Department of Aeronautics
Imperial College London

Scientific Machine Learning for the Analysis and Reconstruction of Turbulent Flows

Yaxin Mo

16 March 2026

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in the Department of Aeronautics.

Statement of Originality

I declare that this thesis is my own work and has not been submitted for any other degree qualification. Where I have used the work of others, full acknowledgement has been given.

Yaxin Mo

August 2025

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Abstract

In this thesis, we tackle two key research questions in the understanding and modelling of turbulent flows. (i) How do we compress and find structures in high-dimensional datasets, and (ii) How do we reconstruct the physical states of flows from sparse and noisy measurements?

Data compression and flow structure identification can be achieved using modal decomposition, commonly using linear methods such as proper orthogonal decomposition (POD). Because of its linearity and orthogonality, POD is interpretable but not efficient in compressing highly nonlinear data. On the other hand, autoencoders can achieve larger compressions, but they are less interpretable. To improve the interpretability of autoencoders, we propose the decoder decomposition, a post-processing method connecting POD modes with the latent space. We apply our method to an experimental turbulent wake of a bluff body at Reynolds number $\sim$200,000, and find that the physical interpretation of the latent variables depends on the user-selected latent dimension.

Flow reconstruction from sparse and noisy data is achieved by combining known flow physics with neural networks. When ground truth data is unavailable, we propose a physics-constrained convolutional neural network and a mean-enforced loss to reconstruct the full flow field from sparse, noisy data. We demonstrate the capability of our method by reconstructing both 2D and 3D synthetic turbulent datasets from sparse, noisy measurements. Our method successfully reconstructs 2D turbulent flows from $<$1% grid points with a signal-to-noise ratio as low as 5, even in intermittent regimes. The same method can also reconstruct 3D turbulent flows from 2.7% grid points. We also propose a weight-sharing network, which takes advantage of the homogeneous direction in the flows. Using the weight-sharing network, we reconstruct the 3D turbulent datasets from three measurement planes.

Acknowledgements

First, I would like to thank my supervisor, Professor Luca Magri, for providing me with the opportunity to work and grow under his guidance. Near the beginning, I had once tried to explain to someone what a PhD is by likening it to being an apprentice to a scientist. I did not realise how true the analogy was until later, when I realised how much I had learned from Luca. He gave me the freedom to explore my subject area, but kept me on track in our meetings. He gave encouragement when I needed it, and stern words when I needed them, too. It is thanks to these interactions that I am finishing this PhD with more love for machine learning and fluids than when I began.

This thesis is made possible by the contributions of our collaborators. I would like to thank Tullio for being my co-author and helping me with writing my first paper. And I thank Nikos for helping me set up Xcompact3D. I acknowledge Dr Georgios Rigas and Professor Jonathan Morrison for providing us with the experimental data. I further thank Georgios for the insightful discussions surrounding autoencoders and the physics of flows. I am also grateful to the Department of Aeronautics for funding my studentship and for the coffee. I also owe parts of my work to the open-source community. Some of you on StackOverflow are true legends.

I would like to express my genuine gratitude to my academic brothers and sisters, Daniel, Defne and Elise, who started with me and went through the milestones together. Thank you for the late-night coding sessions and the occasional chair racing games in the office; for being the sounding board when I test new ideas and checking the derivations later; and for providing a sympathetic ear when I ran into trouble and doubted myself. And to Osama, who joined later but is still a dear friend, the cactus is yours now. Take care of it. Without them, who knew what I would have done during some of the tough times and where I would be now.

Outside of my research group, I would like to thank friends and collaborators from Hazelab, who, without them, I would not have even started a PhD. Particularly Rik, both for being my collaborator and for the friendly advice given over the years at our coffee meetings. I am also grateful for friends from both university and from home. To Alice, Ke and my flatmates of 4 years, who were there for me during the lonely Covid. To colleagues in both 208 and 308 who made coming to work fun. And to Bingru, Xiaoyi, Sharada and Yichen, for the adventures we

have shared outside of work that got me through stressful times.

Last but not least, I am thankful for my family, who always made time to answer my phone calls. I thank my parents, who raised me and gave me the courage to be my own person. And I thank my grandparents, who always showed me support despite not being completely sure what I was doing – I haven't gone to lectures in the past four years! It is because of their support that I am where I am today, writing the last part of my thesis. I cannot ask for better people in my life.

Contents

1 Introduction 25

1.1 Review of data-driven methods in fluid mechanics . . . 25

1.1.1 Extracting features from complete datasets . . . 25

1.1.2 Reconstruction from incomplete datasets . . . 27

1.2 Objective . . . 30

1.3 Overview of this thesis . . . 30

2 Datasets and methods for processing data 32

2.1 Datasets . . . 32

2.1.1 Experimental data . . . 32

2.1.2 Generating synthetic data . . . 32

2.2 Methods for processing data . . . 35

2.2.1 Proper Orthogonal Decomposition . . . 35

2.2.2 Other quantities of interest . . . 35

3 Extracting nonlinear modes via decoder decomposition 38

3.1 Introduction . . . 38

3.2 Preprocessing the datasets . . . 40

3.2.1 Unsteady laminar wake of a two-dimensional cylinder . . . 40

3.2.2 Turbulent wake of a bluff body from wind-tunnel experiments . . . 43

3.3 Autoencoders . . . 45

3.3.1 Standard autoencoders . . . 46

3.3.2 Mode-decomposing autoencoders (MD-AE) . . . 48

3.4 Decoder Decomposition . . . 48

3.4.1 The decoder decomposition for standard autoencoders . . . 49

3.4.2 Decoder decomposition for MD-AEs . . . 50

3.5 Decomposition of the unsteady laminar wake . . . 50

3.5.1 The latent space of a mode-decomposing autoencoder . . . 51

3.5.2 The latent space of standard autoencoders . . . 53

3.5.3 Physical interpretation is more difficult when unsuitable latent dimension is used . . . 58
3.6 Autoencoder decomposition of the turbulent wake from wind-tunnel experiments 60
3.6.1 The limiting factor is network size for turbulence . . . 60
3.6.2 Filtering unwanted latent variables . . . 63
3.7 Conclusion . . . 66

4 Reconstructing 2D turbulent flows from sparse measurements 68
4.1 Introduction . . . 69
4.2 Data and preprocessing . . . 70
4.2.1 Laminar unsteady wake . . . 70
4.2.2 2D Turbulent Kolmogorov flow . . . 70
4.3 Methodology . . . 71
4.3.1 Taking measurements . . . 71
4.3.2 Physics-constrained dual-branch convolutional neural network . . . 72
4.3.3 Loss functions . . . 73
4.4 Reconstructing the laminar wake . . . 78
4.4.1 Reconstructing the laminar wake from non-noisy measurements . . . 78
4.4.2 Reconstructing the laminar wake from noisy observations . . . 81
4.5 Reconstructing the 2D turbulent Kolmogorov flow . . . 84
4.5.1 Reconstructing from non-noisy measurements . . . 84
4.5.2 Reconstructing from noisy measurements . . . 86
4.5.3 Reconstructing from sparse noisy measurements . . . 89
4.5.4 Discussion on periodic boundary conditions . . . 90
4.6 Conclusion . . . 93

5 Reconstructing 2D intermittent and 3D turbulent flows 95
5.1 Introduction . . . 95
5.2 Datasets and methods . . . 97
5.2.1 2D turbulent Kolmogorov flow with extreme events . . . 97
5.2.2 3D turbulent Kolmogorov flow . . . 97
5.2.3 Measurements and network structures . . . 98
5.3 Reconstructing 2D Kolmogorov flow with extreme events . . . 99
5.4 Reconstructing 3D Kolmogorov flows with PC-DualConvNet . . . 103
5.5 Conclusion . . . 104

6 Reconstructing 3D turbulent flows from plane measurements 107
6.1 Introduction . . . 107
6.2 Methodology . . . 109

6.2.1 Dataset and measurements . . . 109
6.2.2 The weight-sharing network . . . 110
6.3 Reconstructing 3D Kolmogorov flow from planes . . . 111
6.3.1 Reconstructing from a single cross-plane . . . 113
6.4 Reconstructing 3D Kolmogorov flow from noisy measurements . . . 114
6.4.1 Selecting the hyperparameters . . . 115
6.4.2 Results from noisy measurements . . . 116
6.5 Conclusion . . . 123

7 Conclusion and future work **124**
7.1 Summary of work . . . 124
7.2 Future work . . . 126

Appendices **139**

A Machine learning model hyperparameters and training details **140**
A.1 Autoencoders in Chapter 3 . . . 140
A.2 Model architectures and training in Chapter 4 . . . 144
A.2.1 Grid sensitivity study for the reconstruction of the Kolmogorov flow . . . 144
A.2.2 Model and training parameters . . . 144
A.2.3 Learning curves . . . 150
A.3 Model architectures in Chapter 5 . . . 152
A.4 Model architectures and training in Chapter 6 . . . 153
A.4.1 Hyperparameters and the minimum number of planes . . . 153
A.4.2 Model training time . . . 156
A.4.3 Model and training parameters for noisy data . . . 157

B Miscellaneous **159**
B.1 Codes and data availability . . . 159
B.2 Linear Autoencoders and POD . . . 159
B.3 Effect of single-precision floating-point in the training of PC-DualConvNet . . . 160
B.4 2D Kolmogorov flow at different Reynolds numbers . . . 161
B.5 Effect of the Fourier layer in PC-DualConvNet . . . 163
B.6 Forcing divergence-free condition . . . 164

List of Figures

2.1 Experimental set-up, reproduced from (Rigas 2021). The dimensions x_1 and x_2 are the measured location nondimensionalized by the diameter of the bluff body. The black dots mark the location of the pressure sensors. 33

3.1 Snapshots of the streamwise velocity of the laminar wake dataset at different times within the same period. The area bounded by the grey box is used for training. 40

3.2 POD of the laminar wake $\boldsymbol{U}$. Left: the percentage energy contained in the first six POD modes of the unsteady wake $\boldsymbol{\Phi}^{lam}$ (data mode). Data modes 1&2, 3&4 and 5&6 contain similar flow energy and oscillate at the same frequency but out of phase. Centre: phase plot of the first two data time coefficients. Right: the frequency spectrum of the data and the data time coefficients 1, 3, 5 and 7, normalized by the standard deviation. The data contains the vortex shedding frequency and its harmonics. (Since each pair has the same frequency spectrum, only the odd data modes are shown here). 41

3.3 The first six POD modes (data modes) of the unsteady wake behind a cylinder dataset $\boldsymbol{\Phi}^{lam}_{:,1},\ldots,\boldsymbol{\Phi}^{lam}_{:,6}$. 42

3.4 The wind-tunnel pressure dataset $\boldsymbol{P}$. Left: Mean pressure. Centre: RMS pressure. Right: The premultiplied PSD ($St\cdot$ PSD) of the wind-tunnel dataset, with peaks at $St \approx 0.002, 0.06$ and $St \approx 0.2$ and its harmonics. The peaks correspond to the three-dimensional rotation of the wake, the pulsation of the vortex core, and the vortex shedding and its harmonics, respectively. 43

3.5 POD of the wind-tunnel dataset $\boldsymbol{P}$. Left: percentage energy of the first 10 data modes. Right: cumulative percentage energy of POD modes. The reconstruction of the pressure dataset to 95% energy needs 21 data modes. 44

3.6 Left: the premultiplied PSD ($St\cdot$ PSD) of their associated time coefficients $\boldsymbol{A}^{exp}_{1,:}$ to $\boldsymbol{A}^{exp}_{5,:}$. Middle: the temporal evolution of the data time coefficients, $\boldsymbol{A}^{exp}_{1,:}$ to $\boldsymbol{A}^{exp}_{5,:}$, for the first 10 seconds of the experiment. Right: the first five data modes of the wind-tunnel dataset $\boldsymbol{\Phi}^{exp}_{:,1}$ to $\boldsymbol{\Phi}^{exp}_{:,5}$. 44

3.7 The schematics of the standard autoencoder (AE). Different AEs are employed for different datasets and tests. The AE architecture and dataset for each test are listed in Table 3.1. (a) AE for decomposing the laminar cylinder wake $\boldsymbol{U}$ with three convolution layers in both the encoder and the decoder. The hyperparameters are in tables A.1 and A.2 in the Appendix. (b) AE for decomposing the wind-tunnel pressure data $\boldsymbol{P}$ with 5 feedforward layers. The input is a flattened vector of measurement taken from all sensors at time t. The hyperparameters are given in tables A.3 and A.4 in the Appendix. 47

3.8 Schematic of the MD-AE (Murata et al. 2020) with two latent variables as an example. Each latent variable is decoded by a decoder to produce a decomposed field. The sum of the decomposed fields $\boldsymbol{M}^1$ and $\boldsymbol{M}^2$ is the output of the MD-AE. 48

3.9 Results of training the MD-AE on the laminar wake dataset with a latent dimension of 2. a) Both latent variables are periodic in time. b) Frequency spectrum of the latent variables, normalised by their standard deviation, compared with the frequency of the data. The latent variables both contain the vortex shedding frequency and the second harmonic of the vortex shedding frequencies. c) A snapshot output of the MD-AE (mean flow added) and the POD modes of the decomposed fields. The POD modes of decomposed field 1 are similar to data mode 1,3 and 6 and the POD modes of decomposed field 2 are similar to data mode 2,4 and 5. 52

3.10 Results from the MD-AE trained with the laminar wake dataset with 2 latent variables. The first four decoder coefficients plotted against the latent variables. Both latent variables affect the magnitude of the first four POD modes of the data in the output of the MD-AE. 53

3.11 Results from the MD-AE trained with the laminar wake dataset with 2 latent variables. The equivalent energy of the two decomposed fields of the MD-AE, showing the first four POD modes of the data. All POD modes of data are present in both decomposed fields 54

3.12 The MSE of AEs trained with the unsteady wake dataset with different numbers of latent variables averaged over five repeats each. The error bars represent the standard deviation of the repeats. At $N_z = 2$ the MSE is approximately 1.3×10^5. 54

3.13 Results from the AE trained with the laminar wake dataset with two latent variables. a) Phase plot of two latent variables. The unit circle indicates harmonic oscillations. b) Time trace of the two latent variables. The latent variables are periodic and 90° out of phase, behaving the same as the first two data time coefficients. c) The frequency spectrum of the latent variables, compared to the frequency in the dataset. The variables are normalized by their standard deviation, σ, before applying the Fourier Transform for visualisation. 55

3.14 Contour plots of the first four decoder coefficients of an AE with two latent variables, trained with the laminar wake dataset. The contours show the values of the decoder coefficients. The grey circle labels the values of the latent variables observed during training, which shows the dynamics of the dataset. The images in the last column show the output of the AE at point 1&2 labelled on the contour plots. 56
3.15 Results from the AE trained with the laminar wake dataset with 2 latent variables. The average rate of change of the decoder coefficients. 57
3.16 Results from the AE trained with the laminar wake dataset with 1 latent variable. a) The latent variable approximates the discontinuity caused by the angle moving from 2π to 0. b) The frequency spectrum of the latent variable, normalized by its standard deviation, compared with the data frequencies. The latent variables contain frequencies that do not exist in the dataset. 58
3.17 Results from the AE trained with the laminar wake dataset with three latent variables. a) The time trace of the latent variables. b) The frequency spectrum of the latent variables, normalized by their standard deviations. The latent variables contain the vortex shedding and the first harmonic frequency. c) The averaged rate of change of decoder coefficients due to latent variables of an AE with three latent variables. Latent variables $Z_{2,:}$ and $Z_{3,:}$ have the same contributions towards the first six data modes, meaning that both latent variables carry the same information. 59
3.18 The AE trained with the wind-tunnel dataset with two latent variables. Top left: predicted RMS pressure. Top right: predicted instantaneous pressure. Bottom left: the premultiplied overall PSD of the prediction. Bottom right: the premultiplied PSD of the latent variables of the AE trained with two latent variables, normalized by their standard deviation. 60
3.19 Same as Figure 3.18 with three latent variables. 61
3.20 The schematics and training process of the decoder-only network. The decoder has 99% more trainable parameters than the decoder in the $N_z = 2$ AE discussed in Section 3.6.1. 62
3.21 Left: predicted RMS pressure of the decoder-only network. Right: The premultiplied PSD of the predicted pressure from the AE with two and three latent variables, and from the large decoder-only network, trained with the wind-tunnel dataset. 62
3.22 AE with different numbers of latent variables and the same hyperparameters, trained with the wind-tunnel dataset. The loss stops decreasing at $N_z \approx 28$. . . . 63

3.23 RMS predicted pressure and premultiplied PSD of the prediction of the AE trained with the wind-tunnel dataset with 28 latent variables. The network attains an MSE equivalent to the reconstruction with 30 POD modes. 63
3.24 The AE trained with the wind-tunnel dataset with 28 latent variables. a) Equivalent energy of the predicted pressure compared to the POD eigenvalues of the reference data. b) The average rate of change of decoder coefficients 1 and 2 due to latent variables, normalized by the standard deviation of the latent variables. The decoder coefficients 1 and 2 are direct analogies of the data POD time coefficients 1 and 2, which are associated with vortex shedding. 64
3.25 Prediction obtained with the AE with 28 latent variables from the filtered latent variables $\boldsymbol{Z}_f$ and the reference data. 65
3.26 Premultiplied PSD of the prediction obtained with the AE with 28 latent variables from the filtered latent variables. All peaks (except for $St \approx 0.2$) are filtered. . . . 65
3.27 POD on the prediction obtained with the AE with 28 latent variables from the filtered latent variables. a) Equivalent energy of the filtered prediction $\hat{\boldsymbol{P}}_f$ compared to the POD eigenvalues of the reference data. Amongst the five most energetic POD modes, the filtered prediction shows a large amount of energy only in data modes 1 and 2, which is the goal we set. b) POD modes 1 and 2 of $\hat{\boldsymbol{P}}_f$, which contain over 99% of the flow energy of $\hat{\boldsymbol{P}}_f$. These two modes show the same frequency peak at $St \approx 0.2$ and have anti-symmetric spatial structures. These two modes represent vortex shedding. 66

4.1 Snapshots of streamwise (left) and wall-normal (centre) velocity and pressure (right) at $t = 2.5$, where $t = 0$ is the end of the transient period discarded when generating the dataset. The area indicated by the brown box is the area considered by the neural networks, with its coordinates $\boldsymbol{x} = [x_1^* - 3.0, x_2^*]^T$ shown in brown. 71
4.2 Top row, left to right: The time-averaged velocity in x_1 direction (u_1), x_2 direction (u_2) and pressure (p). Bottom row, left to right: The global dissipation with a time delay of $\tau \approx 5$ and the turbulent kinetic energy spectrum. 72
4.3 Schematic of the two-branch Fourier convolutional neural network. The number and size of layers are for illustration. The Fourier Transform layer (FT) and the inverse Fourier Transform layer (FT^{-1}) are optional and are activated (or deactivated) for different loss functions. 73
4.4 Computing the snapshot-enforced loss, which involves enforcing measurements $\xi(\boldsymbol{D})$ into $\hat{\boldsymbol{D}}$ to obtain $\boldsymbol{\Phi}$. The snapshot-enforced loss is the physics loss of $\boldsymbol{\Phi}$. . . . 76

4.5 Computing the mean-enforced loss. The mean is taken over a batch of inputs during training, which involves enforcing the measurements' mean $\overline{\xi}(\boldsymbol{D})$ into $\overline{\hat{\boldsymbol{D}}}$, and adding the fluctuating field $\hat{\boldsymbol{D}}'$ to obtain $\boldsymbol{\Phi}$. The snapshot-enforced loss is the weighted sum of the physics loss of $\boldsymbol{\Phi}$ and the data loss $\mathscr{L}_o(\hat{\boldsymbol{D}}, \boldsymbol{D})$. 78
4.6 Sensor placement for laminar wake, non-noisy measurements. The measurements consist of the pressure inlet measurement at $\boldsymbol{x}_{in}$ (red) and measurements of all components at $\boldsymbol{x}_s$ (black). 79
4.7 Reconstructed laminar wake from non-noisy measurements. (a) Snapshots of the reconstructed laminar wake compared with the reference and interpolated snapshot. The last two columns show the instantaneous absolute error of the interpolated snapshot and the reconstructed snapshot, respectively. (b) The probability density of the fluctuating components of the entire flow. Both (a) and (b) show that the reconstructed flow is a closer match with the reference data than mere interpolation. 80
4.8 Sensor placement for laminar wake, noisy measurements. The measurements consist of the pressure inlet measurement at $\boldsymbol{x}_{in}$ (red) and measurements of all components at $\boldsymbol{x}_s$ (black). 81
4.9 Comparison of loss functions for different SNR. The reference physics loss is the residual of the Navier-Stokes equations solved numerically. The mean (markers) and standard deviation (error bars) are calculated over five repeats with randomly initialised weights. 82
4.10 Reconstructed laminar wake from noisy measurements at SNR=20 (4.10a), 10 (4.10b) and 5 (4.10c). 83
4.11 Random sensors used to reconstruct 2D turbulent Kolmogorov flow from non-noisy measurements, showing the input sensors (red) and general sensors (black). 84
4.12 Reconstructed 2D turbulent Kolmogorov flow from non-noisy measurements with the snapshot-enforced loss, at $t = 1.7$. 85
4.13 The flow reconstructed from ten input sensors and ten general sensors. The sensor placement is shown in the top left-hand panel. The reconstruction has recovered the instantaneous vorticity fields and the 2D turbulent kinetic energy spectrum. 86
4.14 (a) Sensor placement for reconstructing the 2D turbulent Kolmogorov flows from noisy measurements. (b) Mean and standard deviation of the reconstructed Kolmogorov flow from noisy measurements at different SNR using different loss functions. 86
4.15 The turbulent kinetic energy (TKE) spectrum. 87
4.16 Reconstructed Kolmogorov flow from noisy measurements at SNR=20 (4.16a), 10 (4.16b) and 5 (4.16c). 88

4.17 Reconstructed Kolmogorov flows from sparse noisy measurements at $SNR = 10$. Only twenty-four input sensors and general sensors are employed (the sensors are shown in the left panels). The relative errors for each reconstruction case are 70% (interpolated), 75% ($\mathscr{L}^c$), 30% ($\mathscr{L}^s$), and 45% ($\mathscr{L}^m$). 89
4.18 Learning curves for sparse and noisy measurements. The left panel shows the training loss, the right panel shows the physics loss (solid line) and the data loss (dotted line). When using either the snapshot-enforced or the mean-enforced loss, the physics and data losses decrease together. On the other hand, when using the softly-constrained loss, the data loss decrease while the physics loss increase. 90
4.19 The energy spectrum (a) and the second-order velocity structure function (b) of the reconstructed flow from non-noisy measurements. The same case was shown in Section 4.4.1, in Figure 4.12. 91
4.20 The longitudinal second-order structure function for the reconstructed flows from noisy measurements. 92
4.21 The flow reconstructed from ten input sensors and ten general sensors. This is a repeat of the case shown in Figure 4.13, but with periodic boundary hard-coded into the network architecture. 92
4.22 The longitudinal second-order structure function of the flow reconstructed from ten input and ten general sensors (a) without periodic padding. (b) with periodic padding. 93

5.1 The mean properties of the 3D Kolmogorov flow dataset. (a) The mean velocities and pressure are shown in 3D. (b) The turbulent kinetic energy spectrum against wavenumber, where an exponential decay of energy between low- to medium wavenumbers can be seen. (c) The mean u_1 averaged over each of the x directions, compared with the forcing term. The mean u_1 across x_2 shows a sinusoidal profile matching the frequency of the forcing term, meaning that the dataset has converged. 98
5.2 Sensor placement for reconstructing 2D Kolmogorov flow with extreme events. The measurements consist of the pressure inlet measurement at x_{in} (red) and measurements of all components at x_s (black). 99
5.3 Snapshots of the vorticity (left) and pressure (right) fields at different instances, where the snapshots at t=32 and 200 show the flow field during extreme events. In all the snapshots shown here, the reconstructed flow fields show structures that closely resemble the reference. Especially the vorticity field, where the interpolated flow has failed to show the correct structures. 100

5.4 Physical metrics of the reconstructed flow compared to the reference. Left: the energy spectrum of the reconstructed flow and the reference data. The spectrum shows exponential decay of energy. Right: The global dissipation of the reconstructed flow and the reference data. The black and blue dashed lines are the extreme event thresholds for the reference and reconstructed flow, respectively. The red vertical lines mark the four snapshots shown in Figure 5.3. The reconstructed flow has the same timing of extreme events as the reference data, even though the overall dissipation has a lower magnitude. 101
5.5 The second-order velocity structure function of the reconstructed flow compared to the reference. 102
5.6 Measurements from each sensor are plotted with a blue translucent line, and the lines from all sensors are overlaid to show the overall input trend. All values are normalised. We can see that not all extreme events (dotted line) have corresponding peaks in the pressure input, but the network is still able to capture the correct timing of all the extreme events. 102
5.7 Sensor placement for reconstructing 3D Kolmogorov flow. General sensors are shown in blue, and the pressure inlet sensors are shown in red. 103
5.8 Slices of the reconstructed and reference flow domain at an instance in time. The top row shows the vorticity magnitude (a) and pressure (b) on x_1-x_2 planes at two different x_3 values, and the bottom row ((c) and (d)) shows the x_2-x_3 planes at two different x_1 values. 105
5.9 Statistical properties of the reconstructed flow, compared with the reference. (a) Time-averaged vorticity magnitude. (b) Time-averaged pressure. (c) Probability density of the entire 3D domain. 106
5.10 The turbulent kinetic energy spectrum of the reconstructed flow (blue) and the reference flow (grey). The spectrum in shown in both logarithmic (a) and linear (b) scales. The reconstructed flow has successfully captured the energy spectrum up to wavenumber 10. When shown on a linear scale, the energy contained within eddies past wavenumber 10 is negligible. 106

6.1 Pressure is measured at the plane $x_2 = 0$. Three planes of velocities are taken at $x_1 = \pi$, $x_3 = 0.5\pi$, and 1.5π. The pressure is used as input to the network, and all measurements are used as collocation points. 109

6.2 Schematics of the weight-sharing network. The input pressure $\boldsymbol{P}_{in}$ (first white block) is a 2D matrix of pressure measurements at $x_2 = 0$. It is then passed through a 2D convolution layer and a fully-connected layer to produce another 2D matrix, which is then split into vectors along an axis, which will later become x_3 direction in the final output (group of white long blocks). Each vector is then passed through the same PC-DualConvNet (orange block) to produce an intermediate 2D result representing a $x_1 - x_2$ plane. These intermediate results are stacked along x_3 direction to form a 3D intermediate result, which is then passed through multiple 3D convolutions and reshaping layers to produce the final reconstructed flow $\hat{\boldsymbol{D}}$. 111
6.3 Time average of the volumes. From left to right, the columns are reference data, weight-sharing network, and PC-DualConvNet. 113
6.4 The energy spectrum of the reconstructed flow. 114
6.5 Two $x_1 - x_2$ planes taken from the reconstructed flow at $x_3 = 0.56\pi$ and π, which are unseen by the network during training. The measurements used in training are taken on planes at $x_3 = 0.5\pi$ and 1.5π. Moving further away from the measured plane, the instantaneous planes reconstructed by the weight-sharing network show a closer resemblance to the reference, whereas the PC-DualConvNet outputs the mean flow. An example of this difference can be seen in the bottom row, which shows a plane taken at $x_3 = \pi$. 114
6.6 The root mean square error (sqaure root of MSE) of of each reconstructed $x_1 - x_2$ plane. 115
6.7 Reconstructed flow from a single cross-plane. (a) The locations of the velocity measurement planes, consisting of one $x_1 - x_2$ measurement plane and a one $x_2 - x_3$ measurement plane, at the centre of the domain. (b) The turbulent kinetic energy of the reconstructed flow fields. (c) Two slices of the reconstructed flow at $x_3 = 1.09\pi$ and 1.72π, which are unseen by the network during training. The measurements used in training are taken on planes at $x_3 = \pi$. When further away from the measured plane, such as the bottom row at $x_3 = 1.72\pi$, the flow reconstructed by the weight-sharing network match the reference data well, while the flow reconstructed by the PC-DualConvNet has lost resemblance to the reference. The difference is especially true for the pressure field. 118
6.8 The root mean square error (sqaure root of MSE) of of each reconstructed $x_1 - x_2$ plane. 119

6.9 Quantities of interest for tests performed with different sets of hyperparameters for (a) the 3D PC-DualConvNet and (b) the weight-sharing network. Each data point represents a test with a distinct set of hyperparameters. Top panel: the validation sensor loss plotted against the physics loss for tests with different sets of hyperparameters, coloured by the relative error (not used in the selection process). The red star marks the set of hyperparameters selected. Validation sensor loss is computed at the plane at $x_3 = \pi$, which is not seen by the networks during training. Bottom panel: validation sensor loss against the training sensor loss. . 120
6.10 The mean flow. From left to right are: reference 3D data, reconstructed with the weight-sharing network, and reconstructed with the PC-DualConvNet. The top and bottom rows are the vorticity and pressure, respectively. 121
6.11 An instantaneous velocity u_1 and pressure p at different x_3. Columns show the reference flow, the flow with added white noise, the flow reconstructed by the weight-sharing network and the flow reconstructed by the PC-DualConvNet (left to right). The top row for each variable shows a plane at $x_3 = 0.5\pi$, which is part of the training dataset. The bottom row for each variable shows a plane at $x_3 = 0.94\pi$, which is unseen by the network during training, and also not a validation plane used for selecting the hyperparameters in Section 6.4.1. 122
6.12 The root mean square error (sqaure root of MSE) of of each reconstructed $x_1 - x_2$ plane. Reconstructed from noisy measurement planes. 122

A.1 Reconstruction error in response to changing the number of (left) pressure inputs $\boldsymbol{P}_{in}$ while holding the number of sensors constant at 200; and (right) sensors $\boldsymbol{D}(\boldsymbol{x}_s)$ while holding the number of pressure inputs constant at 80. This grid sensitivity test is performed on a Kolmogorov flow simulated with a different initial random seed. 145
A.2 The cyclic decay learning rate schedule applied to reconstruct the laminar flow (left) and the Kolmogorov flow (right), where α is the initial learning rate. 145
A.3 Learning curves of the reconstruction of the laminar wake dataset. Learning curves are shown for the test runs plotted in Figure 4.7 and 4.10. 151
A.4 Learning curves of the reconstruction of the turbulent Kolmogorov flow. Learning curves are shown for the test runs plotted in Figure 4.12, 4.13 and 4.16. . . . 151
A.5 The unweighted sum of the physics and sensor loss of the tests in the hyperparameter selection. Left: PC-DualConvNet. Right: weight-sharing network. 154
A.6 (Left) the mean squared error and (right) the physics loss of the reconstructed from fields from different number of $x_1 - x_2$ measurement planes. 156

B.1 (a) The energy contained in the Fourier modes of the $Re = 34$ flow. (b) The energy contained in the Fourier modes of the $Re = 40$ flow. (c) The global dissipation of the two Kolmogorov flow. Extreme dissipation events are observed at $Re = 40$ but not $Re = 34$. 162

B.2 Comparing the estimated power of noise (left panel) and the relative error (right panel) of the flow reconstructed with PC-DualConvNets with and without the FFT layer. The statistics are computed over 50 runs using different hyperparameters for each SNR. 164

B.3 Two slices of the reconstructed flow at $x_3 = 1.77$ and 3.14, which are unseen by the network during training. The measurements used in training are taken on planes at $x_3 = 1.57$ and 4.71. 165

List of Tables

2.1 Summary of the final datasets for the two bluff body cases. 34

3.1 Different autoencoder architectures and the training datasets. 46

4.1 Possible solutions and their corresponding theoretical loss values when training with the softly-constrained loss. The network needs to minimise both physics loss and sensor loss. Scenario 1 has a theoretical minimum loss of 0 and is ideal, but is also unlikely when different numerical schemes are used in data generation and loss computation, or if data is experimental. Scenario 2 is the trivial solution, where the outputs satisfy the governing equations but not the measurements. Depending on the sensor placements, the minimum $\mathscr{L}_o$ could be either approximately 0 or much larger than 0. Scenarios 3 and 4 are desired, but they also have a theoretical minimum total loss larger than 0. 75

4.2 Summary of the losses for the reconstructed laminar wake from non-noisy measurements. Values are shown in the format *mean* ± *standard deviation* calculated over 5 runs initiated with different random weights. 81

4.3 The snapshot-enforced loss leads to the lowest relative error (mean±standard deviation) when reconstructing the 2D turbulent Kolmogorov flow from non-noisy measurements. The relative error is measured over five repeated trainings with different random weight initialisation and sensor locations. 85

5.1 The mean and range *[min, max]* of the relative error and physics loss, tested over three runs using different random sensor placements. The reconstructed flows have relative errors that are approximately 50% compared to the errors of the interpolated flows. The case presented in the rest of this section is the case with the lowest physics loss. 100

5.2 The mean and standard deviation of the errors of the reconstructed 3D flow. . . . 103

6.1 The relative error ϵ, the physics loss $\mathscr{L}_p$, and the sensor loss $\mathscr{L}_o$ (*mean* ± *standard deviation*) of the reconstruction results from planes of measurements, averaged over five runs with different random initialisations of network weights. 112

6.2 The relative error and the physics loss (*mean* ± *standard deviation*) of the reconstruction results from planes of noisy measurements, averaged over five runs with different random noise and different initialisations of network weights. . . 117

A.1 The encoder used in all convolutional AEs and MD-AEs. 140
A.2 The decoder used in all convolutional AEs and MD-AEs. 141
A.3 The encoder used in feedforward AEs. 141
A.4 The decoder used in feedforward AEs. 142
A.5 The hyperparameters used in all networks. CosineDecayRestarts is a built-in tensorflow learning rate schedule (Martín Abadi et al. 2015, Loshchilov & Hutter 2017). 142
A.6 Large decoder used in section 3.6.1 143
A.7 The network structure and hyperparameters used to reconstruct the laminar wake from non-noisy measurements. 146
A.8 The structure and hyperparameters of the network using the softly-constrained loss, from noisy measurements. Showing the difference from A.7 only. 146
A.9 The structure and hyperparameters of the network using the snapshot-enforced loss, from noisy measurements. Showing the difference from A.7 only. 147
A.10 The structure and hyperparameters of the network using the mean-enforced loss, from noisy measurements. Showing the difference from A.7 only. 147
A.11 The structure and hyperparameters of the network used to reconstruct the Kolmogorov flow with softly-constrained loss. The empty cells in column 2 through 4 represent parameters that are the same as in column 1. The very sparse case presented in Section 4.5.3 is trained with the hyperparameters in column 1. 148
A.12 The structure and hyperparameters of the network used to reconstruct the Kolmogorov flow with snapshot-enforced loss. The empty cells in column 2 through 4 represent parameters that are the same as in column 1. The very sparse case presented in Section 4.5.3 is trained with the hyperparameters in column 1. 149
A.13 The structure and hyperparameters of the network used to reconstruct the Kolmogorov flow with mean-enforced loss. The empty cells in column 2 through 4 represent parameters that are the same as in column 1. The very sparse case presented in Section 4.5.3 is trained with the hyperparameters in column 1. . . . 150
A.14 The network structure and hyperparameters used to reconstruct the 2D Kolmogorov flow with extreme events. 152
A.15 The network structure and hyperparameters used to reconstruct the 3D Kolmogorov flow. 153
A.16 The hyperparameters of the PC-DualConvNet used to reconstruct the 3D Kolmogorov flows in Section 6.3. 155

A.17 The hyperparameters of the weight-sharing network used to reconstruct the 3D Kolmogorov flows in Section 6.3. 155
A.18 Number of trainable parameters; the average time to compute one update (including a forward pass, computing the loss, and applying the update); and the average inference time (forward pass only). Timed using a 50-snapshot batch on an NVIDIA RTX8000 GPU. Functions are compiled with `jax.jit`. 156
A.19 The average time to compute loss using a 50-snapshot batch on an NVIDIA RTX8000 GPU. Functions are compiled with `jax.jit`. 157
A.20 The hyperparameters of the PC-DualConvNet used to reconstruct the 3D Kolmogorov flows in Section 6.4. 157
A.21 The hyperparameters of the weight-sharing network used to reconstruct the 3D Kolmogorov flows in Section 6.4. 158

B.1 Comparing the physics losses computed in single- and double-precision floating point format, for the 2D Kolmogorov flow dataset presented in Chapter 4. 161
B.2 The relative error, residual of the momentum and continuity equations of the reconstruction results from Section 6.3, compared with the the same network with an additional divergence-free condition after the last layer. 165

Nomenclature

Acronyms

AE	Standard Autoencoder
CNN	Convolutional Neural Network
MD-AE	Mode-decomposing Autoencoder
ML	Machine Learning
MSE	Mean Squared Error
PIV	Particle image velocimetry
POD	Proper Orthogonal Decomposition
PSD	Power Spectral Density
ROM	Reduced-order modelling
SNR	Signal-to-Noise Ratio
TKE	Turbulent Kinetic Energy

Functions/operators

$\mathscr{L}$	Loss function
$\mathscr{R}$	Residual of a partial differential equation
∇	Nabla operator
Δ	Vector Laplacian operator
$\|\|*\|\|_2^2$	ℓ_2 norm

Matrix/tensors

$\boldsymbol{\Lambda}$	Diagonal matrix of eigenvalues, sorted from largest to smallest
$\boldsymbol{\Phi}$	POD modes
$\boldsymbol{A}$	Time coefficients for POD modes
$\boldsymbol{B}$	Decoder coefficients
$\boldsymbol{C}$	Correlation matrix
$\boldsymbol{D}$	Dataset, including pressure and velocity
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{P}$	Dataset containing only the pressure
$\boldsymbol{P}_{in}$	Pressure input to the network
$\boldsymbol{Q}$	Snapshot matrix

$\boldsymbol{U}$	Dataset containing only the velocity
$\boldsymbol{W}_P$	Weight matrix for the snapshot matrix for POD
$\xi(\boldsymbol{D})$	Measurements of dataset $\boldsymbol{D}$

Non-dimensional group

Re	Reynolds Number
St	Strouhal number

Symbols

Δ	Discrete step size in time or space
Di	Dissipation rate
ϵ	Relative error
$\hat{*}$	Reconstruction of $*$
$\mathcal{N}$	Normal distribution
σ	Standard deviation
$\boldsymbol{\eta}$	Average rate of change
$\boldsymbol{e}_*$	Unit vector in $*$ direction
$\boldsymbol{g}$	Forcing
$\boldsymbol{p}$	Pressure (local, instantaneous)
$\boldsymbol{r}$	Separation between two points
$\boldsymbol{u}$	Velocity (local, instantaneous)
$\boldsymbol{v}$	Vorticity (local, instantaneous)
$\boldsymbol{w}$	Weights of a network
$\boldsymbol{z}$	Latent variables
D_{KL}	The Kullback-Leibler divergence
e	Noise
f	Frequency
k	Wavenumber
S_{LL}	The longitudinal second-order velocity structure function
t	Time
x	Spatial dimension
L	Length
N	Number of grid points in any direction in the domain/matrix

Chapter 1

Introduction

Flows, especially turbulent flows, are present in many aspects of our daily lives, and our ability to understand and model them is crucial in many engineering applications, such as drag reduction for vehicles, weather forecasting, and chemical processes. Even though the governing equations of flows are known, i.e., the Navier-Stokes equations, most types of flows cannot be solved analytically in practice (Holmes et al. 2012*a*). In fact, the Navier-Stokes equations are so difficult to solve that proving the existence and smoothness of the incompressible Navier-Stokes equations is one of the seven Millennium Prize Problems (*The Millennium Prize Problems* n.d.). The study of flows is therefore based on numerical simulations and experiments, both of which generate large amounts of data. Numerous data-driven methods exist to analyse these large datasets, which can be used to either gain a better understanding of the flow physics or to develop models for flow behaviours.

1.1 Review of data-driven methods in fluid mechanics

In this section, we review data-driven methods in fluid mechanics that are commonly used to take us from raw data to the required flow models. This review is divided into two subsections. The compression and feature extraction from complete flow datasets (Section 1.1.1), and the reconstruction of flow fields from incomplete datasets (Section 1.1.2). By 'complete', we mean that data are available at all locations (spatial or temporal) of interest. The content is taken in part from Mo et al. (2024) and Mo & Magri (2025).

1.1.1 Extracting features from complete datasets

Turbulent flows are nonlinear multiscale systems with large numbers of degrees of freedom. High-fidelity simulations of turbulent flows can be performed by solving the governing equations on fine spatio-temporal grids, but the computational cost can be prohibitively high (e.g., Rowley & Dawson 2017). Even when high-fidelity simulations are available, the datasets can be too large to be analysed directly in many engineering applications. Instead, cheaper reduced-order

modelling (ROM) are often applied to approximate the flows with fewer degrees of freedom (Noack et al. 2011, Rowley & Dawson 2017).

Commonly, reduced-order models are constructed via projection-based methods, which is based on the idea of separation of variables to turn a set of partial differential equations into a set of ordinary differential equations (Rowley & Dawson 2017, Holmes et al. 2012*b*). During the process of building a reduced-order model, the equations are projected onto a set of orthogonal basis, obtained via modal decomposition methods, such as proper orthogonal decomposition (POD) (Noack et al. 2011, Rowley & Dawson 2017).

POD is one of the most popular decomposition methods, which dates back to 1970 (Lumley 1970) and enables the computation of an optimal linear subspace based on the energy norm (Taira et al. 2017). Each POD mode is associated with an energy, which ranks the importance of the mode and guides practitioners to decide when to truncate the modes. POD modes can also be interpreted as coherent structures of flows, which are flow patterns that are energy-maximising and spatially orthogonal to each other in the flow (e.g., Kevlahan et al. 1994, Alfonsi & Primavera 2007, Holmes et al. 2012*a*, Rigas et al. 2014). There are other methods of linear decomposition, such as spectral POD (Lumley 1970, Schmidt & Colonius 2020), dynamic mode decomposition (Schmid 2010, Tu et al. 2014), and wavelet analysis (Albukrek et al. 2002), which have been used for dimensionality reduction and discovery of coherent structures. These linear methods are relatively straightforward to implement, but may require large numbers of modes to reduce the approximation error for accurate modelling of nonlinear flows (Alfonsi & Primavera 2007).

On the other hand, nonlinear ROM can achieve a higher compression ratio by using nonlinear mappings instead of increasing the number of projection bases (Touzé et al. 2021). Nonlinear ROM seeks manifolds onto which it performs dimensionality reduction to approximate the dynamics (Magri, L. & Doan, N.A.K 2022, Mazzilli et al. 2022, Racca et al. 2023, Fukami & Taira 2025). These nonlinear models are either built with Nonlinear Normal Modes derived based on the centre manifold theorem (Carini et al. 2015, Touzé et al. 2021), or lately, found by training neural networks (Racca et al. 2023, Kelshaw & Magri 2025).

Machine learning has been increasingly applied for dimensionality reduction of fluids, in particular via nonlinear autoencoders (AEs) (Fukami, Maulik, Ramachandra, Fukagata & Taira 2021, Csala et al. 2022, Eivazi et al. 2022, Magri, L. & Doan, N.A.K 2022, Doan et al. 2023, Fukami & Taira 2023, Fukami et al. 2024). AEs consist of an encoding and a decoding part: the encoding part maps the input (the physical flow field) into a lower-dimensional latent space, whereas the decoding part maps the latent space back to the physical space. The purpose of the autoencoders is to approximate the identity mapping. The dynamics of turbulent systems can then be predicted on the low-dimensional latent space with sequential methods such as reservoir computers (Doan et al. 2021, Racca et al. 2023, Özalp & Magri 2025) and Long short-term memory

networks (Nakamura et al. 2021). AEs have also been used with non-sequential methods such as Neural Ordinary Differential Equations (e.g., Linot & Graham 2022, Varun Shankar et al. 2020) and transformers to predict chaotic systems (e.g., Solera-Rico et al. 2024). The latent space of an AE, which is a nonlinear manifold (Magri, L. & Doan, N.A.K 2022, Kelshaw & Magri 2025), is difficult to interpret due to latent variables being entangled (Eivazi et al. 2022), and their coordinate bases may not be locally orthogonal (Magri, L. & Doan, N.A.K 2022). Efforts have been made to produce interpretable AEs by introducing specific AE architectures (e.g., Murata et al. 2020, Fukami et al. 2020, Eivazi et al. 2022, Kelshaw & Magri 2025). The latent spaces of specific flows have been successfully deciphered by methods such as clustering (Fukami & Taira 2023, 2025), or computing locally linear modes (Magri, L. & Doan, N.A.K 2022, Colanera & Magri 2025, Kelshaw & Magri 2025). However, the general interpretability of latent spaces remains a problem.

1.1.2 Reconstruction from incomplete datasets

In the previous subsection, we reviewed methods for extracting information from complete datasets. However, in many practical applications, data is often incomplete, either due to limited resources or the nature of the measurement process. In this subsection, we review the methods for recovering the flow fields from incomplete, sparse data.

In many flow applications, particularly in experimental settings, estimating a flow field from incomplete data is important. To address this, various data assimilation methods have been developed for flow field reconstruction, employing different strategies. Linear approaches often rely on modal decompositions combined with linear regression (Beneddine et al. 2017, Callaham et al. 2019, Chandramouli et al. 2019) or nonlinear optimisation techniques (Franceschini et al. 2021) to estimate modal coefficients. Although computationally efficient, modal-based methods may not fully fulfil the governing equations and often depend on modes derived from spatially complete data or mean flow data.

To address spatial incompleteness, extensions of modal decomposition, such as gappy POD (Everson & Sirovich 1995) and spectral POD (Nekkanti & Schmidt 2023) have been utilised. These techniques can reconstruct flows from incomplete data (e.g., Venturi & Karniadakis 2004, Gunes et al. 2006, Raben et al. 2012, Nekkanti & Schmidt 2023), although they may struggle when dealing with large spatial or temporal gaps in the data (Venturi & Karniadakis 2004, Gunes et al. 2006, Nekkanti & Schmidt 2023). To incorporate non-linearities and physics-based constraints, adjoint-based optimisation techniques, such as 4DVar (Le Dimet & Talagrand 1986), have been applied. These approaches reconstruct flow fields that fulfil the Navier-Stokes equations (Mons et al. 2016, Symon et al. 2017, Franceschini et al. 2020, Wang & Zaki 2021, Zaki & Wang 2021, Cato et al. 2023). However, adjoint-based methods can be computationally expensive and memory-intensive (Franceschini et al. 2020), and they may become unstable when the optimisation time window exceeds the chaotic time scales of the system (Wang et al. 2014, Huhn & Magri 2020,

Zaki & Wang 2021). Given the existing similarity with many gradient-based data assimilation methods, efforts were made to combine various neural networks with data assimilation methods (Cheng et al. 2023). Karnakov et al. (2023) utilised automatic differentiation to assimilate data directly into a discretised solution of a partial differential equation without needing a differentiable solver. Sequential filtering methods, such as the ensemble Kalman filter (Geir Evensen 2009), offer another strategy for reconstructing flow fields from sensor measurements (Colburn et al. 2011, Mons et al. 2016, Suzuki & Hasegawa 2017, Deng et al. 2018, Zhang et al. 2021). However, the quality of reconstruction using sequential filtering depends on the robustness and accuracy of the adopted reduced-order model (Suzuki & Hasegawa 2017, Nóvoa et al. 2024).

Recently, neural networks, with their ability to handle a large amount of data and approximate nonlinear functions, have become a popular choice for performing flow reconstruction (Brunton & Kutz 2024), which often involves estimating the velocity and/or pressure from sparse measurements of the flow field. Many network architectures have been developed for flow reconstruction when ground-truth flow fields are available during training. Autoencoders have been used to reconstruct various flows, such as bluff body wakes, isotropic turbulence from sparse measurements (Erichson et al. 2020, Dubois et al. 2022). Maulik, Romit et al. (2020) reconstructs the ocean temperature, a bluff body wake and an aerofoil with a mixture density network. Generative adversarial networks have been used to reconstruct rotating turbulence (Buzzicotti et al. 2021, Li et al. 2023) and channel flows (Kim et al. 2021, Yu et al. 2022, Yousif et al. 2023). Other generative networks have also been used to recover turbulence statistics (Sardar et al. 2024). Convolutions are a common feature amongst the networks used for reconstruction and have been used to perform tasks such as recovering high-resolution flow fields from lower-resolution flow fields (Fukami et al. 2019, Fukami, Fukagata & Taira 2021, Güemes et al. 2022, Matsuo et al. 2024), reconstruct flows from sparse measurements (Özbay & Laizet 2022), and obtaining the flow fields from images of geometries of porous media (Zhou et al. 2022).

Fewer works are available when ground truth data (i.e., the full flow field) is not available during the training of the networks. Güemes et al. (2022) reconstructed turbulent flows from noisy point measurements of the flow, in which the points are randomly placed and moving, using a generative adversarial network. Their network requires the point measurements to change locations each snapshot, making it suitable for reconstructing flows from the results of particle image velocimetry (PIV), which generates a large amount of moving point measurements.

When the ground truth is unavailable and measurements are sparse, incorporating prior knowledge into machine learning allows the reconstruction of flow fields from limited measurements. Physics-informed neural networks (PINNs) can infer the unknown variables (e.g., velocity) at unseen locations and have been employed to reconstruct a range of different flows (Raissi et al. 2019, 2020, Arzani et al. 2021, Zhang & Zhao 2021, Eivazi et al. 2024). Convolutional neural networks (CNNs) allow an image-based approach to flow reconstruction, which is suitable for reconstruct-

ing high-dimensional flows when the temporal dimension is not considered/unknown. By incorporating physical knowledge such as the Navier-Stokes equations, CNNs have been employed to reconstruct steady flows (Gao et al. 2021) and turbulent flows from limited measurements (Subramaniam et al. 2020, Kelshaw & Magri 2024, Page 2025).

Depending on the level of understanding that we have of the problem at hand, there are different ways to incorporate physics into training. Hard-coded constraints such as using *sine* activation function or Fourier modes are used to ensure periodicity in the output (Raynaud et al. 2022, Wong et al. 2022, Ozan & Magri 2023). However, the most common way to enforce physics is to introduce the governing equations as a penalty term in the loss function to be minimised (e.g., Raissi et al. 2019, 2020, Gao et al. 2021, Kelshaw et al. 2022, Fukami et al. 2023). Example governing equations which have been used as the penalty term include the wave equation (Ozan & Magri 2023), incompressible Navier-Stokes equations (Raissi et al. 2019, Page 2025), and modified Navier-Stokes or other conservation equations for other physical systems (Sitte & Doan 2022, Sliwinski & Rigas 2023).

When measurements are taken from experiments, they are often noisy. Different network structures have been proposed for reconstructing flows from noisy measurements, such as placing restrictions on the shape of the solution (Raynaud et al. 2022) or using probabilistic networks (Maulik, Romit et al. 2020). Some information on the flow fields can also be recovered from noisy partial observations by reconstructing the mean instead of the instantaneous flow fields (Sun & Wang 2020, Sliwinski & Rigas 2023). Shokar et al. (2024) in particular, used forced noise and an ensemble network to study the long-term statistics of beta-plane turbulence. Reconstruction of instantaneous turbulent flows from noisy partial observations has been performed with CNN-based networks in (Kim et al. 2021, Li et al. 2023, Yousif et al. 2023) as part of a robustness study. However, these studies require knowledge of the reference data (i.e., the full flow field) during training.

The reconstruction of three-dimensional (3D) turbulent flows poses an even greater challenge and has been attempted by some prior works. Chandramouli et al. (2019) reconstructs 3D flows from a library of two-dimensional (2D) POD modes computed with 2D measurements, based on the assumption that the flow is homogeneous in the third dimension. A similar approach has been taken by Chatzimanolakis et al. (2024), who trained a reinforcement learning algorithm on 2D flows and applied it to 3D flows. Ensemble-variational methods have been used to reconstruct 3D unsteady channel flows from instantaneous sparse measurements and statistical observations (e.g., Mons et al. 2021, Wang & Zaki 2021, Zaki & Wang 2021), and to reconstruct mean flows from particle image velocimetry measurements (e.g., He et al. 2025), but these methods require running a large number of 3D simulations, and are therefore expensive. Neural networks have also been used to reconstruct 3D flows. PINNs have been used to reconstruct various 3D flows from different measurement schemes which produce measurements that move in time and space

(e.g., Zhang & Zhao 2021, Clark Di Leoni et al. 2023, Zhu et al. 2024). When initial conditions or the time coordinates are not available, CNNs have been used to reconstruct flows from snapshots of flows (e.g., Özbay & Laizet 2023, Xuan & Shen 2023, Yousif et al. 2023). Instantaneous velocities within wall-bound flows have been reconstructed from wall quantities using CNNs and general adversarial networks (Guastoni et al. 2021, Cuéllar et al. 2024). However, these methods require the full flow field during training. In other prior works, 3D flows are left for future works (Buzzicotti et al. 2021, Page 2025).

1.2 Objective

Previous works from the data-driven fluid mechanics community show that neural networks are now widely applied in all areas of the dataset-to-result pipeline, from data reconstruction to model order reduction. However, we have identified some challenges that still need to be addressed. Particularly, the interpretability of the neural network when used for dimensionality reduction, and the reconstruction of both 2 and 3D flows when the full flow field is not available during training. This thesis addresses these challenges by using machine learning to assist/accelerate the entire data-to-result pipeline, from obtaining data to analysing the data.

We have identified the following milestones within our objective, which we will address in this thesis.

- Develop a method for interpreting and ranking/selecting latent variables of autoencoders, which have been extensively used for nonlinear dimensionality reduction. Test the method on an experimental turbulent wake dataset.
- Develop a method to reconstruct chaotic flows from sparse, noisy measurements, when the ground truth data is not available during training. Test the method on both 2D and 3D synthetic turbulent datasets.
- Develop and test a method to reconstruct 3D synthetic turbulent flows from realistic sensor placements.

1.3 Overview of this thesis

The chapters in this thesis are structured so that each chapter addresses part or one of our milestones. The summaries of each chapter are as follows.

Chapter 2: This chapter is an overview of the data-generation and data-processing methods used throughout the thesis. We describe the simulation software used to generate our datasets, including unsteady wakes of bluff bodies and turbulent Kolmogorov flows. We also define the quantities of interest that we will compute when analysing our results, such as matrix norms and loss measurements.

Chapter 3: In this chapter, we use AEs to perform dimensionality reduction of an unsteady cylinder wake and a wind-tunnel bluff body turbulent wake. We introduce the decoder decomposition, a method for interpreting and selecting latent variables from autoencoders. The decoder decomposition links the latent variables to the POD modes of the flow, which are the coherent structures of the flow. We show that the physical interpretation of the latent variables heavily depends on the user-selected latent dimension. We also demonstrate how to use the decoder decomposition to select latent variables that are physically meaningful.

Chapter 4: Using a physics-constrained convolutional neural network, we reconstruct a 2D unsteady laminar bluff body wake and a 2D turbulent Kolmogorov flow from sparse measurements. No high-resolution data is needed during training. We also investigate the effect of different loss functions on the reconstruction results. We find the snapshot-constrained loss is suitable when the measurements are noise-free. However, measurements are usually noisy when taken from real-life situations. When measurements are corrupted with noise, we propose the mean-enforced loss and find that our loss is more robust at low signal-to-noise ratios.

Chapter 5: In this chapter, we apply the PC-DualConvNet, introduced in Chapter 4, to reconstruct two more realistic cases from sparse measurements — a 2D turbulent Kolmogorov flow exhibiting extreme events, and a set of 3D turbulent Kolmogorov flows. From sparse, noisy measurements, we reconstruct the 2D flow and obtain the distinct spatial structures and the correct timing of the extreme events. We also show that the PC-DualConvNet, originally designed for reconstructing 2D flows, can be extended to reconstruct 3D flows with minimum effort.

