Physics-based Machine Learning Methods for Control and Sensing in Fish-like Robots

Colin Rodwell
Clemson University, crodwel@clemson.edu

Physics-based machine learning methods for control and sensing in fish-like robots

A Dissertation
Presented to
the Graduate School of
Clemson University

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

by
Colin Rodwell
December 2023

Abstract

Underwater robots are important for the construction and maintenance of underwater infrastructure, underwater resource extraction, and defense. However, they currently fall far behind biological swimmers such as fish in agility, efficiency, and sensing capabilities. As a result, mimicking the capabilities of biological swimmers has become an area of significant research interest. In this work, we focus specifically on improving the control and sensing capabilities of fish-like robots.

Our control work focuses on using the Chaplygin sleigh, a two-dimensional nonholonomic system which has been used to model fish-like swimming, as part of a curriculum to train a reinforcement learning agent to control a fish-like robot to track a prescribed path. The agent is first trained on the Chaplygin sleigh model, which is not an accurate model of the swimming robot but crucially has similar physics; having learned these physics, the agent is then trained on a simulated swimming robot, resulting in faster convergence compared to only training on the simulated swimming robot.

Our sensing work separately considers using kinematic data (proprioceptive sensing) and using surface pressure sensors. The effect of a swimming body's internal dynamics on proprioceptive sensing is investigated by collecting time series of kinematic data of both a flexible and rigid body in a water tunnel behind a moving obstacle performing different motions, and using machine learning to classify the motion of the upstream obstacle. This revealed that the flexible body could more effectively classify the motion of the obstacle, even if only one if its internal states is used.

We also consider the problem of using time series data from a 'lateral line' of pressure sensors on a fish-like body to estimate the position of an upstream obstacle. Feature extraction from the pressure data is attempted with a state-of-the-art convolutional neural network (CNN), and this is compared with using the dominant modes of a Koopman operator constructed on the data as features. It is found that both sets of features achieve similar estimation performance using a dense neural network to perform the estimation. This highlights the potential of the Koopman modes as an interpretable alternative to CNNs for high-dimensional time series. This problem is also extended to inferring the time evolution of the flow field surrounding the body using the same surface measurements, which is performed by first estimating the dominant Koopman modes of the surrounding flow, and using those modes to perform a flow reconstruction. This strategy of mapping from surface to field modes is more interpretable than directly constructing a mapping of unsteady fluid states, and is found to be effective at reconstructing the flow. The sensing frameworks developed as a result of this work allow better awareness of obstacles and flow patterns, knowledge which can inform the generation of paths through the fluid that the developed controller can track, contributing to the autonomy of swimming robots in challenging environments.

Acknowledgments

I would like to thank my advisor, Dr. Phanindra Tallapragada, for his invaluable advice and mentorship. His broad expertise has allowed me to explore a wide range of interesting problems, and his insight has allowed my solutions to these problems to have a broad impact. I would also like to acknowledge Dr. Paul Joseph and Dr. Joshua Bostwick, whose undergraduate classes sparked my interest in dynamical systems. I would also like to thank my collaborators, Jake Buzhardt, Prashanth Chivkula, and Kumar Sourav, whose expertise has enabled my work to be more relevant and impactful. I would also like to thank these collaborators, as well as labmates Andrew Zheng, Beau Pollard, Vitaliy Fedonyuk, and Kartik Loya, for helping to keep up morale and liven up the office. Finally, I would like to thank my parents for their boundless support.

Table of Contents

Title Page . . . . **i**

Abstract . . . . **ii**

Acknowledgments . . . . **iv**

List of Tables . . . . **vii**

List of Figures . . . . **viii**

1 Introduction **1**
- 1.1 Summary of Chapters 5

2 Curriculum Reinforcement Learning for Path Tracking **7**
- 2.1 Introduction 7
- 2.2 Nonholonomic Constraints - Chaplygin Sleigh and the Joukowski Foil in an Inviscid Fluid 10
- 2.3 Periodic Forcing and Limit Cycles in Reduced Velocity Space 14
- 2.4 Parameter Estimation for the Surrogate Model 17
- 2.5 Reinforcement Learning 19
- 2.6 Results - Velocity and Path Tracking 27
- 2.7 Conclusion 32

3 Nonholonomic constraint-induced bistability **35**
- 3.1 Introduction 35
- 3.2 Mechanical model and governing equations 37
- 3.3 Tunable limit cycles and configuration changes 41
- 3.4 Conclusion 48

4 Sensing with passive appendages **51**
- 4.1 Introduction 51
- 4.2 Experiments 54
- 4.3 Wake Classification and Neural Network Architecture 59
- 4.4 Wake Classification Results 62
- 4.5 Discussion and Conclusion 72

5 Obstacle Localization using spectral properties of the Koopman operator **74**
- 5.1 Introduction 74
- 5.2 Problem Definition 76
- 5.3 Dynamic Mode Decomposition 80
- 5.4 Estimation Approaches 85
- 5.5 Results 92

5.6 Conclusion 94

6 Flow Field Reconstruction from Surface Measurements in Fluid-Structure Interaction 97

6.1 Introduction 97

6.2 Numerical Simulation of Fluid-Structure Interaction 100

6.3 Dynamic Mode Decomposition 107

6.4 Flow Reconstruction 111

6.5 Results 120

6.6 Conclusion 122

7 Conclusions 124

Appendices 126

A Two-link Sleigh Equations 127

B Kinematic Classification Confusion Matrices 130

List of Tables

6.1 Comparison of characteristic flow and vibration quantities for three different mesh densities. 105

List of Figures

2.1 Diagrams of the Chaplygin sleigh and swimming foil. 11
2.2 Trajectories of the Chaplygin sleigh and simulated swimmer. 16
2.3 Velocity limit cycles for the swimmer and a fit Chaplygin sleigh model. 19
2.4 A schematic of the DPG training procedure for the Chaplygin sleigh. 22
2.5 Reward plots with and without pre-training on the Chaplygin sleigh, and limit cycles for the trained policy. 28
2.6 Trajectories on the hydrofoil after training on a low speed surrogate model. 29
2.7 Training a high-speed surrogate model actor. 30
2.8 RL without surrogate model or curriculum learning. 32
2.9 Pure-pursuit based path tracking for the simulated swimmer. 33

3.1 A diagram of a two link Chaplygin sleigh system. 39
3.2 The largest real component of the eigenvalues of the Jacobian about the forward non-buckled fixed point. 42
3.3 Dependence of the locations of fixed points u_x^* and ω_1^* on ϵ for a sampling of energy levels for the conservative system. 43
3.4 The eigenvalues with the highest real component of the Jacobian $\boldsymbol{J}$ of positive u_x buckled fixed points of the conservative system. 44
3.5 Limit cycles and trajectories of the dissipative system. 46
3.6 The turning limit cycles of the dissipative system at varying forcing amplitudes. . . 47
3.7 The eigenvalues η of the T-periodic Poincaré map of the forced and dissipative system. 49
3.8 A decaying turning limit cycle in three dimensions. 50

4.1 A schematic and photograph of the experimental setup. 55
4.2 The steady-state time series of angular velocities. 57
4.3 Overlayed frequency spectra of the coupled head and coupled tail angular velocity. . 58
4.4 The architecture of the classifier. 61
4.5 The overall cross-entropy loss on the validation dataset by training epoch. 63
4.6 Confusion matrices for classifying the time period T. 65
4.7 Confusion matrices for the forcing amplitude of the upstream hydrofoil. 66
4.8 Confusion matrices for the free stream velocity u_∞. 67
4.9 Confusion matrices for the Strouhal number. 68
4.10 The classification accuracy of the Strouhal number. 69
4.11 Summary of the classification accuracy of all data and parameters. 71

5.1 System Diagram. 78
5.2 The refined computational mesh. 79
5.3 The pressure field and pressure vs. time on the surface. 80
5.4 DMD modes of the field pressure 84
5.5 Cumulative magnitude and eigenvalues of the surface and field modes. 85
5.6 A schematic of the estimation methods. 87

5.7 DMD modes of the surface pressure. 88
5.8 Training curve for the four estimation methods. 93
5.9 Predicted vs. real values of b and d. 96

6.1 Schematic representation of the unforced system. 101
6.2 Schematic representation of the forced system. 102
6.3 The mesh and refinement regions. 104
6.4 Validation of the fluid-structure interaction simulaion. 106
6.5 The pressure field and pressure over time on the surface. 106
6.6 Mode magnitudes and eigenvalues for free oscillations. 112
6.7 Mode magnitudes for forced oscillations. 114
6.8 The actual and reconstructed modes for the free oscillation case. 119
6.9 Pressure reconstruction for the free oscillation case 120
6.10 Velocity reconstruction for the free oscillation case. 121
6.11 Pressure reconstruction for the forced case. 122
6.12 Velocity reconstruction for the forced case. 123

1 Strouhal number classification using coupled head kinematics. 130
2 Strouhal number classification using coupled tail kinematics. 131
3 Strouhal number classification using coupled head and tail kinematics. 132
4 Strouhal number classification using rigid foil kinematics. 133
5 Strouhal number classification using uncoupled head kinematics. 134
6 Strouhal number classification using uncoupled tail kinematics. 135

Chapter 1

Introduction

In recent decades, the emerging field of robotics has sought to use robots to perform tasks in environments that are unsuitable for humans. This has taken them to many extreme environments, from minefields and collapsed buildings to the surface of Mars. However, no terrestrial environment is as simultaneously inhospitable and extensive as the bodies of water covering the Earth. These bodies of water conceal immense quantities of raw materials beneath their surfaces, and are invaluable as routes for communication and transportation. Utilizing these environments requires labor, but humans are poorly suited to working underwater, with shallow water work requiring highly skilled specialists, and deeper work requiring submarine vehicles. However, current submarine vehicles have limited capabilities, and are greatly outperformed by biological swimmers.

The locomotion of fish and other aquatic swimmers has many desirable characteristics such as energy efficiency, agility, and stealth [1–4], which have inspired the design of many biomimetic robots. Designs for fish-like robots include those that are assemblages, rigid links actuated by motors imitating the motion of tails and fins [5], motor-driven flexible links [6, 7], elongated snake and eel like robots [8, 9], soft robots making use of dielectric elastomers, electroactive polymers or fluidic elastomer actuators [10–14], or robots with internal reaction wheels [15, 16]. In all such designs, the small size of the fish-like robots and the resulting constraints on actuation and power require that the morphology and control of the robots are designed to harness the fluid structure and fluid-structure interaction for efficient and agile motion. This motivates the two broad directions of my research: effective sensing of the flow structure, and controls that can exploit this knowledge for efficient autonomous locomotion.

The fields of controls and sensing are currently being revolutionized with parametric learning methods. Control algorithms were historically manually designed for a specific system, often leveraging knowledge of the system's physics, either implicitly using knowledge of the system to approximate appropriate gains, or explicitly using a physics model, such as Model Predictive Control and the Linear Quadratic Regulator. By contrast, the current state-of-the-art parametric controller architecture, Reinforcement Learning, is generally blind to the physics of the system, and can learn an optimal control using data alone. In the field of sensing a similar trend has ocurred, where approaches that rely on the physics of the fluid (such as potential flow models in [17, 18]) are often overtaken by parametric data driven approaches [19], particularly in more complex environments where simple flow models are not accurate, such as near flapping airfoils (a common abstraction for swimming robots) [20, 21]. In both sensing and control, merging the lack of required training and physical understanding generated from these physics-based approaches with the generality and effectiveness of parametric approaches is an important goal and a key theme of this work. However, as was perhaps best stated by Sutton [22], leveraging human knowledge to develop machine learning architectures for specific problems is often counterproductive, as the specially designed architectures often scale poorly with increasing computation compared to more generalist architectures, leading to the generalist architectures ultimately displacing the specialist ones as the available computing power inevitably increases. We incorporate this observation by integrating generalist machine learning architectures with abstracted knowledge of the physics in general frameworks which should be applicable to a broad range of problems and scale well with increasing availability of computing resources.

Utilizing the physics of a swimming robot to improve its controls is challenging because of the complexity of the physics. The fluid in which the robot swims is turbulent and infinite-dimensional, and the interaction with that fluid and the surfaces of the body is difficult to even simulate, and much more challenging (or perhaps impossible) to incorporate fully into a reduced-order model. This makes approaches such as model-based RL [23] intractable unless a neural network is used to fit the model, which sacrifices many of the advantages of physics-based approaches, such as human understandability and leveraging existing knowledge. Our contribution is the concept that even inaccurate models with vastly lower order than the modelled system can be leveraged to improve reinforcement learning, as long as the model captures the qualitative physics of the full system. We demonstrate this by modelling a swimming airfoil by a Chaplygin sleigh, a two-dimensional nonholonomic system that has been shown to have many parallels to the basic physics of swimming. We design a curriculum where reinforcement learning is first trained on the sleigh model to incorporate knowledge of the underlying physics of the system, and is then transferred to a simulation of the full system to complete training.

With the utility of the Chaplygin sleigh for training a rigid swimmer demonstrated, a flexible Chaplygin sleigh composed of two rigid links and a flexible joint is investigated to determine if it can model the qualitative physics of a flexible swimmer, which is a precursor to applying the curriculum technique to flexible swimmers. The resulting dynamics were surprisingly rich (curious readers can see [24] for an extensive discussion of these dynamics), but perhaps the most interesting feature to emerge from the dynamics is a pair of stable turning gaits, where a symmetric forcing can result in a 'bent' configuration of the flexible sleigh to either the left or right, which results in turning. This type of bistability can be useful for control and is typically introduced to a system intentionally through a geometric bistability, but our result is unique in finding this bistability to be present simply from the interaction between the sleigh and its substrate, or if this extends to realistic swimmers, between the tail of a swimmer and the water. This work adds to our theoretical understanding of turns by swimmers while motivating the idea that a curriculum using the flexible sleigh may be able to teach efficient turning behavior to a flexible swimmer.

In the sensing problem, such pre-training is generally not useful for convergence, so we require a different approach. As mentioned previously, simple potential flow models can be used for simple sensing or estimation problems by simply inverting the flow model (e.g. if you know what the distribution of pressures on a set of sensors are as a function of the position of a source (x, y), given a specific distribution of pressures, solve for (x, y)). Unlike the control problem where errors in the model are acceptable because the controller will be honed further on the actual system, in this case there is no equivalent final training step, so errors in the model equate directly to errors in the estimation or classification. For the turbulent, rotational flows around a swimmer and the various obstacles that can appear in realistic navigation scenarios, the only reasonably accurate models are simulations of the Navier-Stokes equations, which are not easily invertible. Instead, our work to incorporate the physics of swimming into sensing has two main directions. One is to change the dynamics of the swimmer itself to make the data it generates more amenable to black-box parametric classifiers. The other focuses on extracting features from the dynamic modes of a time series of pressure measurements from the surface of the body, which then enable parametric methods to sense obstacles or reconstruct the flow.

The dynamic modes, or Koopman modes, of a fluid (or complex system in general) provide an interesting means to explore the dynamics of a system, even when the governing equations are not known or are untractable. As a brief summary (more details are available in chapter 5), they emerge from the Koopman-von Neumann formulation of dynamics [25], whereby finite dimensional, generally nonlinear systems can be represented as a generally infinite dimensional linear system in a lifted space. For a typical dynamical system

$$x_{n+1} = F(x_n), \tag{1.1}$$

there also exists a linear operator such that

$$g(x_{n+1}) = \mathcal{K} g(x_n), \tag{1.2}$$

where g is a typically infinite-dimensional Hilbert space of functions in L^2, and $\mathcal{K}$ is the Koopman operator. Though working with the infinite dimensional operator is not practical, this formulation has an interesting interpretation that bridges between physics-based and data-driven approaches: the time evolution of observations (data) of a system is equivalent to the physics of its underlying states, in the limit as the available observations go to L^2. In practice, g is taken as a finite basis of functions, allowing a finite dimensional approximation of the Koopman operator K to be approximated from the data. This approximated operator is both data-driven and physics-based: it can be calculated with no knowledge of the governing equations as long as data is present, but it also has the useful properties of a linear system, such as eigenfunctions, that can give insight into the underlying dynamics. Because the majority of the dynamics is often governed by a small number of eigenfunctions, this also allows for a reduced representation of the dynamics which can be used to approximate its training data.

This ability to extract a reduced and meaningful representation of time series data originating from a dynamical system has a previously unexplored synergy with dense neural network (or in general parametric) classifiers and estimators. These networks take as input a vector of features, which are a compressed representation of the data in a latent space, where each element contains information about an abstracted quality of the data. The state-of-the-art for feature extraction from time series are parametric methods, such as convolutional neural networks and recurrent neural networks, and the resulting features are selected by the optimization process, and as a result are abstract and difficult or impossible for humans to interpret. Our contribution is taking the most important eigenvectors of the Koopman operator constructed on the data as the features, which results in a more interpretable estimation algorithm. This is applied to two problems. In one, a time series of pressure data from points on an airfoil (representing pressure sensors on a swimmer) with an upstream obstacle is collected, and the position of the obstacle is estimated. In the second, this is extended to using the surface pressure data to reconstruct the entire surrounding flowfield by estimating its dominant Koopman modes, taking advantage of the reduced representation of the broader flowfield that the Koopman modes offer to simplify the problem.

1.1 Summary of Chapters

- **Chapter 2:** Surrogate models that capture the primary physics of the system can be a useful starting point for training a Deep Reinforcement Learning (DRL) agent which is subsequently transferred to train with a higher fidelity simulation. We demonstrate the utility of such physics-informed reinforcement learning to train a policy that can enable velocity and path tracking for a planar swimming (fish-like) rigid Joukowski hydrofoil. This is done through a curriculum where the DRL agent is first trained to track limit cycles in a velocity space for a representative nonholonomic system, and then transferred to train on a small simulation data set of the swimmer. The results show the utility of physics-informed reinforcement learning for the control of fish-like swimming robots.

- **Chapter 3:** This chapter finds that two bodies pinned together with a linear rotational spring can exhibit multistable behavior with the introduction of a nonholonomic constraint. Multistable fixed points of the unforced and undamped system are found to correspond to multistable limit cycles with the introduction of damping and periodic forcing, some of which result in fast net turning. This finding has potential implications in understanding the sharp turns executed by biological swimmers, and could be exploited to perform efficient turns in low degree of actuation robots.

- **Chapter 4:** In this chapter we show that the kinematics of a body with a passive tail encode information about the ambient flow, which can be deciphered through machine learning. We demonstrate this with experimental data of the angular velocity of a hydrofoil with a passive tail that lies in the wake generated by an upstream oscillating body. Using convolutional neural networks, we show that with the kinematic data from the downstream body with a tail, the wakes can be better classified than in the case of a body without a tail. This superior sensing ability exists for a body with a tail, even if only the kinematics of the main body are used as input for the machine learning. This shows that beyond generating ‘additional inputs’, passive tails modulate the response of the main body in manner that is useful for hydrodynamic sensing. These findings have clear application for improving the sensing abilities of bioinspired swimming robots.

- **Chapter 5:** Besides the necessary sensing hardware, a more important aspect of sensing is related to the algorithms needed to extract the relevant information about the flow. This paper advances a framework for such an algorithm using the setting of a pitching hydrofoil in the wake of a thin plate (obstacle). Using time series pressure measurements on the surface of the hydrofoil and the angular velocity of the hydrofoil, a Koopman operator is constructed that propagates the time series forward in time. Multiple approaches are used to extract dynamic information from the Koopman operator to estimate the plate position, and are benchmarked against a state-of-the-art convolutional neural network (CNN) applied directly to the time series. We find that using the Koopman operator for feature extraction improves the estimation accuracy compared to the CNN for the same purpose, enabling ‘blind’ sensing using the lateral line.

- **Chapter 6:** In this chapter, we demonstrate that it is possible to identify certain modes of fluid flow and then reconstruct the entire flow field from these modes. We use Dynamic Mode Decomposition (DMD) to parameterize complex, dynamic features across the entire flow field. We then leverage deep neural networks to infer the DMD modes of the pressure and velocity fields within a large, unsteady flow domain, employing solely a time series of pressure measurements collected on the surface of an immersed obstacle. Our methodology is successfully demonstrated to diverse fluid-structure interaction scenarios, including cases with both free oscillations in the wake of a cylinder and forced oscillations of tandem cylinders, demonstrating its versatility and robustness.

Chapter 2

Curriculum Reinforcement Learning for Path Tracking

This chapter is adapted from a paper in *Scientific Reports*:

C. Rodwell and P. Tallapragada, “Physics-informed reinforcement learning for motion control of a fish-like swimming robot,” Scientific Reports, vol. 13, no. 1, p. 10754, 2023.

2.1 Introduction

Motion control of robots, either physical or simulated with high fidelity, moving in unstructured environments with complex or unmodelled governing physics has traditionally presented many challenges. Low-fidelity models using simplified formulas for drag and lift forces lead to models that are amenable for control, but do not capture key physics that can play an especially important role in the swimming of small-scale robots with limited actuation. Deep Reinforcement Learning (DRL) holds considerable promise for motion control of robots with complex dynamics, such as swimming robots. Reinforcement learning can be particularly useful when precise governing models are absent [26–28].

Reinforcement learning requires the acquisition of large amounts of data from the robotic system in the form of states, control actions, and the resulting state updates. Such data can be acquired either through experiments, field tests, or simulations of the robot and its interaction with the environment. In the case of mobile robots, experiments and field tests could prove to be very expensive or unsafe to the robot. As a result, simulations play an important role in reinforcement learning for robotics [29–31]. Simulations can in theory generate large amounts of data at low cost, exploring a large subset of the state space that is challenging to explore in experiments. This is true in the case of robotic manipulators, inverted pendulums and other toy systems whose physics is well understood and can be efficiently and quickly computed. However, where the governing physics is complex or the state space is high-dimensional with possibly discontinuous dynamics, high fidelity simulators are often impractical and significant model reduction or the use of surrogate models is essential [31].

The control of swimming robots is an important example where the governing physics is complex enough that reinforcement learning should be a tool of choice for control, but the dynamics of the fluid-structure interaction can be challenging to simulate. While reinforcement learning is being widely applied in many areas of robotics [27, 32] and fluid control [33, 34], swimming robots have for the most part not received much attention. A few notable recent exceptions employed reinforcement learning for tasks such as predicting efficient schooling configurations for pairs of swimmers [35] or larger schools [36], gait generation [37], or efficient start and escape patterns [38] . However, important problems related to mobile robotics such as station keeping, velocity tracking under disturbances, or path tracking have not been addressed in the area of fish-like swimming robots due to the computational challenges of running a large number of high-fidelity simulations.

We address this challenge with curriculum learning [39, 40] and transfer learning [41], in the context of a swimming robot. It has been observed that when applying reinforcement learning using the Deterministic Policy Gradient (DPG) algorithm for complex tasks, much of computational time is spent going from the initialized random policy to an intermediate policy that, while sub-optimal, has qualitative similarities to the optimal policy [39–41]. Going from this policy to the optimal policy is often very fast. The intermediate policy does not have to be trained using high-fidelity state data, and can instead be cheaply trained using lower-fidelity simulations or models, before being transferred to a higher-fidelity environment to complete the final steps of training. Efficient multi-model training has been demonstrated in fluids for flow control [42], and here we extend it to the swimming problem. Here we utilize knowledge of the physics of fish-like swimming, specifically two important qualitative features of swimming, to estimate these intermediate policies, from which finding an optimal policy is faster. The first feature is that fish-like propulsion is enabled by periodic 'tail beating' (for carangiform fish) or body undulations (anguiliform fish) [4,43], at least in steady state motion. In a suitable reduced velocity space this feature can be modelled by limit cycles created by periodic forcing. A second surprising feature is that swimming by a hydrofoil, which resembles a cross-section of a fish-like body, can be approximately modelled as motion of a nonholonomically constrained system. This surprising feature arises because the Kutta condition, which creates vorticity at sharp corners of a body moving in an otherwise approximately inviscid fluid, acts as a nonholonomic constraint on a swimming hydrofoil [44,45].

The swimming robot in this paper is modeled as a free-swimming Joukowski hydrofoil, propelled and steered by an internal reaction wheel [15,16]. A particularly simple surrogate model that emulates the dynamics of such a swimmer is a nonholonomic system known as the Chaplygin sleigh. Periodic torque on the Chaplygin sleigh produces 'figure-8' limit cycles in the velocity space, which are similar in structure to the observed limit cycles of swimmers. Analytical approximations for these limit cycles as a function of the forcing frequency and amplitude of the torque, as well as the inverse dynamics problem of finding the periodic torque to generate a limit cycle, have been shown in [46]. Torques for steering the Chaplygin sleigh [16,46] and path tracking for the Chaplygin sleigh using a vector pursuit method were demonstrated in [47]. Since we are interested in transfer learning using the low-dimensional model of the Chaplygin sleigh, we revisit the problem of simultaneous velocity tracking and steering using a reinforcement learning framework. A DPG agent [48,49] is trained to generate the action (torque) to steer a Chaplygin sleigh which has parameters that have been fit to match the dynamics of the swimming Joukowski foil. The agent is trained on the Chaplygin sleigh model to track a limit cycle in the velocity space at a specified translational velocity. This DPG agent of the Chaplygin sleigh is then transferred to a fluid simulation for training to steer the hydrofoil and track a speed. This second training requires fewer simulations to fine tune the DPG agent's precision in tracking a reference velocity. This step-by-step curriculum reinforcement learning circumvents the problem of high computational time to simulate the physics of the system [39,40] and allows a way to *imprint qualitative physics* into a DPG agent.

This paper sets forth a framework for using physics-informed surrogate models to train a DPG agent which is then subsequently trained using data generated from fast simulations of fluid-robot interaction. A very high fidelity computational method need not be used in the second step for two reasons: going from an intermediate sub-optimal policy to an optimal policy can be slowed down significantly, but more importantly the computation of the fluid-robot interaction is intended to be another intermediate step before actual experiments. The paper is organized as follows: in section 2.2 a short review of nonholonomic constraints with particular reference to the Chaplygin sleigh is provided and the Kutta condition on a Joukowski foil is shown to be formally similar to this constraint. In section 2.3 limit cycles are shown to exist via simulations in a reduced velocity space for the Chaplygin sleigh and a hydrofoil excited by a periodic torque. Here we make use of a panel method to simulate the motion of the hydrofoil. In section 2.4 we select two sets of Chaplygin sleigh parameters that model the swimming hydrofoil at different translational velocities. In section 2.5 we describe the reinforcement learning framework with a curriculum to enable path tracking by a Chaplygin sleigh and the transfer of this skill to the simulated swimming hydrofoil.

