Applied machine learning for prediction and control of fluid flows

Thesis by
Peter Ian James Renn

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2023
Defended January 11, 2023

Peter Ian James Renn
ORCID: 0000-0002-5735-3873

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Dr. Mory Gharib, who has provided me support and guidance throughout my time at Caltech. I deeply appreciate the opportunities which he has given me over the past several years as well as the many things that he has taught me.

I also acknowledge and thank the members of my committee for their insightful counsel: Dr. Anima Anandkumar, Dr. Jane Bae, and Dr. John Dabiri. I have particularly benefited through collaborations with Dr. Anandkumar and her students which have deeply influenced the latter half of the research presented in this thesis.

I would like to thank both past and present members of the Gharib group, especially my first mentors in the lab: Chris Dougherty and Marcel Veismann. My early experiences in the lab would not have been nearly as fruitful or fulfilling without their guidance. Special thanks also to Chris Roh and Cong Wang, who have both given me excellent counsel on my research and life pursuits throughout this whole process.

I also must thank CAST managers Noel and Reza for their assistance and advice throughout my time here. Additionally, thank you to Martha Salcedo, Jamie Meighen-Sei, and Sarah Pontes for helping solve countless logistical issues.

I also would like to thank my parents and siblings. This work would not have been possible without their unwavering support and encouragement. I deeply appreciate everything they have given me.

Finally, I would like to thank my partner, Emily. From our first year problem sets to writing this thesis, she has been a great source of joy in my life and I can't imagine having done any of it without her. Being able to share this time of my life with her, and our two dogs Margot and Charles, has been truly a gift.

Financial support for this work came from the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1745301, as well as the Caltech Center for Autonomous Systems and Technologies.

ABSTRACT

Modern aerodynamic technologies such as unmanned aerial systems and horizontal axis wind turbines must regularly contend with forces from highly stochastic and turbulent atmospheric gusts. Conventional methods for modeling and controlling fluid flows are limited in their ability to mitigate these aerodynamic forces in real-time. By applying modern machine learning techniques in an experimental setting, this thesis demonstrates the utility of machine learning in addressing these important problems. We follow two complementary approaches towards this goal.

First, we find an end-to-end solution for control in a gusty environment with model-free reinforcement learning. We deploy state-of-the-art reinforcement learning algorithms on a generalized aerodynamic test-bed consisting of an airfoil with motorized trailing edge flaps. The system features embedded flow sensors, enabling the inclusion of flow measurements in state observations. We place this system in a highly irregular wake behind a bluff-body, dynamically mounted on elastic bands and therefore free to oscillate, and train reinforcement learning agents to minimize the net lifting force on the system by controlling the position of the trailing edge flaps. We find that model-free reinforcement learning agents can outperform basic linear controllers in this gusty, turbulent environment. We also show that augmenting state observations with flow measurements can lead to more consistent learning of the system dynamics.

Next, we explore Fourier neural operators (FNOs) as a method for forecasting the time evolution of turbulent fluid flows. FNOs are capable of learning underlying operator solutions to families of partial differential equations and can be evaluated in just milliseconds. We specifically focus on training FNOs with experimentally measured velocity fields of bluff body wakes in the subcritical regime. To the best of our knowledge, this is the first application of operator learning for fluid mechanics that features experimental measurements. We find that FNOs can accurately predict the evolution of these turbulent wakes even when trained with imperfect measurements. We then show that FNOs can quickly adapt to unseen conditions with minimal data and training through transfer learning. Finally, we consider the performance of FNOs over longer prediction horizons. This approach could enable real-time gust prediction capabilities and monitoring for applied aerodynamic systems.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Peter I. Renn and Morteza Gharib. Machine learning for flow-informed aerodynamic control in turbulent wind conditions. *Communications Engineering*, 1 (45), 2022. doi: 10.1038/s44172-022-00046-z.
P.R. participated in the conception of the project, designed and built the experimental setup, adapted the algorithms for deployment, and wrote the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

Machine learning has attracted massive interest from scientists and engineers in many different fields over the past few decades. This sudden, intense focus on learning-based data-driven techniques has been fruitful; the accelerated development of this burgeoning field has lead to many important advances in both theory and deployment of machine learning methods.

There exists a fundamental overlap between fluid mechanics and applied mathematics that dates back centuries [52]. Theoretical and computational fluid mechanics were borne out of mathematicians trying to model fluid flows. Famous names such as Cauchy, Euler, and Laplace are still known as progenitors of both fields today. It is therefore not surprising that fluid research has not been isolated from developments in machine learning, especially as it relates to modern applied mathematics. Even work of more modern figures, such as Andrey Kolmogorov, can be found in introductory texts for fluid mechanics and machine learning alike.

Applications of machine learning in fluid mechanics have become quite diverse. There has been particular interest on subjects like simulated swimmers learning to navigate under varying environmental conditions [16, 19, 24, 54], flow control for bluff-body wakes [11, 28, 42, 45, 51], flow field reconstruction from sparse data [2, 6, 10, 14, 15, 32, 33, 37], and various reduced order/machine learning-based modeling of flow fields [21, 22, 30, 34, 38, 55, 57].

Despite the huge range of applications for machine learning methods in fluid mechanics, they have largely been limited to the computational domain. There are exceptions to this; for example, several papers have been published using neural networks for higher resolution analysis of particle image velocimetry (PIV) data [4, 18, 27, 35, 41]. Notably, Jin et al. [25] use recurrent neural networks (RNNs) to reconstruct time-resolved velocity data from a combination of time-averaged PIV data and time-resolved probe measurements, proposing a solution to frequency limitations on some PIV systems. Physics informed neural networks (PINNs) [43] have also been applied to enhance flow measurements in several works [5, 9, 20], learning 'solutions' that minimize error with respect to a governing equation for a specific

instance and set of boundary conditions. There have also been a few previous applications of reinforcement learning in experimental fluid mechanics, although these have either been limited in scope due to computational and practical environmental challenges [44] or have focused on settings with known solutions [11, 46].

There exist many modern challenges in applied fluid mechanics that call for powerful, real-time solutions such as those that can be provided by data-driven techniques. While many valuable tools that have been proven effective for fluid flows in computational simulation, transferring these methods to a physical environment is non-trivial. Real-world solutions must contend with limited and potentially imperfect measurements under conditions where the exact solution is not available. In this thesis we demonstrate modern machine learning methods for applied fluid mechanics research, specifically focusing on problems that present substantial challenges for conventional methods. While the machine learning techniques we apply in this work are general enough for many different settings, the next portion of this chapter will detail two specific applications by which this work was inspired: autonomous flight in windy urban environments, and wind turbine control for operation under gusty conditions. The chapter will then conclude with a summary of the work presented.

1.1 Machine learning applications for modern fluid mechanics

Urban flight challenges

There are many potential applications for unmanned aerial systems (UAS) in urban environments. These include autonomous flying taxis, last-mile package delivery, transportation of time-sensitive medical supplies, and infrastructure tracking and management among many others. However, maintaining stable flight in urban environments is not a trivial task. Rows of buildings form channels called "urban canyons" which lead to elevated wind speeds and turbulence [8, 12]. Even a single building (as shown in figure 1.1 can generate updrafts, downdrafts, horseshoe vortices, and trailing vortices [56].

Small, lightweight UAS such as those suited for package delivery and surveillance are especially susceptible to flow disturbances on this scale [26]. Because the conventional aircraft which have dominated aeronautics research for the past century are larger, heavier, and faster, existing work relating to UAS-scale gust interactions is limited [26].

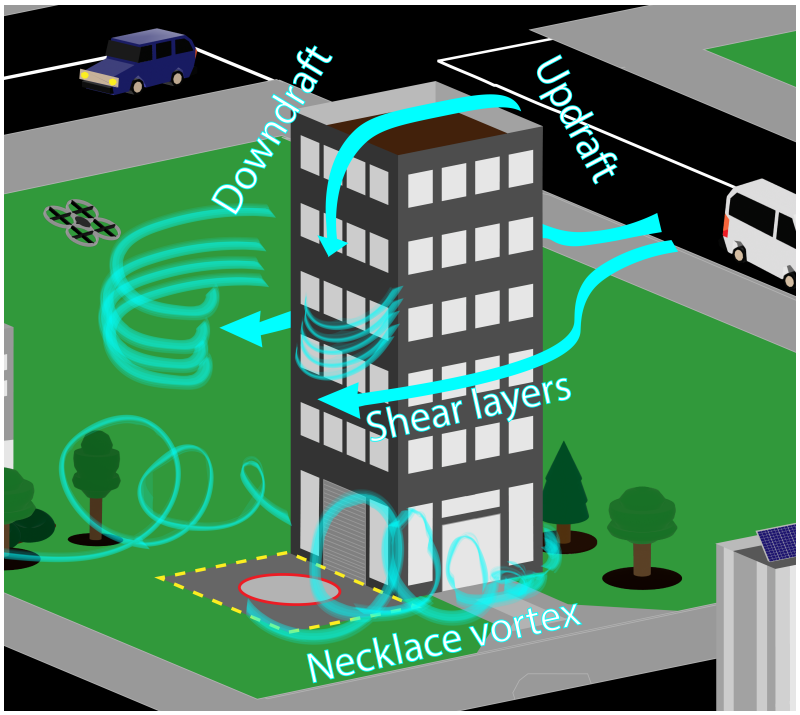

Figure 1.1: Illustration showing the potentially challenging fluid mechanics faced by unmanned aerial systems when flying near buildings.

The cost of failure in a dense urban environments is significant. A delivery UAS weighing tens of pounds could seriously damage buildings and infrastructure in a crash. Worse still, such systems would pose a serious threat to unsuspecting pedestrians when falling after a crash or being driven to ground level by a downward gust.

In order to ensure safe flight through complex urban flow fields, drones must learn to achieve stable control of aerodynamic forces in highly stochastic and non-linear physical environment. In addition to being able to mitigate unexpected extreme flow conditions as they arise, any effective strategy towards safe autonomous flight in cities should monitor and avoid regions where unsafe flow conditions are expected. Machine learning can help address both disturbance rejection and avoidance.

There have been several recent works focused on using machine learning to improve stability for UAS in free flight, however none have demonstrated stable flight in extreme, vortical, turbulent flows. For example, Mysore et al. [36] trained reinforcement learning agents for attitude control using a small racing-style drone and training partly in simulation. However, the resulting controller was not demonstrated in a challenging fluid environment. Also using a free-flight system, O'Connell et al.

[39] used a meta-learning framework for flight control of a multirotor UAS placed in wind, but only demonstrated the technique in a uniform flow with the free-stream velocity being adjusted at a low frequency. Even more recently, Simon et al. [48] trained a drone with a model-free reinforcement learning policy, using an on-board hot-wire anemometer system to observe the flow state. They were able to improve performance in a demonstration that involved hovering in $5ms^{-1}$ wind coming from an array of blowers that are turned on while the drone is already in hover. While each of these contributions is a valuable step towards safer flight in cities, none of them adequately recreate the challenge of a gusty urban environment. To our knowledge, there have been no existing demonstrations of UAS successfully navigating gusty and unsteady flows.

For applications as safety-critical as flight in urban environments, the best mitigation strategy is avoidance. This could mean simply grounding all UAS in adverse weather conditions, however applications such as package delivery require regular and reliable service. Another approach would be modeling the flow field in real-time based on local sensor data. Gianfelice et al. [17] recently demonstrated a "real-time" wind prediction system for drone flight in Toronto, applying commercial Reynolds-averaged Navier-Stokes (RANS) solvers to estimate the flow field. However, conventional techniques in computational fluid mechanics such as this are not yet capable of making temporally resolved predictions in real-time, and so the authors were only able to make time-averaged estimates that were validated by looking at point-wise hourly wind speeds and bearings. As discussed earlier machine learning methods can provide models of complicated fluid dynamics in far less time, and can reconstruct accurate time-resolved flow field representations from sparse measurements. Zhang et al. [58] recently used generative adversarial networks to predict instantaneous flow fields around a single building using surface pressure measurements in a computational environment. This is a promising approach, but the demonstration was limited to an idealized setting with perfect sensor data.

Wind turbine gust mitigation

Growing energy demands and the global environmental crisis have generated increased interest in alternative energy resources, such as horizontal-axis wind turbines. The pressure for more clean energy production has lead to a growth in these systems in both number and size. Transient atmospheric gusts are a major contributor to the fatigue loading of wind turbine blades and can damage internal turbine components. This can lead to increased maintenence requirements, costly

downtime, or even catastrophic failure of the system [47]. Some turbines mitigate the impact of these intermittent loads by simply shutting down in extreme flow conditions. However, this of course prevents the turbine from capturing any energy when winds become gusty. Another damage mitigation strategy in high winds is to use turbine blades with variable collective pitch, which can be adjusted to limit the turbine rotation speed and protect valuable internal components [3]. However shutting down or limiting rotation speed through blade pitch does not entirely prevent blade fatigue loading from gusts, which still necessitates costly inspections and replacement. Additionally, the composite materials from which most wind turbine blades are formed are unrecyclable by current waste-management standards, with predictions for turbine blade waste reaching over 40 million tonnes by 2050 [31].

Wind turbine gust mitigation is fundamentally similar to ensuring safe UAS flight in urban environments. Both systems can suffer damage from atmospheric gusts in regular operation, and both systems would ideally be capable of minimizing undesired aerodynamic forces. However, the problems have distinct considerations. As fixed systems, wind turbines can benefit from expensive and heavy remote sensing methods such as LiDAR to detect free-floating particles carried by incoming gusts [1, 53]. Similarly, traditional flow sensors can be fixed around wind farms to provide measurements of the incoming flow. However, UAS can change their flight plan or altitude to avoid particularly gusty conditions, while wind turbines are subject to whatever disturbances approach them.

There have been several efforts to mitigate wind loadings on turbine blades by actuating the pitch angle [7, 23, 40, 49, 50]. In many of these strategies the pitch of the blades can be controlled independently, which allows for individual blades to actuate in response to gusts to minimize the aerodynamic force. Hxperimental deployment of data-driven methods for turbine control are limited, even on scaled models. Notably, Frederik et al. [13] used a data-driven method known as subspace predictive repetitive control on a scale wind turbine model, and were able to significantly reduce loading in high turbulence. There is great potential in wind turbine operation and control for machine learning methods that can balance the simultaneous needs to maximize energy production, minimize blade loadings, and increase system reliability.

1.2 Summary of work

Learning-based methods could provide new capabilities in both controlling and predicting complicated fluid flows. These techniques could expand the flight capabilities of UAS and increase the performance and longevity of horizontal axis wind turbines. Progress has already been made towards both of these goals in previous works, however there remain important questions as to how these methods can perform when faced with highly stochastic physics and real-world measurements. In this work, we apply machine learning methods to challenging experimental conditions representative of real problems in applied fluid mechanics.

First, in Chapter 2 we provide a more in-depth overview of machine learning in fluid mechanics, specifically focusing on topics relevant to the work presented here. In Chapter 3 we present a previously published work describing an application of machine learning for end-to-end control of an experimental aerodynamic system in an extreme vortical flow field. This serves to demonstrate machine learning methods for control in a challenging and relevant fluid environment. Here we deploy model-free reinforcement learning methods on an aerodynamic test-bed custom built with integrated pressure sensors to observe the state of the surrounding flow. The test-bed acts as a generalized aerodynamic system that could be representative of a wind turbine blade or a fixed-wing UAS. This test-bed is placed on a load cell in the wake of a large, dynamically mounted bluff body ($Re \approx 230000$) which generates a highly irregular and gusty wake. We demonstrate that the use of recurrent neural network (RNN) significantly improves control performance in a highly stochastic flow setting, with the state-of-the-art algorithm that includes RNNs outperforming the conventional linear controls approach by a wide margin. We also find that including fluid flow measurements in addition to inertial measurements improves training stability and disturbance rejection, but that the system can learn effective control policies with either flow measurements or inertial measurements alone. We conclude the chapter by discussing the practical limitations of model-free reinforcement learning methods in a physical environment. Chapter 4 presents work exploring an application of Fourier neural operators (FNOs) for modeling and forecasting the temporal evolution of experimentally measured flow velocity fields. FNOs are a state-of-the-art data-driven technique that can directly approximate solution operators to families of PDEs and be evaluated in milliseconds [29]. This method holds great promise for being able to reconstruct and predict time-resolved flow fields in real-time. To our knowledge, this is the first application of operator-learning for experimental flow data. Here we apply FNOs to predict the time evolution of cylinder

wake velocity fields in the subcritical turbulent regime. We show that FNOs are capable of learning the wake dynamics and can produce accurate predictions over many time-steps at a range of Reynolds numbers. We also show that FNOs are able to generalize to different geometries, especially when transfer-learning techniques are applied. We conclude Chapter 4 by demonstrating FNOs over longer forecast horizons. We find that FNOs eventually lose fine details and structures present in the flow, but instead make predictions that appear similar to a phase-locked time-averaged flow field. Finally, in Chapter 5 we summarize the work presented and discuss the outlook and future research directions.

Bibliography

[1] E A Bossanyi, A Kumar, and O Hugues-Salas. Wind turbine control applications of turbine-mounted lidar. In *The Science of Making Torque from Wind 2012*, number 555, 2014.

[2] Ido Bright, Guang Lin, and J. Nathan Kutz. Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25, 2013.

[3] Tony L. Burton, Nick Jenkins, Ervin Bossanyi, David Sharpe, and Michael Graham. *Wind Energy Handbook, 3rd Edition*. Wiley, 2021.

[4] Shengze Cai, Jiaming Liang, Qi Gao, Chao Xu, and Runjie Wei. Particle image velocimetry based on a deep learning motion estimator. *IEEE Transactions on Instrumentation and Measurement*, 69(6):3538–3554, 2019.

[5] Shengze Cai, Zhicheng Wang, Frederik Fuest, Young Jin Jeon, Callum Gray, and George Em Karniadakis. Flow over an espresso cup: inferring 3-d velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks. *Journal of Fluid Mechanics*, 915, 2021.

[6] Jared L. Callaham, Kazuki Maeda, and Steven L. Brunton. Robust flow reconstruction from limited measurements via sparse representation. *Physical Review Fluids*, 4(10):103907, 2019.

[7] Z. J. Chen and K. A. Stol. An assessment of the effectiveness of individual pitch control on upscaled wind turbines. In *The Science of Making Torque from Wind 2014*, number 524, 2014.

[8] Taylor Cole, Xiangyi Li, Clark Eising, and Marko Princevac. Turbulence and channeling in a simple urban environment. In *17th Symposium on Boundary Layers and Turbulence*. American Meteorological Society, 2006.

[9] Patricio Clark Di Leoni, Karuna Agarwal, Tamer Zaki, Charles Meneveau, and Joseph Katz. Reconstructing velocity and pressure from sparse noisy particle tracks using physics-informed neural networks. *arXiv preprint arXiv:2210.04849*, 2022.

[10] N. Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L. Brunton, Michael W. Mahoney, and J. Nathan Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476(2238):20200097, 2020.

[11] Dixia Fan, Liu Yang, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. Reinforcement learning for bluff body active flow control in experiments and simulations. *Proceedings of the National Academy of Sciences*, 117(42):26091–26098, October 2020. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2004939117. URL http://www.pnas.org/lookup/doi/10.1073/pnas.2004939117.

[12] H.J.S. Fernando. Fluid dynamics of urban atmospheres in complex terrain. *Annual review of fluid mechanics*, 42:365–389, 2010.

[13] Joeri Frederik, Lars Kröger, Gerd Gülker, and Jan-Willem van Wingerden. Data-driven repetitive control: Wind tunnel experiments under turbulent conditions. *Control Engineering Practice*, 80:105–115, 2018.

[14] Kai Fukami, Romit Maulik, Nesar Ramachandra, Koji Fukagata, and Kunihiko Taira. Global field reconstruction from sparse sensors with voronoi tessellation-assisted deep learning. *Nature Machine Intelligence*, 3(11):945–951, 2021.

[15] Kai Fukami, Byungjin An, Motohiko Nohmi, Masashi Obuchi, and Kunihiko Taira. Machine-learning-based reconstruction of turbulent vortices from sparse pressure sensors in a pump sump. *Journal of Fluids Engineering*, 144(12): 121501, 2022.

[16] M. Gazzola, A. A. Tchieu, D. Alexeev, A. de Brauer, and P. Koumoutsakos. Learning to school in the presence of hydrodynamic interactions. *Journal of Fluid Mechanics*, 789.

[17] Michael Gianfelice, Haitham Aboshosha, and Tarek Ghazal. Real-time wind predictions for safe drone flights in toronto. *Results in Engineering*, 15:100534, 2022.

[18] I. Grant and X. Pan. An investigation of the performance of multi layer, neural networks applied to the analysis of piv images. *Experiments in Fluids*, 19(3): 159–166, 1995.

[19] Peter Gunnarson, Ioannis Mandralis, Guido Novati, Petros Koumoutsakos, and John O. Dabiri. Learning efficient navigation in vortical flow fields. *Nature Communications*, 12, 2021.

[20] Gazi Hasanuzzaman, Hamidreza Eivazi, Sebastian Merbold, Christoph Egbers, and Ricardo Vinuesa. Enhancement of piv measurements via physics-informed neural networks. *Measurement Science and Technology*, 2022.

[21] Kazuto Hasegawa, Kai Fukami, Takaaki Murata, and Koji Fukagata. Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. *Theoretical and Computational Fluid Dynamics*, 34, 2020.

[22] Kazuto Hasegawa, Kai Fukami, Takaaki Murata, and Koji Fukagata. Cnn-lstm based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different reynolds numbers. *Fluid Dynamics Research*, 52, 2020.

[23] Chengzhen Jia, Lingmei Wang, Enlong Meng, Liming Chen, Yushan Liu, Wenqiang Jia, Yutao Bao, and Zhenguo Liu. Combining lidar and ladrc for intelligent pitch control of wind turbines. *Renewable Energy*, 2021.

[24] Yusheng Jiao, Feng Ling, Sina Heydari, Nicolas Heess, Josh Merel, and Eva Kanso. Learning to swim in potential flow. *Physical Review Fluids*, 6(5): 050505, 2021.

[25] Xiaowei Jin, Shujin Laima, Wen-Li Chen, and Hui Li. Time-resolved reconstruction of flow field around a circular cylinder by recurrent neural networks based on non-time-resolved particle image velocimetry measurements. *Experiments in Fluids*, 61(4):1–23, 2020.

[26] Anya R. Jones, Oksan Cetiner, and Marilyn J. Smith. Physics and modeling of large flow disturbances: Discrete gust encounters for modern air vehicles. *Annual Review of Fluid Mechanics*, 54(1):469–493, 2022. doi: 10.1146/annurev-fluid-031621-085520. URL https://doi.org/10.1146/annurev-fluid-031621-085520.

[27] Christian Lagemann, Kai Lagemann, Sach Mukherjee, and Wolfgang Schröder. Deep recurrent optical flow learning for particle image velocimetry data. *Nature Machine Intelligence*, 3(7):641–651, 2021.

[28] Jichao Li and Mengqi Zhang. Reinforcement-learning-based control of confined cylinder wakes with stability analyses. *Journal of Fluid Mechanics*, 932, 2022.

[29] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[30] Björn List, Li-Wei Chen, and Nils Thuerey. Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons. *Journal of Fluid Mechanics*, 949:A25, 2022.

[31] Pu Liu and Claire Y. Barlow. Wind turbine blade waste in 2050. *Waste Management*, 62, 2017.

[32] Jean-Christophe Loiseau, Bernd R. Noack, and Steven L. Brunton. Sparse reduced-order modelling: sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844:459–490, 2018.

[33] Kevin H. Manohar, Chris Morton, and Paul Ziadé. Sparse sensor-based cylinder flow estimation using artificial neural networks. *Physical Review Fluids*, 7(2):024707, 2022.

[34] Michele Milano and Petros Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182:1–26, 2002.

[35] Masaki Morimoto, Kai Fukami, and Koji Fukagata. Experimental velocity data estimation for imperfect particle images using machine learning. *Physics of Fluids*, 33(8):087121, 2021.

[36] Siddharth Mysore, Bassel Mabsout, Kate Saenko, and Renato Mancuso. How to train your quadrotor: A framework for consistently smooth and responsive flight control via reinforcement learning. *ACM Transactions on Cyber-Physical Systems (TCPS)*, 5(4):1–24, 2021.

[37] Nirmal J. Nair and Andres Goza. Leveraging reduced-order models for state estimation using deep learning. *Journal of Fluid Mechanics*, 897, 2020.

[38] Taichi Nakamura, Kai Fukami, and Kazuto Hasegawa. Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Physics of Fluids*, 33, 2021.

[39] Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, 2022.

[40] C. Plumley, W. Leithead, P. Jamieson, E. Bossanyi, and M. Graham. Comparison of individual pitch and smart rotor control strategies for load reduction. In *The Science of Making Torque from Wind 2014*, number 524, 2014.

[41] Jean Rabault, Jostein Kolaas, and Atle Jensen. Performing particle image velocimetry using artificial neural networks: a proof-of-concept. *Measurement Science and Technology*, 28(12):125301, 2017.

[42] Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Reglade, and Nicolas Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865, 2019.

[43] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[44] Gautam Reddy, Jerome Wong-Ng, Antonio Celani, Terrence J. Sejnowski, and Massimo Vergassola. Glider soaring via reinforcement learning in the field. *Nature*, 562(7726):236–239, October 2018. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-018-0533-0. URL http://www.nature.com/articles/s41586-018-0533-0.

[45] Feng Ren, Jean Rabault, and Hui Tang. Applying deep reinforcement learning to active flow control in weakly turbulent conditions. *Physics of Fluids*, 33(3): 037121, 2021.

[46] Satoshi Shimomura, Satoshi Sekimoto, Akira Oyama, Kozo Fujii, and Hiroyuki Nishida. Closed-loop flow separation control using the deep q network over airfoil. *AIAA Journal*, 58, 2020.

[47] Md. Abu S. Shohag, Emily C. Hammel, David O. Olawale, and Okenwa I. Okoli. Damage mitigation techniques in wind turbine blades: A review. *Wind Engineering*, 41, 2017.

[48] Nathaniel Simon, Allen Z. Ren, Alexander Piqué, David Snyder, Daphne Barretto, Marcus Hultmark, and Anirudha Majumdar. Flowdrone: wind estimation and gust rejection on uavs using fast-response hot-wire flow sensors. *arXiv preprint arXiv:2210.05857*, 2022.

[49] Michael Sinner, Vlaho Petrovic, Apostolos Langidis, Lars Neuhaus, Michael Hölling, Martin Kühn, and Lucy Y. Pao. Experimental testing of a preview-enabled model predictive controller for blade pitch control of wind turbines. *IEEE Transactions on Control Systems Technology*, 30, 2021.

[50] Michael N. Sinner and Lucy Y. Pao. A comparison of individual and collective pitch model predictive controllers for wind turbines. In *American Control Conference*, 2018.

[51] Hongwei Tang, Jean Rabault, Alexander Kuhnle, Yan Wang, and Tongguang Wang. Robust active flow control over a range of reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Physics of Fluids*, 32(5):053605, 2020.

[52] Gregory A Tokaty. *A history and philosophy of fluid mechanics*. Courier Corporation, 1994.

[53] Paul D. Towers and Bryn Ll. Jones. Wind turbine gust estimation using remote sensing data. In *UKACC International Conference on Control*, 2014.

[54] Siddhartha Verma, Guido Novati, and Petros Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 2018.

[55] Pantelis R. Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.

[56] S. Watkins, J. Burry, A. Mohamed, M. Marino, S. Prudden, A. Fisher, N. Kloet, T. Jakobi, and R. Clothier. Ten questions concerning the use of drones in urban environments. *Building and Environment*, 167, 2020.

[57] Li Xu, Guanhao Zhou, Fengfeng Zhao, Zhaoliang Guo, and Kaijun Zhang. A data-driven reduced order modeling for fluid flow analysis based on series forecasting intelligent algorithm. *IEEE Access*, 10:60163–60176, 2022.

[58] Bingchao Zhang, Ryozo Ooka, Hideki Kikumoto, Chaoyi Hu, and Tim K.T. Tse. Towards real-time prediction of velocity field around a building using generative adversarial networks based on the surface pressure from sparse sensor networks. *Journal of Wind Engineering and Industrial Aerodynamics*, 231:105243, 2022.

Chapter 2

BACKGROUND: MACHINE LEARNING IN FLUID MECHANICS

There are many ways in which learning-based methods can enhance fluid mechanics research. In this thesis, we explore machine learning methods for two tasks that are related but distinct: controlling aerodynamic forces in a stochastic and turbulent flow field, and predicting the time evolution of unsteady velocity field measurements. Despite how different these applications are in structure, they share an underlying goal common to most, if not all, machine learning techniques: to learn to achieve a set task by identifying patterns and underlying relationships in sets of data.

The rest of this chapter provides an overview of the methods applied in this thesis and reviews related previous works in fluid mechanics. We note that Brunton et al. [2] provides a comprehensive review of other machine learning applications for fluid mechanics.

2.1 Reinforcement learning

First we introduce model-free reinforcement learning, which is the primary topic of Chapter 3. Comprehensive details on the fundamental principles of general reinforcement learning can be found in Sutton and Barto [24]. After introducing the structure and function of model-free reinforcement learning, we will review previous work featuring these methods in fluid mechanics.

Background on model-free reinforcement learning

Reinforcement learning is a subclass of machine learning that encompasses methods that learn to perform tasks through trial-and-error interactions. These methods are commonly formulated to learn end-to-end control of a system (from sensing to actuation) but can be applied in other contexts as well [1]. Most reinforcement learning methods used for control are designed to address problems formulated as a Markov decision process (MDP). MDPs can be used to model sequential decision-making processes, such as those commonly encountered in control. There are two primary components in an MDP: an agent and an environment. The agent is the decision-maker that chooses actions. In the context of reinforcement learning, the agent is also trying to learn which action choices are best to achieve the desired task.