Chapter 6: This chapter focuses on reconstructing turbulent 3D Kolmogorov flows from a more realistic measurement setup. The measurements consist of planes of in-flow velocity measurements, either parallel or perpendicular to each other, and an additional plane of pressure at a boundary. We design the weight-sharing network, which utilises the homogeneous direction in the flow to improve the reconstruction in inaccessible regions of the flow. Using the weight-sharing network, we reconstruct the 3D Kolmogorov flow from the plane-sensor setup, with measurements corrupted with added white noise.

Chapter 7: In this chapter, we summarise the findings of this thesis and discuss future works.

Chapter 2

Datasets and methods for processing data

This chapter introduces the datasets and the terminologies used in the rest of the thesis. The content in this section is adapted from Mo et al. (2024) and Mo & Magri (2025).

2.1 Datasets

This section describes experimental setup used to obtain the wind-tunnel dataset (Section 2.1.1), and the programs and methods to generate the synthetic datasets (Section 2.1.2).

2.1.1 Experimental data

A wind-tunnel dataset of a turbulent bluff body wake[1] at Reynolds number $Re \approx 200{,}000$ (Rowan D. Brackston 2017) is used in Chapter 3.

Figure 2.1 shows the experimental setup. The axisymmetric body has a diameter of $D = 196.5$ mm and a length-to-diameter ratio $L/D = 6.5$. Pressure measurements at the base of the body are collected via 64 static pressure taps placed on a regular polar grid, with eight grid points in the radial direction and eight grid points in the azimuthal direction (Figure 2.1). An analysis of the data is provided in Chapter 3.

2.1.2 Generating synthetic data

Two canonical flows are used as examples in this thesis: unsteady laminar wakes of cylinders and Kolmogorov flows. All synthetic datasets are generated by solving the dimensionless Navier-Stokes Equations

$$\begin{cases} \nabla \cdot \boldsymbol{u} = \mathscr{R}_d(\boldsymbol{u}) \\ \frac{\partial \boldsymbol{u}}{\partial t^*} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re} \triangle \boldsymbol{u} - \boldsymbol{g} = \mathscr{R}_m(\boldsymbol{u}, p), \end{cases} \tag{2.1}$$

where $\boldsymbol{u}(\boldsymbol{x}^*, t^*) \in \mathbb{R}^{N_u}$ and $p(\boldsymbol{x}^*, t^*) \in \mathbb{R}$ are the velocity and pressure at location $\boldsymbol{x}^*$ and time t^*, N_u is the number of velocity components, and $\boldsymbol{g}$ is a forcing term applied to the flow. The

[1] We thank G. Rigas and J. Morrison for providing the experimental bluff body wake dataset.

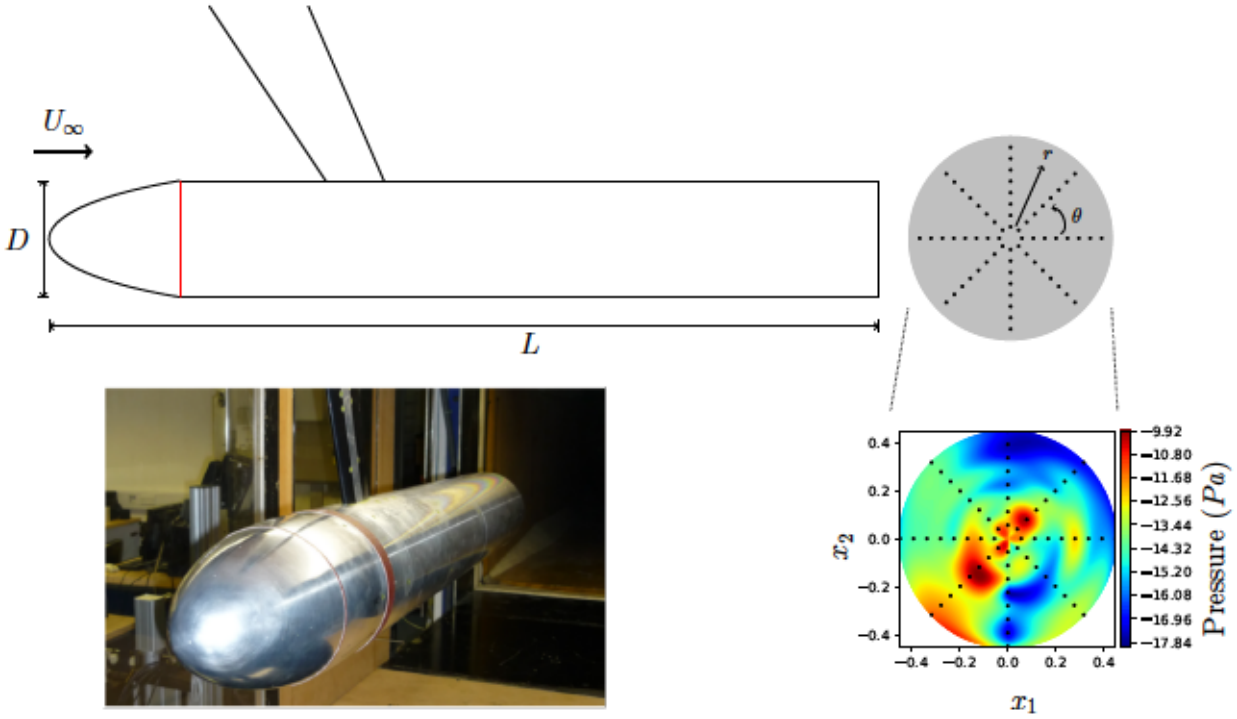


Figure 2.1: Experimental set-up, reproduced from (Rigas 2021). The dimensions x_1 and x_2 are the measured location nondimensionalized by the diameter of the bluff body. The black dots mark the location of the pressure sensors.

Reynolds number Re is the inverse of the kinematic viscosity.

For our convenience, we cast the Navier-Stokes equations into residual form. When solved numerically, the simulated flow has near zero residuals, $[\mathscr{R}_d, \mathscr{R}_m]^T \approx \mathbf{0}$.

2D unsteady laminar wakes

We use 2D unsteady laminar wakes of bluff bodies as a benchmark case to demonstrate nonlinear decomposition and flow reconstruction.

For the 2D unsteady wakes, $N_u = 2$ and the velocity and length are non-dimensionalised by the inlet velocity u_∞ and the characteristic length of the bluff body. Two bluff body wakes at $Re = 100$ are considered in this thesis — circle and equilateral triangle. The characteristic length is the diameter of the circle in Chapter 3 or the length of one side of the equilateral triangle in Chapter 4. The unsteady wakes are simulated with direct numerical simulation using Xcompact3D (Bartholomew et al. 2022). The computational domain has size L_1 and L_2 in x_1 (streamwise) and x_2 (wall-normal) directions, with 513 and 129 uniformly spaced grid points in each direction, respectively. Zero gradients are applied at $x_1^* = L_1$, and slip walls are applied at $x_2^* = 0$ and $x_2^* = L_2$; The time step is chosen to be $\Delta t^* = 0.0002$ to satisfy the Courant-Friedrichs-Lewy condition. Equation (2.1) is solved using the 3$^{\text{rd}}$-order Adams-Bashforth scheme in time and the 6$^{\text{th}}$-order compact scheme in space (Laizet & Lamballais 2009).

Table 2.1: Summary of the final datasets for the two bluff body cases.

Bluff body	Re	Δt	Downstream area bounded by (x_1^{*1}, x_1^{*2})	Grid points (N_1, N_2)	N_t
Cylinder	100	0.125	3.51, 8.18	200, 129	720
Equilateral Triangle	100	0.125	3.00, 8.86	250, 129	600

Periodic boundary conditions are applied to the x_3 (spanwise) direction to simulate an infinitely long cylinder. The 2D data is taken from an x_1-x_2 plane of the resulting simulations. The third velocity components are zero everywhere on the plane at all snapshots. The transient period from $t^* = 0$ to 100 is discarded to ensure that the final dataset consists of only periodic vortex shedding (Ganga Prasath et al. 2014) at a statistically stationary regime. From here on, $t = 0$ denotes the time of the first snapshot of the final dataset, i.e. $t = t^* - 100$. A snapshot of the flow is saved every 625 steps, resulting in a time step $\Delta t = 0.125$ for the final dataset, i.e. over 40 snapshots per vortex shedding period.

For computational efficiency, we consider only the area capturing the near wake and the parallel vortex shedding. For constructing the final dataset, a snapshot of the flows contains the grid points on the plane at $x_3^* = 0$ bounded by $x_1^* = x_1^{*1}$, $x_1^* = x_1^{*2}$, $x_2^* = 0$ and $x_2^* = L_2$. The details for the bluff body wakes used in Chapter 3 and 4 are summarized in Table 2.1. For simplicity, we define the coordinates of the dataset to be $\boldsymbol{x} = [x_1^* - x_1^{*1}, x_2^*]^T$. The snapshots are organized into snapshot matrices $\boldsymbol{U} \in \mathbb{R}^{N_t \times N_1 \times N_2 \times N_u}$ and $\boldsymbol{P} \in \mathbb{R}^{N_t \times N_1 \times N_2 \times 1}$ for velocity and pressure, where N_t is the number of snapshots, N_1 is the number of grid points in the x_1 direction and N_2 is the number of grid points in the wall-normal direction.

Two- and three-dimensional Kolmogorov flows

The Kolmogorov flow is a turbulent flow developed to study the transition to complex turbulence (Fylladitakis 2018). Though originally in 2D, the Kolmogorov flow is now studied in both 2D and 3D (Borue & Orszag 1996, Shebalin & Woodruff 1997), and is used as test cases in the development of many ML-based methods (e.g., Kelshaw et al. 2022, Page 2025). We use KolSol (Kelshaw, Daniel 2023), a pseudo-spectral solver, to generate Kolmogorov flows by solving (2.1) with the forcing term $\boldsymbol{g} = \boldsymbol{e}_1 \sin kx_2$, with k being the forcing wavenumber, in a periodic box with length 2π on all sides. The unit vector $\boldsymbol{e}_1$ is $[1,0]^T$ for 2D flows and $[1,0,0]^T$ for 3D flows. A user-selected random seed is used to initialise the solver.

Following the same style as Section 2.1.2, we set the end of the transient period to be $t = 0$, and $\boldsymbol{x} = \boldsymbol{x}^*$. The Kolmogorov flow is simulated in the frequency domain, then interpolated into the physical domain on an N^{N_u} grid, where $N_u = 2$ for 2D flows and 3 for 3D flows. Each Kolmogorov dataset, $\boldsymbol{D} = [\boldsymbol{U}^T, \boldsymbol{P}^T]^T \in \mathbb{R}^{N_t \times N^{N_u} \times (N_u+1)}$, contains snapshots of the full state with velocity and

pressure.

The datasets used in different chapters are generated with different parameters; the detailed parameters and characteristics of the datasets will be discussed in their respective chapters.

2.2 Methods for processing data

This section describes the methods used to extract information from data.

2.2.1 Proper Orthogonal Decomposition

Let $\boldsymbol{Q} \in \mathbb{R}^{N \times N_t}$ be a generic dataset of a measured fluctuating quantity from a flow field such that each snapshot is a column, where N is the product of the number of grid points, the number of variables, and N_t is the number of snapshots. The i-th row, $\boldsymbol{Q}_{i,:}$, is the time series of the measured quantity at grid point i. The t-th column, $\boldsymbol{Q}_{:,t}$, is the snapshot of the measured quantities at the discrete time step t. The covariance matrix $\boldsymbol{C} \in \mathbb{R}^{N \times N}$ is

$$\boldsymbol{C} = \frac{1}{N_t - 1} \boldsymbol{W}_p \boldsymbol{Q} \boldsymbol{Q}^T \boldsymbol{W}_p^T = \boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{\Phi}^T, \tag{2.2}$$

where $\boldsymbol{W}_p$ is the POD weight matrix given to each element of $\boldsymbol{Q}$, $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues and $\boldsymbol{\Phi} \in \mathbb{R}^{N \times N}$ is the matrix of the eigenvectors. Unless specified, we will compute the unweighted POD, meaning that $\boldsymbol{W}_p = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. The i-th column of the matrix of eigenvalues, $\boldsymbol{\Phi}_{:,i}$, is the POD mode i, which represents the i-th principal axis of the space occupied by the observed data ranked by the eigenvalues, i.e., the energy of the mode (Taira et al. 2017, Weiss 2019-06-17/2019-06-21). The matrix of temporal coefficients $\boldsymbol{A}$ is obtained by projecting the data matrix onto the POD modes

$$\boldsymbol{A} = \boldsymbol{Q}^T \boldsymbol{\Phi}, \tag{2.3}$$

which are the temporal coordinates of the data snapshots along the principal axes (Weiss 2019-06-17/2019-06-21). The i-th row of $\boldsymbol{A}$, denoted $\boldsymbol{A}_{i,:}$, contains the time series of the time coefficient of the i-th POD mode. The time coefficient for the mode i at time step t is $A_{i,t}$. By setting $N_m < N$ as the truncated number of modes for reconstructing the flow, we approximate the flow field in the subspace spanned by the first N_m POD modes

$$\hat{\boldsymbol{Q}} = \sum_{i=1}^{N_m} \boldsymbol{\Phi}_{:,i} \boldsymbol{A}_{i,:}^T. \tag{2.4}$$

If $N_m = N$, then $\hat{\boldsymbol{Q}} = \boldsymbol{Q}$, i.e., no approximation is made, and only a linear change of coordinate system is performed.

2.2.2 Other quantities of interest

This section describes other quantities of interest that are used in the thesis. The definitions of these quantities may differ from other literature, and are defined here for consistency.

Norms of matrix

We define here the ℓ_2 norm of matrices following the machine learning convention, which may differ from the definitions in the Mathematics literature. The ℓ_2 norm of a matrix $\boldsymbol{Q}$ is defined as the sum of the squares of all elements in the matrix, i.e.,

$$\|\boldsymbol{Q}\|_2^2 = \sum_{i=1}^{N}\sum_{t=1}^{N_t} Q_{i,t}^2. \tag{2.5}$$

Mean squared error

We follow the machine learning convention and define Mean Squared Error (MSE) between the matrix $\boldsymbol{Q}$ and the reconstructed $\hat{\boldsymbol{Q}}$ as

$$MSE(\hat{\boldsymbol{Q}}, \boldsymbol{Q}) = \frac{1}{N \times N_t}\|\hat{\boldsymbol{Q}} - \boldsymbol{Q}\|_2^2. \tag{2.6}$$

Relative error

The relative ℓ_2 error between the matrix $\boldsymbol{Q}$ and the reconstructed $\hat{\boldsymbol{Q}}$ is given by

$$\epsilon(\%) = 100\sqrt{\frac{\|\hat{\boldsymbol{Q}} - \boldsymbol{Q}\|_2^2}{\|\boldsymbol{Q}\|_2^2}}. \tag{2.7}$$

Dissipation

Dissipation is the rate of loss of kinetic energy in a flow (Pope 2000). The global dissipation rate Di is defined as (Farazmand & Sapsis 2017)

$$Di(\boldsymbol{u}) = \frac{\nu}{|\Omega|}\int_{\Omega} |\nabla \boldsymbol{u}|^2 dx, \tag{2.8}$$

where ν is the kinematic viscosity, $|\Omega|$ is the area (or volume in the case of 3D flows) of the domain, and $\nabla \boldsymbol{u}$ is the gradient of the velocity field. In the cases of data generated using the procedures described in Section 2.1.2, the kinematic viscosity ν is the inverse of the Reynolds Number $\frac{1}{Re}$, and the gradients are taken using finite differences.

Signal-to-noise ratio

The signal-to-noise ratio (SNR) is a measure of the ratio of the power of the signal to the power of the noise, defined as

$$\text{SNR} = 10\log(\sigma^2/\sigma_e^2), \tag{2.9}$$

where σ is the standard deviation of the data and σ_e is the standard deviation of the noise. The noise is taken from a Gaussian distribution $e \sim \mathcal{N}(0, \sigma_e)$.

Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence, D_{KL}, a measure of how much a probability distribution, $\mathscr{P}_2$, differs from the true probability distribution, $\mathscr{P}_1$. The discrete approximation of the KL divergence is

$$D_{\mathrm{KL}}(\mathscr{P}_1 \| \mathscr{P}_2) = \sum_{x \in X} \Delta x \mathscr{P}_1(x) \log \frac{\mathscr{P}_1(x)}{\mathscr{P}_2(x)}, \tag{2.10}$$

where x is the independent variables, X denotes the set of discrete sampling points, and $\mathscr{P}_1$ and $\mathscr{P}_2$ denote the probability density function for the true and the approximated probability distribution. D_{KL} is 0 if the two distributions are identical.

Second-order velocity structure function

The second order velocity structure function is the covariance of the velocity between two points separated by a separation r (Pope 2000). The longitudinal second-order velocity structure fuction is defined as

$$S_{LL}(r) = \left\langle \left(\left[\boldsymbol{u}'(\boldsymbol{x}+\boldsymbol{r}) - \boldsymbol{u}'(\boldsymbol{x}) \right] \cdot \boldsymbol{e}_r \right)^2 \right\rangle, \tag{2.11}$$

where <> denotes the spatial and temporal average, $\boldsymbol{u}'$ is the fluctuating velocity, $\boldsymbol{e}_r$ is the unit vector in the $\boldsymbol{r}$ direction and r is the magnitude of the separation vector $\boldsymbol{r}$.

Chapter 3

Extracting nonlinear modes via decoder decomposition

Whether for the study of turbulent flows out of academic interest or building computationally viable models of turbulence for practical purposes, low-dimensional models of turbulent flows are often employed. These models are often built from a compressed version of a large amount of existing data. Recently, autoencoders (AEs) have been used to perform nonlinear dimensionality reduction. AEs can effectively compress high-dimensional data into low-dimensional latent spaces, but are difficult to interpret. In this chapter, we introduce the decoder decomposition, a method to connect the latent space of AEs to the coherent structures of flows. We apply the decoder decomposition to the latent space of a wind-tunnel experimental dataset and show that the interpretability of the latent space is significantly impacted by the user-defined size of the latent space. The decoder decomposition takes a step towards building interpretable models with nonlinear autoencoding, potentially leading to more trustworthy ML-assisted data analysis.

This chapter is adapted from Mo et al. (2024), published in *Data-Centric Engineering*. The relevant codes for this chapter are available at `https://github.com/MagriLab/MD-CNN-AE`, and the relevant synthetic data is available at `https://doi.org/10.5281/zenodo.15623774`.

3.1 Introduction

Neural networks, particularly AEs, have been increasingly used for nonlinear model reduction of data. It has successfully been used to compress data and forecast flows when used in conjunction with another time-marching method. However, interpretability remains a problem when using it for physical analysis.

The mode-decomposing autoencoder (MD-AE) was developed by Murata et al. (2020) to improve the interpretability of nonlinear decomposed fields. MD-AEs assign a decoder to each variable of the latent space vector (latent variable), and then superpose the single-decoder outputs to

generate the MD-AE output. MD-AEs improve the visualisation of flow decomposition because the effect of each latent variable is isolated in the decoding part, unlike standard AEs, in which one decoder contains the effect of all latent variables. However, an AE outperforms an MD-AE in terms of reconstruction error with the same latent space dimension (Eivazi et al. 2022). This is because the design of MD-AEs does not capture the nonlinear coupling of the decoded latent variables. For highly nonlinear flows, the difference between the reconstruction errors may be significant (Csala et al. 2022). Other autoencoder architectures, such as the hierarchical autoencoder (Fukami et al. 2020) and the β-variational autoencoder (Eivazi et al. 2022, Solera-Rico et al. 2024), which has been proposed to disentangle the latent variables to improve the interpretability, but with a larger reconstruction error as compared to an AE. The decomposed fields, nonetheless, may still be influenced by more than one latent variable, so the exact effect of each latent variable on the output is difficult to isolate.

A comparison between mode decomposition methods, including MD-AE and POD, can be found in the work by Csala et al. (2022). Efforts have been made to obtain interpretable autoencoder latent space for specific problems (Fukami & Taira 2023, Fukami et al. 2024), but general interpretability of autoencoder latent space is an open problem (Vinuesa & Sirmacek 2021, Magri, L. & Doan, N.A.K 2022). The works of Csala et al. (2022) and Eivazi et al. (2022) show that methods which have stronger nonlinear interaction between latent variables produce more accurate reconstruction, albeit at the expense of interpretability.

The objective of this chapter is to improve the interpretability of the latent space of AEs. For this purpose, we propose the *decoder decomposition*, which is a post-processing method for ranking and selecting the latent space of nonlinear autoencoders. The decoder decomposition is based on POD modes and the decoders' sensitivities computed as gradients. In this analysis, to interpret the latent space is to find the statistical correlations between the POD modes of the data and the latent space of the AEs, which differs from physical interpretation. Specifically, the objectives are (i) propose the decoder decomposition, which disentangles the contribution of latent variables in the decoded field; (ii) analyse and verify the decoder decomposition of two commonly used autoencoders (the standard autoencoder and the mode-decomposing autoencoder) with synthetic data from numerical simulation of the unsteady two-dimensional wake past a cylinder; and (iii) apply the decoder decomposition to gain physical insight into a realistic flow (three-dimensional turbulent flow from wind-tunnel experiments) and isolate the latent variables of physical interest with a filtering strategy. The three-dimensional turbulent flow is the high-speed wake past a bluff body.

This chapter is organised as follows: First, we provide details of our datasets in Section 3.2. Next, we introduce the autoencoder architectures in Section 3.3 and propose the decoder decomposition in Section 3.4. We apply the decoder decomposition to decompose the unsteady

laminar cylinder wake and investigate the impact of the dimension of the latent space on the interpretability of autoencoders (AEs) in Section 3.5. We decompose the wind-tunnel wake in Section 3.6 and demonstrate how to rank and select latent variables to filter for coherent structures in the output of the AEs. We present our conclusions in Section 3.7.

3.2 Preprocessing the datasets

Two datasets are considered in this chapter. First, an unsteady laminar wake behind a cylinder, which is the benchmark case whose dynamics are well-known. Second, a wind-tunnel dataset of a three-dimensional turbulent bluff body wake (Rowan D. Brackston 2017). Both datasets are characterized by POD analysis.

3.2.1 Unsteady laminar wake of a two-dimensional cylinder

We generate an unsteady laminar wake of a 2D circular cylinder at Reynolds number $Re = 100$ using the steps detailed in 2.1.2. The unsteady wake is generated with a domain size $L_1 = 12$ and $L_2 = 5$. The final dataset contains 720 snapshots with time interval $\Delta t = 0.125$ between

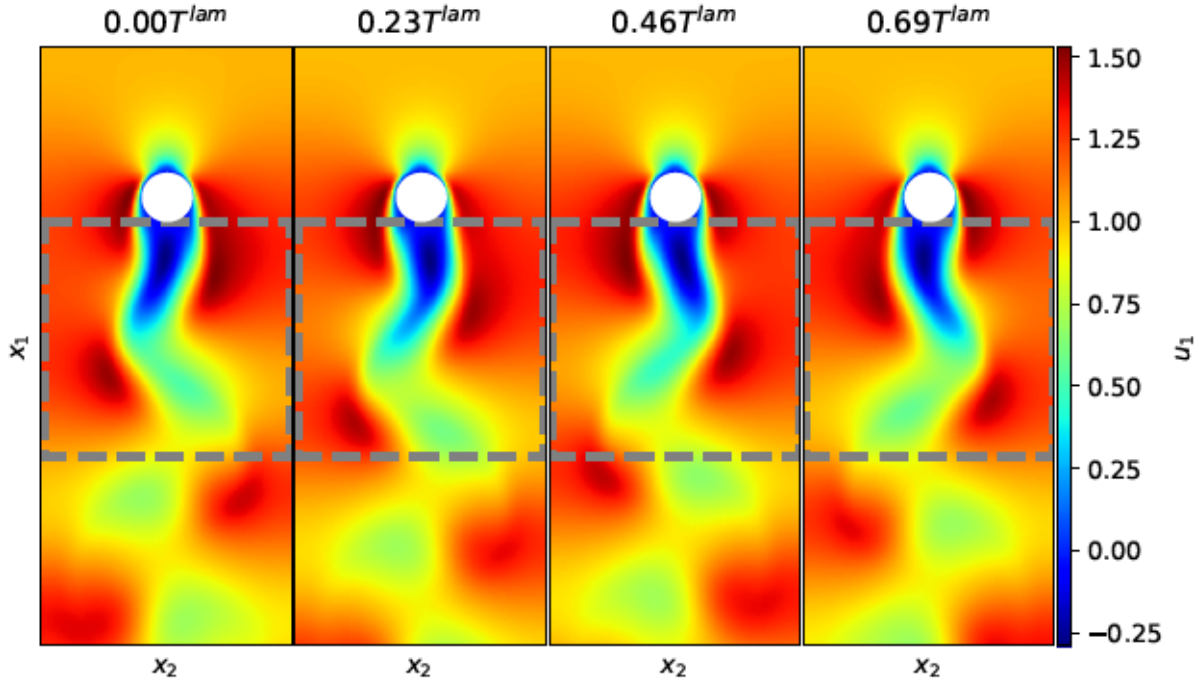


Figure 3.1: Snapshots of the streamwise velocity of the laminar wake dataset at different times within the same period. The area bounded by the grey box is used for training.

snapshots. We consider a 200-by-129 grid behind the cylinder (grey box in Figure 3.1), which captures the near wake and the vortex shedding (M. M. Zdravkovich 1997). The dataset used by the AEs, $\boldsymbol{U} \in \mathbb{R}^{N_t \times N_1 \times N_2}$, consists of the fluctuating streamwise velocity, where N_t is the number

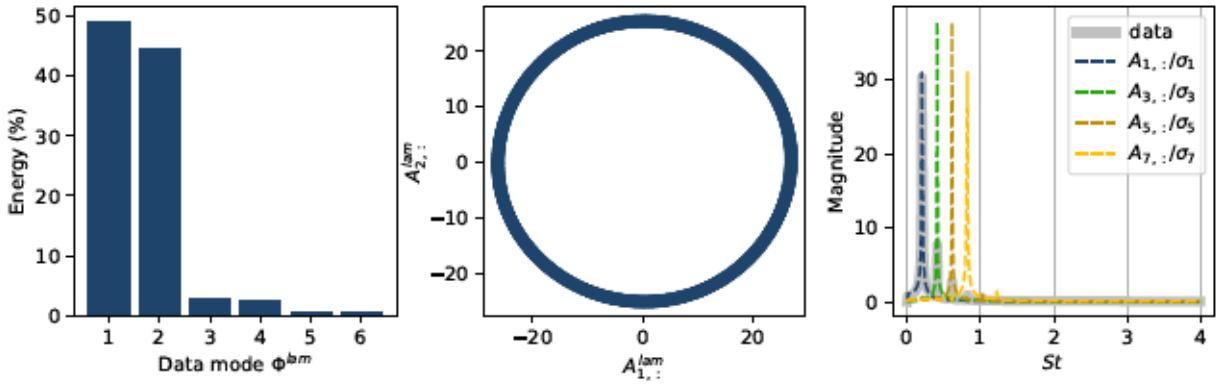


Figure 3.2: POD of the laminar wake $\boldsymbol{U}$. Left: the percentage energy contained in the first six POD modes of the unsteady wake $\boldsymbol{\Phi}^{lam}$ (data mode). Data modes 1&2, 3&4 and 5&6 contain similar flow energy and oscillate at the same frequency but out of phase. Centre: phase plot of the first two data time coefficients. Right: the frequency spectrum of the data and the data time coefficients 1, 3, 5 and 7, normalized by the standard deviation. The data contains the vortex shedding frequency and its harmonics. (Since each pair has the same frequency spectrum, only the odd data modes are shown here).

of snapshots, $N_1 = 200$ and $N_1 = 129$ are the number of grid points in the streamwise and wall-normal directions, respectively.

We perform POD on the dataset $\boldsymbol{U}$ to obtain the POD modes $\boldsymbol{\Phi}^{lam}$ and the matrix of time coefficients $\boldsymbol{A}^{lam}$, referred to as the "data modes" and "data time coefficients", respectively, where *lam* stands for "laminar". Examination of the eigenvalues shows that the data modes are out-of-phase pairs, the two modes in a pair contain a similar amount of flow energy (Figure 3.2, left panel) and their time coefficients are 90° out-of-phase (Figure 3.2, centre panel). Figure 3.3 shows the first six data modes of the laminar dataset. The first two data modes represent the vortex shedding, at the vortex shedding frequency, and the higher modes oscillate at the harmonic frequencies (Loiseau et al. 2020). The magnitude of the fast Fourier transform in time, summed over the grid points, is used to measure the overall frequency content of the dataset, shown in Figure 3.2 (right panel). The frequency content of the flow is described in terms of the Strouhal number, $St = fD/U_\infty$, where f is the frequency measured in Hz. The dominant frequency is the vortex shedding frequency at $St = 0.23$, and the dataset also contains higher harmonic frequencies. Each pair of data modes oscillates at a single, distinct frequency (Figure 3.2, right panel). The first pair, comprised of data modes 1 and 2, oscillate at the vortex shedding frequency, while the subsequent pairs, composed of data modes 3 and 4, oscillate at the first harmonic, and so forth. The concentration of the flow energy in the leading data modes and the distinct frequencies of the data time coefficients make the unsteady wake behind a cylinder suitable for dimensionality reduction and interpretation with POD.

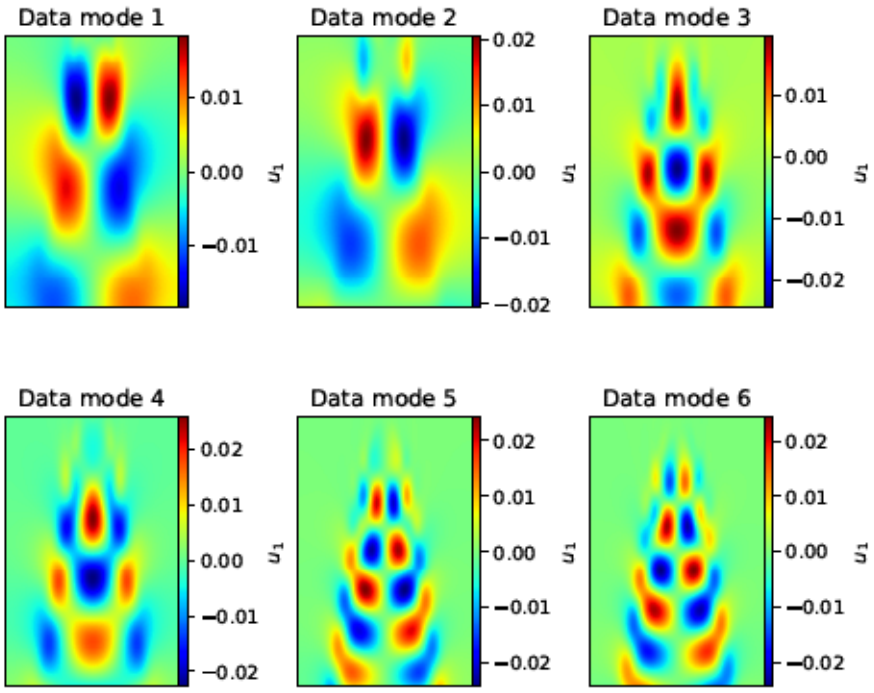


Figure 3.3: The first six POD modes (data modes) of the unsteady wake behind a cylinder dataset $\mathbf{\Phi}^{lam}_{:,1},\ldots,\mathbf{\Phi}^{lam}_{:,6}$.

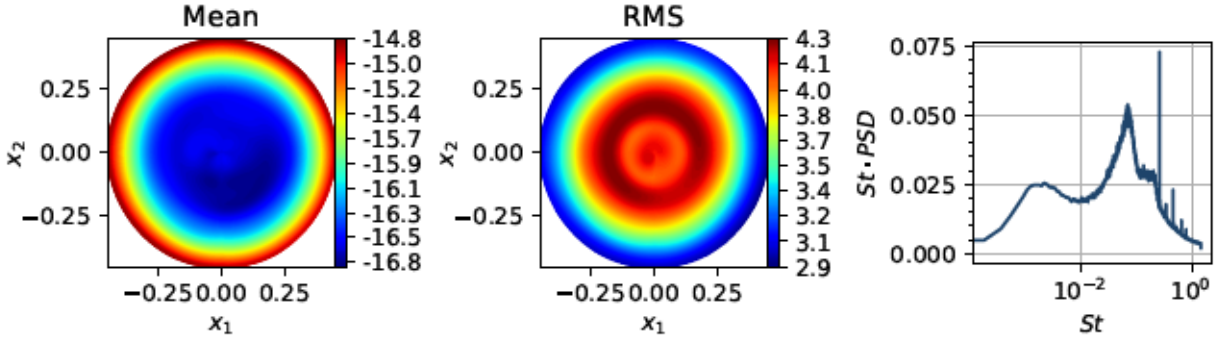


Figure 3.4: The wind-tunnel pressure dataset $\boldsymbol{P}$. Left: Mean pressure. Centre: RMS pressure. Right: The premultiplied PSD ($St\cdot$ PSD) of the wind-tunnel dataset, with peaks at $St \approx 0.002, 0.06$ and $St \approx 0.2$ and its harmonics. The peaks correspond to the three-dimensional rotation of the wake, the pulsation of the vortex core, and the vortex shedding and its harmonics, respectively.

3.2.2 Turbulent wake of a bluff body from wind-tunnel experiments

The turbulent wake of an axisymmetric body from wind tunnel experiments (Rowan D. Brackston 2017, Rigas et al. 2014) is employed in the second part of this chapter. A description of the experimental setup can be found in Section 2.1.1. We provide here an overview of the dataset. For a more in-depth analysis of the dataset and the experimental setup, the readers are referred to Rigas et al. (2014).

The mean and root-mean-squared (RMS) pressures are both axisymmetric (Figure 3.4). The power spectral density (PSD) of the pressure data is shown in Figure 3.4 (right), premultiplied by the Strouhal number to improve the visualisation (Rigas et al. 2014). The peaks at $St \approx 0.002$, 0.06 and 0.2 are associated with the 3D rotation of the symmetry plane, pulsation of the vortex core and vortex shedding, respectively.

The weighted POD (each data point is weighted by the area of the element of the polar grid) is applied to the dataset (Rigas et al. 2014, Rowan D. Brackston 2017). The resulting POD modes $\boldsymbol{\Phi}^{exp}$ and time coefficients $\boldsymbol{A}^{exp}$ are the data modes and data time coefficients, respectively, where *exp* stands for "experimental". Figure 3.5 shows the energy contained in each data mode of the wind-tunnel dataset and the cumulative energy of the modes. The flow energy is spread over more modes than the laminar case, and 21 data modes are needed to reconstruct the dataset to recover 95% of the energy. The modes and the premultiplied PSD of their time coefficients are shown in Figure 3.6. Data modes 1 and 2 are antisymmetric and have frequency peaks at $St \approx 0.002, 0.1$ and 0.2. As a pair, data modes 1 and 2 represent vortex shedding and the 'very-low-frequency' 3D rotation of the symmetry plane (Rigas et al. 2014). Data mode 3 has a frequency peak at $St \approx 0.06$ and is associated with the pulsation of the vortex core (Rigas et al. 2014, Rowan D. Brackston 2017). The data modes 4 and 5 contain the sub-harmonic frequencies. Data modes 1 to 5 contain the coherent structures of the dataset. We base our analysis and latent

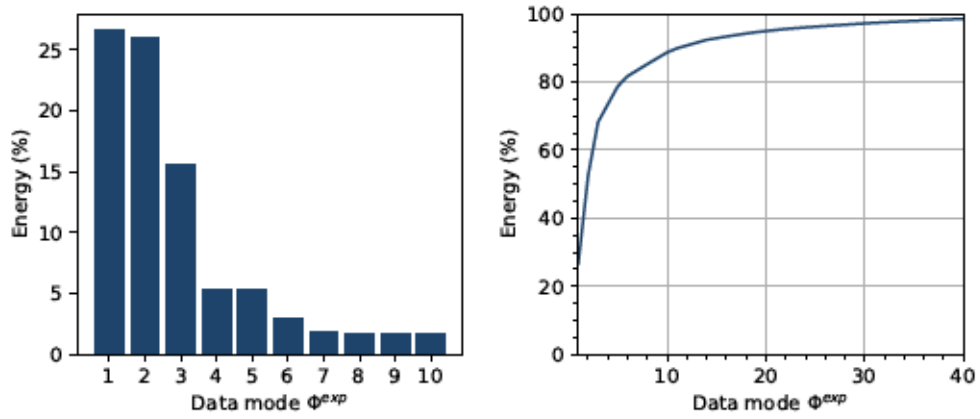


Figure 3.5: POD of the wind-tunnel dataset $\boldsymbol{P}$. Left: percentage energy of the first 10 data modes. Right: cumulative percentage energy of POD modes. The reconstruction of the pressure dataset to 95% energy needs 21 data modes.

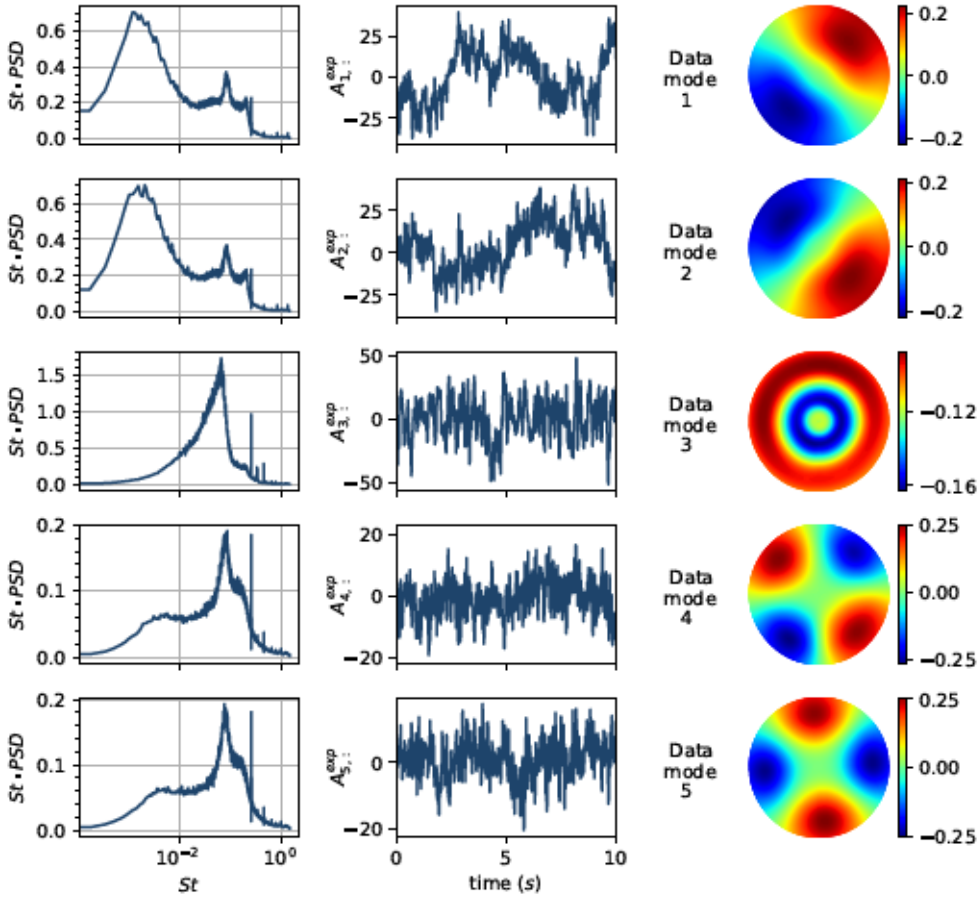


Figure 3.6: Left: the premultiplied PSD ($St\cdot$ PSD) of their associated time coefficients $A^{exp}_{1,:}$ to $A^{exp}_{5,:}$. Middle: the temporal evolution of the data time coefficients, $A^{exp}_{1,:}$ to $A^{exp}_{5,:}$, for the first 10 seconds of the experiment. Right: the first five data modes of the wind-tunnel dataset $\boldsymbol{\Phi}^{exp}_{:,1}$ to $\boldsymbol{\Phi}^{exp}_{:,5}$.

variable selection on the interpretation of the data modes $\mathbf{\Phi}^{exp}$ provided here. We consider the fluctuating field, following the same logic as for the laminar cylinder wake dataset. The final experimental pressure dataset employed for training is $\boldsymbol{P} \in \mathbb{R}^{N \times N_t}$, where $N_t = 420000$ is the number of available snapshots in time, sampled at a sampling frequency 225 Hz and $N = 64$ is the number of pressure sensors. The dataset $\boldsymbol{P}$ contains data collected for over 1800 seconds.

3.3 Autoencoders

An autoencoder approximates the identity function by means of an encoder and a decoder. We define a generic $\boldsymbol{Y} \in \mathbb{R}^{N^* \times N_t}$, where N^* is the spatial dimension. For all time steps, the encoder $F_{en} : \mathbb{R}^{N^*} \rightarrow \mathbb{R}^{N_z}$ maps the snapshot of the data at time t, $\boldsymbol{Y}_{:,t}$ into a latent space, represented by a latent vector $\boldsymbol{Z}_{:,t}$, where $N_z \ll N^*$ is the dimension of the latent space. The i-th latent variable is $\boldsymbol{Z}_{i,:}$. The encoder, F_{en}, is a composition of layers $f_{en}^{(1)}, f_{en}^{(2)}, \ldots, f_{en}^{(n)}$ and activation functions $\theta_{en}^{(1)}, \theta_{en}^{(2)}, \ldots, \theta_{en}^{(n)}$ applied to each layer. The decoder, F_{de}, is a composition of layers $f_{de}^{(1)}, f_{de}^{(2)}, \ldots, f_{de}^{(n)}$ and the activation functions $\theta_{de}^{(1)}, \theta_{de}^{(2)}, \ldots, \theta_{de}^{(n)}$ applied to each layer

$$\begin{aligned} F_{en} &= \theta_{en}^{(n)} \circ f_{en}^{(n)} \circ \theta_{en}^{(n-1)} \circ f_{en}^{(n-1)} \cdots \circ \theta_{en}^{(1)} \circ f_{en}^{(1)}, \\ F_{de} &= \theta_{de}^{(n)} \circ f_{de}^{(n)} \circ \theta_{de}^{(n-1)} \circ f_{de}^{(n-1)} \cdots \circ \theta_{de}^{(1)} \circ f_{de}^{(1)}, \end{aligned} \tag{3.1}$$

where n is the number of layers. Each layer ($f_{en}^{(*)}$ or $f_{de}^{(*)}$) maps the output of the previous layer to the input of the next layer, and is a function of the trainable parameters ($w_{en}^{(*)}$ or $w_{de}^{(*)}$). Each layer is either a convolution or a linear mapping depending on the test case and the datasets. Details of layers and activation functions are explained in sections 3.3.1 and 3.3.2.

The decoding part maps the instant latent vector at t back to the original space, with the output $\hat{\boldsymbol{Y}}_{:,t}$ approximating the input $\boldsymbol{Y}_{:,t}$. The autoencoder, comprised of the encoding and decoding parts, is trained with all t to output $\hat{\boldsymbol{Y}}$, which approximates the input $\boldsymbol{Y}$. The error between $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$ is measured with MSE (Eq. (2.6)). The parameters of the autoencoder, $\boldsymbol{w} = \{w_{en}^{(1)}, \ldots, w_{en}^{(n)}, w_{de}^{(1)}, \ldots, w_{de}^{(n)}, \}$, are obtained by minimizing the MSE

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} MSE\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right). \tag{3.2}$$

This analysis includes two different autoencoder architectures — the standard autoencoder (AE) (Section 3.3.1) and the mode-decomposing autoencoder (MD-AE) (Section 3.3.2). Because we use networks with no bias and train on the fluctuating fields, the trivial latent vector $\mathbf{0}$ maps to the trivial output $\mathbf{0}$. Two types of intermediate layers are employed — convolutional layers for the laminar dataset, and feedforward layers for the wind-tunnel dataset. A feedforward neural network was selected in this case for two main reasons: its ease of implementation and the relatively small number of sensors in the experimental dataset, leading to a manageable number of network parameters. Efficiency is one of the reasons convolutional neural networks are often chosen for computer vision tasks. The specific choice of network types does not compromise

Table 3.1: Different autoencoder architectures and the training datasets.

Results in section	Autoencoder architecture	Inputs $\boldsymbol{Y}$	Spatial dimension N^*
3.5.1	MD-AE (CNN)	**U**	200×129
3.5.2	AE (CNN)	**U**	200×129
3.6	AE (feedforward)	**P**	64

the core of this study, which is to propose a post-processing method not restricted to a single type of AE. Nevertheless, CNN can be adapted to work with data defined in polar coordinates. For example, the cylindrical CNN applies cyclic padding in the angular direction (Kim et al. 2020) to respect periodicity. Weightings based on the radial coordinates should also be included in a modified discrete kernel to account for the sensor arrangement in the experimental dataset, which forms a regular grid in polar coordinates, but a non-uniform grid in Cartesian coordinates.

In this section, we define the autoencoders for a generic input dataset $\boldsymbol{Y}$ where the spatial dimension N^* depends on the dataset under investigation (numerical and wind-tunnel datasets) and the autoencoder architectures. The inputs and autoencoder architectures are summarized in Table 3.1

In this chapter, the norm to be minimized during the training of the AEs does not differentiate between the datasets. Specifically, MSE is minimized for both the laminar wake defined on a Cartesian grid and the experimental dataset defined on a polar grid. Since the experimental measurements are obtained on a polar grid, the sensor distribution is non-uniform, and unweighted MSE may lead to a latent representation that under-emphasizes the flow field in areas where the sensors are more spread out. A better approach would be to weight the MSE using the same weightings as the ones used in POD, as discussed in Section 3.2.2.

3.3.1 Standard autoencoders

We refer to a standard autoencoder (AE) as an autoencoder with the structure shown in Figure 3.7, which consists of one encoder F_{en} and one decoder F_{de}. In compact notation, an AE is

$$\begin{aligned} \boldsymbol{Z}_{:,t} &= F_{en}(\boldsymbol{Y}_{:,t}; \boldsymbol{w}), \\ \hat{\boldsymbol{Y}}_{:,t} &= F_{de}(\boldsymbol{Z}_{:,t}; \boldsymbol{w}). \end{aligned} \tag{3.3}$$

Details of the layers are given in the Appendix A.1 (Table A.1 to A.4). All neural networks that have the convolution encoder and decoder share the same layers and filter size (tables A.1 and A.2). The feedforward AEs have layers given in tables A.3 and A.4 unless stated otherwise. For all the autoencoders, we use the activation function 'tanh' because the outputs contain both positive and negative values. (We performed further tests in which we found that the autoencoders do not train well with an activation function that does not contain 0, such as the 'sigmoid'.) All hyperparameters are given in Table A.5.

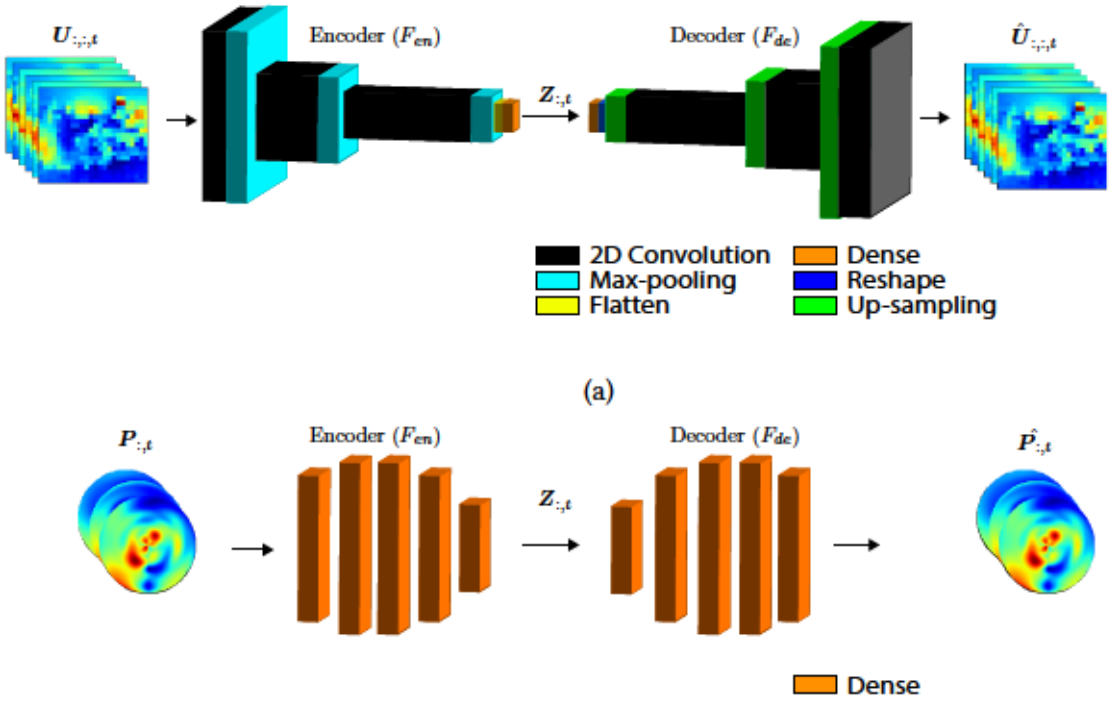


Figure 3.7: The schematics of the standard autoencoder (AE). Different AEs are employed for different datasets and tests. The AE architecture and dataset for each test are listed in Table 3.1. (a) AE for decomposing the laminar cylinder wake $\boldsymbol{U}$ with three convolution layers in both the encoder and the decoder. The hyperparameters are in tables A.1 and A.2 in the Appendix. (b) AE for decomposing the wind-tunnel pressure data $\boldsymbol{P}$ with 5 feedforward layers. The input is a flattened vector of measurement taken from all sensors at time t. The hyperparameters are given in tables A.3 and A.4 in the Appendix.

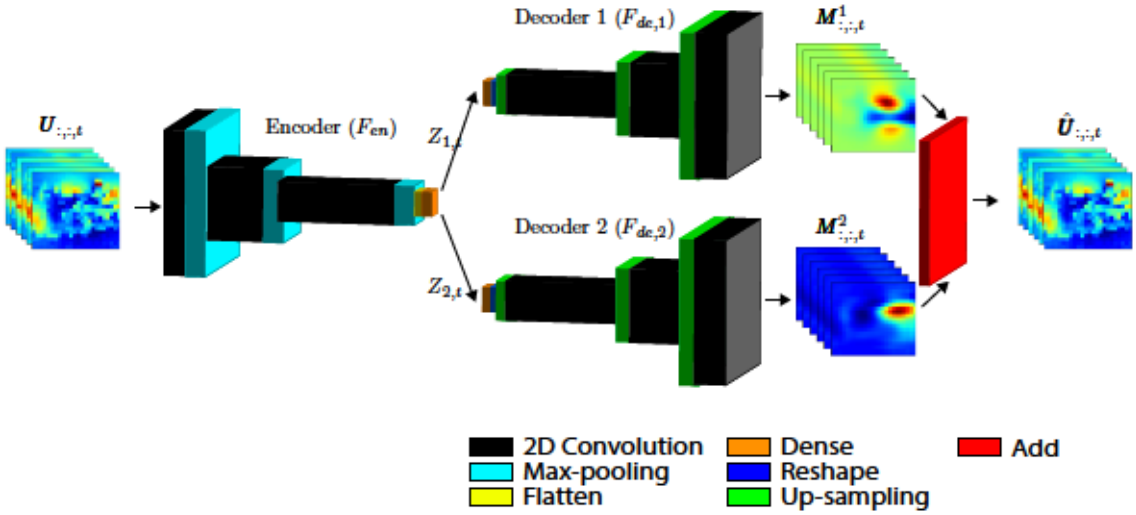


Figure 3.8: Schematic of the MD-AE (Murata et al. 2020) with two latent variables as an example. Each latent variable is decoded by a decoder to produce a decomposed field. The sum of the decomposed fields $\boldsymbol{M}^1$ and $\boldsymbol{M}^2$ is the output of the MD-AE.

3.3.2 Mode-decomposing autoencoders (MD-AE)

An MD-AE associates one decoder to each latent variable (Figure 3.8), so we can visualize the effect of each latent variable separately (Murata et al. 2020). The MD-AE is

$$\begin{aligned} \boldsymbol{Z}_{:,t} &= F_{en}(\boldsymbol{Y}_{:,t}; \boldsymbol{w}), \\ \hat{\boldsymbol{Y}}_{:,t} &= \sum_{i=1}^{N_z} \underbrace{F^i_{de}(\boldsymbol{Z}^i_{:,t}; \boldsymbol{w})}_{\equiv \boldsymbol{M}^i_{:,:,t}} \end{aligned} \tag{3.4}$$

where $\boldsymbol{M}^i$ denotes the i-th decomposed field of the MD-AE and $\boldsymbol{M}^i_{:,:,t}$ is the i-th decomposed field at time step t, corresponding to the output of the i-th decoder. The MD-AE is only applied to the laminar cylinder wake case (the reasoning behind this decision is explained in Section 3.5).

3.4 Decoder Decomposition

The dynamics of POD modes are encapsulated in their time coefficient. To gain physical insight into the decomposed fields, we propose the decoder decomposition to obtain a relationship between the data time coefficients (Eq. (2.3)) and the latent variables trained with the same data. The decoder decomposition is a post-processing method that applies to a trained network. We define the decoder decomposition for AEs in Section 3.4.1 and for MD-AEs in Section 3.4.2.

