*, 771: 303–326, 5 2015.

[224] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All you Need. *Advances in Neural Information Processing Systems*, 30, 2017.

[225] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):5999–6009, 6 2017.

[226] 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*, 126:191–217, 2020.

[227] 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(2213):20170844, 2018.

[228] P. R. Vlachas, G. Arampatzis, C. Uhler, and P. Koumoutsakos. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.

[229] V. A. Volkovich, D. S. Maltsev, S. Yu Melchakov, al, K. Sato, C. Shinozaki, S. Goto, and J. C. Vassilicos. Particle pair diffusion and persistent streamline topology in two-dimensional turbulence. *New Journal of Physics*, 6(1):65, jun 2004.

[230] F. Waleffe. On a self-sustaining process in shear flows. *Physics of Fluids*, 9(4):883–900, 1997.

[231] F. Waleffe. Exact coherent structures in channel flow. *Journal of Fluid Mechanics*, 435:93–102, may 2001.

[232] G. Wang, J. C. Ye, K. Mueller, and J. A. Fessler. Image reconstruction is a new frontier of machine learning. *IEEE Transactions on Medical Imaging*, 37(6):1289–1296, 2018.

[233] L. W. Wang, C. Pan, and J. J. Wang. Wall-attached and wall-detached eddies in proper orthogonal decomposition modes of a turbulent channel flow. *Physics of Fluids*, 34(9): 095124, sep 2022.

[234] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7 (40), oct 2021.

[235] Z. Wang and C. Guet. Reconstructing a dynamical system and forecasting time series by self-consistent deep learning. *arXiv preprint arXiv:2108.01862*, 2021.

[236] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in Time Series: A Survey. *IJCAI International Joint Conference on Artificial Intelligence*, 2023-Augus:6778–6786, 2023.

[237] J. Wentz and A. Doostan. Derivative-based SINDy (DSINDy): Addressing the challenge of discovering governing equations from noisy data. *Computer Methods in Applied Mechanics and Engineering*, 413:116096, aug 2023.

[238] J. P. Williams, O. Zahn, and J. N. Kutz. Sensing with shallow recurrent decoder networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 480(2298), sep 2024.

[239] J. P. Williams, O. Zahn, and J. N. Kutz. Sensing with shallow recurrent decoder networks. *Proceedings of the Royal Society A*, 480(2298):20240054, 2024.

[240] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[241] E. S. Winkel, J. M. Cutbirth, S. L. Ceccio, M. Perlin, and D. R. Dowling. Turbulence profiles from a smooth flat-plate turbulent boundary layer at high Reynolds number. *Experimental Thermal and Fluid Science*, 40:140–149, 2012.

[242] M. Yamada and K. Ohkitani. An identification of energy cascade in turbulence by orthonormal wavelet analysis. *Progress of Theoretical Physics*, 86(4):799–815, 1991.

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

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

[245] Y. Yin, M. Kirchmeyer, J.-Y. Franceschi, A. Rakotomamonjy, and P. Gallinari. Continuous pde dynamics forecasting with implicit neural representations. *arXiv preprint arXiv:2209.14855*, 2022.

[246] C. D. Young and M. D. Graham. Deep learning delay coordinate dynamics for chaotic attractors from partial observable data. *Physical Review E*, 107(3):034215, 2023.

[247] C. D. Young and M. D. Graham. Deep learning delay coordinate dynamics for chaotic attractors from partial observable data. *Physical Review E*, 107(3):034215, 2023.

[248] M. V. Zagarola and A. J. Smits. Mean-flow scaling of turbulent pipe flow. *Journal of Fluid Mechanics*, 373:33–79, 1998.

[249] S. V. Zelik. Attractors. then and now. *Russian Mathematical Surveys*, 78:635–777, 8 2022.

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

[251] 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. *Proceedings of the Royal Society A*, 478(2267):20220297, 2022.

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

[253] K. Zeng, C. E. Pérez-De-Jesús, A. J. Fox, and M. D. Graham. Autoencoders for discovering manifold dimension and coordinates in data from complex dynamical systems. *MLST*, 5(2):025053, may 2024.

[254] J. Zhang, P. Zhang, B. Kong, J. Wei, and X. Jiang. Continuous Self-Attention Models with Neural ODE Networks. In *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, volume 16, pages 14393–14401, may 2021.

[255] D. Zhao, B. Guo, J. Wu, W. Ning, and Y. Yan. Robust feature learning by improved auto-encoder from non-Gaussian noised images. *IST 2015 - 2015 IEEE International Conference on Imaging Systems and Techniques, Proceedings*, oct 2015.

[256] C. Zhou and R. C. Paffenroth. Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674, New York, NY, USA, aug 2017. ACM.

[257] J. Zhou, R. J. Adrian, S. Balachandar, and T. M. Kendall. Mechanisms for generating coherent packets of hairpin vortices in channel flow. *Journal of Fluid Mechanics*, 387: 353–396, may 1999.

[258] Z. Zhou, C. X. Xu, and J. Jiménez. Interaction between near-wall streaks and large-scale motions in turbulent channel flows. *Journal of Fluid Mechanics*, 940:A23, jun 2022.

[259] E. Özalp, G. Margazoglou, and L. Magri. Reconstruction, forecasting, and stability of chaotic dynamics from partial data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(9), Sept. 2023.

ProQuest Number: 32403426

INFORMATION TO ALL USERS
The quality and completeness of this reproduction is dependent on the quality
and completeness of the copy made available to ProQuest.



Distributed by
ProQuest LLC a part of Clarivate (2026).
Copyright of the Dissertation is held by the Author unless otherwise noted.

This work is protected against unauthorized copying under Title 17,
United States Code and other applicable copyright laws.

This work may be used in accordance with the terms of the Creative Commons license
or other rights statement, as indicated in the copyright statement or in the metadata
associated with this work. Unless otherwise specified in the copyright statement
or the metadata, all rights are reserved by the copyright holder.

ProQuest LLC
789 East Eisenhower Parkway
Ann Arbor, MI 48108 USA