2.2 Nonholonomic Constraints - Chaplygin Sleigh and the Joukowski Foil in an Inviscid Fluid

2.2.1 Chaplygin sleigh

The Chaplygin sleigh [50] or cart, shown in Fig. 2.1, has a knife edge or a small inertia-less wheel at the rear at point P and is supported on a single castor or wheel at the front that allows motion in any direction. The sleigh is also assumed to have an internal reaction wheel whose angular acceleration can apply a torque τ to the sleigh. The configuration manifold of the physical system is $Q = SE(2) \times S^1$. Because the rotor angle and angular velocity are not coupled with the rest of the system, we eliminate the rotor coordinate except for the torque τ, reducing the system to $Q = SE(2)$. This is parameterized locally by $q = (x, y, \theta)$, where (x, y) denote the position of the sleigh center of mass and θ denotes its fixed frame angle relative to the horizontal. The generalized velocities are $\dot{q} = (\dot{x}, \dot{y}, \dot{\theta})$. The tangent space to Q at q is denoted by $T_qQ \cong \mathbb{R}^3$ and spanned by combinations of the generalized velocities; the standard basis being $\{[1, 0, 0], [0, 1, 0], [0, 0, 1]\}$. These three basis vectors are respectively translations $\dot{x}$, $\dot{y}$ and rotation $\dot{\theta}$. The spatial frame is denoted by $\mathcal{F}_S$ with axes $X-Y$ and the body frame, which is collocated at the mass center C and rotated by the yaw angle θ with respect to the spatial frame, is denoted by $\mathcal{F}_B$ with axes X_b-Y_b. The velocity $(\dot{x}, \dot{y})$ in the spatial frame transforms to velocity (u, v) in the body frame as $[u\ v]^\intercal = \mathbf{R}(\theta) \cdot [\dot{x}\ \dot{y}]^\intercal$, where $\mathbf{R}(\theta)$ is the rotation matrix. We assume that the rear wheel at P prevents slipping in the transverse (Y_b) direction but rolls freely in the longitudinal direction along (X_b). While $dim(T_qQ) = 3$, the

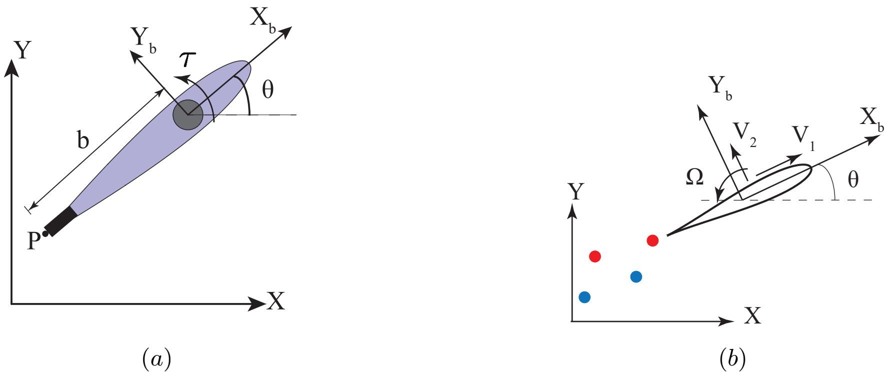


Figure 2.1: (a) A Chaplygin sleigh shaped as a Joukowski foil with a no slip constraint at P in the transverse (Y_b) direction. The internal reaction wheel is shown by the grey circle. (b) A Joukowski foil with singular distributions of vorticity (red circles corresponding to positive (counterclockwise) vorticity, and blue circles corresponding to negative (clockwise) vorticity) in an otherwise inviscid flow.

velocity constraint at point P is given by

$$-\sin\theta\dot{x}+\cos\theta\dot{y}-b\dot{\theta}=0 \tag{2.1}$$

or in the body frame $v-b\dot{\theta}=0$. In terms of the standard basis for T_qQ, the velocity of sleigh is restricted to lie in the subspace $W=span\{[\cos\theta,\sin\theta,0]^\intercal,[-b\sin\theta,b\cos\theta,1]^\intercal\}$, with the complementary subspace being defined by (2.1), $W^\perp=span\{-\sin\theta,\cos\theta,-b\}$. Physically, this means that the allowable motions are such that the sleigh can translate along its longitudinal (X_b) direction and have no spin velocity i.e. in the fixed frame the velocity of the center of the sleigh is $\cos\theta\dot{x}+\sin\theta\dot{y}$, or if the spin angular velocity of the sleigh is $\dot{\theta}$ and the constraint point does not translate, then the center of the sleigh can only translate with ($\dot{x}=-b\sin\theta\dot{\theta},\dot{y}=b\cos\theta\dot{\theta}$). The distribution (a smooth assignment of a subspace of the tangent space at each $q\in Q$) $D(q)=\{w\in W(q)\subset T_qQ\}$ is nonholonomic if and only if $D(q)$ is closed under the Jacobi-Lie brackets of the vector fields in $D(q)$. The Lie-bracket of two vector fields w_1 and w_2 is formally defined as $[w_1,w_2]=(\nabla w_2)\cdot w_1-(\nabla w_1)w_2$. Setting the vectors w_1 and w_2 as the basis vectors defining the span of W respectively, their Lie bracket yields $[w_1,w_2]=[-\sin\theta,\cos\theta,0]^\intercal\notin W$ showing that the constraint (2.1) is nonholonomic, see [45] for further details.

2.2.2 Nonholonomic constraints and swimming in an inviscid fluid

Nonholonomic constraints on the locomotion of a body in a fluid are easiest to realize if one considers the motion of a body with corners in an inviscid fluid. One such common example of a body relevant to both flight and fish-like swimming is the motion of a Joukowski foil whose geometry is described by mapping its boundary from a circle of radius r_c centered at the origin in the mapped plane through the Joukowski transformation

$$z = F(\zeta) = \zeta + \zeta_c + \frac{a^2}{\zeta + \zeta_c}, \tag{2.2}$$

where $\zeta_c \in \mathbb{C}$ and $a \in \mathbb{R}$ are geometric parameters. We refer to the plane of the foil's motion as the foil plane and the plane of the circle's motion as the circle plane. For the symmetrical shape of the foil shown in Fig. 2.1 (b), $\zeta_c \in \mathbb{R}$. The pre-image of the sharp trailing edge of the foil is given by $\zeta_t = a - \zeta_c$. We assume the fluid could contain singular distributions of vorticity in the form of N point vortices as shown in Fig. 2.1 (b). The motion of the fluid is governed by a linear superposition of potential functions. Following Milne-Thomson [51], the complex potential $W(z) = w(\zeta)$ describing the velocity of the fluid in a body frame of reference $(X_b - Y_b)$ (see Fig. 2.1(b)) may be decomposed in terms of its dependence on the translation of the foil, the rotation of the foil, and each of the N point vortices in the form

$$w(\zeta) = W(z) = V_1 w_1(\zeta) + V_2 w_2(\zeta) + \Omega w_3(\zeta) + \sum_{n=1}^{N} w_v^n(\zeta).$$

Here w_1, w_2 and w_3 are the rigid-body (Kirchoff) potential functions due to the translation of the foil in the X_b and Y_b directions and rotation about the out-of-plane Z axis, respectively. The potential function $w_v^n(\zeta)$ due to the nth point vortex with circulation Γ_n located at ζ_n outside a circular cylinder can be constructed according to the *Milne-Thomson circle theorem* [51] in terms of an image vortex of circulation $-\Gamma_n$ located inside the cylinder at $r_c^2/\bar{\zeta}_n$. Thus $w_v^n(\zeta) = \frac{\Gamma_n}{2\pi\imath}\left(\log(\zeta - \zeta_n) - \log\left(\zeta - \frac{r_c^2}{\bar{\zeta}_n}\right)\right)$.

The image vortex inside the cylinder introduces a net circulation around the cylinder, consistent with Kelvin's circulation theorem. This development of net circulation around the foil introduces a lift force on the foil and is essential to its propulsion. The complex velocity of the fluid in the foil plane in a body fixed frame is related to the complex velocity of the fluid in the circle plane by the equation,

$$\frac{dW}{dz} = \frac{dw}{dz}\frac{dz}{d\zeta} = \frac{dw}{dz}\frac{1}{F'(\zeta)} \tag{2.3}$$

where $F'(\zeta) = 1 - \frac{a^2}{(\zeta+\zeta_c)^2}$. Since the fluid has been modeled to be inviscid, the boundary conditions on the body of the foil allow the fluid to slip along the surface. An additional constraint on the velocity of the fluid is necessitated by the geometry of the foil. The pre-image of the trailing edge of the foil is a singularity of the Joukowski transformation, i.e., the derivative $F'(\zeta_t) = 0$. It can be seen from equation 2.3 that the complex velocity of the fluid in the circle plane has to be zero to ensure that the velocity of the fluid at the trailing edge of the foil in the foil plane does not become undefined; this is the Kutta condition. The Kutta condition requires that at the pre-image ζ_t of the trailing edge,

$$\left.\frac{dw}{d\zeta}\right|_{\zeta=\zeta_t} = V_1\left.\frac{dw_1}{d\zeta}\right|_{\zeta=\zeta_t} + V_2\left.\frac{dw_2}{d\zeta}\right|_{\zeta=\zeta_t} + \Omega\left.\frac{dw_3}{d\zeta}\right|_{\zeta=\zeta_t} + \sum_{1}^{n=N}\left.\frac{dw_v^n}{d\zeta}\right|_{\zeta=\zeta_t} = 0.$$

A formal calculation shows that $\left.\frac{dw_1}{d\zeta}\right|_{\zeta=\zeta_t} = 0$ and $\left.\frac{dw_2}{d\zeta}\right|_{\zeta=\zeta_t} = -2$. The term $\frac{1}{2}\left.\frac{dw_3}{d\zeta}\right|_{\zeta=\zeta_t}$, subsequently denoted by $-b$, is a constant that is determined by the numerical values of the parameters a and ζ_c. Denoting the velocity of the fluid at the trailing edge due to the distribution of point vortices $u_v = \frac{1}{2}\sum_1^{n=N}\left.\frac{dw_v^n}{d\zeta}\right|_{\zeta=\zeta_t}$, the Kutta condition can be re-written as

$$-\dot{x}\sin\theta + \dot{y}\cos\theta - b\dot{\theta} = u_v. \tag{2.4}$$

Equation (2.4) constrains the velocity of the foil and it is an *affine-nonholonomic* constraint. This condition constrains the velocity in the system's phase space to an affine distribution A of the form $A(q) = \{w - w_0 \in W_e(q) | w_0 = [-u_v\sin\theta, u_v\cos\theta, 0]^\intercal\}$, where $W_e(q) = span\{W, V^1, ..., V^{2N}\}$ is the subspace containing the allowable vector fields w_1 and w_2 associated with the motion of the foil and the vector field $V^1, ..., V^{2N}$ being the velocities of each of the N point vortices. This is an affine-distribution not simply on $SE(2)$ but on the Cartesian product of $SE(2)$ with configuration manifold $\mathbb{R}^{2N}$ for the vortices. The vector fields $V^1, ..., V^{2N}$ represent translational velocities of the vortices, potentially dependent on (x, y, θ), that are compatible with the constraint when the position and orientation of the foil are fixed.

2.3 Periodic Forcing and Limit Cycles in Reduced Velocity Space

2.3.1 Limit Cycles in Reduced Velocity Space of the Chaplygin Sleigh

The equations of motion of the Chaplygin sleigh can be calculated in a straightforward manner. Here we will assume that the Chaplygin sleigh experiences viscous resistance as it moves on the ground and that it is actuated by periodic torque generated by the periodic oscillation of the reaction wheel. The Lagrangian of the system is $\mathcal{L} = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}I_c\dot{\theta}^2$ where m is the mass of the sleigh, I is its moment of inertia, and x and y are the coordinates of the center of mass. Assuming a viscous resistive force to motion described by the Rayleigh dissipation function $\mathcal{R} = \frac{1}{2}(c_u(\dot{x}\cos\theta + \dot{y}\sin\theta)^2 + c_\omega\dot{\theta}^2)$, where c_u and c_ω are viscous damping coefficients for the translational and rotational velocity respectively. The Euler-Lagrange equations are

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{q}^i}\right) - \frac{\partial\mathcal{L}}{\partial q^i} = B_{ij}\lambda_j - \frac{\partial\mathcal{R}}{\partial\dot{q}^i} + \tau_i(t), \tag{2.5}$$

where $B = [-\sin\theta, \cos\theta, -b]$ (obtained from $W^\perp$, the complementary space of W), λ_j is the Lagrange multiplier for each j constraint and τ_i are any external forces or torques acting on sleigh. Here i varies from 1 to 3 and $j = 1$, so only τ_1 is nonzero and consequently it is denoted henceforth as just τ. The equations can be transformed to a body frame using $[u\ v]^\intercal = \mathbf{R}\cdot[\dot{x}\ \dot{y}]^\intercal$ to obtain the dimensionless *reduced velocity* equations

$$\dot{u} = b\omega^2 - \frac{c_u}{m}u \tag{2.6}$$

$$\dot{\omega} = \frac{\tau - mb\omega u - c_\omega\omega}{I + mb^2}, \tag{2.7}$$

where $u = \dot{x}\cos\theta + \dot{y}\sin\theta$ is the translational velocity at the constraint. Due to the nonholonomic constraint, the evolution of the velocities is governed by two equations instead of three. The evolution of the configuration variables is given by

$$\dot{\theta} = \omega \tag{2.8}$$

$$\dot{x} = u\cos\theta - \omega b\sin\theta \tag{2.9}$$

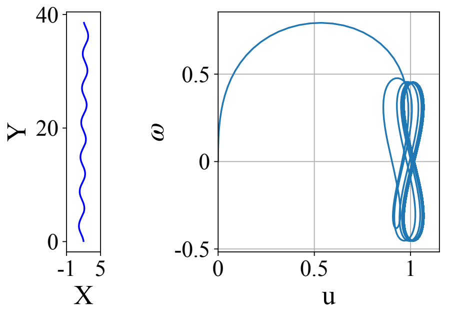
$$\dot{y} = u\sin\theta + \omega b\cos\theta. \tag{2.10}$$

When the torque due to the reaction wheel is periodic, $\tau = \tau_0 \sin\Omega t$, a limit cycle exists in the reduced velocity space [52] and the sleigh moves along a serpentine path with its time averaged path converging to a straight line illustrated by a sample result in Fig. 2.2(a).

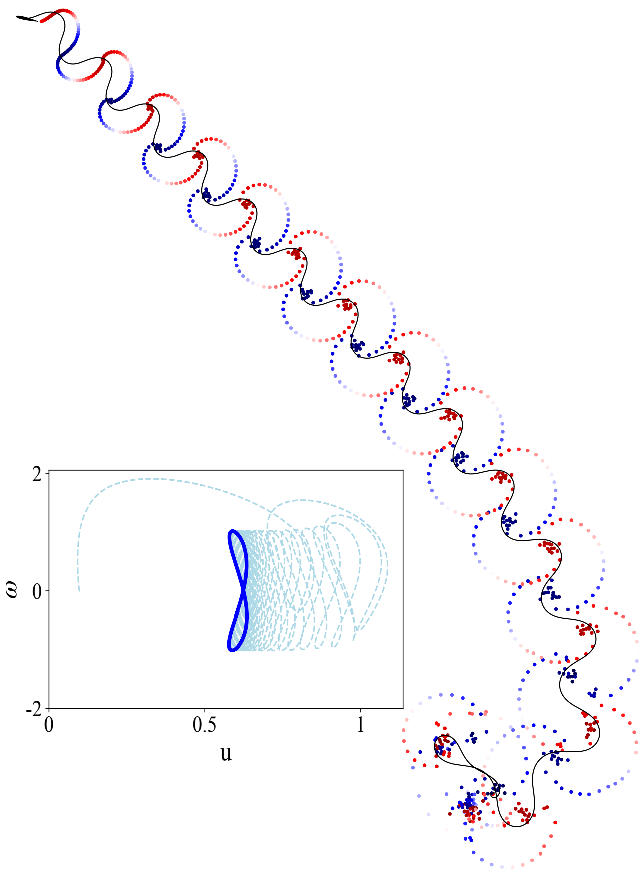
2.3.2 Limit Cycles in Reduced Velocity Space of a Hydrofoil

We consider a hydrofoil modeled as a NACA 0018 symmetrical airfoil that contains an internal reaction wheel with a moment of inertia I_r and angular velocity Ω_r. The oscillatory motion of the reaction wheel generates a periodic torque on the hydrofoil given by $\tau = -I_r\dot{\Omega}_r$. We simulate its motion due to a periodic torque τ using a vortex panel method. Panel methods are a form of computational fluid dynamics (CFD) that relies entirely on potential flow theory with point and line vortices. In these methods boundary (structure) surfaces are decomposed into discrete *panels* with source and vortex distributions on each panel such that flow does not pass through the surface. While the lack of viscous effects can reduce simulation fidelity compared to modern meshed Navier-Stokes solvers, panel methods have lower computational cost and easily incorporate body movement due to the lack of meshing. For these reasons, panel methods continue to be the preferred simulation tool for high Reynolds number fluid interaction problems [53,54], where the flow is largely dominated by inertial effects which are easily captured by the panel method, and the neglected viscous effects are comparatively insignificant. This includes the dynamics of swimming of large and/or fast swimmers, where experimentally validated panel methods have long been used from quantifying the efficiency of flipper shapes [55] or flapping airfoil kinematics [56] and explaining the flow dynamics of full fish models [57]. Though the decreasing cost of computing power has lead to increasing use of meshed finite volume techniques in swimming problems, new panel methods are still in development [58], and they are still widely used for swimming problems, including anguilliform [59] and cetacean [60] swimming.

In a vortex panel method, each of the N panels has a source distribution of strength σ_i,



(a)



(b)

Figure 2.2: (a) (left) A sample serpentine trajectory for the Chaplygin sleigh where the mean converges to a straight line. (right) In the reduced velocity space the trajectory converges to a 'figure-8' limit cycle in (u, ω). (b) A sample trajectory of the simulated swimmer starting from rest when forced by a periodic torque $\tau = A \sin \omega t$. The inset figure shows convergence to a limit cycle in the reduced velocity, (u, ω), space that is similar to the that of the Chaplygin sleigh, indicating similar underlying dynamics. The velocity is scaled into body lengths per second ([BL/s]). The swimmer moves along a serpentine path (in black) with the average path converging to a straight line.

which varies between panels, and a vortex distribution of strength γ which is constant over the body. The Neumann boundary condition is applied, which stipulates that no flow passes through a panel midpoint, or $\langle u_i, \eta_i \rangle = 0 \; \forall \, i \in \{1, \ldots, N\}$, where u_i is the flow velocity vector at the center of panel i relative to the body, and η_i is the surface normal unit vector at the same control point. The Kutta condition enforces the condition that static pressure p is continuous at two panel midpoints adjacent to the tail, i.e., $p_1 = p_N$. To enforce this condition, vortex shedding occurs at the tail, by calculating the change in circulation about the body at every time step and applying an opposite circulation to a *wake panel* at the tail, which is then shed as a point vortex of equal circulation at its center. The structure of the system allows γ to be solved independently, and because the flow velocity due to a specific source or vortex panel is linear in its unknown strength, the N unknown σ_i values can be found by solving the linear system of N equations arising from the Neumann boundary condition. With γ and the σ_i found, the flow field is fully determined, and the pressure distribution around the body can be computed from the unsteady Bernoulli equation. This is implemented by calculating the velocity potential at each point on the body relative to the leading edge via a path integral of fluid velocity along the body, neglecting circulation. The time derivatives of these potentials are calculated by finite differences. The moving potential reference point results in a value that varies with time but is equal across the body added to all of the pressures, but because of body closure, this value has no effect on the calculated forces or moments. From the pressures, the resultant torques and forces acting on the body can be computed. An additional linear dissipation force is applied on each body degree of freedom linearly proportionally to the velocity, which prevents drag-free gliding that would otherwise be possible without skin friction. A snapshot from a sample simulation of the motion of the vortex wake due to a periodic torque on a Joukowski foil is shown in Fig. 2.2b. The panel code was validated in steady flow by comparing computed lift coefficients to known experimental values for a range of angles of attack [61]. It was also validated in unsteady flow [46] where it was compared to a rotor-driven swimming hydrofoil experiment, and it was found that the swimming model results in similar trajectories to the experimental system.

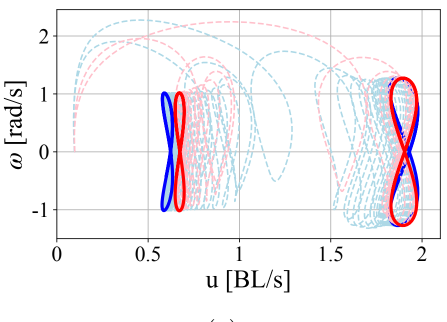
2.4 Parameter Estimation for the Surrogate Model

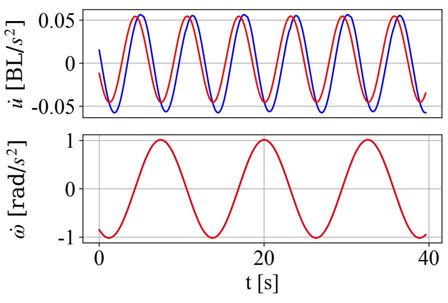
The similarity of the limit cycles of the Chaplygin sleigh and the hydrofoil and their trajectories in the plane, in response to periodic control input (torque) together with the similar nonholonomic constraints on the both the systems, motivates the use of the Chaplygin sleigh as a surrogate model for the swimming foil. The limit cycles of the Chaplygin sleigh in the reduced velocity space depend on the parameters $p = (m, I, b, c_u, c_\omega)$. For an accurate surrogate model, these parameters need to be chosen such that resultant limit cycles due to periodic torques are nearly the same (with the same mean value and amplitudes) as those of a hydrofoil. This is accomplished by first gathering data from simulating the motion of the hydrofoil, with each simulation being 100 seconds long with data acquired at time increments $dt = 0.1$ with forcing $\tau = A \sin t$. Two different models are fit, one where $A = 1.0$ and another where $A = 1.2$, and the sensitivity of the results to the forcing amplitude used to fit the surrogate model is explored. The surrogate model fit from simulations using the lower forcing amplitude resulting in a smaller $u_0 = 0.606$ (of the swimmer) will be referred to as the “low speed surrogate model", while the latter model fit from simulations using the higher forcing amplitude with a higher $u_0 = 1.90$ will be referred to as the “high speed surrogate model". For each time step, the current state $s = (u, \omega)$, action τ, and next state $s^{+1} = (u^{+1}, \omega^{+1})$ are stored in buffer B. An optimization routine is then performed to minimize the least squares error

$$p_0 = \arg\min_p \sum_B \left(\left(u + \left(b\omega^2 - \frac{c}{m} u \right) dt - u^{+1} \right)^2 + \left(\omega + \left(\frac{\tau - mb\omega u}{I + mb^2} \right) dt - \omega^{+1} \right)^2 \right). \tag{2.11}$$

Figure 2.3(a) shows the trajectories of the swimmer (blue) and the surrogate model (red) obtained from (2.11) for the low speed model (on the left) and the high speed model (on the right). In both the cases the surrogate models' trajectories in the reduced velocity space converge to limit cycles that are qualitatively similar to those of the swimmers. In the pair of limit cycles on the left, the apparent mismatch of the limit cycles is largely due to an error (about 10%) in the mean value of the speed u_0. This is evident in Fig. 2.3(b) which shows the evolution (in red) of $\dot{u}$ and $\dot{\omega}$ for the low speed surrogate model. The evolution of these state variables for the hydrofoil are also shown in Fig. 2.3(b) in blue. The error in the angular velocity (and its derivative) is negligible between the hydrofoil and its nonholonomic surrogate model and the error in $\dot{u}$ is small, with a value of 0.021 for the low velocity surrogate, which results in an error in mean translational velocity u_0 of 0.066. Error in u_0 for the higher velocity surrogate model is significantly lower, at 3.11×10^{-5}.

Here we note that the surrogate modeling fits the parameters of a three degree of freedom rigid body model to approximate the complex high-dimensional interaction with the vortex wake of a hydrofoil. Therefore, the mapping from the dynamics of a swimmer to that of a Chaplygin sleigh cannot be unique. The same surrogate model can be mapped non-uniquely to the dynamics of different swimmers. For example, for the low speed surrogate model, the dimensionless parameter values are found to be $m = 0.93$, $I = 0.98$, $b = 0.068$, $c_u = 0.041$, and $c_\omega = 0.0043$. For the high speed surrogate model, the corresponding parameters are $m = 0.36$, $I = 0.93$, $b = 0.17$, $c_u = 0.026$, and $c_\omega = 0.058$. Moreover, the error between the dynamics of the surrogate model and the true swimmer it models, increases with changing the velocity or forcing. We use both of these surrogate





(a) (b)

Figure 2.3: (a) Limit cycles of the surrogate Chaplygin sleigh (red) and the swimmer (blue) for the same periodic forcing, demonstrating convergence to similar limit cycle trajectories in the reduced velocity space. Two sets of limit cycles are shown, one due to applied periodic torque $\tau = \sin t$ (limit cycles on the left) and the other due to $\tau = 1.2 \sin t$ (limit cycle on the right). (b) The vector field of the governing equations for the surrogate Chaplygin sleigh (red) and the swimmer (blue) at the lower velocity. The units shown are for the simulated swimmer, the Chaplygin sleigh states are dimensionless.

models in the subsequent curriculum learning to demonstrate that the qualitative similarity between the physics (mainly that efficient motion lies on an invariant manifold) is more important to the training than quantitative similarity, as the two surrogate models have quantitatively dissimilar parameters.

2.5 Reinforcement Learning

Control for fish robots is challenging due to the high complexity of the fluid-body interaction and the need for periodic motion. The most popular current approach is based on the central pattern generator (CPG), a neural assembly found in vertebrates that has periodic outputs which provide the rhythm for periodic locomotion [62]. Artificial attempts to reproduce this functionality often determine the deflection of each actuator as an oscillator, where parameters determine the amplitude, frequency, and relative phases of the actuators [63]. While this does typically result in swimming behavior, the exact oscillator parameters are typically chosen heuristically, and though formal parameter optimization can result in large improvements in swimming speed, they must be performed by algorithms such as particle-swarm optimization that select parameters to optimize the reward over entire trajectories in an open-loop fashion [64]. Reinforcement learning has emerged as a more efficient way to optimize controller parameters than pure trajectory optimization because it can utilize knowledge about intermediate states within the trajectory to improve the controller (policy), provided that the control action is only a function of the states and controller parameters. Here, we teach the controller to perform oscillations based on feedback using a curriculum, instead of encoding the oscillator directly into the controller architecture. This allows parameter optimization to be performed on comparatively few trajectories of the simulated swimmer.

We apply a Deterministic Policy Gradient (DPG) algorithm with curriculum learning to control the foil in the panel method simulated swimming environment to track a reference path at a reference velocity. This path tracking algorithm is decomposed into two parts: a pure pursuit algorithm [65,66] that determines a target turning angle given the path geometry, and a DPG-trained actor that performs the specified turn at the desired velocity.

The pure pursuit algorithm is the simpler of the two: given a sequential list of points (X, Y) defining a path, a list of vectors $\bar{\mathbf{r}}$ spanning from the center of the pursuer to each point can be constructed, and a target point (X_i, Y_i) with vector $\bar{\mathbf{r_i}}$ can be selected with i initialized at 0. At every time interval, all vectors $\bar{\mathbf{r}}$ are reconstructed based on the current pursuer position, and if $|\bar{\mathbf{r_i}}| < d$, where d is a constant which specifies the sight horizon, then i is increased iteratively until the inequality becomes false. The vector $\bar{\mathbf{r_i}}$ is then taken as the pursuit vector, and the angle of $\bar{\mathbf{r_i}}$ in the fixed frame is recorded as θ_{target}, and in the local frame the error between the current and desired heading is defined as $\theta_e = \theta_{target} - \theta$. In the limit as $(d, \theta_e) \to 0$ and for a continuous path (X, Y), the tracking error goes to zero. Such a pursuit algorithm was implemented for path tracking by a Chaplygin sleigh in Fedonyuk [47]. The trajectory $(x(t), y(t), \theta(t))$, of the Chaplygin sleigh in the plane when the torque is periodic is serpentine, but the time averaged trajectory converges to a straight line. If the sight horizon d is too small, i.e. $(\frac{d}{u_r} \ll \frac{1}{\Omega})$ for a reference velocity u_r and stroke frequency Ω, the pursuit vector $\bar{\mathbf{r_i}}$ and θ_{target} oscillate rapidly to direct the body back to the path, which interferes with the agent's derived stroke frequency. In contrast, a control method chasing a more distant point tends to smooth the small-scale features of the path and minimize oscillations of the trajectory, but where the desired path is sharply curved, track a chord cutting through the required path. Here we take $d = 10$ body lengths, such that $d \sim \frac{u_r}{\Omega}$ for speeds considered, which we find to be appropriately small to capture the details of the paths considered, while not small enough to interfere significantly with the frequency of oscillations.

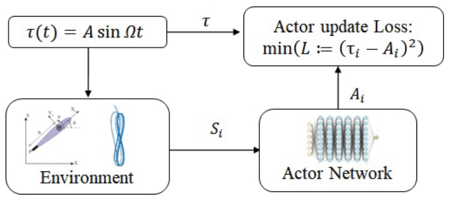
2.5.1 Curriculum Learning

The curriculum learning has three training steps: a supervised step of pre-training the policy to model a given control on the surrogate Chaplygin sleigh, a DPG training step to optimally control the surrogate Chaplygin sleigh model, and a third step transferring the same trained model into the fluid-robot simulation environment for further training.