3.4.1 The decoder decomposition for standard autoencoders

The data modes $\boldsymbol{\Phi}^Y$ (the matrix of POD modes of $\boldsymbol{Y}$, see Table 3.1) form a basis of the subspace in which the output $\hat{\boldsymbol{Y}}_{:,t}$ is approximated as

$$\hat{\boldsymbol{Y}}_{:,t} = \sum_{k=1}^{N^*} B_{k,t} \boldsymbol{\Phi}^Y_{:,k} = \boldsymbol{\Phi}^Y \boldsymbol{B}_{:,t}, \tag{3.5}$$

with $\boldsymbol{B} \in \mathbb{R}^{N^* \times N_t}$ being the temporal coefficients (decoder coefficients). The decoder coefficient for the data mode k at time step t is $B_{k,t}$. The matrix form of equation (3.5) is

$$\hat{\boldsymbol{Y}} = \boldsymbol{\Phi}^Y \boldsymbol{B}^T. \tag{3.6}$$

In other words, the reconstructed flow $\hat{\boldsymbol{Y}}$ is expressed as a linear combination of the POD modes of the original training data $\boldsymbol{Y}$ with $\boldsymbol{B}$ being the matrix of coefficients. In a trained network, $\boldsymbol{B}$ depends on only $\hat{\boldsymbol{Y}}$, which depends only on the latent variables $\boldsymbol{Z}$. Thus, the gradient of the output of the decoder with respect to the latent variables is

$$\frac{d\hat{\boldsymbol{Y}}_{:,t}}{d\boldsymbol{Z}_{:,t}} = \frac{d\hat{\boldsymbol{Y}}_{:,t}}{d\boldsymbol{B}_{:,t}} \frac{d\boldsymbol{B}_{:,t}}{d\boldsymbol{Z}_{:,t}}, \tag{3.7}$$

where the results of the gradients following the conventions of vector-vector gradients. For all time steps, we collect the gradients into gradient matrices

$$\begin{aligned} \frac{d\hat{\boldsymbol{Y}}}{d\boldsymbol{Z}}^T &= \left[\frac{d\hat{\boldsymbol{Y}}_{:,1}}{d\boldsymbol{Z}_{:,1}}^T, \frac{d\hat{\boldsymbol{Y}}_{:,2}}{d\boldsymbol{Z}_{:,2}}^T, \ldots, \frac{d\hat{\boldsymbol{Y}}_{:,N_t}}{d\boldsymbol{Z}_{:,N_t}}^T \right]^T, \\ \frac{d\boldsymbol{B}}{d\boldsymbol{Z}}^T &= \left[\frac{d\hat{\boldsymbol{B}}_{:,1}}{d\boldsymbol{Z}_{:,1}}^T, \frac{d\hat{\boldsymbol{B}}_{:,2}}{d\boldsymbol{Z}_{:,2}}^T, \ldots, \frac{d\hat{\boldsymbol{B}}_{:,N_t}}{d\boldsymbol{Z}_{:,N_t}}^T \right], \end{aligned} \tag{3.8}$$

and combine with equation (3.6) to yield

$$\frac{d\hat{\boldsymbol{Y}}}{d\boldsymbol{Z}} = \boldsymbol{\Phi}^Y \frac{d\boldsymbol{B}}{d\boldsymbol{Z}}. \tag{3.9}$$

The gradient of the decoder coefficients is the sensitivity to changes in the latent variables. Each element of the gradient matrix $\left(\frac{d\hat{\boldsymbol{Y}}}{d\boldsymbol{Z}}\right)_{j,i,t}$ is the rate of change of the output at grid point j and time step t ($\hat{Y}_{j,t}$) with respect to the latent variable $Z_{i,t}$. Because a single decoder coefficient corresponds to a single data mode, which represents a coherent structure, the gradient of the decoder coefficients also reflects the sensitivity of the coherent structure to changes in the latent variables. Physically, the sensitivity quantifies the relative importance of the different latent variables for decoder coefficients and the data modes they represent. We quantify the relative importance of the decoder coefficient $\boldsymbol{B}_{j,:}$ due to latent variable $\boldsymbol{Z}_{i,:}$ by defining the average rate of change

$$\eta_{i,j} = \frac{\int \left| \frac{\partial \boldsymbol{B}_{j,:}}{\partial \boldsymbol{Z}_{t,:}} \right| d\boldsymbol{Z}_{1,:} \ldots d\boldsymbol{Z}_{N_z}}{\int d\boldsymbol{Z}_{1,:} \ldots d\boldsymbol{Z}_{N_z}}. \tag{3.10}$$

The average rate of change of a decoder coefficient due to a latent variable i quantifies its contribution to the data mode j in the output associated with that decoder coefficient (Eq. 3.5). The larger the $\eta_{i,j}$, the more important the latent variable i is in representing the data mode j in the output. This can be used to rank and select latent variables, as demonstrated in Section 3.6.2. In the context of this chapter, 'ranking' is selecting latent variables based on their importance towards a user-defined goal, by analysing the sensitivities of the data POD modes to the changes in the latent variables. For example, if the user wants to further compress the latent space, they may rank the latent variables which have stronger correlations with the energy-containing POD modes higher. An example of using the ranking to make certain POD modes more prominent in the reconstructed flow is shown in Section 3.6.2. Algorithm 1 shows how the average rate of change is computed.

Algorithm 1: Computing the average rate of change

Input : F_{de} - trained decoder
Δz - constant step size

Output : $\boldsymbol{\eta}$ - average rate of change

for $i \leftarrow 1, \ldots, N_z$ **do**
  $\frac{d\hat{\boldsymbol{Y}}}{d\boldsymbol{Z}}_{i,:} \leftarrow \frac{1}{\sigma(\boldsymbol{Z}_{t,:})} \frac{F_{de}(\boldsymbol{Z}_{t,:}+\Delta z) - F_{de}(\boldsymbol{Z}_{t,:}-\Delta z)}{2\Delta z}$; // approximate derivatives & normalize
end
Collect $\frac{d\hat{\boldsymbol{Y}}}{d\boldsymbol{Z}}$;
$\frac{d\boldsymbol{B}}{d\boldsymbol{Z}} \leftarrow {\boldsymbol{\Phi}^Y}^T \frac{d\hat{\boldsymbol{Y}}}{d\boldsymbol{Z}}$;
$\boldsymbol{\eta}^T \leftarrow \frac{1}{N_t} \sum_{t=1}^{N_t} \left| \frac{d\boldsymbol{B}}{d\boldsymbol{Z}} \right|_{:,:,t}$; // average rate of change
return $\boldsymbol{\eta}$

3.4.2 Decoder decomposition for MD-AEs

In an MD-AE, the decomposed fields $\boldsymbol{M}^i$ is decomposed into

$$\boldsymbol{M}^i = \boldsymbol{\Phi}^Y {\boldsymbol{B}^i}^T, \tag{3.11}$$

where $\boldsymbol{B}^i$ is the matrix of the i-th decoder coefficients of the i-th decomposed field. Combined with equation (3.4), the output of the MD-AE becomes

$$\hat{\boldsymbol{Y}} = \boldsymbol{\Phi}^Y \sum_{i=1}^{N_z} {\boldsymbol{B}^i}^T. \tag{3.12}$$

Equations (3.9) to (3.10) still apply as they are.

3.5 Decomposition of the unsteady laminar wake

In this section, we analyse the latent space of AEs trained with the laminar cylinder wake dataset (Section 3.2.1). We first decompose the dataset with an MD-AE and apply the decoder decomposition in Section 3.5.1, which serves as a benchmark to compare against the literature. We then

apply the decoder decomposition to correlate the latent variables of AEs with the data modes in Section 3.5.2.

3.5.1 The latent space of a mode-decomposing autoencoder

We decompose the unsteady wake dataset, described in Section 3.2.1, with an MD-AE with $N_z = 2$ (Figure 3.9) and compare our results with (Murata et al. 2020) to verify the decoder decomposition. The reconstruction loss, measured with MSE (Eq. (2.6)), is 2.6×10^{-5}, showing that the output $\hat{\boldsymbol{U}}$ is an accurate approximation of $\boldsymbol{U}$. Figure 3.9a shows that the latent variables have the same periodic behaviour as the data time coefficients, matching the results of (Murata et al. 2020). By applying POD to the decomposed fields and by inspection, the decomposed field 1 contains the data modes 1, 3 and 6, and the decomposed field 2 contains the data modes 2,4 and 5 (Figure 3.9c), similarly to what was observed by (Murata et al. 2020). This observation is based on visual inspection of the POD modes of the decomposed fields (Figure 3.9c) and the data modes (Figure 3.3), which is a qualitative comparison. Therefore, the POD modes of the decomposed fields may have different physical interpretations to the data modes they resemble. A quantitative comparison between the decomposed fields and the data modes is enabled later in this section by applying the decoder decomposition. Figure 3.9b shows that the two latent variables have the same frequency spectrum, with peaks at $St = 0.23$ and 0.69, corresponding to the vortex shedding frequency and its second harmonic. Because the decomposed field 1 contains a POD mode similar to data mode 1 and the decomposed field 2 contains a similar mode to data mode 2, and the two latent variables have an identical frequency spectrum, are periodic and out of phase, we conclude that the two latent variables form a pair similar to the data time coefficients. However, the latent variables have two frequency peaks instead of a singular peak of the first two data time coefficients, showing that the latent variables contain nonlinear temporal information.

We apply the decoder decomposition to the MD-AE and plot the first four decoder coefficients for the two decomposed fields in Figure 3.10. None of the first four decoder coefficients are constant in Figure 3.10, meaning that the first four data modes all contribute toward both decomposed fields. The relative frequencies of the latent variables and the decoder coefficients can also be deduced from Figure 3.10. As the latent variables change from -1 to 1, decoder coefficients 1 and 2 in both decomposed fields complete a quarter of a sine wave, but the decoder coefficients 3 and 4 complete half a sine wave, meaning that decoder coefficients 3 and 4 oscillate at double the frequency of decoder coefficients 1 and 2.

For a more quantitative measure of how the data modes are split into the decomposed fields of the MD-AE, we calculate the equivalent energy (Kneer et al. 2021) for each decomposed field separately. The matrix of equivalent energy for the i-th decomposed field and the data modes of

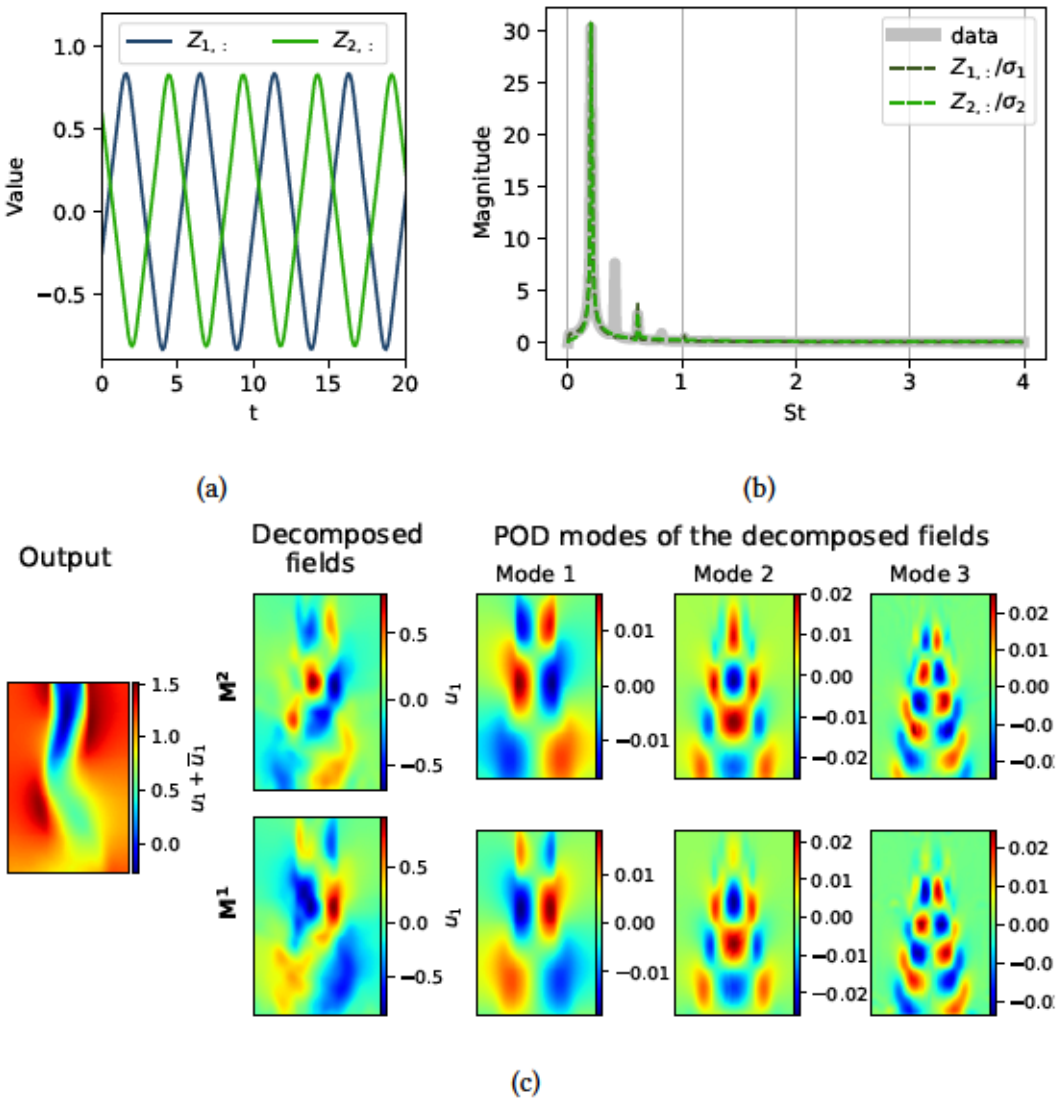


Figure 3.9: Results of training the MD-AE on the laminar wake dataset with a latent dimension of 2. a) Both latent variables are periodic in time. b) Frequency spectrum of the latent variables, normalised by their standard deviation, compared with the frequency of the data. The latent variables both contain the vortex shedding frequency and the second harmonic of the vortex shedding frequencies. c) A snapshot output of the MD-AE (mean flow added) and the POD modes of the decomposed fields. The POD modes of decomposed field 1 are similar to data mode 1,3 and 6 and the POD modes of decomposed field 2 are similar to data mode 2,4 and 5.

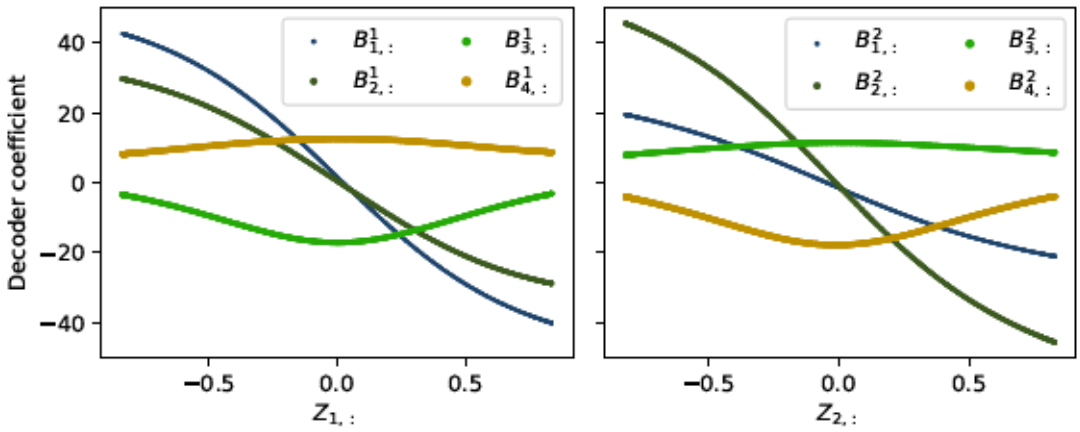


Figure 3.10: Results from the MD-AE trained with the laminar wake dataset with 2 latent variables. The first four decoder coefficients plotted against the latent variables. Both latent variables affect the magnitude of the first four POD modes of the data in the output of the MD-AE.

the laminar flow, $\hat{\boldsymbol{\Lambda}}^i$, is given by

$$\hat{\boldsymbol{\Lambda}}^i = \frac{1}{N_t - 1} \boldsymbol{B}^i \boldsymbol{B}^{i\,T}, \tag{3.13}$$

which is the variance of the i-th decomposed field, $\boldsymbol{M}^i$, projected onto the data modes. The equivalent energy of the i-th decomposed field and the k-th data mode is $\hat{\Lambda}^i_{kk}$. As $\boldsymbol{M}^i$ approaches the first data POD decomposed field ($\boldsymbol{\Phi}^{lam}_{:,1} \boldsymbol{A}^T_{i,:}$), $\hat{\Lambda}^i_{11}$ approaches Λ_{11} while $\hat{\Lambda}^i_{kk}$, where $k \neq 1$, approaches 0.

Figure 3.11 shows the equivalent energy of the first six data modes in the two decomposed fields and in the predicted output $\hat{\boldsymbol{U}}$. The equivalent energies of the predicted output for all data modes are approximately 100%, meaning that the output $\hat{\boldsymbol{U}}$ is an accurate approximation of $\boldsymbol{U}$. Figure 3.11 also shows that all latent variables jointly contribute toward the presence of the data modes in the output, which also shows that the MD-AE does not fully separate data modes into different decomposed fields.

In summary, the decoder decomposition applied to an MD-AE (Murata et al. 2020) leads to a detailed interpretation of the decomposed fields. By computing the equivalent energy, we also show that, although the MD-AE works well for visualisation, the decomposed fields are entangled.

3.5.2 The latent space of standard autoencoders

Unlike the MD-AE, an AE has no decomposed fields to visualize. We apply the decoder decomposition to AEs trained with the unsteady wake dataset to connect the data time coefficients to the latent variables. Since the frequency content of the laminar, periodic wake of a cylinder and the data modes are well known (M. M. Zdravkovich 1997), we use the physical interpretation of the data modes and time coefficients to provide a physical interpretation of the latent variables.

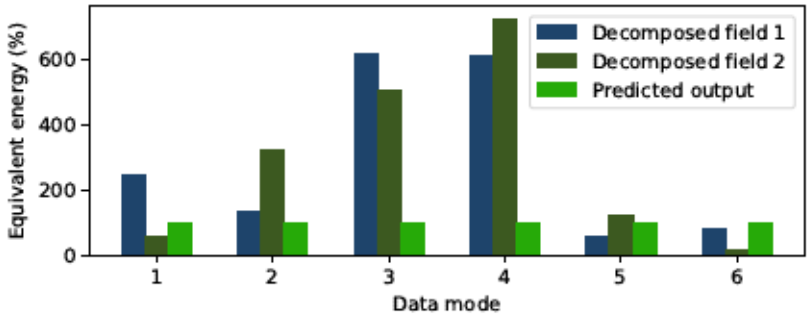


Figure 3.11: Results from the MD-AE trained with the laminar wake dataset with 2 latent variables. The equivalent energy of the two decomposed fields of the MD-AE, showing the first four POD modes of the data. All POD modes of data are present in both decomposed fields

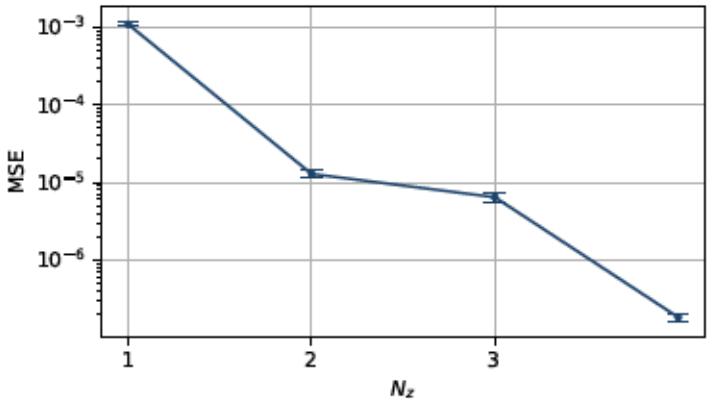
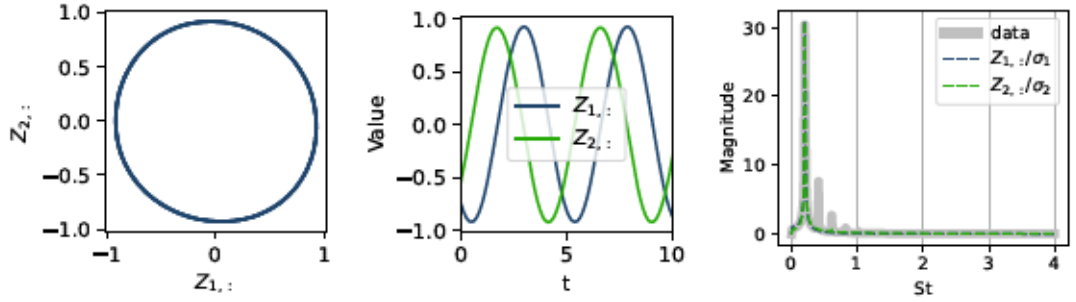


Figure 3.12: The MSE of AEs trained with the unsteady wake dataset with different numbers of latent variables averaged over five repeats each. The error bars represent the standard deviation of the repeats. At $N_z = 2$ the MSE is approximately 1.3×10^5.

Figure 3.12 shows the loss of AEs trained with the unsteady wake dataset using 1, 2, 3 and 4 latent variables. The first plateau of loss occurs at $N_z = 2$ before the loss decreases again at $N_z = 4$. Similar behaviour is also observed by (Csala et al. 2022). The AEs with two latent variables achieve a significantly smaller loss than AEs with a single latent variable but the difference in loss between using two and three latent variables is small. The laminar wake $\boldsymbol{U}$ has periodic underlying dynamics and thus can be described by a single variable in polar coordinates (the angle) or two variables in Cartesian coordinates. However, the AEs do not find the best representation when forced to use only one latent variable due to their inability to perform Cartesian to polar transformation. Polar to Cartesian relies on the sin function, which is a many-to-one function and can be learned by neural networks either through approximation with any activation function or by using sin as the activation function (Wong et al. 2022). However, the Cartesian to polar transformation relies on arcsin, which is not defined for all inputs. The Cartesian to polar transformation is a known difficulty in machine learning, with benchmark problems such as the



(a) (b) (c)

Figure 3.13: Results from the AE trained with the laminar wake dataset with two latent variables. a) Phase plot of two latent variables. The unit circle indicates harmonic oscillations. b) Time trace of the two latent variables. The latent variables are periodic and 90° out of phase, behaving the same as the first two data time coefficients. c) The frequency spectrum of the latent variables, compared to the frequency in the dataset. The variables are normalized by their standard deviation, σ, before applying the Fourier Transform for visualisation.

two-spirals problem built around it (Fahlman & Lebiere 1989, Liu et al. 2018), this is consistent with our observations (Figure 3.12). Therefore, we analyse an AE with $N_z = 2$ with decoder decomposition to obtain physical interpretation in the current section, even though $N_z = 1$ should be the theoretical minimum representation of the laminar wake.

The phase plot of the two latent variables is a unit circle (Figure 3.13a), which indicates that the latent variables of an AE are periodic; the same phase plot also describes the first two data time coefficients. The same periodic behaviour also shows in the time trace of the latent variables (Figure 3.13b). Figure 3.13c compares the frequency spectrum of the latent variables and the data. We find that both latent variables oscillate at the vortex shedding frequency, which is the lowest frequency in the dataset (Figure 3.2, right panel). The higher harmonic frequencies are not included in the latent variables because they are functions of the vortex-shedding frequencies, and the decoders can represent the spatial patterns of multiple data modes. The AE has produced latent variables which have a singular peak in the frequency spectrum, less than the frequency spectrum of the latent variables of the MD-AE, without sacrificing accuracy, showing that AEs are more expressive given the same number of latent variables than MD-AEs.

Figure 3.14 shows the sensitivity of the decoder coefficients, $\boldsymbol{B}_{1,:}, \ldots, \boldsymbol{B}_{4,:}$ to the two latent variables. The contours are the values of the decoder coefficients, shown for all values of the latent variables allowed by the nonlinear activation function. The dynamics of the dataset are shown as the grey circle, which shows the values of the latent variables that are observed during training.

The decoder coefficients $\boldsymbol{B}_{3,:}$ and $\boldsymbol{B}_{4,:}$ have double the frequency of $\boldsymbol{B}_{1,:}$ and $\boldsymbol{B}_{2,:}$, consistent with

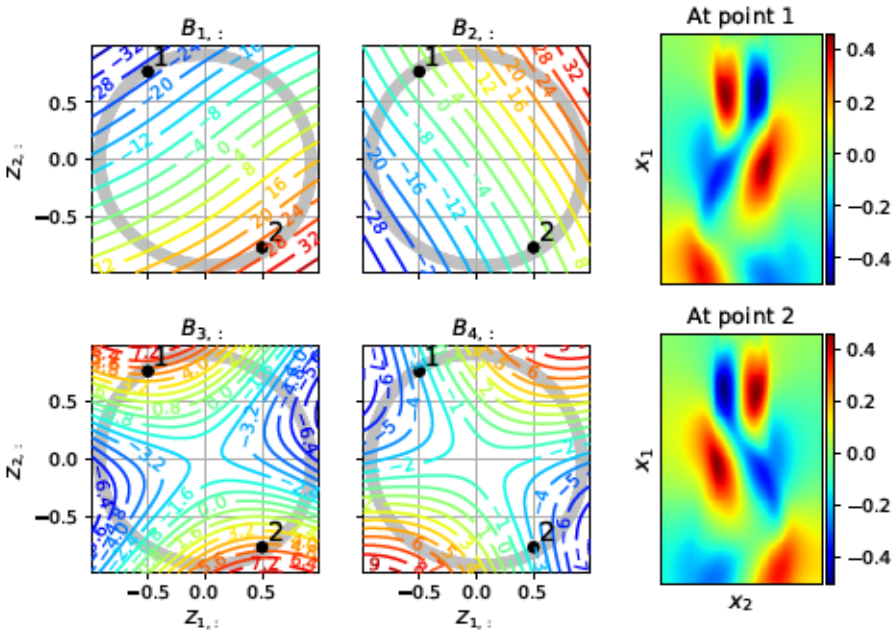


Figure 3.14: Contour plots of the first four decoder coefficients of an AE with two latent variables, trained with the laminar wake dataset. The contours show the values of the decoder coefficients. The grey circle labels the values of the latent variables observed during training, which shows the dynamics of the dataset. The images in the last column show the output of the AE at point 1&2 labelled on the contour plots.

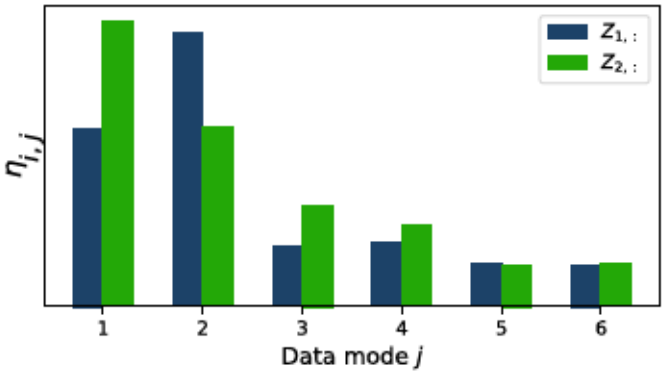
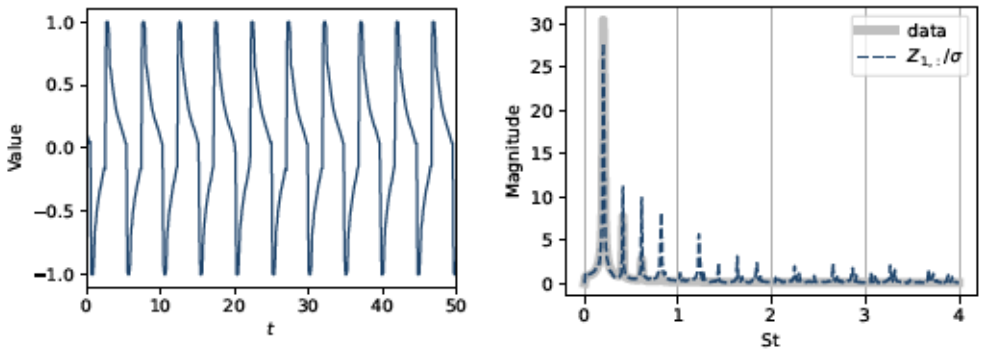


Figure 3.15: Results from the AE trained with the laminar wake dataset with 2 latent variables. The average rate of change of the decoder coefficients.

the characteristics of the first four data modes. At $\boldsymbol{Z} = \boldsymbol{0}$, all decoder coefficients have the value 0 because the network has no bias. Figure 3.14 provides a qualitative overview of the sensitivity of the data modes to the latent variables.

Because the decoder output $\hat{\boldsymbol{U}}$ depends on multiple latent variables, the equivalent energy (Eq. (3.13)) cannot be used to identify the contribution of each latent variable. We use the average rate of change (Eq. (3.10)) to quantify the contribution of individual latent variables of the AE (Figure 3.15). Amongst the first four data modes, latent variable $\boldsymbol{Z}_{1,:}$ contributes most to the presence of the second data mode $\boldsymbol{\Phi}^{lam}_{:,2}$ in the output $\hat{\boldsymbol{U}}$, while $\boldsymbol{Z}_{2,:}$ mainly contributes towards the presence of the other three data modes. The first 6 data modes depend on both latent variables, so no latent variable can be removed without affecting the dynamics of the output. However, the average rate of change suggests that the smaller-scale energy-containing structures between the vortex streets (M. M. Zdravkovich 1997), namely the pair comprised of data mode 3 and 4, are mainly represented by $\boldsymbol{Z}_{2,:}$. Therefore, $\boldsymbol{Z}_{2,:}$ should be used if the representation of data modes 3 and 4 in the output is required.

By comparing the frequency spectrum of the latent variables of the AE and the MD-AE (Section 3.5.1), both with $N_z = 2$, we find that the AE has learned a more accurate representation of the data than the MD-AE because the latent variables have the minimum number of frequency peaks needed to represent the data. The frequency of the two latent variables coincides with the lowest frequency in the frequency spectrum of the data, meaning that the data modes with higher frequencies are added by the decoder. By introducing the average rate of change, we show that both latent variables are equally important in representing the vortex shedding, but the first harmonic is contained mainly in one of the latent variables in this case. Because the training of an AE depends on random seeds, the physical meaning of the latent variables may change for different cases. During training, we noticed that the dimension of latent space has a significant



(a) (b)

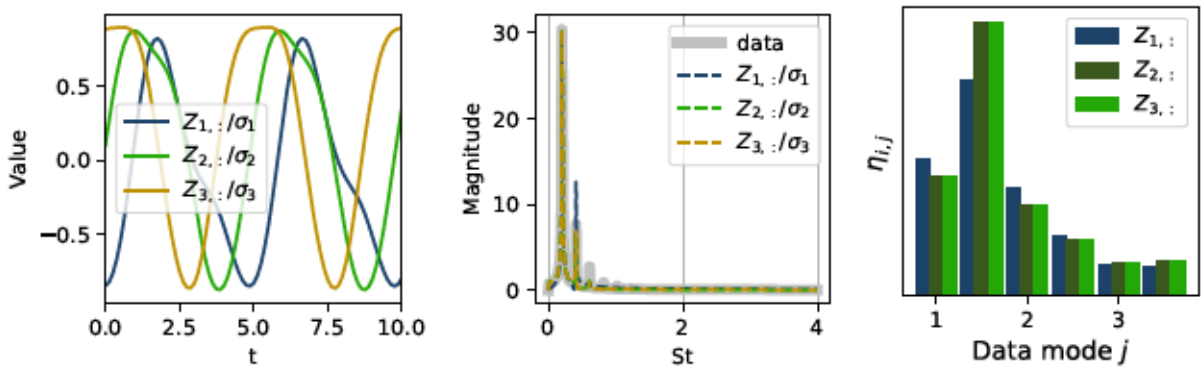
Figure 3.16: Results from the AE trained with the laminar wake dataset with 1 latent variable. a) The latent variable approximates the discontinuity caused by the angle moving from 2π to 0. b) The frequency spectrum of the latent variable, normalized by its standard deviation, compared with the data frequencies. The latent variables contain frequencies that do not exist in the dataset.

impact on the physical interpretation of the latent variables.[1]

3.5.3 Physical interpretation is more difficult when unsuitable latent dimension is used

In Section 3.5.2, we show that the MSE of the AEs converges for $N_z = 2$, and that at least two latent variables are needed to describe the periodic behaviour. However, it is difficult to know the theoretical number of latent variables to use in common machine learning tasks. The common approach in machine learning is to treat the latent dimension as a hyperparameter of the networks. By using the laminar wake as an example, we discuss how the choice of the latent variables affects the physical interpretation of the latent variables. First, we show the results from an AE with $N_z = 1$ in Figure 3.16. The latent variable is periodic in time, but the changes are not smooth (Figure 3.16a) and have frequency peaks at high frequencies not present in the data (Figure 3.16b). When a single variable (the angle) is used to represent the periodic behaviour of the dataset $\boldsymbol{U}$, there is a discontinuity when the angle moves from 2π to 0. The latent variable is forced to use higher frequencies to try to approximate the discontinuity. These higher frequencies are the results of numerical approximation and are unphysical. In an AE with $N_z = 3$, the latent variables contain the first harmonic in addition to the vortex shedding frequency (Figure 3.17b), which is present in the data but not in the first two data time coefficients (Figure 3.2, right panel) or the latent variables from an AE with two latent variables (Figure 3.13b). We discussed (Section 3.5.2) that the additional frequency peak is not necessary for the accurate reconstruction of the dataset.

[1]The results presented in Section 3.5.2 are from AEs trained with the 'tanh', which are consistent. Training an AE with $N_z = 2$ using 'ReLU' occasionally lead to results similar to those shown in Appendix 3.5.3.



(a) (b) (c)

Figure 3.17: Results from the AE trained with the laminar wake dataset with three latent variables. a) The time trace of the latent variables. b) The frequency spectrum of the latent variables, normalized by their standard deviations. The latent variables contain the vortex shedding and the first harmonic frequency. c) The averaged rate of change of decoder coefficients due to latent variables of an AE with three latent variables. Latent variables $Z_{2,:}$ and $Z_{3,:}$ have the same contributions towards the first six data modes, meaning that both latent variables carry the same information.

The average rate of change shows that latent variables $Z_{2,:}$ and $Z_{3,:}$ contribute equally to the first six data modes (Figure 3.17c). Equal contribution to all data modes indicates that the two latent variables have the same role in representing the data, which means that the information carried by one of the latent variables is duplicate information contained in the other latent variables. The three latent variables are not suitable because they lead to redundant information. All three latent variables contribute similarly to the first six data modes, making the selection of latent variables more difficult. In the present case, the three latent variables form a group to represent the counter-rotating vortices, even though two latent variables should be, in principle, sufficient.

Both the AE with $N_z = 3$ and the MD-AE with $N_z = 2$ (Section 3.5.1) have latent variables with an additional frequency peak compared to the latent variables of the AE with $N_z = 2$ (Section 3.5.2), even though there is no substantial difference in the reconstruction error. Additionally, unphysical frequencies also arise from numerical approximations when using AE with $N_z = 1$. Therefore, a design consideration of AEs for nonlinear reduced-order modelling is to use a latent dimension that results in the least number of frequency peaks in the latent variables whilst maintaining the reconstruction accuracy of the AEs.

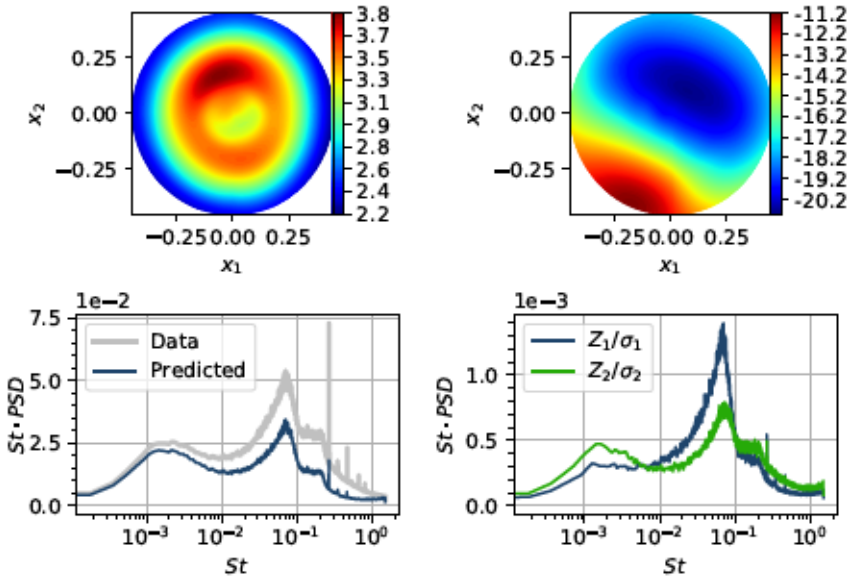


Figure 3.18: The AE trained with the wind-tunnel dataset with two latent variables. Top left: predicted RMS pressure. Top right: predicted instantaneous pressure. Bottom left: the premultiplied overall PSD of the prediction. Bottom right: the premultiplied PSD of the latent variables of the AE trained with two latent variables, normalized by their standard deviation.

3.6 Autoencoder decomposition of the turbulent wake from wind-tunnel experiments

We analyse the AEs to decompose the wind-tunnel wake dataset $\boldsymbol{P}$ (Section 3.2.2). We show that, for a turbulent dataset, the size of the network becomes an additional limiting factor when trying to correlate the latent variables with the data modes in Section 3.6.1. We demonstrate how to select latent variables to single out a particular flow structure in Section 3.6.2.

3.6.1 The limiting factor is network size for turbulence

At least two latent variables are needed to represent flows with periodicity (Section 3.5.2). Because the turbulent wake of the axisymmetric body contains periodic behaviours such as vortex shedding (Rowan D. Brackston 2017), we first train an AE with two latent variables on the wind-tunnel dataset (feedforward, see Table 3.1). Figure 3.18 (left, centre panels) shows the RMS pressure and the frequency content of the predicted pressure. The frequency peaks of the reconstructed pressure field match the data, but the RMS pressure and the magnitude of the PSD differ from the reference pressure field. We investigate the effect of increasing the latent variables from two to three. The prediction made by the AE with $N_z = 3$ is more accurate in both the RMS pressure and frequency content (Figure 3.19, left panel, centre panels) compared to the prediction by the AE with $N_z = 2$. By increasing the number of the latent variables from two to three, the prominent frequencies of the prediction remain unchanged, but the magnitude is

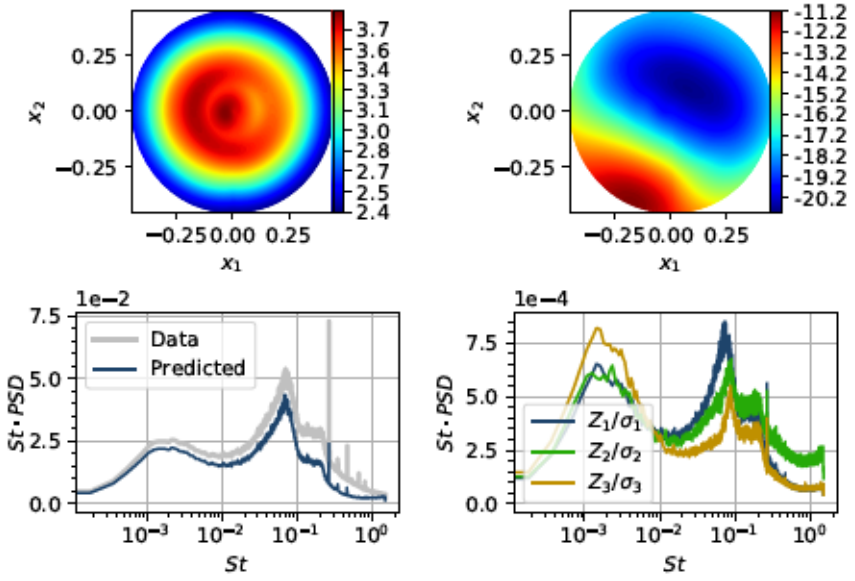


Figure 3.19: Same as Figure 3.18 with three latent variables.

closer to the reference data. By increasing the dimension of the latent space from $N_z = 2$ to $N_z = 3$, the prediction has improved without changing any frequency-related behaviour. This suggests that the benefit of increasing the latent dimension comes from increasing the size of the decoder, thus improving the decoder's ability to express spatial patterns.

To further understand the cause of the improved prediction, we train a large decoder-only network (Figure 3.20) with the latent variables obtained with the previous $N_z = 2$ AE. (The details of the large decoder are given in Table A.6.) The large decoder has approximately twice the number of trainable parameters than the decoder in the previous $N_z = 2$ AE. The input to the large decoder is the latent variables obtained from the trained $N_z = 2$ AE, which have already been analysed in Figure 3.18 (right panel); the output of the large decoder approximates the reference fluctuating pressure $\boldsymbol{P}$.

The PSD of the large decoder-only network overlaps with the PSD of the $N_z = 3$ smaller-sized AE (Figure 3.21), both are closer to the PSD of the reference data than the smaller $N_z = 2$ AE. When the latent variables remain unchanged, the large decoder-only $N_z = 2$ network can make predictions that match the predictions made by a smaller dimension $N_z = 3$ AE. By comparing the results of the smaller size $N_z = 2$ and $N_z = 3$ AEs to the large decoder-only network with identical latent variable to the $N_z = 2$ AE, we find that using a larger decoder has the same effect as increasing the number of latent variables. More importantly, the frequency peaks in the latent variables and predictions remain the same for all three networks. Therefore, we conclude that the critical factor for an accurate prediction using the wind-tunnel dataset, which contains multiple spatiotemporal scales, is the size of the decoder. In an AE, the weights are responsible for representing the spatial patterns in the flow, and the latent variables represent the time-

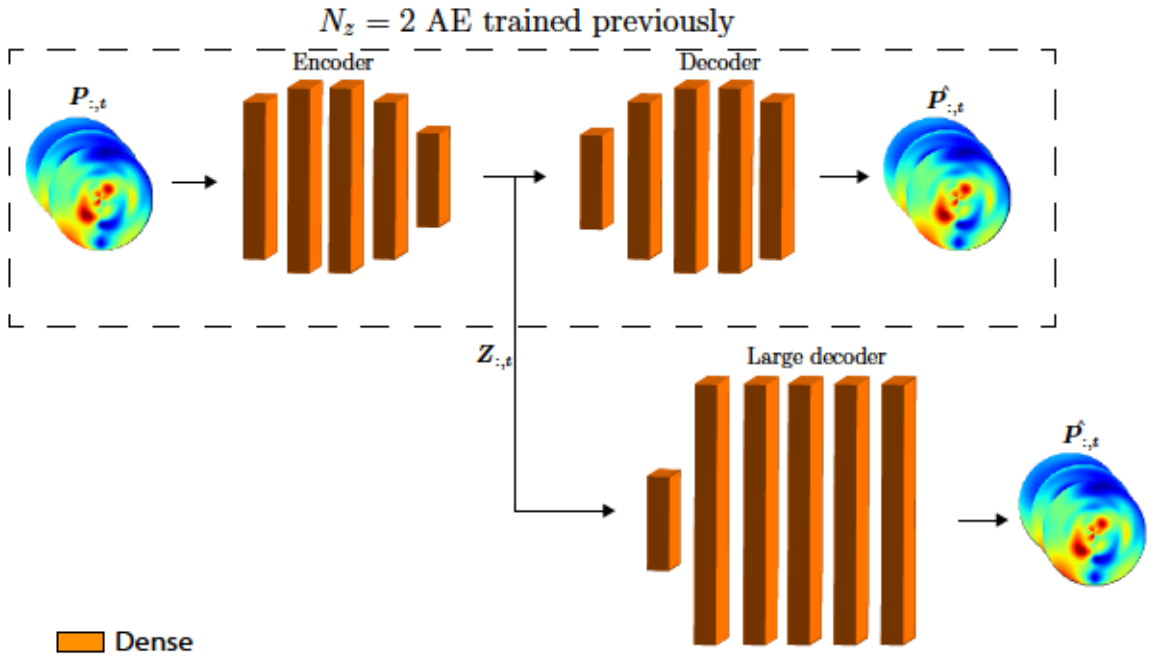


Figure 3.20: The schematics and training process of the decoder-only network. The decoder has 99% more trainable parameters than the decoder in the $N_z = 2$ AE discussed in Section 3.6.1.

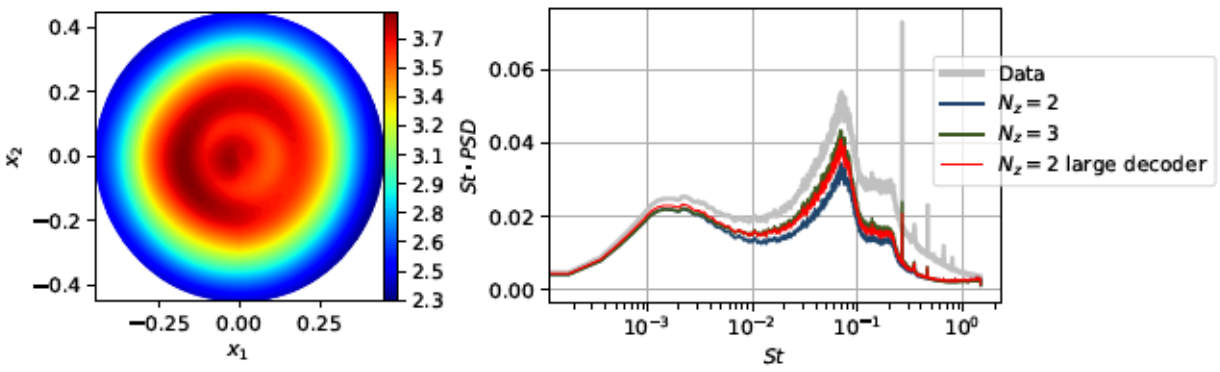


Figure 3.21: Left: predicted RMS pressure of the decoder-only network. Right: The premultiplied PSD of the predicted pressure from the AE with two and three latent variables, and from the large decoder-only network, trained with the wind-tunnel dataset.

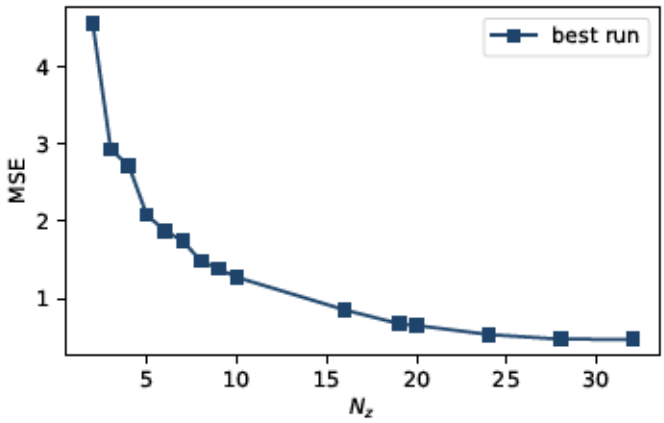


Figure 3.22: AE with different numbers of latent variables and the same hyperparameters, trained with the wind-tunnel dataset. The loss stops decreasing at $N_z \approx 28$.

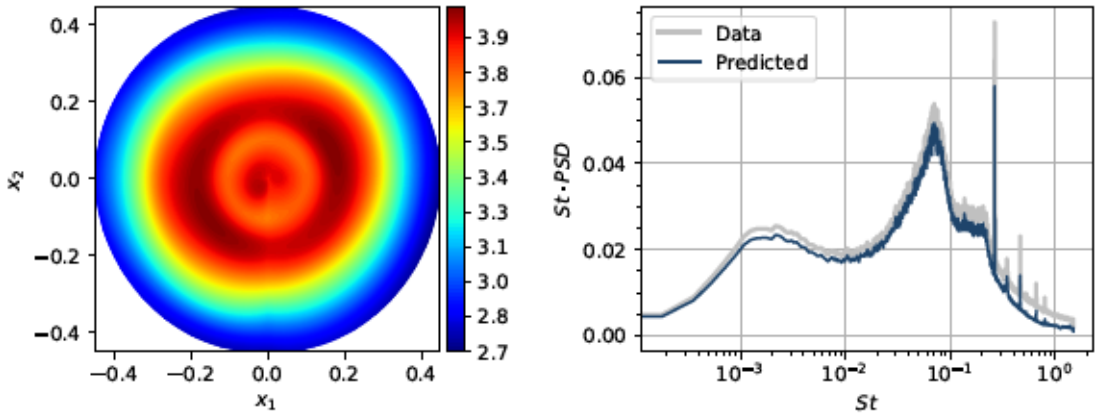
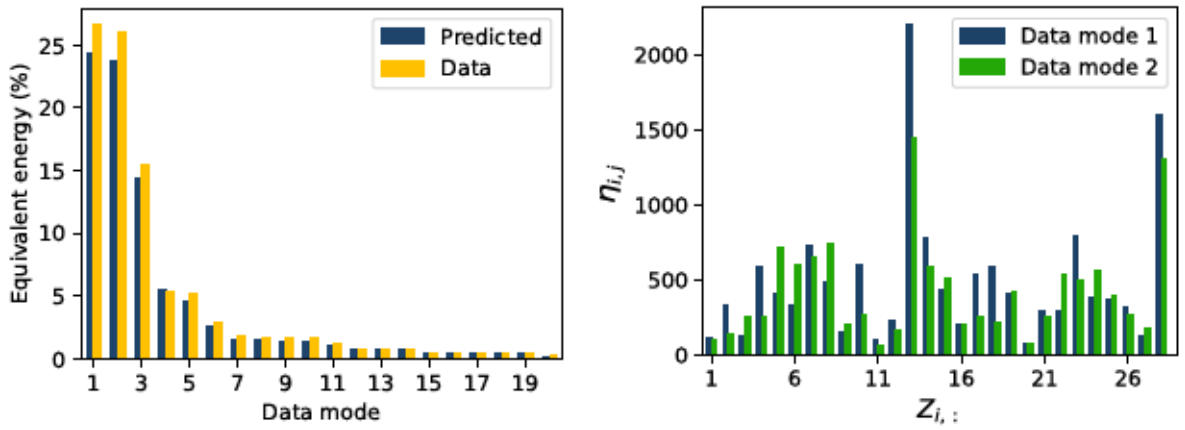


Figure 3.23: RMS predicted pressure and premultiplied PSD of the prediction of the AE trained with the wind-tunnel dataset with 28 latent variables. The network attains an MSE equivalent to the reconstruction with 30 POD modes.

dependent behaviours. We found that the optimal dimension of the latent space is a function of both of the underlying manifold's dimension (Magri, L. & Doan, N.A.K 2022, Doan et al. 2023, Racca et al. 2023) and the size of the decoder.

3.6.2 Filtering unwanted latent variables

Section 3.6.1 shows that changing both the dimension of the latent space, N_z, and the size of the AEs have a similar effect for the turbulent datasets. In this section, we apply the proposed decoder decomposition to filter out unwanted latent variables. When training on the turbulent pressure dataset, $\boldsymbol{P}$, we find that the MSE is sufficiently small at $N_z \approx 28$ (Figure 3.22). Therefore, we employ the AE with 28 latent variables. We filter the latent variables to enhance or reduce the importance of certain data modes amongst other data modes in the output of the AEs, similar to selecting data modes for reconstructing only a particular coherent structure. Figure 3.23 shows the results from the AE with 28 latent variables. The RMS predicted pressure is in agreement



(a) (b)

Figure 3.24: The AE trained with the wind-tunnel dataset with 28 latent variables. a) Equivalent energy of the predicted pressure compared to the POD eigenvalues of the reference data. b) The average rate of change of decoder coefficients 1 and 2 due to latent variables, normalized by the standard deviation of the latent variables. The decoder coefficients 1 and 2 are direct analogies of the data POD time coefficients 1 and 2, which are associated with vortex shedding.

with the reference RMS pressure, and the prediction has PSD that approximates the data's PSD in terms of both magnitude and frequency content. To understand which data modes are present in the prediction, we define the equivalent energy for an AE with a POD weight matrix, $\boldsymbol{W}_p$ (Eq. 2.2), as

$$\hat{\boldsymbol{\Lambda}} = \frac{1}{N_t - 1} \boldsymbol{\Phi}^{Y\,T} \left((\boldsymbol{W}_p \boldsymbol{\Phi}^Y \boldsymbol{B}^T)(\boldsymbol{B} \boldsymbol{\Phi}^{Y\,T} \boldsymbol{W}_p^T) \right) \boldsymbol{\Phi}^Y \tag{3.14}$$

from equation 3.13 (Kneer et al. 2021). The matrix $\boldsymbol{\Phi}^Y = \boldsymbol{\Phi}^{exp}$ contains the data modes. Figure 3.24a compares the equivalent energy of the predicted pressure from the AE with $N_z = 28$ with the POD eigenvalues of the reference data. The prediction contains a similar amount of energy to the reference data for the more energetic coherent structures identified by POD. We focus our analysis on data modes 1 and 2, which represent vortex shedding. Figure 3.24b shows how each latent variable affects the decoder coefficients 1 and 2 in the prediction, which are associated with POD modes 1 and 2 in the prediction. We compute the average rate of change $\eta_{i,j}$ (Eq. (3.10)) and focus the analysis on data modes 1 and 2, which represent vortex shedding. Figure 3.24b shows how each latent variable affects the decoder coefficients 1 and 2 in the prediction, which are associated with POD modes 1 and 2 in the prediction. For both data modes, the largest contribution comes from the latent variable 13, followed by the latent variable 28. The numbering changes with the seeds and gradient update algorithms, but the conclusions do not. For the output of the AE to represent mainly the data mode 1 and 2, we set all latent variables (except for $\boldsymbol{Z}_{13,:}$ and $\boldsymbol{Z}_{28,:}$) to $\boldsymbol{0}$. This yields the filtered latent space $\boldsymbol{Z}_f$ such that

$$\hat{\boldsymbol{P}}_f = F_{de}(\boldsymbol{Z}_f), \tag{3.15}$$

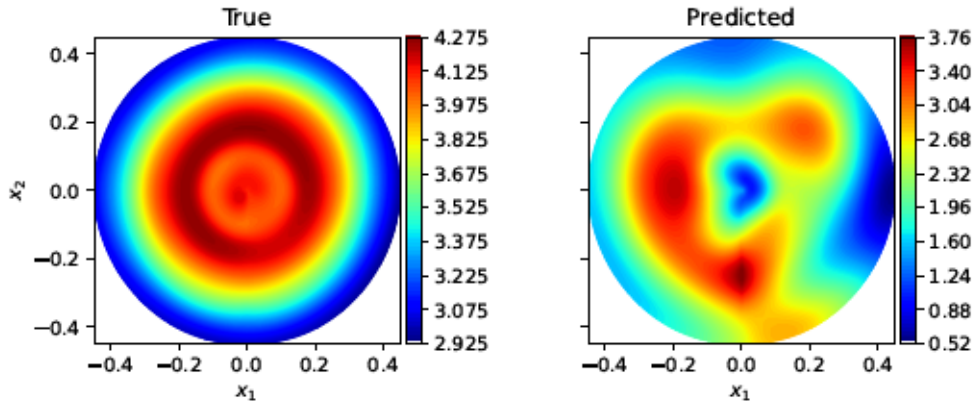


Figure 3.25: Prediction obtained with the AE with 28 latent variables from the filtered latent variables $\boldsymbol{Z}_f$ and the reference data.

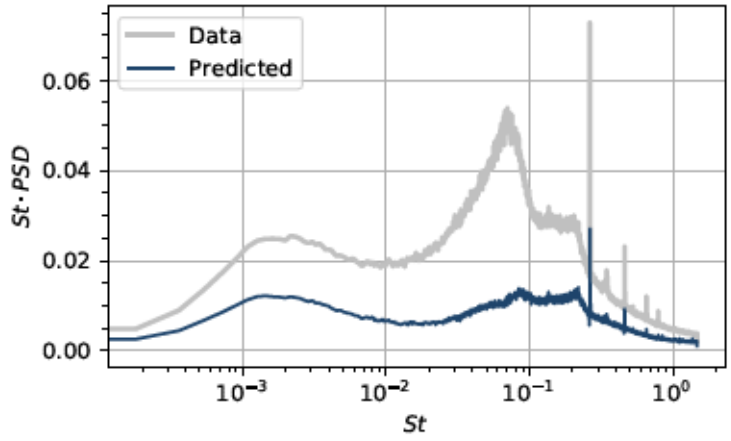
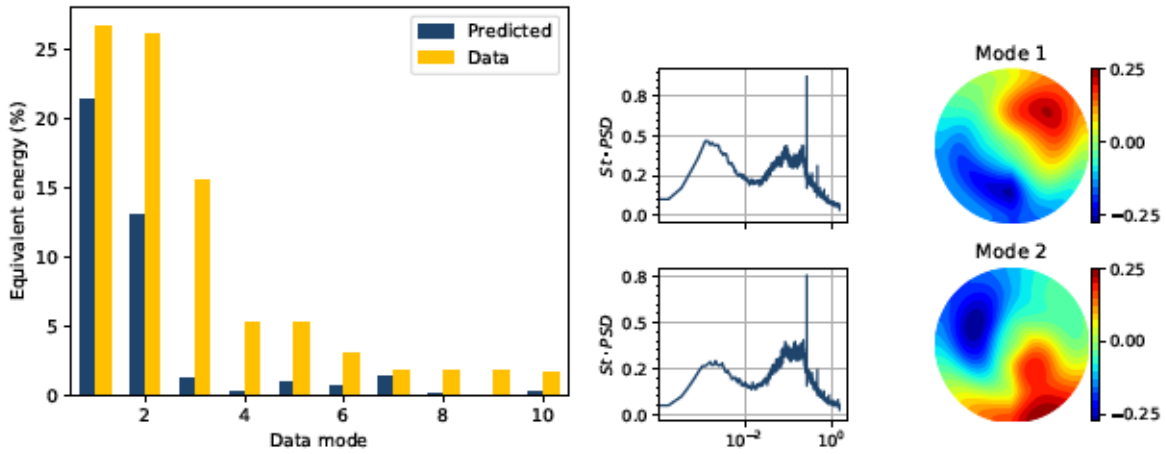


Figure 3.26: Premultiplied PSD of the prediction obtained with the AE with 28 latent variables from the filtered latent variables. All peaks (except for $St \approx 0.2$) are filtered.

which contains mostly data modes 1 and 2. By filtering, we identify the contribution that the latent variables have to the data modes in the output and manipulate some latent variables based on that contribution. Technically, by setting the latent variables to $\mathbf{0}$, we eliminate the fluctuation of the data modes in the output of the AE. Since the AE is nonlinear, setting all but latent variables $\boldsymbol{Z}_{13,:}$ and $\boldsymbol{Z}_{28,:}$ to $\mathbf{0}$ is not equivalent to truncating the data modes from mode 2. Thus, the filtering process aims to minimise the fluctuation of the data modes $\boldsymbol{\Phi}^{exp}_{:,3}$ to $\boldsymbol{\Phi}^{exp}_{:,N_z}$ in the output. The results of decoding the filtered latent variables are shown in Figure 3.25. The RMS of the filtered prediction shows strong fluctuations only in the outer region. The PSD of the filtered prediction (Figure 3.26) shows that $\hat{\boldsymbol{P}}_f$ has a large amount of energy at $St \approx 0.2$ and $St \approx 0.002$. The frequency $St \approx 0.06$ does not appear in the PSD of the filtered prediction, meaning that by keeping only the two most contributing latent variables for data modes 1 and 2, the AE no longer models the pulsation of the vortex core. This means that the filtering has been successful.