The environment consists of everything that the agent cannot arbitrarily control. At discrete time-steps, the agent receives observations describing the state of the system, and then chooses an action based on these observations. Observations can consist of any available information that characterizes the state of the environment, and actions can take any form of manipulation or actuation available to the agent. These actions can (but do not always) change the state of the system. The agent then receives an updated observation, and the process repeats itself. In the context of a classical MDP, the observations fully define the underlying state. This assumption often fails in real-world settings due to sensor noise or incomplete state information. In this case, the decision-making process is called a partially observable Markov decision process (POMDP). We provide additional context for MDPs when introducing the problem addressed Chapter 3.

There are two broad categories of reinforcement learning algorithms: model-free and model-based. Model-based algorithms contain explicit models of the environment, and choose actions based on planned trajectories based on modeled interactions. These methods are typically more data-efficient as they generally are initialized with a relatively accurate environmental model, which they can use to refine their action selection without relying entirely on trial-and-error interactions. On the other hand, model-free reinforcement learning methods require no a priori model, and are often initialized as a random set of weights. As they have no knowledge of the environment or their own actions at the beginning of training, the learning process for these algorithms is entirely reliant on experience gained through trial-and-error interactions. In this thesis, we focus only on model-free reinforcement learning algorithms, with only brief reference to model-based reinforcement learning methods.

Model-free reinforcement learning agents learn behaviors to maximize a numerical reward signal. The reward signal prescribes the task desired of the agent. Choosing the reward signal can in itself be challenging, as it must represent only the desired outcome and not any preconceived notions of what behavior will result in this outcome. Sutton and Barto [24][p. 54] provides a clear example of this: in chess, capturing an opponent's pieces may seem an important part of any winning strategy, but a chess-playing agent rewarded for capturing pieces will only learn to maximize the number of pieces captures remaining indifferent towards the greater outcome of

the game. That is, it may learn a strategy that captures many pieces without ever winning. In this sense, it is important that the reward signal represents the desired goal (e.g. winning the game) without arbitrarily imposing ultimately misleading and unnecessary strategy.

There are many different formulations for model-free reinforcement learning agents. Much of the foundational work in this field has focused on action-value methods. Action-value methods learn to value each available action based on the performance of previous action choices given the current state. That is, given a state observation they approximate the expected value of each available action based on previous results. These methods can use bootstrapping to estimate not only the expected value of each action at the following time-step, but the expected long-term cumulative return based on taking that action. This allows the agents to choose actions with the best long-term value, rather than just maximizing the reward at the next step. This far-sighted decision making is key to successful reinforcement learning applications.

In action-value methods the learned control policy typically consists of estimating the value of all available actions given the current state and then applying some choosing algorithm to select an action based on these estimated values. Perhaps the most common example of this is the ϵ-greedy algorithm, in which the agent chooses actions stochastically. The action with the highest estimated value is chosen with probability $1 - \epsilon$. In the case where the action with the highest estimated value is not chosen (occurring with probability ϵ), a different action is chosen randomly with no consideration for estimated value. For this method, ϵ is typically small so that the agent most often chooses the action with the highest value estimate, but still occasionally explores other actions. The ϵ-greedy algorithm addresses the problem of "explore vs. exploit" which is especially important in model-free reinforcement learning algorithms. An agent that explores new actions too much will essentially act randomly and be incapable of following any successful learned behaviors. However, if an agent always chooses to exploit the learned value of actions, that is to always choose the action with the highest value, then the agent will tend to converge to a fixed, non-optimal policy based on only the earliest experiences. This almost always results in the agent overlooking action choices which were initialized as having lower values, which means high-performing actions may never be visited at

all. Therefore, balancing exploration and exploitation is important for model-free reinforcement learning methods, as it ensures that the agent will continue to explore the action space and learn better policies while still choosing sensible actions most of the time.

The model-free reinforcement learning methods used in Chapter 3 are known as actor-critic methods, which fall under the category of policy gradient algorithms rather than action-value methods. Rather than choosing the best action from the expected value of each action based on the state as in action-value algorithms, policy gradient methods directly learn a continuous parameterized policy that tries to maximize the return without explicitly calculating values. For example, in a game of tic-tac-toe an action-value method would observe the state, calculate the expected return based on each available move, and then choose one through a method such as the ϵ-greedy algorithm discussed above. In the same game, a policy gradient method would learn a policy that chooses actions based on the state state observation directly without referring to the estimated value. Instead, policy gradient methods learn to choose actions with high returns implicitly through training the policy directly: continuous, differentiable, parameterized policies can learn through gradient descent via the policy gradient theorem. This theorem provides that a value proportional to the gradient of policy performance can be calculated without knowing the state-value function, which allows for policy parameter updates to be made by typical gradient descent methods. More details on the policy gradient theorem can be found in Sutton and Barto [24][pp. 325-326].

As the name suggests, actor-critic methods have two components: an actor and a critic. The actor holds the parameterized control policy. It takes observations as an input and chooses actions as an output. The critic estimates the value of states and/or actions similar to action-value methods. However, the critic is not consulted when the actor is choosing actions making this a policy gradient method (rather than an action-value method). Still, the critic receives the state observations, actions chosen, and the resulting reward, and tries to approximate an action-value function. The critic then serves the actor during training, so that the actor can update its own parameters to to choose actions with estimated higher value. In this manner, the the critic is used to essentially bootstrap the learning process which makes actor-critic methods comparatively data-efficient and robust. In most modern applications, both the actor and critic take the form of neural networks which can be evaluated fast enough for real-time tasks.

Reinforcement learning in fluid mechanics

As an end-to-end black-box solution for control that requires no explicit modeling, model-free reinforcement learning has especially captivated the fluid mechanics community [2, 5, 28]. Previous applications of reinforcement learning have mostly focused on flow control in numerical simulations. For example, one focus of model-free reinforcement learning in fluid mechanics has been controlling the drag on a bluff body in a computational environment. In these works, an agent is typically set to suppress vortex shedding or reduce the drag force on a cylinder by rotating or actuating synthetic jets [12, 19, 20, 22, 25, 26, 29]. Simulated swimmers learning to navigate various flow conditions have also been a popular target for model-free reinforcement learning. Many of these studies have focused on schooling formations and sensing, among other topics [6, 7, 9, 27, 30].

Despite the apparent interest, there have been very few experimental applications of reinforcement learning in fluid mechanics. Arguably the first experimental application of reinforcement learning considering fluid mechanics came from Reddy et al. [21], wherein a glider was trained to soar on thermal plumes. While Reddy et al. [21] did find limited success in this application, the system featured a relatively simple algorithm, a coarsely discretized set of states, and faced practical challenges in training. Another seminal work, Fan et al. [4] applied reinforcement learning in a more explicit flow-control environment, using a state-of-the-art model-free algorithm to solve a well-studied bluff-body wake supression problem and performing complimentary physical and computational experiments. Shimomura et al. [23] used the Deep Q Network (DQN) algorithm to train agents for flow separation control on an airfoil with a plasma actuator. While important works, both of these applications used reinforcement learning for well-studied tasks with known solutions in conventional controls.

2.2 Neural operators

Multilayer feed-forward neural networks are universal approximators for functions that map between finite dimensional spaces Hornik et al. [8]. This means that any function between finite dimensional spaces can be approximated by a multi-layer feed-forward neural network given sufficient depth. Neural operators are a recently developed method in machine learning that share much of the same structure as standard neural networks [13, 14, 18]. However, neural operators are distinct in their ability to map between infinite-dimensional function spaces [11]. That is, they

can approximate mappings from one function to another function rather than just from one finite-dimensional input to a different finite-dimensional output. Certain neural operators, such as the Fourier neural operator (FNO) has been proven to be a universal approximator for function space [10].

There are several advantages to mapping between function spaces that are especially powerful in the context of fluid mechanics. First, mappings between function spaces are not necessarily bound by any set mesh or grid. This means that neural operators can be made mesh invariant, allowing for models to be trained and evaluated at arbitrary resolutions. In the context of fluid mechanics where both computational and experimental methods produce data in grids of varying resolution, this enables training a single model with different data sources without the need for interpolation. It also allows for the resulting model to be evaluated at higher resolution than the input provides (i.e., super-resolution).

Secondly, mapping between function spaces allows us directly approximate the solution operator to a family of partial differential equations [11]. This means that neural operators are especially well-suited for learning system for which the dynamics can be described by a set of partial differential equations, such as fluid flows. Directly learning the solution operator allows for neural operators to learn generalized solutions not limited to a single set of boundary conditions or initial conditions.

Being able to estimate solutions for various conditions with a single set of parameters is what sets neural operators apart from physics-informed neural networks (PINNs). As mentioned briefly in the previous chapter, PINNs are standard neural networks that are trained to estimate continuous solutions that fit a set of finite measurements to a known governing equation. These estimated continuous solutions can then be evaluated at points not included in the original measurement. However, PINNs must be retrained entirely for each set of conditions as each solution requires a unique set of parameters. However, neural operators have previously been demonstrated in a similar physics-informed manner [15].

Previous applications of neural operators to fluid flows have been limited to the computational domain. Fourier neural operators, which we deploy in Chapter 4, were demonstrated on fluid flow solutions when first introduced in Li et al. [13]. They have since been applied to other fluid flows in a follow up work, Li et al. [16], where they approximated solutions to a wavy pipe flow and a transonic airfoil to

demonstrate how FNOs can learn with general geometries. Another neural operator, DeepONet, has been applied to computational simulations as well [3, 17]. To the best of our knowledge neural operators of any form have not been previously applied to experimental fluid measurements.

Bibliography

[1] H. Jane Bae and Petros Koumoutsakos. Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nature Communications*, 13, 2022.

[2] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[3] Patricio Clark Di Leoni, Lu Lu, Charles Meneveau, George Em Karniadakis, and Tamer A. Zaki. Neural operator prediction of linear instability waves in high-speed boundary layers. *Journal of Computational Physics*, 474:111793, 2023.

[4] Dixia Fan, Liu Yang, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. Reinforcement learning for bluff body active flow control in experiments and simulations. *Proceedings of the National Academy of Sciences*, 117(42):26091–26098, October 2020. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2004939117. URL `http://www.pnas.org/lookup/doi/10.1073/pnas.2004939117`.

[5] Paul Garnier, Jonathan Viquerat, Jean Rabault, Aurelien Larcher, Alexander Kuhnle, and Elie Hachem. A review on deep reinforcement learning for fluid mechanics. *Computers and Fluids*, 225, 2021.

[6] M. Gazzola, A. A. Tchieu, D. Alexeev, A. de Brauer, and P. Koumoutsakos. Learning to school in the presence of hydrodynamic interactions. *Journal of Fluid Mechanics*, 789.

[7] Peter Gunnarson, Ioannis Mandralis, Guido Novati, Petros Koumoutsakos, and John O. Dabiri. Learning efficient navigation in vortical flow fields. *Nature Communications*, 12, 2021.

[8] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[9] Yusheng Jiao, Feng Ling, Sina Heydari, Nicolas Heess, Josh Merel, and Eva Kanso. Learning to swim in potential flow. *Physical Review Fluids*, 6(5): 050505, 2021.

[10] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22, 2021.

[11] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *arXiv preprint arXiv:2108.08481*, 2021.

[12] Jichao Li and Mengqi Zhang. Reinforcement-learning-based control of confined cylinder wakes with stability analyses. *Journal of Fluid Mechanics*, 932, 2022.

[13] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[14] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Proc. of Neural Information Processing Systems*, 2020.

[15] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

[16] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022.

[17] Chensen Lin, Martin Maxey, Zhen Li, and George Em Karniadakis. A seamless multiscale operator neural network for inferring bubble dynamics. *Journal of Fluid Mechanics*, 929, 2021.

[18] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[19] Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Reglade, and Nicolas Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865, 2019.

[20] C. Raibaudo, P. Zhong, B.R. Noack, and R.J. Martinuzzi. Machine learning strategies applied to the control of a fluidic pinball. *Physics of Fluids*, 32, 2020.

[21] Gautam Reddy, Jerome Wong-Ng, Antonio Celani, Terrence J. Sejnowski, and Massimo Vergassola. Glider soaring via reinforcement learning in the field. *Nature*, 562(7726):236–239, October 2018. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-018-0533-0. URL `http://www.nature.com/articles/s41586-018-0533-0`.

[22] Feng Ren, Jean Rabault, and Hui Tang. Applying deep reinforcement learning to active flow control in weakly turbulent conditions. *Physics of Fluids*, 33(3): 037121, 2021.

[23] Satoshi Shimomura, Satoshi Sekimoto, Akira Oyama, Kozo Fujii, and Hiroyuki Nishida. Closed-loop flow separation control using the deep q network over airfoil. *AIAA Journal*, 58, 2020.

[24] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Mass., second edition, 2018. ISBN 978-0-262-03924-6.

[25] Hongwei Tang, Jean Rabault, Alexander Kuhnle, Yan Wang, and Tongguang Wang. Robust active flow control over a range of reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Physics of Fluids*, 32(5):053605, 2020.

[26] Mikhail Tokarev, Egor Palkin, and Rustam Mullyadzhanov. Deep reinforcement learning control of cylinder flow using rotary oscillations at low reynolds number. *Energies*, 13(22):5920, 2020.

[27] Siddhartha Verma, Guido Novati, and Petros Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 2018.

[28] J. Viquerat, P. Meliga, and E Hachem. A review on deep reinforcement learning for fluid mechanics: an update. *arXiv:2107.12206v2*, 2021.

[29] Hui Xu, Wei Zhang, Jian Deng, and Jean Rabault. Active flow control with rotating cylinders by an artificial neural network trained by deep reinforcement learning. *Journal of Hydrodynamics*, 32(2):254–258, 2020.

[30] Yi Zhu, Fang-Bao Tian, John Young, James C Liao, and Joseph Lai. A numerical study of fish adaption behaviors in complex environments with a deep reinforcement learning and immersed boundary–lattice Boltzmann method. *Scientific Reports*, 11(1):1–20, 2021.

Chapter 3

MACHINE LEARNING FOR FLOW-INFORMED AERODYNAMIC CONTROL IN TURBULENT WIND CONDITIONS

Peter I. Renn and Morteza Gharib. Machine learning for flow-informed aerodynamic control in turbulent wind conditions. *Communications Engineering*, 1(45), 2022. doi: 10.1038/s44172-022-00046-z.
P.R. participated in the conception of the project, designed and built the experimental setup, adapted the algorithms for deployment, and wrote the manuscript.

3.1 Chapter abstract

Control of aerodynamic forces in gusty, turbulent conditions is critical for the safety and performance of technologies such as unmanned aerial vehicles and wind turbines. The presence and severity of extreme flow conditions are difficult to predict, and explicit modeling of fluid dynamics for control is not feasible in real-time. Model-free reinforcement learning methods present an end-to-end control solution for nonlinear systems as they require no prior knowledge, can easily integrate different types of measurements, and can adapt to varying conditions through interaction. Here, we show that reinforcement learning methods can achieve effective aerodynamic control in a highly turbulent environment. Algorithms are trained with different neural network structures, and we find that reinforcement learning agents with recurrent neural networks can effectively learn the nonlinear dynamics involved in turbulent flows and strongly outperform conventional linear control techniques. We also find that augmenting state observations with measurements from a set of bioinspired flow sensors can improve learning stability and control performance in aerodynamic systems. These results can serve to inform future gust mitigation systems for unmanned aerial vehicles and wind turbines, enabling operation in previously prohibitively dangerous conditions.

3.2 Introduction

Atmospheric winds are often turbulent, containing transient flow disturbances which result in intermittent aerodynamic forces [5]. These forces affect many systems and structures, but are especially impactful on inherently aerodynamic technologies. For example, unmanned aerial vehicles (UAVs) and wind turbines both rely on fluid

interaction for normal operation, but can be damaged or destroyed when operating in turbulent wind conditions [18, 41, 53]. Mitigating the effects of these turbulent forces through active control strategies is an ongoing challenge for both UAV and wind turbine applications [8, 16, 18, 19, 33, 41, 46]. However, turbulent conditions in the atmosphere are highly nonstationary and nonlinear making them difficult to model or control in real-time [5, 6].

Biological systems have long inspired engineers aspiring to develop systems robust to chaotic environments. Along with inertial and visual cues, sensed through vestibular, proprioceptive, or ocular systems, some animals navigate turbulent and unsteady environments through use of biological flow sensors [4]. For example, the lateral line is a sensory system common to most fish species, and is typically made up of hundreds of neuromasts located all along the fish's body [3]. The flow information observed by the lateral line allows fish to sense disturbances remotely, which can be used for finding prey, avoiding predators, achieving schooling formations, and navigating turbulent waters [9, 10, 17, 47]. Similarly, often admired for their acrobatic flight capabilities, bats have a set of microscopic hairs located on their wings that sense airflow and may enable enhanced control of unsteady aerodynamics [25, 43, 44]. Finally, large sea birds of the Procilliforme family (e.g., Wandering Albatross, Giant Petrel) use their tubular-shaped nostrils to sense and ride turbulent air-gusts from breaking waves, allowing for long-distance flight with minimal energy expenditure [32].

The impressive capabilities of these biological systems has previously motivated research for enhanced control of underwater and aerial autonomous vehicles through bio-inspired flow sensors [12, 14, 20, 29, 35, 48, 49, 52]. This work has generally implemented basic proportional-integral-derivative (PID) control techniques when testing flow sensory systems. While PID control is very effective for systems that can be linearized, the dynamics of turbulent flow are strongly nonlinear and cannot be reduced to even locally linearized approximations [6, 26, 42]. To properly realize the potential of bio-inspired flow sensing for control, nonlinear control methods are needed.

Model-free reinforcement learning (RL) is a machine learning framework that can be formulated to control nonlinear systems without any prior knowledge or modeling of the system dynamics. RL methods were principally inspired by biological learning theories regarding how animals learn new behaviors through repeated trial-and-error [45]. In RL, these trial-and-error interactions fit the formal framework of

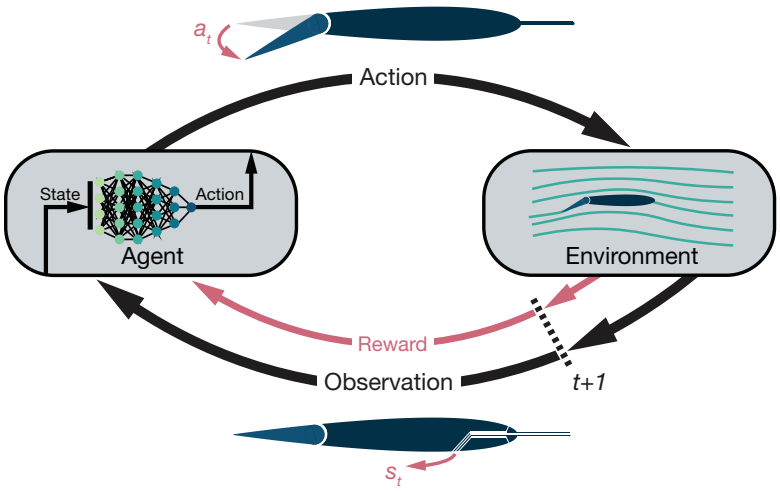

Figure 3.1: Schematic depicting the use of a Markov decision process framework, where agents take actions based on their observations of their environment, to address the problem of turbulent disturbances.

a Markov decision process framework (figure 3.1), where discrete time-steps consist of observing the state of the environment and choosing an action based on that observation. A numerical reward associated with each previous interaction is used to learn and improve the action decision-making. Observations can be comprised of any available state information, which the agent learns to interpret through experience alone. Actions can consist of any actuations or manipulations that the agent can physically realize. Capable of learning control policies and observation interpretations through direct interactions with physical phenomena, it may hold potential for flow-informed aerodynamic control in turbulent, non-stationary conditions [7].

Here, we experimentally investigate the use of state-of-the-art RL methods provided integrated flow information for aerodynamic control in a highly turbulent and vortical environment. RL algorithms are implemented on an aerodynamic testbed consisting of a wing with actuated trailing-edge flaps. Responding to incoming disturbances at each time-step by adjusting the position of the trailing-edge flaps to control the aerodynamics of the system, we set the goal of minimizing the standard deviation of lift in an unsteady, turbulent flow field. The wing system features an array of pressure sensors used to observe the aerodynamic state, and is mounted on a load-cell which can be used to observe the inertial state. We use recurrent neural networks (RNNs) to improve learning in a highly stochastic and partially observable physical environment that cannot be simulated trivially in a computational setting. Through

these wind tunnel experiments, we show that RL agents are able to effectively integrate flow knowledge and achieve superior disturbance rejection relative to a conventional linear controller (i.e., PID). Overall, we find that model-free RL methods are capable of learning and controlling physical aerodynamic systems in turbulent and highly irregular flow fields.

3.3 Results

Flow-informed aerodynamic testbed

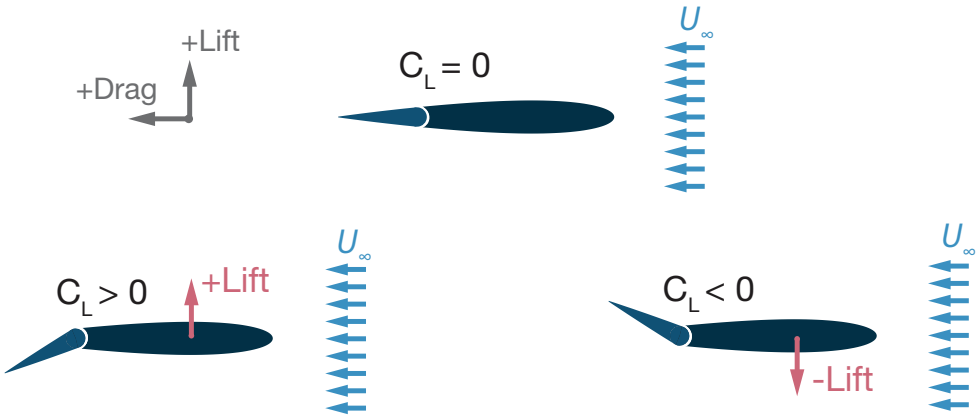

Figure 3.2: Symmetric airfoils produce zero lift when aligned with uniform flow. Changing position of a trailing edge flap can change the coefficient of lift and create an aerodynamic force either upward or downward.

The problem of an aerodynamic system in turbulence was abstracted and generalized to a basic setting. We developed a testbed consisting of a symmetric airfoil with motorized trailing-edge flaps and integrated flow sensors. A generic platform such as this is ideal for aerodynamic study as it can easily be abstracted to more specific applications such as fixed-wing UAVs and wind turbine blades. In a uniform flow symmetric airfoils have lift coefficient $C_L = 0$ at 0 deg angle-of-attack, which means no aerodynamic force is produced in the upwards or downwards direction. As illustrated in figure 3.2, the lift coefficient of a symmetric airfoil in a uniform flow can be manipulated by adjusting the position of a trailing-edge flap which results in a non-zero force along the lifting axis.

The wing system (figure 3.3) was designed to be modular, allowing for variation of sensor type and placement. For this work, the model was configured to include nine sensors at different positions placed 10 cm apart from one another along the spanwise axis. The center position sensor featured a pitot-static tube for measuring

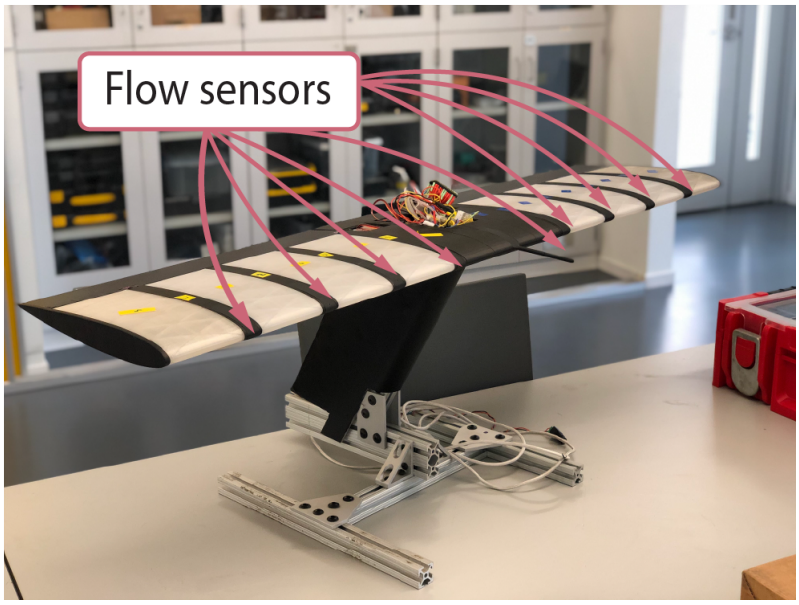

Figure 3.3: The wing system used for training featured a modular design, as shown by the different color sections. Each section is removable and replaceable, and the locations of flow sensors are labelled. The wing is 1 m in length (see Methods for more details).

mean flow velocity. The remaining eight sensors consisted of pressure taps placed at various chord lengths near the leading edge (details in Methods). The locations of the sensors were chosen based on previous works for aerodynamic parameter estimation from sparse on-body flow measurements [23, 39].

Turbulent environment

Figure 3.4: Smoke visualizations showing the turbulent wake of a standard cylinder at Reynolds number $Re_D = 50000$. This was taken in the Caltech Center for Autonomous Systems and Technologies fan-array wind-tunnel at lower Reynolds number for purposes of visualization. The actual flow conditions used for testing and training were too turbulent for effective smoke visualization.

Bluff body wakes are a well-studied problem in fluid mechanics. Famously, these wakes can feature a phenomenon commonly known as a Kármán vortex street, which consists of alternating vortices shedding at a fixed frequency. However the behavior of vortex shedding in a bluff body wake can change, and there exists a well-established relationship between this behavior and the Reynolds number of the flow [1, 2, 37, 54]. At a sufficiently high Reynolds number, the wake becomes turbulent and vortex shedding, while still present, becomes less regular (figure 3.4) [1]. To simulate a gusty environment, we placed our wing system in the wake of a large, asymmetric bluff body which was mounted on elastic bands in a wind tunnel (figure 3.5). The asymmetry of the body and dynamic mounting produces highly irregular turbulent disturbances. This flow field is not intended exactly match any specific atmospheric conditions, but rather to create challenging environment with frequent large amplitude vortical disturbances.

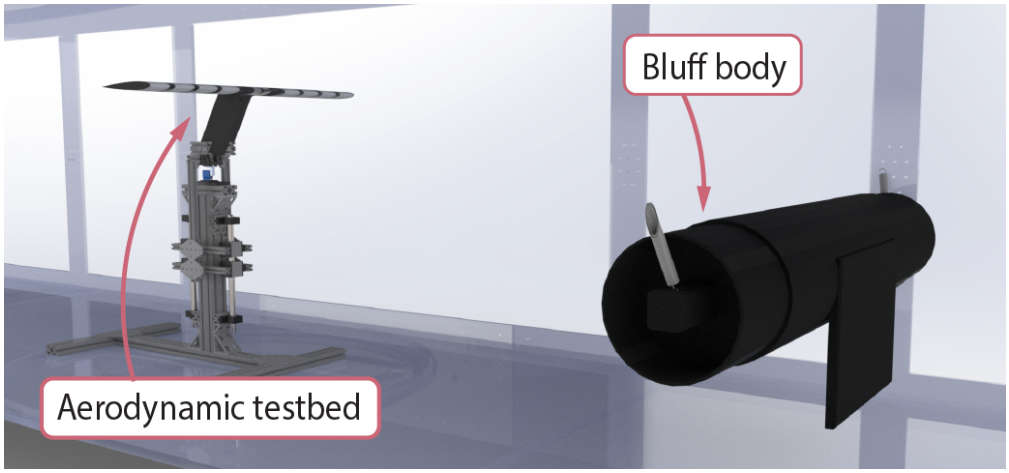

Figure 3.5: The system was trained in the wake of an asymmetric bluff body in a conventional closed-loop wind tunnel. The cylindrical portion of the bluff body has diameter of 30 cm, and was placed 170 cm upstream of the wing system.

A hot-wire anemometer, placed near the leading edge of the wing, characterized the velocity, turbulence intensity, and frequency spectrum of the flow (see Methods for details). The mean velocity was recorded as 6.81 m s^{-1}, which corresponds to a Reynolds number approximately $Re_D = 230000$ over the bluff body. Figure 3.5 depicts the power spectral density calculated from the hot-wire anemometer measurements. Here we see a peak at 2.44 Hz, which can be assumed to be the primary vortex shedding frequency. This corresponds to a Strouhal number of St= 0.19, which is in good agreement with the expected value for a bluff body wake at this Reynolds number [1]. However, the power spectrum also suggests that there is

much energy stored at frequencies lower than the primary vortex shedding frequency, given the width of the high-energy, low frequency region. This indicates that flow disturbances are highly irregular in length and time scales and demonstrates the presence of gust-like disturbances arriving at random intervals. We also note the energy decay beginning at the primary shedding frequency which follows a -5/3 power law, agreeing with theory of the turbulent energy cascade [34, 38].