The supervised step of pre-training the sleigh to model a given control on the Chaplygin sleigh imprints into an actor a known control algorithm as an initial policy from which further exploration can enable the actor to learn policies for other reward functions (control goals), as will be seen by results in Fig. 2.5 in the following section. The first supervised step is inspired by previous work to control the surrogate Chaplygin sleigh, such as [47] in which velocity tracking with a purely sinusoidal input and in [67] where steering with a purely proportional control was investigated. Initializing the actor to perform a superposition of these control methods with arbitrarily chosen constants is a better starting point than a fully random actor. The initial control function is selected as

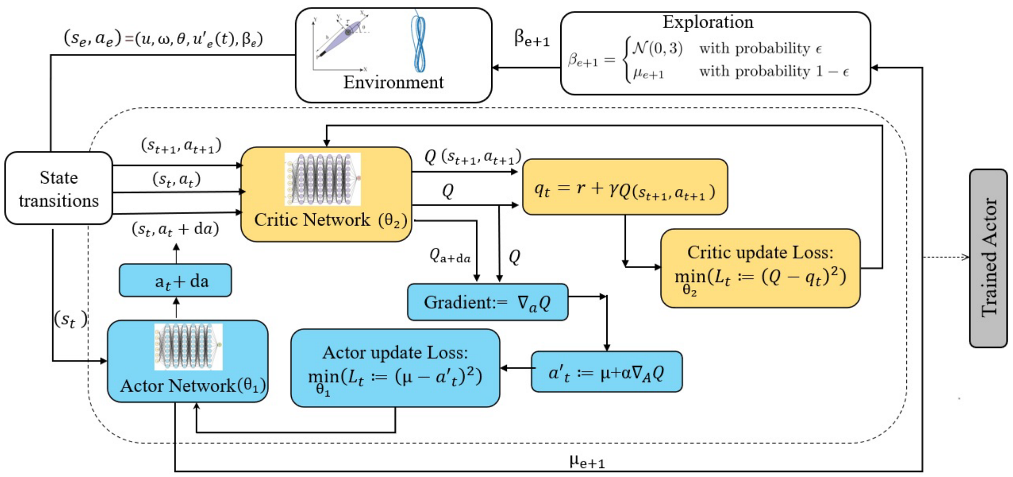
$$\tau = 0.3 \sin t - 0.05\theta_e, \tag{2.12}$$

which, with constant θ_{target} converges to a limit cycle and $\bar{\theta_e} \to 0$ and $u_0 \to 0.32$ for any initial conditions $(u(0), \theta_e(0))$ in a neighborhood of zero. This pre-training control function is arbitrary and any other target velocity within a pre-defined feasible range $u_0 \in [0, u_{max}]$ could have been achieved without affecting the subsequent algorithm. The pre-training generates a deterministic policy $\mu(s|\theta_1)$ with weights θ_1 and no explicit time dependence that generates a similar periodic gait to that generated by the time-dependent prescribed forcing in (2.12). This is achieved by collecting state and action vectors (s, a) of a trajectory using the latter time-dependent policy, and finding θ_1 (the network architecture corresponding to these weights is explained in section 2.5.2) that minimize the sum of the error $(a - \mu(s|\theta_1))^2$ over the points sampled from the trajectory using stochastic gradient descent. The chosen trajectory has length $t = 100$ seconds at intervals $dt = 0.1$s. This optimization is done in 5 epochs which prevents over fitting, which is particularly important as data is used from only one trajectory. A schematic of this pre-training procedure is shown in Fig. 2.4(a).



$\tau(t) = A\sin\Omega t$
τ
Actor update Loss: $\min(L := (\tau_i - A_i)^2)$
A_i
S_i
Environment
Actor Network

(a)



$(s_e, a_e) = (u, \omega, \theta, u'_e(t), \beta_e)$
Environment
β_{e+1}
Exploration
$\beta_{e+1} = \begin{cases} \mathcal{N}(0,3) & \text{with probability } \epsilon \\ \mu_{e+1} & \text{with probability } 1-\epsilon \end{cases}$
State transitions
(s_{t+1}, a_{t+1})
(s_t, a_t)
$(s_t, a_t + da)$
Critic Network (θ_2)
$Q\,(s_{t+1}, a_{t+1})$
Q
$q_t = r + \gamma Q(s_{t+1}, a_{t+1})$
Critic update Loss: $\min_{\theta_2}(L_t := (Q - q_t)^2)$
Q_{a+da}
Q
Gradient:= $\nabla_a Q$
$a_t + da$
(s_t)
Actor Network(θ_1)
Actor update Loss: $\min_{\theta_1}(L_t := (\mu - a'_t)^2)$
$a'_t := \mu + \alpha\nabla_A Q$
μ_{e+1}
Trained Actor

(b)

Figure 2.4: (a) A schematic of pre-training to encode the limit cycle features of the reduced velocity space and periodic gaits into the policy output of an actor. (b) A schematic illustration of the application of the modified DPG algorithm to train a policy that can make the surrogate Chaplygin sleigh track a limit cycle and heading angle.

2.5.2 Deterministic Policy Gradient for Tracking Limit Cycles and Heading angle by Chaplygin Sleigh

Knowing θ_e and prescribing a target velocity u_t, the optimal control problem can be reduced into a regulation problem: find a policy $\mu(u, \omega, \theta_{target}, u_t)$ such to maximize the cumulative reward

$$r = -(u - u_t)^2 - 0.2|u - u_t| - 0.5(\theta_e)^2 - 0.1|\theta_e| \tag{2.13}$$

which can otherwise be denoted as

$$\arg\max_{\mu} \int_{t_0}^{t_f} r(u, \omega, \theta_e, u_t)dt \tag{2.14}$$

regulating $u \to u_t$ and $\theta_e \to 0$. The policy $\mu(u, \omega, \theta_e, u_t)$ is the same as the (control) torque τ with state feedback of $(u, \omega, \theta_e, u_t)$. Though u_t is not strictly a state of the system and does not have any dynamics except for those prescribed to it (variable speed tracking for instance), it must be relayed to the policy together with the states to allow appropriate control, so from the perspective of the DPG algorithm it is a state, and we will henceforth refer to it as such. This reward function is designed to simultaneously minimize the error in angle and velocity tracking, which are mutually exclusive goals: maintaining the target velocity requires "flapping" motion that results in periodic deviation from the desired heading angle. While the L^2 norm is traditionally used in cost functions for regularization problems and regression, it penalizes large deviations from the target value much more harshly than small ones, which makes it impossible to select a weight on the angle tracking reward that will adequately penalize small biases, while not penalizing flapping oscillations too much, which would result in swimming speeds well below the velocity target. However, allowing small biases to exist in the tracking angle can cause the swimmer to deviate from the prescribed path over time. This problem is alleviated by introducing an L^1 term on the angle tracking error to penalize those small biases. Similarly, an L^1 term is included in the velocity tracking reward to reduce the small negative bias in the velocity that results from the oscillation magnitude-velocity tradeoff. Specific weights were tuned on the surrogate model by repeated trials.

Approaches to this and similar problems include using an analytical harmonic balance calculation to solve the inverse problem of finding a biased sinusoidal input capable of reaching desired limit cycles for the Chaplygin sleigh such as in [52]. An extension for the case of a hydrofoil tracking a limit cycle was shown in [46] but this approach cannot be extended for path tracking or tracking variable velocities. The approach in [67] successfully uses proportional control on the heading angle, but there is not clear way to expand this approach to simultaneous velocity tracking. In the absence of a standard control method for even the path tracking problem for the Chaplygin sleigh, and the associated difficulty in carrying a similar approach to a swimmer we consider this problem well suited for reinforcement learning. Since the action space (allowable control) τ is continuous, DPG has the ability to utilize the entire action continuum [48, 49], which makes it suitable for this problem. Our DPG implementation is explained graphically in Fig. 2.4(b), and explained in text below.

Denote $s = (u, \omega, \theta_e, u_t)$ and define the policy $\mu(s)$ as a neural network with weights $\mathbf{w}$. The reward maximization problem can be approached with any sufficiently general numerical optimization algorithm, by perturbing the weights of μ, then recalculating the trajectory and its associated cumulative reward. While this Monte-Carlo approach will typically arrive at a maximum, it requires many trajectory iterations and an extensive training process, which combined take a long time to achieve convergence to any optimal policy. Policy gradient methods make this much faster by also calculating the value of a given state-action pair using a *critic* network $Q(s,a)$. Quantifying the value of an action in the context of a specific state allows for specific updates to the policy to perform higher-value actions, which requires fewer trajectories than the Monte-Carlo approach because there is only one cumulative reward per simulation, but many actions, generating more data that allows for finer training. In simple terms, it is more effective to promote and curtail specific actions than it is to promote or curtail a set of weights based on the value of its average action.

In a traditional policy gradient algorithm, the policy π is a probability distribution that estimates the appropriate action out of a discrete pre-determined set. While this is effective in many benchmark problems that feature discrete (and often small set of) actions [68, 69] or more commonly in games with discrete on/off controls [70], it is a limitation where a continuous spectrum of allowable actions is available. DPG has the ability to utilize the entire action continuum [48, 49], which makes it an intuitive choice for this problem.

More formally, the traditional DPG algorithm features two function approximators: an *actor* $\mu_{\theta 1}(s)$ and a *critic* $Q_{\theta 2}(s,a)$ where θ_1 and θ_2 are weights that define the approximation. The approximators used with this algorithm, such as in [49], are typically neural networks due to their utility as universal function approximators [71] and the ill-suitedness of competing discrete approximators, such as tables, to a continuous state and action space. These weights are typically initialized in a Gaussian distribution, and an empty experience buffer R is also generated. We use an ϵ *greedy* exploratory policy $\beta(s)$, which when sampled has a $1-\epsilon$ chance of returning $\mu(s)$ and an ϵ probability of returning a random value pulled from a normal distribution with deviation σ. The environment is simulated and the results stored in the experience buffer in the form (s, r, a, s^{+1}), where s^{+1} is the next state which is transitioned into. The discounted expected future reward $Q(s,a)$ can be updated to equal the sum of the current reward and the discounted approximated future reward for the known next state, a process known as *bootstrapping* which can be written as

$$Q_{target}(s,a) = r + \gamma Q_{\theta_2}(s^{+1}, \mu_{\theta_1}(s^{+1})), \tag{2.15}$$

where γ is the discount factor. A new set of actions can then be selected that maximize expected reward by performing one step of gradient ascent:

$$\mu_{target}(s) = \mu_{\theta_1}(s) + \alpha \frac{\partial Q_{\theta_2}(s,a)}{\partial a} \tag{2.16}$$

where the gradient can be computed efficiently along one dimension by a finite difference approximation

$$\frac{\partial Q_{\theta_2}(s,a)}{\partial a} \approx \frac{Q_{\theta_2}(s,a+da) - Q_{\theta_2}(s,a-da)}{2da}. \tag{2.17}$$

This gradient can also be computed by automatic differentiation, which is more efficient than finite difference methods for higher-dimension action spaces. Updating both networks $Q_{\theta_2}(s,a) \to Q_{target}(s,a)$ and $\mu_{\theta_1}(s,a) \to \mu_{target}(s,a)$ is then a supervised learning problem, where we minimize the least-square-error using 5 epochs of the *adam* algorithm [72], which is a stochastic gradient descent algorithm with momentum. This supervised approach is a departure from the original algorithm, which instead updates both sets of weights with a single step of gradient descent; we find that higher optimization speed of the *adam* algorithm compared to deterministic gradient descent makes this supervised approach faster on this problem. We use the *adam* parameters recommended in page 2 of [72], except with $\varepsilon = 10^{-7}$. These updates are repeated n times before a new batch of m environment simulations are generated and appended to the experience buffer, ensuring that there is enough data about perturbations local to the current trajectories to allow local optimization.

We selected initial parameter values of $n = 10$, $m = 10$ for the Chaplygin sleigh training, and $n = 10$ and $m = 1$ for the fluid simulation training. Additionally, we used values of $\alpha = 0.01$, $\gamma = 0.99$ $\sigma = 3$, $a = 10^{-5}$, and an experience buffer of size up to 3×10^5. As the training progresses, the values of u_t for each simulations are drawn from a widening probability distribution. At first, training is performed with $u_t = 1$. After 500 iterations, the target velocity for each trajectory is drawn from a uniform distribution such that $0.8 \leq u_t \leq 1.2$, and the target velocity is held constant within a trajectory. This is broadened to $0.6 \leq u_t \leq 1.6$, $0.4 \leq u_t \leq 2.0$, and $0.2 \leq u_t \leq 3.0$ after 1000, 1500, and 2500 iterations, respectively. This gradual introduction of different target velocities serves as another layer of curriculum, and reduces the risk of non-convergence. One modification is made to the DPG algorithm described thus far: because the critic is initialized with random weights but the actor is pre-trained, actor updates are disabled by defining $\alpha = 0$ for the first 10 iterations, at which point the critic has grown consistent with the actor and the learn rate can be reset to $\alpha = 0.01$.

When the episode reward on the sleigh environment qualitatively is seen to reach a peak or asymptotic convergence to a solution, the actor and critic are transplanted to the fluid simulation. The states used as policy inputs remain the same, however in the Chaplygin sleigh they correspond to full state feedback; in the simulated swimmer, the limited state feedback is insufficient to even fully describe the motion of the rigid body, let alone the high-dimensional fluid. This lack of observability makes training on the swimming simulations more data expensive, as the underlying Markov assumption that every state-action pair has a fixed probability distribution of states that it will transition to is not accurate. Though the overall system is Markov in that a complete state-action pair does transition deterministically to a specific next state, because the “state” given to the policy is a small subspace of the true state, it appears as if the system is non-Markov from the perspective of the agent. This reduced data efficiency, combined with the increased computational cost of generating data, makes training very slow in terms of wall-time.

Dense neural networks are the most popular choice of function approximator for DPG. A dense network contains one or more layers of neurons, and the value of each neurons is computed as a weighted sum of the neurons of the previous layer, with different weights for each neuron. The values are then passed through an activation function before being passed to the next layer, where the process is repeated until it reaches a neuron that is read as the output. We selected a 6-layer dense neural network, featuring an input layer of size 3 for the actor and size 4 for the critic, 5 layers of 40 neurons each with rectified linear unit (ReLU) activation, and a single output node with linear activation. In the policy a function $\tau = \tau_m \tanh \frac{a}{\tau_m}$ was used to smoothly attenuate the prescribed output a to a bounded torque value τ, which improves convergence. We select $\tau_m = 4$ as the range of allowed forcing, which we find to be large enough to perform effective control but small enough to not cause numerical problems in the fluid simulation.

The reinforcement learning uses an in-house implementation of DPG in Python with Tensorflow, accessed through the Keras API. Both of the neural networks are implemented in Keras, and the supervised updates of both the actor and critic networks are performed with $Model.fit()$, the Keras supervised learning utility, on batches of 1000 state transitions from the experience buffer. Both environments are implemented in Python with state transitions saved at interval $\Delta t = 0.1$ s. The sleigh simulations are trained with an ϵ-*greedy* exploration strategy with $\epsilon = 0.2$. The fluid simulations instead follows a Gaussian exploration approach, where normally distributed noise of deviation 0.2 is added to every action during exploration, which reduces the large discontinuities in torque that are seen in ϵ-*greedy* exploration, because they in turn cause large magnitudes of shed vorticity which results in an unrealistic wake.

2.6 Results - Velocity and Path Tracking

The supervised step of pre-training the sleigh to model a given control on the Chaplygin sleigh imprints into an actor a known control algorithm as a starting policy from which further exploration can enable the faster learning of policies for other reward functions (control goals). Figure 2.5(a) shows the reward during training by the low speed surrogate model that has been pre-trained to output periodic control actions (given in eq. (2.12)). The red curve shows the reward while the agent is learning a policy to track a high speed ($u_t = 2.5$) while the blue graph is for tracking a low speed ($u_t = 0.8$). Figure 2.5(b) shows the reward function during training of an actor without pre-training to a sub-optimal policy and instead using an actor network with weights initialized to Gaussian noise. The pre-training torque has an amplitude of only 0.3 while the torques required to track the low and high speed velocities are 1 and 1.2 respectively. Despite pre-training with such a suboptimal policy, the pre-trained actor learns faster and better, with the reward converging to higher values -350 (blue) for lower speeds while the reward for the agent without pre-training converges to -800 while also taking more epochs.

The limit cycles generated in the reduced velocity space as a result of this trained policy are shown for three different target velocities $u_t = (0.8, 1.5, 2.5)$ in Fig. 2.5(c) along with the limit cycle produced in pre-training where $u_0 = 0.32$. It can be seen that the actor adapts from generating a quazi-periodic trajectory to generating consistent limit cycles near the desired velocities. It also becomes more efficient at converting rotation into translation; the blue (trained) and green (untrained) trajectories have similar maximum amplitudes of ω, but the trained trajectory achieves a higher velocity u.

The mean value of the velocity u for the limit cycles shown in Fig. 2.5(c) have an error that does not decay, due to the fact that the reward function is a sum of the rewards accrued by minimizing the error $u - u_t$ as well as the error θ_e. The serpentine motion of the sleigh results in a necessary non-zero error in the heading angle, which can be reduced by smaller amplitude torques which slows down the sleigh. The learned policy, with the reward shown in Fig.2.5(d), is a compromise between achieving u_t and minimizing the error in the heading angle θ.

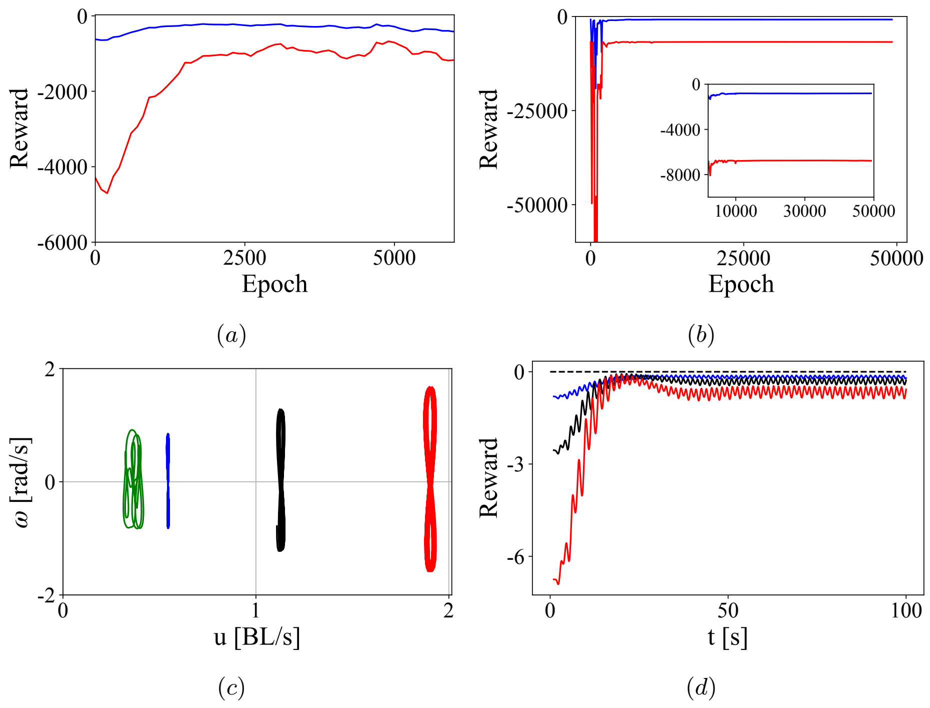


Figure 2.5: Total reward during epochs of training on the low-speed Chaplygin sleigh surrogate model for (a) the pre-trained actor and (b) an actor without pre-training. The red and blue lines show the reward while learning a policy to track a high speed ($u_t = 2.5$) versus a low speed ($u_t = 0.8$) respectively. (c) Limit cycles resulting from the policy learned on the surrogate model for target velocities of 0.8, 1.5 and 2.5 (blue, black, and red respectively) as well as the policy before training (green), and (d) The reward function for this policy in each case.

In the next stage of the curriculum learning, the agent trained on the surrogate Chaplygin sleigh is transferred to train in the fluid-hydrofoil simulation environment using the panel method, for the velocity and heading angle tracking problem with the aim of improving the sub optimal policy of the surrogate sleigh actor. A total of 150 trajectories of the swimmer and 7500 DPG iterations were used to adapt the policy of the Chaplygin sleigh to the swimmer, at a rate of 16.1 minutes per 1000 DPG updates and 5.49 minutes per trajectory generated on an Intel Xeon Gold 6148F processor. For comparison, a batch of 10 Chaplygin sleigh trajectories can be generated in 6.1s. As a result, the training time with the curriculum is much smaller than what would usually be required if the swimmer were to be directly trained without the intermediate training on the surrogate Chaplygin sleigh model. Figure 2.6(a) shows the reward function during each epoch using the low speed surrogate sleigh model to track a high (red) speed of $u_t = 2.5$ and low (blue) speed of $u_t = 0.8$. The reward at the beginning of the first epoch is due to the optimal policy learned on the surrogate

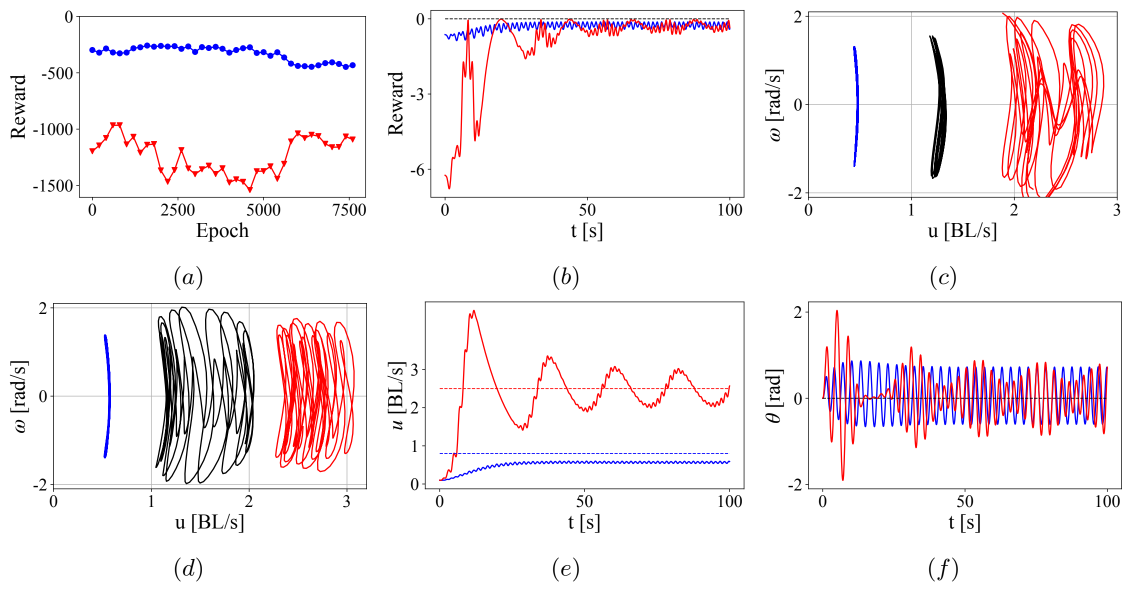


Figure 2.6: Trajectories on the hydrofoil after training on a low speed surrogate model - (a) Reward function during the epochs of training on fluid-hydrofoil simulations and (b) reward while executing the optimal policy tracking velocities and heading angle starting from rest. (c) Trajectories in the reduced velocity space due to the optimal policy of an actor trained on just the surrogate model and (d) produced by the optimal policy by an actor trained on additional fluid-hydrofoil simulation. (e) Velocity tracking by the hydrofoil for two tracking velocities and (f) simultaneously tracking 0° heading angle. Color legend - tracking speed $u_t = 0.8$ blue , $u_t = 1.5$ black and $u_t = 2.5$ red.

sleigh. The reward during the training on the fluid-hydrofoil interaction simulation does not change significantly. However this seeming lack of learning is deceptive, as borne out by examining the velocity of the hydrofoil. Figure 2.6(c) shows the trajectories in the reduced velocity space (ignoring a 50 second transient solution) for a swimmer produced by the optimal policy of the actor trained only on the surrogate sleigh model while 2.6(d) shows the same trajectories for a swimmer produced by the optimal policy of the actor trained further on the fluid-hydrofoil simulations. The trajectories in the reduced velocity space visit a larger range of u values after training on the fluid simulation. While the velocity of the swimmer does not converge to the target velocities, it does oscillate with a mean value close to the target velocity, when the target velocity is high (red) as shown in Fig. 2.6(d). This strategy forgoes the compromise between oscillation angle and velocity seen in the Chaplygin sleigh training, and instead features intervals of high-torque, low-reward periods of acceleration, followed by high-reward periods of low-torque coasting. The low-frequency periodic component of the velocity and pitch angle for the high-velocity tracking in this swimming strategy are shown in Fig. 2.6(e) and (f), respectively. This style of 'burst-and-coast' swimming is frequently observed in fish [73], and is consistent with other results in the RL swimming literature [38]. This strategy does not emerge during training on the surrogate model, but does emerge for high swimming velocities after transfer to the swimming simulation, even before continuing training. The continued training on the fluid simulations expands the range of target velocities for which the 'burst-and-coast' strategy is used.

A similar result is seen when the policy trained on the high speed surrogate model is further trained using the fluid-hydrofoil simulations to learn a policy to track different velocities and a heading angle of 0°. The reward function during the training does not increase significantly or can in fact decrease as seen in Fig. 2.7(a). Picking the optimal policy using the actor that produces the highest epoch reward as described in section 2.5 (B), produces a reward shown in Fig. 2.7(b). The burst and coast technique is once again seen in Figs.2.7(c)-(d) when tracking a high velocity (red) while a steady error is seen when tracking a lower velocity (blue).

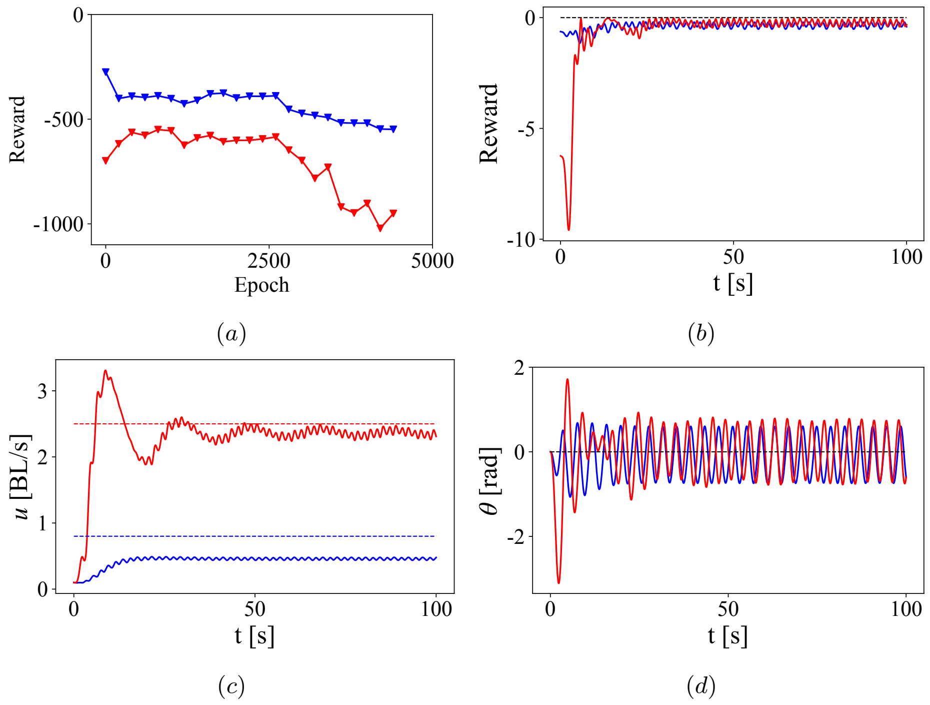


Figure 2.7: Training a high-speed surrogate model actor - (a) Reward function during the epochs of training on fluid-hydrofoil simulations and (b) the reward while executing the optimal policy tracking velocities and heading angle while starting from rest. (c) Velocity tracking by the hydrofoil for two tracking velocities and (d) simultaneously tracking 0° heading angle. Color legend - tracking speed $u_t = 0.8$ blue and $u_t = 2.5$ red.

It is to be emphasized that the lack of a significant increase in the reward function during the training on the fluid-hydrofoil simulations, compared to the reward obtained from the just the optimal policy of the surrogate sleigh actor is not a failure of the training; it can be attributed to two reasons. The first reason is that the reward function consists of a sum of terms that individually seek to minimize the error in the tracking velocity and heading angle. The oscillatory nature of the motion resulting in an oscillating heading angle, guaranteeing that the reward is always negative. The second reason is more subtle: the states s used in training the actor for swimming are merely a subset of the kinematic variables of the swimmer and do not include the distribution of vorticity in the fluid. The kinematics of a swimmer can be the same (or nearly so) for two different distributions of the vortex field. Moreover, an action (torque) for a given state s can produce a transition to completely different states that depend on the distribution of the vortex field. Since the vortex field is not an observed variable or state in the training, the state-action pair do not have a constant probability distribution, making the system seemingly non Markov from the perspective of the agent.