The equivalent energy in Figure 3.27a further shows that the filtered prediction represents only



(a) (b)

Figure 3.27: POD on the prediction obtained with the AE with 28 latent variables from the filtered latent variables. a) Equivalent energy of the filtered prediction $\hat{\boldsymbol{P}}_f$ compared to the POD eigenvalues of the reference data. Amongst the five most energetic POD modes, the filtered prediction shows a large amount of energy only in data modes 1 and 2, which is the goal we set. b) POD modes 1 and 2 of $\hat{\boldsymbol{P}}_f$, which contain over 99% of the flow energy of $\hat{\boldsymbol{P}}_f$. These two modes show the same frequency peak at $St \approx 0.2$ and have anti-symmetric spatial structures. These two modes represent vortex shedding.

the data modes 1 and 2, which is the objective that we set. Figure 3.27b shows the two leading POD modes of the filtered prediction. The modes are approximately antisymmetric with frequency peaks at $St \approx 0.2$ and $St \approx 0.002$, which represent vortex shedding. The structures in Figure 3.27b are not perfectly antisymmetric due to information being lost by removing 26 out of 28 latent variables. The two leading modes contain over 99% of the energy in the filtered prediction, showing that the filtering has significantly compressed the spatial information. Combining the energy, the structures and frequency content, we conclude that the only coherent structures in the filtered prediction are modes associated with vortex shedding. By selecting latent variables based on contribution to the decoder coefficients, we have successfully filtered the coherent structures of the output of the AE. Thus, we have shown decoder decomposition as a viable method for selecting latent variables based on coherent structures of the flow for an AE with a large latent dimension. This is useful for filtering spurious or unwanted latent variables, as well as singling out specific modes of interest.

3.7 Conclusion

We propose the decoder decomposition to help practitioners design and interpret the latent space of nonlinear autoencoders. In this chapter, 'interpretation of latent variables' refers to the identification of statistical correlations between the latent variables and the POD modes of the data. Physical interpretation of the latent variables may then be inferred from the physical interpre-

tation of the POD modes with which they are strongly correlated. The analysis presented here focuses on identifying correlations, rather than providing a definite physical interpretation of the latent variables. The decoder decomposition is a post-processing method which ranks the latent variables of an autoencoder. Two types of autoencoders are considered: the standard autoencoders (AE) with one decoder and one encoder, and the mode-decomposing autoencoders (MD-AEs) with one decoder per latent variable. First, we apply the decoder decomposition to analyse the latent space of the two-dimensional unsteady laminar cylinder wake. Both AEs and MD-AEs are applied to the laminar wake and their results are compared. By analysing different dimensions of the latent space, we show that the dimension of the latent space significantly impacts the physical meanings of the latent variables even if the reconstruction errors remain unaffected. Second, we apply the decoder decomposition to AEs trained with wind-tunnel experimental data of a three-dimensional turbulent wake past a bluff body. We show that increasing the size of the decoder has a similar effect to increasing the number of latent variables when the number of latent variables is small. Third, we apply the decoder decomposition to rank and select the latent variables that are associated with the vortex shedding structures. We apply the average rate of change to rank the latent variables based on their contribution to the data modes 1 and 2, which correspond to vortex shedding. Finally, we filter the latent space to minimise the effect of the unwanted data modes in the AE output. The output contains mainly the two coherent structures associated with vortex shedding, which verifies the method. The decoder decomposition is a simple yet robust post-processing method for ranking and selecting latent variables based on the flow physics and coherent structures of interest.

In retrospect, the analysis presented in this chapter could be further improved by introducing area-based weightings when training autoencoders using the experimental dataset, thereby ensuring consistency in the latent representation and the weighted POD employed in this chapter.

Chapter 4

Reconstructing 2D turbulent flows from sparse measurements

In Chapter 3, we worked on ML-assisted data analysis for the nonlinear compression of large data. However, in many practical applications, data is often incomplete. We now investigate methods to recover the missing flow field, which is necessary before ML-assisted data analysis can occur.

In this chapter, we develop a physics-constrained convolutional neural network, which is a deterministic tool, to reconstruct the full flow field from incomplete data. We explore three loss functions, both from machine learning literature and newly proposed: (i) the softly-constrained loss, which allows the prediction to take any value; (ii) the snapshot-enforced loss, which constrains the prediction at the sensor locations; and (iii) the mean-enforced loss, which constrains the mean of the prediction at the sensor locations. The proposed methods do not require the full flow field during training, making it suitable for reconstruction from incomplete data. We apply the method to reconstruct a laminar wake of a bluff body and a 2D turbulent Kolmogorov flow. First, we assume that measurements are not noisy and reconstruct both the laminar wake and the Kolmogorov flow from sensors located at fewer than 1% of all grid points. The snapshot-enforced loss reduces the reconstruction error of the Kolmogorov flow by ~ 25% compared to the softly-constrained loss. Second, we assume that measurements are noisy and propose the mean-enforced loss to reconstruct the laminar wake and the Kolmogorov flow at three different signal-to-noise ratios. We find that, across the ratios tested, the loss functions with harder constraints are more robust to both the random initialisation of the networks and the noise levels in the measurements. At high noise levels, the mean-enforced loss can recover the instantaneous snapshots accurately, making it a suitable choice when reconstructing flows from data corrupted with an unknown amount of noise. The proposed method opens opportunities for physical flow reconstruction from sparse, noisy data.

This chapter is adapted from Mo & Magri (2025), published in *Physical Review Fluids*. The relevant codes are available at
https://github.com/MagriLab/FlowReconstructionFromExperiment/tree/paper/flow-reconstruction-2d.
The triangular body wake dataset is available at https://doi.org/10.5281/zenodo.15623774.
The Kolmogorov flow dataset is available upon request.

4.1 Introduction

In most physics-constrained neural networks, the loss function to be minimised in training is a combination of data loss and physics loss, taking the form

$$\text{Loss}_{network} = \lambda_{data}\text{Loss}_{data} + \lambda_{equation}\text{Loss}_{equation},$$

where λ is the coefficient, the data loss, Loss_{data}, is a function of only prediction and observations and physics loss, $\text{Loss}_{equation}$, is a function of the prediction derived from the governing equation (Fukami et al. 2023). The coefficients are hyperparameters, which need to be tuned. By introducing the governing equations as a penalty term in the loss function, we place a soft constraint on the physics of the prediction. Although their excellent capabilities, soft physics constraints may produce trivial solutions (Wong et al. 2022). Since the importance of the physics depends on the coefficients, the physics loss is "competing" with the data loss. Sometimes networks learn a trivial solution, which satisfies the governing equations without satisfying the data loss, e.g., flows where the velocity is constant everywhere. Trivial solutions occur because a physics loss of 0 with a larger data loss gives a lower overall loss than a small physics and a small data loss.

The snapshot-enforced loss, proposed by Gao et al. (2021), places a harder constraint on the data loss. The idea of the strictly enforced data loss is to force the prediction to be equal to the observed values at observed locations so that only the physics loss is minimised during training. Gao et al. (2021) successfully reconstructed steady flows using the snapshot-enforced loss. However, the snapshot-enforced loss is not robust to noisy observations.

In this chapter, we reconstruct the unsteady wake of a triangular cylinder and a 2D turbulent Kolmogorov flow from both clean and noisy sparse measurements to infer the entire flow field. Kolmogorov flow and circular cylinder wakes are common test cases in flow reconstruction literature (Erichson et al. 2020, Kelshaw et al. 2022, Maulik, Romit et al. 2020), with the wake of a triangular cylinder being important in flame holders in combustion chamber design (Agrawal et al. 2013, Ghani et al. 2015). First, we investigate the effect of hard and soft constraints placed on the instantaneous measurements in the loss functions on flow reconstruction with a physics-constrained neural network. Then, we propose a new loss function for reconstructing flows from noisy measurements. Finally, we compare the loss functions at different signal-to-noise ratios.

In restrospect, we also investigated the effect of periodic boundary conditions on reconstructing and post-processing the 2D Kolmogorov flow with an additional testcase.

4.2 Data and preprocessing

This section presents the unsteady laminar wake of a triangular body and the 2D turbulent Kolmogorov flow we reconstruct in this chapter.

4.2.1 Laminar unsteady wake

The laminar dataset is the unsteady laminar wake of a 2D triangular body at $Re = 100$, generated with methods detailed in Chapter 2, Section 2.1.2. The computational domain has size $L_1 = 12$ and $L_2 = 4$ in streamwise and wall-normal directions, with 513 and 129 uniformly spaced grid points in each direction, respectively. The centre of the bottom edge of the equilateral triangle is placed at (3.0,2.0) oriented such that the base is perpendicular to the streamwise velocity. The time step is $\Delta t^* = 0.0002$. Only data in the brown box of figure 4.1 is considered by the neural networks for computational efficiency, capturing the near wake and the parallel vortex shedding. For constructing the final dataset, a snapshot of a flow (area within the brown box) contains the grid points on the plane at $x_3^* = 0$ bounded by $x_1^* = 3.0$, $x_1^* = 8.86$, $x_2^* = -2$ and $x_2^* = 2$. For simplicity, we define the coordinates of the dataset to be $\boldsymbol{x} = [x_1^* - 3.0, x_2^*]^T$, shown in figure 4.1 in brown.

The snapshots are organised into snapshot matrices $\boldsymbol{U} \in \mathbb{R}^{N_t \times N_1 \times N_2 \times N_u}$ and $\boldsymbol{P} \in \mathbb{R}^{N_t \times N_1 \times N_2 \times 1}$ for velocity and pressure, where $N_t = 600$ is the number of snapshots, $N_1 = 250$ is the number of grid points in the x_1 direction and $N_2 = 129$ is the number of grid points in the wall-normal direction. The dataset $\boldsymbol{D}^T = [\boldsymbol{U}^T, \boldsymbol{P}^T]$ is the snapshot matrix of the full state. Each full state vector is $\boldsymbol{D}(\boldsymbol{x}, t)^T = [\boldsymbol{u}^T, p]$, where $\boldsymbol{u}^T = [u_1(\boldsymbol{x}, t), u_2(\boldsymbol{x}, t)]$ is the 2D velocity vector and $p = p(\boldsymbol{x}, t)$ is the pressure.

4.2.2 2D Turbulent Kolmogorov flow

The 2D Kolmogorov flows are generated at $Re = 34$, with 32 wavenumbers, the forcing frequency $k = 4$ and the time step $\Delta t^* = 0.01$, following the methods detailed in Chapter 2, Section 2.1.2. The resulting dataset is a weakly turbulent flow (Racca 2023). We use the word 'turbulent' in this thesis to keep the terminology consistent. In reality, the 2D 'turbulent' Kolmogorov flow dataset presented here is chaotic, sensitive to initial conditions, but lacks the broadband energy spectrum of a true turbulent flow (Platt et al. 1991) and is always effectively differentiable. A snapshot is saved every 10 Δt^*, resulting in $\Delta t = 0.1$ for the datasets. Two datasets are generated by using different initial random seeds — one for the grid sensitivity study in Section A.2.1 and one for the results in Section 4.5.1 and 4.5.2. Each Kolmogorov dataset, $\boldsymbol{D} = [\boldsymbol{U}^T, \boldsymbol{P}^T]^T \in \mathbb{R}^{N_t \times N \times N \times (N_u+1)}$, is the snapshot matrix of the full state with velocity and pressure, where $N_t = 6000$, $N = 128$ and $N_u = 2$. Figure 4.2 shows an overview of a Kolmogorov flow dataset. The dissipation rate is

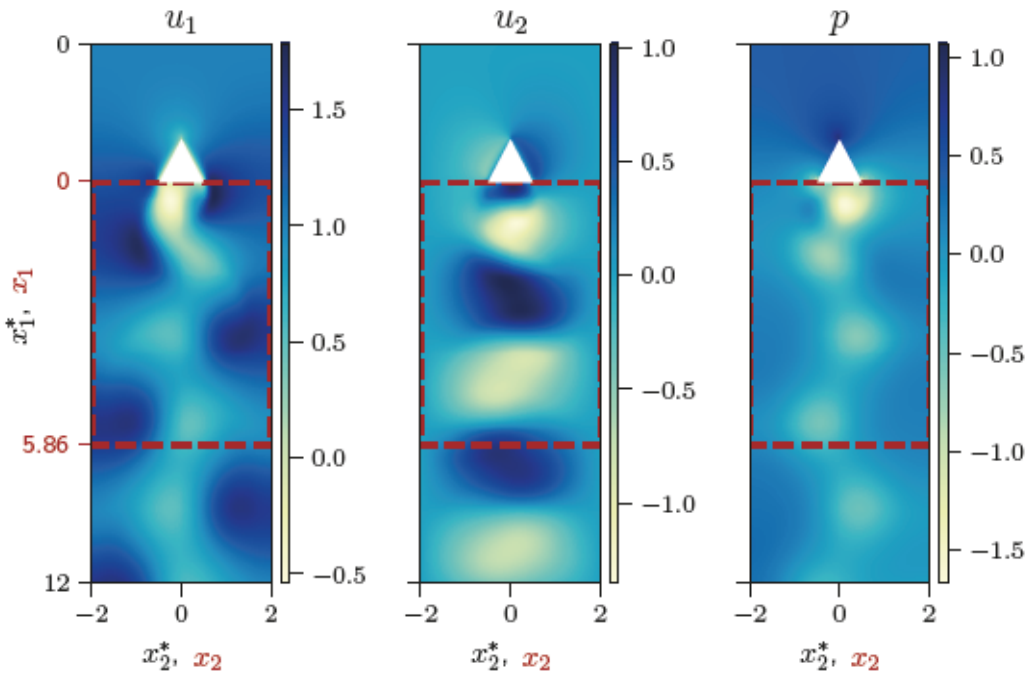


Figure 4.1: Snapshots of streamwise (left) and wall-normal (centre) velocity and pressure (right) at $t = 2.5$, where $t = 0$ is the end of the transient period discarded when generating the dataset. The area indicated by the brown box is the area considered by the neural networks, with its coordinates $\boldsymbol{x} = [x_1^* - 3.0, x_2^*]^T$ shown in brown.

computed with Equation (2.8). The forced wavenumber $k = 4$ shows as a peak in the turbulent kinetic energy spectrum in Figure 4.2. The integral length scale is approximately 0.4π, computed as the integral of the two-point correlation functions of the velocities.

4.3 Methodology

In this section, we first present how the measurements used in the reconstruction are selected from our simulated datasets (Section 4.3.1). We then present the physics-constrained dual-branch convolutional neural network (Section 4.3.2). Finally, we present the different formulations of loss functions we will investigate (Section 4.3.3).

4.3.1 Taking measurements

From each dataset, we select a set of sparse measurements from which we will reconstruct the flows. This set of sparse measurements is the only data-related information available for the network in training, i.e., no high-resolution data is needed in the methods that we propose. Measurements of all velocity components and pressure are taken at sensor locations $\boldsymbol{x}_s$ (general sensors). Additional pressure measurements are taken at input locations $\boldsymbol{x}_{in}$. The separate input pressure measurements simulate a common situation in bluff body wake experiments (e.g., Rowan D. Brackston 2017). For the bluff body wake, $\boldsymbol{x}_{in}$ is selected to be immediately behind the

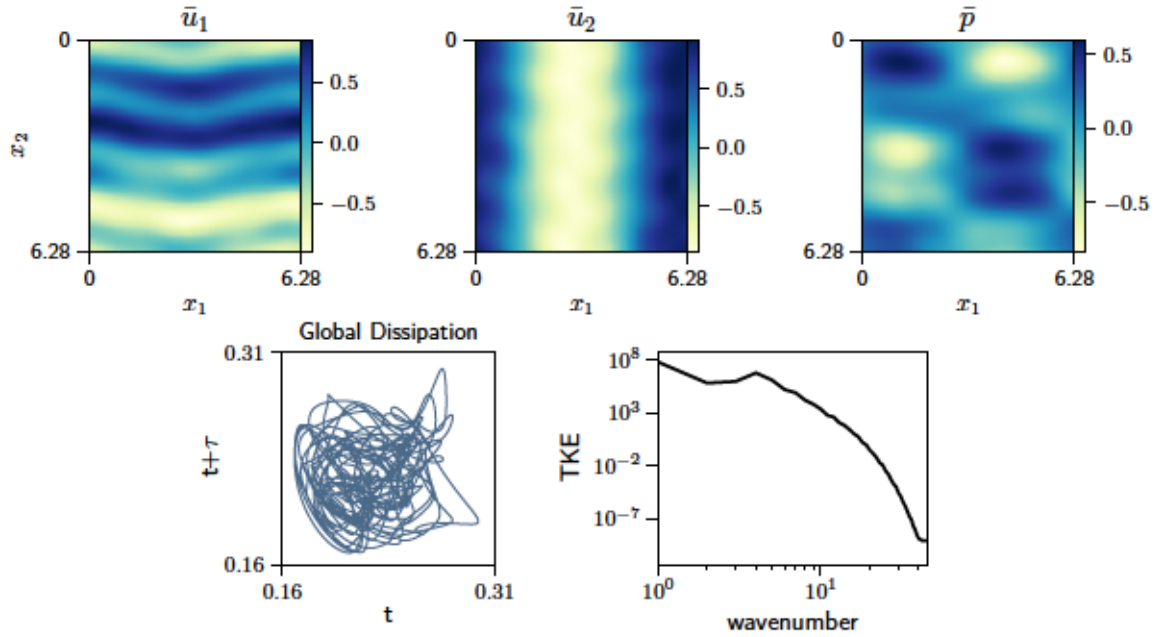


Figure 4.2: Top row, left to right: The time-averaged velocity in x_1 direction (u_1), x_2 direction (u_2) and pressure (p). Bottom row, left to right: The global dissipation with a time delay of $\tau \approx 5$ and the turbulent kinetic energy spectrum.

triangular body; for the Kolmogorov flow, $\boldsymbol{x}_{in}$ is randomly selected. The set of measurements $\xi(\boldsymbol{D})$ is a collection of the input pressure, and the velocities and pressure measured at the sensor locations, $\xi(\boldsymbol{D}) = \{\boldsymbol{U}(\boldsymbol{x}_s), \boldsymbol{P}(\boldsymbol{x}_s), \boldsymbol{P}(\boldsymbol{x}_{in})\}$. The measurements are taken at different locations for different tests. The exact locations will be shown together with the test results in Section 4.4 and Section 4.5.

4.3.2 Physics-constrained dual-branch convolutional neural network

We develop a physics-constrained dual-branch convolutional neural network (PC-DualConvNet) for the reconstructions, inspired by (Kelshaw et al. 2022, Shokar et al. 2024, Ronneberger et al. 2015, Li et al. 2021). The advantage of the PC-DualConvNet is that it does not require the full state to be known at all grid points during training, and that the measurements do not need to be in a regular grid. The goal of the network is to infer unseen spatial information for a given set of measurements.

PC-DualConvNet aims to produce the flow field $\hat{\boldsymbol{D}}$ to approximate the data $\boldsymbol{D}$ from pressure input $\boldsymbol{P}_{in}$, using sparse measurements $\xi(\boldsymbol{D})$ as collocation points. PC-DualConvNet consists of a U-Net style (Ronneberger et al. 2015) convolutional branch, which is common in image processing (lower branch in Figure 4.3), and an optional Fourier branch (top branch in Figure 4.3). Inspired by Shokar et al. (2024) and Sardar et al. (2024), the Fourier transform layer is introduced to help convolutions capture global information. For each snapshot, Fourier Transform is applied over the spatial dimensions, which are all dimensions of a tensor except for the first (representing time) and last (representing convolution channels). The Fourier transform and the

inverse Fourier transform are taken at layers labelled FT and FT^{-1} in Figure 4.3, respectively. During training, we find that the Fourier Transform in the upper branch is particularly helpful in improving the robustness of the snapshot-enforced loss $\mathscr{L}^s$ (Section 4.3.3). The benefit of activating the Fourier layer varies with different datasets and is considered as a hyperparameter. Depending on the test case, as explained in the later sections, the top branch may perform Fourier Transform and inverse Fourier Transform immediately before and after the convolution layers. The output of the top and lower branches are stacked channel-wise before the final convolution layers, which produce the output $\hat{\boldsymbol{D}}$.

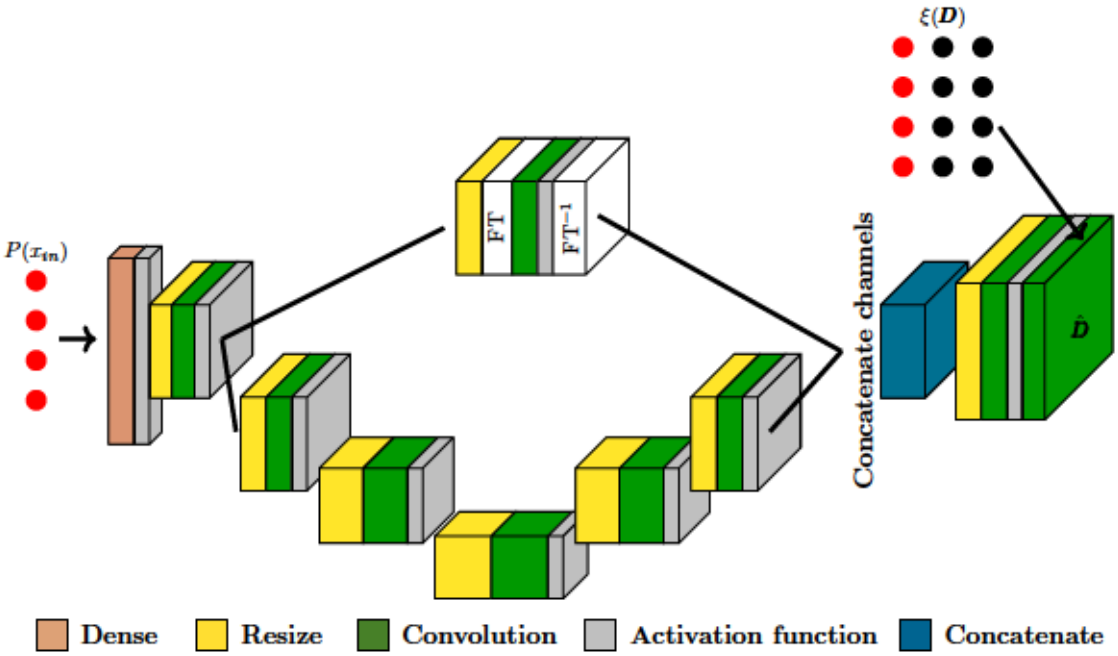


Figure 4.3: Schematic of the two-branch Fourier convolutional neural network. The number and size of layers are for illustration. The Fourier Transform layer (FT) and the inverse Fourier Transform layer (FT^{-1}) are optional and are activated (or deactivated) for different loss functions.

The PC-DualConvNet's parameters $\boldsymbol{w}$ are the (local) minimisers of a loss function $\mathscr{L}$, which measures the error between $\hat{\boldsymbol{D}}$ and $\boldsymbol{D}$. The optimisation process is

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathscr{L}(\boldsymbol{P}(\boldsymbol{x}_{in}); \boldsymbol{w}), \tag{4.1}$$

in which the design of the loss formulation $\mathscr{L}$ is a goal of this chapter and will be discussed in detail in Section 4.3.3.

4.3.3 Loss functions

The loss $\mathscr{L}$ is the quantity to be minimised when training a neural network (Equation (4.1)). The governing equations are incorporated into the loss formulations for the physical reconstruction. We define the momentum and continuity loss terms of the dataset $\boldsymbol{D}$ to be the ℓ_2 norm of the

residuals of the continuity and momentum equations, as defined in Equation (2.1) at all grid points, as

$$\mathscr{L}_{div}(\boldsymbol{D}) = \|\mathscr{R}_d(\boldsymbol{U})\|_2^2, \tag{4.2}$$

$$\mathscr{L}_{mom}(\boldsymbol{D}) = \|\mathscr{R}_m(\boldsymbol{U},\boldsymbol{P})\|_2^2. \tag{4.3}$$

To compute the residual of the momentum equations $\mathscr{R}_m$, we assume the external forcing mechanism and the Reynolds number are known. The derivatives are computed with second-order forward or backward finite difference schemes on the boundaries, and fourth-order central difference schemes at the interior points. The finite difference approach to computing the equation-based loss on a discretised grid was also taken by Karnakov et al. (2023). However, they solved directly for the flow fields using the gradient of the loss with respect to the missing values, whereas we try to approximate a mapping from measurements to the missing values with a neural network. A data-related loss is also needed for the output to minimise the error with the measurements selected from the reference datasets. We define the sensor loss to be the ℓ_2 norm of the difference of the output $\hat{\boldsymbol{D}}$ and the reference data $\boldsymbol{D}$ at observed locations, as

$$\mathscr{L}_o(\hat{\boldsymbol{D}},\boldsymbol{D}) = \|\xi(\hat{\boldsymbol{D}}) - \xi(\boldsymbol{D})\|_2^2. \tag{4.4}$$

Softly-constrained loss

A widely applied method to include physics into the network is to define the loss function to be the weighted sum of a sensor loss and a physics loss, applied in various super-resolution convolutional neural networks and physics-informed neural networks (Raissi et al. 2020, Kelshaw et al. 2022, Raissi et al. 2019, Fukami et al. 2023). We define the softly-constrained loss $\mathscr{L}^c$ to be

$$\mathscr{L}^c(\boldsymbol{P}_{in};\boldsymbol{w}) = \lambda_o\mathscr{L}_o(\hat{\boldsymbol{D}},\boldsymbol{D}) + \lambda_{div}\mathscr{L}_{div}(\hat{\boldsymbol{D}}) + \lambda_{mom}\mathscr{L}_{mom}(\hat{\boldsymbol{D}}), \tag{4.5}$$

where λ denotes the non-negative regularisation factor. Both the physics of the output and the values of the output at the sensor locations are included in the loss function as regularisation terms, which means both the sensor loss and the physics-based losses will be minimised, but not guaranteed to be a specific value, hence its name 'softly-constrained'.

The components of the softly-constrained loss can be grouped into the sensor loss $\mathscr{L}_o$ and the physics loss $\mathscr{L}_p$ which is the combined continuity and momentum loss $\mathscr{L}_{div} + \mathscr{L}_{mom}$. A trivial solution is a solution which has $\mathscr{L}_p = 0$ but a large $\mathscr{L}_o$, meaning that the network finds another solution to the governing equation that is different from the reference datasets. An example of a trivial solution is a flow field that is constant everywhere. Trivial solutions may develop when both $\mathscr{L}_p$ and $\mathscr{L}_o$ cannot, theoretically, be 0 at the same time, which can be caused by noise, either experimental or numerical. When reconstructing a synthetic dataset, the source of the noise could come from the use of different numerical schemes when estimating the derivatives.

For example, the residuals in $\mathscr{L}_p$ for the reconstructed laminar dataset, defined in (4.3), are computed with a mix of second- and fourth-order finite differences. However, the laminar dataset is generated with a sixth-order scheme (see Section 2.1.2). Therefore, if the measurements at sensor locations from the reference and the reconstructed datasets are the same, meaning $\mathscr{L}_o(\hat{\boldsymbol{D}}, \boldsymbol{D}) = 0$, and the reference dataset has $\mathscr{L}_p(\hat{D})$ when using the sixth-order scheme, then $\mathscr{L}_p(\hat{\boldsymbol{D}})$ cannot be zero when using the mix of second- and fourth-order scheme. The possible scenarios are summarised in Table 4.1. Scenario 1 is not possible when different numerical schemes are used in data generation and training, or when the reference data is collected from experiments. For scenarios 2 to 4, the sensor and physics losses are in conflict with each other, meaning that they both have a theoretical minimum of 0 if considered independently, but their sum (the total loss value) cannot reach 0. Depending on the sensor placements, scenario 2 may have a theoretical sensor loss close to 0 (when using a few sensors) or much larger than 0 (large numbers of sensors or good sensor locations). Scenarios 3 and 4 are acceptable reconstructions, but their theoretical minimum total losses are also not 0. Therefore, the network may produce trivial solutions instead of the correct reconstructed flows. In other words, given that the ideal scenario 1 is unlikely in practice, we normally want to be in scenarios 3 or 4, but they have the same non-zero theoretical minimum total loss as the unwanted scenario 2.

Table 4.1: Possible solutions and their corresponding theoretical loss values when training with the softly-constrained loss. The network needs to minimise both physics loss and sensor loss. Scenario 1 has a theoretical minimum loss of 0 and is ideal, but is also unlikely when different numerical schemes are used in data generation and loss computation, or if data is experimental. Scenario 2 is the trivial solution, where the outputs satisfy the governing equations but not the measurements. Depending on the sensor placements, the minimum $\mathscr{L}_o$ could be either approximately 0 or much larger than 0. Scenarios 3 and 4 are desired, but they also have a theoretical minimum total loss larger than 0.

	Total loss value	Sensor loss	Physics loss	Numerical scheme	Reconstructed flow
1	0	0	0	same	$\hat{\boldsymbol{D}} = \boldsymbol{D}$
2	>0 or ≈0	>0 or ≈0	0	different	trivial solution
3	≈0	0	≈0	different	$\hat{\boldsymbol{D}} = \boldsymbol{D}$
4	≈0	≈0	≈0	different	$\hat{\boldsymbol{D}} \approx \boldsymbol{D}$

In principle, by carefully tuning the ratio between λ_o, λ_{div} and λ_{mom} in Eq. (4.5), we can penalise the occurrence of trivial solutions. In practice, good sets of hyperparameters become difficult to find as the measurements become noisier or more sparse, and trivial solutions are not effectively prevented, as shown in Section 4.5.3.

Snapshot-enforced loss

To eliminate trivial solutions and improve the training, we strictly enforced the measurements and minimise physics loss. The snapshot-enforced loss for steady flows was proposed by (Gao et al. 2021). The snapshot-enforced loss minimises only the physics loss while the sensor loss is enforced to be 0, providing a harder constraint on the predicted flow at sensor locations compared to the softly-constrained By enforcing a harder constraint on the measurements and minimising only the physics loss, we eliminate trivial solutions. In this thesis, we define the snapshot-enforced loss $\mathscr{L}^s$ as

$$\mathscr{L}^s = \lambda_{div}\mathscr{L}_{div}(\mathbf{\Phi}) + \lambda_{mom}\mathscr{L}_{mom}(\mathbf{\Phi}), \tag{4.6}$$

where $\mathbf{\Phi}^T = [\mathbf{\Phi}_u^T, \mathbf{\Phi}_p^T]$ is defined as

$$\mathbf{\Phi_u}(x) = \begin{cases} \boldsymbol{U}(x) & \text{where } x \in \boldsymbol{x}_s, \\ \hat{\boldsymbol{U}}(x) & \text{otherwise.} \end{cases} \qquad \mathbf{\Phi_p}(x) = \begin{cases} \boldsymbol{P}(x) & \text{where } x \in \{\boldsymbol{x}_s, \boldsymbol{x}_{in}\}, \\ \hat{\boldsymbol{P}}(x) & \text{otherwise.} \end{cases} \tag{4.7}$$

Practically, we start with the reconstructed flow field, $[\hat{\boldsymbol{U}}, \hat{\boldsymbol{P}}]$, then replace the reconstructed velocities and pressure at the sensor locations with sensor measurements to obtain the processed reconstructed flow field, $\mathbf{\Phi}$. In regions without sensors, we keep the reconstructed values. Finally, we compute the physics loss of $\mathbf{\Phi}$, which is minimised during training. In short, the network is not allowed to modify the output values at the sensor locations. Figure 4.4 is a graphical representation of training a PC-DualConvNet with the snapshot-enforced loss $\mathscr{L}^s$.

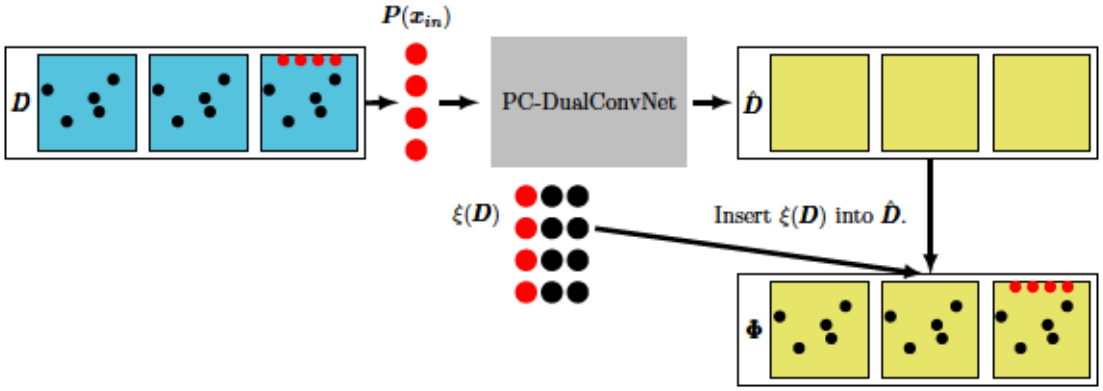


Figure 4.4: Computing the snapshot-enforced loss, which involves enforcing measurements $\xi(\boldsymbol{D})$ into $\hat{\boldsymbol{D}}$ to obtain $\mathbf{\Phi}$. The snapshot-enforced loss is the physics loss of $\mathbf{\Phi}$.

The snapshot-enforced loss may struggle with noisy measurements. Because the noisy measurements are enforced at the sensors' locations, the network's outputs and gradients at the sensors' locations are not updated by training. As a result, regularizing the loss function is not effective in mitigating the impact of noise at the sensors' locations.

Algorithm 2: Computing the snapshot-enforced loss

Input : $\hat{\boldsymbol{D}}$ - the output of the PC-DualConvNet
$\boldsymbol{x}_s, \boldsymbol{x}_{in}$ - the set of grid indexes where the sensors are located
$\xi(\boldsymbol{D}) = \{\boldsymbol{D}(\boldsymbol{x}_s), \boldsymbol{P}(\boldsymbol{x}_{in})\}$ - the measurements
$\lambda_{div}, \lambda_{mom}$ - regularisation factors

Output : $\mathscr{L}^s$ - the strictly-enforced loss

$\boldsymbol{\Phi} \leftarrow \hat{\boldsymbol{D}}$ // start with the output of the network

$\boldsymbol{\Phi}(\boldsymbol{x}_s) \leftarrow \boldsymbol{D}(\boldsymbol{x}_s)$ // insert the velocities and pressure at general sensor locations $\boldsymbol{x}_s$ with the measured

$\boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{x}_{in}) \leftarrow \boldsymbol{P}(\boldsymbol{x}_{in})$ // insert the pressure at input sensor locations $\boldsymbol{x}_{in}$ with the measured pressure inputs

return $\lambda_{div}\mathscr{L}_{div}(\boldsymbol{\Phi}) + \lambda_{mom}\mathscr{L}_{mom}(\boldsymbol{\Phi})$

Mean-enforced loss

The snapshot-enforced loss $\mathscr{L}^s$ ensures the reconstructed flow fits the measurements, but it also enforces the noise on the measurements if the measurements are noisy. (Gao et al. 2021) suggest using the softly-constrained loss for noisy measurements. To deal with noisy measurements, we propose the mean-enforced loss. The mean-enforced loss $\mathscr{L}^m$ combines the advantages of the softly-constrained and the snapshot-enforced loss, defined as

$$\mathscr{L}^m = \lambda_o \mathscr{L}_o(\hat{\boldsymbol{D}}, \boldsymbol{D}) + \lambda_{div}\mathscr{L}_{div}(\boldsymbol{\Phi}) + \lambda_{mom}\mathscr{L}_{mom}(\boldsymbol{\Phi}), \tag{4.8}$$

where $\boldsymbol{\Phi}^T = [\boldsymbol{\Phi}_u^T, \boldsymbol{\Phi}_p^T]$ is

$$\boldsymbol{\Phi}_{\boldsymbol{u}}(\boldsymbol{x}) = \begin{cases} \overline{\boldsymbol{U}}(\boldsymbol{x}) + \hat{\boldsymbol{U}}'(\boldsymbol{x}) \text{ where } \boldsymbol{x} \in \boldsymbol{x}_s, \\ \hat{\boldsymbol{U}} \text{ otherwise.} \end{cases} \qquad \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{x}) = \begin{cases} \overline{\boldsymbol{P}}(\boldsymbol{x}) + \hat{\boldsymbol{P}}'(\boldsymbol{x}) \text{ where } \boldsymbol{x} \in \{\boldsymbol{x}_s, \boldsymbol{x}_{in}\}, \\ \hat{\boldsymbol{P}} \text{ otherwise.} \end{cases} \tag{4.9}$$

The symbols $\overline{*}$ and $*'$ denote the time-averaged and fluctuating components of any variable $*$, respectively. In the mean-enforced loss, we split the reconstructed flow into mean and fluctuating components at the sensors' locations to obtain $\boldsymbol{\Phi}$. In regions without sensors, we keep the reconstructed values, whereas at the sensors' locations, we keep the fluctuating components of the reconstructed values whilst enforcing the mean of the sensors' measurements. Hence, we compute the physics loss of $\boldsymbol{\Phi}$ and the sensor loss $\mathscr{L}_o(\hat{\boldsymbol{D}}, \boldsymbol{D})$, which are both minimised during training. In short, the network can modify the fluctuating components of the output values at the sensor locations, but not the mean. A pseudoalgorithm is shown in Algorithm 3. Figure 4.5 is a graphical representation of $\mathscr{L}^m$. By changing the ratios of the coefficients λ, we can adjust how much smoothing to apply when reconstructing from noisy observation. The smaller the λ_o compared to λ_{div} and λ_{mom}, the stronger the smoothing.

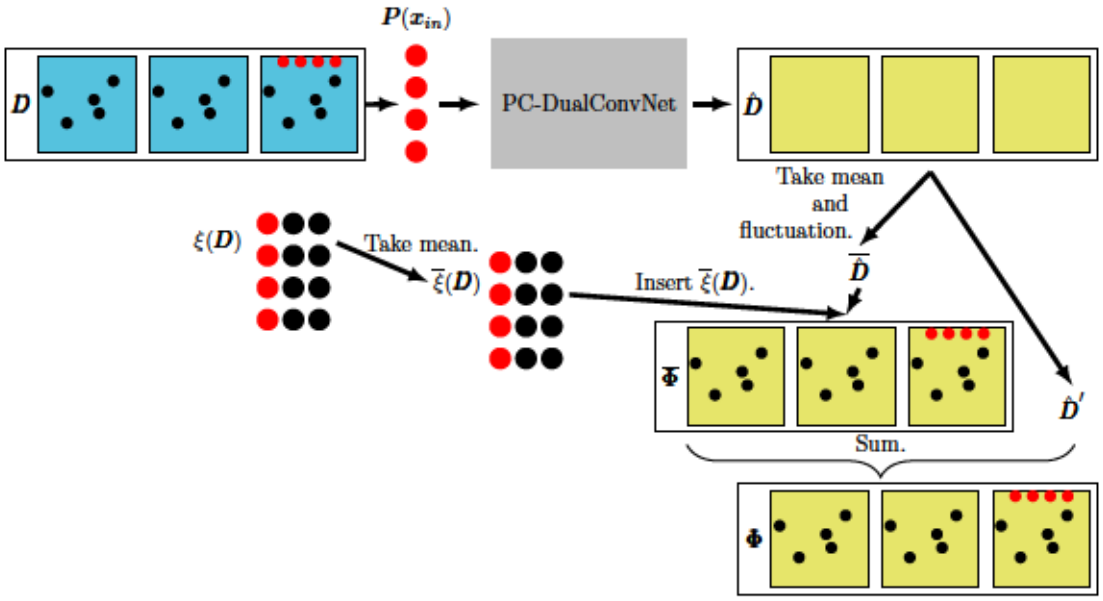


Figure 4.5: Computing the mean-enforced loss. The mean is taken over a batch of inputs during training, which involves enforcing the measurements' mean $\overline{\xi}(\boldsymbol{D})$ into $\overline{\hat{\boldsymbol{D}}}$, and adding the fluctuating field $\hat{\boldsymbol{D}}'$ to obtain $\boldsymbol{\Phi}$. The snapshot-enforced loss is the weighted sum of the physics loss of $\boldsymbol{\Phi}$ and the data loss $\mathscr{L}_o(\hat{\boldsymbol{D}}, \boldsymbol{D})$.

4.4 Reconstructing the laminar wake

In this section, we reconstruct the laminar wake from both non-noisy measurements and noisy measurements. We first reconstruct the laminar wake from 18 sparse sensors placed in the domain in Section 4.4.1. We then compare the three loss functions introduced in Section 4.3.3 in reconstructing the wake from sparse measurements contaminated with white noise.

4.4.1 Reconstructing the laminar wake from non-noisy measurements

For all reconstructions of the laminar wake, pressure sensors are placed on all grid points immediately behind the bluff body (31 in total) at $\boldsymbol{x}_{in}$. These pressure measurements $\boldsymbol{P}(\boldsymbol{x}_{in})$ are used as the input to the network (red dots in Figure 4.6). When reconstructing from non-noisy measurements, 12 general sensors (black dots in Figure 4.6) are placed at the maximum and minimum of the first two leading POD modes of the pressure and velocity components (Cohen et al. 2003). A further 6 general sensors are placed randomly near the edge of the domain. The sensor locations $\boldsymbol{x}_s$ account for only $\sim 0.06\%$ of all grid points. The POD-guided sensor placement is a data-driven sensor placement method which relies on prior knowledge of the flow field, specifically, the leading POD modes. In addition to the 12 sensors placed at the maxima/minima of leading POD modes, we find that the 6 sensors near the boundaries are essential for the network to find the correct solution to the Navier-Stokes equation. Without the sensors placed near the boundary in the far wake, we have observed reconstructed flow that differs from the ground truth in the far wake but has a small physics loss. Without prior knowledge, comparable recon-

Algorithm 3: Computing the mean-enforced loss

Input : $\hat{\boldsymbol{D}}$ - the output of the PC-DualConvNet
$\boldsymbol{x}_s, \boldsymbol{x}_{in}$ - the set of grid indexes where the sensors are located
$\xi(\boldsymbol{D}) = \{\boldsymbol{D}(\boldsymbol{x}_s), \boldsymbol{P}(\boldsymbol{x}_{in})\}$ - the measurements
$\lambda_o, \lambda_{div}, \lambda_{mom}$ - regularisation factors

Output : $\mathscr{L}^m$ - the mean-enforced loss

$\boldsymbol{\Phi} \leftarrow \hat{\boldsymbol{D}}$ // start with the output of the network

$\overline{*}$ are the time-averaged components over a training batch, $*'$ are the fluctuating components

$\boldsymbol{\Phi}(\boldsymbol{x}_s) \leftarrow \overline{\boldsymbol{D}}(\boldsymbol{x}_s) + \hat{\boldsymbol{D}}'(\boldsymbol{x}_s)$ // Insert the time-averaged components of the measurements

$\boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{x}_{in}) \leftarrow \overline{\boldsymbol{P}}(\boldsymbol{x}_{in}) + \hat{\boldsymbol{D}}_{\boldsymbol{p}}'(\boldsymbol{x}_{in})$

return $\lambda_o \mathscr{L}_o\left(\xi(\hat{\boldsymbol{D}}), \xi(\boldsymbol{D})\right) + \lambda_{div}\mathscr{L}_{div}(\boldsymbol{\Phi}) + \lambda_{mom}\mathscr{L}_{mom}(\boldsymbol{\Phi})$

struction can be achieved by placing 32 sensors in a uniform grid covering the entire domain. This difference in the number of sensors shows that prior knowledge of the flow fields can help reduce the sensor requirement. Although the difference is 0.04% of total grid points in the case presented in this section (the number of sensors was reduced from 32 to 18).

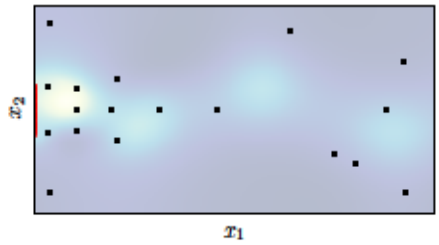


Figure 4.6: Sensor placement for laminar wake, non-noisy measurements. The measurements consist of the pressure inlet measurement at $\boldsymbol{x}_{in}$ (red) and measurements of all components at $\boldsymbol{x}_s$ (black).

The softly-constrained loss $\mathscr{L}^c$ and the snapshot-enforced loss $\mathscr{L}^s$ are tested for their ability to reconstruct the wake. For each loss function, a hyperparameter optimisation is carried out[1] and the hyperparameters that minimise total loss $\mathscr{L}_p + \mathscr{L}_o$ are selected. (There is a significant difference in the reconstruction results between $\mathscr{L}^c$ and $\mathscr{L}^s$ when applied to reconstruct the laminar wake from non-noisy measurements.)

The model with the lowest total loss in this case uses $\mathscr{L}^c$ and is used to generate the results in this section (see Table A.7 for the network structures and hyperparameters). A snapshot of the reconstructed flow is compared with the reference and interpolated snapshot in Figure 4.7a, which

[1] Using Bayesian hyperparameter optimisation provided by Weights and Biases (http://wandb.ai/).

shows that the reconstructed flow resembles closely to the reference data and has a smaller error. Vorticity $\boldsymbol{v}$ is the cross product of the velocity $\nabla \times \boldsymbol{u}$. The probability density of the fluctuating components is shown in Figure 4.7b. The KL divergence of the reconstructed u_1', u_2' and p' from the reference flow is 0.005, 0.014 and 0.005, respectively. In comparison, the KL divergence of the interpolated components is 1.880, 0.120 and 0.529, respectively. The KL divergence is computed with (2.2.2). The probability density of the fluctuating components of the reconstructed flow is a better approximation of the reference flow, compared to the interpolated flow.

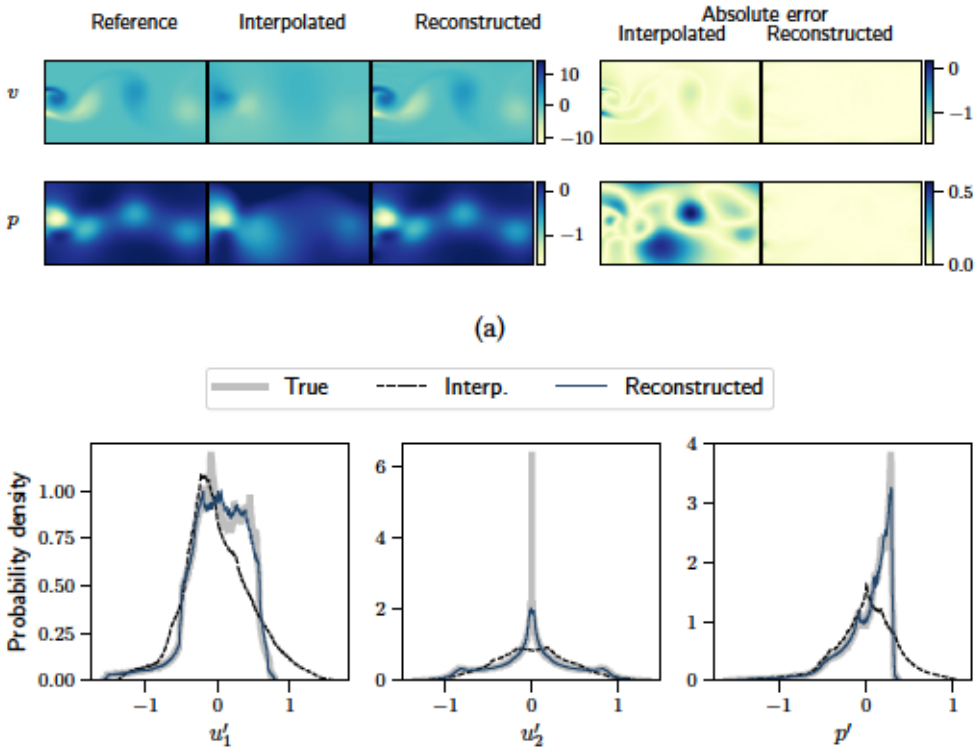


Figure 4.7: Reconstructed laminar wake from non-noisy measurements. (a) Snapshots of the reconstructed laminar wake compared with the reference and interpolated snapshot. The last two columns show the instantaneous absolute error of the interpolated snapshot and the reconstructed snapshot, respectively. (b) The probability density of the fluctuating components of the entire flow. Both (a) and (b) show that the reconstructed flow is a closer match with the reference data than mere interpolation.

Table 4.2 compares the relative error and the physics loss of the reconstructed flow with the reference data and the interpolated flow. Interpolation of the laminar wake is performed per snapshot using thin-plate spline to handle the very sparse and irregular data points. The physics loss of the reference data is non-zero because the physics loss in training is computed with a mix of second- and fourth-order numerical schemes when a different numerical scheme is used in data generation. Detailed explanations on the effect of using different numerical schemes are

Table 4.2: Summary of the losses for the reconstructed laminar wake from non-noisy measurements. Values are shown in the format *mean* ± *standard deviation* calculated over 5 runs initiated with different random weights.

	ϵ (%)	$\mathscr{L}_p$
Reference data	N/A	0.04338±0.00000
Interpolated	43.26±0.00	1.06721±0.00000
Reconstructed	3.34±0.18	0.00028±0.00002

given in Section 2.1.2, 4.2 and 4.3.3. The small errors numerically show that the network has successfully reconstructed the laminar wake from sparse measurements.

4.4.2 Reconstructing the laminar wake from noisy observations

For reconstructing the laminar wake from noisy measurements, 250 general sensors are randomly placed in the domain (Figure 4.8). The sensor locations $\boldsymbol{x}_s$ account for only ~ 0.8% of all grid points.

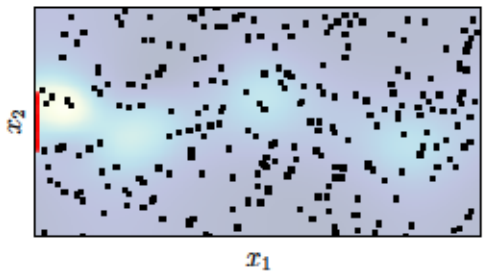


Figure 4.8: Sensor placement for laminar wake, noisy measurements. The measurements consist of the pressure inlet measurement at $\boldsymbol{x}_{in}$ (red) and measurements of all components at $\boldsymbol{x}_s$ (black).

The noise $\boldsymbol{e} \in \mathbb{R}^{N_u+1}$ for the full state $\boldsymbol{D}(\boldsymbol{x}, t)$ at each discrete grid point is generated from a Gaussian distribution $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}_e)$, where the components of $\boldsymbol{\sigma}_e$ correspond to the components of the flows $[\boldsymbol{u}, p]$. The noisy data is $\boldsymbol{D}_n(\boldsymbol{x}, t) = \boldsymbol{D}_n(\boldsymbol{x}, t) + \boldsymbol{e}$ ($\boldsymbol{e}$ is different at each discrete grid point); the noisy measurements are $\xi(\boldsymbol{D}_n)$; the inputs to the network is $\boldsymbol{P}_n(\boldsymbol{x}_{in})$. Three different *SNR*s are considered in this chapter, $SNR = 20, 10$ and 5.

For consistency of the results, the same random seed is used to generate the noise for all test cases shown in this section. The hyperparameters for the networks are shown in Table A.8 to A.10. Figure 4.9 summarises the effect of different loss functions on reconstructing the laminar wake from noisy, sparse measurements. At $SNR = 20$, all loss functions result in reconstruction errors under 10% (Figure 4.9 left panel), showing that all loss functions can lead to good reconstruction with small levels of noise. The reconstruction error of the softly-constrained loss reaches above

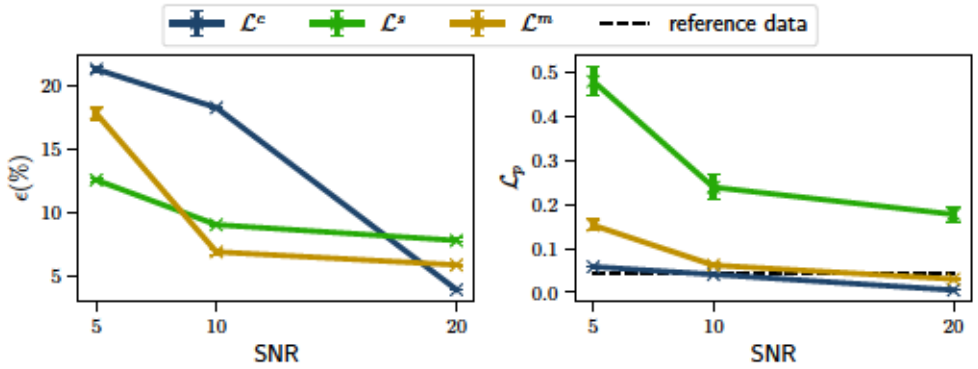
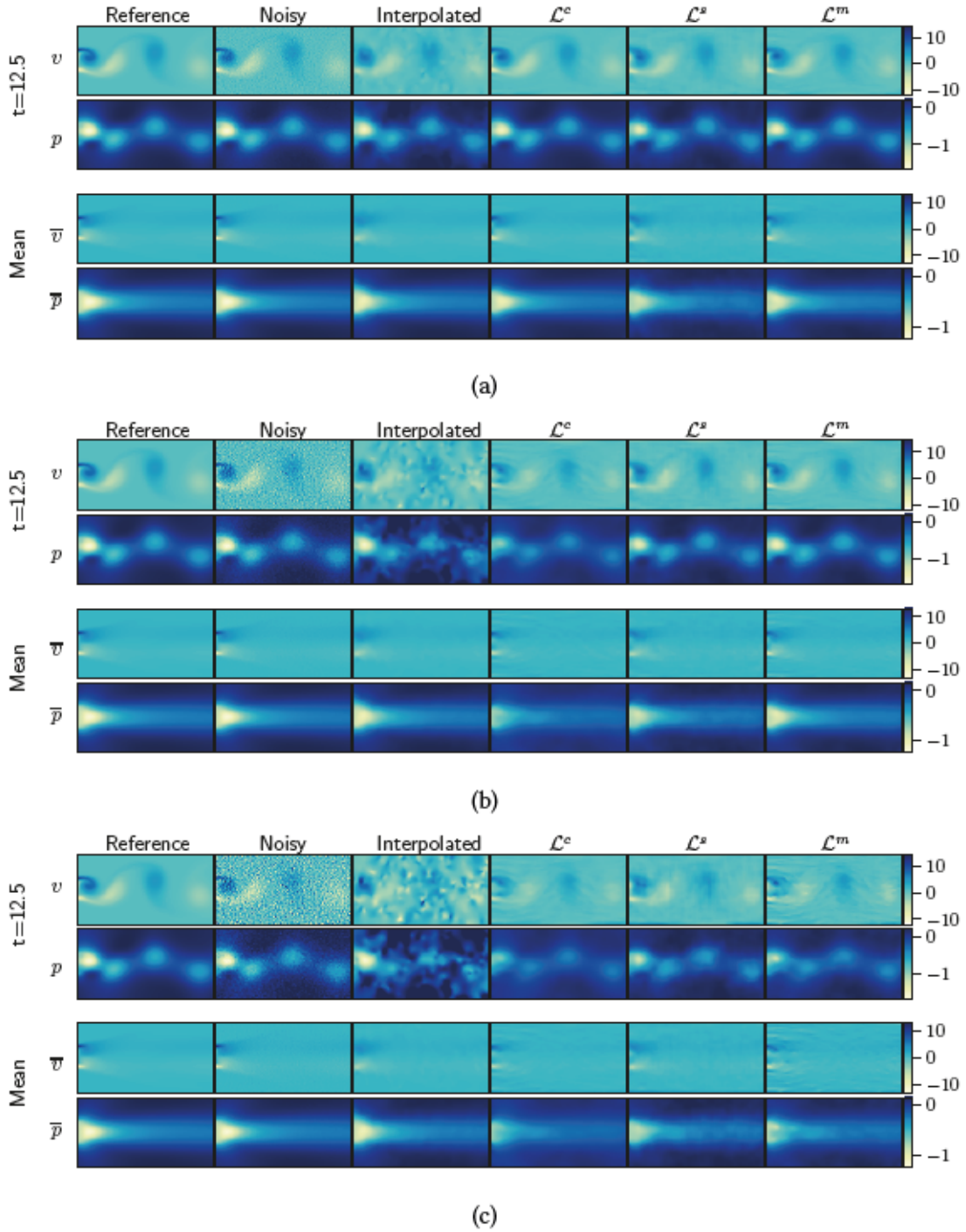


Figure 4.9: Comparison of loss functions for different SNR. The reference physics loss is the residual of the Navier-Stokes equations solved numerically. The mean (markers) and standard deviation (error bars) are calculated over five repeats with randomly initialised weights.