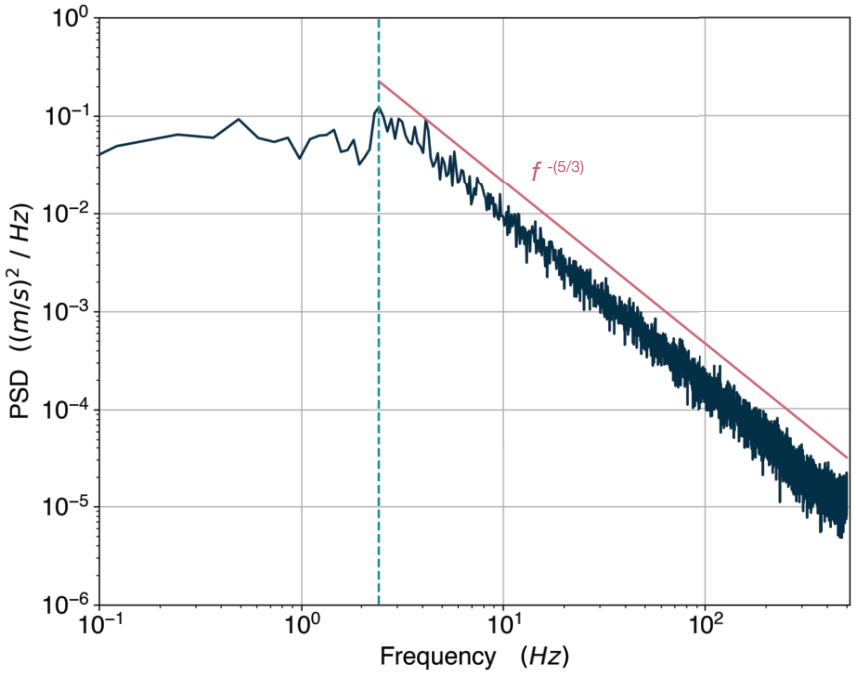

Figure 3.6: Power spectrum measured in the bluff-body wake plotted logarithmically. We note the peak frequency at 2.44 Hz (dashed line), the relative width of the high-energy low frequency region (left of dashed line), and the –5/3 slope energy decay that agrees with turbulence theory (right of dashed line)

Model-free reinforcement learning

To achieve flow-informed aerodynamic control of our wing system, we implemented the Twin Delayed Deep Deterministic Policy Gradient algorithm (TD3) as well as variant known as LSTM-TD3 [13, 27]. These are off-policy actor-critic type algorithms which use neural networks to make control policy decisions (see Methods). TD3 was previously deployed successfully for experimental flow control in a different setting [11]. LSTM-TD3 features a modification to the neural network structure of TD3 to include recurrent Long-Short-Term-Memory (LSTM) cells. The pres-

ence of recurrent cells in neural networks can considerably improve performance in partially observable systems, which can impact performance and prediction in highly stochastic settings such as the development of turbulent flows [27, 51]. Since training RL algorithms is an inherently stochastic process, we trained each of the agents presented here with five separate random seedings and averaged the results to show general performance. Each agent was trained for 200 episodes which took approximately 150 minutes per agent.

Through training, RL algorithms attempt to learn control policies that maximize a numerical reward signal which is prescribed beforehand to set the desired goal of learning. We designed our reward to hold a constant lifting force (arbitrarily set to zero) in the presence of flow disturbances, setting it equal to:

$$R_i = -(L_{i+1})^2 \tag{3.1}$$

where R_i is the reward at time-step i, and L_{i+1} is the lifting force at time-step $i + 1$ (see Methods for details). A perfect system would achieve zero lifting force at each time-step, giving a maximum possible reward equal to zero. We can use the mean accumulated reward of each episode as a measure for comparison between algorithms and as a basic indicator of learning behavior. As the name implies, the mean accumulated award is the sum of rewards accumulated in a single episode averaged across the five agents. We also use the standard deviation of lift calculated from the time-histories of each episode to evaluate performance directly related to disturbance rejection.

Comparison with baseline control methods

As a metric of baseline performance, we compared the RL algorithms with basic PID control. The PID controller was tuned manually and set to achieve constant zero lift with feedback from forces measured by the load cell. LSTM-TD3 and TD3 agents were provided near identical network parameters (see Methods). To gauge how the respective control policies reduce the effect of flow disturbances, we also measured the wing system sitting passively in the turbulent environment for comparison. Due to the randomized nature of training model-free algorithms, performance and consistency of learning are both important metrics when evaluating the behavior

of RL agents. The fully trained RL agents, PID control, and passive configuration were set to perform over approximately one minute each, and the resulting standard deviations of lift and calculated episodic mean accumulated reward are shown in Table 3.1.

Controller	Standard deviation of lift (mN)	Mean accumulated reward
No Control	305 ± 20	-
PID	264 ± 6	-
TD3	266 ± 79	-8960 ± 10728
LSTM-TD3	176 ± 11	-1716 ± 452

Table 3.1: Statistical comparison of control schemes. Rewards and standard deviation of lift values were averaged over five agents trained for both of the reinforcement learning algorithms and five separate runs for the proportional-integral-derivative (PID) control. They were calculated over a 4000 time-step horizon, which corresponds to approximately 1 minute of testing or four-times the length of a training episode. Uncertainty shown is equal to the standard deviation in the presented value across five separate training sessions. Supplementary figure 1 shows examples of the load signal over a 60 second interval for all four methods listed in the table below

PID offered only modest improvements in disturbance rejection over no control, with a 13% reduction in standard deviation of lift. The TD3 algorithm had a similar reduction in standard deviation as PID, however also demonstrated large variation between agents, indicating inconsistency in its ability to learn the system dynamics. The LSTM-TD3 algorithm performed well, reducing standard deviation of lift by 42% relative to the passive case. Noting that the maximum reward possible is zero, we also see that TD3 accumulates a negative over five-times greater than that of LSTM-TD3. Additionally, as indicated by the uncertainties, LSTM-TD3 was also more consistent across agents despite the stochasticity of training. These uncertainties, calculated as the standard deviation of the respective quantities across the five agents are themselves indicators of training stability.

Further, the mean accumulated reward (per episode) plot (figure 3.7) indicates that LSTM-TD3 agents consistently improved throughout training, whereas the TD3 agents struggled to find even locally optimal behaviors. While the reward signal returned for the TD3 agents became less erratic with training, it eventually decays and showed a downward trend during the final episodes with increased variance. This suggests that the dynamics learned by TD3 are not representative of the real system. As expected from the uncertainty in the standard deviation of lift (Table 1),

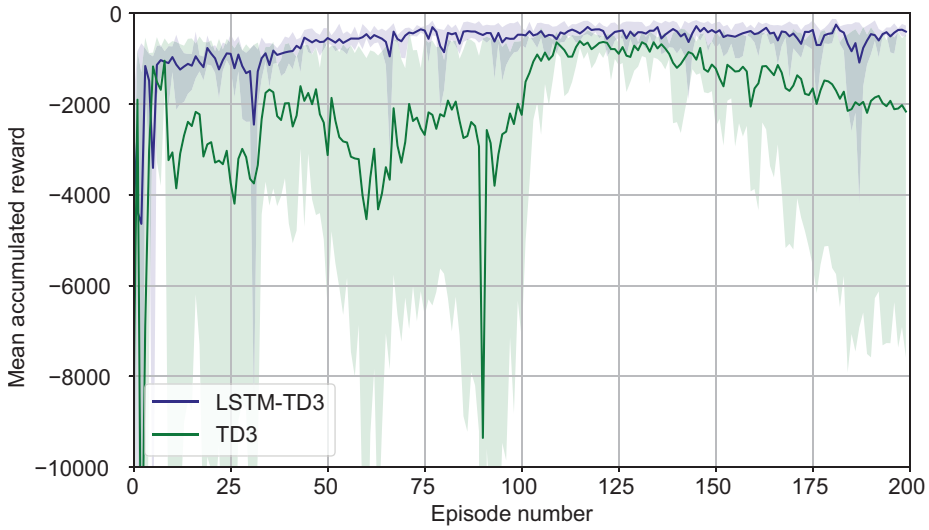

Figure 3.7: Training performance of TD3 and LSTM-TD3. The respective shaded regions represent the full range of performance of each algorithm at each episode. Here we plot the learning curve showing the episodic mean accumulated reward across the five agents trained for each algorithm.

we can confirm that the learning process for the TD3 is inconsistent and there exists variation across the separately trained agents. The episodic standard deviation of lift (figure 3.8) remains relatively stable after approximately 100 episodes of training for both algorithms.

The lack of reduction in standard deviation of lift from both agents across such a large span of episodes suggests that they have reached asymptotic performance for the given conditions. Despite these two methods being nearly identical algorithmically, it seems that the simple inclusion of RNNs in LSTM-TD3 makes aerodynamic control tractable in this turbulent environment.

Effect of flow sensing

Conventional control strategies for UAVs mitigate turbulent disturbances by sensing and correcting the resulting inertial deviations. They have no knowledge of the flow or source disturbance itself. This purely reactive-corrective strategy is insufficient for maintaining stability under extreme atmospheric turbulence [28]. Alternatively, as biological swimmers and flyers would imply, directly observing the physics responsible for inertial disturbances may allow for aerodynamic systems

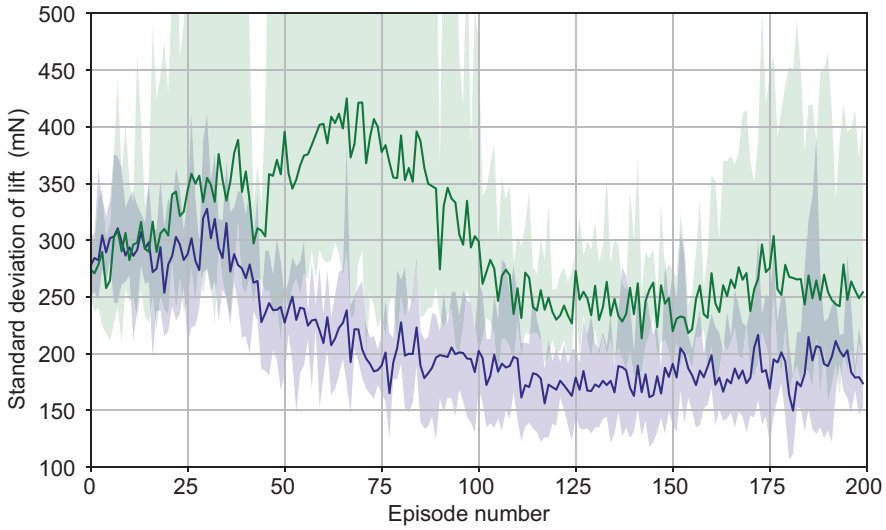

Figure 3.8: Training performance of TD3 and LSTM-TD3. The respective shaded regions represent the full range of performance of each algorithm at each episode. Here we plot the episodic standard deviation for the two algorithms as it decreases with training.

to react before inertial effects are realized. The flow sensing capabilities of biological systems can then be used as inspiration to improve these strategies, given the direct correlations between easily measurable flow quantities, such as pressure, and aerodynamic forces.

To show the effect of flow sensing on the performance of aerodynamic control in turbulence, we conducted ablation studies wherein we maintained the same reward signal but varied the sensory information provided to the agent. We considered three cases to establish how flow sensing impacts the ability to learn system dynamics. In Case I, the RL agents observed and chose actions based on the value of the lift force alone through the real-time load cell values. Observing only the lift force, the RL agent in Case I were effectively provided with inertial information and therefore acted equivalent to conventional UAV controllers. In Case II, actions were selected using only flow measurements as the observation, though the lift measurements were used to calculate rewards during an offline training phase. In Case III, the agents observed both the lift force and flow measurements. With both inertial and flow information, Case III was afforded a set of information similar to a flow-sensing biological flyer. All three cases were trained using the LSTM-TD3 algorithm; they differed only in the sensory information provided to the agent for action selection.

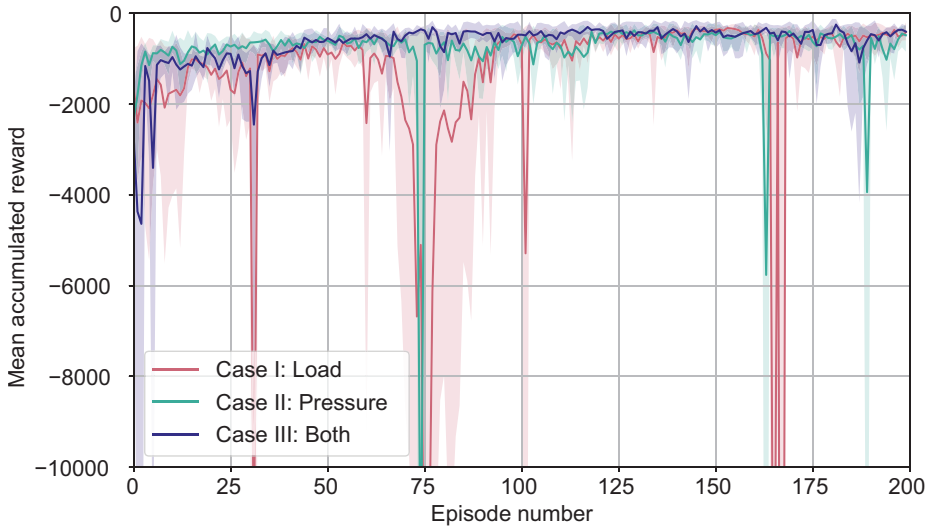

Figure 3.9: Training performance of reinforcement learning algorithms with varying observations. The respective shaded regions represent best and worst performing agent of each case at each episode. Here we show the learning curves for the three respective cases, plotting episodic mean accumulated reward.

From a comparison of the respective episodic reward signals (figure 3.9), we found that the three cases seem to learn similarly effective control strategies in after training 200 episodes. While all three cases occasionally experienced policy updates that decreased performance (as indicated by downward spikes), these detrimental updates appeared most frequently and most strongly for the Case I agents. The Case II agents also had several notable bad policy updates, but recovered more quickly than the Case I agents. The Case III agents learned more stably and reliably than the Case I and Case II agents, with the best Case III agents consistently outperforming the best Case I and Case II agents throughout training, and with the worst Case III agents rarely performing worse than equivalent Case I and Case II agents (highlighted regions-figure 3.9). Case III agents also achieved the lowest mean standard deviation of lift and lowest minimum standard deviation across agents for most episodes throughout training (figure 3.10).

In addition to considering the performance during learning, we also compared performance of the fully trained RL agents for all three cases. We averaged the performance of fully trained models for all three cases over a time interval of approximately one-minute to reduce the effect of stochasticity in the flow. From the final time-averaged standard deviation of lift values for the three cases (Table

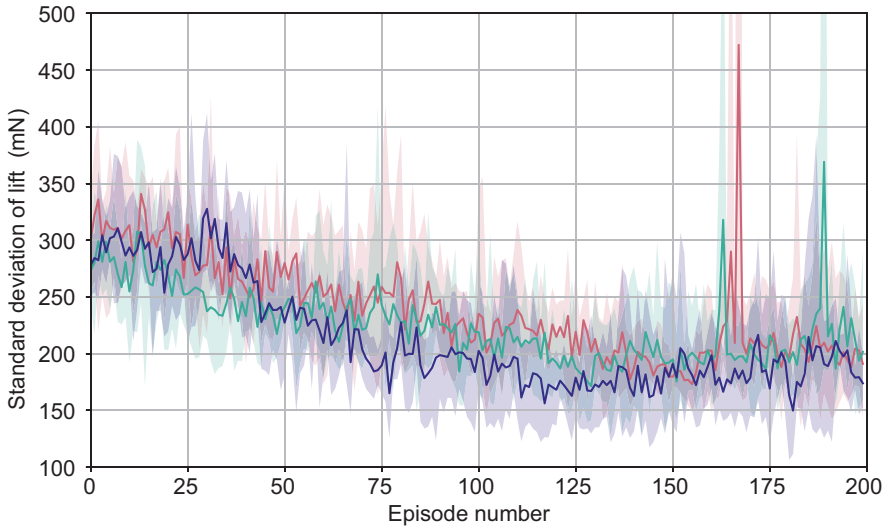

Figure 3.10: Training performance of reinforcement learning algorithms with varying observations. The respective shaded regions represent best and worst performing agent of each case at each episode. Here we show the change in standard deviation throughout learning plotted as a metric for disturbance rejection.

3.2) we see all three agents considerably reduced the variance of lift relative to the passive case. We find that that Case III shows superior performance in reducing the standard deviation of lift and is the most consistent in that metric across the five separately trained agents. Both the training process (figure 3.9-3.10) and the final standard deviation (Table 3.2) suggests that the addition of flow sensing helps RL agents learn a more stable approximation of the system dynamics and improved performance in terms of disturbance rejection (i.e., reduction of standard deviation).

Case	Standard deviation of lift (mN)	Mean accumulated reward
No Control	305 ± 20	-
I (Load)	191 ± 20	-1696 ± 492
II (Pressure)	199 ± 33	-1860 ± 596
III (Both)	176 ± 11	-1716 ± 452

Table 3.2: These values were averaged over five agents trained for each of the respective cases, and were calculated over a 4000 time-step horizon, which corresponds to approximately 1 minute of testing or four-times the length of a training episode. Uncertainty shown is equal to the standard deviation in the presented value across five separate training sessions. Supplementary figure 2 shows examples of the load signal over a 60 second interval for the three different observations.

Interestingly, we find that Case I achieves the best (least negative) mean accumulated reward out of the three cases, with Case III falling closely behind by only a small margin. We note that the uncertainty reported for the reward of these two cases is approximately twenty-times the apparent difference in performance, which reduces the significance of this comparison. Still, it is not surprising that the Case I performance excels in terms of raw reward, as the only observation given in Case I is directly proportional to the reward itself. Considering that the Case I agents achieve a higher standard deviation of lift, the performance in terms of reward is the result of the agents holding a lower mean lift. Although Case I agents are given less information about the surrounding physics, the information they are given has only one dimension and excludes highly non-linear flow sensor readings. These attributes would enable a more simple and less sensitive control policy which simultaneously explains a lower mean lift value and less responsive disturbance rejection.

3.4 Discussion

We have demonstrated how properly configured RL agents can effectively learn control of nonlinear stochastic physics with which conventional methods struggle. Despite the seemingly chaotic nature of the turbulent environment used for training, our results indicate that RNNs enhance the ability to learn accurate system dynamics. Further, the inclusion of flow sensors, as inspired by biological systems, showed potential for enhanced aerodynamic control in turbulence.

We found that the performance achieved by the TD3 agents was very similar to that of a conventional PID controller (Table 3.1). This result was surprising, as the TD3 agent should be able to better handle the non-linearities of the system dynamics than the inherently linear PID controller. The poor performance of the TD3 algorithm (in comparison with LSTM-TD3) may be explained by the partial observability of our system. It is likely that observing inertial and sparse flow measurements at a single time-step does not adequately define the state; the probability distribution of state transitions is dependent on the surrounding flow which is chaotic and impossible to fully observe in real. Therefore, without being explicitly given a more comprehensive state observation, the TD3 agents are unable to infer the underlying state and effectively learn the underlying physics. This difficulty to learn the underlying physics of the problem would help explain the large variation in both training and end performance for the TD3 agents (figure 3.7-3.8).

We showed that LSTM-TD3 agents were able to achieve effective control of the system aerodynamics by analyzing the available observations sequentially, and outperformed both PID controllers and TD3 agents. In fact, the LSTM-TD3 agents were able to decrease the standard deviation in lift by more than three-times the reduction achieved by PID control. Further, despite being trained in five separate randomized processes, the final performance of the LSTM-TD3 agents is nearly as consistent as the five trials used to average the fixed PID controller. This demonstrates the ability to learn accurate estimations of the state-probability distribution functions. Due to the similarity of the two algorithms, the addition of recurrent LSTM cells is very likely the reason for the difference in performance between the TD3 and LSTM-TD3 agents. The potential performance-enhancing nature of LSTMs is well established in many settings, including flow-control [27, 50]. When included in an RL agent recurrent networks, such as those including LSTM cells, are able to learn latent states and patterns underlying the received observations. Because of the black-box nature of these methods, it is only possible to speculate what aspects of the physics were learned through the addition of LSTM cells. However, the fundamental mechanism through which LSTM cells can typically improve performance is by integrating temporal information to improve state estimates. Therefore, the increase in performance associated with the LSTM cells is likely due to improved state estimates which would effectively increase the observability of the partially-observable process.

Flow sensing was shown to slightly improve mitigation of turbulent disturbances for our system, although it resulted in a larger bias in the averaged value than inertial information alone. Still, when provided flow and inertial information the RL agents did learn more consistently and achieved superior disturbance rejection than when given partial information (figure 3.9-3.10). Further, the fully trained control policies given both flow and inertial information varied less in all metrics, suggesting more stable estimates of system dynamics. It is also noteworthy that agents observing only flow measurements showed robust control improvement through learning; that is, agents were able to learn to control inertial dynamics from sparse flow sensing alone. While the load cell used for testing completely defines the inertial state of the system in the lifting direction, the pressure sensors used to make flow observations are relatively sparse and do not completely define the aerodynamic state in the lifting

direction. It is likely that the performance of flow-informed agents would increase further given additional or improved flow sensing capabilities, while the inertial aspects of the lifting force cannot be defined further than the direct measurement used here. This warrants further exploration of flow sensor types and configurations.

Though the RL controllers outperformed a conventional linear control scheme, there are several drawbacks to model-free reinforcement learning methods. Training RL agents is intensive in terms of both time and data. Each RL agent was trained for 200 episodes which took approximately 150 minutes per agent. Since we averaged the performance of five agents for each algorithm or case shown, the data presented here represents over 50 hours of wind-tunnel hours between training and testing the policies. There are also inherent difficulties associated with troubleshooting "black-box" controllers such as RL agents. The algorithms are sensitive to many hyperparameters that control the neural network structure and training procedure, and tuning these hyperparameters in an experimental setting relies on intuition, experience, and patience. The hyperparameter tuning process itself required hundreds of hours of additional training and testing not shown here. Further, it is possible that a given set of hyperparameters may be suitable for a subset of tasks but not truly generalizable. Consistent and deliberate experimental design helps constrain troubleshooting to the algorithmic aspects of training. Even with this, it should not be expected that these agents trained in a single set of conditions will hold policies generalizable across Reynolds numbers or testing geometries. To create truly generalizable RL policy capable of controlling an aerodynamic system ready for real-world deployment, the agent would need to train in various conditions and would need to expand its capabilities to control all forces and moments in three-dimensions.

While model-free reinforcement learning methods impressively learn dynamics of highly nonlinear and chaotic systems without any prior knowledge, it should be noted that model-based reinforcement learning and other non-linear control methods can be more data-efficient. Model-based methods do require prior sampling of system dynamics and can be more computationally intensive, but many are similarly able to adapt and learn when exposed to new conditions. Implementing known flow physics into model-based reinforcement learning methods or non-linear controllers could lead to superior performance with reduced data requirements.

The generic testbed and methods developed here may serve to inform future implementations of flow-informed RL for control of aerodynamic systems in extreme turbulence. While our experiments focus on specifically controlling turbulent disturbances along a single axis, this system is simply a representative proof-of-concept for multi-dimensional unconstrained aerodynamic interactions. These methods can be expanded or adapted to systems with higher degrees of freedom by augmenting state observations and adjusting reward signals. While it is possible for RL methods to be used for full control and navigation of autonomous systems [15, 36], the most direct and practical application of aerodynamic control for UAVs is in flow-informed inner-loop attitude control for fixed wing vehicles. By reducing the effect of turbulent disturbances, drones can maintain more stable flight in more extreme conditions. Though training RL agents to achieve full control of free-flight systems can be challenging experimentally due to the trial-and-error nature of the learning process, flow-informed agents even have the potential to learn to take advantage of natural flow structures through energy-efficient soaring behaviors [30, 32, 36]. This technology could allow wind turbines to safely operate at an increased range of conditions by reducing loads from potentially damaging gusts through actuation of blade pitch [19, 41, 46]. In the case of a static system such as wind turbines, off-board remote sensing upwind could further enhance performance. We believe that the potential of this work can be realized through several next generation technologies such as flow-informed wind turbines with built-in gust mitigation capabilities, bioinspired UAVs capable of maintaining steady flight in a windy urban environment, and other unrealized aerodynamic applications that have been too chaotic for engineered systems.

3.5 Methods

Wing system design and manufacturing

The wing system featured a NACA0012 airfoil, which is a common standardized airfoil shape. The dynamics of this airfoil shape in a bluff-body wake at similar Reynolds number has been the subject of previous study [24]. The body of the wing was 3D printed using a combination of materials, and was designed to be modular and allow for various sensor configurations (figure 3.3). The central section, which housed the primary electronics and secured the system to its mounting, was printed with micro carbon fiber filled nylon (Markforged Onyx) and was reinforced with carbon fiber for added strength and rigidity. The spanwise sections designed to house the individual pressure sensors were also printed with micro carbon filled

nylon, but were not reinforced. The sensor housing sections were printed with large slots, so that different pressure taps or probe types could be used. These pressure tap slots and the pitot-static tube were printed with an SLA printer (Formlabs Form3) for improved surface feature accuracy. The pressure ports were placed at locations 0.4%, 0.7%, 1.5% and 6% of the chord length from the leading edge on both the pressure and suction sides of the wing. The sections between sensors were printed with clear PLA. The sections were aligned and conjoined by a set carbon fiber spars, which added rigidity. The trailing edge flaps were cut out of insulation foam and covered with an adhesive-backed coating for protection.

The wing had a total chord length of 25 cm, with 5 cm trailing edge flaps. This gave a Reynolds number over the wing of approximately $Re_c = 110000$. The spanwise length of the wing was 1 m, with a total of 9 sensor locations. There was exactly 10 cm between each sensor location, with one of the locations being centered on the wing. The wing was mounted on a fairing which was set back with an angle of 60 degrees to reduce aerodynamic interactions between the fairing and the wing. The fairing was reinforced with carbon fiber and aluminum, and was connected to a set of air bearings (New Way) which are aligned vertically with the tunnel to define the lifting direction. The air bearing system allowed for nearly frictionless motion along a single axis while constraining all other directions. The constrained fairing was mounted directly to a single-axis load cell (Interface SM-50), which passed signal through an amplifier (Interface Model SGA) with a 50 Hz low pass filter, and was read by a DAQ (NI USB-6008). The pressure values were measured by a set of nine ultra-low range digital pressure sensors (Honeywell RSCDRRM2.5MDSE3) which communicated to a microcontroller (Teensy 4.0). The microcontroller also controlled the high speed brushless servo motors (MKS HBL6625) which drove the trailing edge flaps. Due to mechanical restraints, the actuation for the servo motors has maximum/minimum position of +40 deg/-40 deg. Both the microcontroller and the DAQ communicate with a desktop computer which receives states and sends actions. The full control loop ran at approximately 67 Hz with the serial communications being the biggest bottleneck.

Generation and characterizations of turbulence

All quantitative results presented are from experiments performed in Caltech's John W. Lucas Wind Tunnel (LWT). The LWT is a closed-loop wind tunnel, with test section dimensions of 130 cm × 180 cm. The turbulence used for training and testing formed in the wake of a large, asymmetric bluff body mounted to the wind tunnel

with bungee cords. The bluff body can be described as a large diameter cylinder (30 cm), with a normal flat plate mounted asymmetrically to the front giving the full body an effective diameter of 53 cm (figure 3.5). The cylinder spanned the entire width of the tunnel, while the flat plate had a width of only 60 cm. This was done to encourage vortex dislocation, which added irregularity in vortex shedding [54]. The bungee cord encouraged oscillations due to the vortex shedding, which we observed to be present and irregular. The bluff body was mounted 170 cm upstream of the wing system with a vertical offset of 48 cm. Sparse elastic bands aligned horizontally were mounted across the test section directly upstream of the bluff body, to further increase the turbulence intensity of the flow.

A hot-wire anemometer (TSI IFA-300) was used to characterize the mean velocity, turbulence intensity, and frequency spectrum of the flow near the wing system. The hot-wire anemometer was mounted approximately 2 cm upstream of the leading edge of the airfoil, and measurements were taken for 120 seconds at 1000 Hz. The turbulence intensity was measured with the hot-wire anemometer to be 10.6%. The power spectra was calculated with ThermalPro software, using the entire 120 second run averaged over sets of 8192 datapoints (frequency resolution of 0.122 Hz).

Reward functions for training

Reinforcement learning agents learn to achieve tasks by choosing actions that maximize a numerical reward function. Choosing the reward function is a critical part of experimental design for RL implementations, as it sets the primary directive of learning. The goal of our experiments was to reduce the standard deviation of the lifting force, and we tested several different reward functions to achieve this goal. The final reward function used in this work was set to

$$R_i = -(L_{i+1})^2 \tag{3.2}$$

where R_i is the reward associated with the action taken at time-step i, and L_{i+1} is the lifting force observed at the time-step $i + 1$. The reward was a function of the lift at the following time-step rather than the current time-step, because this value is the direct result of the previous action. Since RL agents are designed to maximize reward signals, we used the negative square of this value to encourage net-zero lifting force and discourage deviations. Under these conditions, a perfect agent would achieve a maximum reward equal to zero.

This reward function is undeniably simplistic; it integrates virtually no prior domain knowledge and makes unproven assumptions about the physical system. This highlights a significant challenge in the deployment of model-free reinforcement learning algorithms for real-world applications. While we may like to shape our reward with domain knowledge, the flow sensor state observations do not provide an interpretable or intuitive description of the non-linear dynamics with which we can embed physics without explicit modeling (therefore negating the primary benefit of model-free reinforcement learning methods). Additionally this reward function assumes that the force observed at a given time-step is determined entirely by the previous action. That is, that the aerodynamic state is not being influenced by transient effects of previous actions as well. This itself is likely violated by the use of Kalman filtering on the load signal alone, which filters current observations based on the change from previous observations. It is therefore possible that the improvement seen from integrating LSTM cells can be, in-part, attributed to the critic network learning action-values based on a time series of actions rather than the action at a single point. However, we also note that the best TD3 agents were able to learn policies approaching the average LSTM-TD3 agents learning only the association between a single action and reward, which indicates that a load at a given time-step is at least strongly influenced by the action taken at the previous time-step.

This simple reward function was ultimately chosen out of several tested which featured additional terms (e.g., inclusion of future loads, the difference between loads observed before and after, etc.) and none of which notably improved performance in our preliminary tests. We note that this reward function was even used for the agents trained with flow sensor values only. The agents in this case had no direct knowledge of the lift while making decisions, and it was measured and saved separately from the state observations. These flow-only agents were only affected by the actual lift measurements through off-line training with the reward as shown.

Reinforcement learning algorithms

Both the TD3 and LSTM-TD3 fall in a category of RL algorithms known as actor-critic methods. As the name suggests actor-critic methods consist of two parts, the actor and the critic. The actor portion holds the direct control policy of the agent; it is called at each time-step, with observations as inputs and actions as outputs. The critic portion is used to estimate the value of each available action given a state. The value of an action is equal to the expected cumulative future reward [45]. In most modern applications, both actors and critics take the form of

artificial neural networks. The critic networks become approximators for the value function, and the actor network becomes an approximator for an optimal control policy. The agent learns by first training the value function estimation of the critic based on past state-action-reward experiences. The actor is then trained using the critic network to choose the actions which the critic estimates will have the highest expected value. Actor-critic methods are known to have reduced variance in updates, which accelerates learning and makes them well-suited for real-world applications [45]. The TD3 (provided in Algorithm 1) and LSTM-TD3 (provided in Algorithm 2) algorithms build on this basic framework with several modifications to improve performance.