The combination of the surrogate model and curriculum learning outperforms direct reinforcement learning of a policy using fluid-hydrofoil simulations for tracking velocity and heading angle. Figure 2.8(a) shows the learning curve during such direct training, and Fig. 2.8(b-d) shows the trajectory generated by the trained policy after 4750 training iterations. The reward converges to a lower value than seen in Fig. 2.6(a) or Fig. 2.7(a). The reward per step due to the optimal policy ≈ -3 (shown in 2.8(b)) is also lower than the reward due to the optimal policies seen in Fig. 2.6(b) or Fig. 2.7(b), which have a value of > -1 per step. This is a result of poor velocity and angle tracking, and consequently graphs of $u(t)$ and $\theta(t)$ show large errors from the target velocities and heading angles.

The last step in path tracking is to give the time dependent $\theta_e(t)$ determined by a pure pursuit algorithm as an input (amongst the other states s) to the DPG agent trained in the fluid-robot simulation environment. We show the results of three such simulations of the agent tracking a path while also simultaneously tracking a specified time-varying velocity. Figure 2.9 shows the swimming hydrofoil tracking (pa) a straight line, (pb) a sinusoidal path and (pc) a circle. In each case the reference velocities for the hydrofoil are shown in Fig.2.9 (va)-(vc). The reference velocities are piecewise constant and either increase or decrease midway during the simulation time. The action (torque) due to the policy before attenuation by the inverse tangent in each case is shown in Fig. 2.9(Ta)-(Tc). The control torques are not just sinusoidal as in the pre-training, but have significant amplitude modulation and multiple harmonics, as a result of the series of training with

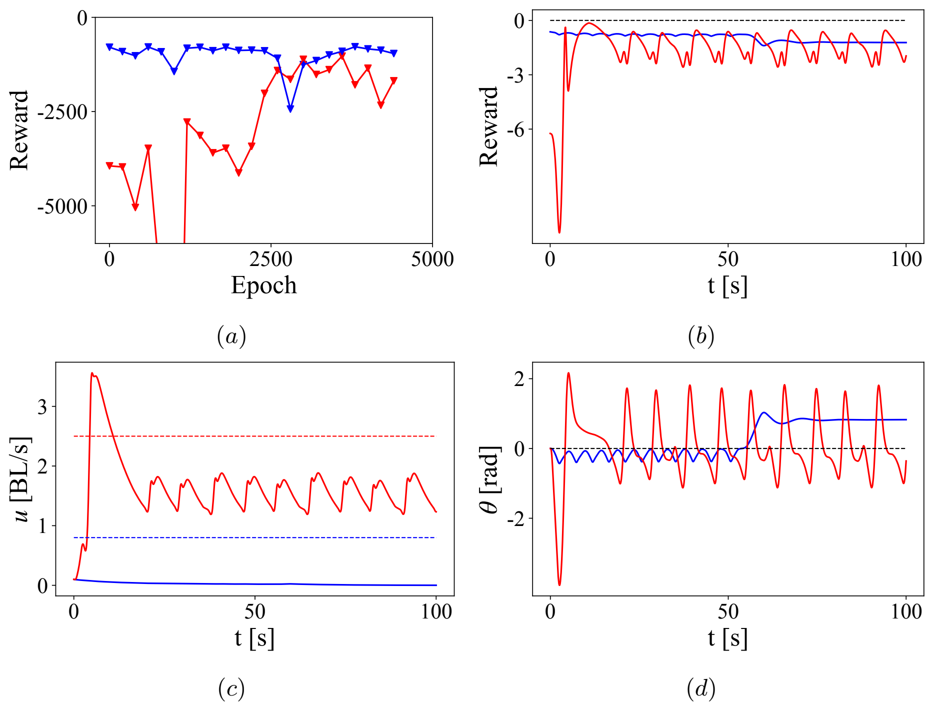


Figure 2.8: RL without surrogate model or curriculum learning - (a) Reward function for epoch training and (b) Reward function executing the optimal policy by a hydrofoil starting from rest. (c) Tracking low (blue) speed and high speed (red) and (d) simultaneously tracking 0° heading angle. For both cases, reward is lower than when trained with a curriculum. This is largely due to higher velocity error, with the swimmer tasked with reaching the low target speed instead coasting to a near-stop.

different models and curriculum which fine tuned the policy into a 'burst-and-coast' strategy.

2.7 Conclusion

Reinforcement learning methods in mobile robotics, particularly swimming robots with complex unmodelled physics, require large amounts of data covering a large subset of the relevant state space, which can be expensive and time consuming to obtain in both simulations and experiments. The results in this paper show the utility of surrogate models and curriculum-based reinforcement learning, wherein a DPG agent is trained in steps, for the control of a planar swimming robot. In each step the agent learns to generate and control certain features of the dynamics of the robot-environment action. An agent was trained to perform a series of increasingly complex tasks: tracking limit cycles in a reduced velocity space for a surrogate model, tracking heading angle and speed in the surrogate model and then finally transferred to learn doing these same tasks in a fluid-robot

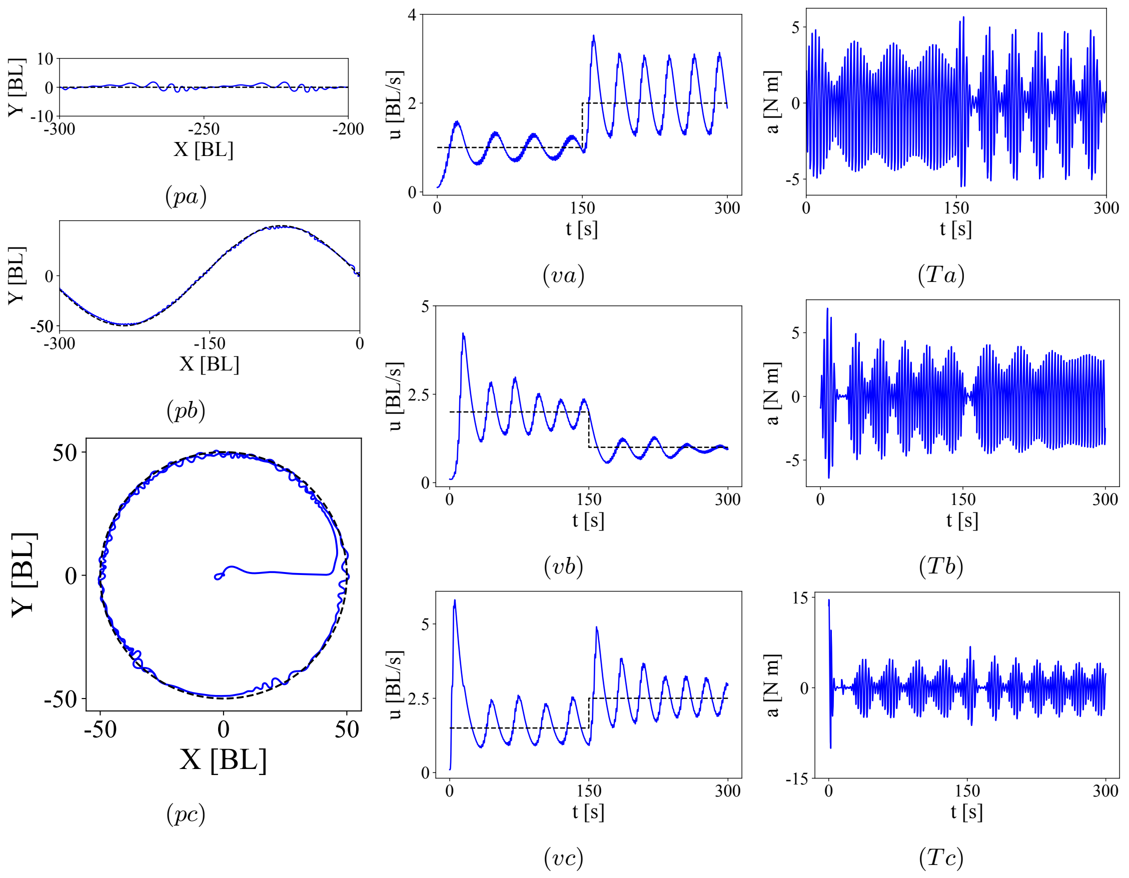


Figure 2.9: Pure-pursuit based path tracking for the simulated swimmer on (pa) a straight line and (pb) sinusoidal path and (pc) a circle. The target velocities for the case of the straight line, sinusoidal path and circle are shown by the dashed lines in (va), (vb) and (vc) respectively. The torques generated to track the straight line, sinusoidal path and circle are shown in (Ta), (Tb) and (Tc), respectively.

interaction environment. This approach reduces computational time, but more importantly creates a *physics-informed Reinforcement Learning* framework.

As a proof of concept, we have demonstrated only a limited range of curriculum on which a DPG agent can be trained. For instance, in the path tracking results in Fig. 2.9 the policy is trained for constant values of θ_{target}, and compensates for the variable θ_{target} values prescribed by the pure pursuit algorithm with higher torque values, resulting in higher velocities and deviations from the reference velocity. Further improvement is possible by including variable θ_{target} cases in a curriculum. While the robot considered in this paper is a planar swimming oscillating hydrofoil, more ‘fish-like’ robots can be trained using a similar framework to the one proposed in this paper. Fish-like robots usually have more than one body segment in their skeleton with joints that could be elastic. Recent work, for example [74], shows that in robots with elastic joints, tunable stiffness enables faster and more efficient swimming. Surprisingly, analogous results exist for ground-based multi-segment nonholonomic systems, see for example [75, 76], where effective stiffness tunable by periodic forcing leads to different limit cycles of varying efficiency, and multistable configurations for fast turning. Such nonholonomic systems can be used as surrogate models for multi-body fish-like robots using the same curriculum learning framework. With low-latency velocity feedback (either inertially on the body or from external measurement), the actor trained on the fluid surrogate model can be directly transplanted to the physical system, a new experience buffer can be generated, and the training can be continued on the physical system with little required setup.

This approach can be extended to three-dimensional swimming with suitable surrogate models. In three dimensions, additional control goals such as stabilization of roll and pitch may exist, while simultaneously tracking a path and/or a velocity. Both simulations of fluid-robot interactions and acquiring a large experimental data set in such cases can be even more challenging, further necessitating physics-based reinforcement learning with surrogate models. The current paper describes a preliminary proof of concept for such a framework.

Chapter 3

Nonholonomic constraint-induced bistability

This chapter has been adapted from a paper appearing in *Nonlinear Dynamics*:

C. Rodwell and P. Tallapragada, "Induced and tunable multistability due to nonholonomic constraints," Nonlinear Dynamics, vol. 108, no. 3, pp. 2115–2126, 2022.

3.1 Introduction

Mechanical systems with two or more stable configurations are of increasing interest in soft robotics from the perspective of generating gaits and locomotion, stabilization, and manipulation [77–79]. The general approach to achieve multistability in mechanical systems is to design a potential energy function for the system that has multiple local minima. Each of the minima of the potential function represents a stable configuration and the system's state can transition from one potential minimum to another under certain mechanical actuation. Much of the research has focused on two means to achieve a multi-well potential function: either by designing a system with geometric nonlinearities or by using materials or elastic elements with intrinsic or material nonlinearities. A variety of compliant systems and soft robotic systems possess one or both types of nonlinearities [77, 79–82].

In this paper we forego both of the above approaches and instead show that it is possible for a mechanical system whose associated potential energy has a single well to perform multiple unique stable limit cycle oscillations under the same periodic excitation. Motivated by the rich history of the use of pseudo-rigid body models of continuum and soft mechanical systems [83–87, 87–89], we base our investigation on a planar system with two rigid bodies connected by a linear torsional spring. Specifically, we consider the example of a four degree of freedom planar system with one nonholonomic constraint that is subjected to an internal periodic torque generated between the bodies. This mechanical system is a variation of the Chaplygin sleigh [90–94], a well known nonholonomic system. In earlier works [76] it was shown that a similar idealized version of this system in the absence of any damping, frictional resistance, or forcing has multiple stable configurations that arise from the nonlinear torsional spring or could demonstrate chaotic motion [95] in the absence of any spring connecting the two bodies.

In this paper we show that when damping is introduced into this system, along with periodic forcing, limit cycles in reduced velocity space are produced. We show that depending on the amplitude and frequency of the applied torque, the state of the system can oscillate around different mean states. The feature of tunable multistability that is demonstrated here is not merely one of possessing multiple stable static configurations, but is of a more dynamic nature. When subjected to periodic excitation the system has multiple stable limit cycles in its reduced state space, and depending on which limit cycle the system's state switches to, the motion of the body can be qualitatively different: in one case the mean path of the body is a straight line, while in the other it moves along a circular arc. We thus show gait selection that is determined by switching between limit cycles governed by the interplay of the nonholonomic constraint and the periodic forcing. Such interaction between nonholonomic constraints and periodic excitation has received less attention from the robotics perspective. It has however been investigated in the context of a spherical bearings such as for support of long span bridges, ball vibration absorbers. For instance [96, 97] investigated the problem of a ball rolling in a spherical cavity subjected to parametric excitation and showed rich variety of dynamics, limit cycles and quasi-periodic motion depending on the damping and the excitation parameters.

The results in this paper have applications to mobile soft robots which, by the nature of their interaction with the ground or a substrate, can be subject to nonholonomic constraints. Designing soft robots which exploit such constraints on their dynamics can open up a new means to achieve the properties of multistable mechanical systems. This work also has potential significance to the design and control of swimming robots; in previous work it was demonstrated that nonholonomic constraints are present in the swimming motion of fish-like bodies [44,98,99] and this recognition can enable the design of better controllers for swimming robots [16,47,99,100]. The results in this paper show the possibility of achieving different stable limit cycles in physically different configurations and associated gaits for swimming robots without changing the input forcing, which can greatly improve their agility. Within the context of nonholonomic systems, which is a very well studied topic [92, 101–104] in mechanics and robotics, the findings in this paper show the importance of periodic excitation in producing a variety of gaits.

The paper is organized as follows. In section 3.2 the model of Chaplygin sleigh with an additional appendage and the governing equations of motion are discussed and their relation to previous work is reviewed. In section 3.3.1 the special case of a non dissipative and unforced Chaplygin sleigh with an elastic appendage is discussed. The fixed points for velocity of the undamped, unforced sleigh exist as shown in [76].The undamped unforced case bears significance to the damped and periodically forced Chaplygin sleigh; although these fixed points disappear for the damped and forced sleigh, limit cycles are produced around them. The stability and bifurcations of these limit cycles and their relevance to the physical gaits of the Chaplygin sleigh are discussed in section 3.3.

3.2 Mechanical model and governing equations

The Chaplygin sleigh is a rigid body with a knife edge at one end that prevents slipping in the transverse direction as it moves in a plane. This no-slip condition is a nonholonomic constraint, and this seemingly simple mechanical system can exhibit very rich dynamics [50,90–94,105]. In this paper we consider a modified two-link Chaplygin sleigh as depicted in fig. 3.1. The rear link of length ϵl will be referred to as the “tail", and it is connected to the other link (referred to as the “head" link) at a revolute joint with a linear torsional spring of constant stiffness K. The two link Chaplygin sleigh is motivated by potential applications to fish-like swimming robots as well as the multi-link snake-like terrestrial robots. The torsional spring at the revolute joint mimics the stiffness of the tail of a fish or a reptile. The spatial frame is denoted by $X-Y$, and the body frame denoted by $X_b - Y_b$ is attached to the tail link with its origin collocated with the revolute joint. The tail link has a massless wheel or a knife edge at its end which prevents slipping in the transverse (Y_b) direction but allows motion in the longitudinal (X_b) direction.

The configuration space of the system is $Q = SE2 \times S^1$ and is parameterized by the coordinates $\boldsymbol{q} = (x_1, y_1, \theta_1, \theta_2)$ where (x_1, y_1) are the coordinates of the center of mass of the sleigh tail link, θ_1 is the angle made by the body frame with respect to the spatial frame and θ_2 is the angle of the vector between the revolute joint and the center of mass of the head link with respect to the spatial frame. The Lagrangian of the system is $\mathcal{L} = \mathcal{T}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathcal{V}(\boldsymbol{q})$ with kinetic energy $\mathcal{T} = \frac{1}{2}\dot{\boldsymbol{q}}^T \mathcal{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$ and potential energy $\mathcal{V}(\boldsymbol{q}) = \frac{1}{2}K\delta^2$. It is emphasized that K is taken as a constant, so the potential function is a single-well potential and does not by itself produce multiple equilibria. The mass matrix $\mathcal{M}$ is defined in Appendix A. We imagine that a motor at the revolute joint applies an internal periodic torque $A \sin \Omega t$, which results in equal and opposite torques on the head and tail links. The resulting spin and translation of the tail link has to satisfy the nonholonomic constraint, which defines that transverse velocity u_y (along the Y_b direction) of the point P is constrained to be zero,

$$u_y = -\sin\theta_1\, \dot{x}_1 + \cos\theta_1\, \dot{y}_1 - \epsilon\, l\, \dot{\theta}_1 = 0. \tag{3.1}$$

The constraint can be expressed compactly as $\mathcal{W}(\boldsymbol{q})\dot{\boldsymbol{q}} = 0$ with $\mathcal{W}(\boldsymbol{q}) = [-\sin\theta \ \ \cos(\theta) \ \ -\epsilon l \ \ 0]$.

We further assume that the sleigh experiences viscous dissipation with the Rayleigh function

$$\mathcal{R} = \frac{1}{2}\left(c_u(\dot{x}_1\cos\theta_1 + \dot{y}_1 \sin\theta_1)^2 + c_\omega \dot{\theta_1}^2 + c_\delta \dot{\delta}^2\right), \tag{3.2}$$

where $\delta = \theta_2 - \theta_1$ and c_u, c_θ, and c_δ refer to the damping on the translational velocity at the constraint, the rotational velocity of the tail link, and the interlink rotational velocity, respectively. The Euler-Lagrange equations with the constraint forces are

$$\begin{bmatrix} \boldsymbol{\mathcal{M}} & -\boldsymbol{\mathcal{W}}^T \\ \boldsymbol{\mathcal{W}} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mathcal{B}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ -\dot{\boldsymbol{\mathcal{W}}}\dot{\boldsymbol{q}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\tau}(t) \end{bmatrix}. \tag{3.3}$$

The forcing term is $\boldsymbol{\tau}$ and the only non-zero forcing terms are the joint torque $\boldsymbol{\tau}_3 = -\boldsymbol{\tau}_4 = A\sin\Omega t$. The vector $\boldsymbol{\mathcal{B}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ contains the gyroscopic terms and the elastic forces, the ith entry of which is

$$\boldsymbol{\mathcal{B}}_i = \frac{1}{2}\left(\frac{\partial \boldsymbol{\mathcal{M}}_{ik}}{\partial \boldsymbol{q}^j} + \frac{\partial \boldsymbol{\mathcal{M}}_{ji}}{\partial \boldsymbol{q}^k} - \frac{\partial \boldsymbol{\mathcal{M}}_{jk}}{\partial \boldsymbol{q}^i}\right)\dot{\boldsymbol{q}}^j \dot{\boldsymbol{q}}^k - \frac{\partial \mathcal{V}}{\partial \boldsymbol{q}^i} - \frac{\partial \mathcal{R}}{\partial \dot{\boldsymbol{q}}^i}$$

The resulting Euler-Lagrange equations are independent of (x, y, θ_1) and a reduction of the equations to the body frame creates a reduced system of velocity equations decoupled from the

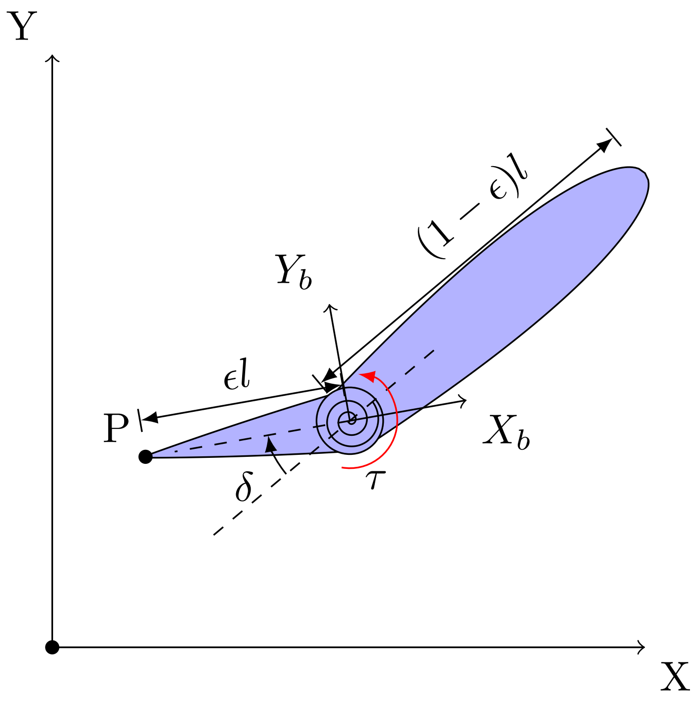


Figure 3.1: A two link Chaplygin sleigh system of total length l and total mass m. The tail link of length ϵl, mass m_1, and moment of inertia I_1 and the head link of mass m_2 and moment of inertia I_2 are connected by a revolute joint. The body frame $X_b - Y_b$ is fixed to the revolute joint and the X_b axis is aligned with the tail longitudinal axis. The velocity of the point P on the tail link is constrained to be zero in the Y_b direction. The angle θ_1 is the angle between the body frame and the spatial frame while angle δ is the relative angle between the links, and $\delta = 0$ corresponds to the fully extended configuration. The center of mass of the tail link is along the line joining P to the revolute joint. The center of mass of the head link is at a distance of $\frac{l}{2}(1-\epsilon)$ from the joint and the special case where the head link length is $l(1-\epsilon)$ is considered.

grouped variables (x, y, θ_1). Setting $\boldsymbol{\xi} = [u_x, \omega_1, \omega_2, \delta]^T$ where u_x is the velocity at the constraint, $\omega_1 = \dot{\theta}_1$, and $\omega_2 = \dot{\delta}$, the velocities can be transformed to the body frame as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \mathbf{R} \begin{bmatrix} u_x \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \mathbf{R} \begin{bmatrix} \dot{u_x} \\ 0 \end{bmatrix} + \omega_1 \times \mathbf{R} \begin{bmatrix} u_x \\ 0 \end{bmatrix} \tag{3.4}$$

for rotation matrix $\mathbf{R}(\theta)$. Note that (x_1, y_1) denote the position of the center of mass of the tail link, and (x, y) denote the position of the constraint.

While the derivation to this point is applicable for arbitrary geometry, for our analysis we will consider a specific geometry where both links are rectangular with total width $\frac{l}{2}$, moment of inertia coefficient $\gamma = 1/12$ and constant density. In this case the mass and inertia parameters of the links defined in the caption of fig. 3.1 can be defined in terms of the geometric parameters and overall mass defined in the same figure as $m_1 = \frac{\epsilon l}{m}$, $m_2 = \frac{(1-\epsilon)l}{m}$, $I_1 = 4\gamma m_1 \left((\epsilon l)^2 + \frac{l^2}{4}\right)$ and $I_2 = 4\gamma m_2 \left(((1-\epsilon)l)^2 + \frac{l^2}{4}\right)$. The variables and parameters are then rescaled to eliminate m and l by defining $u_x' = \frac{u_x}{l}$, $A' = \frac{A}{ml^2}$, $\alpha = \sqrt{\frac{K}{ml^2}}$, $c_u' = \frac{c_u}{m}$, $c_\omega' = \frac{c_\omega}{ml^2}$, $c_\delta' = \frac{c_\delta}{ml^2}$, $I_1' = \frac{I_1}{ml^2}$, $I_2' = \frac{I_2}{ml^2}$, $m_1' = \frac{m_1}{m}$, and $m_2' = \frac{m_2}{m}$. A new variable E' representing the rescaled energy is introduced as $E' = \frac{E}{ml^2}$ where $E = \mathcal{T} + \mathcal{V}$. As a further simplification, we consider the rescaled damping ratios equal, so $c = c_u = c_\theta = c_\delta$, and rescaled stiffness α is taken to be $\sqrt{10}$ throughout the paper. For simplicity of notation, the $'$ superscripts are dropped, and all instances of the redefined constants henceforth are taken to be in their scaled form, unless noted otherwise.

The rescaled equations of motion are of the form

$$\boldsymbol{\mathcal{N}}\dot{\boldsymbol{\xi}} = \boldsymbol{g}(\boldsymbol{\xi}) + \boldsymbol{f}(t), \tag{3.5}$$

where the rescaled inertia-like matrix $\boldsymbol{\mathcal{N}}(\boldsymbol{\xi})$ is described by (2) in Appendix A. The four entries of the vector field $\boldsymbol{g} \in \mathbb{R}^4$ are

$$\begin{aligned}\boldsymbol{g}_1 = {} & (\omega_1 + \omega_2)^2 (\epsilon - 1)^2 \cos\delta + (2 - \epsilon)\,\epsilon\omega_1^2 \\ & - c_u u_x\end{aligned} \tag{3.6}$$

$$\begin{aligned}\boldsymbol{g}_2 = {} & 2\epsilon\omega_2 (2\omega_1 + \omega_2)(\epsilon - 1)^2 \sin\delta \\ & - \omega_1 u_x (\epsilon - 1)^2 \cos\delta + (\epsilon - 2)\,\omega_1 u_x \epsilon - c_\omega \omega_1\end{aligned} \tag{3.7}$$

$$\begin{aligned}\boldsymbol{g}_3 = {} & -\omega_1 u_x (\epsilon - 1)^2 \cos\delta - 2\omega_1^2 \epsilon (\epsilon - 1)^2 \sin\delta \\ & - \alpha^2 \delta - \omega_2 c_\delta\end{aligned} \tag{3.8}$$

and

$$\boldsymbol{g}_4 = \omega_2. \tag{3.9}$$

The forcing term $\boldsymbol{f}(t)$ is periodic with frequency Ω,

$$\boldsymbol{f}(t) = \begin{bmatrix} 0 \\ 0 \\ A\sin\Omega t \\ 0 \end{bmatrix}. \tag{3.10}$$

3.3 Tunable limit cycles and configuration changes

The periodically forced two-link Chaplygin sleigh demonstrates complex dynamics that depend on the frequency and amplitude of the forcing as well as geometric parameters such as ϵ. Underlying this behavior are dynamics of the unforced and non-dissipative two-link Chaplygin sleigh system. This is a general theme in many dynamical systems, such as the multi-well Duffing oscillator, where the fixed points of the unforced Duffing oscillator form the skeleton of the dissipative forced dynamics. Therefore we first discuss the somewhat simpler case of the dynamical system with no dissipation and free of forcing, i.e. $c = 0$ and $\boldsymbol{f}(t) = \boldsymbol{0}$.

3.3.1 Multistability in the conservative system

Suppose $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{g}(\boldsymbol{\xi}; c_u = 0, c_\omega = 0, c_\delta = 0, A = 0)$: it will be shown that the limit cycles of interest of the dynamical system (3.5) are created around the fixed points of the dynamical system

$$\boldsymbol{\mathcal{N}}\dot{\boldsymbol{\xi}} = \boldsymbol{h}(\boldsymbol{\xi}). \tag{3.11}$$

One observation that can be made about the dynamical system (3.11) is that the total energy $E = \mathcal{T} + \mathcal{V}$ is constant along its integral curves in the absence of any damping or forcing. The inertia tensor $\boldsymbol{\mathcal{N}}$ is symmetric and positive definite and thus invertible except at $\epsilon = 1$. The fixed points of (3.11) are therefore given by $\boldsymbol{\mathcal{N}}^{-1}\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{0}$, and since the determinant $\det(\boldsymbol{\mathcal{N}}) \neq 0$, the only solution to this is $\boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{0}$. The fixed points of (3.11) are non-isolated due to the conservation of energy; a continuous family of fixed points exists as the energy of the system is varied, and perturbations from a fixed point that change the energy of the system can never decay to the same fixed point. To isolate the fixed points, we perform a reduction of dimension of (3.11) by considering dynamics restricted to a constant energy manifold. This reduced dynamical system denoted by $\dot{\boldsymbol{\xi}}_r = \boldsymbol{H}_r(\boldsymbol{\xi}_r; E)$

is described in Appendix A (13) where $\boldsymbol{\xi}_r = [\omega_1, \omega_2, \delta]^T$ and the longitudinal velocity denoted as a function of $\boldsymbol{\xi}_r$ as $u_x = u_x(\boldsymbol{\xi}_r; E)$. We will then consider perturbations around a fixed point that preserve energy.