10% by $SNR = 10$ whilst the errors of the strictly enforced and the mean-enforced loss only reach above 10% at $SNR = 5$. The slower rate of increase of error as the measurements become noisier shows that losses with harder constraints are less sensitive to noise.

Figure 4.9 (right panel) shows a comparison of the physics loss $\mathscr{L}_p$ for the networks using the different loss functions. At any level of noise, $\mathscr{L}^s$ leads to the highest $\mathscr{L}_p$ amongst the three loss functions tested. $\mathscr{L}^m$ also has a higher physics loss than $\mathscr{L}^c$, but close to the reference physics loss. The harder constraint on the instantaneous measurements in $\mathscr{L}^s$ does not allow for de-noising, thus leading to a higher $\mathscr{L}_p$. Since $\mathscr{L}^m$ is specifically designed to avoid placing hard constraints on the instantaneous measurements, the impact of noise on the reconstruction error of $\mathscr{L}^m$ is smaller compared to $\mathscr{L}^s$. The effect of noise can be appreciated in Figure 4.10, in which snapshots of the instantaneous and the mean reconstructed wake are compared with the reference and the interpolated. At $SNR = 10$ (Figure 4.10b), the wake reconstructed with $\mathscr{L}^c$ shows the same flow features as the reference, but with a reduced variation of the flow field (the difference between the maximum and minimum value is smaller than the reference). A reduced variation of the predicted flow field is the first sign of a network breaking down. In comparison, the wake reconstructed with $\mathscr{L}^m$ and $\mathscr{L}^s$ at $SNR = 10$ and at $SNR = 20$ shows no visible difference. At both *SNR*s, the wake reconstructed with $\mathscr{L}^s$ shows visible noise in the snapshots, which is not present in the snapshots reconstructed with $\mathscr{L}^m$. $\mathscr{L}^m$ and $\mathscr{L}^s$ also start to show a reduced variation of the flow field at $SNR = 5$, even though the reduction is stronger for $\mathscr{L}^m$. At $SNR = 5$, the mean of the interpolated data is more similar to the reference than the reconstructed, particularly in the near wake. However, at all SNRs, the instantaneous snapshots reconstructed wake resemble the reference more closely than the interpolated, showing that regardless of noise, the PC-DualConvNet can reconstruct the flow with a lower relative error than interpolation. Both the quantitative (Figure 4.9) and qualitative comparison of the loss functions (4.10) show that applying a harder constraint on the measurements delays the breakdown of the



(a)

(b)

(c)

Figure 4.10: Reconstructed laminar wake from noisy measurements at SNR=20 (4.10a), 10 (4.10b) and 5 (4.10c).

networks, so the networks are more robust to noise. At $SNR = 20$ and 10, $\mathscr{L}^m$ is able to achieve the same level of de-nosing as $\mathscr{L}^c$ while being more robust to noise, making it more suitable for medium levels of noise in the measurements.

4.5 Reconstructing the 2D turbulent Kolmogorov flow

In this section, we apply and compare the three loss functions described in Section 4.3.3 to reconstruct 2D turbulent Kolmogorov flows from sparse measurement. We also apply the three loss functions to reconstruct the Kolmogorov flow in a realistic scenario in Section 4.5.3 with twenty-four sensors with noise. Section 4.5.4 discuss the effect of not imposing periodic bounary condition in the network architecture on the reconstructed Kolmogorov flow.

4.5.1 Reconstructing from non-noisy measurements

In this section, we reconstruct the Kolmogorov flow from 80 input sensors and 150 general sensors placed randomly in the domain. The general sensors account for approximately 0.9% of all grid points. The sensor placement is shown in Figure 4.11. The number of inputs and general sensors are determined through a grid sensitivity study, which has been included in Appendix A.2.1. The network parameters for both the current section (Section 4.5.1) and the next section (Section 4.5.2) are found in Table A.11 to A.13.

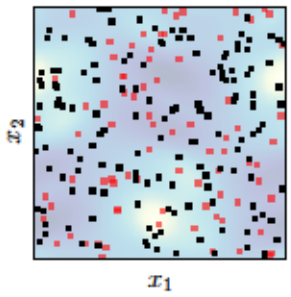


Figure 4.11: Random sensors used to reconstruct 2D turbulent Kolmogorov flow from non-noisy measurements, showing the input sensors (red) and general sensors (black).

We investigate different loss functions for reconstructing the Kolmogorov flow. Same as the reconstruction of the laminar wake (Section 4.4.1), all loss functions can achieve reconstruction errors of less than 10%. In comparison, reconstruction using 1% of the highest-energy Fourier modes leads to a reconstruction error of 20%. Unlike the laminar wake, the loss functions have a stronger influence on the reconstruction of the Kolmogorov flow due to its chaotic nature. Table 4.3 shows the mean standard deviation of the relative error computed over five trainings with different random weight initialisation. The mean relative error of networks trained with the snapshot-enforced loss $\mathscr{L}^s$ is over 20% lower than the networks trained with $\mathscr{L}^c$ and $\mathscr{L}^m$, while the mean relative errors of the networks trained with $\mathscr{L}^c$ and $\mathscr{L}^m$ are within 3% of each other.

The standard deviations of the flow reconstructed with loss functions with harder constraints ($\mathscr{L}^s$ and $\mathscr{L}^m$) are approximately half of the flow reconstructed with the softly-constrained loss $\mathscr{L}^c$. This shows that, as the dynamics become more nonlinear, loss functions with harder constraints on the measurements improve the robustness towards network initialisation and the accuracy of the output. Figure 4.12 shows that the reconstructed flow has a smaller instantaneous absolute error.

Table 4.3: The snapshot-enforced loss leads to the lowest relative error (mean±standard deviation) when reconstructing the 2D turbulent Kolmogorov flow from non-noisy measurements. The relative error is measured over five repeated trainings with different random weight initialisation and sensor locations.

	$\mathscr{L}^c$	$\mathscr{L}^s$	$\mathscr{L}^m$	Interpolated
ϵ (%)	7.19 ± 0.73	5.51 ± 0.34	7.38 ± 0.33	16.46 ± 0.00

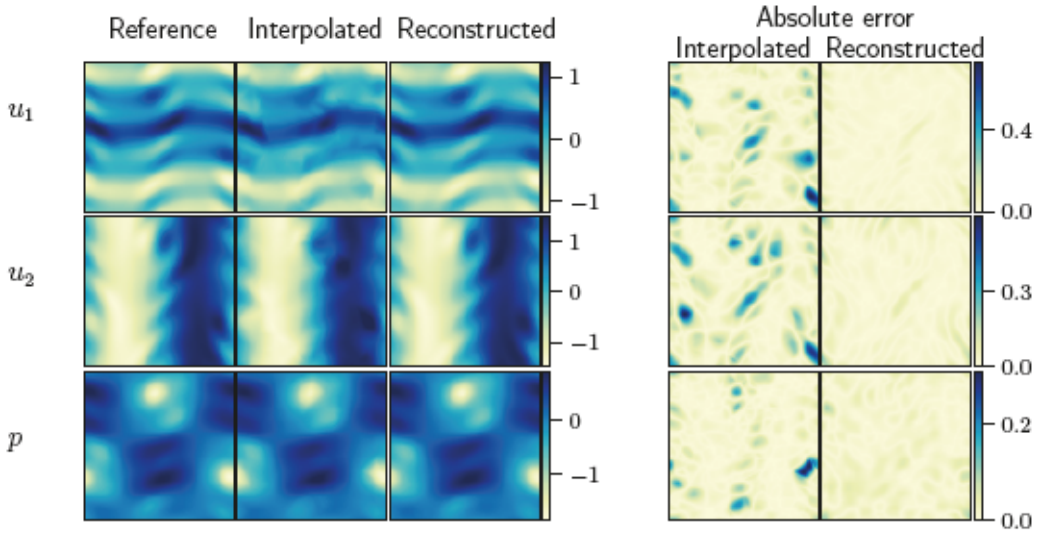


Figure 4.12: Reconstructed 2D turbulent Kolmogorov flow from non-noisy measurements with the snapshot-enforced loss, at $t = 1.7$.

We also tested how the PC-DualConvNet behaves in extreme situations by randomly removing the sensors in Figure 4.11 until only ten input sensors and ten general sensors remain. The reconstructed flow using ten sensors has a reconstruction error of 73.1%, higher than the 69.9% of the interpolated flow. However, the instantaneous and mean vorticity field (Figure 4.13) shows that the reconstructed flow is physical and shows a closer resemblance to the reference dataset. The reconstructed flow has also recovered the turbulence energy spectrum more accurately than the interpolated flow (Figure 4.13 right panel). The results with ten sensors highlight the limitation of measuring the quality of reconstruction using only the ℓ_2 error. Even if the interpolated flows have a lower relative error compared to the reconstructed flow, it clearly failed to reconstruct

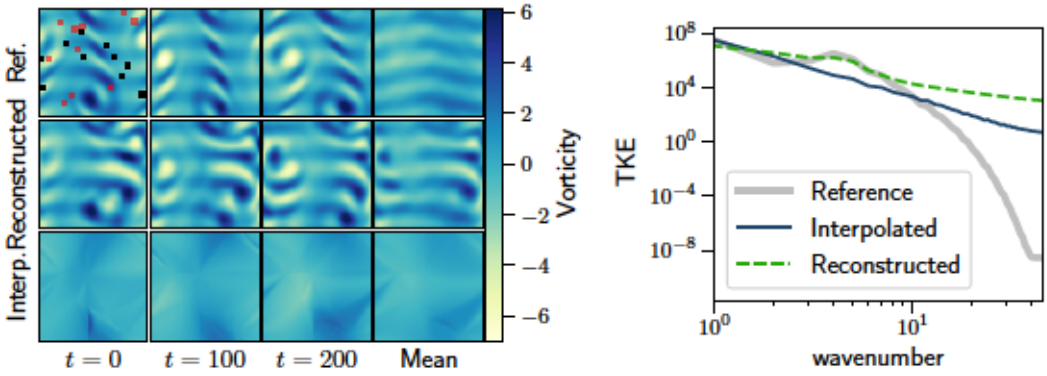
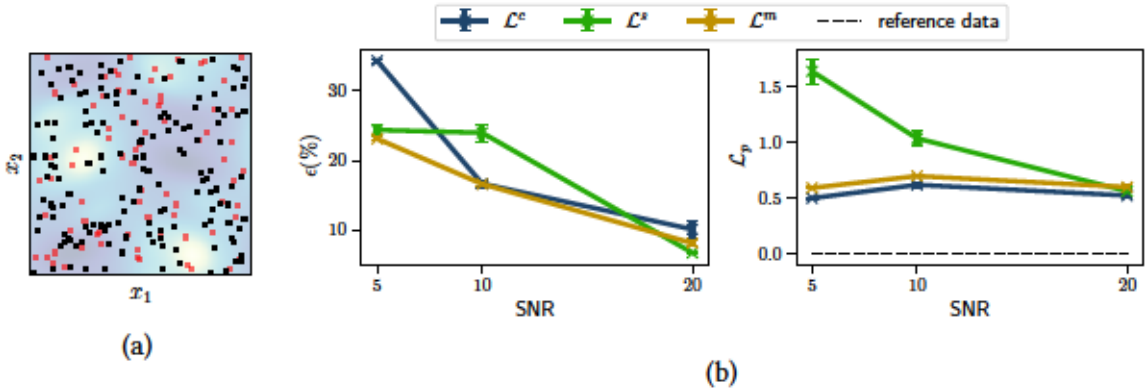


Figure 4.13: The flow reconstructed from ten input sensors and ten general sensors. The sensor placement is shown in the top left-hand panel. The reconstruction has recovered the instantaneous vorticity fields and the 2D turbulent kinetic energy spectrum.

the flow by both the TKE and visual inspection. There are clear discrepancies between the reconstructed and the reference flow in Figure 4.13, showing that more sensors are necessary to achieve a good reconstruction.

4.5.2 Reconstructing from noisy measurements

In this section, we reconstruct the 2D turbulent Kolmogorov flow from sparse measurements corrupted with white noise at $SNR = 20, 10$ and 5. The same number of sensors (80 inputs, 150 general) are placed at random locations, shown in Figure 4.14a. The relative errors and the



(a) (b)

Figure 4.14: (a) Sensor placement for reconstructing the 2D turbulent Kolmogorov flows from noisy measurements. (b) Mean and standard deviation of the reconstructed Kolmogorov flow from noisy measurements at different SNR using different loss functions.

physics loss of the flow reconstructed with $\mathcal{L}^c, \mathcal{L}^s$ and $\mathcal{L}^m$ are summarised in Figure 4.14b. The physics loss shows a similar trend to Section 4.4.2 — harder constraints on the measurements lead to higher physics losses as the noise in the measurements increases. At a low level of noise ($SNR = 20$) $\mathcal{L}^s$ leads to the lowest reconstruction error, although all loss functions achieve a relative error lower than 10%. Unlike when reconstructing the laminar wake, the softly-constrained

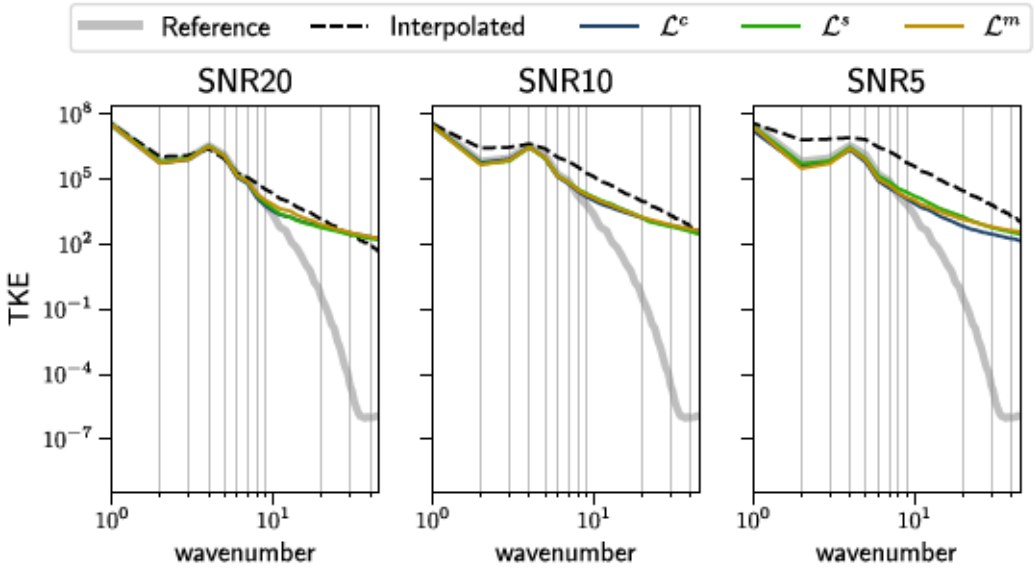


Figure 4.15: The turbulent kinetic energy (TKE) spectrum.

loss is able to reconstruct the Kolmogorov flow from measurements at $SNR = 10$, suggesting that some variation of performance is the result of hyperparameter selection when the measurements are corrupted with a medium amount of noise. One of such hyperparameters is whether to perform the optional Fourier Transform in the upper branch of the PC-DualConvNet, the details of its effect on the reconstruction can be found in Appendix B.5. Hyperparameters may be cumbersome to tune, and a (globally) optimal set is rarely found. On the other hand, $\mathscr{L}^m$ has achieved consistent reconstructions of both the 2D laminar flow (Section 4.4.2) and the 2D turbulent flow from noisy measurements. As the signal-to-noise ratio decreases further, the reconstruction error of all loss functions increases. At $SNR = 5$ for $\mathscr{L}^m$ and $\mathscr{L}^c$, the physics loss decreased compared to $SNR = 10$, showing that the reconstruction is favouring satisfying the physics over the noisy measurements. At $SNR = 5$, the reconstruction error of the flow reconstructed with $\mathscr{L}^c$ increased to 34.4%, higher than both the snapshot-enforced loss and the mean-enforced loss, showing again that harder constraints improve the robustness of the PC-DualConvNet to noise. The strictly-enforced mean loss achieves a reconstruction error of 23.1%, similar to the error of the snapshot-enforced loss, but with much lower physics loss. For all SNRs, the reconstructed flows have recovered the correct energy spectrum (Figure 4.15) up to wavenumber 1, showing that the reconstructed flow is more physical than the interpolated flow even when starting from noisy measurements.

Figure 4.16 compares the instantaneous snapshots and the mean of the reconstructed flows. At SNR=20 (Figure 4.16a) there is no visible difference between the reconstructed flows using different loss functions, even though the relative error shows that $\mathscr{L}^c$ is slightly worse than others. The flow reconstructed with $\mathscr{L}^s$ starts showing visible noise from SNR=10 (Figure 4.16b and 4.16c). At SNR=5, the flow reconstructed with $\mathscr{L}^c$ shows a visible reduction in the variation

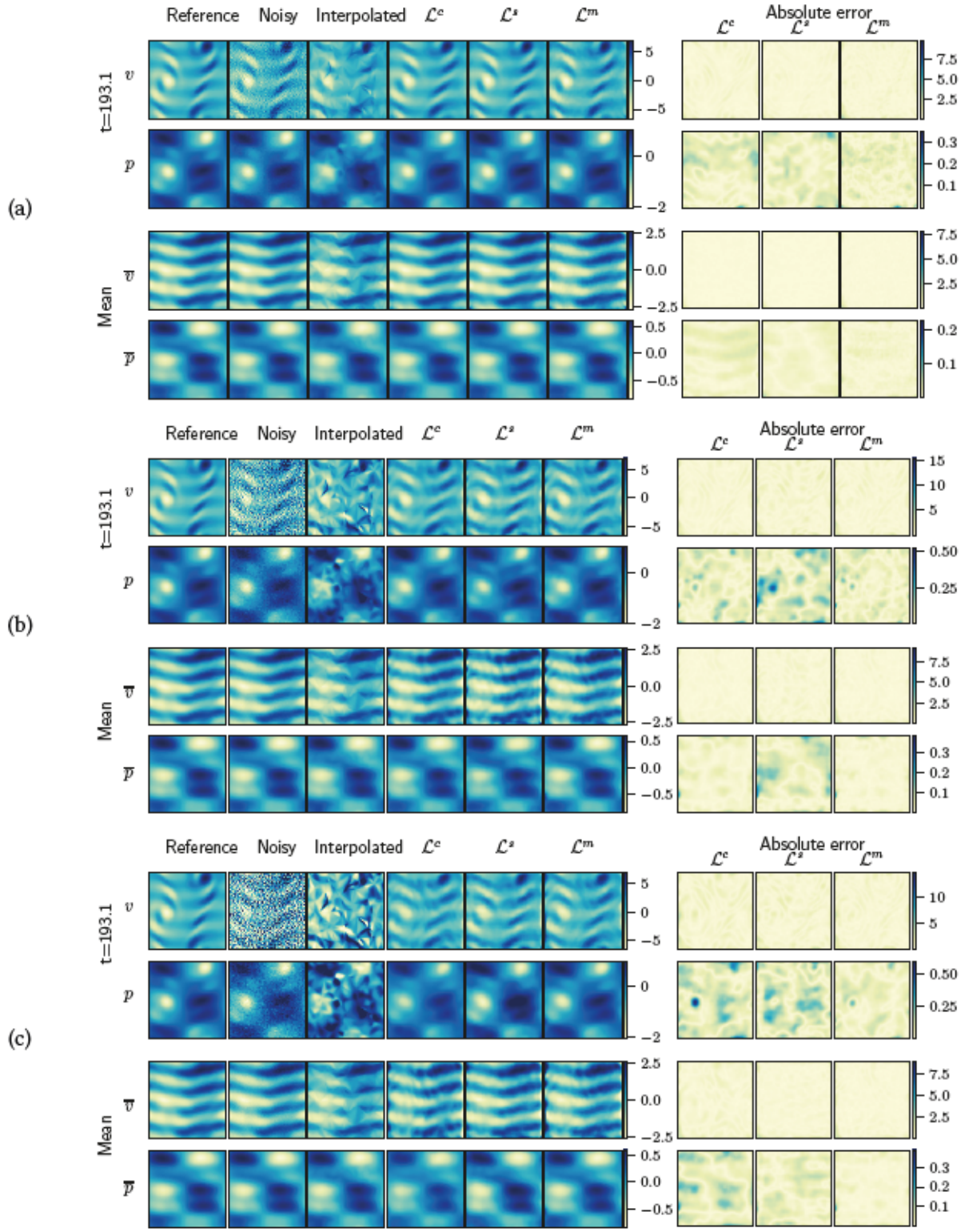


Figure 4.16: Reconstructed Kolmogorov flow from noisy measurements at SNR=20 (4.16a), 10 (4.16b) and 5 (4.16c).

of the flow field, while the flow reconstructed with $\mathscr{L}^m$ and $\mathscr{L}^s$ is not affected. At all SNRs using any loss function, the instantaneous reconstructed flow shows a closer resemblance to the reference data than the interpolated flow. Consider the relative error, the physics loss (Figure 4.14b) and the snapshots (Figure 4.16), $\mathscr{L}^m$ provides the most consistent performance at different levels of noise, providing a small relative error while keeping the physics loss close to the reference physics loss at high levels of noise.

4.5.3 Reconstructing from sparse noisy measurements

In this section, we present a case in which the sensors are sparse and noisy. From the sensors' locations of Section 4.5.2, we randomly select twenty-four general sensors and twenty-four input sensors. In the current set-up, the general sensors account for 0.14% of all grid points. The measurements contain white noise at $SNR = 10$.

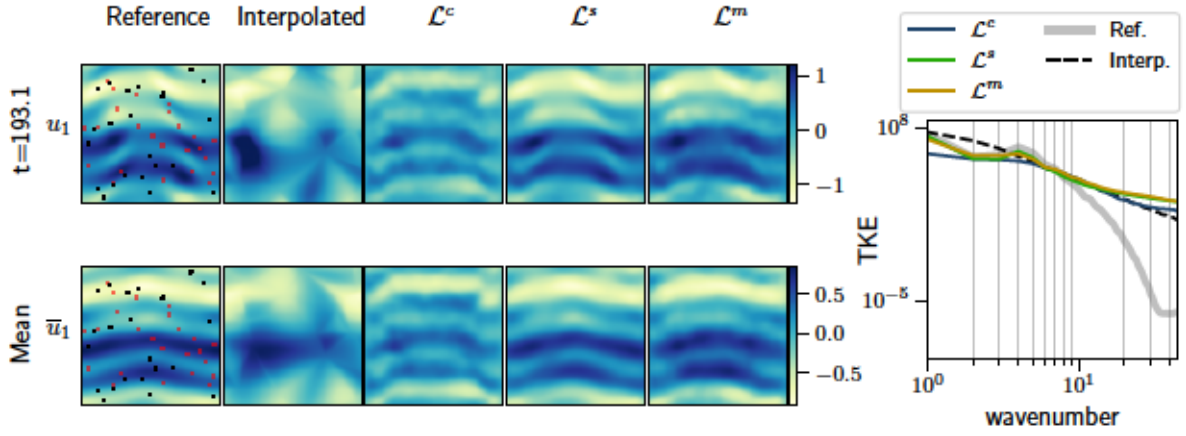


Figure 4.17: Reconstructed Kolmogorov flows from sparse noisy measurements at $SNR = 10$. Only twenty-four input sensors and general sensors are employed (the sensors are shown in the left panels). The relative errors for each reconstruction case are 70% (interpolated), 75% ($\mathscr{L}^c$), 30% ($\mathscr{L}^s$), and 45% ($\mathscr{L}^m$).

The reconstructed flows are shown in Figure 4.17. (The hyperparameters can be found in the Appendix A.2.2.) The flows reconstructed with the snapshot-enforced loss and the mean-enforced loss closely match the reference energy spectrum up to a wavenumber ≈ 7 (Figure 4.17 right panel). In contrast, both interpolation and the softly-constrained loss have larger reconstruction errors. For the network trained with the softly-constrained loss, the data loss decreases while the physics loss increases during training (Figure 4.18, right panel), ultimately causing the network to terminate training prematurely (Figure 4.18, left panel). This difficulty in simultaneously optimising the physics and data loss terms forces the network to prioritize one over the other, which can result in trivial solutions (see also Section 4.3.3).

When measurements are sparse, the primary challenge in reconstructing flows with neural net-

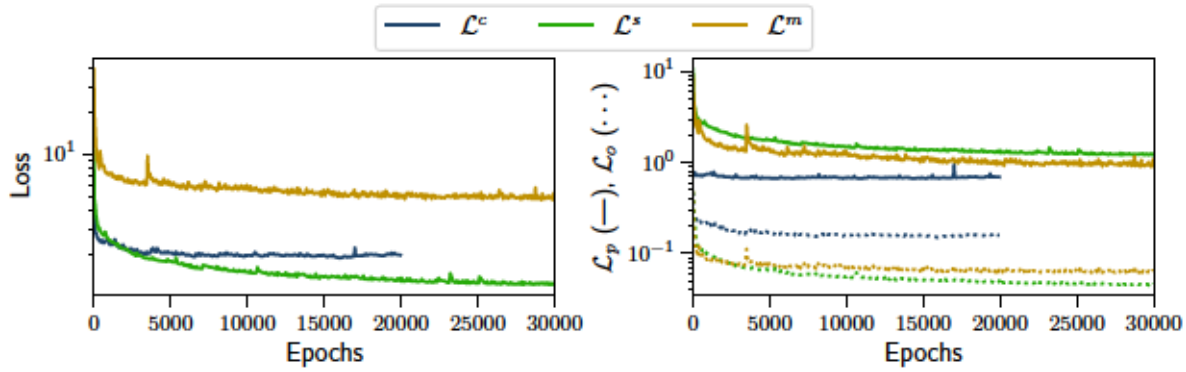


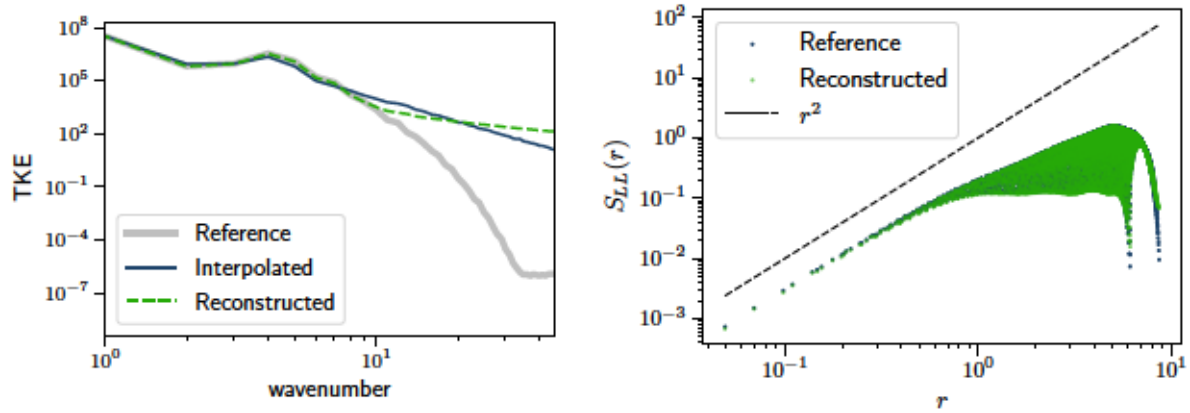
Figure 4.18: Learning curves for sparse and noisy measurements. The left panel shows the training loss, the right panel shows the physics loss (solid line) and the data loss (dotted line). When using either the snapshot-enforced or the mean-enforced loss, the physics and data losses decrease together. On the other hand, when using the softly-constrained loss, the data loss decrease while the physics loss increase.

works is finding a solution that fits the limited data, rather than filtering out noise as in Section 4.5.2. The snapshot-enforced loss prioritizes fitting the measurements more than the other loss functions, making it the most effective choice when the availability of data is limited and denoising is not the primary objective.

4.5.4 Discussion on periodic boundary conditions

In the neural networks presented in this chapter, no explicit boundary conditions are imposed as a hard constraint. This simplified the implementation and allows it the maximum flexibility to reconstruct flows with different boundary conditiions, such as the slip walls in unsteady wake dataset, and the periodic boundary conditions in the Kolmogorov flow. However, not imposing the boundary conditions in the network architecture may have an influence on the boundary points in the reconstructed flows. In retrospect, it may be helpful to impose the boundary conditions through the network architecture in the future. In this subsection, we discuss the effect of the missing periodic bounary conditions in the work presented in Section 4.5.

Fourier transform was employed in the computation of the energy spectrums shown in Figure 4.13, 4.15 and 4.17. Because of the discontinuity on the boundaries, spectral leakage can be expected when computing the Fourier transform of the entire domain for the reconstructed and interpolated flow. In the energy spectrum, spectral leakage can be observed as an excessive, spurious energy in the higher wavenumbers. To understand the effect of the missing periodic boundary conditions, we compute the second-order structure function for grid points within the domain (excluding the grid points on the boundary), which is not affect by the discontinuity at boundaries. The longitudinal second-order structure function, S_{LL}, is computed for the reconstructed Kolmogorov flow presented in this chapter using (2.2.2).



(a) (b)

Figure 4.19: The energy spectrum (a) and the second-order velocity structure function (b) of the reconstructed flow from non-noisy measurements. The same case was shown in Section 4.4.1, in Figure 4.12.

Figure 4.19 shows a comparison between the energy spectrum and the structure function, computed for the reconstructed Kolmogorov flow from non-noisy measurements, originally presented Section 4.5.1, Figure 4.12. Both the energy spectrum (Figure 4.19a) and the structure function (Figure 4.19b) shows that the reconstructed flow has correctly captured the energy for wavenumber up to 10 (corresponds to a separation of $r = 0.88$). The two measurements differs at smaller structures. While the energy spectrum shows that the reconstructed flow has a large amount of energy in smaller structure, the structure function shows that the reconstructed flow contains similar amount of energy to the reference data, and exhibits the expected scaling to r^2 for small seaprations. This difference shows that the flat tail observed in the energy spectrum was caused by spectral leakage when computing the Fourier transform of non-periodic data. The same difference of energy reconstructed for the smaller structures can be seen between Figure 4.20 and 4.15 for the noisy cases, originally presented in Section 4.5.2.

To further investigate the effect of the periodic boundary condition, we repeated the ten-general-sensor case from Section 4.5.1 (see also Figure 4.13) with the same neural network, but with periodic padding in convolution layers, which is the method we used for 3D reconstructions in Chapter 5 and 6. The repeated case achieved a relative error of 6% higher than the previous case reconstructed without using periodic padding. In both cases, the energy spectrums for both reconstructed flows fail to identify the peak at the forced wavenumber (Figure 4.13 and 4.21).

However, the problem with discontinuity at the boundaries is still present, albeit to a smaller extend. Discontinuity at boundaries is still introduced into the neural network through the feed-forward layers. To completely elimintate the discontinuity that is causing the spurious energy at high wavenumber, periodic boundary conditions must be enforced at every layer within the

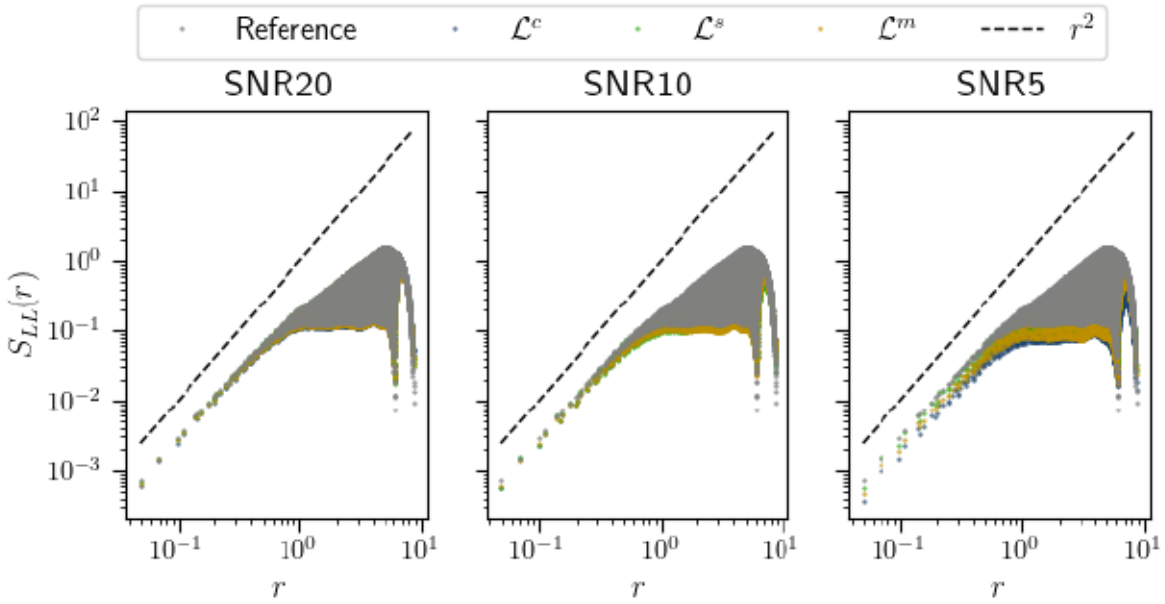


Figure 4.20: The longitudinal second-order structure function for the reconstructed flows from noisy measurements.

network, which is a limitation of the methodology of this thesis, also present in the subsequent chapters, which will need to be addressed in future works.

Still, applying periodic padding in all convolution layers has led to improvement in the reconstruction of smaller structures. Both the energy spectrum (Figure 4.21 right) and the structure function (Figure 4.22) shows an improvement in the energy contained in the smaller structures (smaller separation) after periodic boundary conditions have been included in the network, where the energy in general is lower at all scales without using the periodic padding in the network.

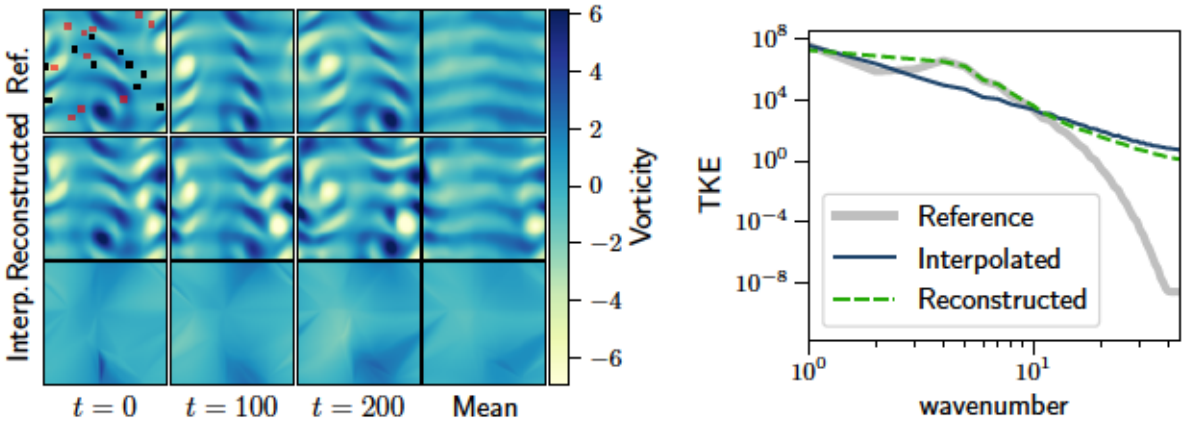
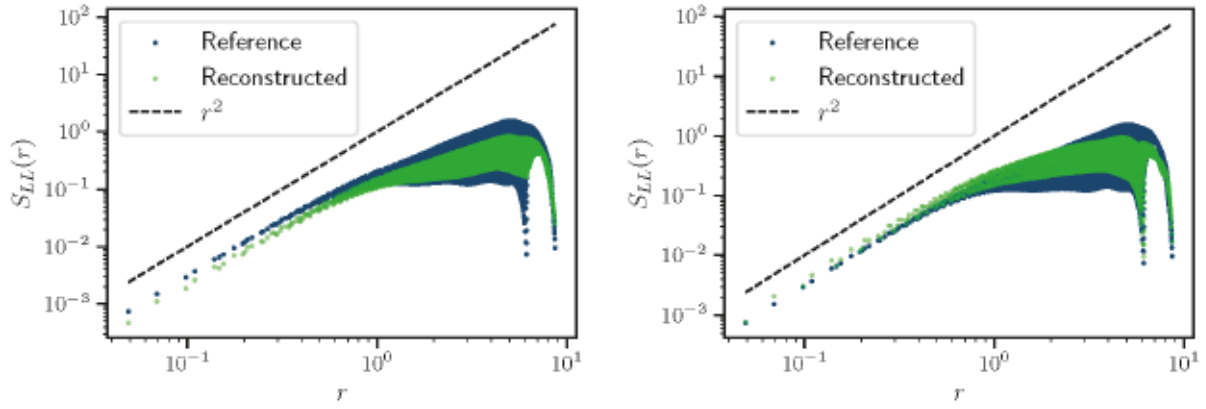


Figure 4.21: The flow reconstructed from ten input sensors and ten general sensors. This is a repeat of the case shown in Figure 4.13, but with periodic boundary hard-coded into the network architecture.



(a) (b)

Figure 4.22: The longitudinal second-order structure function of the flow reconstructed from ten input and ten general sensors (a) without periodic padding. (b) with periodic padding.

4.6 Conclusion

In this chapter, we reconstruct flow fields from sparse, noisy, and heterogeneous measurements (incomplete data) with a physics-constrained dual-branch convolutional neural network (PC-DualConvNet). The network is trained using only incomplete data. To recover the missing information and infer the full flow field, the governing equations of the flow are embedded into the loss function. We investigate three loss functions, i.e., the softly-constrained loss, which is common in the flow reconstruction literature (Raissi et al. 2019); the snapshot-enforced loss (Gao et al. 2021) for steady flows; and the mean-enforced loss (proposed here). First, we reconstruct both the 2D laminar wake and the 2D turbulent Kolmogorov flow from non-noisy measurements from a number of sensors of $\sim 1\%$ of all grid points. When reconstructing the laminar wake, we find no significant difference between the softly-constrained and the snapshot-enforced losses. However, when reconstructing the 2D turbulent Kolmogorov, we find that the snapshot-enforced loss leads to a $\sim 25\%$ reduction in the reconstruction error compared to the softly-constrained loss due to the flow being chaotic. Second, we tested the loss functions on measurements corrupted with white noise at signal-to-noise ratios (*SNR*) 20, 10 and 5. At $SNR < 10$, the snapshot-enforced loss leads to noisy reconstructed flows because the hard constraint on the instantaneous measurements prevents the network from denoising. On the other hand, the softly-constrained loss leads to a smaller variability in the reconstructed flow dynamics because the network favours minimising the physics loss over the data loss. Finally, combining the benefit of the softly-constrained and the snapshot-enforced losses, we propose the mean-enforced loss. At high levels of noise, the flow reconstructed with the mean-enforced loss is physical and does not spuriously attenuate the flow variations, while achieving similar reconstruction errors to the snapshot-enforced loss. The reconstruction errors show that hard constraints in the loss functions make the PC-DualConvNet more generalisable. In conclusion, we find the snapshot-enforced loss is

suitable for reconstructing flows from non-noisy measurements, and the mean-enforced loss is suitable for reconstructing flows in more general situations, i.e., when the measurements are noisy.

Chapter 5

Reconstructing 2D intermittent and 3D turbulent flows

In the previous chapter, we reconstructed a 2D laminar unsteady wake of a triangular body and a 2D weakly turbulent Kolmogorov flow. In this chapter, we extend our PC-DualConvNet, designed and used in Chapter 4, to reconstruct a 2D turbulent Kolmogorov flow exhibiting extreme dissipation events, and a set of 3D turbulent Kolmogorov flows.

The relevant codes are available at
`https://github.com/MagriLab/FlowReconstructionFromExperiment`.
The current version of the codes have been updated to work with 3D data, therefore there may be compatibility issues when reproducing the results in Section 5.3. Please see the README.md in the linked repository for more details. All data is available upon request.

5.1 Introduction

Numerous ML-based methods have been developed to reconstruct flows from sparse measurements, as we have presented in the review in Section 1.1.2. Some tackle the reconstruction of 2D flows from sparse measurements without using the ground truth during training (Raissi et al. 2019, Arzani et al. 2021, Gao et al. 2021, Kelshaw & Magri 2024, Page 2025). Amongst the works where CNNs are employed, Gao et al. (2021) reconstruct 2D steady biological flows, while Kelshaw & Magri (2024) and Page (2025) reconstruct 2D turbulent Kolmogorov flows. In our own work in Chapter 4, we reconstructed 2D Kolmogorov flows that are weakly turbulent with a Reynolds number of $Re = 32$, which is the same as Kelshaw & Magri (2024). This chapter extends the PC-DualConvNet, introduced in Chapter 4 for reconstructing 2D weakly turbulent flows, to more realistic cases with minimum modifications. We consider two cases, each with additional complexity compared to the cases in Chapter 4.

The first case is a 2D Kolmogorov flow at a higher Reynolds number $Re = 40$. At high Reynolds

numbers, the Kolmogorov flow shows intermittent bursts of energy dissipation, referred to as extreme events (Farazmand & Sapsis 2017). Extreme events, also known as intermittency or bursting in turbulence, are a common phenomenon in many types of flows (Lohse & Grossmann 1993, Vassilicos 2001, Chandler & Kerswell 2013), forming an important part of the turbulence dynamics. Extreme events are identified with a sudden increase in the energy dissipation rate (Farazmand & Sapsis 2017). Extreme events are associated with the production of Reynolds stress and the energy transfer between coherent structures in wall-bounded turbulence (Blossey & Lumley 2000, Panton 2001, Farano et al. 2017). In forced, unbounded flows, extreme events are found to be associated with distinct structures in the flow and the transfer of energy from large-scale structures to the mean flow (Yeung et al. 2015, Farazmand & Sapsis 2017). Our reconstructed Kolmogorov flows should capture the correct extreme events to recover the correct physical information.

The second case we look at is 3D Kolmogorov flows. Most of the works reviewed in Chapter 1 reconstruct 2D flows, including our own previous works. Even though 2D turbulence is a suitable simplification in certain problems, such as geophysical flows, it avoids some 3D complications such as vortex stretching, limiting the applicability of results obtained with 2D turbulence (Boffetta & Ecke 2012). In Chapter 1, we reviewed statistical and variational methods used to reconstruct 3D flows. PINNs have been used to reconstruct 3D flows in various sensor setups, such as turbulent channel flow from tracked particles (Clark Di Leoni et al. 2023), wake from sparse, point measurements (Zhang & Zhao 2021, Rui et al. 2023), and a stratified flow from plane measurements (Zhu et al. 2024). Without initial conditions, and with data organised on regular grids, computer vision approaches have been taken to reconstruct 3D flows. CNNs (e.g., Özbay & Laizet 2023, Xuan & Shen 2023) and generative models (e.g., Dubois et al. 2022, Yu et al. 2022, Yousif et al. 2023) have been used to recover missing data from limited measurements in a 3D domain. Instantaneous velocities within wall-bound flows have been reconstructed from wall quantities only using CNNs and generative adversarial networks (Guastoni et al. 2021, Cuéllar et al. 2024). However, these methods require the full flow field to be known at some point during the training process, which may not be possible in practice. Other works which do not require the full flow field have tested their methods in 2D, but have speculated that their methods are extendable to 3D (Buzzicotti et al. 2021, Page 2025). In this chapter, we extend the method we developed for 2D to 3D flows.

This chapter is organised as follows. First, we introduce the datasets (2D with extreme events and 3D Kolmogorov flows) and our methods in Section 5.2. Then, we present our reconstructed 2D Kolmogorov flow with extreme events (Section 5.3) and our reconstructed 3D Kolmogorov flows (Section 5.4). Finally, we draw our conclusion in Section 5.5.

5.2 Datasets and methods

Since this chapter aims to test whether the method we developed for 2D reconstruction can be extended to more realistic cases, we will use the same methodology as in Chapter 4 with minimum modifications. We present two cases of Kolmogorov flows in this Section, which we consider as more realistic compared to Chapter 4 and the articles we reviewed. First, a 2D turbulent Kolmogorov flow exhibiting extreme dissipation events (Section 5.2.1), and second, turbulent 3D Kolmogorov flows (Section 5.2.2). The methods for taking measurements and reconstructing the flows are presented in Section 5.2.3.

5.2.1 2D turbulent Kolmogorov flow with extreme events

The 2D Kolmogorov flow dataset is generated at $Re = 40$, with 32 wavenumbers and the time step $\Delta t^* = 0.005$, using the method described in Section 2.1.2 for 2D flows. The generated flow is saved every 20 time steps and moved onto a grid of 128×128 points in the physical space via inverse Fourier Transform. The physical resolution is increased via zero-padding in the Fourier space. No additional de-aliasing is performed. The chosen number of grid points in each direction is double the number of grid points required to meet the Nyquist sampling rate, providing a smoother visualisation and gradient field, which we compute in the physical space via numerical differentiation. The procedure is also described in Kelshaw & Magri (2024). The resulting dataset has 5000 snapshots with a time step $\Delta t = 0.1$. The dataset with extreme events, $\boldsymbol{D} = [U^T, P^T]^T \in \mathbb{R}^{N_t \times N \times N \times (N_u+1)}$ is the snapshot matrix of the full state, where $N_t = 6000$ is the number of snapshots, $N = 128$ is the number of grid points in each direction, and $N_u = 2$ is the number of velocity components. The dataset exhibits extreme dissipation events, which are identified using a threshold value $\mathbb{E}[Di] + 2\sqrt{\mathbb{E}[Di^2] - \mathbb{E}[Di]^2}$, where Di is the dissipation rate and $\mathbb{E}[*]$ is the expection, here approximated by the mean (Farazmand & Sapsis 2017). The dissipation rate is computed using Equation (2.8). These extreme events introduces additional difficulties when reconstructing due to more effective degrees of freedoms compared to the weakly turbulent flow we reconstructed in Chapter 4. A detailed comparison of the two Kolmogorov flow at different Reynolds number can be found in Section B.4.

5.2.2 3D turbulent Kolmogorov flow

The 3D Kolmogorov flow dataset consists of four time series of 3D Kolmogorov flows, initialised with different initial conditions. Each time series is generated at $Re = 34$, with 32 wavenumbers and the time step $\Delta t^* = 0.005$, using the method described in Section 2.1.2 for 3D flows. A snapshot is saved every 20 time steps and interpolated onto a grid of $64 \times 64 \times 64$ points in the physical space, resulting in a time step of $\Delta t = 0.1$. Each time series contains 500 snapshots, which is longer than the decorrelation time of the flow. The combined dataset $\boldsymbol{D}$ contains 2000 snapshots. The 3D Kolmogorov flow dataset here is chaotic and sensitive to initial conditions, where a small

perturbation added to the initial condition causes the flows to diverge exponentially. However, the flow is not fully-developed turbulence and does not have a power-law spectrum (Figure 5.1b).

The dataset is validated by comparing its time-averaged properties with those found in the literature. The mean u_1 averaged across three directions is shown in Figure 5.1c, where the average u_1 in the forced direction x_2 has a sinusoidal profile matching the frequency of the forcing term, and the averaged velocity is 0 in other directions. Given a long enough dataset, the mean of a 3D Kolmogorov flow is expected to have the same velocity profile as its laminar form, independent of the Reynolds Number (Borue & Orszag 1996), which is also shown in the 3D plots of velocities and pressure in Figure 5.1a. The energy spectrum shows the exponential decay of energy with increasing wavenumber, which matches the results obtained by Shebalin & Woodruff (1997). Figure 5.1 shows that the combined dataset has converged and has the expected mean velocity profile.

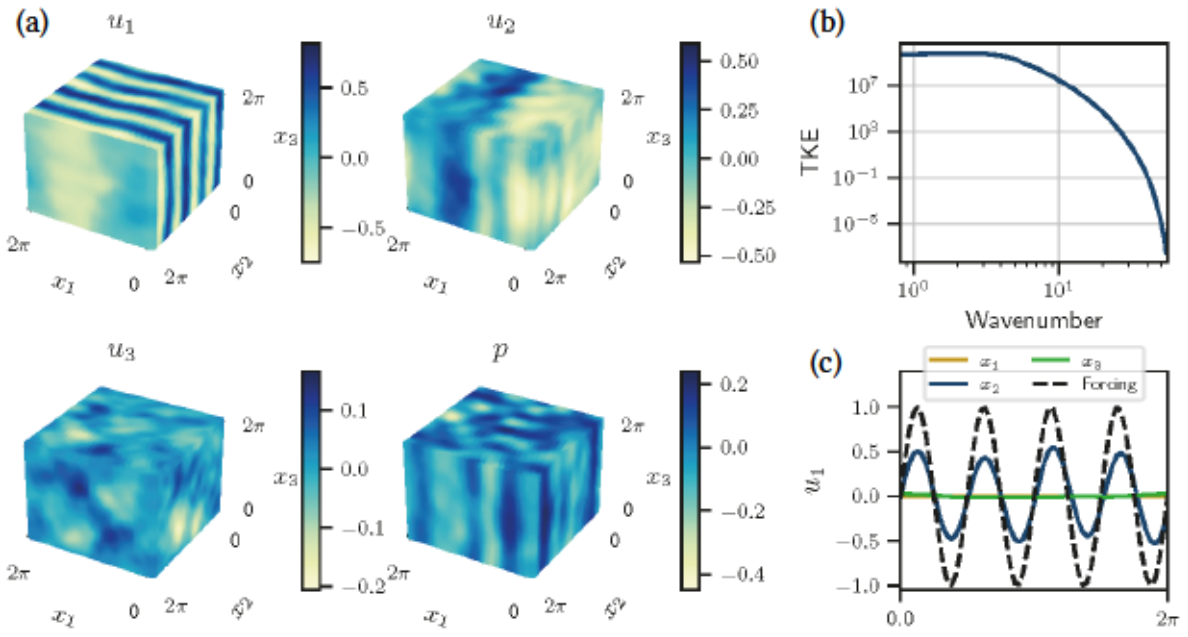


Figure 5.1: The mean properties of the 3D Kolmogorov flow dataset. (a) The mean velocities and pressure are shown in 3D. (b) The turbulent kinetic energy spectrum against wavenumber, where an exponential decay of energy between low- to medium wavenumbers can be seen. (c) The mean u_1 averaged over each of the x directions, compared with the forcing term. The mean u_1 across x_2 shows a sinusoidal profile matching the frequency of the forcing term, meaning that the dataset has converged.

5.2.3 Measurements and network structures

Measurements are taken randomly in the flow domains following the same method as Section 4.3.1. We use the PC-DualConvNet architecture presented in Chapter 4, with minimum

modifications. When reconstructing the 2D Kolmogorov flow (Section 5.2.1), no modification is made apart from updating the hyperparameters. When reconstructing the 3D Kolmogorov flow (Section 5.2.2), the 2D convolution layers in Section 4.3.2 are replaced with 3D convolution layers. Periodic padding is used in the 3D convolution layers for the periodic boundary conditions in Kolmogorov flows.

5.3 Reconstructing 2D Kolmogorov flow with extreme events

We reconstruct a 2D Kolmogorov flow which exhibits extreme dissipation events from sparse measurements taken from the simulation with added white noise at SNR=10. As this section is an extension to the methods presented in Chapter 4, we start directly with the reconstruction from noisy measurements. We also use this section to explore the robustness of the snapshot-enforced loss to noisy measurements.

Input measurements are taken from an 8-by-8 regular grid in the domain, shown in red in Figure 5.2. There are also 80 general sensors, shown in black, placed randomly in the domain. The

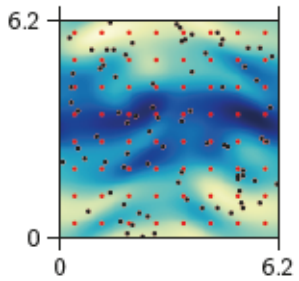


Figure 5.2: Sensor placement for reconstructing 2D Kolmogorov flow with extreme events. The measurements consist of the pressure inlet measurement at $\boldsymbol{x}_{in}$ (red) and measurements of all components at $\boldsymbol{x}_s$ (black).

mean and range of the reconstruction errors from three different random sensor placements are reported in Table 5.1. The sensor placement which leads to the lowest physics loss (Figure 5.2) is presented in this section. We select the case with the lowest physics loss because the snapshot-enforced loss function minimises only the physics loss, so the relative error is expected to decrease with the physics loss. The hyperparameters of the network used to obtain all results in this section can be found in Appendix A.3. As part of the hyperparameter study, we also tested the effect of the FFT layer, which we discuss in Appendix B.5. We found that turning on the FFT layer improves the denoising ability in PC-DualConvNet, at the expense of a small increase in the relative error.

In the case presented, the relative error of the reconstructed flow is 56% lower than that of the interpolated field. The instantaneous vorticity field of the reconstructed flow is compared to the

Table 5.1: The mean and range *[min, max]* of the relative error and physics loss, tested over three runs using different random sensor placements. The reconstructed flows have relative errors that are approximately 50% compared to the errors of the interpolated flows. The case presented in the rest of this section is the case with the lowest physics loss.

ϵ (reconstructed)	ϵ (interpolated)	**Physics loss** (reconstructed)	**Physics loss** (reference)
23.1 [22.3, 23.5]%	50.6 [49.2, 52.0]%	0.89 [0.92, 0.86]	2.4×10^{-5}

interpolated and the reference flows at $t = 10, 32, 200$ and 500 (Figure 5.3), where $t = 32$ and 200 correspond to the occurrences of extreme events. The reconstructed vorticity fields are calculated as the curl of the reconstructed velocity fields. At all time steps shown, the reconstructed vorticity snapshots are a closer match with the reference than the interpolated snapshots, capturing large-scale structures. At $t = 32$, the distinct structures during an extreme event are clearly discernible in both the reconstructed and the reference snapshots, but not in the interpolated. The pressure fields are reconstructed more accurately than the velocity fields because more pressure measurements are availble, since both the input sensors and the general sensors measure pressure. However, the reconstructed pressure snapshots show less noise than the interpolated fields.