Algorithm 1 TD3 algorithm used for aerodynamic control [11, 13]

Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $n = 1$ **to** N_e **d**
  for $t = 1$ **to** T **do**
    Observe state o_t
    Select action with exploration noise:
    $a_t \leftarrow \text{clip}\big(\pi_\phi(o_t) + \epsilon_t, -1,1\big), \epsilon_t \sim \mathcal{N}(0, \sigma^2)$
    Observe reward r_t
  end for
  Store transition tuples $\{(s_t, a_t, r_t, s_{t+1})\}_{i=1}^{T-1}$ in $\mathcal{B}$
  for $j = 1$ **to** N_s **do**
    Sample N transitions (s, a, r, s') from $\mathcal{B}$
    $\tilde{a} \leftarrow \text{clip}\big(\pi_{\phi'}(s') + \epsilon, -1, 1\big), \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}^2), -c, c)$
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'}(s', \tilde{a})$
    Update critics $\theta_i \leftarrow \min_{\theta_i} N^{-1} \sum \begin{cases} \big(y - Q_{\theta_i}(s,a)\big)^2, & |y - Q_{\theta_i}(s,a)| \le \delta \\ \delta * \left(|y - Q_{\theta_i}(s,a)| - \frac{1}{2}\delta\right), & \text{else} \end{cases}$
    if t mod d **then**
      Update ϕ by the deterministic policy gradient:
      $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
      Update target networks
      $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
      $\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i'$
    end if
  end for
end for

We chose the original TD3 algorithm to start our tests because it is known to among state-of-the-art methods in RL and has been shown to outperform earlier methods in simulated tasks [22]. Further, the original TD3 algorithm was used previously effectively learn control in the first experimental application of RL for explicit control

of fluid dynamics [11]. The LSTM-TD3 algorithm was chosen as a direct successor to the TD3 algorithm, which explicitly addresses the problem of partially observable systems [27]. Gradient clipping was added to both algorithms, to limit the size of updates and encourage training stability. Additionally, as suggested by a previous implementation of RL for experimental fluid mechanics, a Kalman filter was applied to both load and pressure data which was shown to considerably improve learning [11].

Algorithm 2 LSTM-TD3 algorithm used for aerodynamic control [27]

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $n = 1$ **to** N_e **do**
  Initialize past history $h_1^l \leftarrow 0$
  for $t = 1$ **to** T **do**
    Observe state s_t
    Select action with exploration noise:
    $a_t \leftarrow \text{clip}\left(\pi_\phi(s_t, h_t^l) + \epsilon_t, -1, 1\right), \epsilon_t \sim \mathcal{N}(0, \sigma^2)$
      Update history:
      $h_{t+1}^l = (h_t^l - (s_{t-l}, a_{t-l})) \cup (s_t, a_t)$
    Observe reward r_t
  end for
  Store transition tuples $\{(h_t^l, s_t, a_t, r_t, s_{t+1})\}_{i=1}^{T-1}$ in $\mathcal{B}$
  for $j = 1$ **to** N_s **do**
    Sample N transitions (s_t, a_t, r_t, s_{t+1}) from $\mathcal{B}$
    $\tilde{a} \leftarrow \text{clip}\left(\pi_{\phi'}(s_{t+1}, h_{t+1}^l) + \epsilon, -1, 1\right), \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}^2), -c, c)$
    $y \leftarrow r_t + \gamma \min_{i=1,2} Q_{\theta'}(s_{t+1}, \tilde{a}, h_{t+1}^l)$
    Update critics $\theta_i \leftarrow \min_{\theta_i} N^{-1} \sum \begin{cases} \left(y - Q_{\theta_i}(s_t, a_t, h_t^l)\right)^2, & \left|y - Q_{\theta_i}(s_t, a_t, h_t^l)\right| \le \delta \\ \delta * \left(\left|y - Q_{\theta_i}(s_t, a_t, h_t^l)\right| - \frac{1}{2}\delta\right), & \text{else} \end{cases}$
    if j mod d **then**
      Update ϕ by the deterministic policy gradient:
      $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s_t, a_t, h_t^l)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s_t, h_t^l)$
      Update target networks
      $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
      $\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i'$
    end if
  end for
end for

The hyperparameters used for both TD3 and LSTM-TD3 can be found in Table 3.3. We used densely-connected layers for both algorithms, with ReLU activation functions on the hidden layers and hyperbolic tangent for the output. The TD3 algorithm networks had just one hidden layer, and LSTM-TD3 had a separate input layer before the LSTM, then just one hidden layer after the concatenation of the LSTM output and current feature input.

Hyperparameter	TD3	LSTM-TD3
Discount factor - γ	0.99	
Batch size - N	50	
Replay buffer size	50 000	
Target update rate - τ	0.005	
Actor learning rate	10^{-3}	
Critic learning rate - τ	10^{-3}	
Exploration noise - σ	0.025	
Policy smoothing noise - $\tilde{\sigma}$	0.025	
Policy update delay - d	3	
Target noise clip boundary - c	0.5	
Actor gradient clip boundary	0.5	
Critic gradient clip boundary	0.5	
Optimizer	AMSGrad	
Time-steps per episode - T	1000	
Episodes trained - N_e	200	
LSTM length - l	None	10

Table 3.3: Hyperparameters used for training reinforcement learning algorithms. These parameters were selected after manually tuning for both algorithms. We found that with similar network and algorithmic structures (the only difference being an LSTM branch for LSTM-TD3), the two methods performed best with the same parameters.

We chose hyperparameters based on metrics of peak performance and training stability. We used a methodical approach when selecting values however the search was necessarily coarse due to the time intensive nature of training. A more fine parameter search was not practical for our setting as each one of these tests took three hours, and there are many hyperparameters to consider. It was similarly impractical to perform repeated tests for most hyperparameters. Given these limitations, choosing parameter values required some subjective interpretation of agents' performance, especially when several values appeared to perform similarly well.

The algorithms were trained episodically. Each episode began with a policy evaluation phase. During this phase, a fixed control policy was used to choose actions based on observations for a set number of time-steps. This data was then saved for later evaluation of training. After the evaluation phase, the data collection period begins, which consisted of the agent with the same fixed control policy interacting with the environment for a set number of time-steps, however Gaussian noise is injected into the actions chosen by the policy to encourage exploration. Once the set number of time-steps had been reached, all interactions and rewards from the data

collection period are inserted into a replay buffer. Then the agent pauses interaction to train its neural networks. The critic network is trained by recalling interactions from a replay buffer which contains previous interactions from about 50 episodes of training. The actor network is then updated to maximize the value of actions based on the critic network value estimates. This policy evaluation, data collection, and training process completed a single episode. We chose to stop training after 200 episodes because we found that the LSTM-TD3 algorithm approached optimal performance around episode 100 but wanted to show a longer horizon to demonstrate the stability advantages of the algorithm.

3.6 Conclusion (not published)

This work represents one of the first applications of model-free reinforcement in a highly stochastic fluid environment. Training these algorithms proved to be timely, and the learning performance was generally inconsistent. As a black-box end-to-end method that has been shown to outperform optimal control methods in simulation [31], the appeal of model-free reinforcement learning to solve difficult control problems in fluid mechanics is obvious.

In the preceding sections, we do not hide the difficulties involved with training a physical system in such a stochastic and extreme setting. Modeling the wake of an irregular and three-dimensional shape mounted dynamically for many time-steps would be extremely computationally expensive making this problem impractical for simulated environments, and unlike previous experimental applications of model-free reinforcement learning in fluid experiments [11, 40], our problem had no known solution or optimal control to match or beat. With no existing solution to demonstrate an effective control strategy, we had no way of setting performance expectations beyond that which a basic PID controller could provide. Even before the painstaking hyperparameter selection process described above, hundreds more hours were spent adjusting the experimental setting. For example, while we did observe the system learning improved control policies at various wind speeds, there were many considerations when selecting a flow velocity for the demonstrations above. Lower velocities result in more time to react and less turbulent noise in the pressure measurement, but also result in significantly lower aerodynamic forces and a lower signal-to-noise ratio in the reward signal. Higher velocities have less

noise in the load (and therefore reward), but give less time to react and noisier flow measurements. So while the agents were able to learn at a range of flow velocities, we needed to balance these effects to make a tractable but challenging learning environment.

While the notion of a system that learns on its own seems an attractive proposition, the lack of interpretability and the training instability inherent to black-box model-free reinforcement learning can frustrate efforts to gain a greater understanding of both successful and unsuccessful strategies. For example, in this work we found that LSTM cells improved performance, as did augmenting inertial observations with flow information. In theory, these two attributes could be together building predictive latent models of the fluid dynamics involved. This was, in fact, our hope at the outset of the project. However, in our physical system with high-dimensional nonlinear inputs we have no way of testing this hypothesis, and it is also possible that the LSTM cells and flow information improve the latent model of the dynamics in a purely reactive sense. In theory, should we be capable of producing controlled and discrete disturbances we could synchronize flow recordings with the control loop and see at what point the system reacts to the disturbance. However, we would need to be able to generate disturbances of different kinds with arbitrary trajectories to test the general accuracy and advance time of the predictive latent model on average. Despite practical difficulties associated with such a task, this would simply characterize the performance more formally and would still leave us to only speculate about how it achieves accurate predictive control in a highly stochastic fluid environment. Still, it is possible to gain unique insights into observations and control.

Additionally, as mentioned in the discussion section above model-based reinforcement learning methods and model-based control methods more generally can generally outperform model-free reinforcement learning. In fact, immediately following this work Lale et al. [21] used the same system with a novel model-based reinforcement learning algorithm and outperformed all algorithms tested as well as the Soft Actor Critic algorithm. These experiments featured the same experimental setting, but are not being included in this thesis for the sake of brevity.

In summary, outside of the results listed there are some larger lessons to be learned from this work. Firstly, black-box machine learning methods are not magic. While we did eventually find success in our experiments, even then the final performance betrayed our initial (admittedly unrealistic) expectations. Next, performance-

oriented experimental applications of model-free reinforcement learning in stochastic systems, such as this, can be used to benchmark performance, however it takes careful consideration to design a task and environment which is both demanding and feasible for learning. Additionally, performance is the primary metric from which we can learn through model-free reinforcement learning experiments. It can be a useful tool for some things, such as to infer the value of different types of information (as in Gunnarson et al. [15]), but in high dimensional physical systems observing learned behavior is the only available metric. Lastly, model-based learning techniques are often overlooked in fluid mechanics for the more mystifying and open-ended model-free methods, but are often more effective and data-efficient.

Bibliography

[1] P. W. Bearman. On vortex shedding from a circular cylinder in the critical Reynolds number régime. *Journal of Fluid Mechanics*, 37 (3):577–585, July 1969. ISSN 0022-1120, 1469-7645. doi: 10.1017/S0022112069000735. URL https://www.cambridge.org/core/product/identifier/S0022112069000735/type/journal_article.

[2] P. W. Bearman. Vortex shedding from oscillating bluff bodies. *Annual Review of Fluid Mechanics*, 16:195–222, 1984.

[3] Horst Bleckmann and Randy Zelick. Lateral line system of fish. *Integrative Zoology*, 4(1):13–25, March 2009. ISSN 17494877. doi: 10.1111/j.1749-4877.2008.00131.x. URL https://onlinelibrary.wiley.com/doi/10.1111/j.1749-4877.2008.00131.x.

[4] Horst Bleckmann, Joachim Mogdans, and S. Coombs. *Flow Sensing in Air and Water: Behavioral, Neural and Engineering Principles of Operation*. Springer, Heidelberg, 2014. ISBN 978-3-642-41445-9.

[5] F. Boettcher, Ch. Renner, H.-P. Waldl, and J. Peinke. On the Statistics of Wind Gusts. *Boundary-Layer Meteorology*, 108(1):163–173, July 2003.

[6] Steven L. Brunton and Bernd R. Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67 (5):050801, September 2015. ISSN 0003-6900, 2379-0407. doi: 10.1115/1.4031175. URL https://asmedigitalcollection.asme.org/appliedmechanicsreviews/article/doi/10.1115/1.4031175/369906/ClosedLoop-Turbulence-Control-Progress-and.

[7] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[8] C.E. Carcangiu, A. Pujana-Arrese, A. Mendizabal, I. Pineda, and J. Landaluze. Wind gust detection and load mitigation using artificial neural networks assisted control: Wind gust detection and control. *Wind Energy*, 17 (7):957–970, July 2014. ISSN 10954244. doi: 10.1002/we.1611. URL https://onlinelibrary.wiley.com/doi/10.1002/we.1611.

[9] Sheryl Coombs and John Montgomery. The role of flow and the lateral line in the multisensory guidance of orienting behaviors. pages 65–101, 2014. URL http://link.springer.com/10.1007/978-3-642-41446-6_3.

[10] John Elder and Sheryl Coombs. The influence of turbulence on the sensory basis of rheotaxis. *Journal of Comparative Physiology A*, 201(7):667–680, July 2015. ISSN 0340-7594, 1432-1351. doi: 10.1007/s00359-015-1014-7. URL http://link.springer.com/10.1007/s00359-015-1014-7.

[11] Dixia Fan, Liu Yang, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. Reinforcement learning for bluff body active flow control in experiments and simulations. *Proceedings of the National Academy of Sciences*, 117(42):26091–26098, October 2020. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2004939117. URL http://www.pnas.org/lookup/doi/10.1073/pnas.2004939117.

[12] Zhifang Fan, Jack Chen, Jun Zou, David Bullen, Chang Liu, and Fred Delcomyn. Design and fabrication of artificial lateral line flow sensors. *Journal of Micromechanics and Microengineering*, 12(5):655–661, September 2002. ISSN 09601317. doi: 10.1088/0960-1317/12/5/322. URL https://iopscience.iop.org/article/10.1088/0960-1317/12/5/322.

[13] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, Stockholm, Sweden, October 2018. PMLR. URL http://arxiv.org/abs/1802.09477.

[14] Nikola Gavrilovic, Murat Bronz, Jean-Marc Moschetta, and Emmanuel Benard. Bioinspired wind field estimation—part 1: Angle of attack measurements through surface pressure distribution. *International Journal of Micro Air Vehicles*, 10(3):273–284, September 2018. ISSN 1756-8293, 1756-8307. doi: 10.1177/1756829318794172. URL http://journals.sagepub.com/doi/10.1177/1756829318794172.

[15] Peter Gunnarson, Ioannis Mandralis, Guido Novati, Petros Koumoutsakos, and John O. Dabiri. Learning efficient navigation in vortical flow fields. *Nature Communications*, 12, 2021.

[16] Wei Hou, Darwin Darakananda, and Jeff D. Eldredge. Machine-learning-based detection of aerodynamic disturbances using surface pressure measurements. *AIAA Journal*, 57(12):5079–5093, December 2019. ISSN 0001-1452, 1533-385X. doi: 10.2514/1.J058486. URL https://arc.aiaa.org/doi/10.2514/1.J058486.

[17] Yonggang Jiang, Zhiqiang Ma, and Deyuan Zhang. Flow field perception based on the fish lateral line system. *Bioinspiration & Biomimetics*, 14(4):041001, May 2019. ISSN 1748-3190. doi: 10.1088/1748-3190/ab1a8d. URL https://iopscience.iop.org/article/10.1088/1748-3190/ab1a8d.

[18] Anya R. Jones. Gust encounters of rigid wings: Taming the parameter space. *Physical Review Fluids*, 5(11):110513, November 2020. ISSN 2469-990X. doi: 10.1103/PhysRevFluids.5.110513. URL https://link.aps.org/doi/10.1103/PhysRevFluids.5.110513.

[19] Stoyan Kanev and Tim van Engelen. Wind turbine extreme gust control. *Wind Energy*, 13(1):18–35, January 2010. ISSN 10954244, 10991824. doi: 10.1002/we.338. URL https://onlinelibrary.wiley.com/doi/10.1002/we.338.

[20] Michael Krieg, Kevin Nelson, and Kamran Mohseni. Distributed sensing for fluid disturbance compensation and motion control of intelligent robots. *Nature Machine Intelligence*, 1(5):216–224, May 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0044-1. URL http://www.nature.com/articles/s42256-019-0044-1.

[21] Sahin Lale, Peter Renn, Kamyar Azizzandenesheli, Babak Hassibi, Morteza Gharib, and Anima Anandkumar. Falcon: Fourier adaptive learning and control for disturbance rejection under extreme turbulence. *in preparation*, 2023.

[22] Aristotelis Lazaridis, Anestis Fachantidis, and Ioannis Vlahavas. Deep reinforcement learning: A state-of-the-art walkthrough. *Journal of Artificial Intelligence Research*, 69:1421–1471, December 2020. ISSN 1076-9757. doi: 10.1613/jair.1.12412. URL http://jair.org/index.php/jair/article/view/12412.

[23] Mathieu Le Provost, Wei Hou, and Jeff Eldredge. Deep learning and data assimilation approaches to sensor reduction in estimation of disturbed separated flows. In *AIAA Scitech 2020 Forum*, Orlando, FL, January 2020. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-595-1. doi: 10.2514/6.2020-0799. URL https://arc.aiaa.org/doi/10.2514/6.2020-0799.

[24] Jonathan N. Lefebvre and Anya R. Jones. Experimental investigation of airfoil performance in the wake of a circular cylinder. *AIAA Journal*, 57(7):2808–2818, July 2019. ISSN 0001-1452, 1533-385X. doi: 10.2514/1.J057468. URL https://arc.aiaa.org/doi/10.2514/1.J057468.

[25] Kara L. Marshall, Mohit Chadha, Laura A. deSouza, Susanne J. Sterbing-D'Angelo, Cynthia F. Moss, and Ellen A. Lumpkin. Somatosensory substrates of flight control in bats. *Cell Reports*, 11(6):851–858, May 2015. ISSN 22111247. doi: 10.1016/j.celrep.2015.04.001. URL https://linkinghub.elsevier.com/retrieve/pii/S2211124715003769.

[26] T. Tachim Medjo, R. Temam, and M. Ziane. Optimal and robust control of fluid flows: Some theoretical and computational aspects. *Applied Mechanics Reviews*, 61(1):010802, January 2008. ISSN 0003-6900, 2379-0407. doi: 10.1115/1.2830523. URL https://asmedigitalcollection.asme.org/appliedmechanicsreviews/article/doi/10.1115/1.2830523/443747/Optimal-and-Robust-Control-of-Fluid-Flows-Some.

[27] Lingheng Meng, Rob Gorbet, and Dana Kulic. Memory-based deep reinforcement learning for POMDPs. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626, Prague, Czech Republic, September 2021. IEEE. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9636140. URL https://ieeexplore.ieee.org/document/9636140/.

[28] A. Mohamed, R. Clothier, S. Watkins, R. Sabatini, and M. Abdulrahim. Fixed-wing MAV attitude stability in atmospheric turbulence, part 1: Suitability of conventional sensors. *Progress in Aerospace Sciences*, 70:69–82, October 2014. ISSN 03760421. doi: 10.1016/j.paerosci.2014.06.001. URL https://linkinghub.elsevier.com/retrieve/pii/S0376042114000542.

[29] A. Mohamed, M. Abdulrahim, S. Watkins, and R. Clothier. Development and flight testing of a turbulence mitigation system for micro air vehicles. *Journal of Field Robotics*, 33(5):639–660, August 2016. ISSN 15564959. doi: 10.1002/rob.21626. URL https://onlinelibrary.wiley.com/doi/10.1002/rob.21626.

[30] Siddharth Mysore, Bassel Mabsout, Kate Saenko, and Renato Mancuso. How to train your quadrotor: A framework for consistently smooth and responsive flight control via reinforcement learning. *ACM Transactions on Cyber-Physical Systems (TCPS)*, 5(4):1–24, 2021.

[31] Guido Novati, L. Mahadevan, and Petros Koumoutsakos. Controlled gliding and perching through deep-reinforcement learning. *Physical Review Fluids*, 2019.

[32] C.J. Pennycuick. Information systems for flying animals. In *Theoretical Ecology Series*, volume 5, pages 305–331. Elsevier, 2008. ISBN 978-0-12-374299-5. doi: 10.1016/S1875-306X(08)00011-7. URL https://linkinghub.elsevier.com/retrieve/pii/S1875306X08000117.

[33] Johannes E. Pohl, Rolf Radespiel, Benjamin Herrmann, Steven L. Brunton, and Richard Semaan. Gust mitigation through closed-loop control. I. Trailing-edge flap response. *Physical Review Fluids*, 7(2):024705, February 2022. ISSN 2469-990X. doi: 10.1103/PhysRevFluids.7.024705. URL https://link.aps.org/doi/10.1103/PhysRevFluids.7.024705.

[34] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 1 edition, August 2000. doi: 10.1017/CBO9780511840531.

[35] John Quindlen and Jack Langelaan. Flush air data sensing for soaring-capable uavs. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Grapevine (Dallas/Ft. Worth Region), Texas, January 2013. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-181-6. doi: 10.2514/6.2013-1153. URL https://arc.aiaa.org/doi/10.2514/6.2013-1153.

[36] Gautam Reddy, Jerome Wong-Ng, Antonio Celani, Terrence J. Sejnowski, and Massimo Vergassola. Glider soaring via reinforcement learning in the field. *Nature*, 562(7726):236–239, October 2018. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-018-0533-0. URL http://www.nature.com/articles/s41586-018-0533-0.

[37] Anatol Roshko. On the wake and drag of bluff bodies. *Journal of the Aeronautical Sciences*, 22(2):124–132, February 1955. ISSN 1936-9956. doi: 10.2514/8.3286. URL https://arc.aiaa.org/doi/10.2514/8.3286.

[38] Seyed G. Saddoughi and Srinivas V. Veeravalli. Local isotropy in turbulent boundary layers at high Reynolds number. *Journal of Fluid Mechanics*, 268:333–372, June 1994. ISSN 0022-1120, 1469-7645. doi: 10.1017/S0022112094001370. URL https://www.cambridge.org/core/product/identifier/S0022112094001370/type/journal_article.

[39] Aditya Saini and Ashok Gopalarathnam. Leading-edge flow sensing for aerodynamic parameter estimation. *AIAA Journal*, 56(12):4706–4718, December 2018. ISSN 0001-1452, 1533-385X. doi: 10.2514/1.J057327. URL https://arc.aiaa.org/doi/10.2514/1.J057327.

[40] Satoshi Shimomura, Satoshi Sekimoto, Akira Oyama, Kozo Fujii, and Hiroyuki Nishida. Closed-loop flow separation control using the deep q network over airfoil. *AIAA Journal*, 58, 2020.

[41] Md. Abu S. Shohag, Emily C. Hammel, David O. Olawale, and Okenwa I. Okoli. Damage mitigation techniques in wind turbine blades: A review. *Wind Engineering*, 41, 2017.

[42] Eduardo D Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer, New York, NY, 1998. ISBN 978-1-4612-0577-7 978-1-4612-6825-3. URL https://doi.org/10.1007/978-1-4612-0577-7. OCLC: 1165444109.

[43] Susanne Sterbing-D'Angelo, Mohit Chadha, Chen Chiu, Ben Falk, Wei Xian, Janna Barcelo, John M. Zook, and Cynthia F. Moss. Bat wing sensors support flight control. *Proceedings of the National Academy of Sciences*, 108(27): 11291–11296, 2011.

[44] Susanne J. Sterbing-D'Angelo, Mohit Chadha, Kara L. Marshall, and Cynthia F. Moss. Functional role of airflow-sensing hairs on the bat wing. *Journal of neurophysiology*, 117(2):705–712, 2017.

[45] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Mass., second edition, 2018. ISBN 978-0-262-03924-6.

[46] Paul D. Towers and Bryn Ll. Jones. Wind turbine gust estimation using remote sensing data. In *UKACC International Conference on Control*, 2014.

[47] Michael S. Triantafyllou, Gabriel D. Weymouth, and Jianmin Miao. Biomimetic survival hydrodynamics and flow sensing. *Annual Review of Fluid Mechanics*, 48(1):1–24, January 2016. ISSN 0066-4189, 1545-4479. doi: 10.1146/annurev-fluid-122414-034329. URL https://www.annualreviews.org/doi/10.1146/annurev-fluid-122414-034329.

[48] Pablo Valdivia y Alvarado, Vignesh Subramaniam, and Michael Triantafyllou. Design of a bio-inspired whisker sensor for underwater applications. In *2012 IEEE Sensors*, pages 1–4, Taipei, Taiwan, October 2012. IEEE. ISBN 978-1-4577-1767-3 978-1-4577-1766-6 978-1-4577-1765-9. URL http://ieeexplore.ieee.org/document/6411517/.

[49] Roberto Venturelli, Otar Akanyeti, Francesco Visentin, Jaas Ježov, Lily D Chambers, Gert Toming, Jennifer Brown, Maarja Kruusmaa, William M Megill, and Paolo Fiorini. Hydrodynamic pressure sensing with an artificial lateral line in steady and unsteady flows. *Bioinspiration & Biomimetics*, 7(3):036004, September 2012. ISSN 1748-3182, 1748-3190. doi: 10.1088/1748-3182/7/3/036004. URL https://iopscience.iop.org/article/10.1088/1748-3182/7/3/036004.

[50] Siddhartha Verma, Guido Novati, and Petros Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23):5849–5854, June 2018. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1800923115. URL https://pnas.org/doi/full/10.1073/pnas.1800923115.

[51] Pantelis R. Vlachas, Jaideep Pathak, Brian R. Hunt, Themistoklis P. Sapsis, Michelle Girvan, Edward Ott, and Petros Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *arXiv:1910.05266 [physics]*, February 2020. URL http://arxiv.org/abs/1910.05266. arXiv: 1910.05266.

[52] Wei Wang, Xingxing Zhang, Jianwei Zhao, and Guangming Xie. Sensing the neighboring robot by the artificial lateral line of a bio-inspired robotic fish. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1565–1570, Hamburg, Germany, September 2015. IEEE. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7353576. URL http://ieeexplore.ieee.org/document/7353576/.

[53] S. Watkins, J. Burry, A. Mohamed, M. Marino, S. Prudden, A. Fisher, N. Kloet, T. Jakobi, and R. Clothier. Ten questions concerning the use of drones in urban environments. *Building and Environment*, 167, 2020.

[54] C. H. K. Williamson. Vortex dynamics in the cylinder wake. *Annual Review of Fluid Mechanics*, 28(1):477–539, January 1996. ISSN 0066-4189, 1545-4479. doi: 10.1146/annurev.fl.28.010196.002401. URL https://www.annualreviews.org/doi/10.1146/annurev.fl.28.010196.002401.

Chapter 4

FORECASTING SUBCRITICAL CYLINDER WAKES WITH FOURIER NEURAL OPERATORS

Parts of this chapter are adapted from:

Peter I. Renn, Cong Wang, Sahin Lale, Zongyi Li, Anima Anandkumar and Morteza Gharib (2023). "Forecasting subcritical cylinder wakes with Fourier neural operators" (in preparation)
P.R. participated in the conception, recorded and analyzed all the data, trained the models, analyzed prediction performance, and prepared the manuscript.

4.1 Chapter abstract

We present an application of operator learning, specifically Fourier neural operators, for forecasting the temporal evolution of experimental cylinder wakes in the subcritical regime. Fourier neural operators are a recently developed machine learning method capable of approximating solution operators to systems of partial differential equations through data, producing full-field solutions in milliseconds. Here we train Fourier neural operators to predict the time evolution of experimental velocity fields from particle image velocimetry measurements of cylinder wakes at Reynolds numbers ranging from $Re = 240$ to $Re = 3060$. We train the Fourier neural operators at each Reynolds number over a normalized prediction horizon roughly equivalent to one-third of a Kármán vortex shedding cycle to compare performance across the regime. We find that FNOs are capable of accurately predicting the experimental velocity fields (L2 norm error < 0.1) throughout the range of Reynolds numbers tested. Given these results, we conclude that this method holds significant potential for real-time predictive flow control.

4.2 Introduction

Vortex shedding in a cylinder wake is among the most fundamental and well-studied problems in fluid mechanics. This phenomenon, otherwise known as a Kármán vortex street, is still relevant for a huge array of industries and applications today. The vortex shedding process has been studied in detail for decades with several comprehensive review papers on the subject, so we will keep our descriptions here brief [39]. Vortex shedding is first observed around $Re = 50$, at which point it

has been observed that an instability occurs in the previously steady recirculation regions of the wake. This instability results in a famously beautiful and well-ordered pattern of laminar alternating vortices convecting downstream and away from the cylinder. There has been some experimental variation observed in defining where the laminar vortex shedding regime ends but it is generally placed around Re = 190, at which point small-scale three-dimensional instabilities form and the transition to turbulence begins [40]. Following transition, the behavior of cylinder wakes from Re = 300 to Re = 200 000 was labelled the "irregular range" by Roshko [32] (otherwise known as the sub-critical regime). This regime is characterized by increasing three-dimensional effects, irregular velocity fluctuations, and the transition of the outer shear layer (Re = 1 200) [39]. Modeling vortical wakes in the "irregular range" is very possible through conventional numerical solvers, however this can be computationally expensive and time intensive [31].

As a canonical example that is both intuitive and visually striking, cylinder wakes have become a common subject of studies in other data-driven techniques for fluid mechanics as well. Notably, Bright et al. [1] used compressive sensing with a machine learning library of modes in the flow to reconstruct the full flow field. Using experimental particle image velocimetry (PIV) data, Deng et al. [4] trained generative adversarial networks to achieve super-resolution on turbulent cylinder wakes. Fukami et al. [7] used a laminar cylinder flow simulation as a preliminary demonstration of a super-resolution they later showed was capable of reconstructing turbulent flows. Fukami et al. [8] used a low Reynolds number to demonstrate a similar technique capable of super-resolution reconstruction in both space and time.