This system has two distinct types of fixed points. It can be verified that $(u_x = \pm\sqrt{2E}, \omega_1 = 0, \omega_2 = 0, \delta = 0)$ is one set of fixed points. This set of fixed points corresponds to straight line motion where the sleigh shape is straight with $\delta = 0$; the sleigh could be moving forward $\left(u_x = \sqrt{2E}\right)$ or backward $\left(u_x = -\sqrt{2E}\right)$. The Jacobian $\mathbf{J}$ is the gradient of the reduced vector field $\boldsymbol{H}_r(\boldsymbol{\xi}_r; E)$. The eigenvalues of $\boldsymbol{J}$, denoted by (μ_1, μ_2, μ_3) in descending order of their real component, are in the left half plane when evaluated for fixed points corresponding to $u_x = \sqrt{2E}$ as shown in fig. 3.2 and in the right half plane for fixed points corresponding to $u_x = -\sqrt{2E}$ [76].

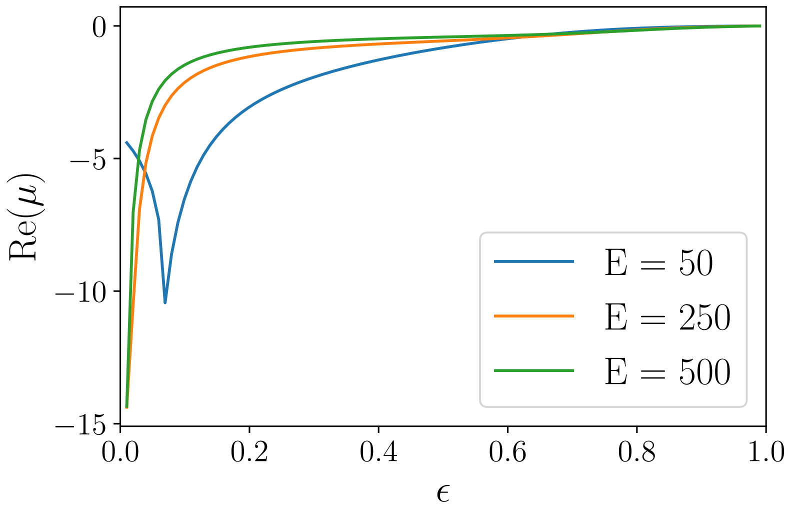


Figure 3.2: The largest real component of the eigenvalues of the Jacobian $\boldsymbol{J}$ about the forward non-buckled fixed point $(u_x = \sqrt{2E}, \omega_1 = 0, \omega_2 = 0, \delta = 0)$ for varying energies with $\epsilon = 0.15$, $c = 0$, and $A = 0$. The real component of the largest eigenvalue, and thus all the eigenvalues, is always negative, indicating that the non-buckled fixed points with positive u_x are always stable within the considered energy range $0 < E < 1000$.

The other set of fixed points corresponds to a buckled shape with $\delta = \pm\,\delta^*$ (symmetrically placed to the “left" and “right" of the longitudinal axis of the tail link) where

$$\delta^* = \cos^{-1}\left(\frac{\epsilon\,(\epsilon - 2)}{\epsilon^2 - 2\,\epsilon + 1}\right) \tag{3.12}$$

is obtained from solving (3.7) $\boldsymbol{g}_1 = 0$ or $\boldsymbol{g}_2 = 0$. The buckled fixed points exist only if $\epsilon < \epsilon_0 = 1 - \frac{1}{\sqrt{2}}$. For each of the $\pm\delta^*$ there exist at most four pairs of ω_1^* and u_x^* which satisfy $\dot{\boldsymbol{\xi}} = 0$ for a given energy level. The values of u_x^* and ω_1^* are given in (14) and (15) in Appendix A. All four fixed points exist for $\epsilon_l < \epsilon < \epsilon_0$, where ϵ_l is a function of energy, with sample values shown in fig. 3.3. There are thus a total of eight fixed points for the buckled state of motion. The four values of u_x^* and ω_1^* corresponding for each of the symmetric shapes $\pm\delta^*$ imply that at each of the symmetric equilibrium shapes, the kinetic energy of the system can be partitioned into two possible values of translational and rotational energies. Because these energies can result in either forward or backwards motion, for the two fixed points in positive u_x, (u_x^*, ω_1^*), the two pairs $-(u_x^*, \omega_1^*)$ are also fixed points. Numerical evaluation of the eigenvalues of the Jacobian, $\boldsymbol{J}(\omega_1^*, \omega_2 = 0, \delta^*, E)$ show that only of one of these partitions of energy is stable, so there are two stable and six unstable equilibrium states of motion in the buckled state. For reasons of symmetry, we show results of these simulations only for the buckled state with $\delta = +\delta^*$ and the same result holds for the other buckled state $\delta = -\delta^*$. Fig. 3.3 shows a plot of the variation of the fixed points (u_x^*, ω_1^*) as the bifurcation parameter ϵ varies for different energies E of the system. The portion of the curves shown by solid lines represent the stable fixed points and the dashed lines represent the unstable fixed points.

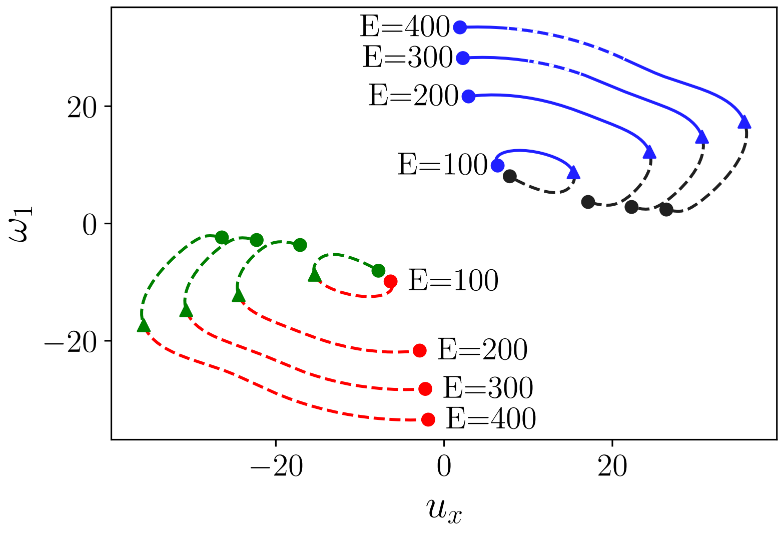


Figure 3.3: Dependence of the locations of fixed points u_x^* and ω_1^* on ϵ for a sampling of energy levels for the conservative system. Circles indicate $\epsilon = 1 - \frac{1}{\sqrt{2}}$, the maximum ϵ value for which $\delta^* \neq 0$ fixed points exist. The fixed point locations then vary with decreasing ϵ until their annihilation at $\epsilon_l(E)$, indicated by triangular markings. The lower bound $\epsilon_l(E)$, varies with energy, and corresponds to $\epsilon = 0.134, 0.053, 0.034, 0.025$ for energies $E = (100, 200, 300, 400)$, respectively. Dashed lines indicate unstable fixed points while solid lines indicate stable fixed points.

Fig. 3.4 shows a plot of the eigenvalues of the Jacobian $\mathbf{J}(\omega_1^*, \omega_2 = 0, \delta^*, E)$ with the largest real component evaluated at the stable and unstable fixed points with positive u_x. For the stable fixed point, the eigenvalues with the highest real part, shown by blue solid lines for different values of E, are in the left half plane, with the exception of a small region in the right half plane at high E values. The black solid line shows the highest real eigenvalue of the Jacobian evaluated at the lower ω_1^* fixed point in positive u_x. The eigenvalues start at the upper ϵ limit ϵ_0 with a value of $\mu = 0+0i$, and as ϵ decreases the stable fixed points gain a complex component, while the unstable eigenvalues remain on the real axis. As ϵ decreases further to a critical value ϵ_l (shown by the triangles in fig. 3.3), the pairs of fixed points collide and annihilate in a saddle-node bifurcation, and the eigenvalues of $\mathbf{J}$ about the four fixed points return smoothly to $\mu = 0 + 0i$. Jacobian matrices calculated about the fixed points in negative u_x^* are found to always have at least one eigenvalue with a positive real component.

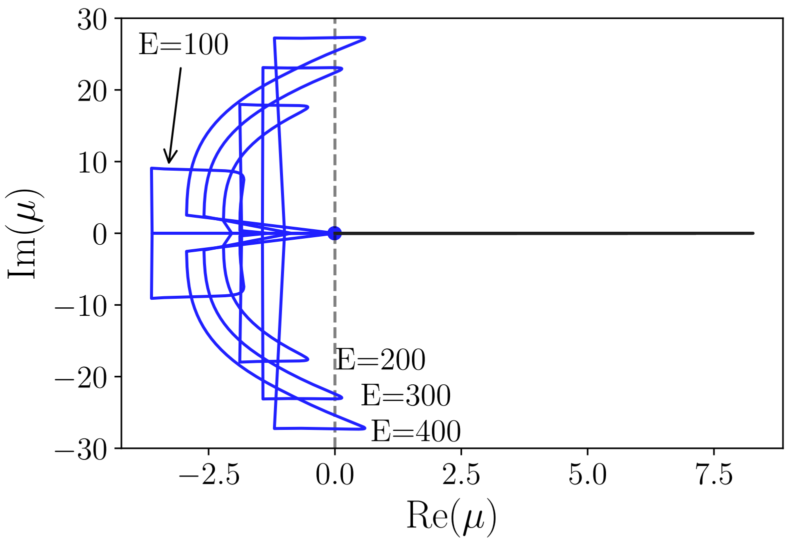


Figure 3.4: The eigenvalues with the highest real component of the Jacobian $\boldsymbol{J}$ of positive u_x buckled fixed points of the conservative system shown in fig. 3.3. Four energy values, $E = 100, 200, 300,$ and 400, are selected, and ϵ is varied between the maximum ϵ value $\epsilon_0 = 1 - \frac{1}{\sqrt{2}}$ and the minimum $\epsilon_l(E)$. The highest eigenvalue of the black curve from fig. 3.3, which has the lower ω_1^* value in positive u_x, is purely real and never negative. The fixed points for the blue line start at $\mu = 0 + 0i$ at the upper epsilon bound and return there at the lower bound. Between those bounds, the eigenvalues can mostly be found in the left-half plane, indicating stability with the exception of a small ϵ region at high energies which coincides with the dashed portion of the blue line in fig. 3.3.

3.3.2 Multistability in the dissipative forced system

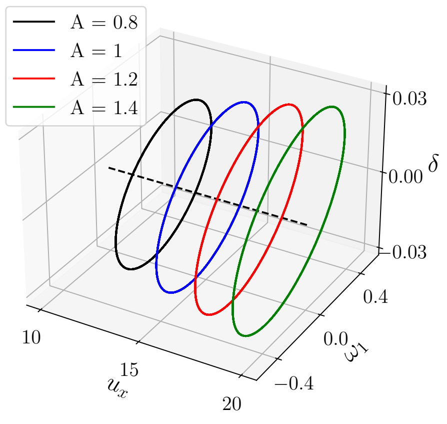
With the addition of damping, all the fixed points of the conservative system (3.11), with the exception of the origin $\boldsymbol{\xi} = [0,0,0,0]^T$, disappear. This can be shown based on the movement of energy into and out of the system, which is given by

$$\frac{dE}{dt} = \left(-c_u u_x^2 - c_\omega \omega_1^2 - c_\delta \omega_2^2 + \omega_2\, A \sin \Omega t\right). \tag{3.13}$$

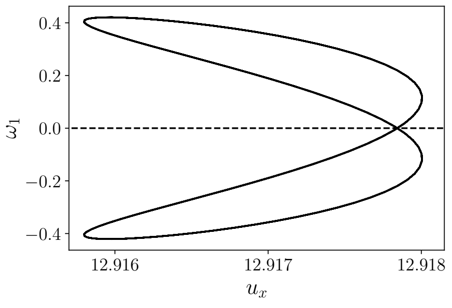
In the absence of any forcing, $A = 0$, $\dot{E} < 0$ except for the rest state $(u_x = 0, \omega_1 = 0, \omega_2 = 0)$. Furthermore the rest state $\boldsymbol{\xi} = [0,0,0,0]^T$ is a fixed point and it can be checked that $\boldsymbol{g}(\boldsymbol{\xi} = [0,0,0,0]^T) = 0$. Setting E as a Lyapunov function, we can therefore conclude that $\boldsymbol{\xi} = [0,0,0,0]^T$ is a global attractor for this system if spring stiffness is positive.

When the forcing $\boldsymbol{f}(t) \neq \mathbf{0}$, the origin is no longer a fixed point of (3.5) since $\boldsymbol{g}(\boldsymbol{\xi}) = 0$ and $\boldsymbol{f}(t) \neq \mathbf{0}$. In fact, the periodically forced dynamical system (3.5) has no fixed points. However, the effect of the "ghost fixed points" of (3.11) persists in that limit cycles of the dissipative system are created around fixed points of the conservative system. The existence and stability of these limit cycles is dependent on the amplitude and frequency of the forcing torque, making it possible to tune or select the limit cycles of the system in real time. Much like the fixed points of the conservative system, the limit cycles of the forced dissipative system can be broadly classified into two types. The first type features oscillations of the tail where the mean value of the angle δ is zero. A set of such limit cycles are shown in fig. 3.5(a) in the space of (u_x, ω_1, δ) for varying amplitudes of the forcing torque. As the amplitude of the forcing increases, the amplitude of the oscillations of the tail and of the flexing of the body increase, as does the longitudinal speed u_x. These limit cycles have a slight twist which manifests in the self-intersecting "figure-8" curves that appear when the limit cycles are projected onto the $\omega_1 - u_x$ subspace. These self-intersecting curves indicate that the longitudinal velocity u_x has a fundamental frequency twice that of the angular velocity ω_1, which is reminiscent of similar behavior for a single link Chaplygin sleigh [94].

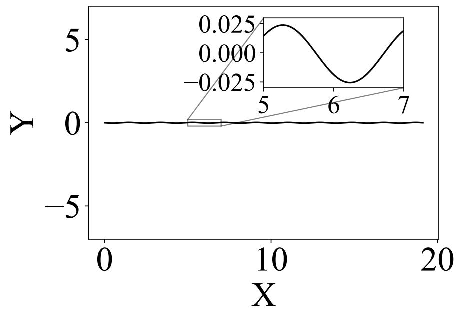
These small amplitude limit cycle oscillations around the $\delta = 0$ non-buckled configuration lead to a trajectory of the constraint point which is serpentine with small peak-to-peak variation in the physical plane such as shown in fig. 3.5(c). However, the time averaged (over integer multiples of $T = \frac{2\pi}{\Omega}$) path is a straight line. The amplitude of tail oscillations is small as shown in fig. 3.5(d). An animation of this motion is shown in the supplementary video.



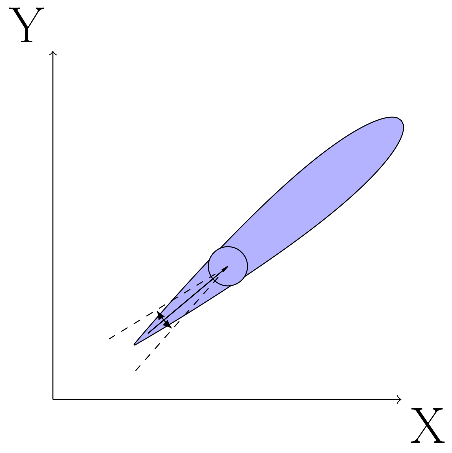
(a)



(b)



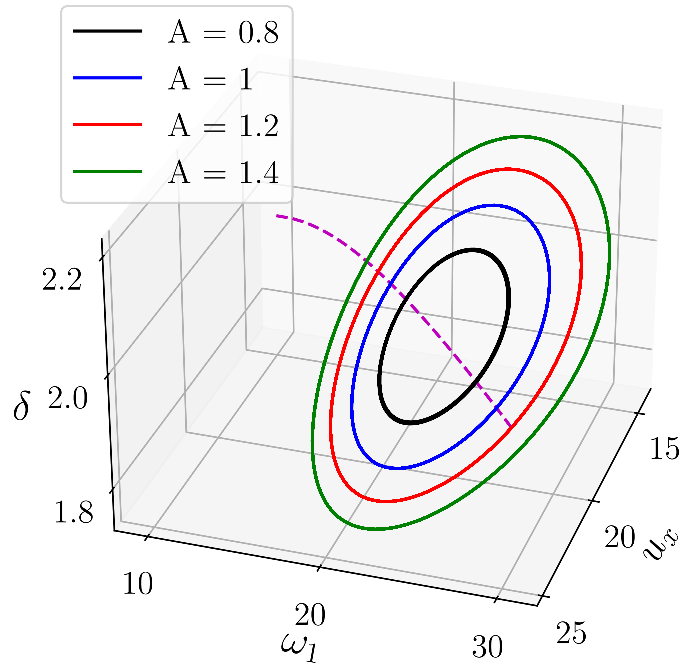
(c)



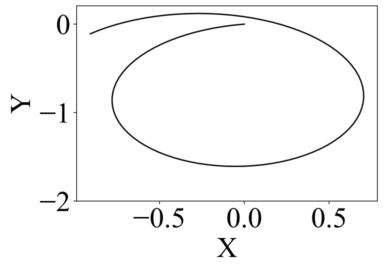
(d)

Figure 3.5: (a) Limit cycles of the forced dissipative system with $c = 0.001$, $\epsilon = 0.15$, and $\Omega = 21$ about the $\delta = 0$ fixed points of the conservative system (indicated by a dashed line). These limit cycles take the form of near-circular orbits. The response to increasing forcing amplitude is an increased amplitude of oscillations and increased velocity u_x. (b) A projection of the limit cycle (for $A = 0.8$) onto the u_x-ω_1 plane with two self crossing loops symmetric about $\omega_1 = 0$. (c) The serpentine path of the sleigh with average straight-line motion. (d) Small amplitude oscillations of the tail with the extreme angles δ shown by the dashed lines.

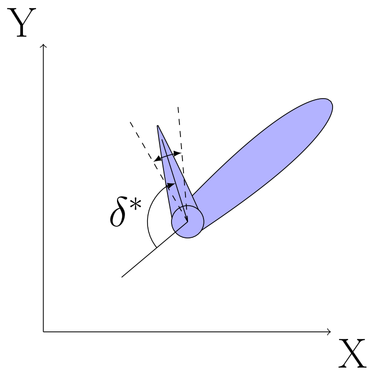
A different set of stable limit cycles coexist when the forcing frequency Ω is close to a natural frequency of oscillations seen in the conservative system around the buckled fixed points. A natural frequency for this system can be defined as the imaginary component of the eigenvalue with the highest real component, μ_1, of the Jacobian $\boldsymbol{J}$ evaluated at the buckled fixed points of the conservative system. Fig. 3.6(a) shows limit cycles in the $u_x - \omega_1 - \delta$ space due to a forcing frequency of $\Omega = 21$. This forcing frequency is close to the natural frequency $Im(\mu_1) = 20.8$ of the stable buckled fixed point of the conservative system at $\epsilon = 0.15$ and $E = 300$. This energy value



(a)



(b)



(c)

Figure 3.6: (a) The limit cycles of the dissipative system at varying forcing amplitudes, with constant damping $c = 0.001$, forcing frequency $\Omega = 21$, and $\epsilon = 0.15$. The locus of stable fixed points of the conservative system as energy varies energy pass through the dissipative limit cycles. Due to symmetry, an identical set of fixed points and limit cycles exists reflected about $\delta = 0$ and $\omega_1 = 0$, corresponding to turning in the other direction. (b) The path traced by the constraint point over a single forcing period, indicating fast turning motion along curved paths with the limit cycle oscillations about the buckled tail configuration. (c) The small amplitude of interlink oscillations are shown by the dashed lines around the buckled state.

is roughly representative of the cycles shown in fig. 3.6(a), which have energies that vary with both time and forcing amplitude, but are bounded between a minimum of $E = 297$ for $A = 0.8$ and a maximum of $E = 317$ for $A = 1.4$. The limit cycles grow in diameter and energy as the amplitude of the forcing increases. Interestingly, the limit cycles are centered around and enclose the locus of the buckled state fixed points of the conservative system, shown by the dashed (magenta) curve. The time-averaged value of ω_1 along these limit cycles is clearly non-zero, and as a result the sleigh moves along a sharply curved path in the $X-Y$ plane as shown in fig. 3.6(b) with the tail oscillating about the buckled state as shown in fig. 3.6(c). The supplementary video also shows an animation of this type of gait.

To understand the stability of the limit cycles in the buckled configuration, we analyze the stability of the fixed points of the time T-Poincare map. Suppose the flow map for the dynamical system (3.5) is $\Phi_{t_0}^t : \boldsymbol{\xi}(t_0) \mapsto \boldsymbol{\xi}(t)$. The T-periodic Poincaré map is

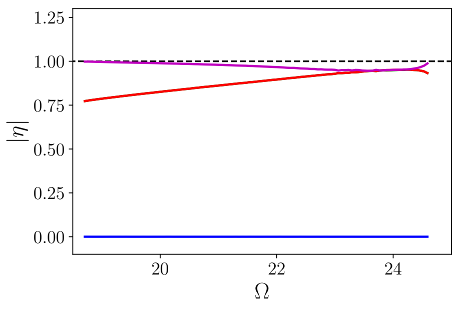
$$\mathcal{P} : \boldsymbol{\xi}(t) \mapsto \boldsymbol{\xi}(t+T). \tag{3.14}$$

Here $T = \frac{2\pi}{\Omega}$ is the time period of the forcing function. Any initial condition on a limit cycle (or any periodic solution) with a fundamental frequency that is an integer multiple of Ω is a fixed point of the Poincaré map $\mathcal{P}$. The stability of a limit cycle is characterized by the stability of the corresponding fixed points of the Poincaré map (3.14). The Jacobian of $\mathcal{P}$ evaluated at a fixed point $\boldsymbol{\xi}^*$ will be denoted by $\boldsymbol{J}_p(\boldsymbol{\xi}^*)$ and its eigenvalues by η, and is calculated numerically using finite differences.

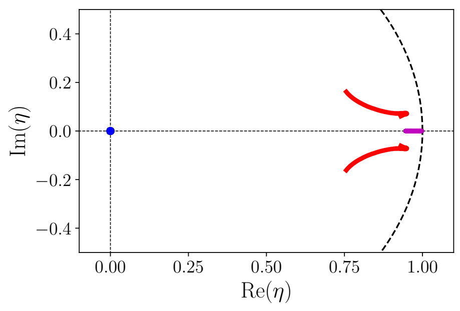
The stability of the fixed points of $\mathcal{P}$ depends on the forcing frequency Ω, which acts as a bifurcation parameter. Figure 3.7(a) shows the magnitude of the eigenvalues of $\mathbf{J}_p(\boldsymbol{\xi}^*)$ for $\boldsymbol{\xi}^*$ on the buckled limit cycle with $A = 1$, and fig. 3.7(b) shows the same eigenvalues in the complex plane. For these parameters all the eigenvalues of $\mathbf{J}_p(\boldsymbol{\xi}^*)$ are less than one in magnitude for forcing frequency Ω in a small range around 21, specifically between two critical values of the bifurcation parameter $\Omega = 18.71$ and $\Omega = 24.65$. At these critical frequencies one of the eigenvalues crosses the unit circle along the real axis. Beyond these critical values, the limit cycle becomes unstable and adjacent trajectories are repelled and instead converge to the limit cycle around the non-buckled $\boldsymbol{\xi} = [u_x, 0, 0, 0]^T$ state as show in fig. 3.8. This bifurcation behavior invites the possibility of tuning Ω to attract or repel trajectories from certain limit cycles. Numerical simulations show that this switchable multistability that depends on the forcing frequency Ω exists for a large range of forcing amplitudes and frequencies.

3.4 Conclusion

This paper demonstrates that a mobile mechanical system with a nonholonomic constraint and periodic forcing can exhibit multistable limit cycles in a reduced velocity space. This multistability is achieved in the absence a of multi-well elastic potential function and is the result of the nonholonomic constraint. The different limit cycles correspond to different types of motion or gaits in the plane: an averaged straight line motion and a rapid turning motion. Additionally, the stability of some of these limit cycles can be changed in real time by changing forcing parameters such as



(a)



(b)

Figure 3.7: The eigenvalues η of the T-periodic Poincaré map of the forced and dissipative system performing limit cycle oscillations with varied forcing frequency Ω in a turning gait with $A = 1$, $\epsilon = 0.15$, and $c = 0.001$. Subfigure (a) shows the evolution of the eigenvalues of the Jacobian about the fixed point with varying Ω, and (b) shows the eigenvalue trajectories in the complex plane for the same range of Ω. Bifurcations at $\Omega = 18.71$ and $\Omega = 24.65$ bound a range of forcing frequency which leads to stable limit cycles around the buckled state. Outside of this range, all tested initial conditions were found to converge to straight-motion limit cycles.

the frequency, so switching between gaits can be achieved by controlling the forcing frequency. The significance of these findings is that tunable multistability can be achieved in mechanical systems using the interplay of nonholonomic constraints and forcing, without the necessity of complex geometric or material nonlinearities. This has important implications for the area of mobile robots, particularly soft robots, where such constraints are often ignored. Such constraints can be designed and exploited to easily induce multistability for manipulation or mobility tasks.

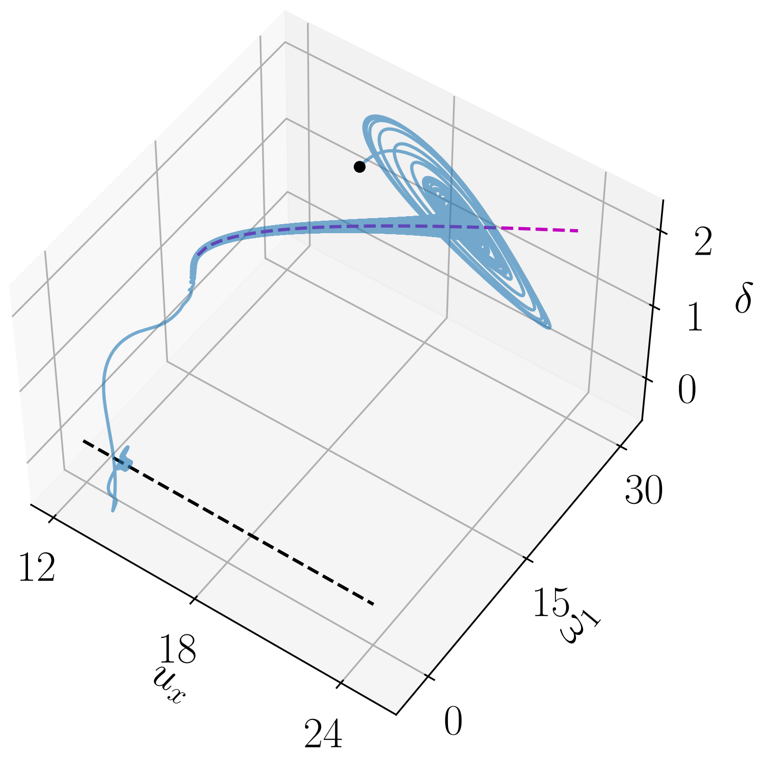


Figure 3.8: A trajectory with $A = 1$, $\Omega = 25$, $\epsilon = 0.15$ and $c = 0.001$, corresponding to slightly above the upper Ω bound for stable limit cycles derived in fig. 3.7(a). Starting at initial conditions indicated by a circle, the trajectory is drawn towards the buckled fixed points of the conservative system, represented by a magenta line. However, in contrast to the stable range in fig. 3.7(a), the oscillations do not provide enough energy to counteract dissipation, and the cycles decay along the fixed point until the fixed point is annihilated in saddle-node bifurcation, at which point it jumps to a non-buckled limit cycle about the $\delta = 0$ fixed point, indicated by a black line.

Chapter 4

Sensing with passive appendages

This chapter is adapted from a paper that appears in *Bioinspiration & Biomimetics*:

C. Rodwell, B. Pollard, and P. Tallapragada, “Proprioceptive wake classification by a body with a passive tail,” Bioinspiration & Biomimetics, vol. 18, no. 4, p. 046001, 2023.