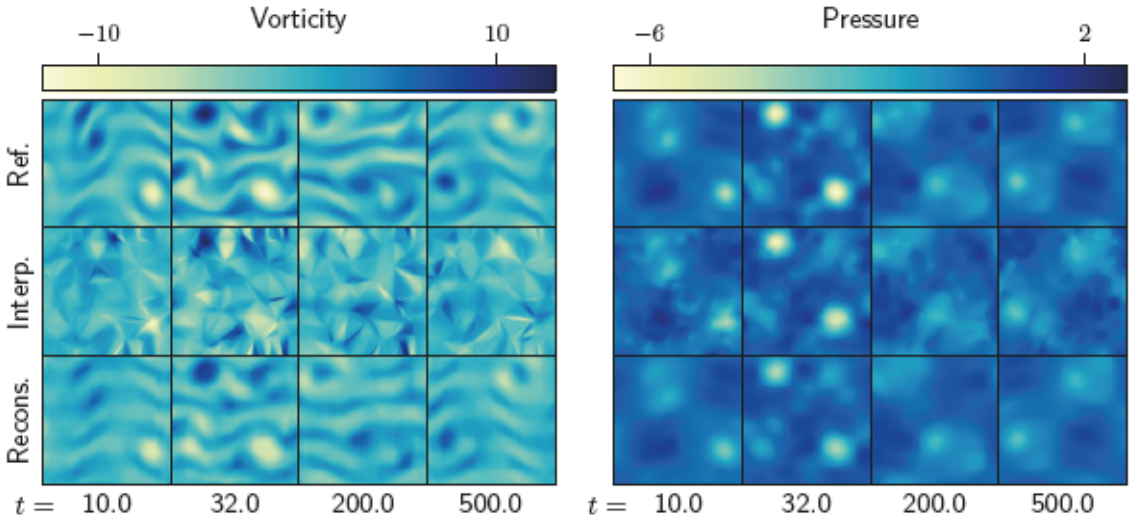


Figure 5.3: Snapshots of the vorticity (left) and pressure (right) fields at different instances, where the snapshots at t=32 and 200 show the flow field during extreme events. In all the snapshots shown here, the reconstructed flow fields show structures that closely resemble the reference. Especially the vorticity field, where the interpolated flow has failed to show the correct structures.

Physical metrics of the reconstructed flow are shown in Figure 5.4. The energy spectrum (5.4 left

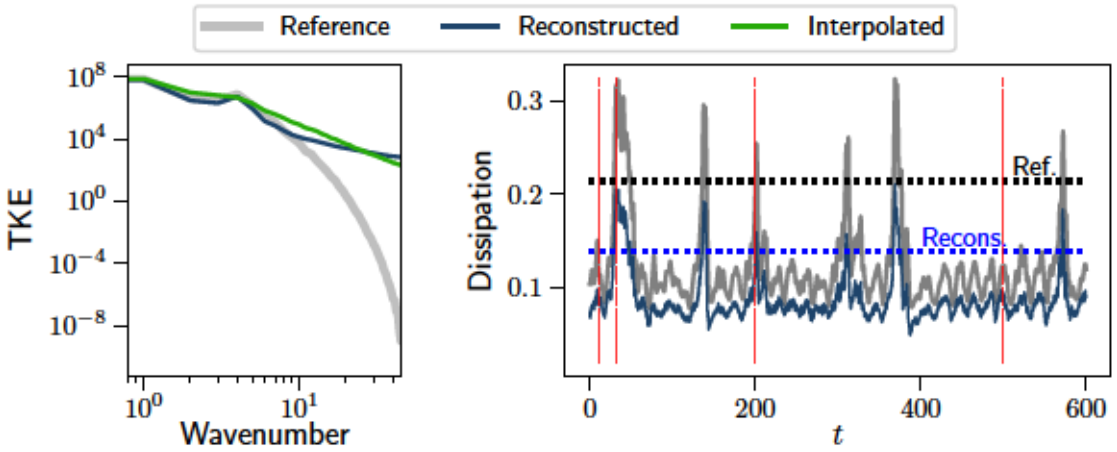


Figure 5.4: Physical metrics of the reconstructed flow compared to the reference. Left: the energy spectrum of the reconstructed flow and the reference data. The spectrum shows exponential decay of energy. Right: The global dissipation of the reconstructed flow and the reference data. The black and blue dashed lines are the extreme event thresholds for the reference and reconstructed flow, respectively. The red vertical lines mark the four snapshots shown in Figure 5.3. The reconstructed flow has the same timing of extreme events as the reference data, even though the overall dissipation has a lower magnitude.

panel) shows that the network has successfully reconstructed the flow structures up to wavenumber 10. The energy spectrum of the reconstructed flow also shows the energy increase caused by the sinusoidal forcing at wavenumber 4, which the interpolated flow has failed to identify. Figure 5.4 (right panel) shows the spatially averaged dissipation of the reconstructed and reference flow. The vertical dashed lines mark the instances where the corresponding snapshots are shown in Figure 5.3, where two extreme events are shown. The horizontal dashed lines are the thresholds for the reference and reconstructed flow above which we consider an extreme event has occurred.

The dissipation of the reconstructed flow has a smaller magnitude and does not reach the threshold of the reference flow. The smaller magnitude indicates that the reconstructed flow has less energy compared to the reference flow. Because the network is unaware of the periodic boundary condition, the energy spectrum of the reconstructed flow in (Figure 5.4 left) shows spurious energy at high wavenumbers, which may be seen as contradicting the dissipation plot (Figure 5.4 right). The second-velocity structure function (Figure 5.5), which is not affected by the discontinuity at the boundaries, also shows that the reconstructed flow has lower energy compared to the reference flow in general. A detailed discussion on how the missing periodic boundary conditions affect the analysis of our results can be found in Section 4.5.4.

When thresholds are calculated separately for the reference (black line) and the reconstructed (blue line), the dissipation of both flows exhibits extreme behaviours at approximately t =32,

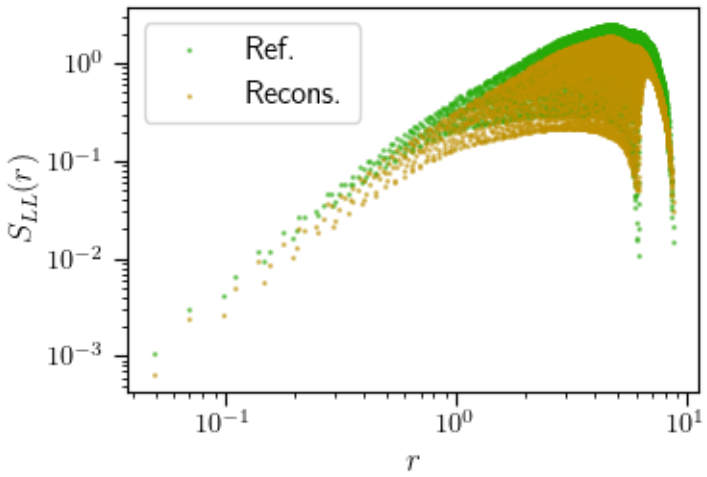


Figure 5.5: The second-order velocity structure function of the reconstructed flow compared to the reference.

135, 200, 309, 370, and 570. The reconstructed flow captures the correct timing of the extreme events, including the extreme events at t =200 and 570, which are not seen in the input pressure measurements in Figure 5.6. By reconstructing the correct extreme events when there is no spike in the input pressure measurements, we know that the PC-DualConvNet produces a reconstruction based on the physics and not the magnitude of the input. Overall, the PC-DualConvNet has reconstructed the Kolmogorov flow to reproduce correct large-scale structures in realistic snapshots and the correct timing of extreme dissipation events, from sparse, noisy measurements.

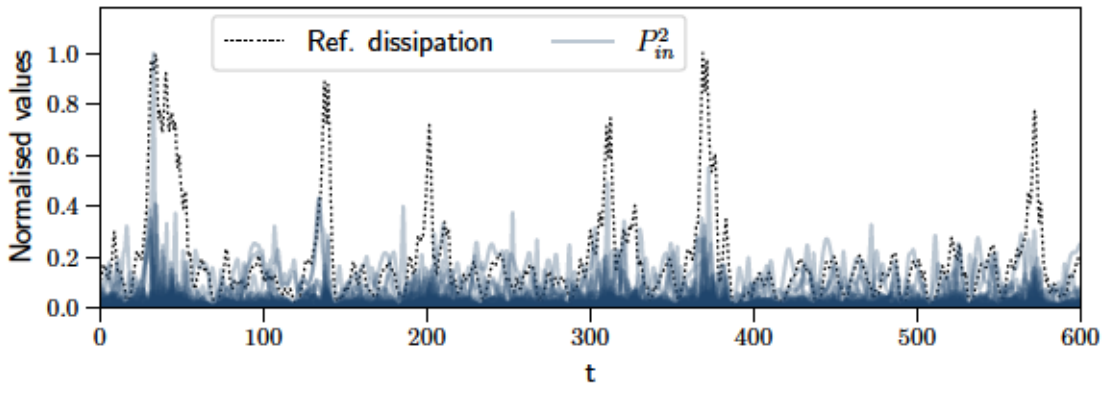


Figure 5.6: Measurements from each sensor are plotted with a blue translucent line, and the lines from all sensors are overlaid to show the overall input trend. All values are normalised. We can see that not all extreme events (dotted line) have corresponding peaks in the pressure input, but the network is still able to capture the correct timing of all the extreme events.

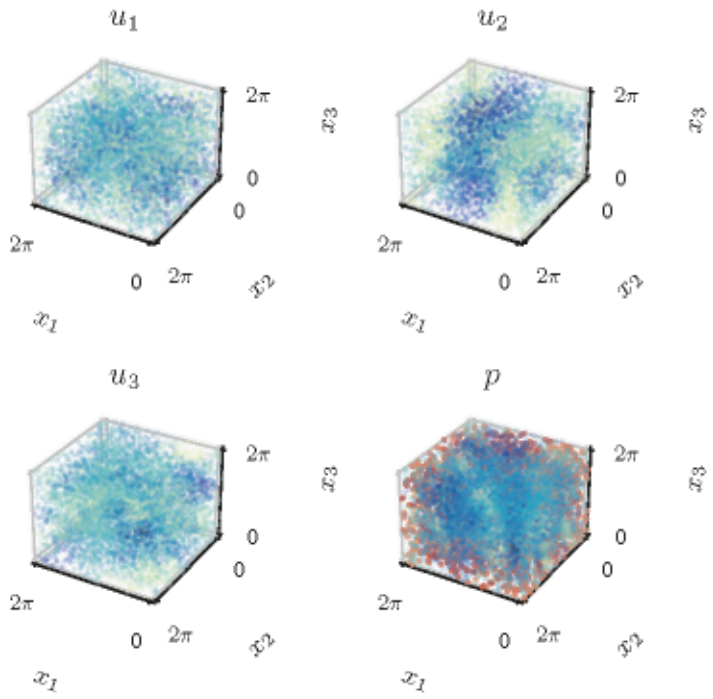


Figure 5.7: Sensor placement for reconstructing 3D Kolmogorov flow. General sensors are shown in blue, and the pressure inlet sensors are shown in red.

Table 5.2: The mean and standard deviation of the errors of the reconstructed 3D flow.

	Relative error	**Physics loss**	**Sensor loss**
Reference	N/A	0.137 ± 0.00	N/A
Reconstructed	$28.8 \pm 0.6\%$	0.733 ± 0.030	0.017 ± 0.001

5.4 Reconstructing 3D Kolmogorov flows with PC-DualConvNet

We reconstruct the 3D Kolmogorov flow dataset, described in Section 5.2.2, from sparse measurements taken randomly from the domain. As we do not consider noise at the current stage, we use the snapshot-enforced loss function to reconstruct the 3D flow. Instead, cases with noise will be presented in Chapter 6. The general sensors, which measure both velocity and pressure, are placed at approximately 2.7% of all grid points. The input sensors, which measure only the pressure, are placed at approximately 1.4% of all grid points. Reconstruction using 2.7% of the most energetic Fourier modes leads to a relative error of 0.7%. These sensor placements are shown in Figure 5.7, where the general sensors are shown in blue and the pressure inlet sensors are shown in red.

The mean and standard deviation of the relative error and physics loss of the reconstructed flow, computed over five runs with different random sensor placements, are shown in Table 5.2. The

hyperparameters of the network are selected based on the unweighted sum of physics and sensor loss, and can be found in Appendix A.3.

Instantaneous slices of the reconstructed 3D flow are compared with the reference flow in Figure 5.8. Slices are taken at different x_1 and x_3 values because these are the two homogeneous directions. Both the vorticity magnitude and pressure fields of the reconstructed flow show good agreement with the reference flow.

The reconstructed flow was also able to recover the statistical properties of the reference dataset. Figure 5.9a and 5.9b show the time-averaged vorticity magnitude and pressure of the reconstructed and reference flow, respectively. The probability densities of the fluctuating velocities and pressure are also a close match to the reference flow (Figure 5.9c), including the two-peak distribution of the fluctuating u_2. The reconstructed flow has also recovered the correct energy spectrum up to wavenumber 10 (Figure 5.10). The energy spectrum shows spurious energy at very high wavenumbers due to the discontinuity at boundaries introduced into the neural network through some layers, the details of which have been discussed in Section 4.5.4. When shown on a linear scale (Figure 5.10b), the energy associated with wavenumbers larger than 10 is negligible. The results show that the PC-DualConvNet, originally designed for 2D flows, can be directly extended to reconstruct 3D Kolmogorov flows.

5.5 Conclusion

In this chapter, we extend the physics-constrained dual-branch convolutional neural network (PC-DualConvNet), introduced in Chapter 4, to reconstruct more realistic cases of Kolmogorov flows. Sensors are randomly placed within the domain using the same method as in Chapter 5 We first reconstruct a 2D Kolmogorov flow which exhibits extreme dissipation events from measurements taken from approximately 0.5% of all grid points, with added white noise at $SNR = 10$. By combining the Fourier layer with the snapshot-enforced loss, the network is able to reconstruct snapshots from noisy, sparse measurements, to match the reference flow. The extreme events identified from the reconstructed flow also match the timing of extreme events of the reference flow. When considering physical metrics, the energy spectrum of the reconstructed flow is accurately reproduced up to wavenumber 10. We then reconstruct a 3D Kolmogorov flow dataset from sparse sensors placed at approximately 2.7% of all grid points. The reconstructed flow has reproduced both the instantaneous flow fields and the time-averaged properties of the flow. Likewise, in the 2D extreme event case, the energy spectrum of the reconstructed 3D flow is accurately reproduced up to wavenumber 10. The results show that the PC-DualConvNet and its associated reconstruction process in Chapter 4, initially designed for 2D flows, can be directly applied to reconstruct 3D Kolmogorov flows from sparse measurements with minimum modifications.

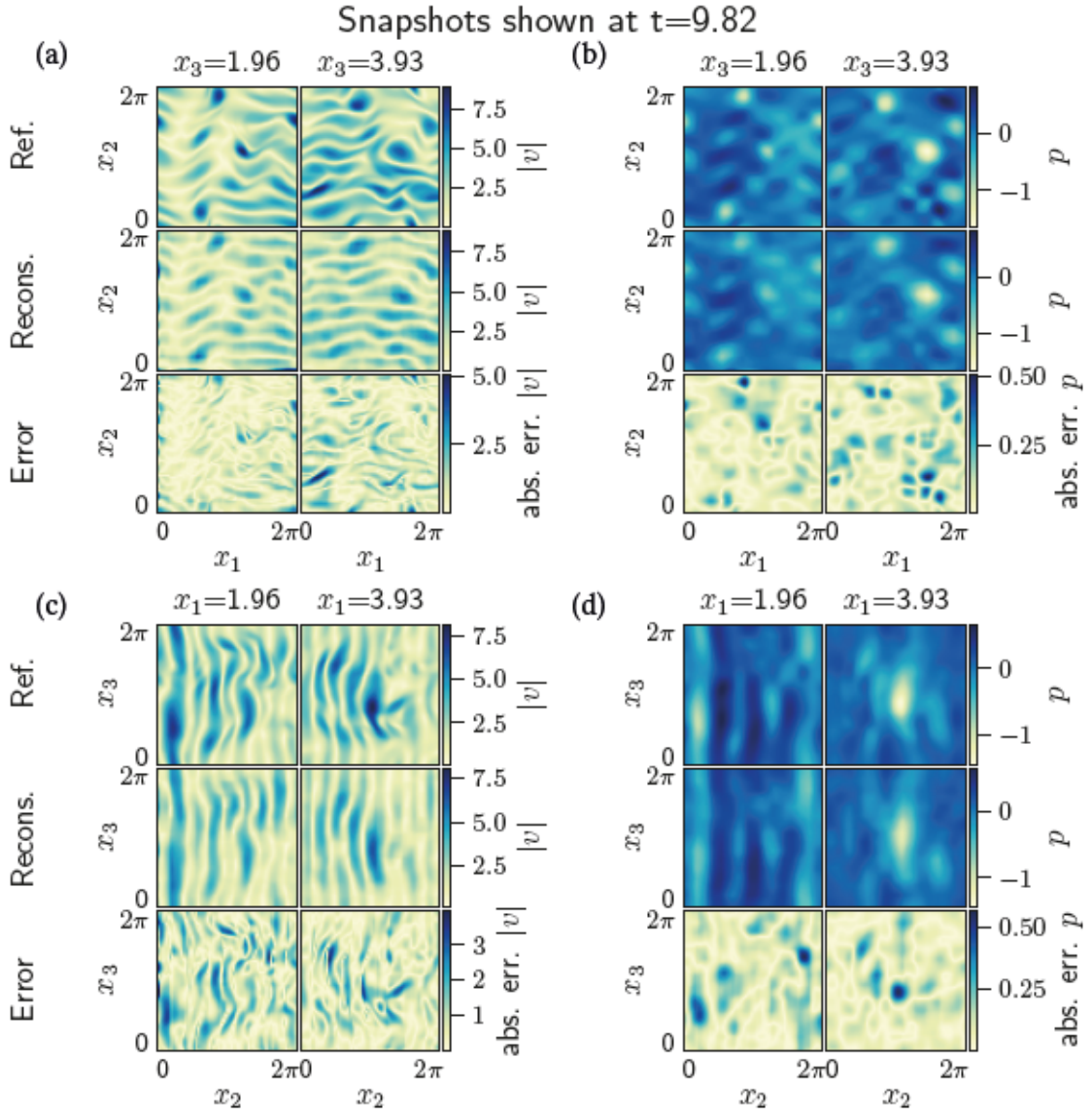


Figure 5.8: Slices of the reconstructed and reference flow domain at an instance in time. The top row shows the vorticity magnitude (a) and pressure (b) on x_1-x_2 planes at two different x_3 values, and the bottom row ((c) and (d)) shows the x_2-x_3 planes at two different x_1 values.

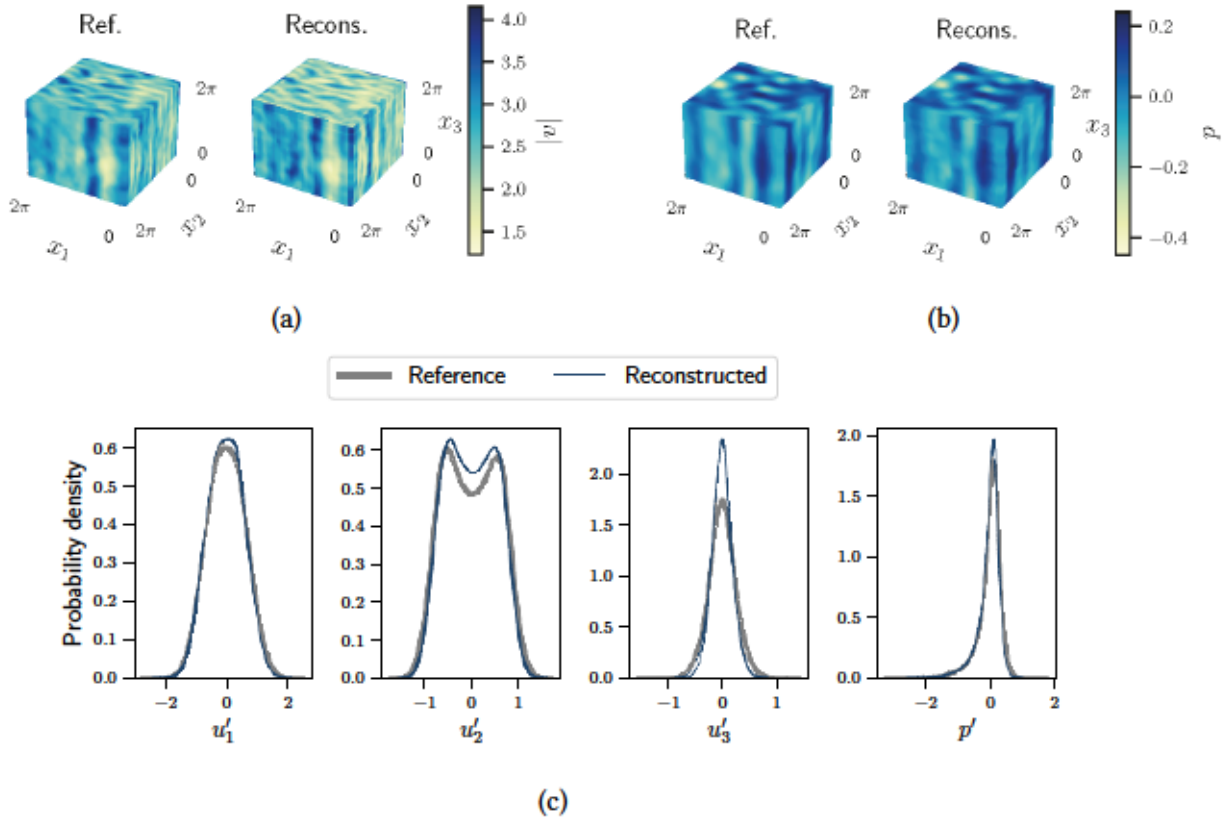


Figure 5.9: Statistical properties of the reconstructed flow, compared with the reference. (a) Time-averaged vorticity magnitude. (b) Time-averaged pressure. (c) Probability density of the entire 3D domain.

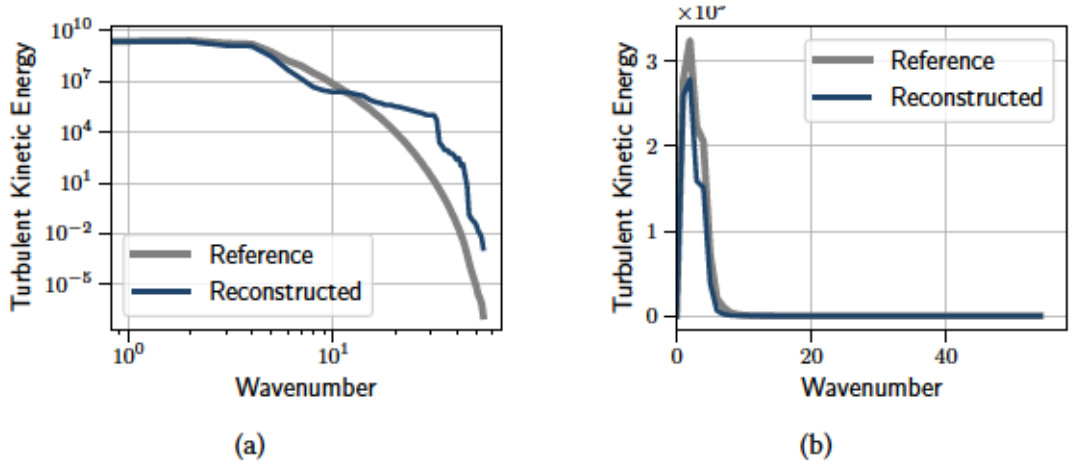


Figure 5.10: The turbulent kinetic energy spectrum of the reconstructed flow (blue) and the reference flow (grey). The spectrum in shown in both logarithmic (a) and linear (b) scales. The reconstructed flow has successfully captured the energy spectrum up to wavenumber 10. When shown on a linear scale, the energy contained within eddies past wavenumber 10 is negligible.

Chapter 6

Reconstructing 3D turbulent flows from plane measurements

Three-dimensional flows are more complex and expensive to simulate compared to 2D flows, but are also important in real-world engineering applications. In Chapter 5, we reconstruct 3D turbulent Kolmogorov flows from sparse measurements taken randomly from the domain. However, sparse random measurements of both velocities and pressure are not the most common in experiments. In this chapter, we consider realistic sensor placements to reconstruct a 3D turbulent Kolmogorov flows.

The work in this chapter has been published in *Data-Centric Engineering* (Mo & Magri 2026). All data is available upon request. The relevant codes are available at
https://github.com/MagriLab/FlowReconstructionFromExperiment.

6.1 Introduction

In the previous chapter, we reconstructed a set of 3D turbulent Kolmogorov flows from measurements taken randomly from the domain. However, large numbers of random point measurements from within the flow domain of interest, especially of both velocities and pressure, are not the most common in experiments. In experiments, there are classes of methods to collect data. Non-intrusive methods, such as methods based on particle tracking, are often preferred over intrusive methods (e.g., hot-wire anemometry or pitot-static tubes), as they do not disturb the flow. Currently, there are no non-intrusive methods to measure static pressure directly (Tavoularis & Nedić 2024), which appears in the incompressible Navier-Stokes equations (2.1). Instead, pressure is often obtained at walls via pressure taps (McKeon & Engler 2007, Cal et al. 2010, Rowan D. Brackston 2017), or solved for from well-resolved velocity fields by using the incompressible Navier-Stokes equations (Pröbsting et al. 2013, van Oudheusden 2013, Tavoularis

& Nedić 2024).

Amongst the non-intrusive methods for measuring velocities, particle image velocimetry (PIV) is the most popular when a large field of spatially-resolved velocity data is required. PIV provides in-flow velocity measurements on a regular grid, and is capable of measuring time-resolved velocities up to three dimensions in a plane (Tavoularis & Nedić 2024). Volumetric velocity can be obtained by measuring multiple planes in the same experiment. For example, by scanning the measurement planes across the domain (e.g., Zhu et al. 2024), measuring simultaneously multiple planes arranged either in parallel (e.g., Ganapathisubramani et al. 2005, Pfadler et al. 2009, Cal et al. 2010) or perpendicular to each other (e.g., Morton & Yarusevych 2016, Chandramouli et al. 2019), or using tomographic PIV (e.g., Pröbsting et al. 2013). The regular grid and spatial resolution of PIV measurements make them a good starting point for flow reconstruction.

Using POD, Druault & Chaillou (2007) reconstructed the mean in-cylinder flow from multiple PIV planes; Hamdi et al. (2018) reconstructed an impinging jet from multiple parallel planes; and Chandramouli et al. (2019) reconstructed a 3D turbulent flow from two planes perpendicular to each other, using both experimental and synthetic data. A 3D stratified flow has also been reconstructed from multiple experimental PIV planes using PINN (e.g Zhu et al. 2024).

Synthetic data on a regular grid have also been used to develop methods for reconstructing 3D flows. Pérez et al. (2020) reconstructed a cylinder wake from multiple parallel planes of velocities. CNN-based methods have been used to reconstruct 3D free surface flow from surface measurements (Xuan & Shen 2023), and other turbulent flows from a cross-plane setup (two planes perpendicular to each other) (Yousif et al. 2023). Özbay & Laizet (2023) reconstructed the 3D wake of a cylinder from both a cross-plane and multiple parallel planes taken from simulations.

Many physical systems such as flows have symmetries and obey conservation laws, which have been exploited in multiple works on flow reconstruction. For a pedagogical account of symmetries and conservation laws in the Navier-Stokes equations, the readers are referred to Frisch (1995). One of these symmetries is the homogeneity, which states that the flow is statistically invariant under translation (Pope 2000). When reconstructing 3D flows from multiple planes, Chandramouli et al. (2019) used the homogeneous assumption to design their reconstruction method such that the method does not require the full 3D flow field.

Incorporating symmetries, such as rotational and translational symmetries, into the neural networks to help standardise the input has improved image reconstruction and classification (Jaderberg et al. 2015, Mehr et al. 2018). A similar approach was taken by Kneer et al. (2021) to obtain physically meaningful compression of flows. Page et al. (2021) found coherent structures of a Kolmogorov flow by utilising the periodic boundary conditions to define and decompose a translation operator in the latent space of an autoencoder. In these cases, symmetries are used to extract relevant information from data before further processing. Neural networks can also

be designed to force certain physical properties on their output, such as conservation of energy (Greydanus et al. 2019), conservation of mass (Mohan et al. 2023, Page 2025), and periodicity (Ozan & Magri 2023).

Motivated by experimental techniques, we reconstruct a 3D Kolmogorov flow from a small number of planes of velocity measurements and a plane of pressure measurements at the boundary of the flow. We designed a CNN-based weight-sharing network to both reduce the number of parameters needed for the reconstruction of the 3D flow, and to exploit the homogeneous directions in the Kolmogorov flow. This chapter is structured as follows: First, we describe the sensor setup and the network in Section 6.2. Then, we present the reconstructed flow from non-noisy measurements in Section 6.3, and from noisy measurements in Section 6.4. Finally, we conclude the chapter in Section 6.5.

6.2 Methodology

In this section, we first describe the dataset to be reconstructed and how the measurements are taken (6.2.1). Then, we introduce the network designed for 3D reconstruction (6.2.2).

6.2.1 Dataset and measurements

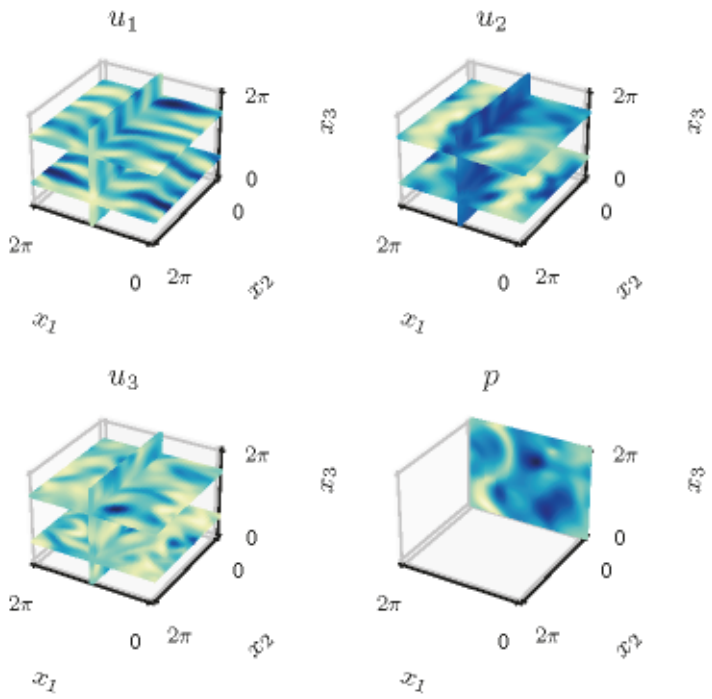


Figure 6.1: Pressure is measured at the plane $x_2 = 0$. Three planes of velocities are taken at $x_1 = \pi$, $x_3 = 0.5\pi$, and 1.5π. The pressure is used as input to the network, and all measurements are used as collocation points.

We use the 3D turbulent Kolmogorov flow dataset, $\boldsymbol{D}$, whose details are given in Section 5.2.2. All three velocity components are measured at all grid points $\boldsymbol{x}_s$ on three planes: an $x_2 - x_3$ plane

at $x_1 = \pi$, and two $x_1 - x_2$ planes at $x_3 = 0.5\pi$ and 1.5π. Pressure is measured at all grid points $\boldsymbol{x}_{in}$ on the $x_1 - x_3$ plane at $x_2 = 0$. Velocities and pressure are measured at different planes to acknowledge that these quantities are measured using different instruments in experiments. The planes are shown in Figure 6.1 The collection of all measurements $\xi(\boldsymbol{D}) = \{\boldsymbol{U}(\boldsymbol{x}_s), \boldsymbol{P}(\boldsymbol{x}_{in})\}$ contains both the velocity measured at $\boldsymbol{x}_s$ and the pressure measured at $\boldsymbol{x}_{in}$. The pressure measurements $\boldsymbol{P}(\boldsymbol{x}_{in})$ are used as input to the network. The measurements account for approximately 3.8% of the total number of variables in a snapshot. The number of $x_1 - x_2$ planes is determined by testing and reducing the number of planes until the relative errors from both networks exceed 50%; the details can be found in Appendix A.4.1.

6.2.2 The weight-sharing network

Part of the difficulty in reconstructing 3D flows is the large demand on computational resources. CNNs are already lightweight compared to other types of commonly used networks, such as fully-connected networks or transformers, but a large amount of resources is still required for three-dimensional convolutions. The Kolmogorov flow is known to be statistically homogeneous in all but the forced direction (Borue & Orszag 1996). When homogeneous directions are present, the statistical dimension of the flow is reduced (Pope 2000). For example, if one direction is homogeneous in a 3D flow, then the flow is statistically 2D. We develop a weight-sharing network to both reduce the number of parameters and to fully utilise the homogeneity in the Kolmogorov flow. Our dataset exhibits translational symmetry $\mathcal{T}_{\Delta x_3} : \boldsymbol{D}(x_1, x_2, x_3) \rightarrow \boldsymbol{D}(x_1, x_2, x_3 + \Delta x_3)$. In practice, we make our network aware of the translational symmetry by sharing some network parameters, $\boldsymbol{w}_{x_3} = \boldsymbol{w}_{x_3+\Delta x_3} = \boldsymbol{w}$ for all Δx_3, across the x_3 direction. In our network, the shared parameters $\boldsymbol{w}$ form the 2D inner network, shown as the orange blocks in Figure 6.2. The implementation of sharing a part of the network weights as a subnetwork is similar to that of a Siamese Neural Network (Bromley et al. 1993). However, instead of identifying the differences between two inputs, our network apply the same reconstruction learned at the measured planes to the unmeasured planes at different x_3.

The network, shown in Figure 6.2, can be broken down into three parts:

- **Input processing**: Pressure input $\boldsymbol{P}_{in}$ at any instance in time, which is a 2D matrix, is passed through a 2D convolutional layer and a fully-connected layer, resulting in a 2D matrix.
- **The 2D inner network**: The 2D matrix from the previous step is split along an axis, which will eventually become x_3 in the output, into vectors. Each vector is passed through the same PC-DualConvNet (orange block) and becomes an intermediate result on a $x_1 - x_2$ plane, to promote the idea that all planes are statistically invariant. These intermediate planes are then stacked along the x_3 direction.

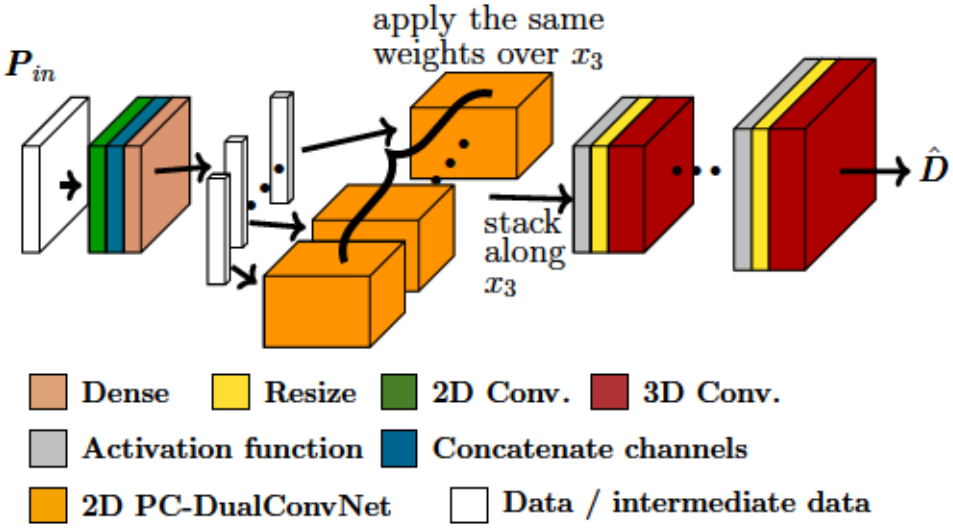


Figure 6.2: Schematics of the weight-sharing network. The input pressure $\boldsymbol{P}_{in}$ (first white block) is a 2D matrix of pressure measurements at $x_2 = 0$. It is then passed through a 2D convolution layer and a fully-connected layer to produce another 2D matrix, which is then split into vectors along an axis, which will later become x_3 direction in the final output (group of white long blocks). Each vector is then passed through the same PC-DualConvNet (orange block) to produce an intermediate 2D result representing a $x_1 - x_2$ plane. These intermediate results are stacked along x_3 direction to form a 3D intermediate result, which is then passed through multiple 3D convolutions and reshaping layers to produce the final reconstructed flow $\hat{\boldsymbol{D}}$.

- **The 3D CNN:** The stacked output from the previous step is passed through multiple layers of 3D convolutions and resizing via linear interpolation to produce the final output with the correct dimensions.

If we had an infinite number of snapshots of a single $x_1 - x_2$ plane, then every possible realisation of the flow on a plane would be represented in the training data. However, we cannot have an infinite number of training snapshots. Therefore, only parts, and not all, of the weights are shared across the x_3 direction to avoid too much restriction on the network. Sharing parts of the weights informs the network that the flow is statistically similar across the x_3 direction, which is a more efficient use of available data.

6.3 Reconstructing 3D Kolmogorov flow from planes

We present here the results of reconstructing the 3D Kolmogorov flow from planes of measurements, shown in Section 6.2.1. We compare the reconstruction results of the weight-sharing network and the PC-DualConvNet for 3D flows from Chapter 5. The network parameters for this section are listed in Appendix A.4.1. The symbols in this, and the rest of this chapter, are defined the same as in Chapter 4.

A summary of the results is shown in Table 6.1. The weight-sharing network achieves lower

Table 6.1: The relative error ϵ, the physics loss $\mathscr{L}_p$, and the sensor loss $\mathscr{L}_o$ (*mean* ± *standard deviation*) of the reconstruction results from planes of measurements, averaged over five runs with different random initialisations of network weights.

	ϵ (%)	$\mathscr{L}_p$	$\mathscr{L}_o$
Reference data	N/A	0.137±0.000	N/A
Weight-sharing network	49.3±0.4	0.324±0.092	0.0182±0.0011
PC-DualConvNet	54.7±7.4	0.354±0.109	0.0070±0.0009

values in both the relative error and the physics loss, and with a smaller standard deviation, compared to the PC-DualConvNet. The sensor loss of the weight-sharing network is over twice that of the PC-DualConvNet, despite the weight-sharing network achieving a lower relative error and physics loss. As the sensor loss measures only the data points on the measurement planes, we can see that the PC-DualConvNet has overfitted to the measurement planes. By partially sharing weights across the x_3 direction, the weight-sharing network has a built-in assumption that all $x_1 - x_2$ planes are similar, thereby reducing overfitting to the measurement planes.

Despite the larger relative error when reconstructing from measurement planes compared to what we have achieved in previous chapters, the results are still showing good agreement with the reference data for both networks statistically. There is little difference between the two networks when comparing the mean flow (Figure 6.3) or the energy spectrum (Figure 6.4). Similar to the results obtained in Section 5.4, the networks here have recovered the correct energy spectrum up to approximately wavenumber 10, which contains the majority of the energy in the flow.

The differences between the networks are more visible when comparing individual planes within the 3D domain. Figure 6.5 shows two instantaneous $x_1 - x_2$ planes taken from the reference and reconstructed flow, the top row at $x_3 = 1.77$, which is close to the measured plane at $x_3 = 0.5\pi$, and the bottom row at $x_3 = \pi$, which is further away from any measured plane. Figure 6.6 shows the error of individual reconstructed planes over x_3, which also shows that the weight-sharing network is more reliable when away from the measured planes. Both planes shown in Figure 6.5 are unseen by the network during training. At both x_3, the reconstructed velocity u_1 (Figure 6.5 left) from the weight-sharing network has retained the flow structures expected of an instantaneous snapshot. However, PC-DualConvNet has failed to recover the flow patterns in $x_3 = \pi$ and instead uses patterns similar to the mean flow in that plane. Pressure reconstruction (Figure 6.5 right) is worse for both networks, given that there are no pressure measurements inside the flow. However, the weight-sharing network has correctly identified most areas of low pressure at $x_3 = \pi$, while the PC-DualConvNet has failed to do so.

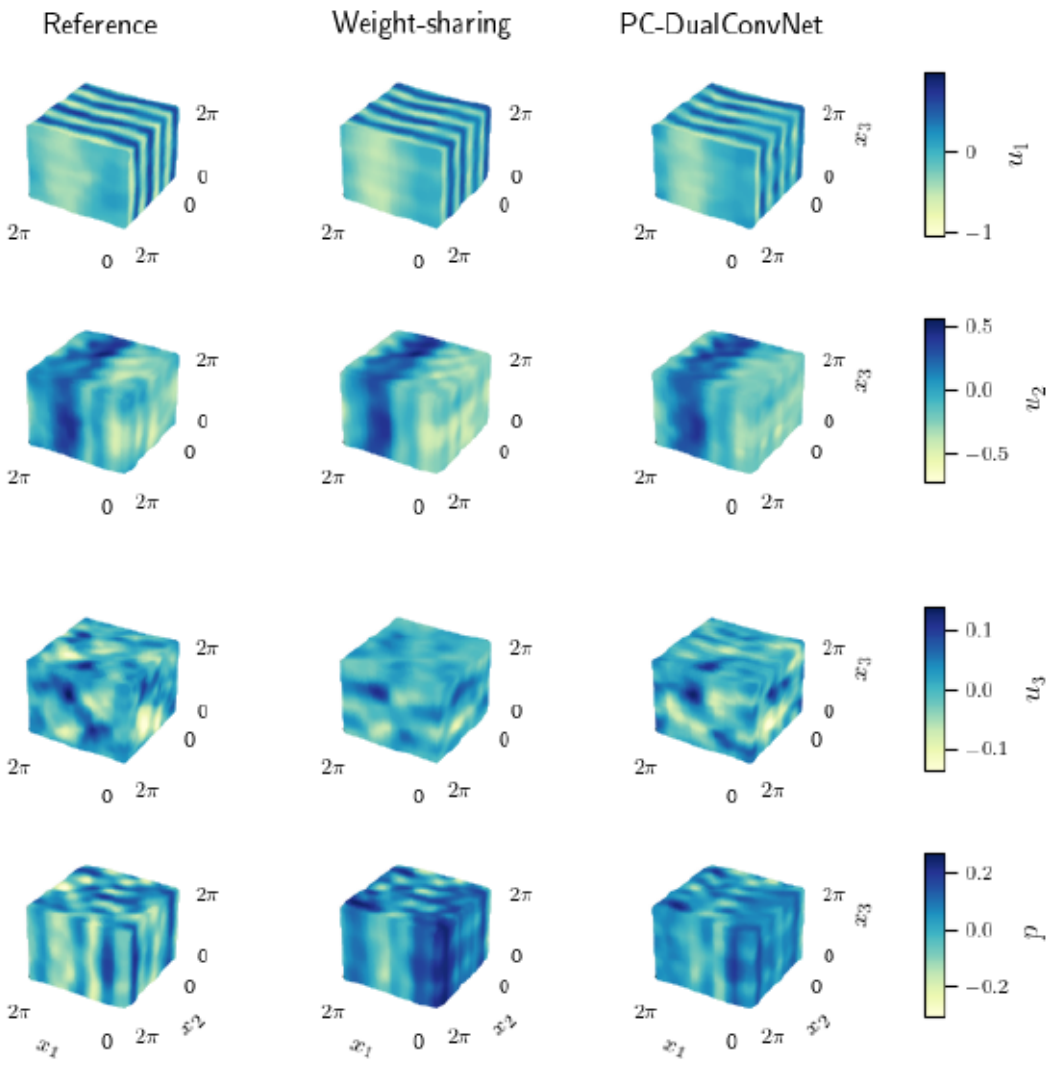


Figure 6.3: Time average of the volumes. From left to right, the columns are reference data, weight-sharing network, and PC-DualConvNet.

6.3.1 Reconstructing from a single cross-plane

In this section, we test the reconstruction from a single cross-plane. Figure 6.7a shows the location of the velocity measurements. The pressure inputs are taken from the same grid points as in Section 6.2.1. Appendix A.4.1 provides more details on the tests to determine the minimum number of planes needed for accurate reconstruction. When only a single cross-plane is used, the reconstruction relative errors for PC-DualConvNet and the weight-sharing network are 78% and 62%, respectively. Both networks achieve a similar reconstructed turbulent kinetic energy (Figure 6.7b), which are not significantly different from reconstructing from two $x_1 - x_2$ planes in Section 6.3. By comparing slices in the reconstructed domain (Figure 6.7c), we can see that the reconstructed flow field by the PC-DualConvNet has lost resemblance to the reference data at $x_3 = 1.72\pi$. The difference between the networks is more pronounced in the pressure field, where the PC-DualConvNet fails to reconstruct the low pressure region in the centre of

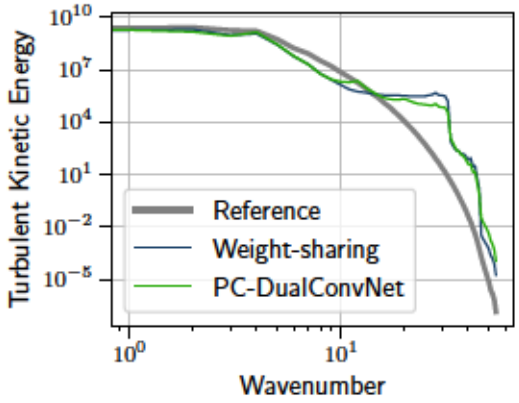


Figure 6.4: The energy spectrum of the reconstructed flow.

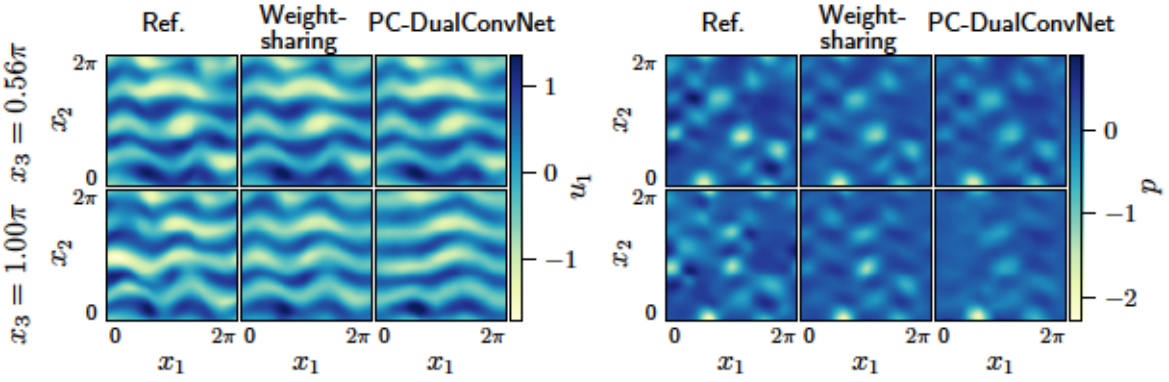


Figure 6.5: Two $x_1 - x_2$ planes taken from the reconstructed flow at $x_3 = 0.56\pi$ and π, which are unseen by the network during training. The measurements used in training are taken on planes at $x_3 = 0.5\pi$ and 1.5π. Moving further away from the measured plane, the instantaneous planes reconstructed by the weight-sharing network show a closer resemblance to the reference, whereas the PC-DualConvNet outputs the mean flow. An example of this difference can be seen in the bottom row, which shows a plane taken at $x_3 = \pi$.

the slices, while the weight-sharing network successfully reconstructs those regions. Numerical error (Figure 6.8) also shows that the reconstruction deteriorates faster as we move away from the measured plane for the PC-DualConvNet.

6.4 Reconstructing 3D Kolmogorov flow from noisy measurements

In this section, we reconstruct the flow using the same sensor setup as described in Section 6.2.1, but with added white noise at SNR=15, using the same method as in Chapter 4. We use the mean-enforced loss as it is shown to be more robust to noise in Chapter 4. We show the process of selecting the hyperparameters in Section 6.4.1, and the reconstructed flow in Section 6.4.2.

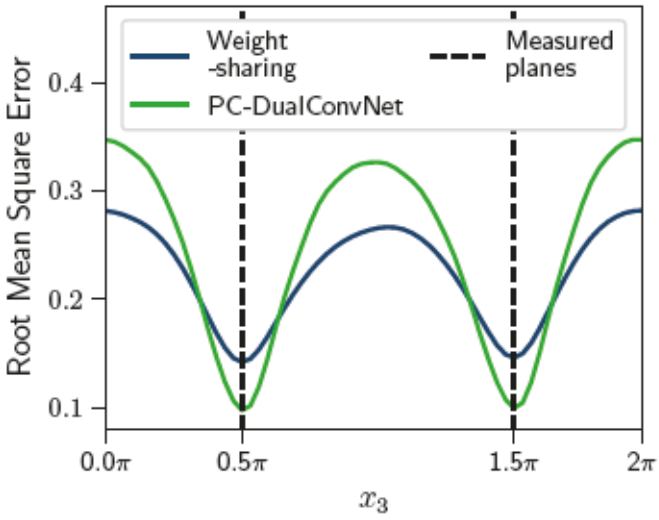


Figure 6.6: The root mean square error (sqaure root of MSE) of of each reconstructed $x_1 - x_2$ plane.

6.4.1 Selecting the hyperparameters

Before we can use the network to reconstruct the flow, we select a set of hyperparameters for the network that we expect will lead to an accurate reconstruct the entire 3D flow. We use only noisy measurements during the selection process since we are not expected to know the true flow. During the selection process, we run multiple tests with the training dataset composed of the measurements taken from the planes shown in Section 6.2.1 with added white noise at SNR=15. The same noisy training dataset will also be used later in Section 6.4.2.

Given that we are interested in a spatial reconstruction from sparse measurements, the validation dataset should provide information about a network's ability to generalise to unseen regions of the flow. Thus, our validation dataset is the measurement from the $x_1 - x_2$ plane at $x_3 = \pi$, which is unseen by the network during training. The validation sensor loss is then $MSE\left(\hat{\boldsymbol{U}}(x_3 = \pi), \boldsymbol{U}_n(x_3 = \pi)\right)$, where $\hat{\boldsymbol{U}}$ is the reconstructed velocity field and $\boldsymbol{U}_n$ is the noisy reference velocity field. The training sensor loss is $MSE\left(\xi(\hat{\boldsymbol{D}}), \xi(\boldsymbol{D}_n)\right)$. Since the training sensor loss is computed with pressure measurements, but the validation sensor loss is not, the two losses are not expected to be similar in magnitude. Instead, we are interested in their correlation.

Figure 6.9 shows the quantities of interest for tests performed with different sets of hyperparameters. The bottom panel of Figure 6.9b shows that the validation sensor loss follows the training sensor loss faithfully, showing a strong positive linear relationship. In contrast, no strong linear relationship between the validation and training sensor loss for the PC-DualConvNet is observed (Figure 6.9a bottom panel). By comparing how the validation sensor loss changes with the training sensor loss for the two different network structures, we can see that the weight-sharing

network generalises better to unseen regions of the flow, which is the purpose of its design. The strong linear relationship also means that we can assess the generalisation error of the weight-sharing network by assessing the reconstructed flow on known grid points. The same cannot be said for the PC-DualConvNet, since a lower training loss does not often correspond to a smaller validation loss.

The top panels of Figure 6.9a and 6.9b show how the validation sensor loss changes with the physics loss for PC-DualConvNet and the weight-sharing network, respectively. The data points are coloured by the relative error. However, we will not use the relative error during the hyperparameter selection process, because the computation of the relative loss requires the full reference flow field, to which we assume we have no access.

To select the hyperparameters, we consider two lossses, the validation sensor loss and the physics loss. If the measurements are not noisy, we will want the training process to minimise both losses. However, in the case of noisy measurements, the lowest validation loss may not correspond to the most accurate reconstruction, because the loss is computed with the noisy data $\boldsymbol{D}_n$. Without knowing the true flow or the SNR, we also cannot estimate the lower bound for the sensor loss. Therefore, we also cannot set a threshold for the validation sensor loss. On the other hand, we cannot choose the set of hyperparameters which leads to the lowest physics loss, because a low physics loss only shows that the reconstructed flow is a solution to the Navier-Stokes equation, but does not show whether this solution corresponds to the measurements. Instead, we look for a compromise by identifying a point where a decrease in the physics loss leads to an increase in the validation sensor loss, and choose the set of hyperparameters at the turning point. The selected sets of hyperparameters for both networks are marked by a star in Figure 6.9, and the values of the selected hyperparameters are listed in Appendix A.4.3.

6.4.2 Results from noisy measurements

Using the hyperparameters selected in Section 6.4.1, we reconstruct the 3D Kolmogorov flow from measurement planes shown in Section 6.2.1 with added white noise at SNR=15. A summary of the results is shown in Table 6.2, the means and standard deviations are computed over five runs, each with different random initialisation of whitenoise and network weights. Similar to the results from non-noisy measurements in Section 6.3, the weight-sharing network has a lower relative error with a smaller standard deviation, showing that the network becomes less sensitive to the realisation of random noise and weight initialisation by sharing weights.

Figure 6.10 shows the time-averaged 3D velocity and pressure fields. Similar to our observation in Section 6.3, the two networks do not show much difference when comparing the time-averaged flow on the boundaries of the periodic box, as only the boundaries are visible in Figure 6.10. The reconstructed u_3 by the weight-sharing network shows a slice-like artefact in the x_3 direction, which is the result of the weight sharing. However, given that the weight-sharing network

achieved a similar level of physics loss compared to the PC-DualConvNet, the effect of this artefact is minimal.

Table 6.2: The relative error and the physics loss (*mean* ± *standard deviation*) of the reconstruction results from planes of noisy measurements, averaged over five runs with different random noise and different initialisations of network weights.

	ϵ (%)	$\mathscr{L}_p$
Reference data	N/A	0.137±0.000
Weight-sharing network	56.7±0.7	1.244±2.185
PC-DualConvNet	59.4±2.9	1.303±2.156

At $x_3 = 0.5\pi$, which is part of the training data, the reconstructed instantaneous u_1 for both networks is visibly less noisy compared to the noisy measurements in the training set (Figure 6.11), and matches well with the reference data. At the same x_3, the reconstructed pressure has a reduced range (the difference between the largest and smallest values is smaller) in the middle of the domain. Near x_2 =0 and 2π the pressure reconstruction is more accurate because pressure data is available at $x_2 = 0$ and periodic boundary conditions are implied via periodic padding in convolution. Comparing the slices at $x_3 = 0.94\pi$, which is unseen in training, we find that the weight-sharing network better captures the main changes in the flow. Especially in the reconstructed pressure, where the weight-sharing network captured a rotation in the alignment of the two low-pressure areas in the middle of the domain, but not the PC-DualConvNet. These results show that both networks are capable of reconstructing the flow from noisy measurements, producing reasonable instantaneous flow fields and accurate reconstructed mean flow fields. However, the weight-sharing network has proven to be more suitable when the available data is limited to a few planes, as it can generalise to areas of the domain that are away from the measured planes, also shown numerically in Figure 6.12 via a comparison of the reconstruction error of individual $x_1 - x_2$ planes.