In recent years machine learning methods have been proposed as a solution for forecasting the future time evolution of fluid flows [2]. Srinivasan et al. [33] used recurrent neural networks to predict the temporal dynamics of a numerical model for a turbulent shear layer which matched the statistical characteristics of the flow field well, but struggled to predict the instantaneous flow fields as accurately. Proposing a hybrid deep neural network consisting of convolutional neural network (CNN) layers, long short-term memory (LSTM) recurrent neural network layers, and de-convolutional neural network layers, Han et al. [13] demonstrated a method for predicting the time evolution of a cylinder flow at various Reynolds numbers including some turbulent, but requires dozens of previous time-steps as input. Hasegawa et al. [14] trained a neural network containing a CNN auto-encoder (CNN-AE) and LSTM layers to predict cylinder wakes for various irregular shapes generated by

DNS, however only demonstrated the method at a single low Reynolds number with laminar shedding. Those authors later explored the impact of Reynolds number on their model within the laminar regime [15], and found that the model performed well when trained and tested at a Reynolds number but struggled to perform across the transition from a steady wake to laminar vortex shedding. Nakamura et al. [30] similarly used a combination of CNN-AE and LSTM layers to predict the time evolution of a three-dimensional turbulent channel flow, finding good agreement in statistical quantities but also struggling with instantaneous time-resolved flow field predictions. Moriomoto et al. [29] notably used neural networks to predict the evolution of the cylinder flow in time, among other tasks, when demonstrating techniques for improved generalization in machine learning for fluid flows. Combining a CNN-AE with sparse identification of nonlinear dynamics (SINDy), Fukami et al. [10] accurately predicted laminar cylinder flow data generated by direct numerical simulation (DNS), as well as a shear flow model. This method was capable of learning the latent dynamics of both cases, however the resulting models were sensitive to learning parameters. Mondal and Sarkar [28] demonstrated a method capable of predicting the temporal development of high-fidelity neural network based models while augmenting the training set with inexpensive low-fidelity data at Reynolds numbers as high as 800. Wu et al. [41] proposed a multi-resolution convolutional interaction network featuring variational auto-encoders (VAE) for forecasting cylinder wakes with Reynolds number varying periodically in the moderate subcritical regime, using eddy-viscosity turbulence models to generate data. Also using VAEs, Vlachas et al. [36] augmented existing multiscale frameworks with machine learning methods, demonstrating accurate predictions of simulated cylinder flows at a Reynolds number of 1000. We note that the vast majority of the aforementioned work focuses only on laminar vortex shedding, and those that feature turbulent training settings generally use simplified numerical models. Further, the methods applied are largely built on classical neural networks that can not provably learn solution operators to these systems.

Neural operators represent a new class of data-driven tools that may be well-suited for many applications within fluid dynamics. Neural operators share the basic structure of conventional neural networks, however are designed to approximate nonlinear operators, such as those between functional spaces. These methods have been shown capable of learning solution operators to systems of partial differential equations (PDEs) such as the constitutive equations underlying physical processes studied in fluid mechanics. Once trained, these methods can produce full-field

solutions in milliseconds, making them orders of magnitude faster than conventional solvers. Fourier Neural Operators (FNOs), introduced by Li et al. [22], are a specific type of neural operator that include integral operators via the Fourier domain within the network architecture. FNOs are discretization invariant, meaning that they can learn the underlying operator and be evaluated with data given arbitrary and varying discretizations. Li et al. [22] applied FNOs to computational solutions of the Navier-Stokes equation, achieving zero-shot super-resolution and successfully predicting the temporal development of vorticity fields. Other neural operator variations (e.g., DeepONet [26]) have been applied to computational flow data as well [5, 24].

While FNOs have proven themselves in several computational domains, the ability to forecast the evolution of experimental fluid flows faster than real-time could have many implications in open fields such as mitigation of atmospheric gusts and control of turbulent boundary layers. However previous studies using neural operators, and machine learning in general, on fluid mechanics have focused on problems given convenient computational parameters with known boundary solutions, and virtually none have used experimental measurements. Here we explore the potential for truly predictive real-time-capable machine learning models. We train and deploy FNOs on experimental data measured via two-dimensional particle image velocimetry (2D2C-PIV) to forecast the time evolution of cylinder wakes at a range of Reynolds numbers in the subcritical vortex shedding regime. We train an FNO to predict the evolution of the flow over a ten time-step period, which is equivalent to over one-third of a shedding cycle. We find that this operator learning approach accurately predicts instantaneous velocity fields over the full range of Reynolds numbers tested (error < 0.1) and is robust to experimental noise.

4.3 Experimental setup

Data acquisition

Our data was collected via experiments performed in a small free-surface water tunnel with a test section of 0.15 m (W) $\times$ 0.15 m (H) and a length of 0.61 m. We performed tests at flow speeds ranging from $U = 0.02$ m s^{-1} to $U = 0.40$ m s^{-1}. The corresponding Reynolds numbers range from about $Re = 240$ to $Re = 3100$, where we define Reynolds number as $Re = UD/\nu$ with D being the cylinder diameter and ν being the kinematic viscosity. The cylinder diameter is held constant at $D = 9.53$

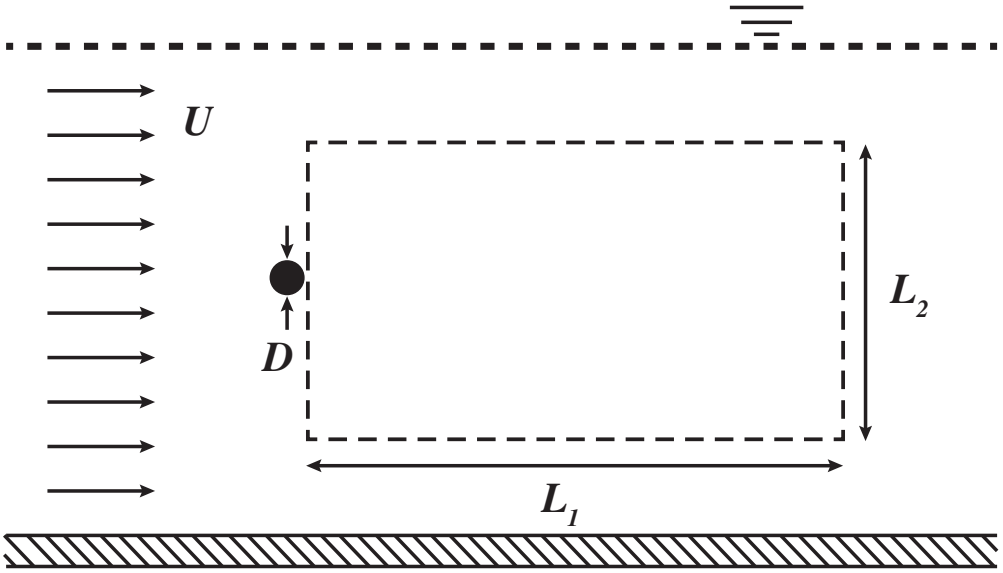

Figure 4.1: Schematic of the imaging region of interest, outlined by black dashed line, relative to the water tunnel and cylinder.

$\times 10^{-3}$ m and is fully submerged and fixed to the tunnel walls on both sides. The cylinder is made of cast acrylic, and is mounted approximately equidistant from the free surface and the tunnel floor to minimize the impact of either boundary on the vortex wake.

Figure 4.1 depicts the region of interest relative to the cylinder and the tunnel boundaries for 2DPIV recordings. Here $L_1 = 0.125$ m ($\approx 13D$) and $L_2 = 0.084$ m ($\approx 9D$). This region, located immediately behind the cylinder, is illuminated by a laser sheet at a streamwise cross section near the center of the tunnel. A high speed camera (IDT XSM-3520 set to 2144×1440 resolution) is used to record the flow in this region. The frame rate of the camera is adjusted based on the mean flow speed to maintain a near-constant non-dimensional time between frames regardless of tunnel speed. The non-dimensional time, otherwise known as the formation time, for a cylinder flow is given by Jeon and Gharib [16] as:

$$t^* = \frac{Ut}{D}. \tag{4.1}$$

Here t^* is the non-dimensional time and t is the dimensional time. In our 2D2C-PIV analysis, an interrogation window size of 32×32 pixels with 50% overlap was used, giving spatial resolution of $< 0.1D$ for our resulting velocity fields. With this data, we were able to construct training sets for each flow configuration tested

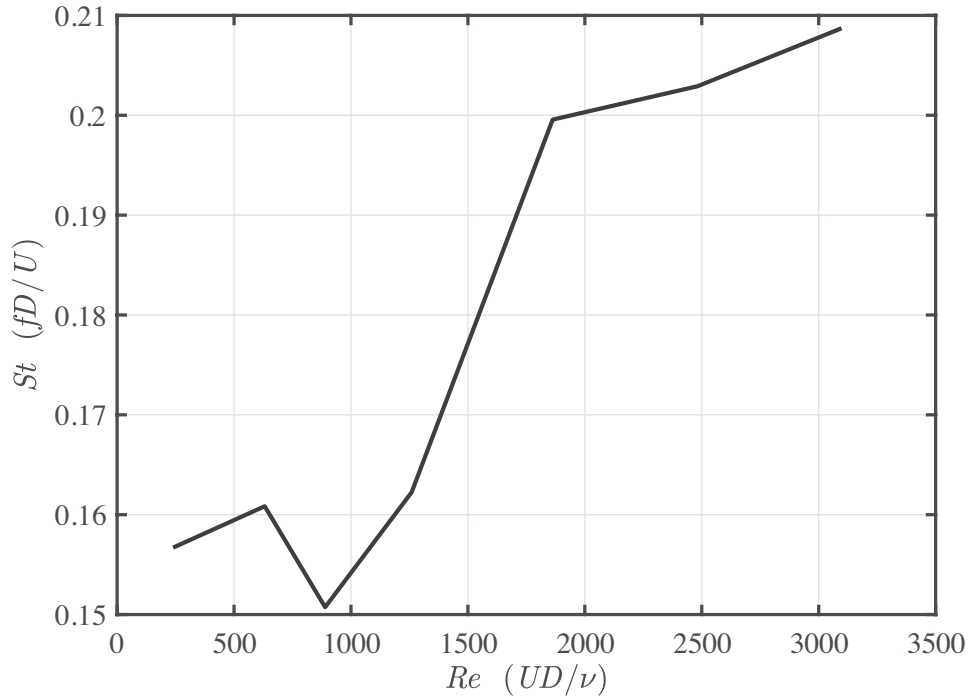

Figure 4.2: Strouhal number plotted against Reynolds number. Note that this relationship diverges from the established values.

that contained 3488 unique time sequences of data with 864 sequences reserved for testing (total of 4352 sequences). To prevent any mixing of training and testing data, we limited the overlap between the sequences so that only the first/last time-step could be shared. That is, the final time-step of each sequence could be used as the first time-step of a different sequence but all intermediate time-steps were not shared. This ensured that no transition between consecutive frames was in more than one time sequence, preserving the uniqueness of each training point and ensuring that no contamination could occur between the training and testing data.

We note that the Strouhal number ($St = fD/U$ where f is shedding frequency) is known to remain near a constant value of $St = 0.21$ across the range of Reynolds numbers tested. This means that scaling the time-steps based on formation time should result in an approximate scaling based on Strouhal period as well. However, from calculating the shedding frequency in the PIV measurements, we observed an irregular distribution of Strouhal numbers that diverge from established results, as seen in figure 4.2

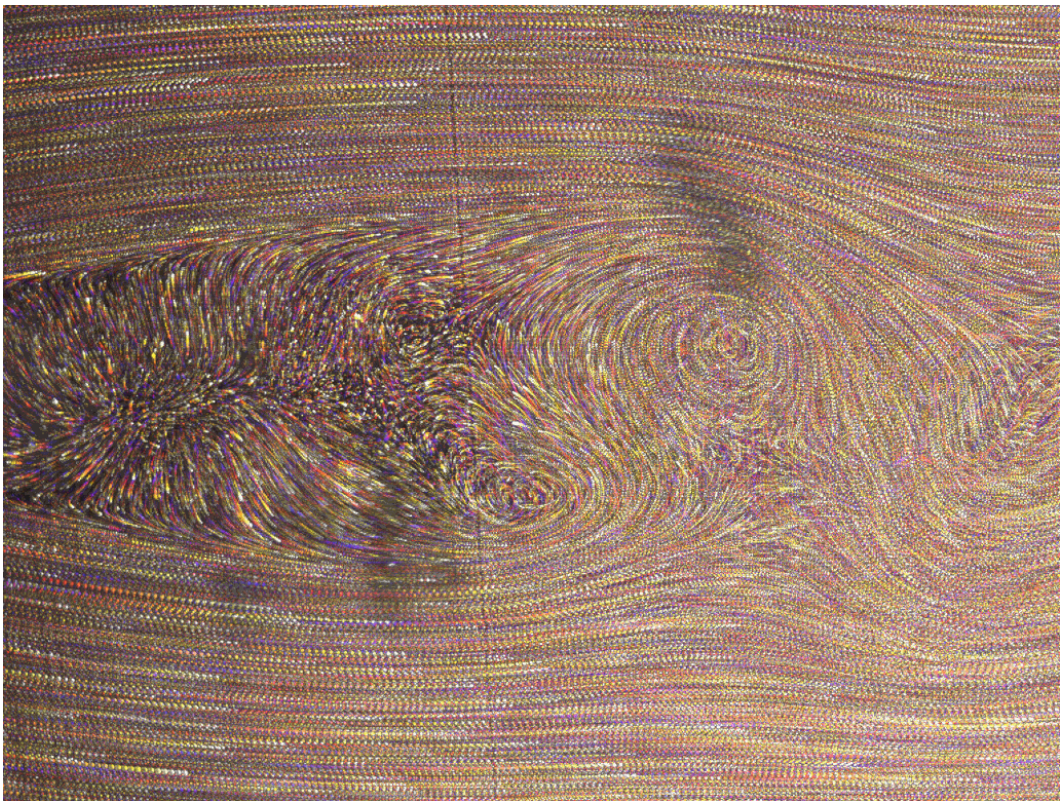

Figure 4.3: Particle trace overlaying 60 images showing the recirculation region of the vortex wake. We can note the three-dimensional effects in the recirculation region due to the high density of pathlines diverging from specific regions.

We note that deviations in shedding frequency are likely the result of vortex shedding occurring at an oblique angle [39]. We can further confirm this by examining a particle trace by overlaying subsequent image pairs, as shown in figure 4.3.

Here we see significant three-dimensional motions in the recirculation region, despite a Reynolds number of only $Re = 630$. This is also indicative of oblique shedding. Since our cylinder is fixed rigidly to the tunnel and not free to vibrate or oscillate, oblique shedding is likely the result of a small difference in boundary conditions at the spanwise ends of cylinder [12]. We note that there is a minor asymmetry in cylinder geometries at the two walls, where a screw was used to adjust the exact cylinder length on one side. It is also possible that the difference in boundary conditions is the result of some spanwise deviation in the incoming velocity due to tunnel geometry (e.g., a blockage or crooked flow manipulator) or pump malfunction (e.g., one pump is less efficient than another). Regardless of the exact source of the oblique shedding angle, this effect should not impact the validity of this data for training and predicting cylinder wake velocity fields. In fact,

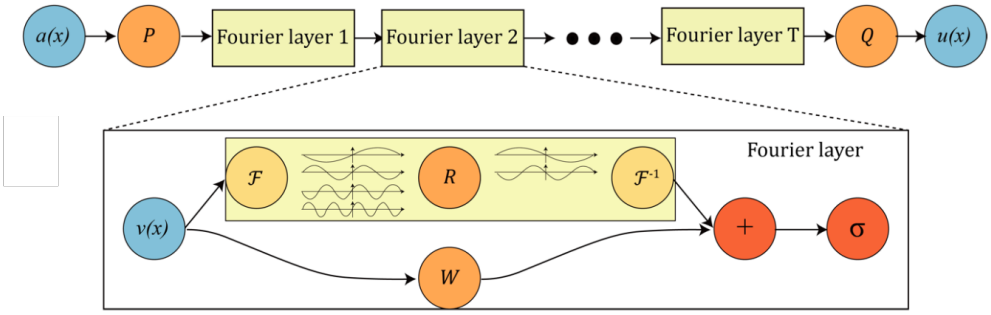

Figure 4.4: Diagram showing composition of Fourier layer in FNO. Taken from Li et al. [22].

the additional three-dimensional effect may provide more challenging dynamics. We simply note that the deviation in shedding frequency was observed, and that assumptions regarding equivalency between Strouhal period and formation time are not valid across the full range of Reynolds numbers tested.

Fourier neural operators

Neural operators are distinct from standard neural networks in their unique ability to learn operator mappings between infinite-dimensional function spaces. Approximating mappings between function spaces allows for directly learning solution operators to families of PDEs from data alone, making these methods particularly well-suited for problems in fluid mechanics. In contrast, classical neural networks map only between finite-dimensional spaces and are not universal approximators of operators. Introduced by Li et al. [22], Fourier neural operators (FNOs) are powerful form of neural operator that are discretization invariant and guarantee universal approximation for continuous operators [18, 19].

The architecture of an FNO, shown in figure 4.4, is made up of unique Fourier layers. These Fourier layers consist of paths: non-linear activation functions as found in classical neural networks, and a linear transform of the input signal performed in Fourier space. The non-linear activation function path serves to approximate local non-linearities, and the transform in Fourier space serves as a global integral operator. The paths are then combined and passed forward. The weights in each layer are trained through back-propagation to minimize the selected loss function,

similar to training classical neural networks. Details on the mathematical principles underlying FNOs and neural operators more generally can be found in Kovachki et al. [19]. All models were trained on NVIDIA RTX A5000 GPUs, and the FNO hyperparameters used for this work can be found in table 4.1

Hyperparameter	Value
Initial learning rate	10^{-3}
Epochs	200
Batch size	16
Train set size	3488
Test set size	864
Optimizer	Adam
Learning rate scheduler	StepLR
LR scheduler step size	5
LR scheduler decay rate	0.90
Fourier layer width	80
Fourier layer modes	24

Table 4.1: Hyperparameters used for training FNOs in all following results, except where noted otherwise

4.4 Results

Prediction approach

In this study, we use FNOs to forecast the time evolution of both x and y components of velocity fields (u, v) in the wake of a cylinder at various Reynolds numbers. In this context, FNOs can learn to predict future state of the velocity fields based on current observations. While not necessary for learning with FNOs, we preprocessed the data by subtracting out the mean field. This approach, borrowing from stability theory [21], essentially decomposed the velocity into steady and unsteady parts:

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u_0}(\boldsymbol{x}) + \boldsymbol{u'}(\boldsymbol{x}, t). \tag{4.2}$$

We then can substitute this into the Navier-Stokes equations. Assuming the steady part alone satisfies the time-independent equations and omitting $\boldsymbol{u'}(\boldsymbol{x}, t)$ terms of order greater than one, we are left with

$$\frac{\partial \boldsymbol{u'}}{\partial t} + (\boldsymbol{u_0} \cdot \nabla)\boldsymbol{u'} + (\boldsymbol{u'} \cdot)\boldsymbol{u_0} = \frac{\nabla p'}{\rho} + \nu \Delta \boldsymbol{u'}, \quad \nabla \cdot \boldsymbol{u'} = 0 \tag{4.3}$$

where p' is the unsteady pressure component (i.e., $p(\boldsymbol{x}, t) = p_0(\boldsymbol{x}) + p'(\boldsymbol{x}, t)$). Therefore, we are left with a set of homogeneous linear differential equations [21], which are perhaps more readily learned by the FNO.

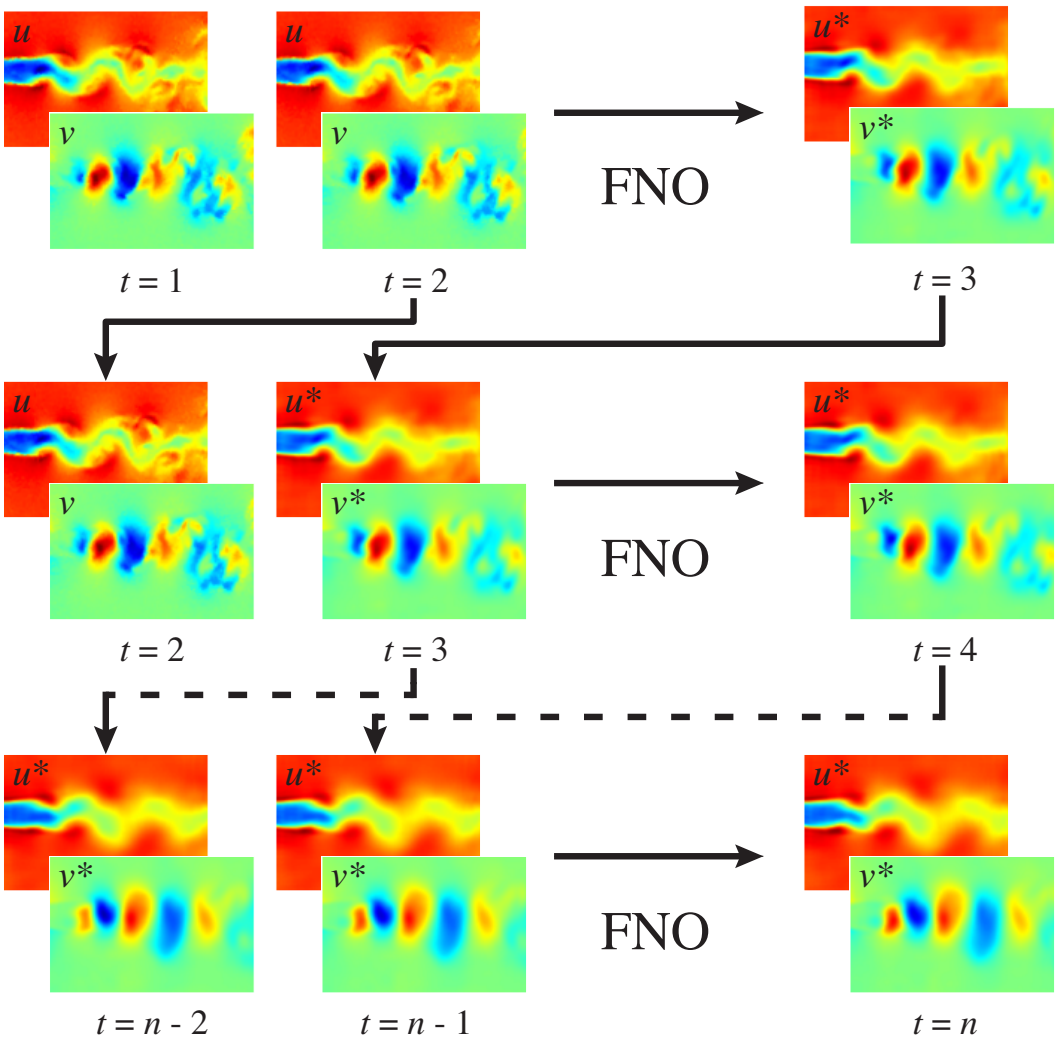

Figure 4.5: Recursive application of Fourier Neural Operators (FNOs).

In addition to simplifying the underlying differential equations, subtracting out the time-averaged velocity ensures that unsteady fluctuations are not dominated out the large free-stream velocity bias. This is common to many modal analysis techniques used in studying fluid flows (e.g., proper orthogonal decomposition) [34], and likely has similar benefits in frequency-based learning methods like FNOs. A comparison of performance between FNOs provided the full velocity components ($\boldsymbol{u}(\boldsymbol{x},t)$) and fluctuation velocity components ($\boldsymbol{u'}(\boldsymbol{x},t)$) can be found in the appendix. Since we train the FNOs to predict the fluctuations alone, we can then reconstruct the full flow field by summing the mean and fluctuating velocity components as defined in equation 4.2.

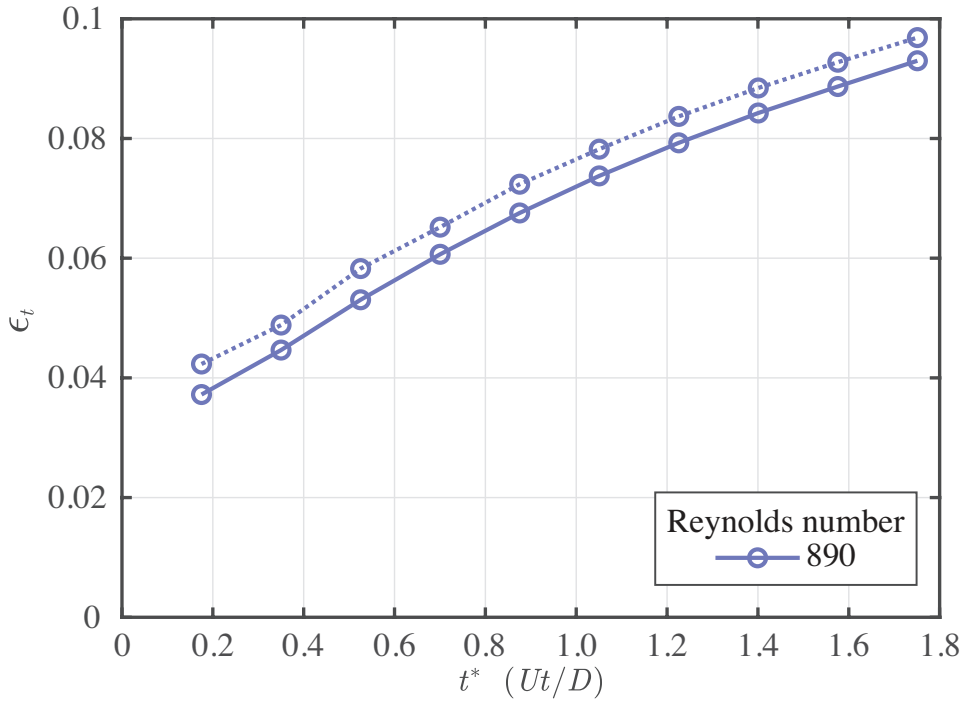

Figure 4.6: Relationship between prediction error and time at a Reynolds number of 890 for two FNOs with identical hyperparameters. The dashed green line directly learns the full velocity ($\boldsymbol{u}(\boldsymbol{x},t)$), and the solid purple line learns the deviations alone ($\boldsymbol{u'}(\boldsymbol{x},t)$) and is then reconstructed.

To demonstrate the benefits of this preprocessing, we can plot the error at each time-step for the two methods, as shown in figure 4.6. From this we see that predicting only the fluctuating velocity component results in lower error at each time-step. This method is used in all results, except when explicitly stated otherwise.

While FNOs can make predictions from only one time-step as input, we chose to use two time-steps to reduce the effects of experimental errors. From this two time-step input, the model predicts the state of the flow field at the following time-step. During training, we set a desired number of time-steps to forecast. The model is then applied recursively, as shown in figure 4.5, to reach the desired number of steps. The loss is calculated at each recursive step and summed to find the overall loss in the prediction. The FNO tries to minimize this loss through the back-propagation algorithm, similar to a classic neural network. Because they are applied recursively, the FNO models can also be used to predict time-steps beyond the trained horizon.

In analyzing predictions, we define the error of the flow field, ϵ, at a given time-step t as calculated as the L2 error norm:

$$\epsilon(t) = \frac{\|\boldsymbol{q}_t^* - \boldsymbol{q}_t\|_2}{\|\boldsymbol{q}_t\|_2} \tag{4.4}$$

where $\boldsymbol{q}_t$ is a $N \times 1$ vector containing the two-component velocity field, such that

$$\boldsymbol{q}_t = \begin{bmatrix} u(x_1, y_1, t) \\ u(x_1, y_2, t) \\ \cdots \\ u(x_1, y_m, t) \\ u(x_2, y_1, t) \\ \cdots \\ u(x_n, y_m, t) \\ v(x_1, y_1, t) \\ \cdots \\ v(x_n, y_m, t) \end{bmatrix} \tag{4.5}$$

and $\boldsymbol{q}_t^*$ is the FNO estimate of $\boldsymbol{q}_t$. The quantities $u(x_i, y_j, t)$ and $v(x_i, y_j, t)$ in equation 4.5 are the full, reconstructed velocity components at (x_i, y_j) and time t.

We non-dimensionalized the time-step across Reynolds numbers using the same formulation shown in equation 4.1. This time-step was set to be exactly five-times the interframe time for each Reynolds number ($\Delta t \approx 0.17$). This relatively large time-step was chosen because the measured difference in consecutive velocity fields was on the order which we would expect experimental noise to occur. Using a larger time-step increased the difference while maintaining the same noise level.

Prediction accuracy and Reynolds numbers

We first apply FNOs to cylinder flows at various Reynolds numbers to gauge the performance of the learned solution operators under varying conditions. To this end, we recorded equivalent data sets at $Re = \{240, 630, 890, 1260, 1860, 2480, 3060\}$ and trained separate FNOs for each. The velocity fields is predicted in increments of $t^* \approx 0.17$, and the FNO is trained to forecast the evolution of the flow over ten time-steps.

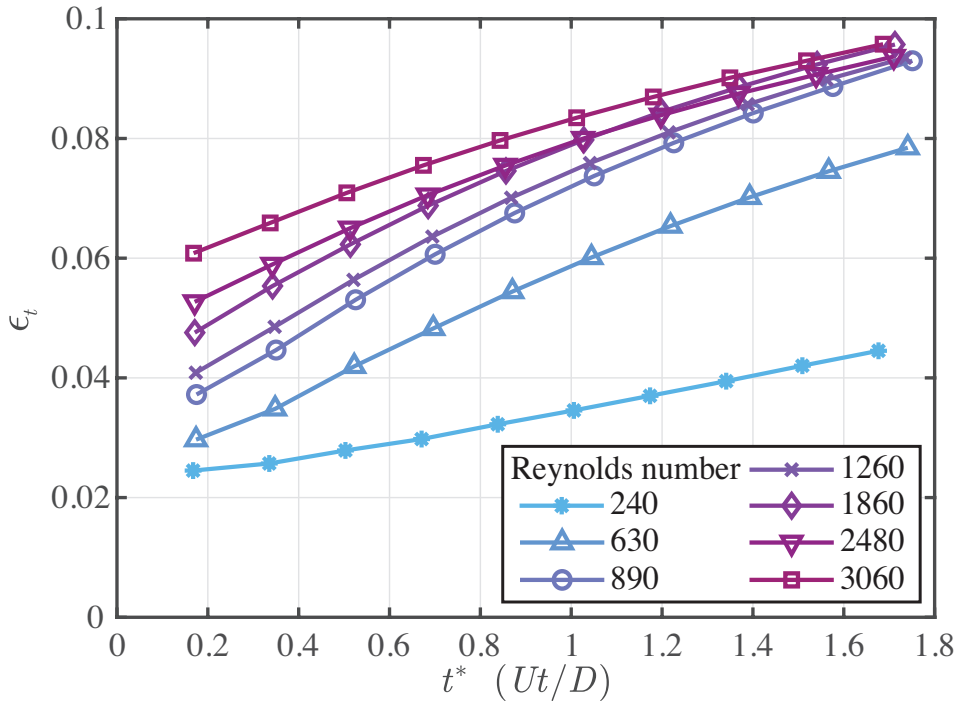

Figure 4.7: The error as a function of prediction time plotted at various Reynolds numbers.