4.1 Introduction

The ability of animals to identify and exploit vortex wakes in water or air is well documented, from the famous ‘V’-shaped formation of migratory geese which allows trailing individuals to derive thrust from the wakes of those in front [106], to the ability of trout in fast-flowing streams to use the wakes of obstacles to allow station keeping with low to zero energy investment, even when the trout in question is dead [107]. Closely related to and aiding the locomotion is the ability of fish to sense and process the spatiotemporal information in the water around them. Objects moving in water or stationary objects in streams perturb the flow in their immediate area, and for a wide range of Reynolds numbers will create a vortex wake. A swimming animal or an underwater robot encountering the wake created by another body experiences disturbance forces and moments. These disturbances can be associated with the disturbance velocity field and the bodies creating them. Essentially, information about fluid flow and the objects that create these flows is encoded in the spatiotemporal evolution of the vortical structures, whether the bodies creating them are cylinders, hydrofoils, underwater robots or fish [108–113]. Many species of fish sense these flow features using their lateral lines, a grouping of mechanosensors utilizing small sensing hairs, as part of their multimodal sensing [107, 114–120]. Researchers have long been captivated by the sensing capabilities of the lateral line and have sought to mimic these. Considerable research and engineering has been devoted to creating artificial lateral lines through a variety of electromechanical sensors such as miniature pressure sensors [121–125], ionic polymer-metal composite sensors [126,127], multi-layered silicon beams [128], and micro-fabricated hot-wire anemometry sensors [129], and these sensors have been useful to perform state estimation [130] and improve the swimming efficiency of robots [123]. The whiskers of aquatic mammals have been shown to serve a similar role in flow detection by translating the fluid flow into detectable whisker vibrations [131].

While these sophisticated sensors mimicking the lateral line have undeniably improved the sensing of the local flow field, alternative sensing abilities that augment the lateral lines in fish are being better understood. The discovery [107] that a dead animal can 'sense' the wake well enough to exploit it, with no sensory input beyond the fluid passively deforming its limp body, offers an indication that passive degrees of freedom may instill this wake sensing ability into an artificial robotic swimmer. Proprioception has been demonstrated to be used by fish as part of their multimodal sensing. For instance, the rays and membranes of fins have been shown to act as mechanosensors in catfish [132], bluegill sunfish [133], and wrasses [134]. Fin mechanosensation has been found to encode the velocity of fin bending as well as respond to cyclic stimuli of biologically relevant frequencies with the mechanosensory system being capable of providing stroke by stroke feedback [135]. Beyond flow sensing, it has been shown that fish can use proprioception to improve their efficiency of swimming with improved energy harvesting from the flow [136], in fact the mere presence of passive tails has been shown to increase the agility of swimming robots [137]. Such recent research in the proprioceptive ability of fish fins suggests that in the context of bioinspired robots, useful information about the flow, and in particular the vortex field around a robot, can be inferred from the kinematics of the robot or a part of it such as its tail, which can passively improve the robot's sensing capabilities in conjunction with existing lateral-line based approaches.

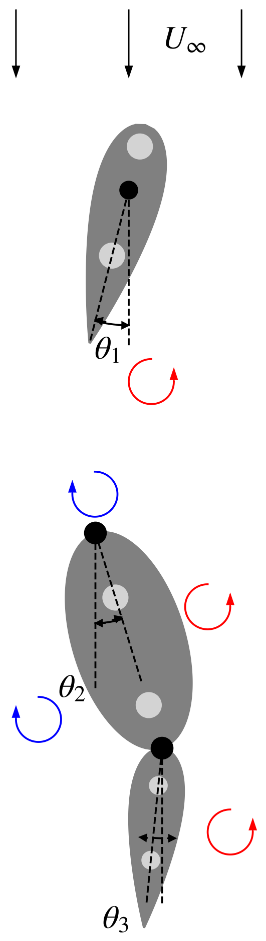
Extracting useful information about the fluid vortex field using even direct measurements of the fluid velocity field is, in general, a non-trivial problem [21, 138–140]; extracting such information using only the kinematic information of a body immersed in the fluid is even more challenging due to the inherent complexities associated with coupled fluid-body dynamics. However, the body immersed in the fluid acts as a reservoir computer, with the input being the hydrodynamic forcing due to a vortex wake and its output being the resultant kinematics. Recent research into the computing power of dynamic systems in the context of reservoir computing [141, 142], finds that applying complex time-series information as forcing to a complex nonlinear system (referred to as a 'reservoir') can result in information about the original system being encoded in simplified form in the states of the reservoir. In previous work [143] we introduced a rigid hydrofoil which is pinned at its leading edge in the vortex wake of a pitching upstream body as a physical reservoir, and showed that the one degree-of-freedom kinematics of its rotation contains enough information to accurately classify the wake Strouhal number without the need for sophisticated sensors. Here we extend that work by showing that a similarly pinned body with an additional freely rotating tail can serve as a more effective physical reservoir and encode more information about the flow into its two-dimensional velocity kinematics, allowing more accurate classification of wake parameters. Performing classification of the resulting time-series data is a well-researched problem in artificial intelligence with multiple viable solutions. In related prior work [143], a shallow dense artificial neural network (DNN) was used to perform a classification of time-series data obtained from the kinematics of the rigid hydrofoil in a vortex wake. Specialist architectures designed to exploit the specific characteristics of time-series data, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been found to match or exceed the performance of the state-of-the-art non-deep learning algorithms on time series classification [144]. RNNs however are more complex than the feed forward CNNs with many more variables in the architecture leading to classification results that are more difficult to explain. CNNs and DNNs with a simpler architecture suffer less from this problem and can enable future physics informed learning. Therefore in this work we use a multiple-input CNN architecture for feature extraction on the kinematic data before using a DNN to classify flow parameters from those features.

We show that the body in the wake encodes more information in its kinematics and thus is a more effective reservoir if the body has a passive tail. We tested the classifying ability based on four cases of different time series data from bodies in a vortex wake. The first is the angular velocity of the large pinned body (head) with a passive tail, the second is the angular velocity of only the passive tail attached to the head, the third is the angular velocity of a the head and tail attached together in a fixed assembly, the fourth case is one where the angular velocities of both the head and tail are fed as inputs to the CNN, the fifth is where the head is tested without a coupled tail, and the sixth where the tail is tested without a coupled head. Not surprisingly, the classification of the wakes was the most accurate in case four when using two inputs, but very surprisingly in both the second and third cases classification shows a significant improvement over the first case. The mere presence of a passive tail on a body can improve its hydrodynamic sensing ability even if data on the tail dynamics is not directly used in the classification. This result implies that the passive tail modulates the motion of the rest of the body to better encode information about the vortex wake.

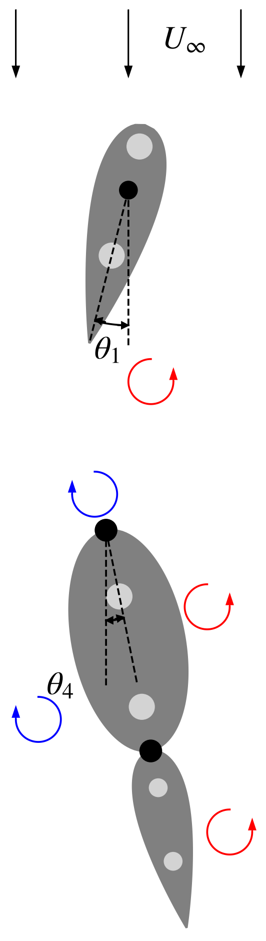
4.2 Experiments

We designed an experimental setup with two hydrofoils placed in a water tunnel as shown in Fig. 4.1(a). The experiments were conducted in an Engineering Laboratory Flow Visualization Tunnel. The water tunnel has a working length of 152 cm and a testing cross-section of 232 cm^2 and is capable of producing laminar flow at speed u_∞ up to 1 m/s. The leading (upstream) hydrofoil has a chord length of 7.24 cm. In the first set of experiments, the downstream body consists of a smaller NACA 0045 airfoil of length 5.39 cm pinned to an ellipse with a major axis of length 7.00 cm and a minor axis length of 5.54 cm, which we refer to as the "pinned assembly". In the second set, the same body is considered but with the pin locked, resulting in a streamlined body that we refer to as the "fixed assembly". In the third and fourth sets of experiments, the downstream body from previous experiments is separated into its two constituent bodies, and each is tested individually. In each experiment, the trailing body is tethered from its leading edge to a bar of extruded aluminum which does not contact the water, so the movement of the hydrofoil is solely the result of its body interacting with the vortex wake. The tether is a lightweight fishing line that is 1 centimeter in length to limit the heaving motion in the trailing hydrofoil, effectively acting as a very low-friction pin. The upstream hydrofoil is actuated to perform periodic pitch oscillations about its centroid by a Spektrum H6210 servo motor controlled by a Raspberry Pi Pico. This generates a reverse Kármán vortex street as shown in the supplementary video 1. Supplementary videos 2-5 show sample dynamics of the pinned-assembly, fixed assembly, the head and the tail respectively. The distance D between the leading edges of the upstream and downstream foils can prescribed by moving the downstream foil's supporting structure. With such an arrangement each experiment is repeated for $D \in \{16.5, 30.0\}$ cm.

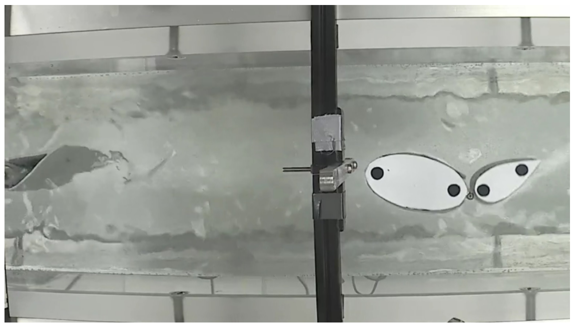
The leading hydrofoil is actuated to execute prescribed yaw oscillations of angular amplitude A and time period T in the prescribed flow of the water tunnel. The trailing hydrofoil is free to



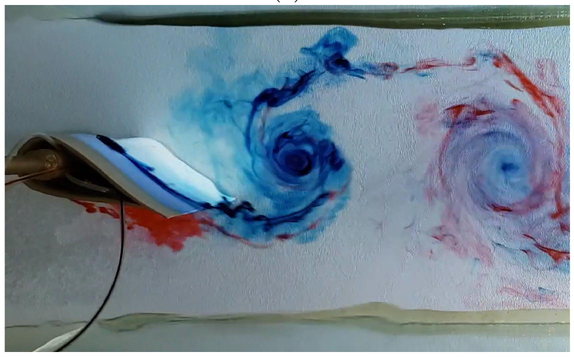
(a)



(a)



(c)



(d)

Figure 4.1: Schematic of (a) a body with two free to rotate segments and (b) a body with two rigidly attached segments in the wake of a forced upstream hydrofoil in a water tunnel with flow velocity u_∞. Both the upstream and downstream hydrofoils are pinned at the black dots with low-friction pins. The two segment body has a second pin connecting the head and tail, where the head is elliptical and the tail is an airfoil. All bodies have circles drawn on their upper surface (in light grey) which are identified by cameras and used to calculate the angles θ_1, θ_2, θ_3, and θ_4 in post-processing. (c) The top view of the experimental setup with a two segment body downstream. (d) A sample dye visualization of the wake generated by the oscillations of the upstream hydrofoil.

execute yaw oscillations in response to the hydrodynamic forcing of the vortex wake created by the leading hydrofoil. This combination of A, T, and u_∞ constitute the control parameters of this experiment.

The oscillations of the leading hydrofoil are programmed to generate nominally periodic motion corresponding to a square wave of angular position. A perfect servo would then generate thin peaks of very high angular velocity, which results in an unrealistic wake. To mitigate this effect, the magnitude of the angular velocity is limited by the controller to 200 deg/s. As a result, the angular velocity takes the form of broad peaks which can be described as the superposition of multiple frequencies (or harmonics). Furthermore, torque constraints and jitter of the forcing servo cause deviations from periodic angular velocity producing small changes in both the amplitude and frequency of oscillations. The angular motion of upstream and downstream hydrofoils is measured using overhead cameras. Multiple circles are drawn on the top of each body (indicated in Fig. 4.1(a) and (b) by pale dots), whose positions are recorded by a video camera at 30 Hz. The positions of the centers of the pale circles are then identified in each frame of the video using a circular Hough transform [145], which are then used to calculate body angles θ_1, θ_2, θ_3, and θ_4. The angular velocities $\omega_1 = \dot{\theta}_1$, $\omega_2 = \dot{\theta}_2$, $\omega_3 = \dot{\theta}_3$, and $\omega_4 = \dot{\theta}_4$ are then computed using first-order forward finite differences. This numerical derivative introduces noise to the velocity data, but this step would not be necessary in an autonomous robot, which would likely have angular velocities calculated by an accelerometer. To preserve as much of the information as possible, no filtering is applied to the data. Figure 4.2 shows a sample time series of these processed kinematics.

Each experiment was performed for a period of 10 minutes, giving angular velocity time series of the same duration at intervals of 1/30 seconds. A discrete Fourier transform is performed on these time series of the angular velocity of the downstream hydrofoil. The calculated peaks in the frequency domain are converted to time periods and shown in Fig. 4.3(a,b). The peaks in this figure correspond to the prescribed nominal forcing periods and their harmonics, though small noise is present about each peak, showing the small variations in frequency. The post processed angular velocity data is broken down into smaller time series, each of duration 5 seconds. The measured amplitudes of velocities of the trailing body are calculated for each of these 5 second windows. A histogram of the overlayed distribution of amplitudes from two experiments measured in this manner is shown in Fig. 4.3(c,d). The amplitudes are broadly distributed within each experiment, and there is overlap between the amplitudes of the two experiments, which indicates that this classification problem is non-trivial.

The angular position θ_1 of the leading hydrofoil with respect to the free stream can therefore be described by the equation

$$\theta_1 = A(t)F(t) \tag{4.1}$$

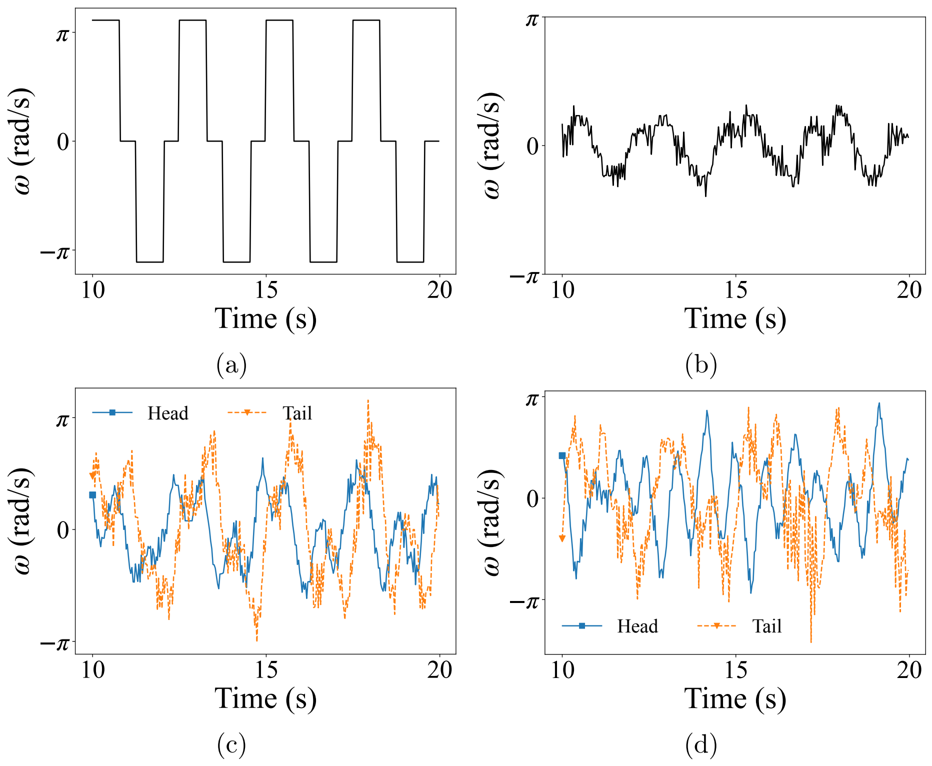
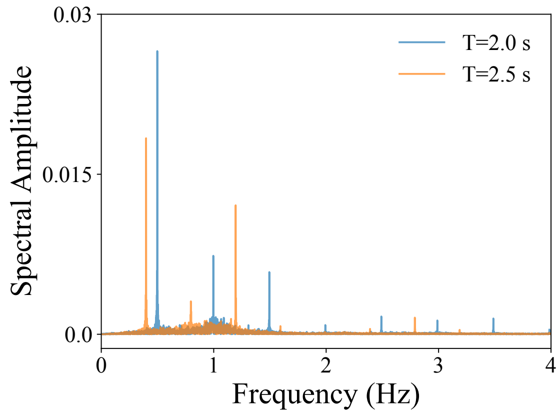


Figure 4.2: The steady-state time series of angular velocities of (a) the upstream forcing body, (b) the single segment hydrofoil, (c) the hydrofoil with a head and a tail segment, and (d) the uncoupled head and tail placed in the water separately, where squares mark the larger head link and triangles mark the smaller tail link. The head and tail respond to the flow differently, with the tail accelerating more sharply and showing larger differences in amplitude between oscillations. The single segment hydrofoil performs oscillations similar to those of the head link of the hinged body, but smaller in magnitude.

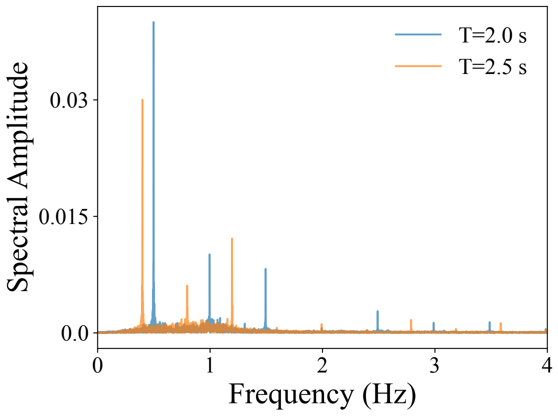
where $A(t)$ is the oscillation amplitude, $F(t)$ is time periodic function of unit amplitude with T as the time period of oscillations. While the actual amplitudes and time periods vary, these are clustered around the nominal prescribed values given by the ordered label sets,

$$\begin{aligned} A =& \{20.0^o, 30.0^o, 40.0^o\} \\ T =& \{1.5, 2.0, 2.5\} \quad \text{seconds} \\ u_\infty =& \{8, 10, 12\} \quad \text{centimeters/second.} \end{aligned} \tag{4.2}$$

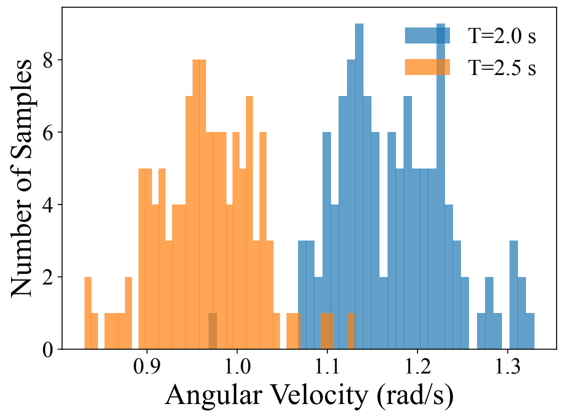
While all of the three parameters above are needed to uniquely specify a wake, a single non-dimensional number reflecting the wake structure is useful to assign as a label to a wake generated



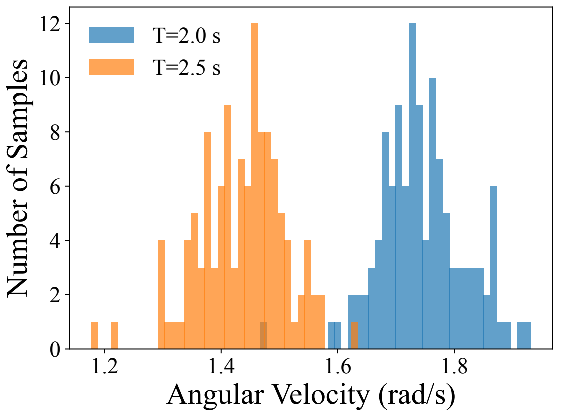
(a)



(b)



(c)



(d)

Figure 4.3: Overlayed frequency spectra of the (a) coupled head and (b) coupled tail angular velocity. The periods are assessed via a discrete Fourier Transform (DFT) of 10 minutes of post-processed angular position at velocity $u = 12$ cm/s. Despite the noisy appearance of the angular velocity time series, the DFT reveals sharp peaks in line with harmonics of the upstream forcing period. This indicates that the angular velocity has encoded information that will enable the period of the upstream body to be classified. A histogram for the root-mean-square value of the angular velocity of each sample is also shown for the corresponding experiments for (c) the coupled head and (d) the coupled tail, where the rms value is computed for each a total of 120 successive 5 second snapshots for each of the two overlayed time series. These results show significant variation between different snapshots of the same dataset, which indicates that features may not cluster clearly and so may not be amenable to traditional classification approaches.

by the leading hydrofoil. The Strouhal number, St, is suitable for this purpose, and is defined as

$$\mathrm{St} = \frac{2f}{u_\infty} L \sin\theta_M \tag{4.3}$$

where $f = \frac{1}{T}$, θ_M is the mean one-sided oscillation amplitude for a dataset, and L is the distance from the center of rotation to the trailing edge of the leading hydrofoil, as shown in Fig. 4.1(a).

The non-dimensional Strouhal number, frequently used in fluid mechanics and specifically in the context of fish swimming to describe the periodic motion of flapping bodies and the associated vortex wakes [146], is a suitable label that encodes information about the motion of the body generating the vortex wake for body Reynolds numbers ranging from 10^3 to 10^8 [112]. The Strouhal number, in this case, contains information about the frequency of the vortex shedding, the distance between the consecutively shed vortices, and the length of the source body that creates the vorticity.

4.3 Wake Classification and Neural Network Architecture

Though multiple minutes of data are available for every wake, our objective is to develop a classifier that can determine changes in the wake in near real time, requiring high performance on only a portion of the available data. Previous work [143] with a different network architecture found that, for a rigid hydrofoil, windows of input exceeding 5 seconds in length do not show significant increases in classification accuracy, so we adopt a 5 second time series, or 150 points at 30 Hz, as the input. We will denote the set of all 5 second windows of time series extracted from the experimental measurements as X. Corresponding to each data window are three known parameters: T, A, and u_∞. The objective of classification is to assign a probabilistic label $\bar{T}$, $\bar{A}$, $\bar{u}_\infty$ and $\bar{S}t$ and compare these with the known values of the labels T, A, u_∞ and St respectively. However, letting the classifier make this assignment directly does not allow the classifier to express uncertainty, and is difficult to train because the gradient of accuracy with respect to classifier parameters is either 0 or infinite throughout the parameter space, because each classification is either correct or incorrect with no partial credit. To generate smoother error gradients, we instead consider estimation of discrete probability distributions. For instance, given a 5 second time series input x_i we estimate a discrete probability distribution $\bar{T}(x_i) : X \mapsto p_t \in \mathbb{R}^3$ such that $p_t(j) \geq 0$ and $\sum_j^3 p_t(j) = 1$. Therefore $\bar{T}(x_i)$ is a probabilistic vector of size 3 (since there 3 labels in the set T) with the jth entry in this vector being the probability that the time period associated with the time series input x_i is T_j. The probabilistic vector labels $\bar{A}(x_i)$ (of length 3), $\bar{u}_\infty(x_i)$ (of length 3) and $\bar{S}t$ (of length 27) are defined similarly. With the classifier output defined as a probability distribution, we can redefine the objective of our classifier: instead of trying to maximize accuracy directly, we attempt to find a classifier that minimizes the cross-entropy

$$e = -\sum_{x_i \in X} T(x_i)\log \bar{T}(x_i) - \sum_{x_i \in X} A(x_i)\log \bar{A}(x_i) - \sum_{x_i \in X} u_\infty(x_i)\log \bar{u}_\infty(x_i) - \sum_{x_i \in X} St(x_i)\log \bar{St}(x_i) \tag{4.4}$$

for every segment of data.

Estimating $\bar{T}(x)$, $\bar{A}(x)$, $\bar{u}_\infty(x)$ and $\bar{St}(x)$ given 150 points of time series data is a general time-series classification problem, and many algorithms exist that can perform such a classification. Recent advances in automatic differentiation have led to the dominance of neural networks for this task, which have recently reached parity with state-of-the-art non-neural network classifiers [144]. In particular, convolutional neural networks are well-suited to time series classification for their tolerance of shifted input [147].

Though CNNs were originally based on the visual cortex of cats and designed primarily in the context of visual processing [147], their properties have recently led to their increased adoption in time-series classification [148]. CNNs apply a constant kernel to sequential windows of data, with each sequential input window mapping to a sequential element of the output. This leads to a tolerance of shifted inputs: a change in the phase of the input layer will only cause a change in the phase of the output layer, a property known as shift equivariance. This property is attractive for this problem because the overall phase of the time series carries no information about which experiment it corresponds to.

Pooling functions are often used to sequentially reduce the data dimension and introduce nonlinearity during each iteration. Here we use max pooling with width two, reducing the dimensionality of the transformed data by half per layer. Convolving and pooling are repeated iteratively until information is condensed into a 'feature vector' of much lower dimension than the input, and the feature vector is typically an input to a dense neural network (DNN) which performs the estimation. The specific network used here is illustrated in Fig. 4.4. We use four convolution iterations with 5 kernels in each iteration, with a kernel size of 7 and step of 1. The dense ReLU-activated neural network has four layers, with 200, 100, 100, and 50 neurons, listed in order from input to output. Directly densely connected to the 50 unit layer are three output layers each with 3 units corresponding to $\bar{T}(x)$, $\bar{A}(x)$, and $\bar{u}_\infty(x)$ respectively, each using a softmax activation function to ensure that the probability vector sums to 1. The Strouhal number St is predicted by a separate network of the same architecture, except with a different output layer corresponding to the 27 labels, also with a softmax function. For the case where time series from both the head and tail are considered simultaneously, a CNN is created for each of the two inputs, with each allowed to have different weights. The two resulting feature vectors are concatentated and input to a single DNN of the same dimensions as above. Evaluating this network for a single time series takes a total of 14 ms on an Intel generation 8 i5 processor, which is fast enough to be run real-time on an onboard microprocessor.

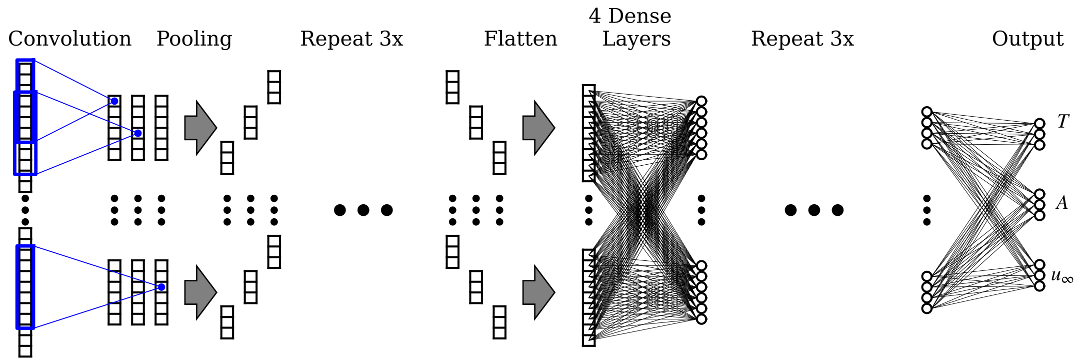


Figure 4.4: The architecture of the classifier for (T, A, u_∞) given a time series of kinematic data.