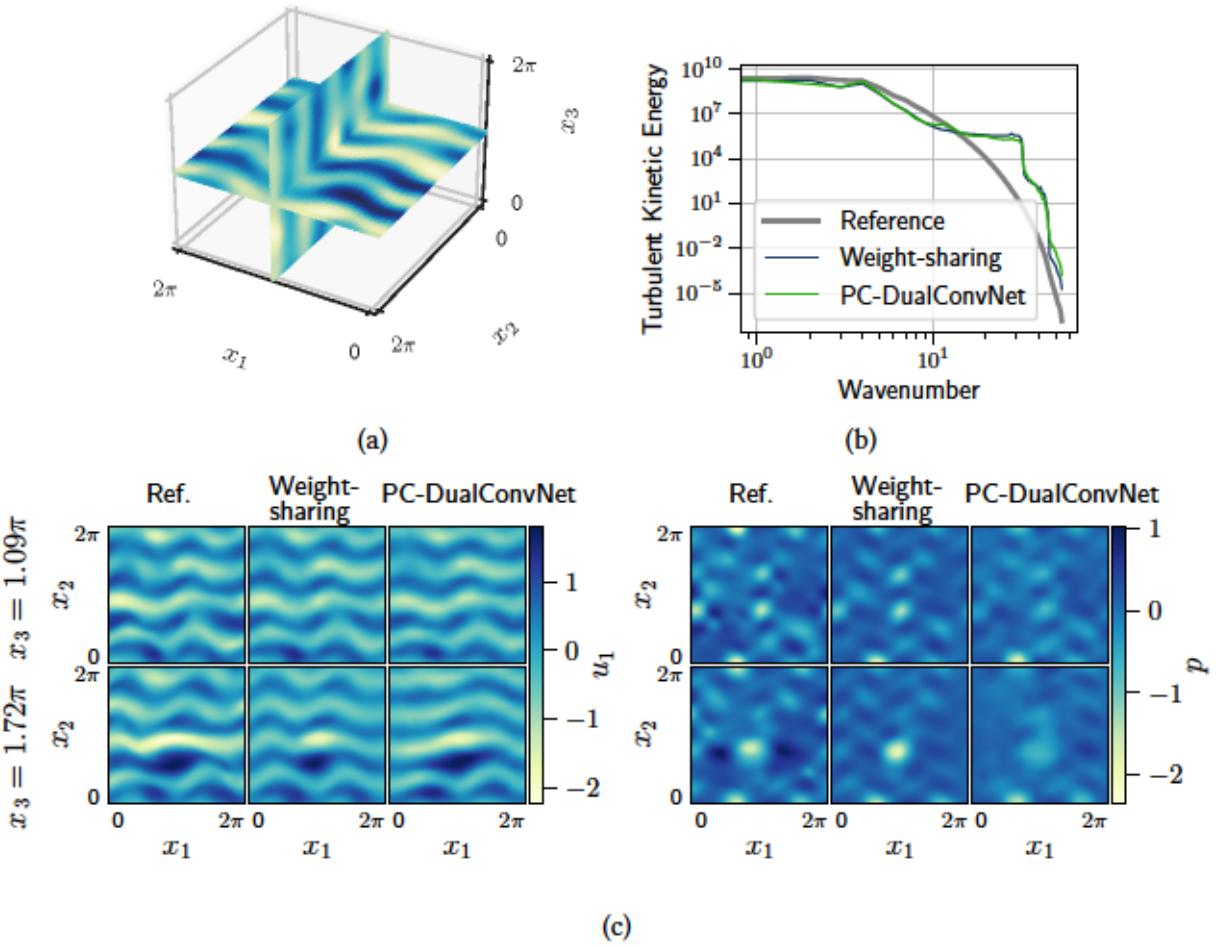


Figure 6.7: Reconstructed flow from a single cross-plane. (a) The locations of the velocity measurement planes, consisting of one $x_1 - x_2$ measurement plane and a one $x_2 - x_3$ measurement plane, at the centre of the domain. (b) The turbulent kinetic energy of the reconstructed flow fields. (c) Two slices of the reconstructed flow at $x_3 = 1.09\pi$ and 1.72π, which are unseen by the network during training. The measurements used in training are taken on planes at $x_3 = \pi$. When further away from the measured plane, such as the bottom row at $x_3 = 1.72\pi$, the flow reconstructed by the weight-sharing network match the reference data well, while the flow reconstructed by the PC-DualConvNet has lost resemblance to the reference. The difference is especially true for the pressure field.

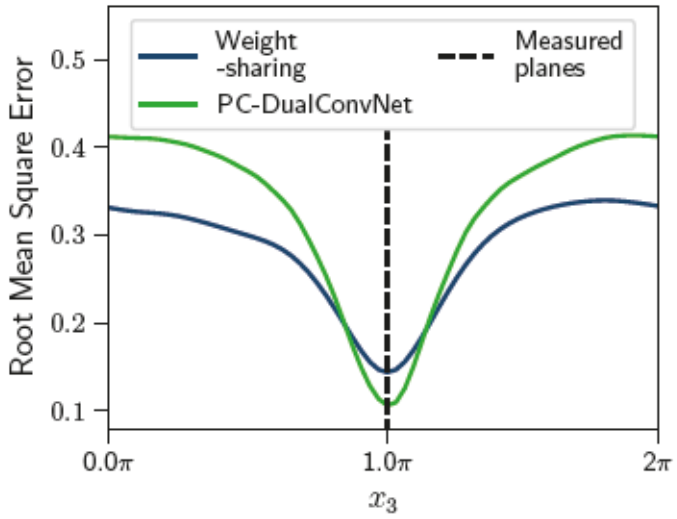


Figure 6.8: The root mean square error (sqaure root of MSE) of of each reconstructed $x_1 - x_2$ plane.

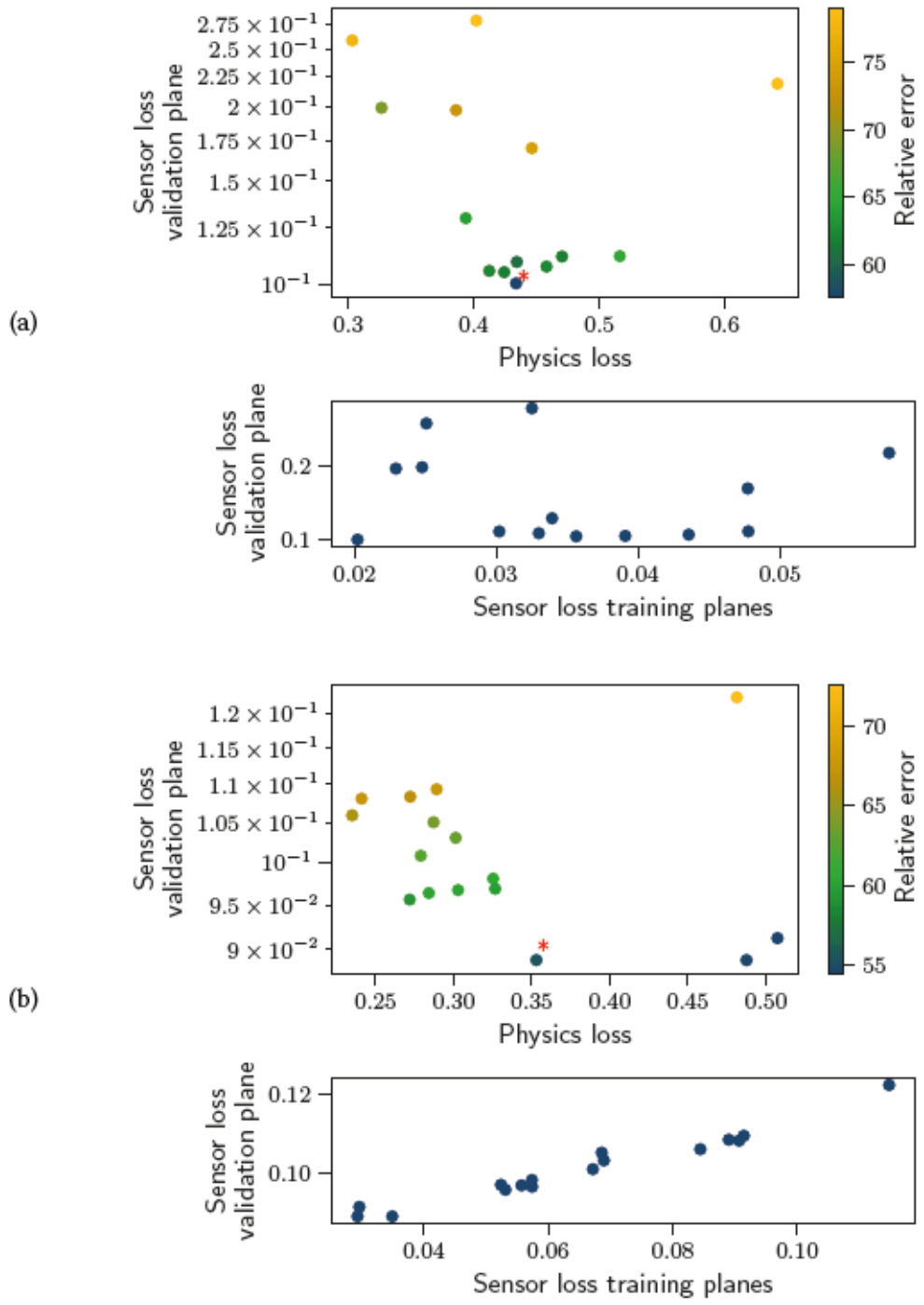


Figure 6.9: Quantities of interest for tests performed with different sets of hyperparameters for (a) the 3D PC-DualConvNet and (b) the weight-sharing network. Each data point represents a test with a distinct set of hyperparameters. Top panel: the validation sensor loss plotted against the physics loss for tests with different sets of hyperparameters, coloured by the relative error (not used in the selection process). The red star marks the set of hyperparameters selected. Validation sensor loss is computed at the plane at $x_3 = \pi$, which is not seen by the networks during training. Bottom panel: validation sensor loss against the training sensor loss.

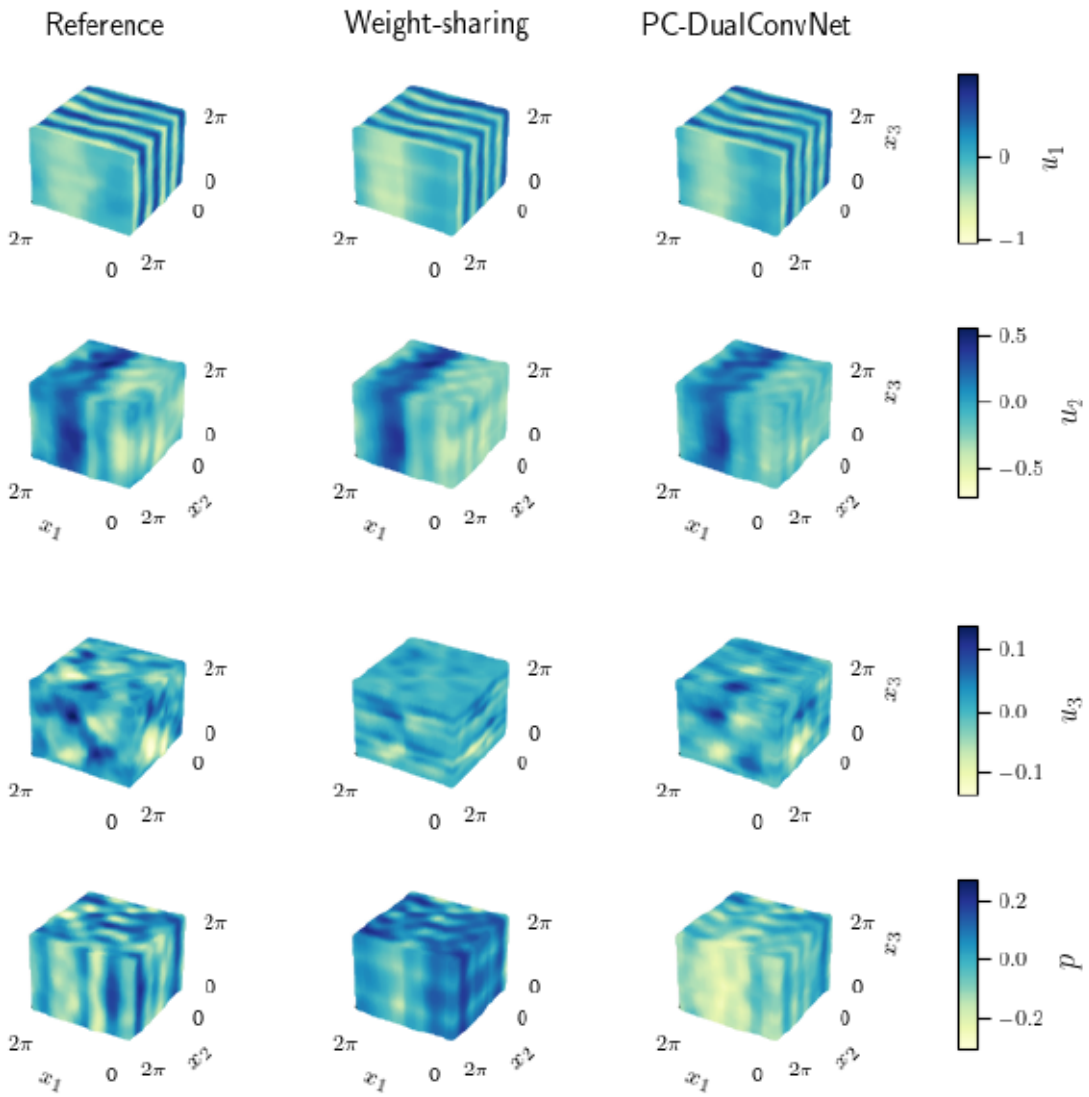


Figure 6.10: The mean flow. From left to right are: reference 3D data, reconstructed with the weight-sharing network, and reconstructed with the PC-DualConvNet. The top and bottom rows are the vorticity and pressure, respectively.

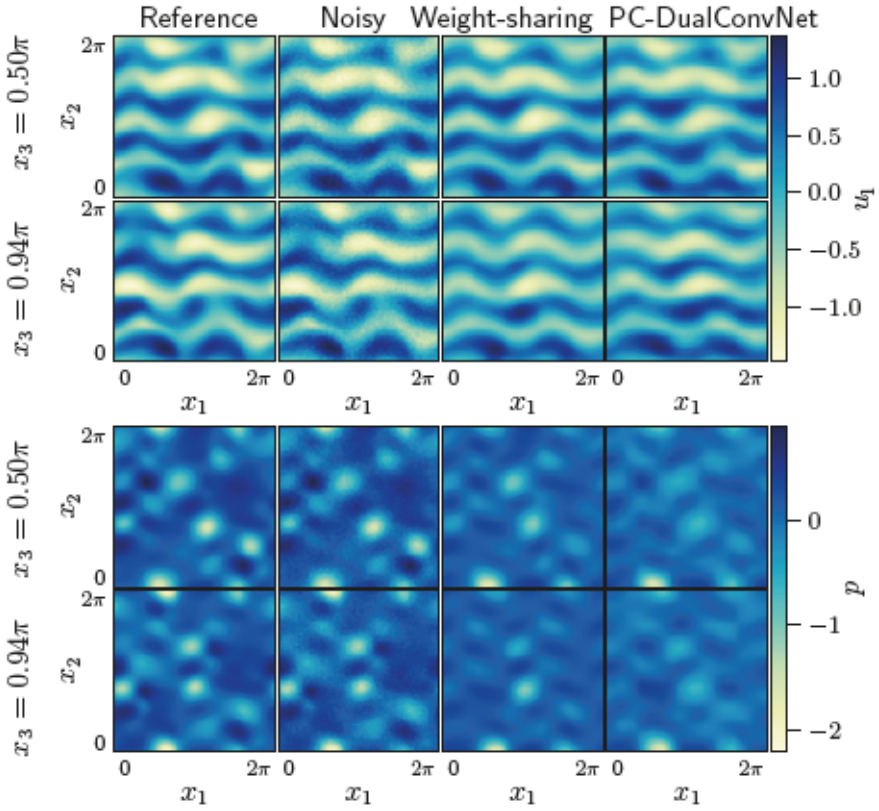


Figure 6.11: An instantaneous velocity u_1 and pressure p at different x_3. Columns show the reference flow, the flow with added white noise, the flow reconstructed by the weight-sharing network and the flow reconstructed by the PC-DualConvNet (left to right). The top row for each variable shows a plane at $x_3 = 0.5\pi$, which is part of the training dataset. The bottom row for each variable shows a plane at $x_3 = 0.94\pi$, which is unseen by the network during training, and also not a validation plane used for selecting the hyperparameters in Section 6.4.1.

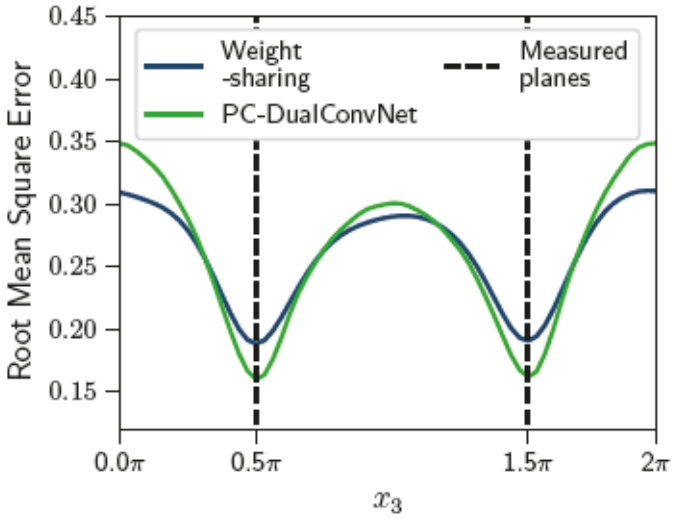


Figure 6.12: The root mean square error (sqaure root of MSE) of of each reconstructed $x_1 - x_2$ plane. Reconstructed from noisy measurement planes.

6.5 Conclusion

In this chapter, we reconstruct 3D turbulent Kolmogorov flows from three measurement planes with a weight-sharing network to recover the full flow field. Two of the measurement planes are parallel to each other, while the third plane is perpendicular to the other two planes. No ground truth data of the full flow field is available during training. The weight-sharing network applies identical network parameters (the shared weights) along a homogeneous direction as part of the forward pass. Sharing weights along the homogeneous direction both reduces the memory requirement of the network and provides a more efficient use of data. First, we reconstruct the 3D Kolmogorov flow from non-noisy measurements and compare the results obtained with the PC-DualConvNet and the weight-sharing network. Both networks are found to accurately reconstruct time-averaged 3D flow fields and recover the correct energy spectrum up to wavenumber 10, which contains most of the flow energy (Chapter 5). The weight-sharing network achieved a relative error of just under 50% while the PC-DualConvNet achieved a relative error of approximately 55%. However, the PC-DualConvNet has a lower sensor error compared to the weight-sharing network, but has a larger relative error, showing that the PC-DualConvNet has overfitted to the measurements. The analysis of instantaneous slices of the flow domain also shows that the weight-sharing network can reconstruction flow structures even in areas away from the measured plane. Then, we reconstruct the flow from the same measurement planes but with added white noise at SNR=15 using the mean-enforced loss. As part of hyperparameter selection, we compute a validation sensor loss on a plane unseen in training and show that the validation sensor loss decreases with the training sensor loss for the weight-sharing network. However, the validation sensor loss does not follow any trend with the training sensor loss of the PC-DualConvNet. Therefore, we conclude that the weight-sharing network generalises better to the unseen regions of the flow, and the training sensor loss is a good estimate of the generalisation error for the weight-sharing network. By identifying that the training sensor loss can be used to estimate generalisation error, we allow more data to be used in training instead of validation, which is helpful when data is limited. The relative errors for the reconstructed flow from noisy measurements are approximately 10% higher than their non-noisy counterparts. Qualitative analysis shows that using noisy measurements as inputs does not significantly impact the reconstructed flow. In summary, we find that both the PC-DualConvNet and the weight-sharing network can reconstruct 3D turbulent Kolmogorov flows from plane measurements, a setup similar to that expected of an experiment. However, the weight-sharing network can generalise to the unseen region of the flow, and the training sensor loss can be used to estimate the generalisation error, while reducing the memory requirements by reducing the number of parameters.

Chapter 7

Conclusion and future work

The study of turbulent flows relies on data collected through either experiments or simulations. The process of investigation usually starts with raw data, which may be incomplete, and ends with a model ready for deployment in an engineering problem. This thesis explores combining flow physics and machine learning (ML) to improve the data-to-result pipeline.

7.1 Summary of work

This thesis can be organised into efforts to tackle two problems: how can we use ML to analyse flow data, and can we reconstruct flows when only sparse, noisy data is available?

Flow analysis with ML

Chapter 3 focuses on analysing bluff body wakes, including an experimental dataset obtained in wind tunnel, with ML. In particular, the chapter deals with the interpretability of using autoencoders as a method for nonlinear model order reduction. In our work, we propose the decoder decomposition to interpret and rank latent variables. The decoder decomposition is a post-processing method used to analyse the sensitivity of the POD modes of the data to the latent variables, thus making the interpretation of POD modes applicable to nonlinear autoencoders. By analysing the latent space of an unsteady laminar cylinder wake using different latent dimensions, we show that the physical interpretation of the latent variables depends heavily on the latent dimension. An unsuitable latent dimension leads the autoencoder to produce spurious latent variables. We then apply the decoder decomposition on autoencoders trained on a wind-tunnel bluff body wake dataset and show that some benefit of increasing the latent dimension is due to the increase in the size of the decoder, not the increase in the latent dimension. We also use the decoder decomposition to demonstrate filtering for latent variables in a way that the output of the autoencoder contains mainly the coherent structures associated with vortex shedding. This method opens up opportunities for a more interpretable nonlinear dimensionality reduction with autoencoders.

Flow reconstruction with ML

From Chapter 4 onwards, we focus on the reconstruction of flows from sparse measurements. In this thesis, we design neural networks to take sparse pressure measurements as input. The networks then output the entire flow field, including both pressure and velocity. During training, the networks see only the sparse measurements as collocation points, but no ground truth data is available.

In Chapter 4, we propose a physics-constrained dual-branch convolutional neural network (PC-DualConvNet) to reconstruct 2D flows from sparse measurements. The test cases are a 2D laminar unsteady wake of a triangular body and a 2D weakly turbulent Kolmogorov flow. We also explore the different designs of loss functions using different levels of physical constraints, and their effects on the reconstructed flow. When the sparse measurements are not noisy, we compare the reconstruction using the softly-constrained loss, which is a weighted sum of the physics and the sensor loss, and the snapshot-enforced loss, which places a hard constraint on the sensor measurements. We find that the snapshot-enforced loss eliminates the possibility of trivial solutions, which was a problem in the softly-constrained loss. Using the snapshot-enforced loss, we reconstruct both the laminar wake and the chaotic Kolmogorov flow from sensors placed on less than 1% of the grid points.

When the measurements are both sparse and noisy, we proposed the mean-enforced loss, which strictly enforces the time-average of the reconstruction at the sensor locations to be equal to the time-average of the measurements. We show that the mean-enforced loss allows for better denoising when the signal-to-noise ratio (SNR) is low. We are able to achieve a relative error under 25% when reconstructing from sparse measurements corrupted with white noise at SNR=5.

Chapter 5 tests the PC-DualConvNet further by applying it to reconstruct two more realistic flow cases. The first new case is a 2D Kolmogorov flow exhibiting extreme dissipation events. The PC-DualConvNet is able to reconstruct this case from sensors placed on approximately 1% of the grid points without any modification. The second new case is a set of 3D turbulent Kolmogorov flows. With minimum modification to the PC-DualConvNet, such as replacing the 2D convolution with a 3D convolution and adding periodic padding to convolutions to represent the periodic boundary conditions, we reconstruct the 3D flow from sensors placed randomly at approximately 3% of the grid points.

Chapter 6 focuses on reconstructing the 3D Kolmogorov dataset from a more realistic sensor setup than velocity and pressure sensors placed randomly within the domain. In Chapter 6, the pressure measurements are taken from a plane on the boundary only, to reflect the fact that inflow pressure measurements are difficult to obtain experimentally. The velocity measurements are taken from two planes parallel to each other, and another plane perpendicular to the previous two planes, all within the domain. We design a weight-sharing network to reconstruct the 3D

Kolmogorov flows from planes of measurements to take advantage of the homogeneous directions present in the flow. We find that the weight-sharing network can generalise to the unseen region of the flow, and its generalisation error can be estimated with the training error, making it more reliable when our measurements are concentrated on a few planes within the flow.

Overall, we have reconstructed flows from a 2D flow and non-noisy measurements to the more realistic 3D turbulent flows and noisy measurements. Throughout this thesis, we focus on reconstruction without access to ground truth measurements, which is key to real-world applications of ML-based flow reconstruction.

7.2 Future work

This thesis focuses on the development of methodologies which can be applied to real-world flows. However, more work is needed to make the methods more scalable and efficient.

Improvement to algorithm

Currently, in the codes for flow reconstruction, the slowest and most memory-consuming part is computing momentum. The issue is not noticeable when reconstructing 2D flow, but computing the momentum causes a significant increase in the update time when working with 3D data. The derivatives in the momentum equations are computed using fourth-order central difference, which requires multiple large matrices to be stored during each update. The memory taken up by the momentum calculation also restricts the maximum number of snapshots per batch on a single GPU, which may impact the performance of the networks. Different numerical schemes should be considered to reduce computation time, thus improving the overall training time. Implementing distributed training may also provide performance gain, especially if the next step is to consider larger datasets for more complicated flow.

Another area of improvement in the neural networks is the implementation of boundary conditions. If known, boundary conditions may be hard-coded into the neural networks to ensure the reconstructed flow capture the correct physics near the boundaries. For conditions such periodic boundary conditions, which enforce smoothness across boundaries, the boundary conditions must be imposed in all layers within the network to avoid introduction of local discontinuity inside the network.

Test the modes selected for building ROM

In Chapter 3, we propose the decoder decomposition to rank and select latent variables. The ability to rank and select variables is crucial to building reduced-order models. There have been studies to model and propagate latent variables of time series by a combination of AEs and other memory-retaining algorithms (Hasegawa et al. 2020, Racca et al. 2023, Özalp & Magri 2025). A natural extension of Chapter 3 would be to apply the decoder decomposition to select the latent variables before propagating them for time series forecasting. For example, from a large

trained AE, we may wish to futher reduce the number of necessary latent variables before feeding the latent variables into the time-propagating algorithm, without losing too much accuracy or changing the coherent structures in the reconstructed flow. In this case, we may rank the latent variables based on their correlations with the leading POD modes and keep only the high-ranking latent variables.

Reconstruction of experimental flows

Even though this thesis works with both experimental and synthetic datasets, the datasets used for the flow reconstruction part of are all generated with numerical simulations. Chapter 6 focuses on a more realistic case where the sensors are noisy and set up in a way similar to experiments. The next step is perform flow reconstruction with real experimental data, which may lead to problems such as smaller numbers of available measurements, different flavours of noise, and larger datasets and memory requirements in general. The first step to tackle the memory issues is to improve the efficiency of our algorithm, which has already been discussed earlier in this chapter. To address the potential issue with limited measurements, we may start by re-designing the network to constrain the boundary conditions if possible, and apply transfer learning or augment the experimental data with simulations.

Other flows

In the flow reconstruction part of this thesis, we have also primarily focused on canonical flows. Specifically, laminar bluff body wakes and turbulent Kolmogorov flows. While these flows are important test cases for the study of turbulence, we should also test our methods on other practical flows, such as turbulent bluff body wakes, which we have touched on in Chapter 3. Using different types of flow will also change the symmetries within the flow, which we may be able to use to our advantage. For example, for flows which exhibit rotational symmetry, like the turbulent wake of a bullet-shaped body we used in Chapter 3 (Rowan D. Brackston 2017), we may consider Group Equivalent Convolutional Networks, which have extended the translational invariance of normal CNNs to rotations (Cohen & Welling 2016).

At the same time, this thesis focuses on structured data, while many available datasets may not be, e.g., data simulated with an unstructured mesh. Other types of networks, such as the Graph Neural Network, where data points are considered nodes with different distances between them, may be more suitable (Ogoke et al. 2021). Careful pre-processing, such as Voronoi tessellation used by Fukami, Maulik, Ramachandra, Fukagata & Taira (2021), could also be used to prepare unstructured data.

Transformer for global pattern recognition

Convolutions can extract local information captured by the size of the filter. However, transformers have been known to capture global information because of their use of the 'attention' algorithm. Attention, including self and cross attention, has the ability to learn correlations be-

tween far-apart data points. Transformer-based neural networks are seeing an increase in applications to flow reconstruction tasks with success (Kang et al. 2023, Santos et al. 2023, McDugald et al. 2024). Our flow reconstruction tasks may also benefit from the global pattern recognition of the self-attention, and cross-attention between different variables, such as velocity and pressure, may be helpful since we have fewer pressure measurements than velocity measurements (setup in Chapter 6).

Reconstruction of missing time segments

This thesis performs flow reconstruction to recover the missing flow field in the unseen region of the flow. For the next step, we should include recovering time segments for consideration. We will consider a realistic condition where we have more snapshots of pressure measurements at the boundary or on a surface than in-flow velocity measurements. At instances when only pressure input measurements are available, the network will run in inference mode. Therefore, we need to include generalising to missing time segments into our criteria of success when we develop our flow reconstruction methods further.

Bibliography

Agrawal, N., Dutta, S. & Gandhi, B. K., 2013. An experimental study on flow past an equilateral triangular prism at intermediate reynolds number and the effect of its orientation, *in* 'Volume 1B, Symposia: Fluid Machinery; Fluid Power; Fluid-Structure Interaction and Flow-Induced Noise in Industrial Applications; Flow Applications in Aerospace; Flow Manipulation and Active Control: Theory, Experiments and Implementation; Fundamental Issues and Perspectives in Fluid Mechanics', American Society of Mechanical Engineers, Incline Village, Nevada, USA.

Albukrek, C. M., Urban, K., Rempferr, D. & Lumley, J. L., 2002. 'Divergence-free wavelet analysis of turbulent flows', *Journal of Scientific Computing* **17**(1-4), 49–66.

Alfonsi, G. & Primavera, L., 2007. 'The structure of turbulent boundary layers in the wall region of plane channel flow', *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **463**(2078), 593–612.

Arzani, A., Wang, J.-X. & D'Souza, R. M., 2021. 'Uncovering near-wall blood flow from sparse data with physics-informed neural networks', *Physics of Fluids* **33**(7), 071905.

Baldi, P. & Hornik, K., 1989. 'Neural networks and principal component analysis: Learning from examples without local minima', *Neural Networks* **2**, 53–58.

Bansal, N., Chen, X. & Wang, Z., 2018. Can We Gain More from Orthogonality Regularizations in Training Deep Networks?, *in* 'Advances in Neural Information Processing Systems', Vol. 31, Curran Associates, Inc., pp. 4261–4271.

Bartholomew, P., Laizet, S., Schuch, F. N., Frantz, R. A. S., Rolfo, S., Hamzehloo, A., Flageul, C., Deskos, G., Bempedelis, N., Schäfer, K., nasos94, Jing, H., vcz18385 & Monteiro, L. R., 2022. 'Xcompact3d/Incompact3d:', Zenodo.

Beneddine, S., Yegavian, R., Sipp, D. & Leclaire, B., 2017. 'Unsteady flow dynamics reconstruction from mean flow and point sensors: An experimental study', *Journal of Fluid Mechanics* **824**, 174–201.

Blossey, P. N. & Lumley, J. L., 2000. Control of intermittency in near-wall turbulent flow, *in* J. C. Vassilicos, ed., 'Intermittency in Turbulent Flows', Cambridge University Press, Cambridge, UK, pp. 1–23.

Boffetta, G. & Ecke, R. E., 2012. 'Two-Dimensional Turbulence', *Annual Review of Fluid Mechanics* **44**(1), 427–451.

Borue, V. & Orszag, S. A., 1996. 'Numerical study of three-dimensional Kolmogorov flow at high Reynolds numbers', *Journal of Fluid Mechanics* **306**, 293–323.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. & Shah, R., 1993. Signature Verification using a "Siamese" Time Delay Neural Network, *in* 'Advances in Neural Information Processing Systems', Vol. 6, Morgan-Kaufmann, pp. 737–744.

Brunton, S. L. & Kutz, J. N., 2024. 'Promising directions of machine learning for partial differential equations', *Nature Computational Science* **4**(7), 483–494.

Buzzicotti, M., Bonaccorso, F., Di Leoni, P. C. & Biferale, L., 2021. 'Reconstruction of turbulent data with deep generative models for semantic inpainting from TURB-Rot database', *Physical Review Fluids* **6**(5), 050503.

Cal, R. B., Lebrón, J., Castillo, L., Kang, H. S. & Meneveau, C., 2010. 'Experimental study of the horizontally averaged

flow structure in a model wind-turbine array boundary layer', *Journal of Renewable and Sustainable Energy* **2**(1), 013106.

Callaham, J. L., Maeda, K. & Brunton, S. L., 2019. 'Robust flow reconstruction from limited measurements via sparse representation', *Physical Review Fluids* **4**(10).

Carini, M., Auteri, F. & Giannetti, F., 2015. 'Centre-manifold reduction of bifurcating flows', *Journal of Fluid Mechanics* **767**, 109–145.

Cato, A. S., Volpiani, P. S., Mons, V., Marquet, O. & Sipp, D., 2023. 'Comparison of different data-assimilation approaches to augment RANS turbulence models', *Computers & Fluids* **266**, 106054.

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

Chandramouli, P., Memin, E., Heitz, D. & Fiabane, L., 2019. 'Fast 3D flow reconstructions from 2D cross-plane observations', *Experiments in Fluids* **60**(2), 30.

Chatzimanolakis, M., Weber, P. & Koumoutsakos, P., 2024. 'Learning in two dimensions and controlling in three: Generalizable drag reduction strategies for flows past circular cylinders through deep reinforcement learning', *Physical Review Fluids* **9**(4), 043902.

Cheng, S., Quilodrán-Casas, C., Ouala, S., Farchi, A., Liu, C., Tandeo, P., Fablet, R., Lucor, D., Iooss, B., Brajard, J., Xiao, D., Janjic, T., Ding, W., Guo, Y., Carrassi, A., Bocquet, M. & Arcucci, R., 2023. 'Machine Learning With Data Assimilation and Uncertainty Quantification for Dynamical Systems: A Review', *IEEE/CAA Journal of Automatica Sinica* **10**(6), 1361–1387.

Clark Di Leoni, P., Agarwal, K., Zaki, T. A., Meneveau, C. & Katz, J., 2023. 'Reconstructing turbulent velocity and pressure fields from under-resolved noisy particle tracks using physics-informed neural networks', *Experiments in Fluids* **64**(5), 95.

Cohen, K., Siegel, S. & McLaughlin, T., 2003. Sensor Placement Based on Proper Orthogonal Decomposition Modeling of a Cylinder Wake, *in* '33rd AIAA Fluid Dynamics Conference and Exhibit', American Institute of Aeronautics and Astronautics, Orlando, Florida.

Cohen, T. S. & Welling, M., 2016. 'Group Equivariant Convolutional Networks'.

Colanera, A. & Magri, L., 2025. 'Quantized local reduced-order modeling in time (ql-ROM)'.

Colburn, C. H., Cessna, J. B. & Bewley, T. R., 2011. 'State estimation in wall-bounded flow systems. Part 3. The ensemble Kalman filter', *Journal of Fluid Mechanics* **682**, 289–303.

Csala, H., Dawson, S. T. M. & Arzani, A., 2022. 'Comparing different nonlinear dimensionality reduction techniques for data-driven unsteady fluid flow modeling', *Physics of Fluids* **34**(11), 117119.

Cuéllar, A., Güemes, A., Ianiro, A., Flores, Ó., Vinuesa, R. & Discetti, S., 2024. 'Three-dimensional generative adversarial networks for turbulent flow estimation from wall measurements', *Journal of Fluid Mechanics* **991**, A1.

Deng, Z., He, C., Wen, X. & Liu, Y., 2018. 'Recovering turbulent flow field from local quantity measurement: Turbulence modeling using ensemble-Kalman-filter-based data assimilation', *Journal of Visualization* **21**(6), 1043–1063.

Doan, N. A. K., Polifke, W. & Magri, L., 2021. Auto-encoded reservoir computing for turbulence learning, *in* 'Computational Science – ICCS 2021', Springer International Publishing, Cham, pp. 344–351.

Doan, N. A. K., Racca, A. & Magri, L., 2023. Convolutional Autoencoder for the Spatiotemporal Latent Representation of Turbulence, *in* J. Mikyška, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra & P. M. Sloot, eds, 'Computational Science – ICCS 2023', Lecture Notes in Computer Science, Springer Nature Switzerland, Cham, pp. 328–335.

Druault, P. & Chaillou, C., 2007. 'Use of Proper Orthogonal Decomposition for reconstructing the 3D in-cylinder

mean-flow field from PIV data', *Comptes Rendus Mécanique* **335**(1), 42–47.

Dubois, P., Gomez, T., Planckaert, L. & Perret, L., 2022. 'Machine learning for fluid flow reconstruction from limited measurements', *Journal of Computational Physics* **448**, 110733.

Eivazi, H., Clainche, S. L., Hoyas, S. & Vinuesa, R., 2022. 'Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows', *Expert Systems with Applications* **202**, 117038.

Eivazi, H., Wang, Y. & Vinuesa, R., 2024. 'Physics-informed deep-learning applications to experimental fluid mechanics', *Measurement Science and Technology* **35**(7), 075303.

Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W. & Kutz, J. N., 2020. 'Shallow neural networks for fluid flow reconstruction with limited sensors', *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **476**(2238).

Everson, R. & Sirovich, L., 1995. 'Karhunen–Loève procedure for gappy data', *Journal of The Optical Society of America* **12**, 1657–1664.

Fahlman, S. & Lebiere, C., 1989. The Cascade-Correlation Learning Architecture, *in* 'Advances in Neural Information Processing Systems', Vol. 2, Morgan-Kaufmann.

Farano, M., Cherubini, S., Robinet, J.-C. & Palma, P. D., 2017. 'Optimal bursts in turbulent channel flow', *Journal of Fluid Mechanics* **817**, 35–60.

Farazmand, M. & Sapsis, T. P., 2017. 'A variational approach to probing extreme events in turbulent dynamical systems', *Science Advances* **3**(9), e1701533.

Franceschini, L., Sipp, D. & Marquet, O., 2020. 'Mean-flow data assimilation based on minimal correction of turbulence models: Application to turbulent high Reynolds number backward-facing step', *Physical Review Fluids* **5**(9), 094603.

Franceschini, L., Sipp, D. & Marquet, O., 2021. 'Mean- and unsteady-flow reconstruction with one or two time-resolved measurements', *arXiv* .

Frisch, U., 1995. *Turbulence: The Legacy of A.N. Kolmogorov*, Cambridge University Press, Cambridge, UK.

Fukami, K., Fukagata, K. & Taira, K., 2019. 'Super-resolution reconstruction of turbulent flows with machine learning', *Journal of Fluid Mechanics* **870**, 106–120.

Fukami, K., Fukagata, K. & Taira, K., 2021. 'Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows', *Journal of Fluid Mechanics* **909**.

Fukami, K., Fukagata, K. & Taira, K., 2023. 'Super-resolution analysis via machine learning: A survey for fluid flows', *Theoretical and Computational Fluid Dynamics* .

Fukami, K., Hasegawa, K., Nakamura, T., Morimoto, M. & Fukagata, K., 2021. 'Model order reduction with neural networks: Application to laminar and turbulent flows', *SN Computer Science* **2**(6), 467.

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

Fukami, K., Nakamura, T. & Fukagata, K., 2020. 'Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data', *Physics of Fluids* **32**(9), 095110.

Fukami, K., Nakao, H. & Taira, K., 2024. 'Data-driven transient lift attenuation for extreme vortex gust-airfoil interactions'.

Fukami, K. & Taira, K., 2023. 'Grasping extreme aerodynamics on a low-dimensional manifold', *Nature Communications* **14**(1), 6480.

Fukami, K. & Taira, K., 2025. 'Observable-augmented manifold learning for multi-source turbulent flow data', *Journal*

of Fluid Mechanics **1010**, R4.

Fylladitakis, E. D., 2018. 'Kolmogorov Flow: Seven Decades of History', *Journal of Applied Mathematics and Physics* **6**(11), 2227–2263.

Ganapathisubramani, B., Longmire, E. K., Marusic, I. & Pothos, S., 2005. 'Dual-plane PIV technique to determine the complete velocity gradient tensor in a turbulent boundary layer', *Experiments in Fluids* **39**(2), 222–231.

Ganga Prasath, S., Sudharsan, M., Vinodh Kumar, V., Diwakar, S., Sundararajan, T. & Tiwari, S., 2014. 'Effects of aspect ratio and orientation on the wake characteristics of low Reynolds number flow over a triangular prism', *Journal of Fluids and Structures* **46**, 59–76.

Gao, H., Sun, L. & Wang, J.-X., 2021. 'Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels', *Physics of Fluids* **33**(7), 073603.

Geir Evensen, 2009. *Data Assimilation : The Ensemble Kalman Filter*, 2 edn, Springer Berlin / Heidelberg.

Ghani, A., Poinsot, T., Gicquel, L. & Staffelbach, G., 2015. 'LES of longitudinal and transverse self-excited combustion instabilities in a bluff-body stabilized turbulent premixed flame', *Combustion and Flame* **162**(11), 4075–4083.

Greydanus, S., Dzamba, M. & Yosinski, J., 2019. Hamiltonian Neural Networks, *in* 'Advances in Neural Information Processing Systems', Vol. 32, Curran Associates, Inc., Vancouver, Canada.

Guastoni, L., Güemes, A., Ianiro, A., Discetti, S., Schlatter, P., Azizpour, H. & Vinuesa, R., 2021. 'Convolutional-network models to predict wall-bounded turbulence from wall quantities', *Journal of Fluid Mechanics* **928**, A27.

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

Gunes, H., Sirisup, S. & Karniadakis, G. E., 2006. 'Gappy data: To Krig or not to Krig?', *Journal of Computational Physics* **212**(1), 358–382.

Hamdi, J., Assoum, H., Abed-Meraïm, K. & Sakout, A., 2018. 'Volume reconstruction of an impinging jet obtained from stereoscopic-PIV data using POD', *European Journal of Mechanics - B/Fluids* **67**, 433–445.

Hasegawa, K., Fukami, K., Murata, T. & Fukagata, K., 2020. 'Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes', *Theoretical and Computational Fluid Dynamics* **34**(4), 367–383.

He, C., Li, S. & Liu, Y., 2025. 'Data assimilation: New impetus in experimental fluid dynamics', *Experiments in Fluids* **66**(5), 94.

Holmes, P., Lumley, J. L., Berkooz, G. & Rowley, C. W., 2012*a*. Coherent structures, *in* 'Turbulence, Coherent Structures, Dynamical Systems and Symmetry', 2 edn, Cambridge Monographs on Mechanics, Cambridge University Press, Cambridge, pp. 17–67.

Holmes, P., Lumley, J. L., Berkooz, G. & Rowley, C. W., 2012*b*. Galerkin projection, *in* 'Turbulence, Coherent Structures, Dynamical Systems and Symmetry', 2 edn, Cambridge Monographs on Mechanics, Cambridge University Press, Cambridge, pp. 106–129.

Huhn, F. & Magri, L., 2020. 'Stability, sensitivity and optimisation of chaotic acoustic oscillations', *Journal of Fluid Mechanics* **882**, A24.

Jaderberg, M., Simonyan, K., Zisserman, A. & kavukcuoglu, k., 2015. Spatial Transformer Networks, *in* 'Advances in Neural Information Processing Systems', Vol. 28, Curran Associates, Inc.

Kang, H., Kim, Y., Le, T.-T.-H., Choi, C., Hong, Y., Hong, S., Chin, S. W. & Kim, H., 2023. 'A new fluid flow approximation method using a vision transformer and a U-shaped convolutional neural network', *AIP Advances* **13**(2), 025233.

Karnakov, P., Litvinov, S. & Koumoutsakos, P., 2023. 'Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks', *PNAS Nexus* **3**(1), pgae005.

Kelshaw, D. & Magri, L., 2024. 'Physics-constrained convolutional neural networks for inverse problems in spatiotemporal partial differential equations', *Data-Centric Engineering* **5**, e43.

Kelshaw, D. & Magri, L., 2025. 'Computing distances and means on manifolds with a metric-constrained Eikonal approach', *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **481**(2312), 20240270.

Kelshaw, D., Rigas, G. & Magri, L., 2022. 'Physics-Informed CNNs for Super-Resolution of Sparse Observations on Dynamical Systems'.

Kelshaw, Daniel, 2023. 'MagriLab/KolSol: Pseudospectral Kolmogorov Flow Solver'.

Kevlahan, N. K. R., Hunt, J. C. R. & Vassilicos, J. C., 1994. 'A comparison of different analytical techniques for identifying structures in turbulence', *Applied Scientific Research* **53**(3-4), 339–355.

Kim, H., Kim, J., Won, S. & Lee, C., 2021. 'Unsupervised deep learning for super-resolution reconstruction of turbulence', *Journal of Fluid Mechanics* **910**, A29.

Kim, J., Jung, W., Kim, H. & Lee, J., 2020. 'CyCNN: A Rotation Invariant CNN using Polar Mapping and Cylindrical Convolution Layers'.

Kneer, S., Sayadi, T., Sipp, D., Schmid, P. & Rigas, G., 2021. 'Symmetry-aware autoencoders: S-PCA and s-nlPCA', *arXiv* .

Laizet, S. & Lamballais, E., 2009. 'High-order compact schemes for incompressible flows: A simple and efficient method with quasi-spectral accuracy', *Journal of Computational Physics* **228**(16), 5989–6015.

Le Dimet, F.-X. & Talagrand, O., 1986. 'Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects', *Tellus A: Dynamic Meteorology and Oceanography* **38**(2), 97.

Li, T., Buzzicotti, M., Biferale, L., Bonaccorso, F., Chen, S. & Wan, M., 2023. 'Multi-scale reconstruction of turbulent rotating flows with proper orthogonal decomposition and generative adversarial networks', *Journal of Fluid Mechanics* **971**, A3.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. & Anandkumar, A., 2021. 'Fourier Neural Operator for Parametric Partial Differential Equations'.

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

Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A. & Yosinski, J., 2018. An intriguing failing of convolutional neural networks and the CoordConv solution, *in* 'Advances in Neural Information Processing Systems', Vol. 31, Curran Associates, Inc.

Lohse, D. & Grossmann, S., 1993. 'Intermittency in turbulence', *Physica A: Statistical Mechanics and its Applications* **194**(1), 519–531.

Loiseau, J.-C., Brunton, S. L. & Noack, B. R., 2020. From the POD-Galerkin method to sparse manifold models, *in* P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders & L. M. Silveira, eds, 'Volume 3 Applications', Volume 3 Applications, De Gruyter, Berlin, Boston, pp. 279–320.

Loshchilov, I. & Hutter, F., 2017. SGDR: Stochastic Gradient Descent with Warm Restarts, *in* 'International Conference on Learning Representations'.

Lumley, J. L., 1970. Stochastic tools in turbulence, *in* 'Stochastic Tools In Turbulence', Applied Mathematics and Mechanics, v. 12, Academic Press, New York.

M. M. Zdravkovich, 1997. *Flow Around Circular Cylinders Vol 1: Fundamentals*, Vol. 1, 1 edn, Oxford Univerisity

Press, Oxford.

Magri, L. & Doan, N.A.K, 2022. 'On interpretability and proper latent decomposition of autoencoders', *Center for Turbulence Research Proceedings of the Summer Program 2022* .

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

Matsuo, M., Fukami, K., Nakamura, T., Morimoto, M. & Fukagata, K., 2024. 'Reconstructing Three-Dimensional Bluff Body Wake from Sectional Flow Fields with Convolutional Neural Networks', *SN Computer Science* 5(3), 306.

Maulik, Romit, Fukami, Kai, Ramachandra, Nesar, Fukagata, Koji & Taira, Kunihiko, 2020. 'Probabilistic neural networks for fluid flow surrogate modeling and data recovery', *Physical Review Fluids* 5.

Mazzilli, C. E. N., Gonçalves, P. B. & Franzini, G. R., 2022. 'Reduced-order modelling based on non-linear modes', *International Journal of Mechanical Sciences* **214**, 106915.

McDugald, E., Mohan, A., Engwirda, D., Marcato, A. & Santos, J., 2024. 'Machine learned reconstruction of tsunami dynamics from sparse observations'.

McKeon, B. & Engler, R., 2007. Pressure Measurement Systems, *in* C. Tropea, A. L. Yarin & J. F. Foss, eds, 'Springer Handbook of Experimental Fluid Mechanics', Springer, Berlin, Heidelberg, pp. 179–214.

Mehr, É., Lieutier, A., Bermudez, F. S., Guitteny, V., Thome, N. & Cord, M., 2018. Manifold Learning in Quotient Spaces, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 9165–9174.

Mo, Y. & Magri, L., 2025. 'Reconstructing unsteady flows from sparse, noisy measurements with a physics-constrained convolutional neural network', *Physical Review Fluids* **10**(3), 034901.

Mo, Y. & Magri, L., 2026. 'Reconstruction of three-dimensional turbulent flows from sparse and noisy planar measurements: A weight-sharing neural network approach', *Data-Centric Engineering* 7, e5.

Mo, Y., Traverso, T. & Magri, L., 2024. 'Decoder decomposition for the analysis of the latent space of nonlinear autoencoders with wind-tunnel experimental data', *Data-Centric Engineering* 5, e38.

Mohan, A. T., Lubbers, N., Chertkov, M. & Livescu, D., 2023. 'Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence', *Physical Review Fluids* **8**(1), 014604.

Mons, V., Chassaing, J. C., Gomez, T. & Sagaut, P., 2016. 'Reconstruction of unsteady viscous flows using data assimilation schemes', *Journal of Computational Physics* **316**, 255–280.

Mons, V., Du, Y. & Zaki, T. A., 2021. 'Ensemble-variational assimilation of statistical data in large-eddy simulation', *Physical Review Fluids* **6**(10), 104607.

Morton, C. & Yarusevych, S., 2016. 'Reconstructing three-dimensional wake topology based on planar PIV measurements and pattern recognition analysis', *Experiments in Fluids* 57(10), 156.

Murata, T., Fukami, K. & Fukagata, K., 2020. 'Nonlinear mode decomposition with convolutional neural networks for fluid dynamics', *Journal of Fluid Mechanics* **882**, A13.

Nakamura, T., Fukami, K., Hasegawa, K., Nabae, Y. & Fukagata, K., 2021. 'Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow', *Physics of Fluids* **33**(2), 025116.

Nekkanti, A. & Schmidt, O. T., 2023. 'Gappy spectral proper orthogonal decomposition', *Journal of Computational*

Physics **478**, 111950.

Noack, B. R., Morzynski, M., Tadmor, G. & Tadmor, G., 2011. *Reduced-Order Modelling for Flow Control*, Vol. 528, Springer Wien, Vienna, Austria.

Nóvoa, A., Racca, A. & Magri, L., 2024. 'Inferring unknown unknowns: Regularized bias-aware ensemble Kalman filter', *Computer Methods in Applied Mechanics and Engineering* **418**, 116502.

Ogoke, F., Meidani, K., Hashemi, A. & Farimani, A. B., 2021. 'Graph convolutional networks applied to unstructured flow field data', *Machine Learning: Science and Technology* **2**(4), 045020.

Oxlade, A. R., Valente, P. C., Ganapathisubramani, B. & Morrison, J. F., 2012. 'Denoising of time-resolved PIV for accurate measurement of turbulence spectra and reduced error in derivatives', *Experiments in Fluids* **53**(5), 1561–1575.

Özalp, E. & Magri, L., 2025. 'Stability analysis of chaotic systems in latent spaces', *Nonlinear Dynamics* **113**(11), 13791–13806.

Ozan, D. E. & Magri, L., 2023. 'Hard-constrained neural networks for modeling nonlinear acoustics', *Physical Review Fluids* **8**(10), 103201.

Özbay, A. G. & Laizet, S., 2022. 'Deep learning fluid flow reconstruction around arbitrary two-dimensional objects from sparse sensors using conformal mappings', *AIP Advances* **12**(4), 045126.

Özbay, A. G. & Laizet, S., 2023. 'FR3D: Three-dimensional flow reconstruction and force estimation for unsteady flows around extruded bluff bodies via conformal mapping aided convolutional autoencoders', *International Journal of Heat and Fluid Flow* **103**, 109199.

Page, J., 2025. 'Super-resolution of turbulence with dynamics in the loss', *Journal of Fluid Mechanics* **1002**, R3.

Page, J., Brenner, M. P. & Kerswell, R. R., 2021. 'Revealing the state space of turbulence using machine learning', *Physical Review Fluids* **6**(3).

Panton, R. L., 2001. 'Overview of the self-sustaining mechanisms of wall turbulence', *Progress in Aerospace Sciences* **37**(4), 341–383.

Pérez, J. M., Le Clainche, S. & Vega, J. M., 2020. 'Reconstruction of three-dimensional flow fields from two-dimensional data', *Journal of Computational Physics* **407**, 109239.

Pfadler, S., Dinkelacker, F., Beyrau, F. & Leipertz, A., 2009. 'High resolution dual-plane stereo-PIV for validation of subgrid scale models in large-eddy simulations of turbulent premixed flames', *Combustion and Flame* **156**(8), 1552–1564.

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

Pope, S. B., 2000. *Turbulent Flows*, 1. publ., 12. print edn, Cambridge Univ. Press, Cambridge.

Pröbsting, S., Scarano, F., Bernardini, M. & Pirozzoli, S., 2013. 'On the estimation of wall pressure coherence using time-resolved tomographic PIV', *Experiments in Fluids* **54**(7), 1567.

Raben, S. G., Charonko, J. J. & Vlachos, P. P., 2012. 'Adaptive gappy proper orthogonal decomposition for particle image velocimetry data reconstruction', *Measurement Science and Technology* **23**(2), 025303.

Racca, A., 2023. Neural Networks for the Prediction of Chaos and Turbulence, PhD thesis, University of Cambridge, Cambridge, UK.

Racca, A., Doan, N. A. K. & Magri, L., 2023. 'Predicting turbulent dynamics with the convolutional autoencoder echo state network', *Journal of Fluid Mechanics* **975**, A2.

Raissi, M., Perdikaris, P. & Karniadakis, G., 2019. '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.

Raissi, M., Yazdani, A. & Karniadakis, G. E., 2020. 'Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations', *Science* **367**(6481), 1026–1030.

Raynaud, G., Houde, S. & Gosselin, F. P., 2022. 'ModalPINN: An extension of physics-informed Neural Networks with enforced truncated Fourier decomposition for periodic flow reconstruction using a limited number of imperfect sensors', *Journal of Computational Physics* **464**, 111271.

Rigas, G., 2021. 'Turbulent Experimental Database1'.

Rigas, G., Oxlade, A. R., Morgans, A. S. & Morrison, J. F., 2014. 'Low-dimensional dynamics of a turbulent axisymmetric wake', *Journal of Fluid Mechanics* **755**, R5.

Ronneberger, O., Fischer, P. & Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation, *in* N. Navab, J. Hornegger, W. M. Wells & A. F. Frangi, eds, 'Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015', Vol. 9351, Springer International Publishing, Cham, pp. 234–241.

Rowan D. Brackston, 2017. Feedback Control of Three-Dimensional Bluff Body Wakes for Efficient Drag Reduction, PhD thesis, Imperial College London, London.

Rowley, C. W. & Dawson, S. T. M., 2017. 'Model Reduction for Flow Analysis and Control', *Annual Review of Fluid Mechanics* **49**(Volume 49, 2017), 387–417.

Rui, E.-Z., Chen, Z.-W., Ni, Y.-Q., Yuan, L. & Zeng, G.-Z., 2023. 'Reconstruction of 3D flow field around a building model in wind tunnel: A novel physics-informed neural network framework adopting dynamic prioritization self-adaptive loss balance strategy', *Engineering Applications of Computational Fluid Mechanics* **17**(1), 2238849.

Santos, J. E., Fox, Z. R., Mohan, A., O'Malley, D., Viswanathan, H. & Lubbers, N., 2023. 'Development of the Senseiver for efficient field reconstruction from sparse observations', *Nature Machine Intelligence* **5**(11), 1317–1325.