The Reynolds numbers tested range from the very early transition where three-dimensional effects begin to be observed well into the shear-layer transition regime [39]. As the Reynolds number increases, the flow becomes noticeably more irregular which presents challenges for learning the solution operator. As the flow becomes more turbulent the wake becomes three dimensional and chaotic [17]. Fluctuations in the shape and motion of the shed vortices become irregular, which generally makes forecasting the time evolution of the velocity fields difficult. Distinct but intrinsically coupled to the progressively complex physics associated with increasing Reynolds number, growing three-dimensional velocity components move particles in and out of the stream-wise imaging plane which negatively impacts the performance of PIV algorithms. In effect, as the incident velocity increases the learning problem becomes more challenging and the data training becomes more noisy.

From figure 4.7 we see that the prediction error does increase with Reynolds number, but remains under 0.10 across the full prediction horizon, suggesting the FNO is accurately predicting the instantaneous velocity fields even in the more difficult high Reynolds number cases. Figure 4.7 also shows that the trajectory of the

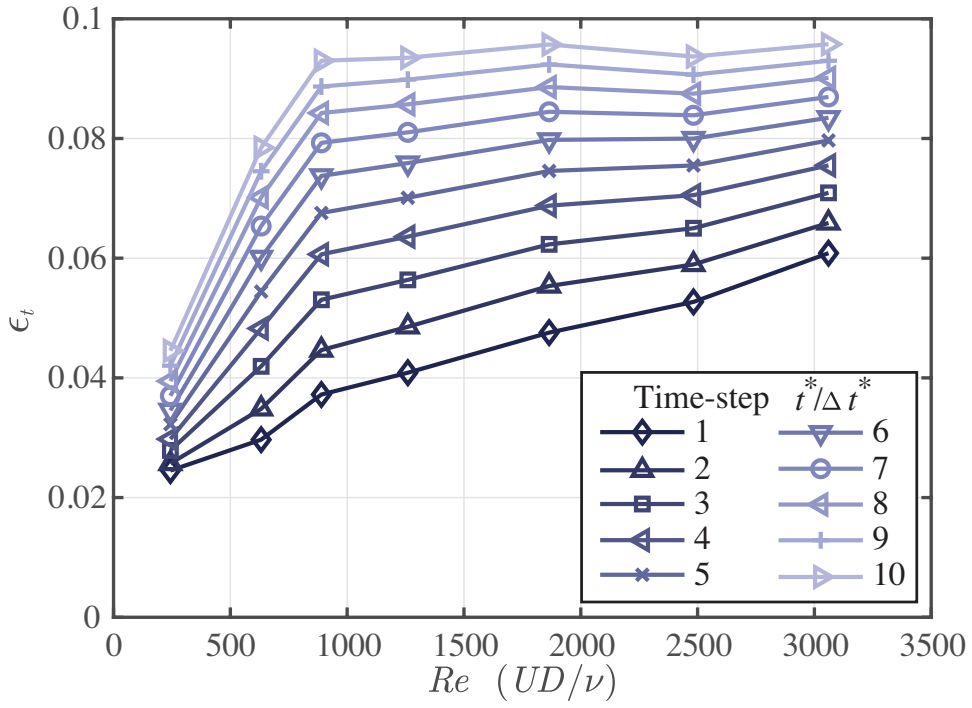

Figure 4.8: The error a a function of Reynolds number plotted at various time-steps.

prediction error does not scale linearly with Reynolds number. Further, we see that the relationship changes depending on the prediction step, as the distribution of the error values at the initial time-step have a very different distribution from those at the final time-step.

Plotting the error at each prediction step as a function of Reynolds number (figure 4.8), we see that the error in the first time-step does have a nearly linear relationship with Reynolds number. However, this does not hold and the error begins to grow non-linearly with Reynolds number. The error value of the final time-step ($t^*/\Delta t^* = 10$) increases sharply in the range from $Re = 240$ to 890, but then remains relatively consistent around $\epsilon \approx 9.3 \times 10^{-2}$.

We can also examine individual true and predicted velocity fields to gain insight into the performance of the FNOs. Figures 4.9 and 4.10 show these values as well as a normalized absolute difference between the measurement and the prediction at the first and last step for the $Re = 240$ and $Re = 3060$, respectively. The first step prediction for the $Re = 240$ case (left-columns, figure 4.9) shows some minor deviations in the outer shear layers in the x-component as well as small magnitude high-frequency error that seems to concentrate near the edge of the wake

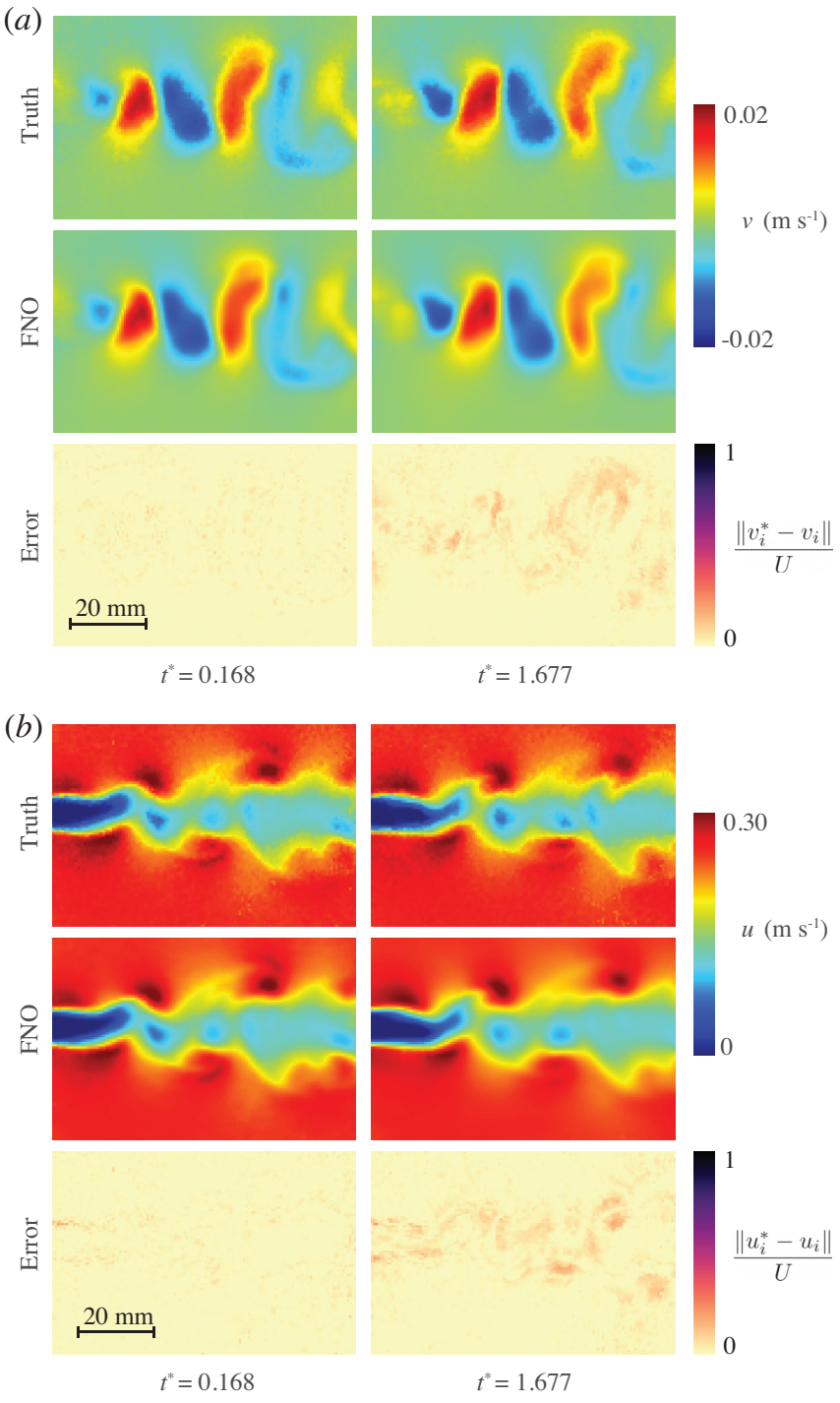

Figure 4.9: Instantaneous (*a*) x and (*b*) y velocity component measurements (top row), predictions (middle row), and errors (bottom row) at the first (first column) and last (second column) prediction for $Re = 240$.

in both velocity measurements. After 10 predictions (right-columns, figure 4.9) more differences between the prediction and experiment have developed on the length scale of wake vortices themselves. We note that the x-component error does seem slightly more interior to the wake in comparison with the y-component. As would be expected, these regions mostly appear near the edges of wake structures in both cases.

Looking at the measured velocity fields in the $Re = 3060$ case (top rows, figure 4.10), we immediately see that these higher Reynolds number measurements appear less smooth than the low Reynolds number measurements. However, the predictions themselves remain smooth. Similar to the low Reynolds number case, the difference between measurement and prediction after the first case primarily consists of high-frequency fluctuations, especially in the x-component velocity field. We see that the y-component velocity field also has high-frequency error, but these are more focused around the center of the near-wake. For the high Reynolds number case, there does appear to be some coherence in the distribution of the differences relative to the vortices themselves after just one time-step. We also note that final prediction and the first prediction for the x-component high Reynolds number case have roughly equivalent high-frequency components in the free-stream portions, indicating that this may be experimental error. The larger regions of error, however, clearly accumulate in the wake where we expect the vortices to be, similar to the lower Reynolds number case.

The error regions with length scales on the order of the primary Kármán vortices are likely attributable to the learned FNO model itself. We see that in both high and low Reynolds number cases, these errors accumulate systematically with increased prediction time, which is indicative of iterating imperfect learned dynamics.

The source of the high-frequency differences we observe is less clear. Since we can expect to see increasing three-dimensional turbulent velocity fluctuations as the Reynolds number increases in the range shown, it is possible that some of this signal can be attributed to error introduced by out-of-plane motion of seeding particles. While the source of these out-of-plane particles is physical, they are not correctly characterized by the two-component PIV algorithm applied here and therefore can appear as noise in the resulting measurement. This is most likely the cause of the small deviations in the free-stream regions of the flow, especially for the high Reynolds number case.

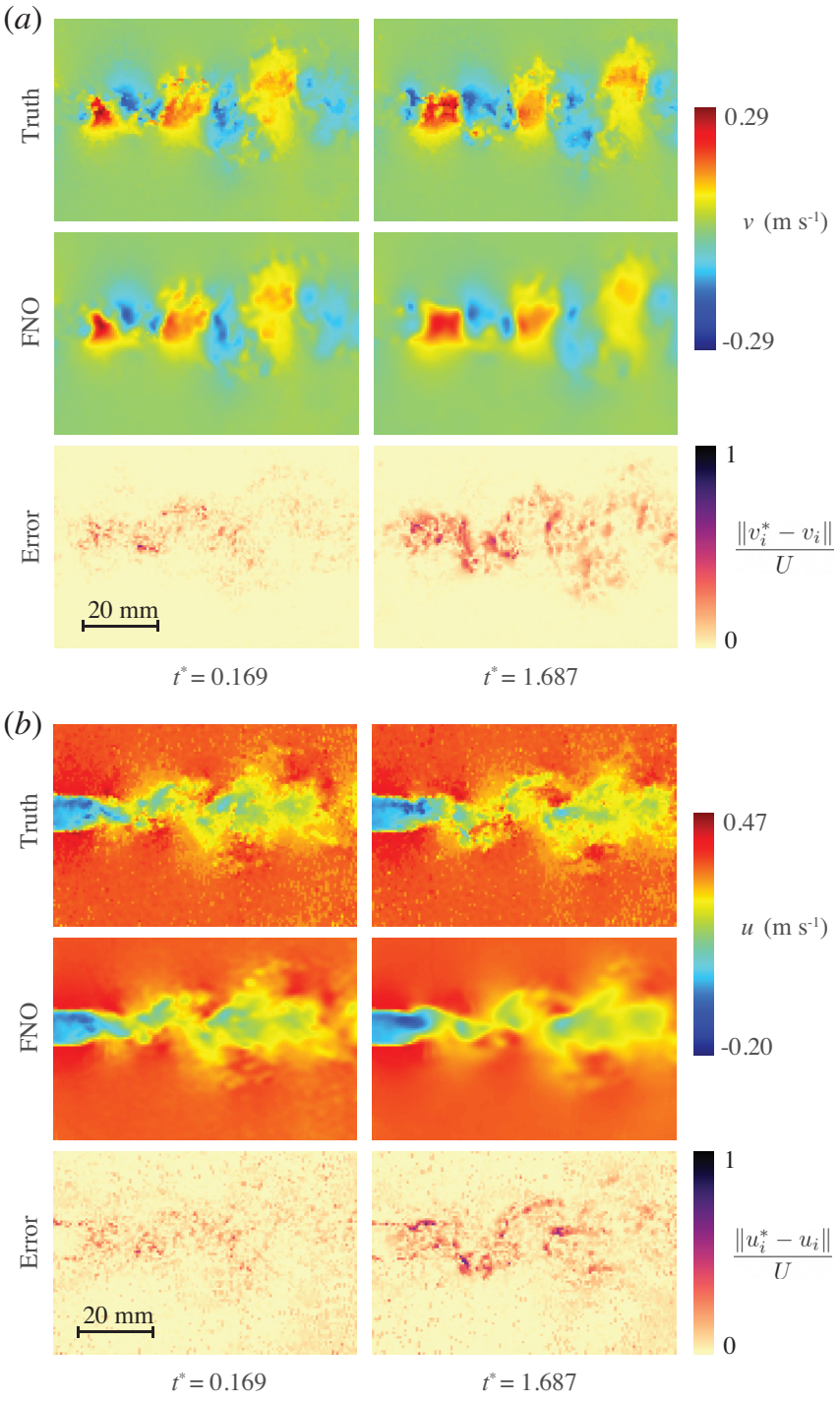

Figure 4.10: Instantaneous (*a*) x and (*b*) y velocity component measurements (top row), predictions (middle row), and errors (bottom row) at the first (first column) and last (second column) prediction for $Re = 3060$.

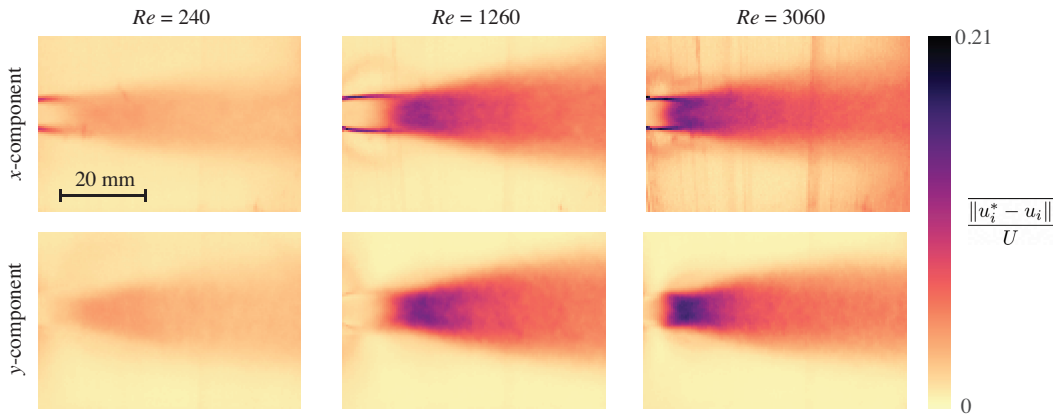

Figure 4.11: Mean absolute error field of x and y components normalized by free-stream velocity at different Reynolds numbers.

However, it is also possible that some of the high-frequency deviations are directly due to FNO modeling error. We see in both cases that the predicted velocity field becomes smoother between the first and final prediction, especially around the edges of the wake and between low-velocity regions associated with vortex cores. These are regions where we would expect to see the most turbulent fluctuations and small scale flow features, especially in the high Reynolds number case where the shear layer has transitioned before the vortex formation process begins [39]. The increasing smoothness observed in the predicted solutions may therefore indicate that the FNO struggles to model the complicated physics in these turbulent regions, and therefore learns to predict a more average solution absent of small, sharp flow features. We also note that the limited spatial resolution of the PIV measurements ($\approx 0.1D$) is such that not all small structures (e.g., shear layer vortices) will be well resolved in our measurement which may in-part make learning these fine-scale dynamics intractable [37, 39]. As mentioned earlier, we know that these regions also tend to accumulate pockets of error with larger length scales, and may be due in part to the smoothing effect learned to minimize the error from small, high-frequency structures. The more turbulent regions of the flow, however, would likely also have more out-of-plane velocity components which could then contribute to the aforementioned measurement noise. It is likely a combination of these factors that contributes to the high-frequency component of the error. Despite these minor deviations, we note that FNO does accurately predict the instantaneous shape and motion of the dominant coherent structures in the turbulent wake.

Figure 4.11 shows both components of the absolute time-resolved testing error averaged over the final prediction time-step. In taking the average of the error across many different predictions, we can establish which regions of the wake tend to accumulate the most error. We see that the shear layers from either side of the near-wake are a major source of x-component velocity error across all three cases. While there is error in the shear layer in all three cases, we see it increase at the higher Reynolds numbers where the shear layer transition occurs earlier. Both velocity components for the $Re = 1260$ and 3060 cases also have strong error signals near the end of the recirculation region where the primary Kármán vortices are formed. This central peak error region also expands and diffuses in the downstream wake in both velocity components, which can likely be attributed to a combination of the vortices themselves diffusing and an increased variance in shape/trajectory causing a wider occupancy region for the coherent structures. For the high Reynolds number case, both components also have regions of elevated error outside the shear layer near the leading boundary of the frame, which can likely be attributed to unknown inlet boundary conditions.

4.5 Generalization

Next we will see how FNOs trained with experimental flow measurements can generalize across different conditions. Generalization in machine learning refers to the ability of a learning machine to produce accurate approximations when presented with test data from a distribution separate from that used for training. This is a major challenge for many machine learning methods as models are trained to minimize the loss for the specific distribution of the training data alone. In order to generalize well, a model must learn some underlying dynamics from the training data which are both valid and significant for out-of-distribution cases as well.

In the context of predicting the time evolution of fluid flows, we test the ability to generalize by training an FNO to learn the dynamics of one flow configuration, and then apply that same FNO to a new and unseen case. Given that FNOs can learn operators, we would expect that the dynamics learned in one flow configurations would include some of the general underlying dynamics for all cases (i.e., the Navier-Stokes equations). However that is no guarantee of generalization.

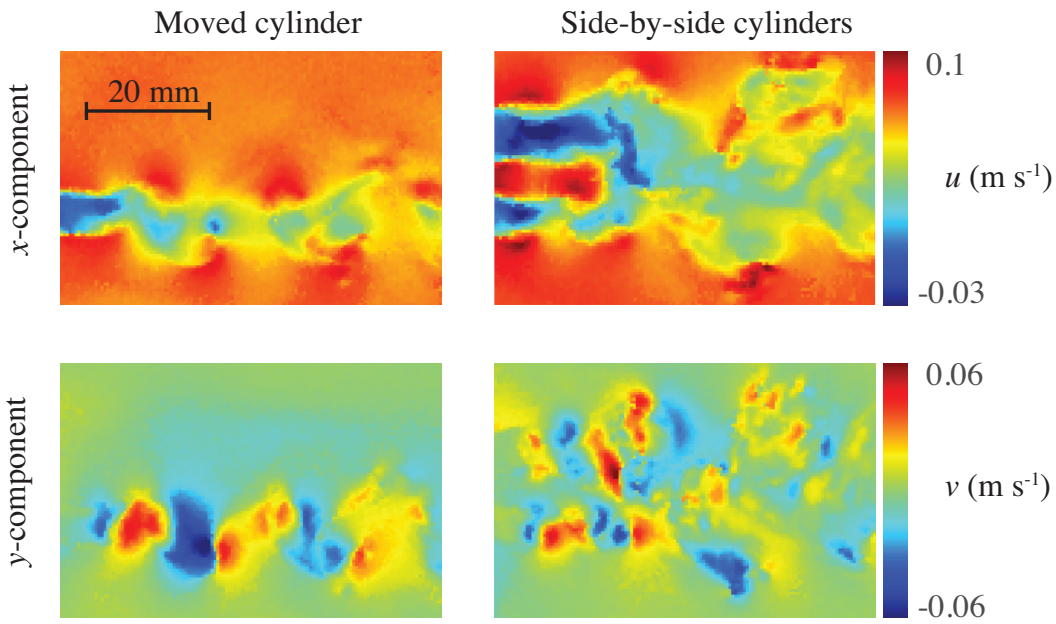

Figure 4.12: Instantaneous velocity field measurements for the two configurations used in testing generalization.

Here, we take the FNO trained at $Re = 630$ in the cylinder configuration shown in figure 4.1 and apply that model to two unseen configurations: (1) flow past the same cylinder moved to the bottom of the frame and (2) flow past two side-by-side cylinders. The Reynolds number in these cases was matched to the training data. Figure 4.12 shows instantaneous velocity fields for the two configurations used to test generalization.

The configuration with the cylinder moved in the PIV frame is the less challenging of the two cases, as the fluid dynamics should be identical to the original case other than the shift in position. However, the FNO has no way of knowing that the system has undergone a simple translation and has no knowledge of the location of the cylinder or the flow boundary conditions making this a larger challenge than it may seem. The side-by-side cylinder combination is an even more challenging configuration, since the shed vortices can interact downstream leading to complicated and previously unseen vortex interactions.

With the FNO trained on data from the original cylinder configuration, we made predictions across the same time horizon used for training (10 time-steps, where one time-step is $\Delta t^* = 0.174$). Just as in the Reynolds number testing cases, we can then calculate the L2 norm error at each time-step using equation 4.4.

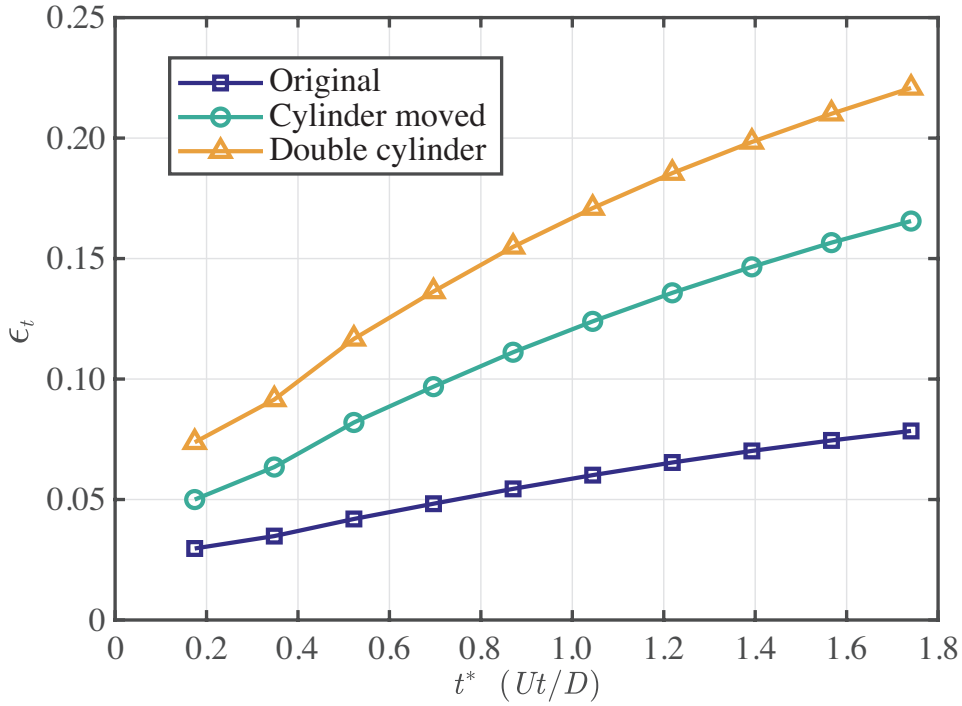

Figure 4.13: Error versus time plot showing the performance of the FNO trained on a nearly centered cylinder when applied to different cases.

We plot this error against prediction step in figure 4.13. While we see that the error for both out-of-distribution test cases is higher than the error on the original test set at all time-steps, the error remains under 0.25 across the full prediction length. As expected, the FNO does generalize to the moved cylinder case better than it does for the double cylinder. Interestingly, the error for the double cylinder case grows only slightly faster than that of the moved cylinder, indicating that the propagating dynamics of the learned model may be performing similarly.

Moved cylinder generalization

We can examine the instantaneous velocity fields, predictions, and errors to gain insight into the generalization performance of the FNO in this context. Starting with the moved cylinder case (figure 4.14, we see that the x-component velocity immediately accumulates some error at the edge of the recirculation region, however the overall shape of the predicted velocity field matches the actual value fairly well. The y component suffers from some error near the vortex formation region in this case, but similarly matches the general structure of the measured flow. As time progresses, we see that some deviations begin to arise. In the fifth time-step of the

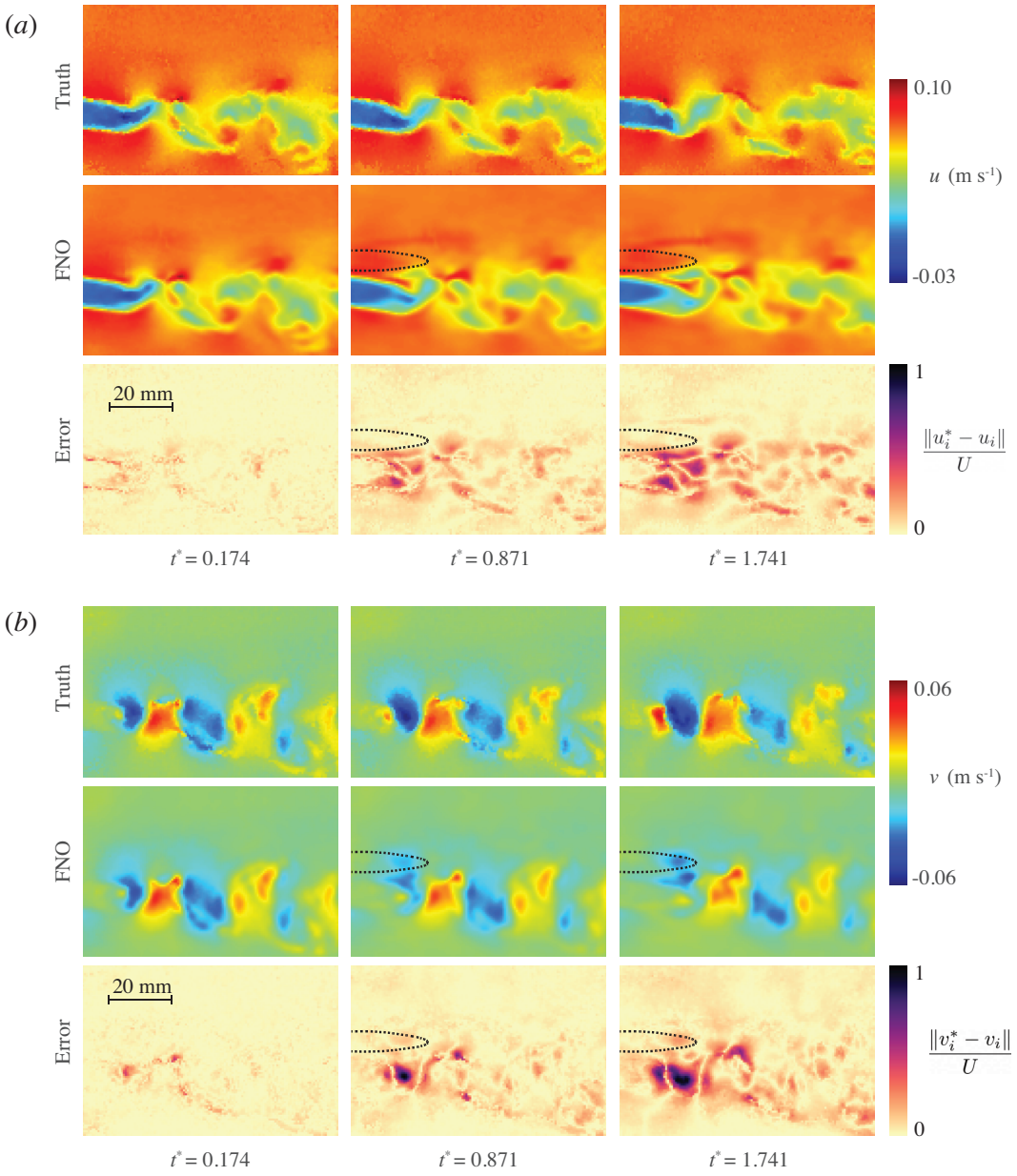

Figure 4.14: Instantaneous (*a*) x-component and (*b*) y-component velocity field measurements, predictions, and error for the moved cylinder generalization case at three different timestamps. The dotted black line included in some of the fields depicts the zero-velocity contour line (i.e., the approximate edge of the recirculation region) from time-averaging the original training data.