Because of the limited amount of training data and the complexity of the network it is susceptible to overtraining, so we use a modified version of the early-stopping algorithm [149] which determines a stopping time based on generalization loss and selects the correct network based on its performance on a separate validation data set, which prevents overfitting to the training data. In every iteration, a new 75 data vectors are derived from the designated training data for each experiment, and stochastic gradient descent is used to minimize the loss. The location that these 5 second time series data vectors are extracted from in the longer training time series is random and allowed to overlap with past and future data vectors, so the total number of 5 second windows that can be constructed from 10 minutes of experiment data is vastly greater than the 120 that would be allowed without overlap. Every five training iterations, cross-entropy loss is then calculated for 150 time series from the validation data and the lowest value seen so far, denoted e_i, is stored with its corresponding network weights. The training is terminated at iteration k when $e_k > 1.2\ e_i$, which indicates that the validation error has passed its minimum and overfitting has begun. The network weights corresponding to the lowest validation loss value are taken as the optimal classifier. This methodology has two potential problems that make comparison for the results between different classifiers difficult: it is possible that the stochastic gradient optimization becomes trapped in a local minimum, and noise is introduced into the validation error because the error calculated on a subset of the validation data is similar but not equal to the true validation error over the entire set of possible validation data vectors, which is too expensive to compute. These problems are both mitigated by repeating this procedure 30 times, and selecting the best of the 30 resulting network weights by their performance on a large validation data set of $60,000$ overlapping time series vectors.

To perform the above procedure, three data sets are needed: training, validation, and testing. These must be selected carefully to avoid overlap between windows, as the classifier will likely be more accurate on data that it has already seen, even if only partially, and that could lead to overconfidence in the classifier's accuracy on truly new data. The simplest method to avoid overlap is to split the time series into three portions of different lengths, from which each type of data can be drawn. We designated the first 70% of each experiment as training data, the next 20% as testing data, and the final 10% as validation data.

4.4 Wake Classification Results

Training on the different datasets resulted in different training rates and convergence to significantly different loss values, as demonstrated in Fig. 4.5, where the evolution of the distribution of loss values for the 30 trained networks for each dataset are shown. The fixed assembly data causes convergence to a high loss value, the head and tail data classifiers both converge to a similar medium loss value, and the classifier using both head and tail data as input reaches the lowest overall loss. However, this loss is the sum of the time period, amplitude, and flow velocity losses, and it also gives no indication what specific parts of the parameter space each classifier has difficulty with. To visualize these details, we use confusion matrices.

The performance of each deep network is quantified through a confusion matrix $\mathbf{C}$ whose elements $\mathbf{C}_{i,j}$ represents the fraction of wake i sample that were classified as wake j. The diagonal elements of this matrix $\mathbf{C}_{i,i}$ represent the fraction of correctly classified wakes of label i. By definition $0 \leq \mathbf{C}_{ij} \leq 1$ for all i and j. For an effective classifier, large values are found along a diagonal, and for our axis labeling convention that axis is from the top-left to bottom-right. Values off of this

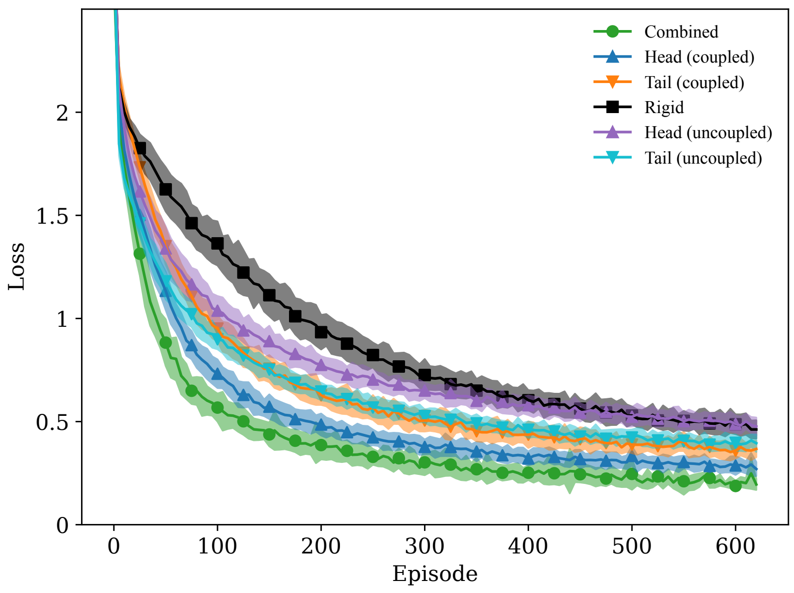


Figure 4.5: The overall cross-entropy loss on the validation dataset by training epoch, which contains the sum of losses on u_∞, A, and T, for the coupled and uncoupled head data (upwards triangle marker), coupled and uncoupled tail data (downward triangle), combined head and tail data (circle), and fixed assembly data (square). On each dataset 30 networks are trained, and the plotted lines represent the mean loss of the networks, surrounded by bands defining one standard deviation distance. The slow asymptotic convergence of the validation loss indicates that overfitting did not occur.

diagonal represent incorrect classification; a large $\mathbf{C}_{ij}$ means that wake label i is frequently confused to be wake label j.

4.4.1 Time period, Amplitude, and Flow Velocity

Wakes can be categorized by the parameters of the motion of the leading hydrofoil generating these wakes. Separate confusion matrices for the classification of wakes based only on the individual parameters of time period T, amplitude A and free stream velocity u_∞ were created. The confusion matrices for the forcing period are shown in Fig. 4.6. Accuracy appears roughly consistent for the different forcing periods, and no individual forcing period has an accuracy of less than 99%. The only bodies to not accurately classify the period to within rounding distance of 100% are the coupled and uncoupled tails, which can likely be attributed to their small inertia and streamlined shape causing excessive sensitivity to components of the wake with frequencies different than the driving frequency. Given the clearly evident forcing period in frequency spectra in 4.3, the upstream forcing period was clearly encoded into the kinematics of the body, so the result that the CNN was able to extract the forcing period from the kinematics with high accuracy is not surprising.

The oscillation amplitude of the upstream hydrofoil is more prone to classification errors, with accuracies for the classification accuracy varying between 78% and 97% as shown in Fig. 4.7. The kinematics of the head body with the attached tail enable better classification for every amplitude than either the uncoupled head or uncoupled tail, indicating that the richer dynamics induced by the coupling between the bodies allows more effective encoding of wake structures corresponding to amplitude into the kinematics. When data from both the head and tail kinematics are used simultaneously, the accuracy improves further for every label, indicating that while the kinematics are coupled, the kinematics of each body still holds information that cannot be easily extracted from the other.

Free stream velocity classification has a higher accuracy than that of classifying amplitude but is less accurate than that of forcing period classification, with Fig. 4.8 showing peak classification accuracy for each dataset falling between 91% and 99%. Similar to the amplitude classification problem, the head kinematics and the tail kinematics yield similar classifications, but the data encoded by each is different enough that the combined classification can greatly outperform both. The coupling of the two bodies yields benefits to both, though is effect is most clear for the coupled head link, which has higher accuracy for all amplitude values than either uncoupled body or the fixed

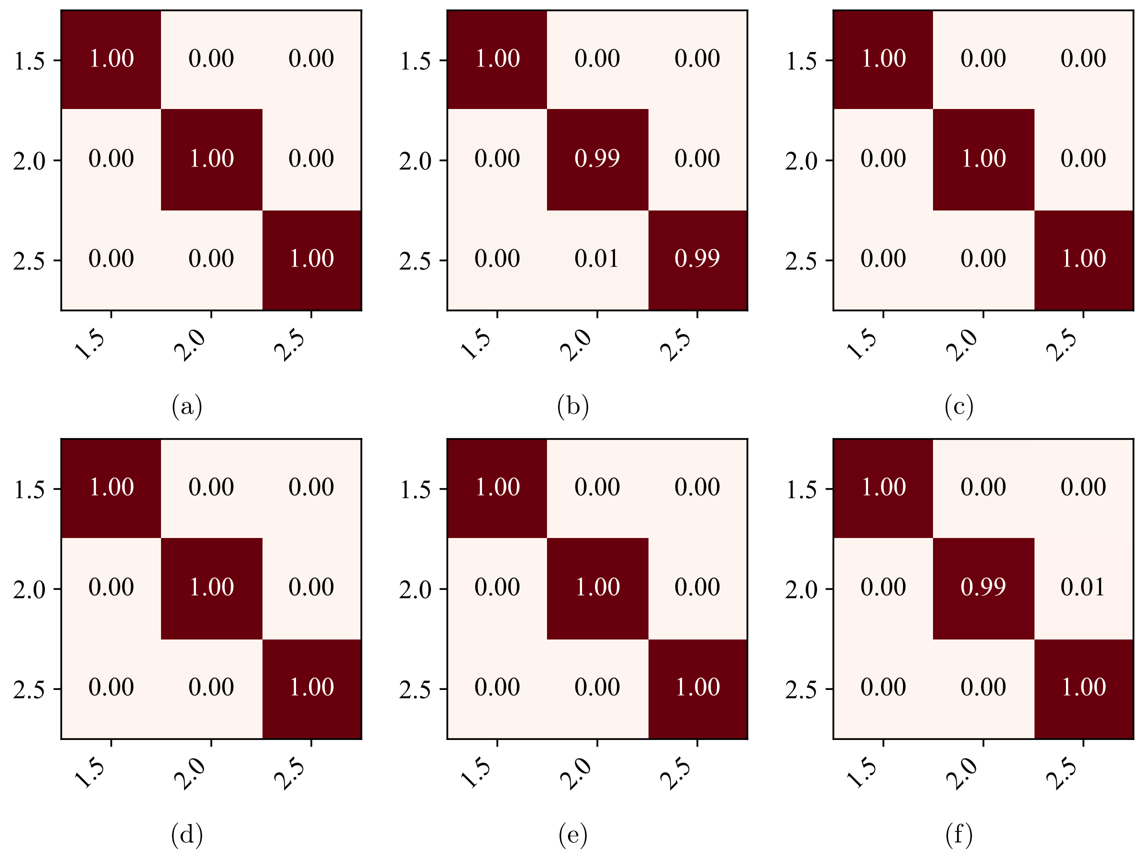


Figure 4.6: Confusion matrices for classifying the time period T, of the motion of the upstream hydrofoil using angular velocity data of (a) only the head and (b) only the tail of the two segment hydrofoil, (c) both the head and the tail of the two segment hydrofoil, (d) using the angular velocity of only the fixed assembly, (e) the uncoupled head, and (f) the uncoupled tail. Accuracy is high for all cases, but the tail segment data, both coupled and uncoupled, appears to have lower accuracy than the other bodies.

assembly. The coupling increases the classification accuracy of the tail on the low amplitude, but decreases it on the medium and large amplitudes. This may indicate that the tail is oversensitive to the flow due to its small mass and sharp edge, and the additional information passed through the coupling is counterproductive when the high amplitude upstream oscillations are already inducing rich tail oscillations. However, when the forcing amplitude and corresponding tail oscillation amplitude are lower, the increased sensitivity to the wake provided by the coupling provides a net benefit.

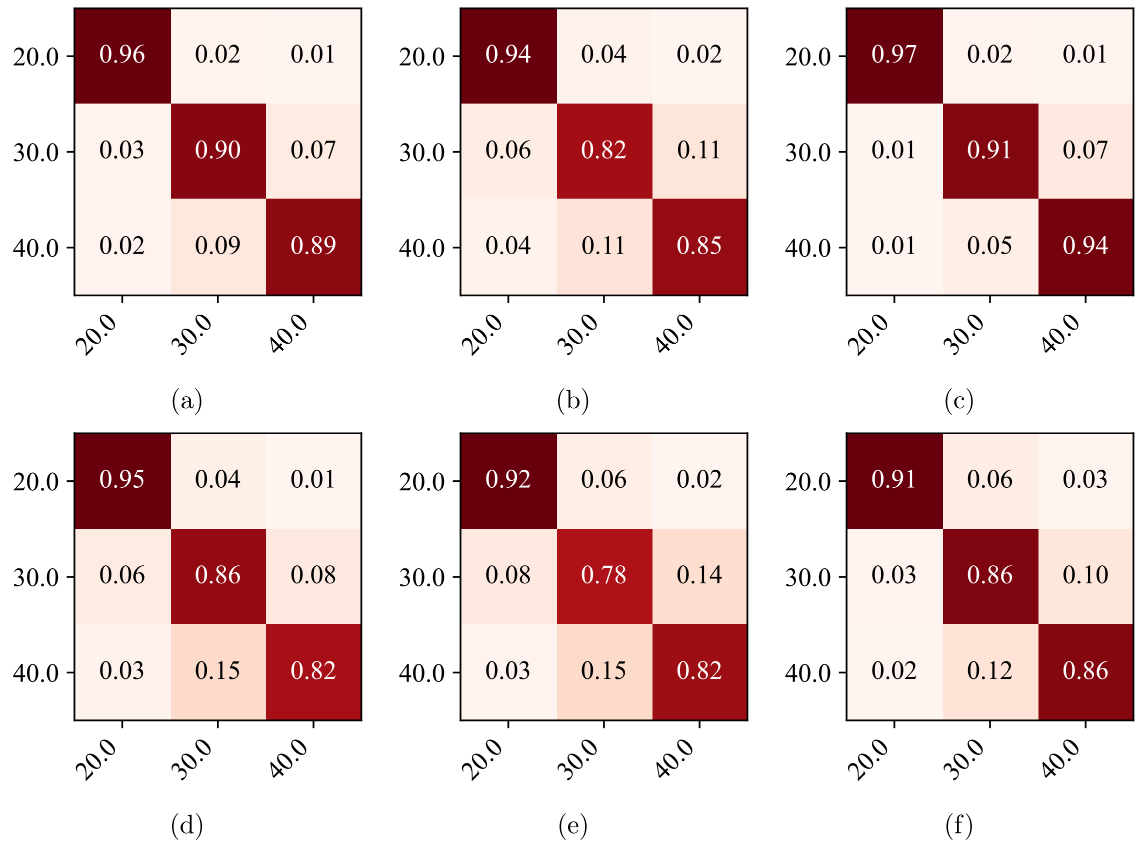


Figure 4.7: Confusion matrices for the forcing amplitude of the upstream hydrofoil in degrees, divided into 3 labels. Kinematic data is derived from from (a) the head and (b) the tail of the pinned assembly, (c) both the head and the tail, (d) the fixed assembly, (e) the uncoupled head, and (f) the uncoupled tail. The higher amplitudes appear more difficult to classify than the lower amplitude, with most mistakes involving mistaking the high amplitude as medium, and conversely the medium amplitude as high.

4.4.2 Strouhal Number Classification

In this problem, the desired information about the upstream hydrofoil is encoded twice: it is first encoded into the wake structure, which then encodes it into the downstream body through a complex fluid-body interaction. In the previous section we demonstrated that different downstream bodies have substantially different classification accuracies when placed in identical wakes, indicating a loss of information in the second encoding step. There is also a loss of information in the first step, because a given wake Strouhal number, which is a scalar quantification of wake structure, cannot be mapped back to a unique combination of parameters for the upstream hydrofoil. Performing a classification of St directly should reduce the loss of information loss in the first encoding step and

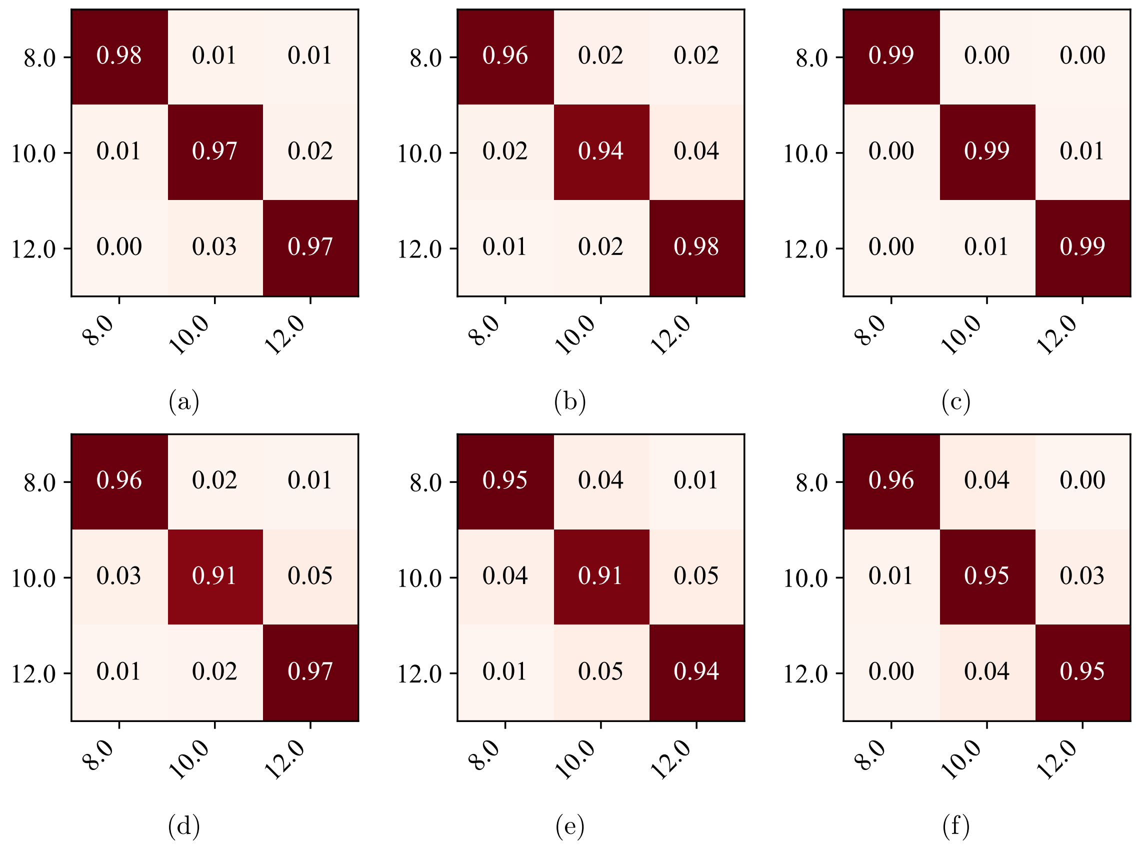


Figure 4.8: Confusion matrices for the free stream velocity u_∞ in cm/s, divided into three labels. Kinematic data is derived from from (a) the head and (b) the tail of the pinned body, (c) both the head and the tail, (d) the fixed assembly, (e) the uncoupled head, and (f) the uncoupled tail. The larger bodies (uncoupled head and rigid foil) appear to be the least accurate at this task, but the coupling appears to improve the classification accuracy from the coupled head to be higher than that of either uncoupled body.

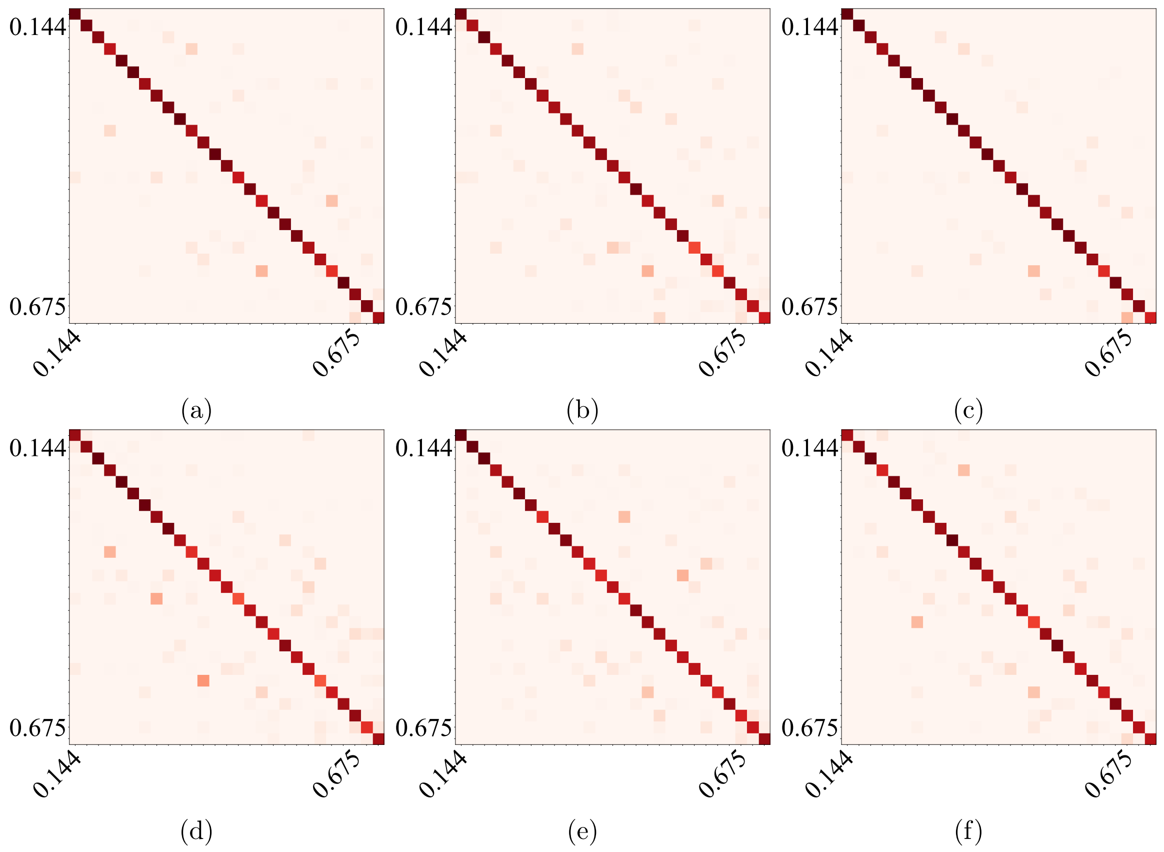


Figure 4.9: Confusion matrices for the Strouhal number, divided into 27 labels. Kinematic data is derived from from (a) the head and (b) the tail of the pinned assembly, (c) both the head and the tail, and (d) the fixed assembly, (e) the uncoupled head, and (f) the uncoupled tail. All cases have a strong diagonal indicating that the classification accuracy is high, though a slight decreasing trend can be seen in the accuracies as the Strouhal number increases.

allow the second encoding step, which is the main interest of this work, to be investigated more directly.

We repeat the classification procedure from the previous section, with identical network hyperparameters (excluding the output layer) and with 30 networks trained per kinematic dataset. As each unique set of (A, T, u_∞) has a unique Strouhal number, there are half as many labels as there are experiment runs (due to the distance parameter D not affecting St), which are not necessarily distributed evenly. Confusion between cases with similar Strouhal numbers but dissimilar underlying parameters indicates confusion due to similar wake structures, and the ability of different classifiers to discern the differences between similar wakes can be observed.

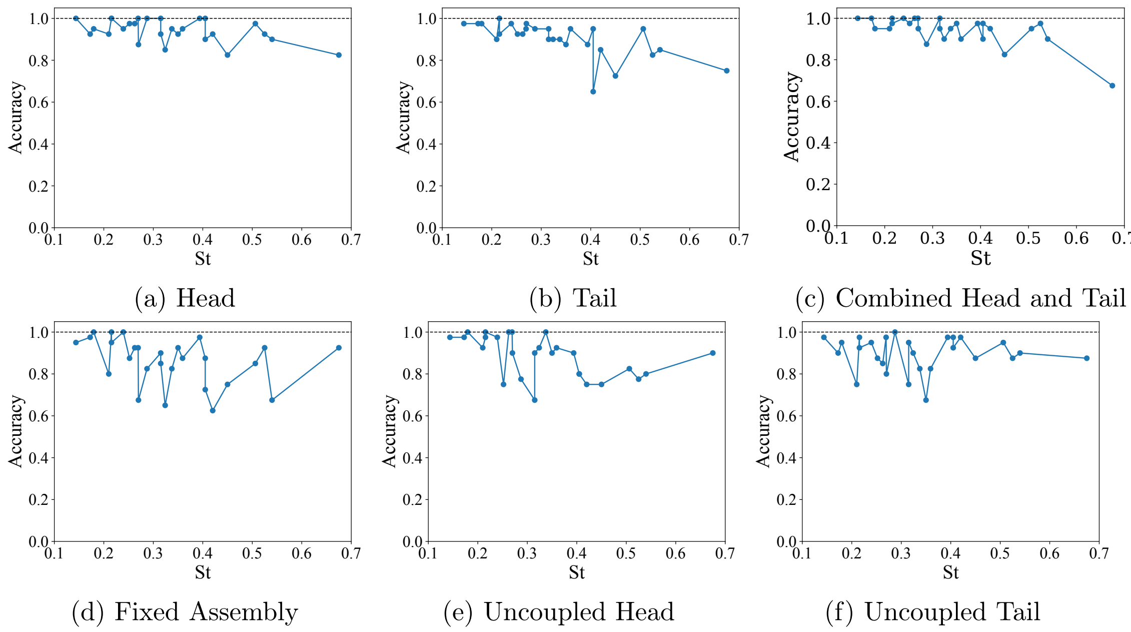


Figure 4.10: The classification accuracy of the Strouhal number determined from kinematic data of (a) the head and (b) the tail of the pinned assembly, (c) both the head and the tail, (d) the fixed assembly, (e) the uncoupled head, and (f) the uncoupled tail. The coupled head and tail both show similar trends, with generally high accuracy for $St < 0.4$, whereas without the coupling, the correlation in accuracies between both bodies is much less clear. For all cases, adjacent Strouhal numbers can have very different accuracies, indicating that the classifier relies on the underlying dimensional features of the wake, and cannot easily classify based on the dimensionless wake structure alone.

The confusion matrices for the Strouhal number classification are provided in condensed form in Fig. 4.9, while the expanded images with exact probabilities are given in Appendix B as Figs. 1- 6. A more compact representation of the classification is via the accuracy defined as the diagonal elements of a confusion matrix. This accuracy of the Strouhal number classification for all the 6 cases, shown in Fig. 4.10, is lower than the accuracy of classifying any of the other three parameters, as in the best case identifying the specific experiment run correctly is equivalent to simultaneously identifying all of the other parameters correctly. Additionally, because there are many labels, experiments with very similar Strouhal numbers must be differentiated, which increases misattribution error.

The variation in overall St classification accuracy between the sets of kinematic data follows a familiar pattern: the classification based on the fixed assembly kinematics is substantially less accurate than that based on either the coupled head or coupled tail kinematics, again indicating that the existence of the tail passively improves the encoding of wake data into the head kinematics on average. Additionally, the coupled head kinematics enable higher accuracy than either the uncoupled head or tail, indicating the positive effect of the coupling.

The results shown in the confusion matrices are using the best neural network, selected from the 30 networks trained from random weights on the each of the kinematic datasets. To represent and rank the results of the classification for such a large number of networks more compactly we choose a single number: *the average accuracy* for each confusion matrix generated from each neural network. The average accuracy is merely the average value of the diagonal elements of a confusion matrix, and the mean of the average accuracy of the T, A, and u_∞ confusion matrices is used to select the representative 'best' overall network, which was used to generate the results in Figs. 4.6 - 4.8. This average accuracy of classification of time period, amplitude, free stream velocity and Strouhal number by each of the networks is shown by a scatter plot in Fig. 4.11. With this measure, the classification results, provided in Fig. 4.11 show that every network yields an accuracy of at least 79% for any parameter on any of the downstream bodies. The accuracy does vary substantially depending on the parameter measured: all of the kinematics can be used to classify frequency with accuracy greater than 98% and flow velocity with accuracy greater than 91%, but the lower bounds for the accuracy in classifying forcing amplitude is 79% and for Strouhal number it is 77%. The high accuracy of the frequency estimation was expected because of the efficient transmission of frequency information by the wake between the forcing hydrofoil and the forced hydrofoil: the forcing hydrofoil sets the dominant frequency of the wake, which in turn sets the dominant frequency of the downstream hydrofoil. This dominant frequency can be seen clearly in Fig. 4.3. By comparison, the encoding of A and u_∞ on the wake manifests itself in a change in wake structure and intensity, which has a far more complex and nonlinear effect on the downstream hydrofoil making it more difficult to classify based on the angular motion alone of the trailing hydrofoil.