Sardar, M., Skillen, A., Zimoń, M. J., Draycott, S. & Revell, A., 2024. 'Spectrally decomposed denoising diffusion probabilistic models for generative turbulence super-resolution', *Physics of Fluids* **36**(11), 115179.

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

Schmidt, O. T. & Colonius, T., 2020. 'Guide to spectral proper orthogonal decomposition', *AIAA Journal* **58**(3), 1023–1033.

Shebalin, J. V. & Woodruff, S. L., 1997. 'Kolmogorov flow in three dimensions', *Physics of Fluids* **9**(1), 164–170.

Shokar, I. J. S., Kerswell, R. R. & Haynes, P. H., 2024. 'Stochastic Latent Transformer: Efficient Modeling of Stochastically Forced Zonal Jets', *Journal of Advances in Modeling Earth Systems* **16**(6), e2023MS004177.

Sitte, M. P. & Doan, N. A. K., 2022. 'Velocity reconstruction in puffing pool fires with physics-informed neural networks', *Physics of Fluids* **34**(8), 087124.

Sliwinski, L. & Rigas, G., 2023. 'Mean flow reconstruction of unsteady flows using physics-informed neural networks', *Data-Centric Engineering* **4**, e4.

Solera-Rico, A., Sanmiguel Vila, C., Gómez-López, M., Wang, Y., Almashjary, A., Dawson, S. T. M. & Vinuesa, R., 2024. 'β-Variational autoencoders and transformers for reduced-order modelling of fluid flows', *Nature Communications* **15**(1), 1361.

Subramaniam, A., Wong, M. L., Borker, R. D., Nimmagadda, S. & Lele, S. K., 2020. 'Turbulence Enrichment using Physics-informed Generative Adversarial Networks'.

Sun, L. & Wang, J.-X., 2020. 'Physics-constrained bayesian neural network for fluid flow reconstruction with sparse

and noisy data', *Theoretical and Applied Mechanics Letters* **10**(3), 161–169.

Suzuki, T. & Hasegawa, Y., 2017. 'Estimation of turbulent channel flow at based on the wall measurement using a simple sequential approach', *Journal of Fluid Mechanics* **830**, 760–796.

Symon, S., Dovetta, N., McKeon, B. J., Sipp, D. & Schmid, P. J., 2017. 'Data assimilation of mean velocity from 2D PIV measurements of flow over an idealized airfoil', *Experiments in Fluids* **58**(5), 61.

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

Tavoularis, S. & Nedić, J., 2024. *Measurement in Fluid Mechanics*, 2 edn, Cambridge University Press.

The Millennium Prize Problems, n.d.. https://www.claymath.org/millennium-problems/.

Touzé, C., Vizzaccaro, A. & Thomas, O., 2021. 'Model order reduction methods for geometrically nonlinear structures: A review of nonlinear techniques', *Nonlinear Dynamics* **105**(2), 1141–1190.

Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L. & Kutz, J. N., 2014. 'On dynamic mode decomposition: Theory and applications', *Journal of Computational Dynamics* **1**(2), 391–421.

van Oudheusden, B. W., 2013. 'PIV-based pressure measurement', *Measurement Science and Technology* **24**(3), 032001.

Varun Shankar, Gavin D Portwood, Arvind T Mohan, Peetak Mitra, Christopher Rackauckas, Lucas A Wilson, David Schmidt & Venkat Viswanathan, 2020. 'Learning non-linear spatio-temporal dynamics with convolutional Neural ODEs'.

Vassilicos, J. C., ed., 2001. *Intermittency in Turbulent Flows*, Cambridge Univeristy Press, Cambridge, UK.

Venturi, D. & Karniadakis, G., 2004. 'Gappy data and reconstruction procedures for flow past a cylinder', *Journal of Fluid Mechanics* **519**, 315–336.

Vinuesa, R. & Sirmacek, B., 2021. 'Interpretable deep-learning models to help achieve the Sustainable Development Goals', *Nature Machine Intelligence* **3**(11), 926.

Wang, M. & Zaki, T. A., 2021. 'State estimation in turbulent channel flow from limited observations', *Journal of Fluid Mechanics* **917**, A9.

Wang, Q., Hu, R. & Blonigan, P., 2014. 'Least Squares Shadowing sensitivity analysis of chaotic limit cycle oscillations', *Journal of Computational Physics* **267**, 210–224.

Weiss, J., 2019-06-17/2019-06-21. A tutorial on the proper orthogonal decomposition, *in* '2019 AIAA Aviation Forum', American Institute of Aeronautics and Astronautics, Dallas, Texas, United States.

Wong, J. C., Ooi, C., Gupta, A. & Ong, Y.-S., 2022. 'Learning in Sinusoidal Spaces With Physics-Informed Neural Networks', *IEEE Transactions on Artificial Intelligence* **5**(3), 985–1000.

Xie, D., Xiong, J. & Pu, S., 2017. All You Need is Beyond a Good Init: Exploring Better Solution for Training Extremely Deep Convolutional Neural Networks with Orthonormality and Modulation, *in* '2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', arXiv, Honolulu, HI, USA„ pp. 5075–5084.

Xuan, A. & Shen, L., 2023. 'Reconstruction of three-dimensional turbulent flow structures using surface measurements for free-surface flows based on a convolutional neural network', *Journal of Fluid Mechanics* **959**, A34.

Yeung, P. K., Zhai, X. M. & Sreenivasan, K. R., 2015. 'Extreme events in computational turbulence', *Proceedings of the National Academy of Sciences* **112**(41), 12633–12638.

Yousif, M. Z., Yu, L., Hoyas, S., Vinuesa, R. & Lim, H., 2023. 'A deep-learning approach for reconstructing 3D turbulent flows from 2D observation data', *Scientific Reports* **13**(1), 2529.

Yu, L., Yousif, M. Z., Zhang, M., Hoyas, S., Vinuesa, R. & Lim, H.-C., 2022. 'Three-dimensional ESRGAN for super-

resolution reconstruction of turbulent flows with tricubic interpolation-based transfer learning', *Physics of Fluids* **34**(12), 125126.

Zaki, T. A. & Wang, M., 2021. 'From limited observations to the state of turbulence: Fundamental difficulties of flow reconstruction', *Physical Review Fluids* **6**(10), 100501.

Zhang, J. & Zhao, X., 2021. 'Three-dimensional spatiotemporal wind field reconstruction based on physics-informed deep learning', *Applied Energy* **300**, 117390.

Zhang, X.-L., Xiao, H., He, G.-W. & Wang, S.-Z., 2021. 'Assimilation of disparate data for enhanced reconstruction of turbulent mean flows', *Computers & Fluids* **224**, 104962.

Zhou, X.-H., McClure, J. E., Chen, C. & Xiao, H., 2022. 'Neural network–based pore flow field prediction in porous media using super resolution', *Physical Review Fluids* **7**(7), 074302.

Zhu, L., Jiang, X., Lefauve, A., Kerswell, R. R. & Linden, P., 2024. 'New insights into experimental stratified flows obtained through physics-informed neural networks', *Journal of Fluid Mechanics* **981**, R1.

Appendices

Appendix A

Machine learning model hyperparameters and training details

A.1 Autoencoders in Chapter 3

This section describes the architectures of the autoencoders (AEs) and mode-decomposing autoencoders (MD-AEs) used in Chapter 3. Adapted from Mo et al. (2024).

The convolutional autoencoders are trained with the laminar wake dataset. All convolutional AEs and MD-AEs are built with encoders described in Table A.1 and decoders described in Table A.2. The encoders are identical except for the latent dimension N_z. For all convolutional AEs, the decoders are identical except for N_z. For all convolutional MD-AEs, the input to the decoder has shape (1), specified in Table A.2. When training on the wind-tunnel dataset, AEs with feed-forward layers are used (tables A.3 and A.4). The large decoder-only network (Table A.6) is only used in Section 3.6.1. The hyperparameters used by all networks can be found in Table A.5.

Table A.1: The encoder used in all convolutional AEs and MD-AEs.

Layers	Output shape	Notes
input	(200,129,1)	
conv2d	(200,129,8)	3-by-3 filter
maxpooling	(100,65,8)	
conv2d	(100,65,16)	3-by-3 filter
maxpooling	(50,33,16)	
conv2d	(50,33,32)	3-by-3 filter
maxpooling	(25,17,32)	
flatten	(13600)	
dense	(N_z)	output shape is (1) if MD-AE

Table A.2: The decoder used in all convolutional AEs and MD-AEs.

Layers	Output shape	Notes
input	(N_z)	shape is (1) if MD-AE
dense	(13600)	
reshape	(25,17,32)	
upsampling	(50,33,32)	bi-linear interpolation
conv2d	(50,33,16)	3-by-3 filter
upsampling	(100,65,16)	bi-linear interpolation
conv2d	(100,65,8)	3-by-3 filter
upsampling	(200,129,8)	bi-linear interpolation
conv2d	(200,129,1)	3-by-3 filter

Table A.3: The encoder used in feedforward AEs.

Layers	Output shape	Notes
input	(64)	
dense	(128)	
batch_normalisation	(128)	
dropout	(128)	
dense	(256)	
batch_normalisation	(256)	
dropout	(256)	
dense	(256)	
batch_normalisation	(256)	
dropout	(256)	
dense	(128)	
batch_normalisation	(128)	
dropout	(128)	
dense	(64)	
batch_normalisation	(64)	
dropout	(64)	
dense	(N_z)	

Table A.4: The decoder used in feedforward AEs.

Layers	Output shape	Notes
input	(N_z)	
dense	(64)	
batch_normalisation	(64)	
dropout	(64)	
dense	(128)	
batch_normalisation	(128)	
dropout	(128)	
dense	(256)	
batch_normalisation	(256)	
dropout	(256)	
dense	(256)	
batch_normalisation	(256)	
dropout	(256)	
dense	(128)	
batch_normalisation	(128)	
dropout	(128)	
dense	(64)	

Table A.5: The hyperparameters used in all networks. CosineDecayRestarts is a built-in tensorflow learning rate schedule (Martín Abadi et al. 2015, Loshchilov & Hutter 2017).

Which network (table/figure number(s) of the network(s) using these hyperparameters)	MD-AEs and AEs for the unsteady laminar wake (Table A.1, A.2)	AEs used to search for suitable latent dimension (Figure 3.22)	AEs for the wind-tunnel turbulent wake (Table A.3, A.4 and A.6)
Learning rate	0.001	0.004	0.0022
Learning rate schedule	n/a	CosineDecayRestarts	CosineDecayRestarts
Activation function	tanh	tanh	tanh
Regularization	0.0	0.0	0.00003
Dropout rate	0.0%	0.0%	1.4%

Table A.6: Large decoder used in section 3.6.1

Layers	Output shape	Notes
input	(2)	The large decoder is only used with $N_z = 2$
dense	(64)	
batch_normalisation	(64)	
dropout	(64)	
dense	(256)	
batch_normalisation	(256)	
dropout	(256)	
dense	(256)	
batch_normalisation	(256)	
dropout	(256)	
dense	(256)	
batch_normalisation	(256)	
dropout	(256)	
dense	(256)	
batch_normalisation	(256)	
dropout	(256)	
dense	(64)	

A.2 Model architectures and training in Chapter 4

This section provides details on the models and training parameters used in Chapter 4, along with the training procedure and a grid sensitivity study. This section is adapted from Mo & Magri (2025).

A.2.1 Grid sensitivity study for the reconstruction of the Kolmogorov flow

We use a Kolmogorov flow dataset generated with a different initial random seed than the dataset used in the rest of Section 4.5 to determine the number of sensors to place in the domain. A different random seed is used to eliminate possible bias in sensor placement. In order words, we want to make sure the reconstruction results presented in the main text come from a previously unseen dataset. We followed the procedure below to determine the number of sensors:

1. Conduct a hyperparameter search for all loss functions $\mathscr{L}^c$, $\mathscr{L}^s$ and $\mathscr{L}^m$ using 80 input sensors ($\boldsymbol{x}_{in}$) and 200 general sensors ($\boldsymbol{x}_s$) randomly placed in the domain.
2. Randomly remove the input pressure sensors while holding the number of general sensors constant at 200, until the relative error starts to increase (Figure A.1 left panel). We find that the relative error of the reconstructed flow starts to increase when the number of input sensors is smaller than 80.
3. Randomly remove the general sensors while holding the number of input sensors constant at 80, until the relative error starts to increase (Figure A.1 right panel). We find that the relative error starts to increase when using fewer than 150 sensors.

For Section 4.5.1 and 4.5.2, we use 80 input sensors and 150 general sensors randomly placed in the domain to reconstruct the flow.

A.2.2 Model and training parameters

This section contains the training- and network-related parameters for all examples shown in Chapter 4. Figure A.2 shows the learning rate schedules employed in the reconstruction of the laminar wake and the Kolmogorov flow. Table A.7 to A.10 contains the network and training parameters for the laminar wake test cases. Table A.11 to A.13 contains the network and training parameters for the Kolmogorov flow test cases. The parameter 'bottleneck image dimension' refers to the smallest image dimension achieved through the resizing layers in the lower branch of the PC-DualConvNet.

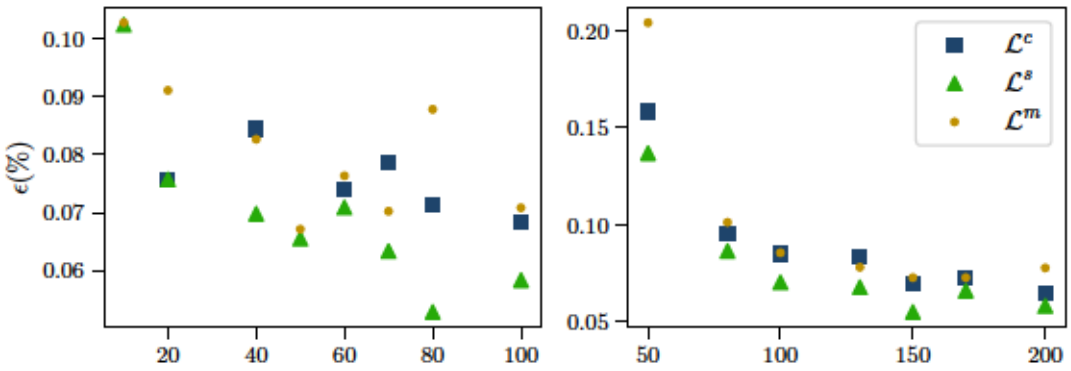


Figure A.1: Reconstruction error in response to changing the number of (left) pressure inputs $\boldsymbol{P}_{in}$ while holding the number of sensors constant at 200; and (right) sensors $\boldsymbol{D}(\boldsymbol{x}_s)$ while holding the number of pressure inputs constant at 80. This grid sensitivity test is performed on a Kolmogorov flow simulated with a different initial random seed.

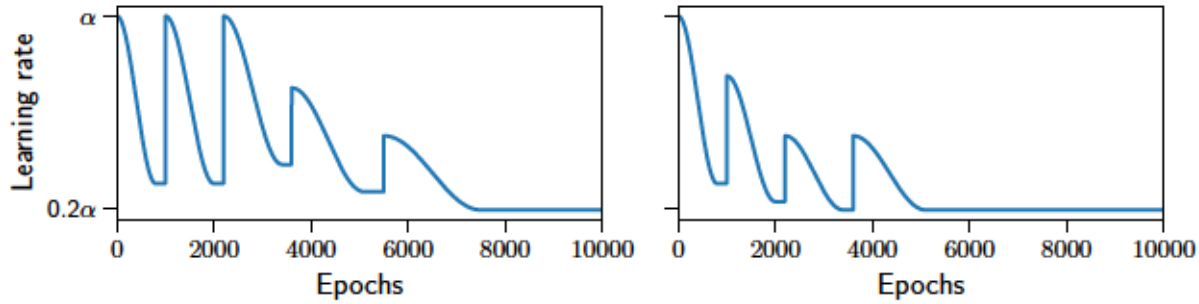


Figure A.2: The cyclic decay learning rate schedule applied to reconstruct the laminar flow (left) and the Kolmogorov flow (right), where α is the initial learning rate.

Table A.7: The network structure and hyperparameters used to reconstruct the laminar wake from non-noisy measurements.

Bottleneck image dimension	(64,16)
Convolution filter size (lower branch)	(5,5)
Convolution filter size (all other branches)	(3,3)
Input branch channels	[3,]
Upper branch channels	[8,]
Lower branch channels	[8,16,8]
Output branch channels	[4,3]
FFT	off
Batch size	[35]
λ_{div}	1.0
λ_{mom}	1.0
λ_o	40.0
Initial learning rate (α)	0.0023
regularization	0.0
Dropout rate	0.0

Table A.8: The structure and hyperparameters of the network using the softly-constrained loss, from noisy measurements. Showing the difference from A.7 only.

	SNR=20	SNR=10	SNR=5
Upper branch channels	[4,4]	[4,4]	[4,4]
Initial learning rate (α)	0.0013	0.0023	0.0012
Batch size	46	67	67
regularization	0.007	0.05	0.022
λ_{div}	2.0	2.1	3.8
λ_o	12.0	11.0	11.0
Dropout rate	0.05%	0.5%	0.5%

Table A.9: The structure and hyperparameters of the network using the snapshot-enforced loss, from noisy measurements. Showing the difference from A.7 only.

	SNR=20	SNR=10	SNR=5
Upper branch channels	[1,]	[1,]	[1,]
FFT	on	on	on
Initial learning rate (α)	0.014	0.0068	0.0088
Batch size	37	46	46
regularization	0.095	0.056	0.070
λ_{div}	1.0	1.0	1.0
λ_o	n/a	n/a	n/a
Dropout rate	0.48%	0.55%	0.6%

Table A.10: The structure and hyperparameters of the network using the mean-enforced loss, from noisy measurements. Showing the difference from A.7 only.

	SNR=20	SNR=10	SNR=5
Upper branch channels	[1,]	[4,4]	[8,]
Initial learning rate (α)	0.003	0.0023	0.0046
Batch size	50	200	120
regularization	0.076	0.09	0.042
λ_{div}	2.5	2.7	2.6
λ_o	31.5	48.5	24.0
Dropout rate	0.22%	0.25%	0.97%

Table A.11: The structure and hyperparameters of the network used to reconstruct the Kolmogorov flow with softly-constrained loss. The empty cells in column 2 through 4 represent parameters that are the same as in column 1. The very sparse case presented in Section 4.5.3 is trained with the hyperparameters in column 1.

	Non-noisy	SNR=20	SNR=10	SNR=5
Bottleneck image dimension	(16,16)			
Convolution filter size (upper branches)	(5,5)			
Convolution filter size (lower branches)	(5,5)		(3,3)	(3,3)
Convolution filter size (all other)	(3,3)			
Input branch channels	[3,]			
Upper branch channels	[4,4]			
Lower branch channels	[4,8,16,8,4]			
Output branch channels	[4,3]			
FFT	off			
Batch size	182		214	240
λ_{div}	1.0			
λ_{mom}	1.0			
λ_o	8.0		26.1	7.6
Initial learning rate (α)	0.004		0.00034	0.00018
regularization	0.0		0.0012	0.0037
Dropout rate	0.0%		0.24%	0.61%

Table A.12: The structure and hyperparameters of the network used to reconstruct the Kolmogorov flow with snapshot-enforced loss. The empty cells in column 2 through 4 represent parameters that are the same as in column 1. The very sparse case presented in Section 4.5.3 is trained with the hyperparameters in column 1.

	Non-noisy	SNR=20	SNR=10	SNR=5
Bottleneck image dimension	(16,16)			
Convolution filter size (upper branch)	(5,5)			
Convolution filter size (lower branch)	(5,5)			(3,3)
Convolution filter size (all other)	(3,3)			
Input branch channels	[3,]			
Upper branch channels	[1,]			
Lower branch channels	[4,8,16,8,4]			
Output branch channels	[4,3]			
FFT	on			
Batch size	158		214	214
λ_{div}	1.0			
λ_{mom}	1.0			
Initial learning rate (α)	0.0008		0.0028	0.0088
regularization	0.0		0.008	0.0056
Dropout rate	0.0%		0.59%	0.17%

Table A.13: The structure and hyperparameters of the network used to reconstruct the Kolmogorov flow with mean-enforced loss. The empty cells in column 2 through 4 represent parameters that are the same as in column 1. The very sparse case presented in Section 4.5.3 is trained with the hyperparameters in column 1.

	Non-noisy	SNR=20	SNR=10	SNR=5
Bottleneck image dimension	(16,16)			
Convolution filter size (upper branch)	(5,5)			
Convolution filter size (lower branch)	(5,5)		(3,3)	(3,3)
Convolution filter size (all other)	(3,3)			
Input branch channels	[3,]			
Upper branch channels	[4,]			
Lower branch channels	[4,8,16,8,4]			
Output branch channels	[4,3]			
FFT	off			
Batch size	600		857	1200
λ_{div}	1.0			
λ_{mom}	1.0			
λ_o	64.0		34.6	5.9
Initial learning rate (α)	0.0047		0.0013	0.0013
regularization	0.0		0.0007	0.0039
Dropout rate	0.0%		0.21%	0.23%

A.2.3 Learning curves

Figure A.3 and A.4 show the learning curves for the laminar wake and Kolmogorov examples shown in this paper.

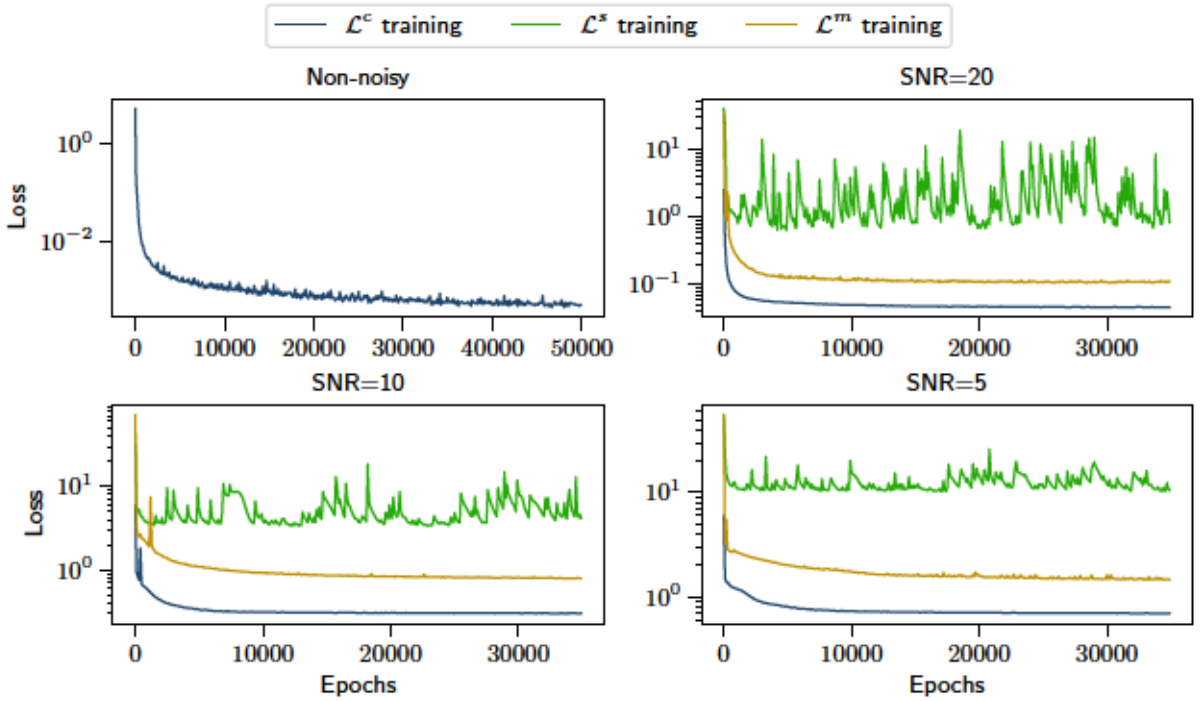


Figure A.3: Learning curves of the reconstruction of the laminar wake dataset. Learning curves are shown for the test runs plotted in Figure 4.7 and 4.10.

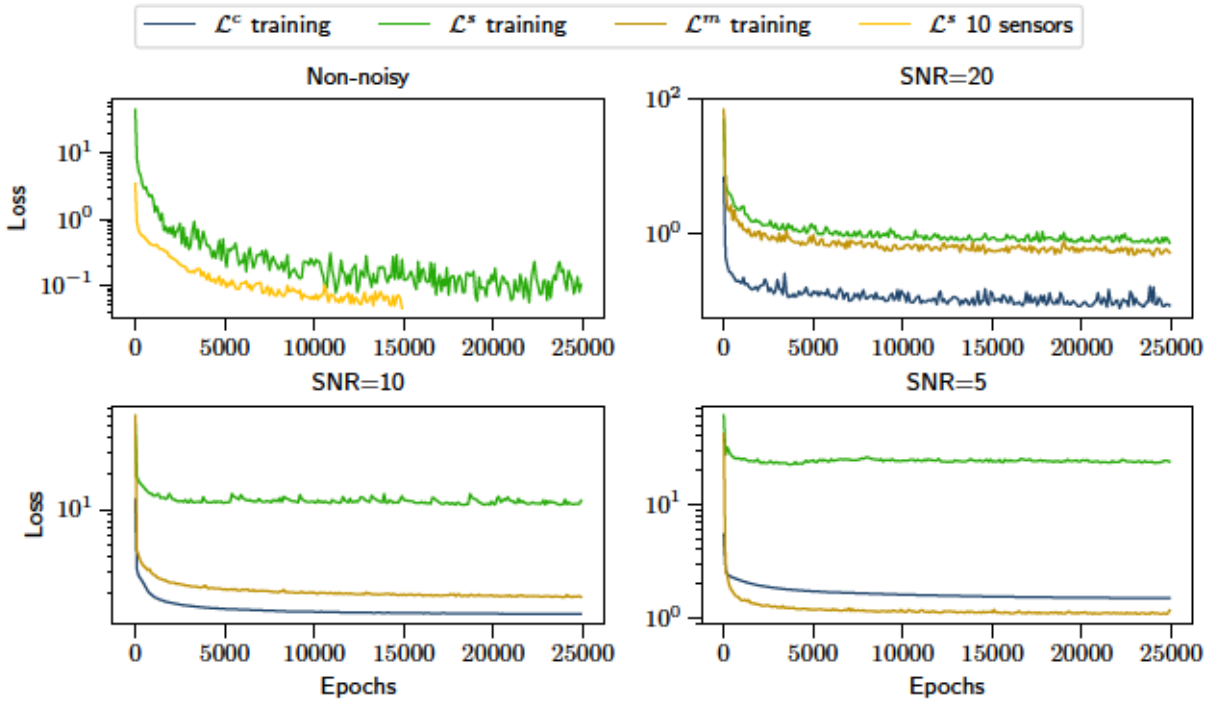


Figure A.4: Learning curves of the reconstruction of the turbulent Kolmogorov flow. Learning curves are shown for the test runs plotted in Figure 4.12, 4.13 and 4.16.

A.3 Model architectures in Chapter 5

This section contains all model and training hyperparameters used in Chapter 5. The learning rate schedule 'cyclic decay' is defined in A.2.2.

Table A.14: The network structure and hyperparameters used to reconstruct the 2D Kolmogorov flow with extreme events.

Bottleneck image dimension	(16,16)
Convolution filter size (lower branch)	(5,5)
Convolution filter size (all other branches)	(3,3)
Input branch channels	[3,]
Upper branch channels	[1,]
Lower branch channels	[4,8,16,8,4]
Output branch channels	[4,3]
FFT	on
Batch size	200
λ_{div}	1.0
λ_{mom}	1.0
Initial learning rate (α)	0.001
Learning rate schedule	Cyclic decay
regularization	0.0005
Dropout rate	0.001

Table A.15: The network structure and hyperparameters used to reconstruct the 3D Kolmogorov flow.

Bottleneck image dimension	(4,4,4)
Convolution filter size (all)	(3,3,3)
Convolution layer padding	Periodic
Input branch channels	[4,]
Upper branch channels	[4,]
Lower branch channels	[4,8,8,4]
Output branch channels	[4,]
FFT	off
Batch size	50
λ_{div}	1.0
λ_{mom}	11.0
Initial learning rate (α)	0.0065
Learning rate schedule	Cyclic decay
regularization	0.0
Dropout rate	0.0

A.4 Model architectures and training in Chapter 6

This section contains details on the models and the training process used in Chapter 6.

A.4.1 Hyperparameters and the minimum number of planes

In this section, we test the minimum number of x_1-x_2 measurement planes needed to reconstruct the 3D Kolmogorov flows using both the PC-DualConvNet and the weight-sharing network. We reduce the number of $x_1 - x_2$ measurement planes, while keeping the single $x_2 - x_3$ plane unchanged, until the relative errors of the reconstructed flows from both networks reach over 50%. We started by selecting a set of hyperparameters to reduce the unweighted sum of the physics and sensor loss ($\mathscr{L}_p + \mathscr{L}_o$) using test cases with eight $x_1 - x_2$ measurement planes, evenly spaced in the x_3 direction. Figure A.5 shows the training curves of the tests conducted during the hyperparameter selection process, each test uses a different set of hyperparameters. The tests with the selected hyperparameters are highlighted in red. Figure A.5 (left panel) shows the tests using PC-DualConvNet. Amongst the unselected tests (black), the difference in the hyperparameters resulted in different final loss values, but the training curves follow a similar downward trend. However, the test with the selected hyperparameters (red) clearly shows a faster decrease of the loss than the other tests (black). The difference in the training curve highlights the importance of hyperparameters in the training of the PC-DualConvNet.

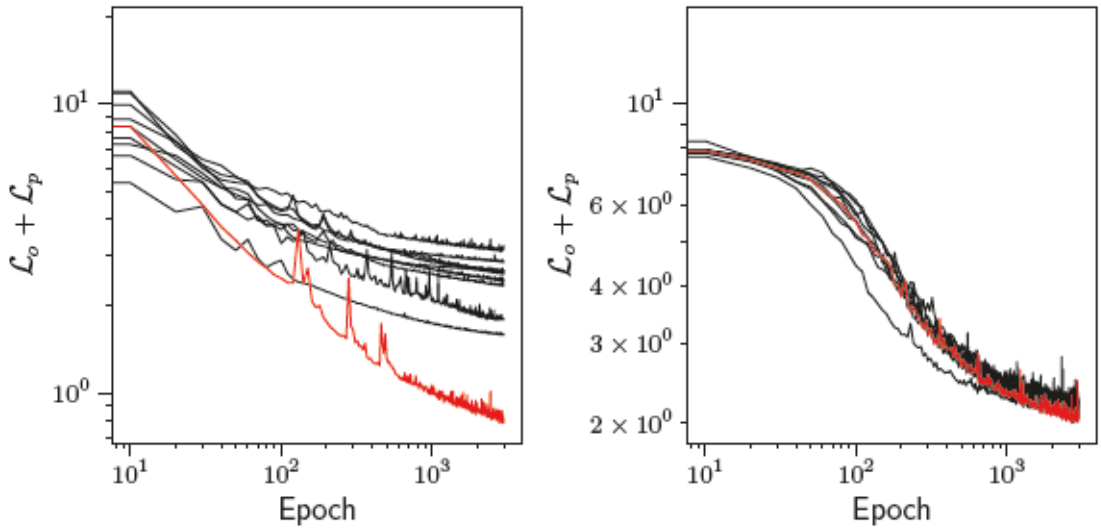


Figure A.5: The unweighted sum of the physics and sensor loss of the tests in the hyperparameter selection. Left: PC-DualConvNet. Right: weight-sharing network.

The selected hyperparameters are detailed in Table A.16 and A.17.

Using the selected set of hyperparameters, we reduce the number of measurement planes and plot the resulting mean squared error and the physics loss of the reconstructed flow fields in Figure A.6.

Using the selected set of hyperparameters, we reduce the number of measurement planes and plot the resulting mean squared error and the physics loss of the reconstructed flow fields in Figure A.6. Both the MSE and the physics loss shows that at the weight-sharing network achieves lower values for both metrics when there are two or less $x_1 - x_2$ measurement planes. The relative errors from both networks when there is only one $x_1 - x_2$ plane are much larger than 50% and the reconstructed flow fields lose resemblance to the reference data. Therefore, we report the results of reconstructing the 3D Kolmogorov flows from two $x_1 - x_2$ planes in the main text (Section 6.3).

Table A.16: The hyperparameters of the PC-DualConvNet used to reconstruct the 3D Kolmogorov flows in Section 6.3.

Bottleneck image dimension	(4,4,4)
Convolution filter size (all)	(3,3,3)
Convolution layer padding	Periodic
Input branch channels	[4,]
Upper branch channels	[4,]
Lower branch channels	[4,8,8,4]
Output branch channels	[4,]
FFT	off
Batch size	250
λ_{div}	1.0
λ_{mom}	3.0
Initial learning rate (α)	0.001
Learning rate schedule	Cyclic decay
regularization	0.0
Dropout rate	0.0

Table A.17: The hyperparameters of the weight-sharing network used to reconstruct the 3D Kolmogorov flows in Section 6.3.

Bottleneck image dimension (2D)	(8,8)
Convolution filter size (2D)	(3,3,3)
Upper branch channels (2D)	[4,]
Lower branch channels (2D)	[4,16,16,8]
FFT (2D)	off
Bottleneck image dimension (3D)	(32,32,32)
Channels (3D)	[8,8,4]
Convolution filter size (3D)	[(3,3,3),(5,5,5),(5,5,5)]
Convolution layer padding (all)	Periodic
Batch size	100
λ_{div}	1.0
λ_{mom}	2.0
Initial learning rate (α)	0.0025
Learning rate schedule	Exponential decay
regularization	0.0
Dropout rate	0.0

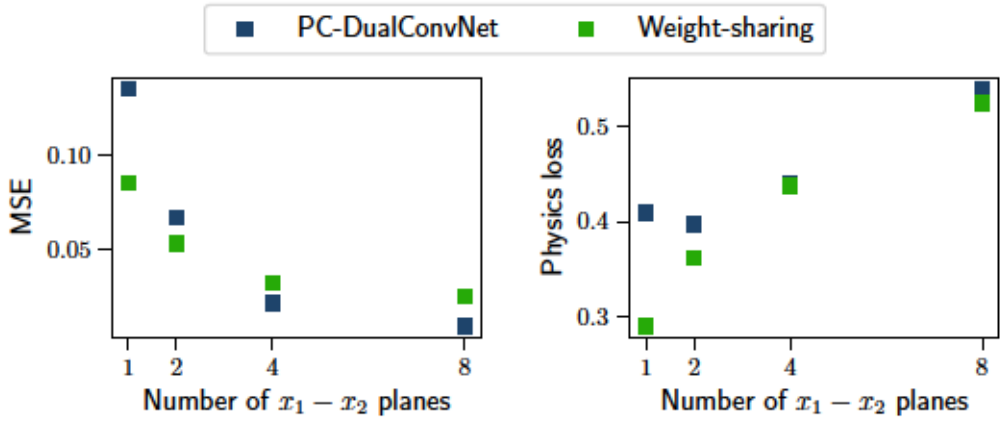


Figure A.6: (Left) the mean squared error and (right) the physics loss of the reconstructed from fields from different number of $x_1 - x_2$ measurement planes.

A.4.2 Model training time

This section discuss the training time of the networks used in Section 6.3. Table A.18 shows the number of parameters and the avergae update and inference time of the PC-DualConvNet and the weight-sharing network. The PC-DualConvNet has approximately 500 times more parameters than the weight-sharing network, and approximately double the inference time, meaning that the weight-sharing network uses less memory and is faster when used for inference. However, the average update time of the two networks are within 10% of each other, which means that both networks take a similar amount of time to train, despite the large difference in the number of parameters. Upon closer inspection, we found the bottleneck of the update time to be computing the physics loss. Table A.19 shows that the average time to compute the physics loss for a 50-snapshot batch is 84.4 ms, which is over 40% of the update times for both networks. The time to compute the sensor loss is negligible in comparison. When scaling up the datasets for future works, we must take into account the time to compute the physics loss and investigate ways to speed up the computation.

Table A.18: Number of trainable parameters; the average time to compute one update (including a forward pass, computing the loss, and applying the update); and the average inference time (forward pass only). Timed using a 50-snapshot batch on an NVIDIA RTX8000 GPU. Functions are compiled with `jax.jit`.

	Num. of params	**Update time** (ms)	**Inference time** (ms)
Weight-sharing	271,805	208	46.4
PC-DualConvNet	134,255,824	189	107

Table A.19: The average time to compute loss using a 50-snapshot batch on an NVIDIA RTX8000 GPU. Functions are compiled with `jax.jit`.

	Computation time (ms)
Physics loss	84.4
Sensor loss	2.2
MSE of full flow field	38.8

Table A.20: The hyperparameters of the PC-DualConvNet used to reconstruct the 3D Kolmogorov flows in Section 6.4.

Bottleneck image dimension	(4,4,4)
Convolution filter size (all)	(3,3,3)
Convolution layer padding	Periodic
Input branch channels	[4,]
Upper branch channels	[4,]
Lower branch channels	[4,8,8,4]
Output branch channels	[4,]
FFT	off
Batch size	50
λ_{div}	1.0
λ_{mom}	4.0
λ_o	32.0
Initial learning rate (α)	0.0043
Learning rate schedule	Cyclic decay
regularization	0.0
Dropout rate	0.0094

A.4.3 Model and training parameters for noisy data

Table A.20 and A.21 show the hyperparameters used to reconstruct the 3D Kolmogorov flows from noisy measurements in Section 6.4 using the PC-DualConvNet and the weight-sharing network, respectively.

Table A.21: The hyperparameters of the weight-sharing network used to reconstruct the 3D Kolmogorov flows in Section 6.4.

Bottleneck image dimension (2D)	(8,8)
Convolution filter size (2D)	(3,3,3)
Upper branch channels (2D)	[4,]
Lower branch channels (2D)	[4,16,16,8]
FFT (2D)	off
Bottleneck image dimension (3D)	(32,32,32)
Channels (3D)	[8,8,4]
Convolution filter size (3D)	[(3,3,3),(5,5,5),(5,5,5)]
Convolution layer padding (all)	Periodic
Batch size	50
λ_{div}	1.0
λ_{mom}	14.0
λ_o	45.0
Initial learning rate (α)	0.0026
Learning rate schedule	Cyclic decay
regularization	0.0028
Dropout rate	0.004

Appendix B

Miscellaneous

B.1 Codes and data availability

The relevant codes for Chapter 3 are available at https://github.com/MagriLab/MD-CNN-AE[1]. The relevant data for Chapter 3 are available at https://doi.org/10.5281/zenodo.15623774. Training records are available at https://wandb.ai/yaxinm/POD_and_AE.

The relevant codes for Chapter 4 to 6 are available at https://github.com/MagriLab/FlowReconstructionFromExperiment[1].
The current codes are not compatible with 2D reconstruction (Chapter 4 and part of Chapter 5), the last compatible versions have been archived. Please refer to the README in the repository for more details. Training records are available at https://wandb.ai/yaxinm/FlowReconstruction.

All data is available upon request.

B.2 Linear Autoencoders and POD

This section gives a more in-depth analysis of the relationship between linear autoencoders (AEs) and proper orthogonal decomposition (POD). Adapted from Mo et al. (2024).

Performing POD is equivalent to solving the quadratic optimisation problem (Fukami, Hasegawa, Nakamura, Morimoto & Fukagata 2021)

$$\boldsymbol{\Phi} = \arg\min_{\boldsymbol{\Phi}^*} ||\boldsymbol{Q} - \boldsymbol{\Phi}^* \boldsymbol{\Phi}^{*T} \boldsymbol{Q}||_2^2. \tag{B.1}$$

Let us consider a linear AE in which the number of latent variables is equal to the number of grid points in the input, i.e. $N_z = N$. A linear AE is an AE with linear activation functions. If $N_z < N$,

[1] We acknowledge the use of ChatGPT and Microsoft Copilot in coding and de-bugging.

the linear AE recovers only the N_z most energetic POD modes in an L_2-norm sense. For a linear AE without biases, we express the training as

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \|\boldsymbol{Q} - \hat{\boldsymbol{W}}\boldsymbol{W}\boldsymbol{Q}\|_2^2, \tag{B.2}$$

which has the solution

$$\hat{\boldsymbol{W}}\boldsymbol{W} = \boldsymbol{I}, \tag{B.3}$$

where $\boldsymbol{W}$ and $\hat{\boldsymbol{W}}$ are the weights of the encoder and the decoder of the linear AE. These matrices are not necessarily orthogonal. To obtain the POD solution, we seek orthogonal $\hat{\boldsymbol{W}}$ and $\boldsymbol{W}$. Baldi & Hornik (1989) showed that for a linear AE without biases, the only solution is the POD solution, whereas all the remaining critical points are saddle points. However, there are saddle points in the optimisation of (B.2). We apply L_2-regularisation to the weights for the linear AE to converge to the POD solution, which is equivalent to minimizing the Frobenius norm of weights, denoted $\|\boldsymbol{W}\|_F$. When the autoencoder is linear, applying L_2-regularisation of weights is known as applying a "soft constraint of orthogonality" (Bansal et al. 2018, Xie et al. 2017). Training the linear AE with L_2-regularisation is solving

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \|\boldsymbol{Q} - \hat{\boldsymbol{Q}}\|_2^2 + \gamma\left(\|\boldsymbol{W}\|_F^2 + \|\hat{\boldsymbol{W}}\|_F^2\right), \tag{B.4}$$

where $\gamma \geq 0$ is the regularisation factor. The only solution to the optimisation problem (B.2) which minimizes (B.4) is

$$\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I} \tag{B.5}$$

where $\boldsymbol{I}$ is the identity matrix. Therefore, $\hat{\boldsymbol{W}} = \boldsymbol{W}^T$, i.e., the decoder and encoder matrices are orthogonal. A similar analysis can be applied to a linear MD-AE to show that each decomposed field is a POD mode.

B.3 Effect of single-precision floating-point in the training of PC-DualConvNet

The training and inference of machine learning models commonly use single-precision floating point format (`float32`) for efficiency. Compared with double-precision (`float64`), single-precision format is more prone to round off errors due to fewer available digits. The lower precision may have a stonger detrimental effect when computing the physics loss, which involves computing derivatives and multiple mathematical operations. Table B.1 shows a comparison between the divergence and momentum losses of the 2D Kolmogorov flow dataset presented in Chapter 4 computed in single- and double-precision floating-point format. The differences between the two floating point format are six orders-of-magnitude smaller than the expected values for both the divergence and the momentum losses. This small difference show that the floating-point precision does not cause a significant issue in imposing physical constraints when working

Table B.1: Comparing the physics losses computed in single- and double-precision floating point format, for the 2D Kolmogorov flow dataset presented in Chapter 4.

	single-precision	**double-precision**	**Difference**
$\mathscr{L}_{div}$	1.2263612×10^{-5}	1.2263612×10^{-5}	$\mathcal{O}(10^{-11})$
$\mathscr{L}_{mom}$	2.6355257	2.6355242380402246	$\mathcal{O}(10^{-6})$

with the 2D Kolmogorov flow dataset. Nevertheless, more care needs to be taken when working with flows that are expected to have more energy in small-scale structures, as round-off errors are more likely to cause higher relative error when taking derivatives at higher wavenumbers.

B.4 2D Kolmogorov flow at different Reynolds numbers

This section presents a more detailed comparison of the Characteristics of our 2D Kolmogorov flow test cases at two different Reynolds number, which are used in Chapter 4 and 5. In the main text, we use the term 'weakly turbulent' to describe the 2D Kolmogorov flow at $Re = 34$ and the term 'turbulent' to describe the flow at $Re = 40$.

By 'weakly turbulent', we describe a flow that displays spatial-temporal chaotic behaviours, but has yet to develope other characteristics of turbulence, such as board energy spectrums or intermittency. The $Re = 34$ case in Chapter 4 is chaotic with a positive Lyapunov exponent (Özalp & Magri 2025), meaning that it is sensitive to initial conditions. However, energy is concentrated in a few Fourier modes (Figure B.1a), particularly at the forced wavenumber and its harmonics. In comparison, energy in the 'turbulent' flow at $Re = 40$, which we reconstructed in Chapter 5, is spread over a cloud of Fourier modes (Figure B.1b). The higher number of active Fourier modes indicates more interscale mixing, more effective degrees of freedom, requiring more sensors for reconstruction and therefore, the flow becomes less compressible. Intermittency, which is a characteristic of turbulence, is observed in the flow at $Re = 40$, but not at $Re = 34$ (Figure B.1c). Extreme events is localised in time, and the spatial patterns displayed by the flow during an extreme event are very different to its typical state. While extreme events are known to be difficult to predict, prediction is out of the scope of this thesis. For instantaneous reconstruction, the difficulties of these extreme events lies in requiring the network to learn statistically rare events. Although, neither of the flows are considered 'fully developed', as their energy spectrum do not display a distinct inertial range.

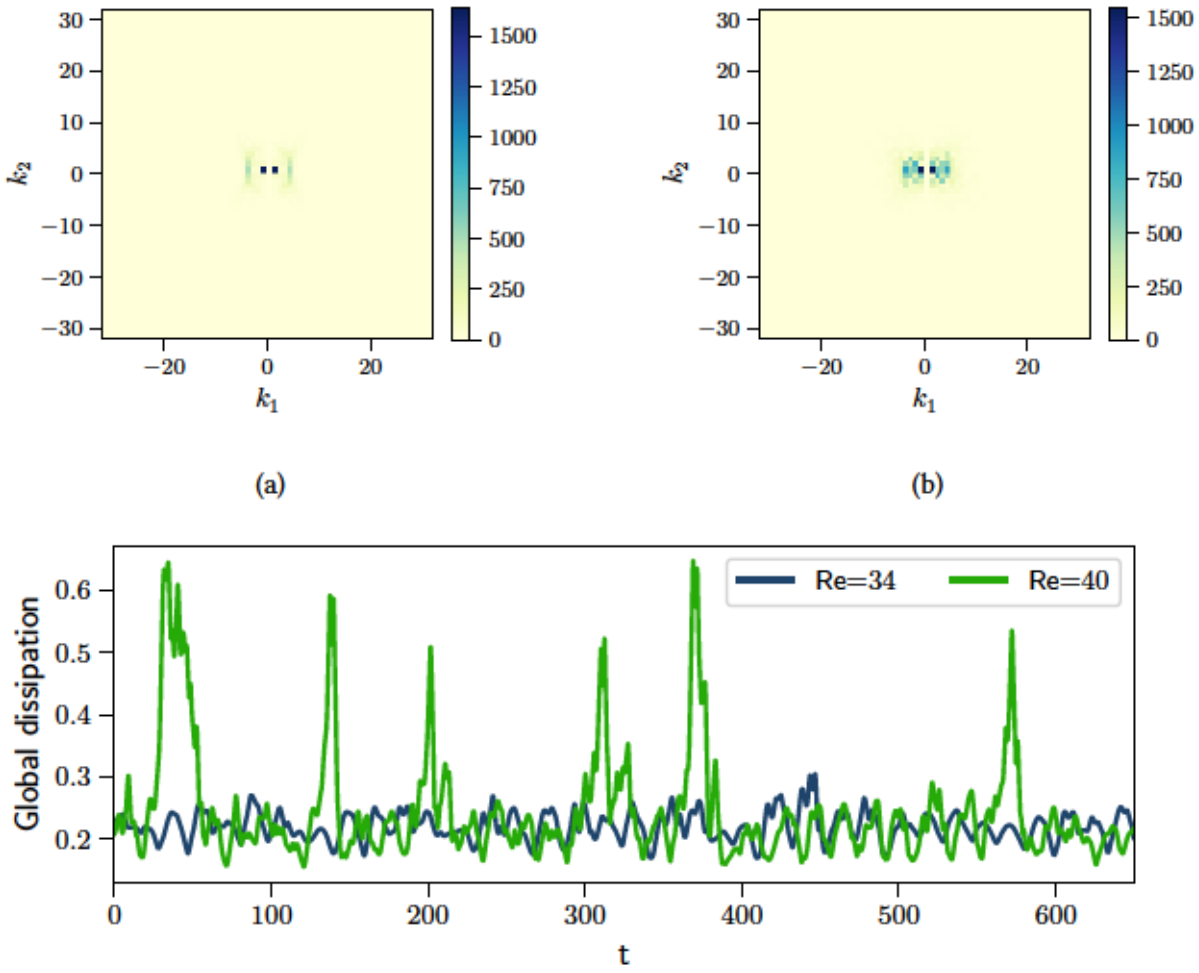


Figure B.1: (a) The energy contained in the Fourier modes of the $Re = 34$ flow. (b) The energy contained in the Fourier modes of the $Re = 40$ flow. (c) The global dissipation of the two Kolmogorov flow. Extreme dissipation events are observed at $Re = 40$ but not $Re = 34$.

B.5 Effect of the Fourier layer in PC-DualConvNet

In Chapter 4, we introduced the PC-DualConvNet with a Fourier transform layer in the upper branch for the reconstruction of flows. In the same chapter, we also found that the snapshot-enforced loss is effective in reconstructing flows from sparse measurements, but less so when the measurements are noisy. As part of the hyperparameter study in Chapter 5, Section 5.3, we discuss the impact of the FFT layer in the PC-DualConvNet has on the robustness of the snapshot-enforced loss to noisy measurements.

We trained multiple PC-DualConvNets, both with and without the FFT layer in the upper branch, using a range of hyperparameters without affecting the number of trainable weights. All networks trained in this section use the snapshot-enforced loss. All test cases in this section starts with the same randomly placed general (50 in total) and input pressure (50 in total) sensors, using the 2D Kolmogorov flow with extreme events in Chapter 5, Section 5.3. We summarize the relative error and estimated power of noise of the flows from noisy and very noisy measurements in Figure B.2. The power of the noise for a time series is estimated by finding the power which remains constant at high frequencies in the power spectrum (Oxlade et al. 2012). We estimate the power of the noise for each measurement time series, then take the average, which is shown in Figure B.2.

When reconstructing from noisy measurements, there is a large overlap between the range of relative error achieved by the networks with and without the FFT layer. The networks without the FFT layer generally achieve a 10% lower relative error. The opposite is true for the estimated power of noise — networks with FFT layer shows stronger de-noising effect than the networks without the FFT layer. The difference between the networks are larger when the measurements are noisier. The comparison of the metrics shown in Figure B.2 shows that the FFT branch provides built-in de-noising for the networks and the snapshot-enforced loss, leading to more physical reconstructions at the expense of a small increase in the relative error.

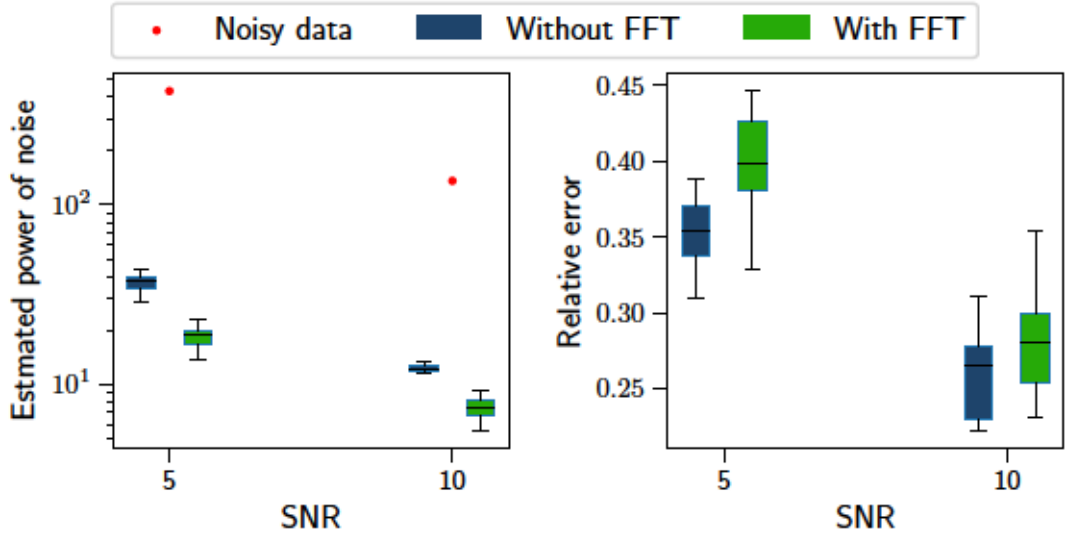


Figure B.2: Comparing the estimated power of noise (left panel) and the relative error (right panel) of the flow reconstructed with PC-DualConvNets with and without the FFT layer. The statistics are computed over 50 runs using different hyperparameters for each SNR.

B.6 Forcing divergence-free condition

Preivous works on flow reconstruction (Page 2025) and compression (Mohan et al. 2023) have enforced the divergence-free condition of structured data through vector operations and reported positive results. At the same time, strict constraints may not always lead to better results, as seen in Cato et al. (2023) and Chapter 4. In this section, we investigate the effect of enforcing the divergence-free condition on the weight-sharing network from Chapter 6. The vector identity $\nabla \cdot (\nabla \times *) = 0$, where $*$ is any vector, states that the divergence of the curl of a vector is zero. Therefore, we enforce the divergence-free condition by defining a custom last layer of the weight-sharing network to compute the curl of the output vector field. The output of the curl layer is then used as the final output.

Table B.2 compares the result of the divergence-free network with the results in Section 6.3. The divergence $\mathscr{L}_{div}$ of the divergence-free network is near zero, which is lower than the reference data. However, there is a 10% increase in the relative error of the reconstructed flow field compared to the network without the curl layer, and a 25% increase in the residual of the momentum equation $\mathscr{L}_{mom}$. Similar observations are made in Figure B.3, where there is no significant qualitative difference between the network with and without the curl layer. At $x_3 = 1.77$, the reconstructed planes from the divergence-free network is a slightly closer match to the reference data, but at $x_3 = 3.14$, the weight-sharing network without the divergence-free condition is a closer match. From these results, we conclude that there is no current benefit to enforcing the divergence-free condition in the weight-sharing network.

Table B.2: The relative error, residual of the momentum and continuity equations of the reconstruction results from Section 6.3, compared with the the same network with an additional divergence-free condition after the last layer.

	ϵ (%)	$\mathscr{L}_{mom}$	$\mathscr{L}_{div}$
Reference data	N/A	0.137	2.50×10^{-4}
Weight-sharing network	49.3	0.274	0.051
Divergence-free	54.2	0.342	8.52×10^{-12}

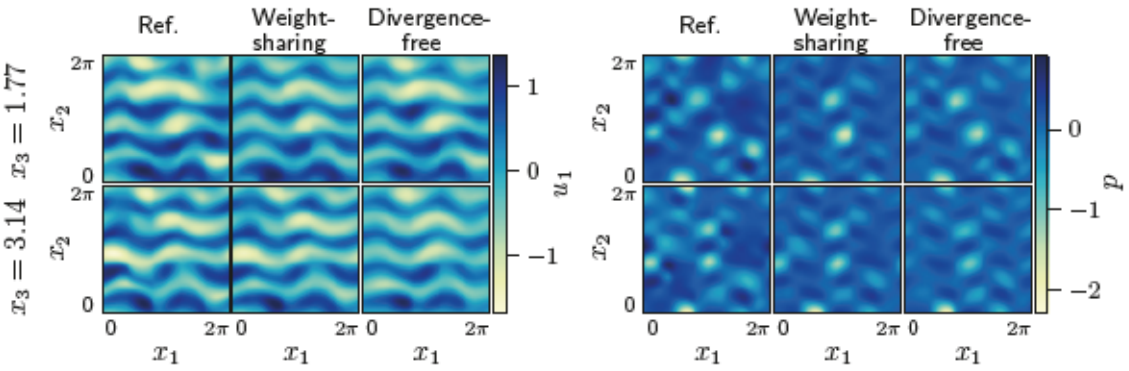


Figure B.3: Two slices of the reconstructed flow at x_3 =1.77 and 3.14, which are unseen by the network during training. The measurements used in training are taken on planes at x_3 =1.57 and 4.71.