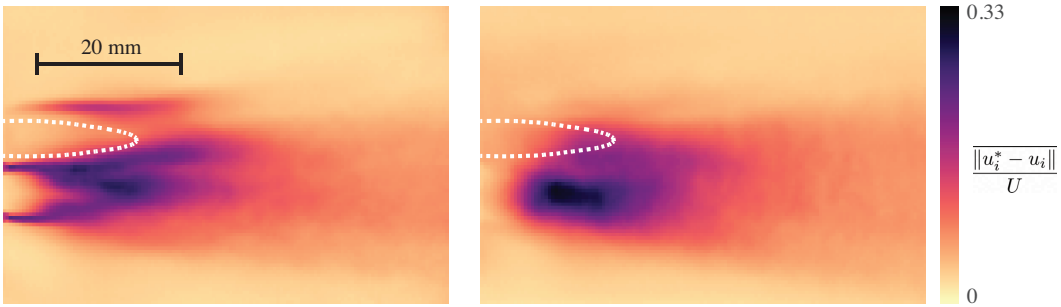

Figure 4.15: Time-averaged error fields of the prediction at $t^* = 1.741$ normalized by free-stream velocity. The dotted white line depicts the zero-velocity contour line (i.e., the approximate edge of the recirculation region) from time-averaging the original training data.

x-component field (center column of figure 4.14a, $t^* = 0.871$), we see a small pocket of low-velocity flow predicted just above the tail of the recirculation region, which is not present in the measured velocity field. We note that this occurs immediately below the average edge of the recirculation region from the training data. At the same time-step, we also see a nonphysical negative y-component velocity region begin to form immediately above the previously mentioned nonphysical x-component low velocity region. Examining the evolution of the flow between the first time-step and the fifth time-step there is no intuitive physical explanation as to how this could occur, as there is no indication of these deviations in the up-stream flow, and they are well above the vortex formation region for the moved cylinder. This suggests that it is an artifact of the training set. Advancing to the final prediction step (last column figure 4.14, $t^* = 1.741$), we see that these nonphysical structures continue to develop in the flow. We also note that a strong error signal accumulates in the vortex formation region for both velocity fields. In fact, looking at the y-component velocity field, we can note that the structures which were developed during the vortex formation process are virtually absent from the predicted field by the last time-step. The predicted downstream propagation of fully formed structures become overly smooth which also contributes to the error, although to a lesser extent.

We can also plot the time-averaged error of both components in the final prediction step, shown in figure 4.15. We see here that the x-component error tends to accumulate along a line approximately one diameter above the physical wake, which seems likely to be an artifact from the upper shear-layer in the original training set. We see even more error accumulate along the bottom outer edge of the training set

recirculation region, which agrees well with the nonphysical low-velocity region in that area seen in the x-component of figure 4.14. In the time-averaged y-component error, we also see a high concentration of error near the lower boundary of the training set recirculation region. However, the dominant source of error in this case comes from the vortex formation region for the moved cylinder. This agrees well with what was observed for the instantaneous case, where it appeared that the FNO was not predicting new vortices should form there. The interior of the recirculation region has very little x-component error, which is notable since the FNO has not learned that a recirculation region can be maintained anywhere other than in the original location.

Side-by-side cylinder generalization

From figure 4.13, we know that the FNO experiences higher generalization error for the challenging side-by-side cylinder wake case. We can again examine representative examples of the instantaneous measurements, predictions, and errors to gain more insight into the overall performance for this configuration.

Figure 4.16 shows the double-cylinder flow fields at the first, fifth, and tenth time-step. Immediately, we can note that the double cylinder wake dynamics fill much more of the image-frame than the previous single cylinder cases. Since the prediction error tends to accumulate in the actual wake regions (rather than the surrounding free-stream), the fact that the wake region occupies about twice as much of the frame helps explain why this case has a higher error than the moved cylinder case at the very first time-step.

As noted earlier, the prediction error for the double cylinder case grows only slightly faster than for the moved cylinder case. It is therefore not surprising that we see very similar performance looking at the instantaneous error fields themselves. In the x-component case, we once again see a nonphysical low velocity region accumulate near the trailing edge of the bottom side of what was the recirculation region in the training set. Interestingly, we do not see a similar effect on the top side, although this may be because it is largely enveloped by the physical recirculation region of the top cylinder. We also do not see any corresponding regions in the y-component velocity field, which we had previously seen in the moved cylinder case. We alo still see the predictions become overly smooth by the final step and perform especially poorly in predicting the regions associated with active vortex formation.

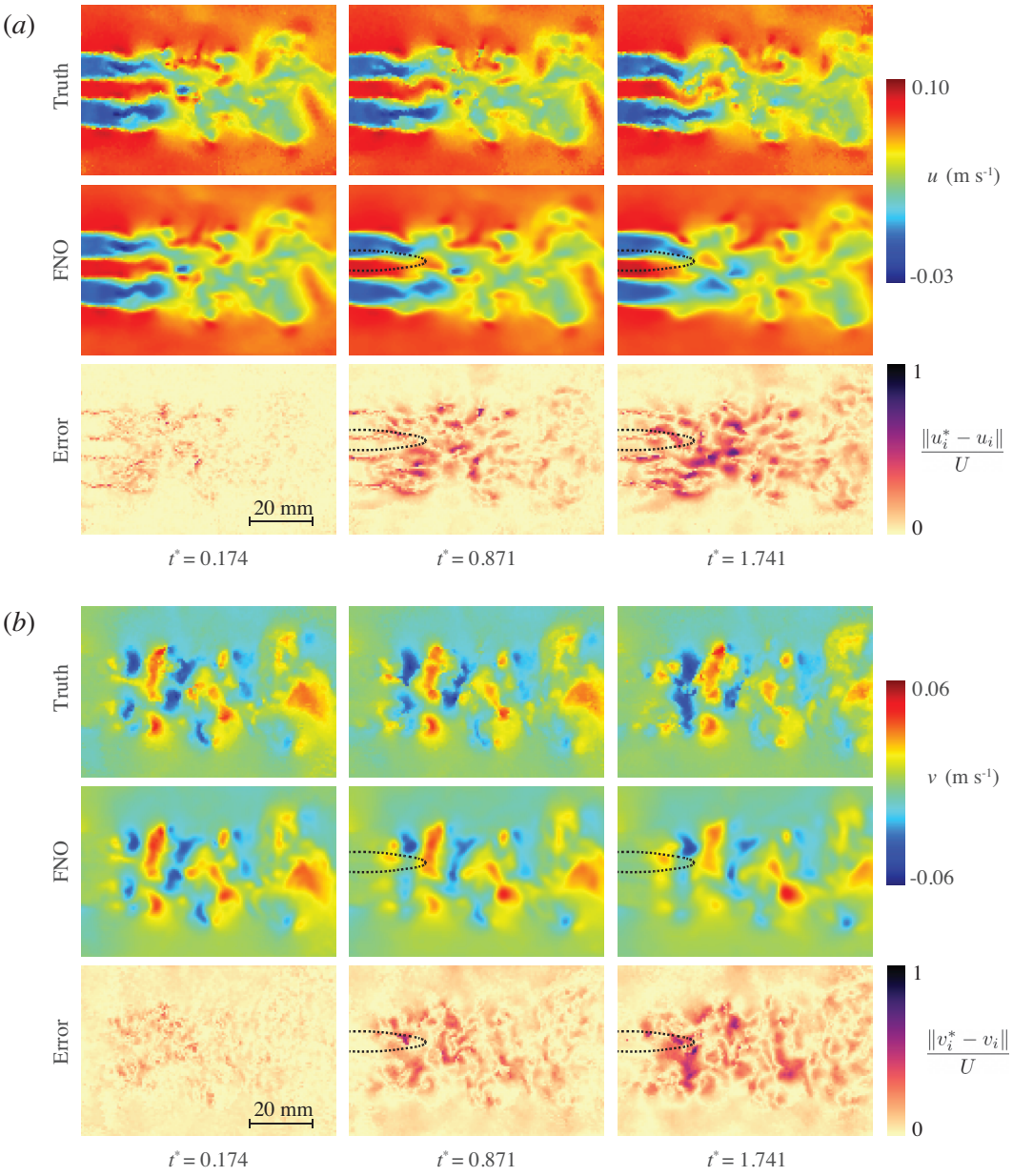

Figure 4.16: Instantaneous (*a*) x-component and (*b*) y-component velocity field measurements, predictions, and error for the double cylinder generalization case at three different timestamps. The dotted black line included in some of the fields depicts the zero-velocity contour line (i.e., the approximate edge of the recirculation region) from time-averaging the original training data.

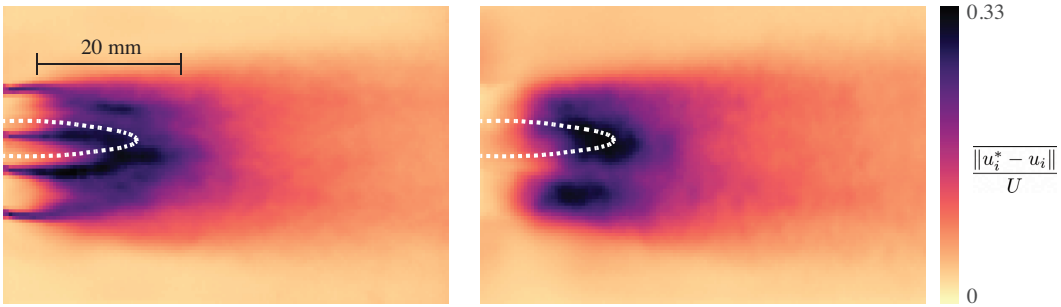

Figure 4.17: Time-averaged error fields of the prediction at $t^* = 1.741$ normalized by free-stream velocity. The dotted white line depicts the zero-velocity contour line (i.e., the approximate edge of the recirculation region) from time-averaging the original training data.

Looking at the time-averaged error fields (figure 4.17), we see that the region associated with the highest error in both fields is near the bottom trailing edge of the white doted line representing the average recirculation region. While we saw elevated error in this region in the moved-cylinder case, it was less dominant compared with the physical vortex formation region in that case. Also distinct from the moved cylinder case, the error in this region for the double-cylinder case is not as obviously correlated with the training recirculation region; while the x-component time-averaged error in the moved cylinder case closely followed the dotted white line, the error fields in figure 4.17 seem to have a distinct geometry. Moreover, the x-component error appears fairly symmetric between the upper and lower cylinder wakes across an axis that is not shared with the training set recirculation region. Looking at the instantaneous velocity fields in figure 4.16, we see that this region is also approximately the midway point between the two cylinders and that the y-component of the velocity fields become very close here. So while this error is likely in-part driven by predictive artifacts as observed in the case where the cylinder was moved, it appears that it is dominated by interactions between the two wakes. While we know that the interactions between the two vortex wakes share largely the same set of physical principles as a single vortex wake, the FNO struggles to generalize these relationships between the cases.

Generalization Improvements

The fact that the FNO struggles to predict vortex formation in the correct region for these cases is unfortunate, but not surprising considering the limitations on the training set. We saw in both cases that the model could not accurately predict the formation of vortices outside of the vortex formation region from the training set data. While it may seem obvious and intuitive that vortex street should move with the recirculation region, there is nothing in the training set that demonstrates this relationship. Additionally, there are cases where we do have a recirculation region but do not have vortex shedding (e.g., very low Reynolds numbers or cases with active vortex suppression) which would generalize poorly without knowing exact boundary conditions even if the cylinder location did not change.

There are ways to improve generalization performance of machine learning models, and some studies have even specialized on this topic specifically for applications in fluid mechanics [29]. We explore some of these generalization methods as shown in figure 4.18. Here we compare the generalization performance the following cases: FNO trained with the full velocity field ($\boldsymbol{u}(\boldsymbol{x},t)$) and the unmodified training set, FNO trained with the full velocity field ($\boldsymbol{u}(\boldsymbol{x},t)$), but some time sequences in the training set are flipped across the horizontal axis, FNO trained to predict only the velocity fluctuations ($\boldsymbol{u'}(\boldsymbol{x},t)$) with unmodified training set, and lastly FNO trained to predict only the velocity fluctuations ($\boldsymbol{u'}(\boldsymbol{x},t)$) with the unmodified training set initially but then trained for 20 more steps on a small set of data matching the test distribution (i.e., transfer learning).

We see that flipping half of the training data sequences did significantly improve generalization for long term predictions, especially in the case where the cylinder simply moved. This is somewhat surprising, as the cylinder was near the center of the frame and did not translate significantly when flipped. However, we find that learning the velocity fluctuations alone (as has been done for all the previous cases, see equation 4.3) results in better generalization than learning the full velocity signal even when flipped.

However, the transfer learning model significantly outperforms all other methods in both generalization cases tested. Here we took the fully trained model that learned the velocity fluctuations, and used the network parameters as the initial weights for a new model. We then trained this new model for only 20 epochs on a limited set of data containing only 800 time sequences. Each epoch took less than 30 seconds on a single NVIDIA RTX-A5000, resulting in the full FNO transfer learning training

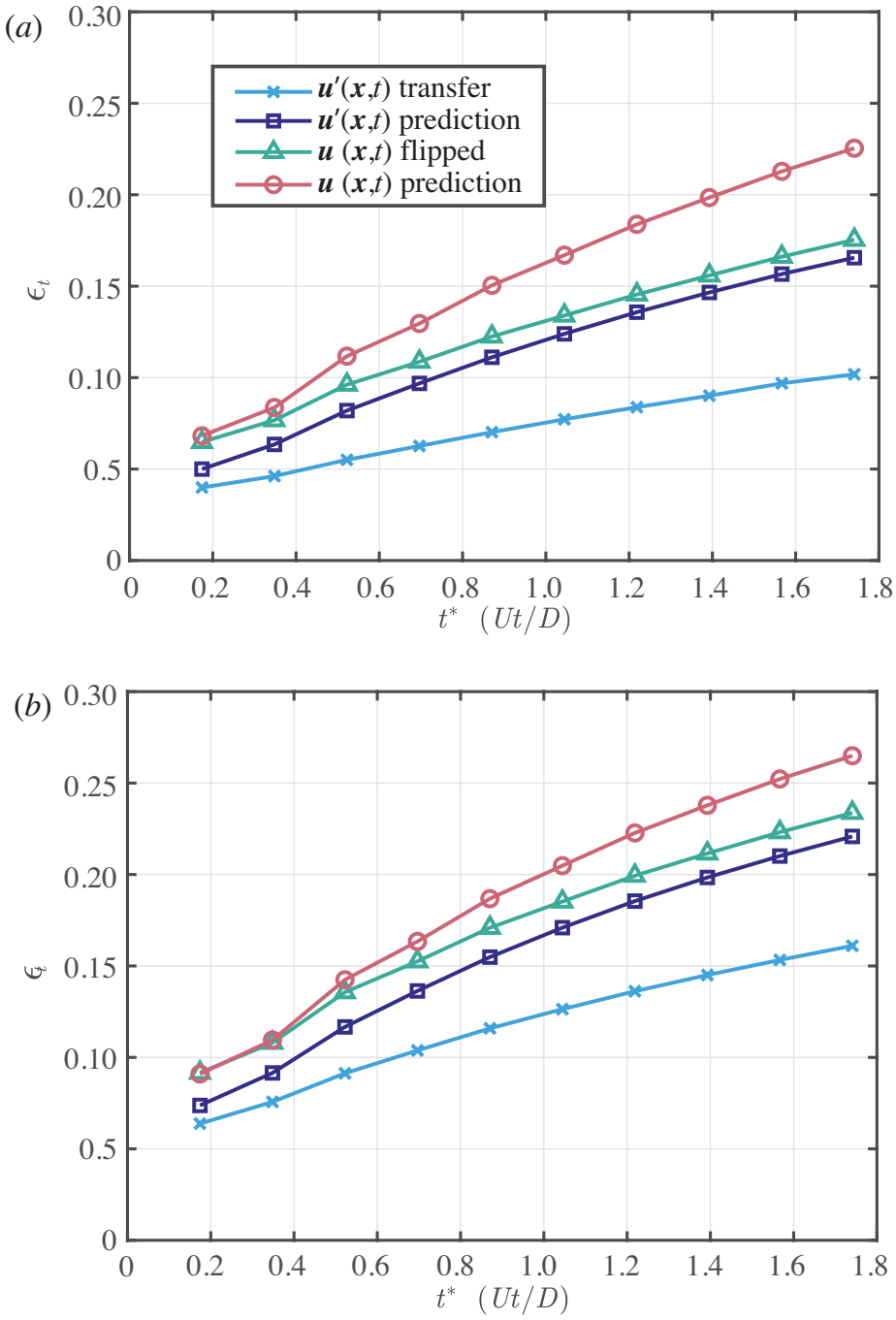

Figure 4.18: Comparing how different training methods generalize for (*a*) moved cylinder and (*b*) double cylinder.

time to be less than 10 minutes. Despite the limited training data and number of epochs, we see a significant reduction in the error in both cases, with the single moved cylinder configuration (figure 4.18a) achieving $\epsilon \approx 0.10$ after 10 time-steps. While the double cylinder model (figure 4.18b) is still at $\epsilon \approx 0.16$ after ten time-steps, we know that complicated wake dynamics occupy a larger portion of the frame in this case which significantly increases the error.

Moved cylinder transfer learning

We can refer to the instantaneous velocity fields, predictions, and errors for the transfer learning case (figure 4.19) to compare the performance to the previous non-transfer learning predictions for the moved cylinder case. Once again we mark the outline of the recirculation region in the training case. However, here we do not see any nonphysical structures or clear correlations between the error and the previous geometry. Additionally, we see non-zero y-component velocity being correctly predicted in the vortex formation region. This demonstrates that the updated FNO model can now predict vortex shedding in the correct region for the moved cylinder case. Overall, the instantaneous performance now appears similar to the non-generalization cases shown in the previous section.

Looking at the time-averaged error for the same case (figure 4.20), we confirm that there is no obvious modeling error due to the difference in geometry from the training set. The error has overall lower magnitude compared to the non-transfer learning generalization case (figure 4.15), and the error only accumulates in regions of the wake similar to those we saw when the model was tested on geometry for which it was trained.

Double cylinder transfer learning

Although the error remains higher than in the moved cylinder case, the FNO model also improves considerably for the double cylinder geometry.

The instantaneous velocity fields and error for the transfer learning FNO are shown in figure 4.21. While the smoothing that seems characteristic for these FNO predictions does impact the accuracy of some of the smaller features in the y-component, the general patterns in the wake are matched well. We note that the FNO even correctly predicts that a portion of the recirculation region will disconnect from the top cylinder wake in the x-component velocity.

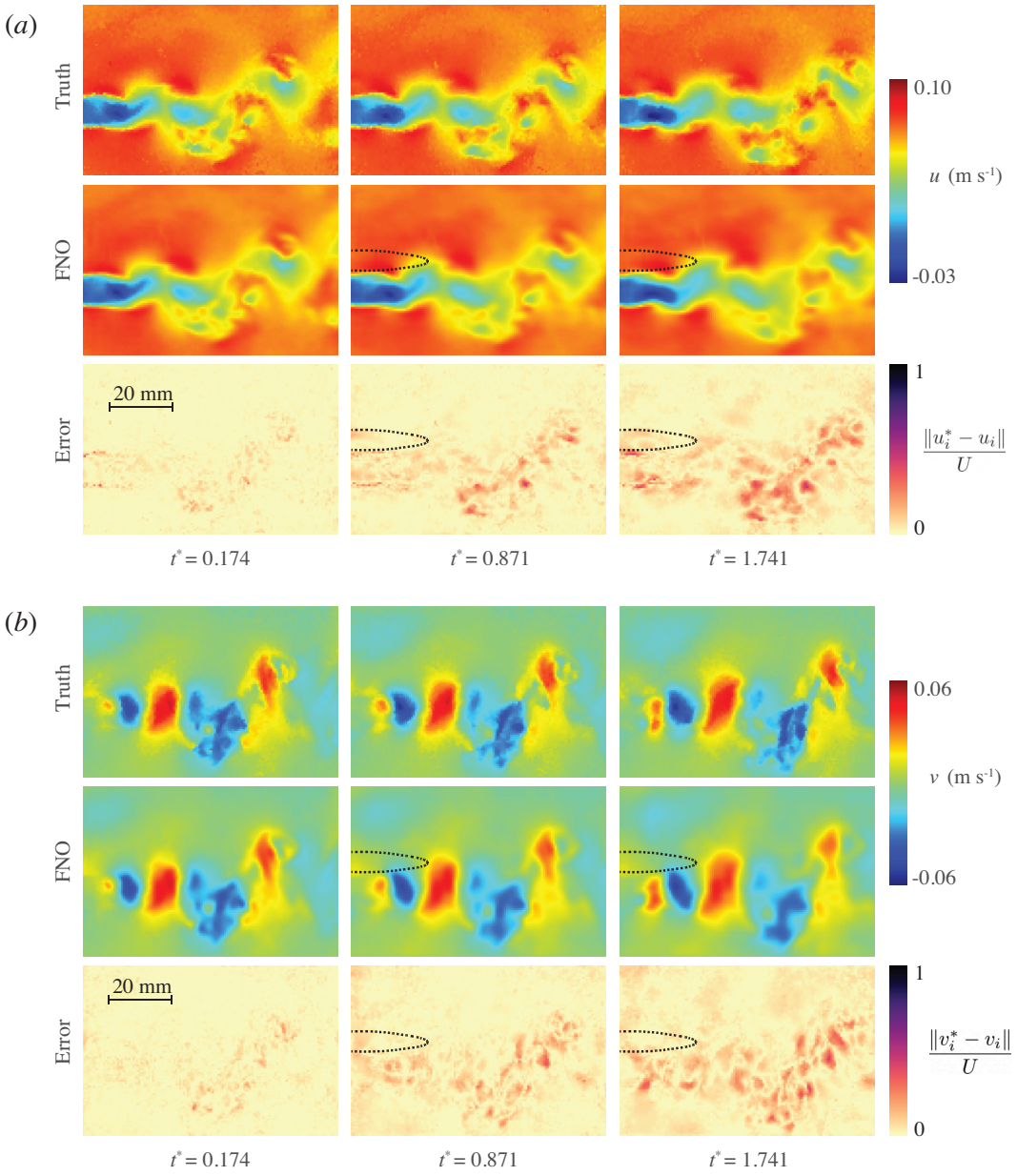

Figure 4.19: Instantaneous (*a*) x-component and (*b*) y-component velocity field measurements, predictions, and error for the transfer learning moved cylinder generalization case at three different timestamps. The dotted black line included in some of the fields depicts the zero-velocity contour line (i.e., the approximate edge of the recirculation region) from time-averaging the original training data.

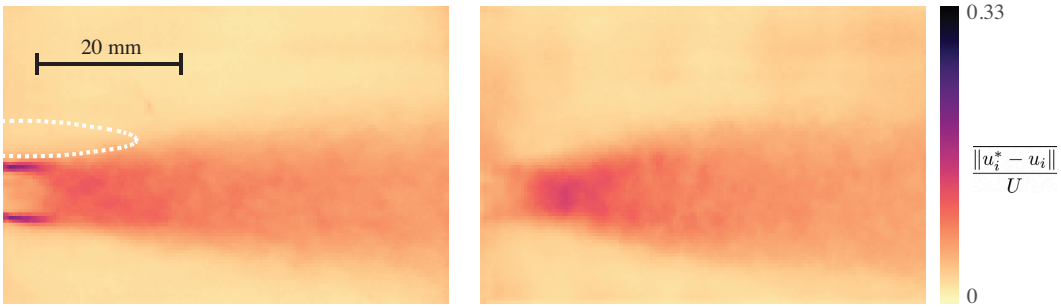

Figure 4.20: Time-averaged error fields of the prediction at $t^* = 1.741$ normalized by free-stream velocity for transfer learning FNO. The dotted white line depicts the zero-velocity contour line (i.e., the approximate edge of the recirculation region) from time-averaging the original training data.

Looking at the time-averaged velocity fields for the FNO with transfer learning for the double cylinder geometry (figure 4.22), we see lower magnitude errors than achieved by the FNO before updating. Interestingly these errors follow roughly the same shape as those produced by the initial FNO predictions. The error we see now for this case is approximately equivalent to two single cylinder error fields superimposed.

4.6 Long prediction times for FNO

So far we have considered a prediction horizon of ten time-steps for each case. These time-steps are non-dimensionalized, as shown in equation 4.1, so that the free stream velocity translates the same amount between each set of frames. Although the time-steps are relatively large, the full time horizons used to this point represent much less than a full cycle of vortex shedding. Since we see that the prediction error increases with the number of time-steps, it is useful to know the approximate prediction length for which a trained FNO can produce meaningful estimates from real world data.

Here we train two FNOs to predict the evolution of cylinder flows over thirty time-steps. Since training for thirty time-steps requires three times the amount of data for any given case we test, we only recorded sufficient training data at two Reynolds numbers: $Re = 630$ and $Re = 3060$.

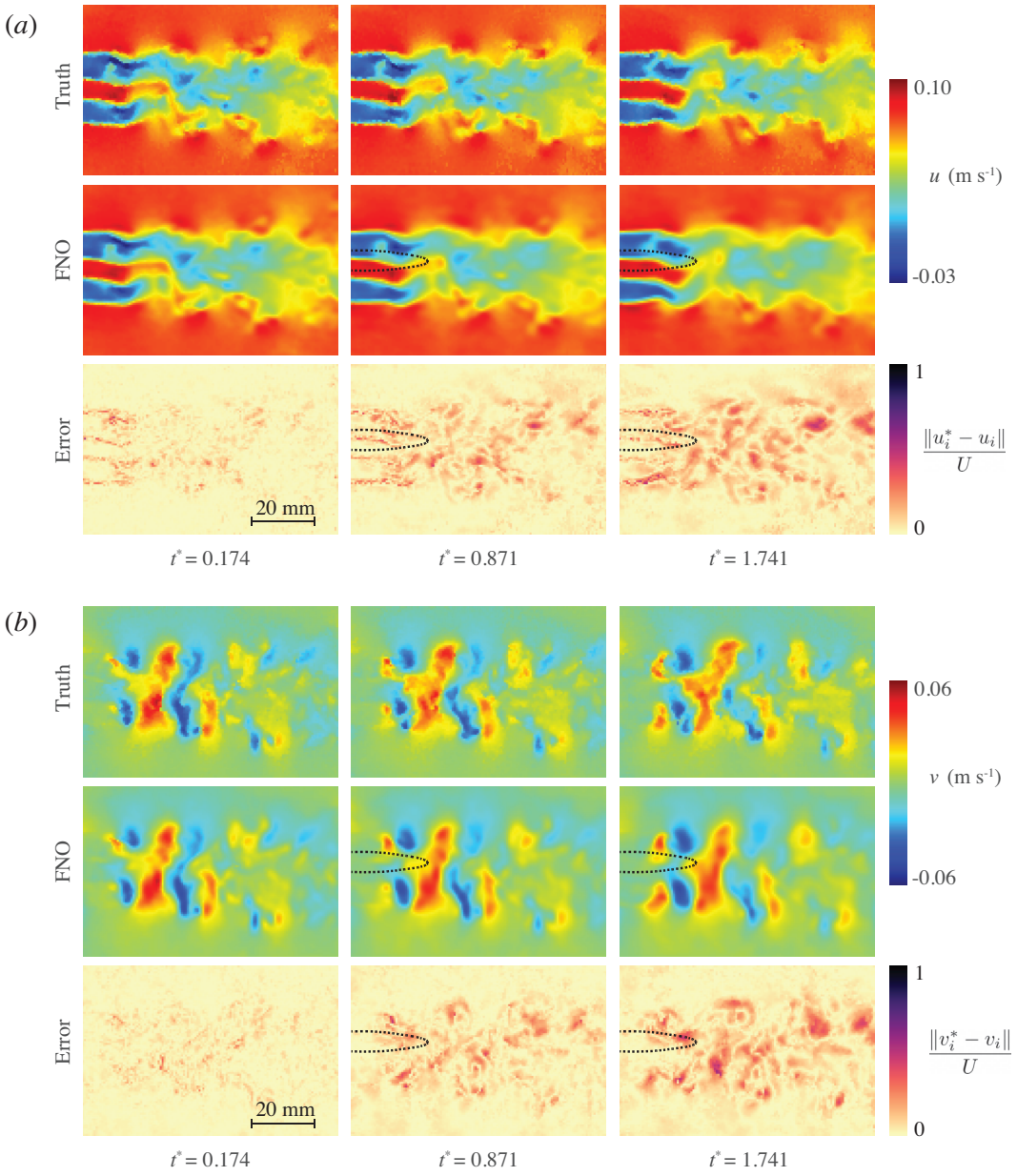

Figure 4.21: Instantaneous (*a*) *x*-component and (*b*) *y*-component velocity field measurements, predictions, and error for the transfer learning double cylinder generalization case at three different timestamps. The dotted black line included in some of the fields depicts the zero-velocity contour line (i.e., the approximate edge of the recirculation region) from time-averaging the original training data.

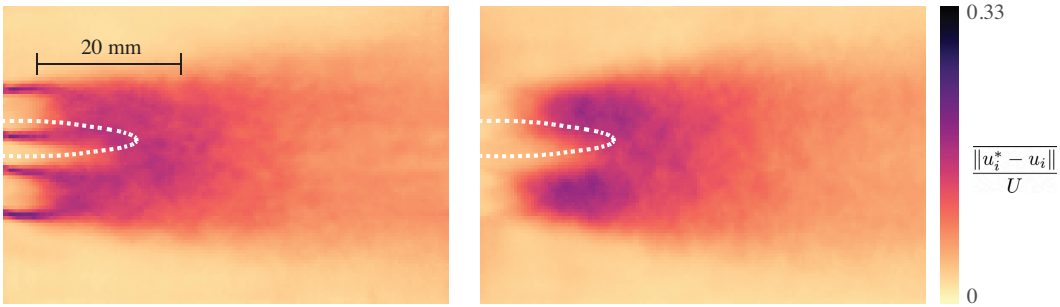

Figure 4.22: Time-averaged error fields of the prediction at $t^* = 1.741$ normalized by free-stream velocity for transfer learning FNO. The dotted white line depicts the zero-velocity contour line (i.e., the approximate edge of the recirculation region) from time-averaging the original training data.

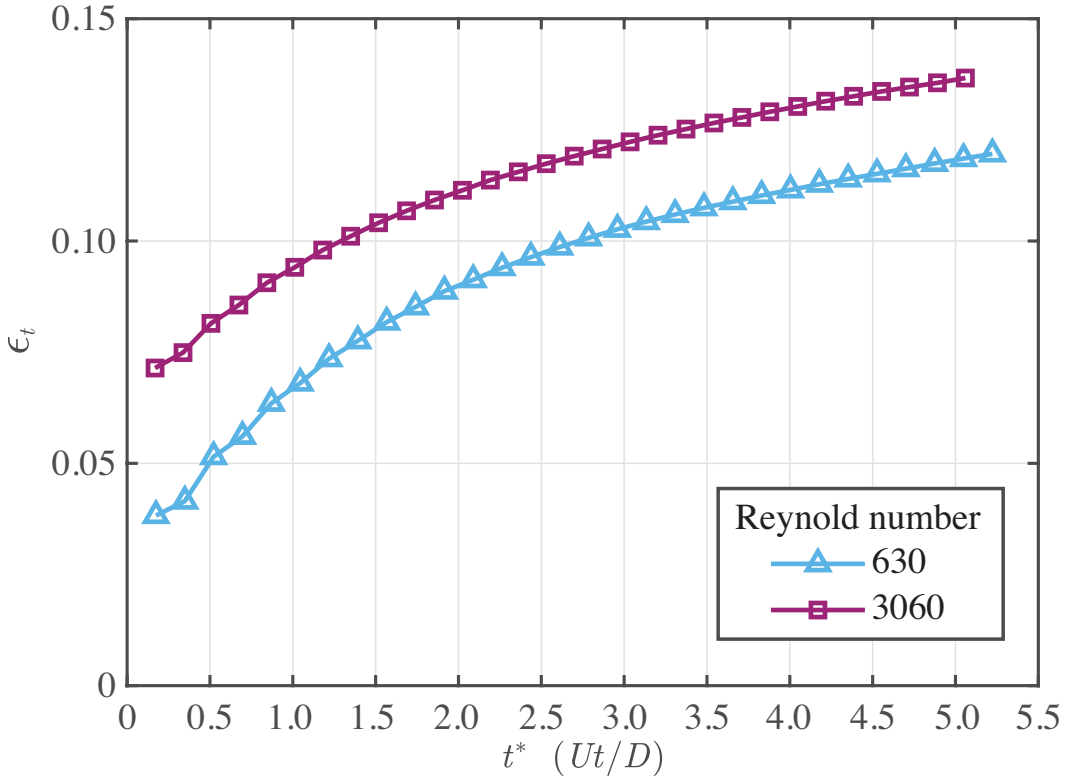

Figure 4.23: Plotting error against time for longer prediction horizons.

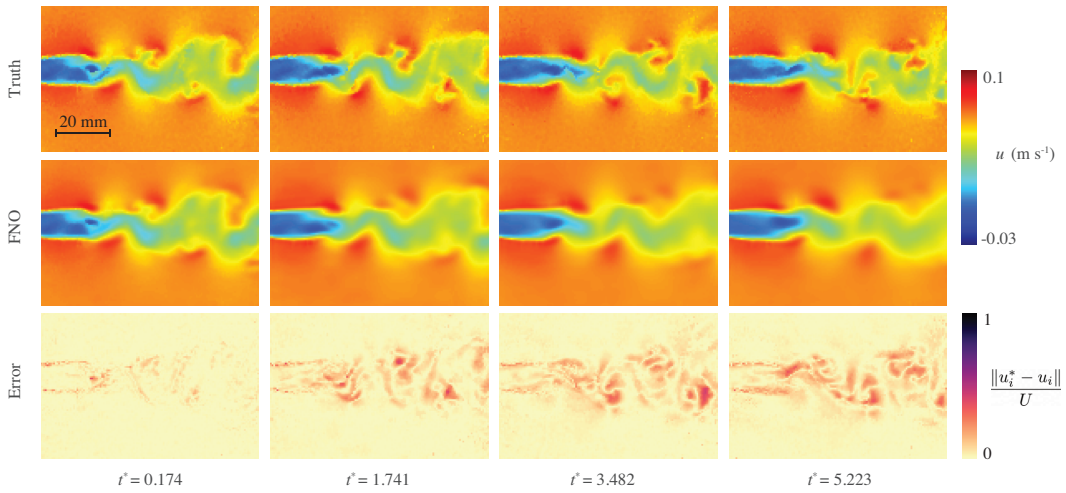

Figure 4.24: Instantaneous x-component velocity field measurements, predictions, and error after 1, 10, 20, and 30 prediction steps for $Re = 630$.

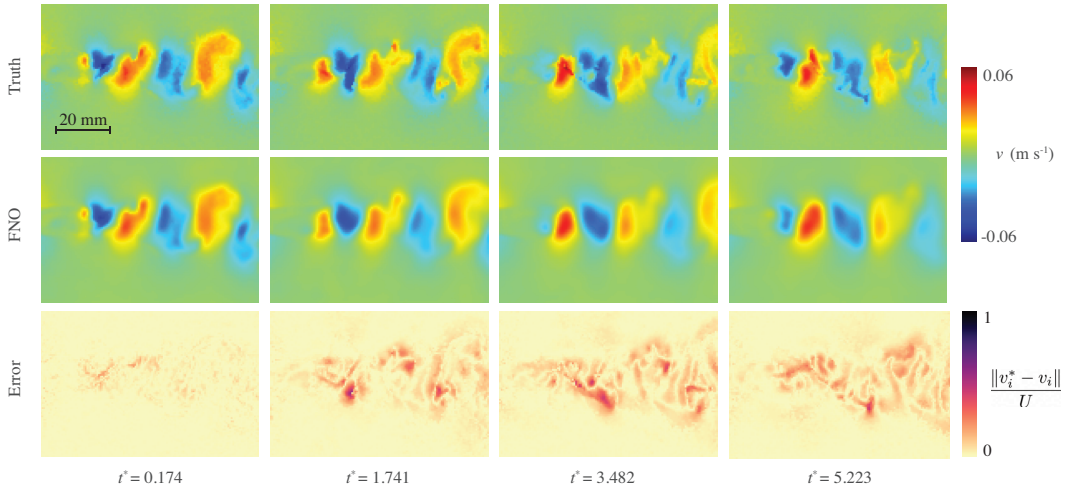

Figure 4.25: Instantaneous y-component velocity field measurements, predictions, and error after 1, 10, 20, and 30 prediction steps for $Re = 630$.

Plotting error against prediction time, figure 4.23 shows the full prediction horizon. We see that the total error remains less than 0.15, even for the more turbulent high Reynolds number case. Again, the relationship between error and time is not exactly linear; over this long horizon, the error contribution of each time-step actually decreases as the prediction continues. The relatively sharp rise in error over the first ten time-steps is likely attributable to an initial smoothing of the model estimates. To confirm this, we can examine velocity fields at various stages of the prediction.

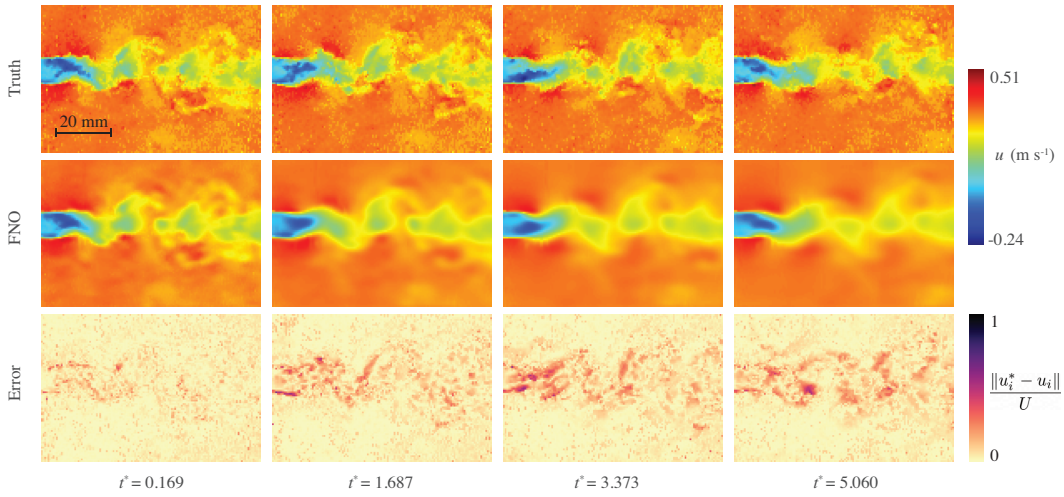

Figure 4.26: Instantaneous x-component velocity field measurements, predictions, and error after 1, 10, 20, and 30 prediction steps for Re = 3060.

Figures 4.24 and 4.25 depict instantaneous velocity fields at the first, tenth, twentieth and thirtieth time-step prediction for Re = 630. As expected the predictions get considerably smoother between the first and tenth prediction steps, and then remain similarly smooth over the rest of the forecast. Although the predictions become smooth, they do maintain some notable characteristics. For example, we see in the final frame of the x-component measurement that the low-velocity region immediately behind the recirculation region is separated by a small higher velocity flow region. Although the feature is significantly smoother, we see a similar effect in the same region of the predicted case. This feature is not obvious or intuitive based on the state of the flow in the first time-step shown, but is somehow still included in the prediction nearly a full vortex shedding cycle later.

Figures 4.26 and 4.27 depict instantaneous velocity fields at the first, tenth, twentieth and thirtieth time-step prediction for the Re = 3060 case. As we did for the predictions made for the shorter time horizon at this Reynolds number, we see considerable error contribution from high-frequency fluctuations, particularly in the x-component velocity fields. Once again we see the predictions smooth out between the first and the tenth prediction step. The predicted shedding frequency and translational velocity of the resulting vortices is matched very well here even over a long horizon.

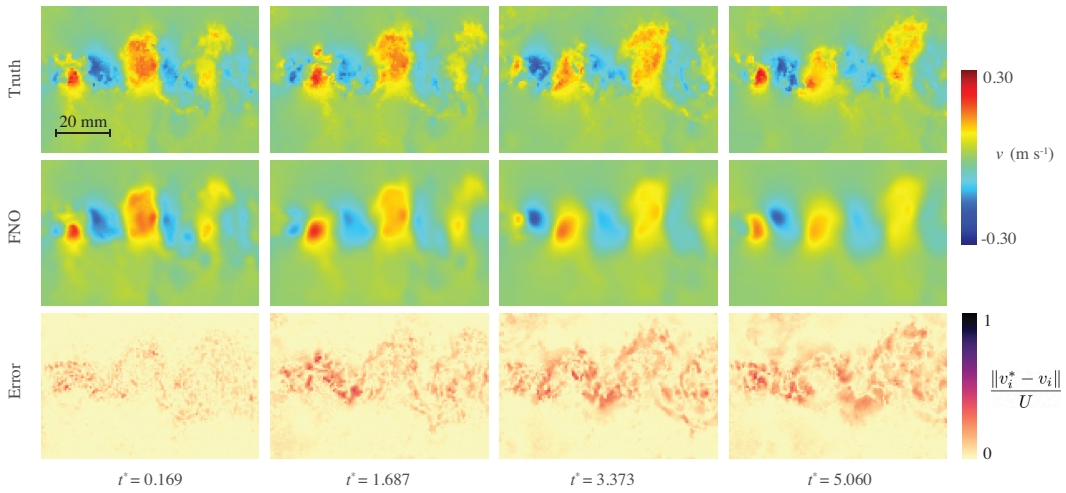

Figure 4.27: Instantaneous y-component velocity field measurements, predictions, and error after 1, 10, 20, and 30 prediction steps for $Re = 3060$.

While we see that forecasting flow fields with FNO over long time horizons results in very smooth predictions, they maintain useful information about the actual system. At sufficiently large horizons, the predictions seem to become something similar to phase-locked time-averages that represent the state of the average velocity field at the given point in the shedding cycle. While this means that information about the fine-scale dynamics can be lost, it is important to once again remark on the efficiency of this prediction method. The flow measurements shown in figures 4.24 and 4.25 span 0.75 seconds, while the prediction of all thirty time-steps can be made in less than 100 ms.

4.7 Conclusion

We apply a state-of-the-art operator learning technique (FNOs) to forecast the time evolution of cylinder flows at a range of Reynolds numbers in the subcritical regime. Subtracting out the time-averaged solution, we simplify the underlying PDE and improve learning performance. We find that the resulting FNOs achieve low errors ($\epsilon < 0.1$) even after ten prediction steps roughly equivalent to one-third of a vortex shedding cycle.

We find that error increases with Reynolds number at early prediction time-steps, but that this relationship changes after several iterations. We see that the error in the final prediction step increases sharply with Reynolds number between $Re = 240$ and $Re = 890$, but then does not change significantly between higher Reynolds numbers tested. This may be due to an accumulating smoothness learned to minimize error from unsteady turbulent dynamics that were not learned. However, it is possible that these fine-scale interactions were not learned due to insufficient spatial resolution in our measurements. Given sufficient information to better resolve small secondary flow structures known to be present in the wake at higher Reynolds numbers (e.g., shear vortices), the FNO could potentially learn to accurately predict flow across a wider set of length scales. While out-of-plane velocities likely contributed to error in our measurements which also impacted FNO performance, FNOs are not limited to 2D solutions, and it is possible that they could properly resolve 3D structures if given volumetric three-dimensional flow data.

We also explore the generalization capabilities of FNO on our system by applying a fully trained model to unseen conditions. Specifically, we take our model trained on data with $Re = 630$ in the default cylinder configuration and try to predict the evolution for two different geometries: (1) the cylinder moved to the bottom of the frame and (2) a second cylinder added for side-by-side vortex shedding. The default FNO velocity predictions show acceptable estimates for translation of existing structures and stability of the recirculation region, but fail to correctly predict the formation of new vortices in the correct locations. We then show that learning the PDE alone improves generalization considerably over learning the full velocity signal, and even outperforms learning the full velocity signal while taking specific steps to improve generalization (i.e., flipping some time sequences). Finally, we perform transfer learning and find that our original learned model can rapidly adapt to new configurations with just a fraction of the training time or data provided. The transfer learning method shows error fields with characteristics similar to the original fully trained model, despite being updated with a set of data 20% the size of the original training set and for only 20 additional epochs. This is extremely promising for real-world applications of predictive flow technologies where training data may be difficult to acquire.

Lastly, we examine the behavior of FNOs for longer prediction horizons, training and testing for a prediction length three-times that used in the previous examples. We find that the error accumulation per time-step actually decreases after approximately 10 prediction time-steps, but that this tapering effect coincides with the velocity field predictions becoming very smooth. This strategy is likely learned by the FNO to compensate for stochastic turbulent fluctuations in the flow field which the model is not capable of predicting over such a significant span of time; if it can not exactly match the irregular fluctuations in the flow predicting something similar to a phase-averaged flow field for every point in the shedding cycle is an effective strategy at minimizing error.

This work represents a first step towards operator learning in experimental fluid mechanics. While this application required PIV processing that precluded FNO from being applied in real-time here, we have shown that this real-time capable prediction method is robust and accurate when trained with experimental data. While real-time PIV systems do exist [20, 35, 38], full-field velocity information is not a practical reality for control in most deployed engineered systems. However, flow field reconstruction from sparse measurements has been a focus of many recent works [1, 3, 6, 9, 11, 25, 27]. These reconstruction methods could be used directly in conjunction with the FNO methods applied here to predict the future state of full-field data from a set of distributed sparse sensors. Additionally, recent developments for FNOs have enabled more flexible spatial measurements [23], making the FNO itself a potential end-to-end solution for flow field reconstruction and forecasting the time evolution. The development of systems with the ability to predict complicated fluid dynamics faster than their physical realization is an exciting new endeavour in fluid mechanics research.

Bibliography

[1] Ido Bright, Guang Lin, and J. Nathan Kutz. Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25, 2013.

[2] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[3] Jared L. Callaham, Kazuki Maeda, and Steven L. Brunton. Robust flow reconstruction from limited measurements via sparse representation. *Physical Review Fluids*, 4(10):103907, 2019.

[4] Zhiwen Deng, Chuangxin He, Yingzheng Liu, and Kyung Chun Kim. Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework. *Physics of Fluids*, 31, 2019.

[5] Patricio Clark Di Leoni, Lu Lu, Charles Meneveau, George Em Karniadakis, and Tamer A. Zaki. Neural operator prediction of linear instability waves in high-speed boundary layers. *Journal of Computational Physics*, 474:111793, 2023.

[6] N. Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L. Brunton, Michael W. Mahoney, and J. Nathan Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476(2238):20200097, 2020.

[7] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstrucion of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870, 2019.

[8] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909, 2021.

[9] Kai Fukami, Romit Maulik, Nesar Ramachandra, Koji Fukagata, and Kunihiko Taira. Global field reconstruction from sparse sensors with voronoi tessellation-assisted deep learning. *Nature Machine Intelligence*, 3(11):945–951, 2021.

[10] Kai Fukami, Takaaki Murata, Kai Zhang, and Koji Fukagata. Identification of nonlinear dynamics with low-dimensionalized flow representations. *Journal of Fluid Mechanics*, 926, 2021.

[11] Kai Fukami, Byungjin An, Motohiko Nohmi, Masashi Obuchi, and Kunihiko Taira. Machine-learning-based reconstruction of turbulent vortices from sparse pressure sensors in a pump sump. *Journal of Fluids Engineering*, 144(12): 121501, 2022.

[12] M. Hammache and M. Gharib. An experimental study of the parallel and oblique vortex shedding from circular cylinders. *Journal of Fluid Mechanics*, 232:567–590, 1991.

[13] Renkun Han, Yixing Wang, Yang Zhang, and Gang Chen. A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network. *Physics of Fluids*, 33, 2019.

[14] Kazuto Hasegawa, Kai Fukami, Takaaki Murata, and Koji Fukagata. Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. *Theoretical and Computational Fluid Dynamics*, 34, 2020.

[15] Kazuto Hasegawa, Kai Fukami, Takaaki Murata, and Koji Fukagata. Cnn-lstm based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different reynolds numbers. *Fluid Dynamics Research*, 52, 2020.

[16] David Jeon and Morteza Gharib. On the relationship between the vortex formation process and cylinder wake vortex patterns. *Journal of Fluid Mechanics*, 519, 2004.

[17] George Em Karniadakis and George S. Triantafyllou. Three-dimensional dynamics and transition to turbulence in the wake of bluff objects. *Journal of Fluid Mechanics*, 238, 1992.

[18] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22, 2021.

[19] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *arXiv preprint arXiv:2108.08481*, 2021.

[20] Mark Kreizer, David Ratner, and Alex Liberzon. Real-time image processing for particle tracking velocimetry. *Experiments in Fluids*, 48(1):105–110, 2010.

[21] L. D. Landau and E. M. Lifshitz. *Fluid Mechanics*. Elsevier, 1987.

[22] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[23] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022.

[24] Chensen Lin, Martin Maxey, Zhen Li, and George Em Karniadakis. A seamless multiscale operator neural network for inferring bubble dynamics. *Journal of Fluid Mechanics*, 929, 2021.

[25] Jean-Christophe Loiseau, Bernd R. Noack, and Steven L. Brunton. Sparse reduced-order modelling: sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844:459–490, 2018.

[26] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[27] Kevin H. Manohar, Chris Morton, and Paul Ziadé. Sparse sensor-based cylinder flow estimation using artificial neural networks. *Physical Review Fluids*, 7(2):024707, 2022.

[28] Sudeepta Mondal and Sournalya Sarkar. Multi-fidelity prediction of spatiotemporal fluid flow. *Physics of Fluids*, 34, 2022.

[29] Masaki Moriomoto, Kai Fukami, Kai Zhang, and Koji Fukagata. Generalization techniques of neural networks for fluid flow estimation. *Neural Computing and Applications*, 34, 2021.

[30] Taichi Nakamura, Kai Fukami, and Kazuto Hasegawa. Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Physics of Fluids*, 33, 2021.

[31] F.S. Pereira, G. Vaz, L. Eça, and S.S. Girrimaji. Simulation of the flow around a circular cylinder at Re = 3900 with partial-averaged Navier-Stokes equations. *International Journal of Heat and Fluid Flow*, 2017.

[32] Anatol Roshko. Report 1191: On the development of turbulent wakes from vortex streets. *National Advisory Committee for Aeronautics*, 1958.

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

[34] Kunihiko Taira, Steven L. Brunton, Scott T. M. Dawson, Clarence W. Rowley, Tim Colonius, Beverley McKeon, Oliver T. Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S. Ukeiley. Modal analysis of fluid flows: An overview. *AIAA Journal*, 52, 2017.

[35] Eliott Varon, Jean-Luc Aider, Yoann Eulalie, Stephie Edwige, and Philippe Gilotte. Adaptive control of the dynamics of a fully turbulent bimodal wake using real-time piv. *Experiments in Fluids*, 60(8):1–21, 2019.

[36] Pantelis R. Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.

[37] T. Wei and C. R. Smith. Secondary vortices in the wake of circular cylinders. *Journal of Fluid Mechanics*, 1968.

[38] Christian Willert, Matthew Munson, and Morteza Gharib. Real-time particle image velocimetry for closed-loop flow control applications. In *15th International Symposium on Applications of Laser Techniques to Fluid Mechanics*, 2010.

[39] C.H.K. Williamson. Vortex dynamics in the cylinder wake. *Annual Review of Fluid Mechanics*, 28, 1996.

[40] C.H.K. Williamson. Three-dimensional wake transition. *Journal of Fluid Mechanics*, 328, 1996.

[41] Pei Li Wu, Hang Shan Gao, Qiong Wang, and Pei Yan Wang. A novel forecast framework for unsteady flows based on a convolutional neural network. *Physics of Fluids*, 34, 2022.

Chapter 5

CONCLUSION

This thesis presented applications of machine learning for experiments in applied fluid mechanics. We first exploref model-free reinforcement learning for end-to-end control of aerodynamic systems, and then presented the first application of operator-learning to experimental fluid measurements for forecasting the time evolution of fluid flows. Here we provide a summary of our findings, and discuss future directions for this work.

5.1 Model-free reinforcement learning for aerodynamic control in a turbulent environment

In Chapter 3, we deployed a state-of-the-art model-free reinforcement learning algorithm to control a highly stochastic and challenging fluid environment. Model-free reinforcement learning methods present a black-box end-to-end solution to the problem of control; they learn to improve performance based on past experiences without the need of prior knowledge or modeling. These algorithms take observations as inputs and can choose actions as outputs. Many reinforcement learning algorithms are capable of learning effective control policies even when faced with stochastic learning environments and nonlinear dynamics [7].

A novel aerodynamic test-bed was developed, which consisted of a symmetric airfoil shape with actuated trailing flaps. Flow sensors were integrated into the design of the test-bed, to provide information about the surrounding fluid flow to the reinforcement learning agents. This system was mounted on a one-dimensional load cell, and placed in a highly vortical and irregular flow field. The challenging flow conditions were generated by a large asymmetric bluff body mounted upstream on a set of bungee cords, allowing it to oscillate and rotate freely.

First, we compared the performance of two model-free reinforcement learning algorithms and a conventional proportional integral derivative (PID) controller. The first reinforcement learning algorithm tested, Twin Delayed Deep Deterministic Policy Gradient (TD3) [2], was an actor-critic type method that had been deployed in one of the few previous applications of reinforcement learning for experimental flow control [1]. The second learning algorithm, known as LSTM-TD3 [5], has

the same algorithmic structure as TD3 with minor modifications to allow for Long Short-Term Memory (LSTM) layers to be included. LSTMs are a type of recurrent neural network (RNN) cell, which have been shown to significantly improve performance in partially observable environments. The reinforcement learning agents were provided flow measurements and inertial measurements, and given the goal of minimizing the magnitude of the lifting force in the turbulent bluff body wake.

We found that the average performance of the TD3 algorithm was very similar to that of the linear PID controller. Additionally the variance in the learning performance between models was significant for the TD3 algorithm which suggests that it struggled to reliably learn the dynamics of the system. The LSTM-TD3 algorithm outperformed the PID controller and TD3 algorithm by a wide margin, and learned consistently across separately trained agents. This suggests that the inclusion of LSTM cells enables learning effective dynamics of highly stochastic flow conditions.

Next, inspired by biological swimmers and flyers we studied the value of different sets of observations provided to our reinforcement learning agents. We were especially interested in the impact of flow measurements on learned performance as conventional UAS tend to rely entirely on inertial measurement systems. We found that agents observing both flow measurements and inertial measurements learned more consistently than agents given only one of the two types. The performance of the fully trained policy was similar between agents given either flow, inertial measurements, or combined measurements. However the combined measurement agents appeared to achieve better disturbance rejection. Highlighting the value of flow observations, the agents provided only flow measurements performed similarly to the other two types of agents despite having no way to directly observe the inertial values tied to the reward function.

We conclude this chapter by discussing limitations with model-free reinforcement learning in an experimental context. We also reference a recent collaboration, not included in this thesis, for a model-based reinforcement learning algorithm which we found significantly outperformed state-of-the-art model-free methods with far less data.

5.2 Fourier neural operators for turbulent cylinder wakes

In Chapter 4, we present an application of Fourier neural operators (FNOs) for predicting the time evolution of experimental fluid flows. FNOs are a recently introduced machine learning tool capable of estimating solution operators to families of PDEs such as Navier-Stokes. Directly approximating the solution operator, FNOs are mesh-invariant and orders-of-magnitude faster than conventional solvers [3]. This makes them ideal for applications requiring real-time modeling of complicated fluid mechanics.

We used particle image velocimetry (PIV) to record the wake behind a cylinder at a range of Reynolds numbers within the subcritical turbulent regime. We then decomposed the velocity field into steady and non-steady parts and assumed that the steady part satisfies the time-independent Navier-Stokes equation. We then tried to learn only the unsteady velocity term, which reduced the Navier-Stokes equations to a set of homogeneous linear differential equations. Using the unsteady component, we trained FNOs to predict the state of the experimental flow at future time-steps. We calculated the L2 norm error between the model predictions and the measurement to establish the accuracy of the forecast at various time-steps. We found that this error remained below 0.1 for our full prediction horizon consisting of ten time-steps in advance from the last measured value.

We then then examined how FNOs can generalize across unseen flow conditions. As before we first trained an FNO model with experimental flow measurements of a cylinder wake, where the cylinder is located such that the wake was nearly centered in the frame. We then applied this model to two distinct geometries. First, we moved the same cylinder to the bottom of the frame so that the wake was far from its original location. Then, keeping the cylinder moved to the bottom of the frame in place, we added a second cylinder near the top of the frame in a side-by-side configuration. We found that the FNO is able to generalize much better when learning only the unsteady velocity field, and could predict the convection and diffusion of existing structures. However, we also found that it struggles to correctly predict the vortex formation process behind the new cylinder locations. We resolved this with an application of transfer learning, by updating our models with small training sets from the new distribution for only 20 epochs. The updated FNOs were able to accurately predict vortex shedding in the correct regions in both generalization cases, and reduced the overall error significantly.

Lastly, we increased the prediction horizon of the training data to train FNOs for a prediction horizon of thirty time-steps (three-times the length previously used). We found that the prediction error grew with time most quickly over the first ten time-steps, at which point the predictions became smooth and averaged and the error increased less significantly at each time-step. This suggested that the FNO models struggled to forecast the fine-scale dynamics of the flow over longer prediction horizons and so instead produced something similar to a phase-averaged field as an estimate to minimize the overall error.

5.3 Future directions

We have demonstrated some of the first applications of machine learning in the context of applied experimental fluid mechanics. The combination of these two fields holds significant promise for developing solutions to open problems in engineering and presents many exciting opportunities for researchers.

Model-free reinforcement learning methods, such as those demonstrated in Chapter 3, are conceptually an attractive option for control of nonlinear stochastic systems such as those commonly encountered in fluid mechanics. These methods synthesize diverse sensor information as part of their natural learning process, require no a priori modeling or system information, adapt to new environments, and can even learn policies that outperform optimal control methods [6]. However, model-free reinforcement learning largely lack performance and learning guarantees, produce uninterpretable control policies, and have significant data requirements for training. Being able to physically interpret control decisions is especially important in deploying systems to the real world, as is guaranteeing that a safe and reliable policy will be learned. Model-based reinforcement learning can provide many of the benefits of model-free methods (e.g., sensor fusion, learn to adapt to new environments) but can provide guarantees and more physically interpretable models. While some a priori knowledge of the system is necessary for building an initial model, the result is an accelerated learning process with reduced data requirements. Model-based reinforcement learning may therefore be a better solution to future control problems in fluid mechanics.

Fourier neural operators, which we applied to experimental flow measurements in Chapter 4, are an efficient and powerful learning method capable of modeling complicated flow physics. They can forecast the time evolution of experimental flow fields faster than real-time, and can be generalized with minimal data and training

requirements. This method could provide a solution to real-time flow modeling for aerodynamic control of UAS and wind turbines. However, our current application has been provided full velocity fields as input, which cannot be practically obtained in most deployed systems. While there already exist efforts to apply FNOs to irregular geometries [4], the problem of making full field predictions from sparse sensing as an end-to-end solution has not been fully addressed to our knowledge. Making predictions from sparse measurements would enable implementation of real-time predictive control methods provided with full-field estimates for the evolution of the flow. Additionally, future work applying these methods to applied fluid mechanics should focus on generalizing models more diverse boundary conditions where incident flow is not limited to a single principle direction. This includes generalizing the sensor-selection problem, where optimal sensor locations for flow along one axis is unlikely to be optimal for all other directions. This would present a more challenging and realistic setting for real-world prediction, as atmospheric winds can change heading depending on season, time of day, or one of many other factors.

Bibliography

[1] Dixia Fan, Liu Yang, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. Reinforcement learning for bluff body active flow control in experiments and simulations. *Proceedings of the National Academy of Sciences*, 117(42):26091–26098, October 2020. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2004939117. URL http://www.pnas.org/lookup/doi/10.1073/pnas.2004939117.

[2] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, Stockholm, Sweden, October 2018. PMLR. URL http://arxiv.org/abs/1802.09477.

[3] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[4] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022.

[5] Lingheng Meng, Rob Gorbet, and Dana Kulic. Memory-based deep reinforcement learning for POMDPs. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626, Prague, Czech Republic, September 2021. IEEE. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9636140. URL https://ieeexplore.ieee.org/document/9636140/.

[6] Guido Novati, L. Mahadevan, and Petros Koumoutsakos. Controlled gliding and perching through deep-reinforcement learning. *Physical Review Fluids*, 2019.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Mass., second edition, 2018. ISBN 978-0-262-03924-6.