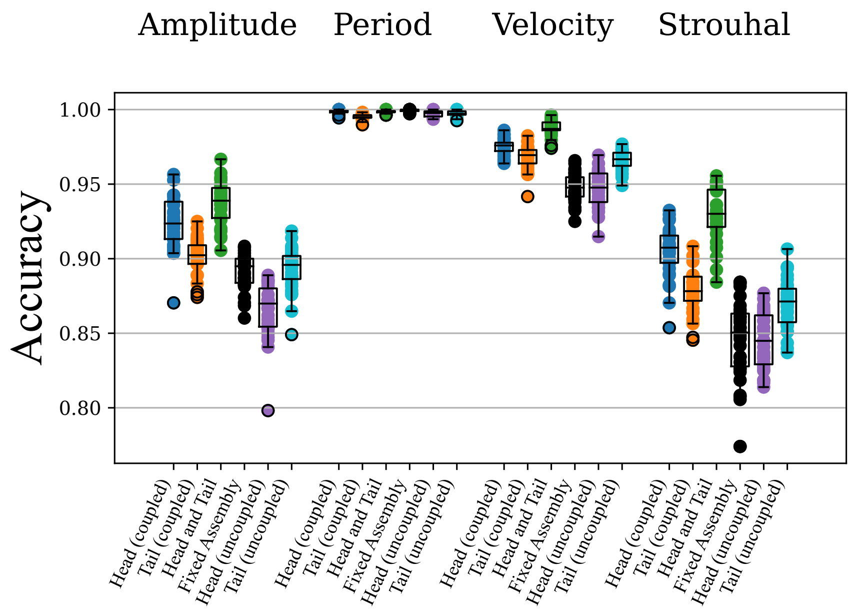


Figure 4.11: The accuracy of each of the 30 trained networks for the coupled and uncoupled head and tail, combined data for the pinned body, and fixed assembly classification tasks on the testing dataset, shown for forcing period, amplitude, flow velocity, and Strouhal number from left to right. Accuracy is defined here as the ratio of time-series vectors for which the highest probability corresponds to the known correct value. The large difference in outcomes for networks trained with the same procedure but different random weights and stochastic gradient descent choices indicates the non-convexity of this problem, with classifiers in some local minima having roughly double the loss of the best found classifier, even trained and evaluated on the same sets of kinematic data.

The significant qualitative result that emerges from the average accuracy of multiple networks is that the angular velocity data of just the head of the two segment hydrofoil encodes more information about the ambient wakes than the angular velocity of a single segment hydrofoil, or of either segment of the two segment hydrofoil when tested individually and uncoupled. Using two time series data, that of both the coupled head and coupled tail, further improves the classification accuracy. We further performed statistical hypothesis testing on the classification shown in Fig. 4.11 to see how likely the improved classification was just a lucky outcome. We assumed the null hypothesis, H_0 to be that "the angular velocity of only a head segment of the pinned assembly does not result in higher average classification accuracy of the wake Strouhal number than does the angular velocity of the fixed assembly". We used the p-value to accept or reject the null hypothesis with the significance level, α set at 5%, which is very commonly used. Using the χ^2 test, the p-value was found to be $p \approx 0$ which is below the chosen α level, therefore the null hypothesis is rejected. The very low value $p \approx 0$ is due to the fact that the classification accuracy of St the best network for each dataset are substantially different, with average accuracy 0.94 and 0.88 for the head data and fixed assembly data, respectively. Combined with a very large number of test data sets $(48,000)$ used to evaluate these accuracies, the probability that the underlying networks are not significantly different becomes negligibly small.

4.5 Discussion and Conclusion

The results in this paper show that the hydrofoil with a passive tail acts as a superior reservoir that generates a kinematic response encoding more useful information about the ambient wake than an equivalent hydrofoil without the additional degree of freedom. This is not just because the additional degree of freedom provides more kinematic information: the mere presence of a tail modulates the response of the head in a manner such that the head's own kinematics encode more information about the wake. The classification results are based on training 30 neural networks, each of which were randomly initialized. The qualitative result, that the kinematics of the body with two segments can better classify wakes, is therefore independent of any one particular network. This result has significance to further understanding the role of passive tail or fin like segments on a fish-like robot; the resulting kinematics of these passive segments can provide useful information about the ambient flow to the robot and enhance real-time multi-modal sensing by underwater robots.

The results in this paper as well as in [143] are valid under the assumption that the the sensing body is directly behind the forced one in the flow. Some lateral displacement (offset) of the trailing body from the center line of the reverse Kármán vortex wake created by the leading body, occurs naturally in the experiments due to the short tether with which the trailing foil is connected to the water tunnel. However, the more general setting where the trailing body has significant lateral displacement from the center line of the vortex wake can make the classification more challenging, and may first require an estimation of the lateral offset distance such as in [150]. While it is likely that passive degrees of freedom will yet confer a sensing advantage in that setting, we leave further investigation to future work.

Future work on sensing and classification of wakes and identification of flows in water can be in the direction of combining models and data driven methods using operator methods such as in [150], physics informed machine learning by making use of concepts like Local Interpretable Model-Agnostic Explanation (LIME) [151], Layer-wise Relevance Propogation (LRP) or Taylor Decomposition [152], where the parts of the signal and their combination that led to the classification and the evolution of the weights and layers in the network can be identified. Uncertainties (epistemic and aleatory) due to real flow conditions can be better handled by Bayesian neural networks (BNNs) and the work in this paper can be a starting point towards such Bayesian classification under uncertainties.

Besides the aspects of machine learning, the problem of proprioceptive wake identification and similar sensing problems can lead to questions and a verifiable pathway for experimental investigation of the role of the shape, placement and stiffness of tails or fins of a fish. Such structural or morphological aspects have usually been investigated from the perspective of swimming efficiency, speed and agility and less so from their role in sensing flow structures.

Chapter 5

Obstacle Localization using spectral properties of the Koopman operator

This chapter has been adapted from a paper which is in review at the time of writing:

C. Rodwell and P. Tallapragada, "Localization of upstream obstacles using spectral properties of the Koopman operator," Submitted, 2023.

5.1 Introduction

Closely related to and aiding the locomotion is the ability of fish to sense and process the spatiotemporal information in the water around them. Objects moving in water or stationary objects in streams create a vortex wake. An underwater robot encountering the wake created by another body experiences disturbance forces and moments. These disturbances can be associated with the disturbance velocity field and the bodies creating them. Essentially, information about fluid flow and the objects that create these flows is encoded in the spatiotemporal evolution of the vortical structures, whether the bodies creating them are cylinders, hydrofoils, underwater robots, or fish. Underwater robots that often function with constrained sensing capabilities can benefit from extracting this information from vortex wakes. Many species of fish do exactly this, by sensing flow features using their lateral lines as part of their multimodal sensing [2, 153, 154].

The complexity and high (infinite) dimensionality of fluid flows around a swimmer present significant challenges to emulate fish-like hydrodynamic sensing and extract the relevant information from sensor data of the flow. This particular challenge is not restricted to bioinspired fish-like swimmers, but has been present in the broad areas of fluid flow estimation, model reduction, and active flow control. Proper orthogonal decomposition (POD) [155, 156] and gappy POD [157] have been tools for model reduction in turbulent flows for decades, and have also been applied for unsteady flow sensing past an hydrofoil and estimation of surface pressure [158, 159]. Model reduction of complex flows using the Koopman operator approach has extended the POD approach to a dynamical systems framework [160, 161]. Subsequent developments in the application of machine learning in dynamical systems have created algorithms for learning the dynamic modes or Koopman modes of a dynamical system from often sparse data [162–164]. Similar methods combining machine learning with dynamical systems are increasingly playing an important role in model reduction in fluid mechanics [143, 165–171]. Flow estimation in the near field of a body by selecting from known fluid DMD modes using surface pressure data has been studied in [172], and incorporating traditional filtering into this approach was recently shown to allow updating the estimation in real time [173,174].

This paper considers a different but related problem motivated by underwater robots where on-board sensors such as inertial motion units (IMUs) and pressure sensors can measure only dynamic and kinematic variables of the robot itself or pressure on the surface of the robot but not measure the ambient pressure and velocity fields. We consider the problem of the estimation of the spatial location of an upstream obstacle in a flow past a pitching hydrofoil. It is assumed that pressure on the surface of the hydrofoil can be measured at a small number of fixed locations on the body. Using time series pressure measurements on the surface of the hydrofoil, a Koopman operator is constructed that propagates the snapshots of pressure data forward in time, thereby encoding the system dynamics. Multiple approaches are considered to extract the encoded information for use in estimating the position of an upstream obstacle. These include the 'direct mode estimation' approach, where the most important modes (eigenvectors) of the operator are into a dense neural network (DNN), the 'spectral image estimation' approach where the spectrum of the operator is extracted and input into a convolutional neural network (CNN), and comparing the unknown modes with known training modes in the 'mode-kernel estimation' approach. This is benchmarked against the Time CNN [148], a recent CNN architecture designed for classification of multi-variate time series, in the 'CNN-based estimation' approach.

The remainder of the paper is organized as follows. In section 5.2, we define the exact fluid-interaction problem considered, and discuss its implementation in simulation. In section 5.3 we review the theory behind standard and exact DMD and its connection to the Koopman operator and their relevance to the estimation problem are explained in section 5.4. The training speed and accuracy of the estimation methods are investigated on the training data in section 5.5.

5.2 Problem Definition

We consider the problem of flow past a symmetric NACA-0018 hydrofoil of unit chord length, representing a streamlined swimmer, pinned at its leading edge with a linear spring of stiffness $k = 6$ Nm/rad and damping coefficient $c = 2$ Nms/rad. When the spring is at rest, the hydrofoil is horizontal with the leading edge pointing left. The hydrofoil is immersed in a free-stream flow with with velocity $U_\infty = 10$ m/s. The fluid is of unit density (equal to that of the hydrofoil) and has viscosity is $\nu = 0.001$ m^2/s. A rectangular bluff body of unit height and width 0.1 m is placed upstream of the hydrofoil and disturbs the incoming freestream flow by shedding and unsteady wake in the fluid. This disturbed flow interacts with the downstream hydrofoil, inducing angular motion and a time-varying pressure profile on the surface. The relative position of the two bodies is defined by the tuple (b, d) as shown in figure 5.1. Along the body, 10 pressure sensors are placed that are evenly spaced in the horizontal direction, and record the absolute pressure at a frequency of 40 Hz.

This flow is simulated on a two-dimensional domain of width 30 m and height 16 m. The leading edge of the hydrofoil defines the origin of the rectangular domain, and is located 1 m to the left of and vertically collocated with the domain's center. The center of the rectangular obstacle is placed b m to the left and d m above the origin. At the left boundary of the domain, an inlet condition $u = 20$ m, $v = 0$ is imposed. To allow unimpeded exit of the flow at the right boundary, a zero gradient condition is imposed on the velocity, so $\nabla u = \nabla v = \mathbf{0}$. At the top and bottom of the domain, a 'slip' condition enforces that no flow passes through the boundary ($v = 0$) and that there is no shear force at the boundary ($\frac{\partial u}{\partial y} = 0$). On the walls of the plate and hydrofoil, a no-slip condition ensures that the velocity of the flow relative to the bodies is zero at the surface.

The system is simulated in the open-source Computational Fluid Dynamics (CFD) software OpenFOAM® 9. The fluid is modeled by the incompressible two-dimensional Navier-Stokes equations

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)=-\nabla\mathbf{p}+\mu\nabla^{\mathbf{2}}\mathbf{u}, \tag{5.1}$$

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

These are numerically solved with the Pressure-Implicit with Splitting of Operators (PISO) algorithm over finite volumes of fluid. Turbulence is modeled by a Large Eddy Simulation with $k-\omega$ Shear Stress Transport.

The moment f_m on the hydrofoil about the pin is calculated by integrating the pressure on its surface. This moment then allows solving the hydrofoil equations of motion

$$I_p\ddot{\theta}+c\dot{\theta}+k\theta=f_m, \tag{5.3}$$

where θ is the angle of the hydrofoil and I_p is the mass moment of inertia of the hydrofoil about the pin. Through the forcing term and a surface boundary condition, this equation is coupled with the fluid equations 5.1 5.2, resulting in complex fluid-body interaction. This interaction is solved by iteratively solving the flowfield and body acceleration until convergence is achieved. These states are then integrated forward using the forward Euler method with an adaptive timestep Δt selected such that the Courant number $C < 0.85$ and $\Delta t \leq 0.02$. This is repeated until time $t = 50$ s.

The fluid domain is discretized with a finite-volume square mesh, as show in fig. 5.2(a). The mesh consists of two identical two-dimensional meshes stacked on top of each other, resulting in a single layer of volumes. Two layers of mesh refinement are used to improve the simulation fidelity in the region between the bodies and their immediate wake. A further three layers of refinement are applied to capture the flow near the surface. When the hydrofoil rotates, the mesh points in a region of 0.3 m from its surface rotate rigidly with it, while the mesh points greater than 1m from the surface remain fixed. The mesh between these regions deforms while maintaining topology.

5.2.1 Simulations

We use a supervised learning approach to the estimation problem, which necessitates collecting multiple sets of simulations with non-overlapping values of (b, d): one set to train the estimator (the 'training' set), one set to validate how well trained the estimator is (the 'validation' set), and

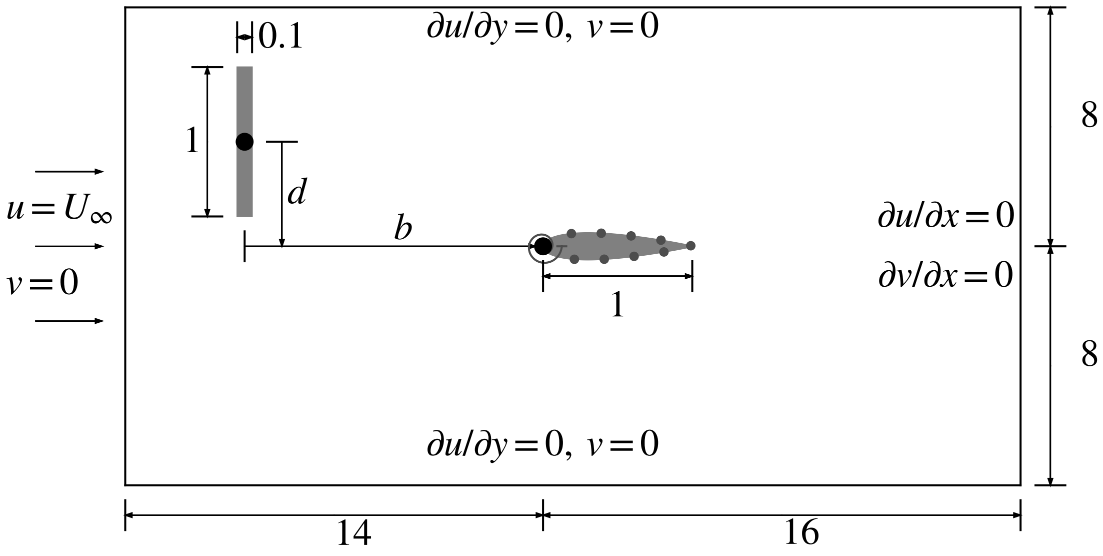
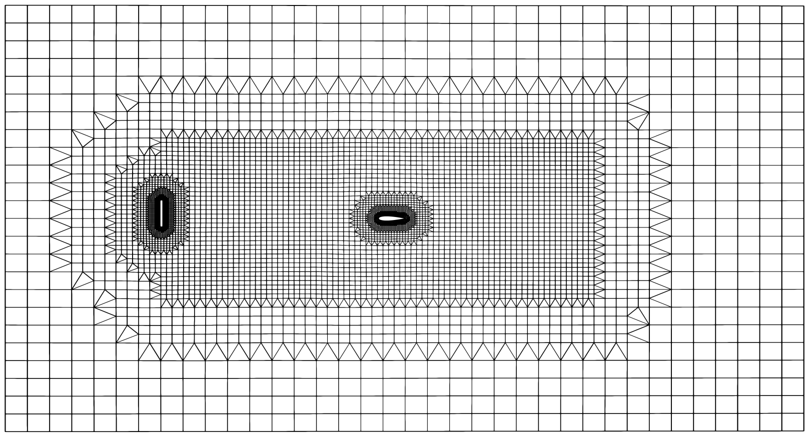

0.1
$\partial u/\partial y=0,\ v=0$
1
d
$u=U_\infty$
$v=0$
b
1
$\partial u/\partial x=0$
$\partial v/\partial x=0$
8
8
$\partial u/\partial y=0,\ v=0$
14
16

Figure 5.1: System diagram (border not to scale). A hydrofoil is pinned with a torsional spring at its leading edge downstream from a vertical plate, with relative position parameterized by b and d. On the hydrofoil, 10 pressure sensors are placed evenly.

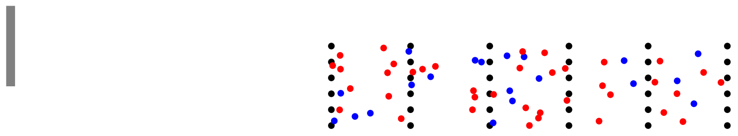
one set to test the accuracy of the resulting estimator (the 'testing' set). The parameter values used for each dataset is show in fig. 5.2(b). The training set has parameters on an even 6×6 grid filling the region $4 \leq b \leq 9$ and $0 \leq d \leq 1$, which is selected because the wake in that region is well-developed and has not fully dissipated. The even spacing of these training points is designed to sample the entire space. However, in practical applications the parameters would likely not fall on a grid, but rather be scattered throughout the acceptable parameter space. For this reason, the validation and testing data are selected from a uniform probability distribution in the region $4 \leq b \leq 9$ and $0 \leq d \leq 1$, with 20 points for validation and 38 for testing.

From each simulation, pressure is known at 15,493 points in the domain, with 80 of those points on the surface of the body. The pressures are interpolated with a spline function to determine the pressures at 10 evenly spaced points, which represent physical pressure sensors and is the only data extracted from the simulation for the estimation procedure. There is no requirement for any knowledge of the flowfield beyond the surface of the body to implement this procedure, either in training or in application.

The pressure field from a simulation is shown in fig. 5.3(a). Regions of alternating low pressure to the right of the flat plate show a $2S$ vortex street. As the vortices interact with the



(a)



(b)

Figure 5.2: (a) The refined computational mesh, with a high density in important flow regions near surfaces and between the bodies. (b) Positions of the pinned leading edge of the hydrofoil relative to the upstream plate for training (black), validation (blue), and testing (red) data sets. Training points are on a grid for even coverage of the parameter space, while validation and testing points are placed randomly.

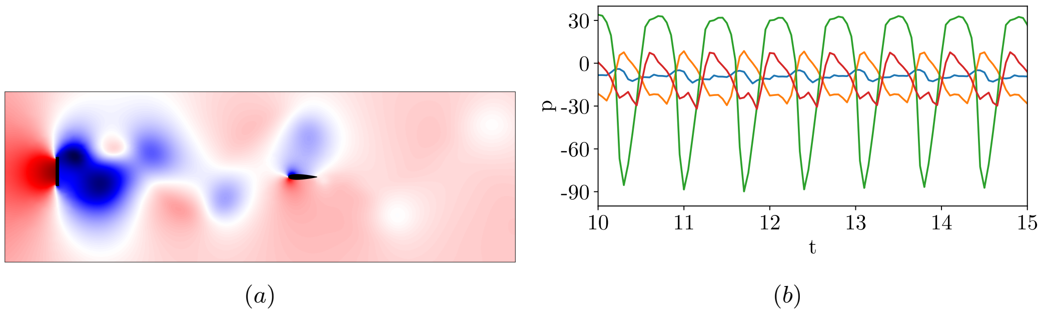


Figure 5.3: (a) The pressure field for a test case where $b = 8.15$ and $d = 0.18$ after the transient period, where red indicates high pressure and blue low pressure. The formation of a periodic vortex wake is evident, leading to a time-varying pressure distribution on the downstream hydrofoil. (b) The pressure at four points in the same simulation on the hydrofoil over time, with blue indicating the trailing edge, orange the top center, green the leading edge, and red the bottom center. The non-zero value of d results in an asymmetrical pressure profile in time.

hydrofoil, a high-pressure region is generated that, when combined with the low pressure inside the vortex, generates a net moment on the hydrofoil and thus motion. This pattern repeats in time, symmetrically on both sides of the hydrofoil for $d = 0$ and asymmetrically for $d \neq 0$. This results in periodic pressures measured on the surface of the hydrofoil, as shown in fig. 5.3(b).

5.3 Dynamic Mode Decomposition

5.3.1 Theory

Consider a body B immersed in a two-dimensional fluid flow, with the pressure known at a number of points on its surface. We assume that the system has no explicit time dependence. While the Navier-Stokes equation governs the fluid flow, we will assume that the flow is spatially and temporally discretized, i.e. the flow domain is discretized by N points $(x_i, y_i) \in \mathbb{R}^2$, and the vector-valued states are $u(t), v(t), p(t) \in \mathbb{R}^N$, corresponding the horizontal velocity, vertical velocity, and pressure, respectively. In the Eulerian approach considered here, each element of these vectors corresponds to a specific point (x_i, y_i) which does not vary with time, though the states themselves do evolve with time. Though the evolution of these variables is governed by a partial differential equation, there exists an unknown flow map F that can predict the evolution of the states after a fixed length of time Δt, e.g.

$$(u_{n+1}, v_{n+1}, p_{n+1}) = F(u_n, v_n, p_n), \tag{5.4}$$

where $u_n = u(n\Delta t)$. It follows that by defining a generalized coordinate $z(t) = (u(t), v(t), p(t)) \in \mathbb{R}^{3N}$, the map can be generalized to

$$z_{n+1} = F(z_n). \tag{5.5}$$

It has been shown [25] that the vector-valued Hilbert space of all functions in L^2 (often called observations) of z, denoted $g(z) \in \mathbb{R}$, can also be mapped forward by an infinite-dimensional operator $\mathcal{K}$ as

$$g(z_{n+1}) = \mathcal{K} g(z_n). \tag{5.6}$$

The operator $\mathcal{K}$ is linear and known as the Koopman operator, named after its inventor who derived it for conservative Hamiltonian systems in [25]. This linearity has the potential to enable the application of linear system analysis techniques (such as modal analysis) to complex and high-dimensional nonlinear systems such as fluids, however, the infinite-dimensional nature of the operator has precluded practical applications. This challenge has been mitigated by the development of DMD, an algorithm that uses a subset of the observations $h(z) \in \mathbb{R}^k$ and $h(z) \subseteq g(z)$ for $k < \infty$ and identifies a linear operator K that minimizes the residual

$$\min_K \sum_{n=1}^{m-1} |h(z_{n+1}) - K h(z_n)|^2 \tag{5.7}$$

over m total snapshots can enable practical linear analysis, while introducing some (often small) amount of error dependent on how effective $h(z)$ is as a basis. In fluids problems where z (typically extracted from discrete meshpoints) is high-dimensional, it is typical for $h(z) \subseteq z$. In extended DMD [175], which is often used in lower-dimensional systems, the lifting $h(z)$ is constructed with a dictionary of lifting functions, such as radial basis functions [175] or neural networks [176].

Here, we consider two different observable functions, resulting in two different operators. One observes all of the field pressures at mesh points, $z_f = h_f(z) = p$, and is mapped forward by the operator K_f. The second observable function $z_s = h_s(z) = p(x, y \in B) \subseteq z_f$ also contains pressures at mesh vertices, but only on the vertices that lie in the downstream body B. As a result, every data point in z_s also belongs to z_f, and the operator on the second observable function K_s acts on a subspace of the space acted on by K_f.

5.3.2 DMD

Here we use the DMD algorithm to approximate the modes of K_s and K_f. Consider first the surface observations z_s, which are few in number. We construct an observation matrix

$$X_s = \begin{bmatrix} z_{s,1} & z_{s,2} & \cdots z_{s,m-1} \end{bmatrix}, \quad Y_s = \begin{bmatrix} z_{s,2} & z_{s,3} & \cdots z_{s,m} \end{bmatrix}.$$

The operator K_s can then be found by recognizing that these matrices can be used to reconstruct the optimization problem in 5.7 as

$$\min_{K_s} |Y_s - K_s X_s|^2, \tag{5.8}$$

which has the convex solution

$$K_s = Y_s X_s^+, \tag{5.9}$$

where + represents the Moore-Penrose Pseudoinverse. The modes (eigenvectors) of the fluid Φ and their eigenvalues Λ are then calculated as the eigenvectors and eigenvalues of K_s

$$K_s \Phi = \Lambda \Phi, \tag{5.10}$$

where

$$\Phi = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_n \end{bmatrix}, \tag{5.11}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_n \end{bmatrix}. \tag{5.12}$$

By convention, the L^2 norm of each component of Φ is unity, $||\phi_i||_2 = 1$. The modes and eigenvalues can be used to reconstruct the pressure field q timesteps after an initial data snapshot $z_{s,0}$ (taken at t=10) by

$$z_{s,q} \approx \Phi \Lambda^q \Phi^{-1} z_{s,0} = \Phi \Lambda^q \boldsymbol{\alpha}, \tag{5.13}$$

where

$$\boldsymbol{\alpha} = \Phi^{-1} z_{s,0} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \end{bmatrix} \tag{5.14}$$

is a vector of complex numbers defining the relative magnitude of the modes, as well as their relative phases. The magnitude of α is roughly equivalent to the concept of 'mode energy' in Proper Orthogonal Decomposition (POD), and in the case that most of the 'energy' is concentrated in a small number of modes, the flow can be reconstructed with high accuracy using only those few modes. Similarly, neglecting modes with small corresponding values of α causes only minimal reconstruction error.

While this approach, often referred to as 'exact DMD', is applicable for the surface measurements because N is small, applying this method on the field data would be computationally intractable because of the size of K_f and the resulting complexity of calculating its eigenvalues. For the field measurements, we use 'Standard DMD', which introduces a degree of truncation to greatly increase the speed of the calculation. Shifted data matrices are first constructed similarly to eq. 5.3.2

$$X_f = \begin{bmatrix} z_{f,1} & z_{f,2} & \cdots z_{f,m-1} \end{bmatrix}, \quad Y_f = \begin{bmatrix} z_{f,2} & z_{f,3} & \cdots z_{f,m} \end{bmatrix}.$$

Instead of directly calculating K_f using the pseudoinverse, first the Singular Value Decomposition (SVD) is computed

$$X_f = U\Sigma V^*, \tag{5.15}$$

where U and V are the left and right singular matrices respectively, and Σ is a diagonal matrix containing the singular values. To improve computation speed, Σ can be truncated starting with its smallest singular values to a smaller $r \times r$ matrix Σ_t, and the corresponding rows and columns of U and V can be removed, resulting in truncated matrices U_t and V_t, respectively. This allows approximating the matrix K_f (in a reduced-dimensional space) as

$$\tilde{A} = U_t^* Y_f V_t \Sigma_t^{-1}, \tag{5.16}$$

and its eigenvalues and eigenvectors are computed as

$$\tilde{A}\Psi_r = \Lambda_f \Psi_r. \tag{5.17}$$

The eigenvectors in the reduced space can be projected back to the full space using the truncated left singular matrix,

$$\Psi = U_t \Psi_r, \tag{5.18}$$

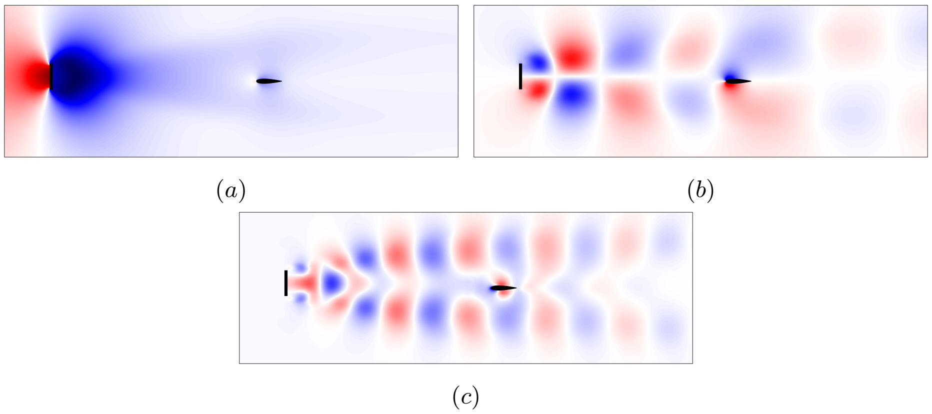


Figure 5.4: Field pressure modes (a) ψ_1, (b) ψ_3, and (c) ψ_5 for $b = 8.15$ and $d = 0.18$. Due to the phase of modes being arbitrary, the red and blue colors are interchangeable, and white signifies a value of zero.

where this Ψ physically corresponds to modes of the fluid field

$$\Psi = \begin{bmatrix} \psi_1 & \psi_2 & \cdots & \psi_r \end{bmatrix}. \tag{5.19}$$

Using a known initial condition, it is again possible to reconstruct the flow field using these truncated modes as

$$z_{f,q} \approx \Psi \Lambda_f^q \Psi^{-1} z_{f,0} = \Psi \Lambda^q \boldsymbol{\alpha}_f, \tag{5.20}$$

where

$$\boldsymbol{\alpha}_f = \Psi^{-1} z_{f,0} = \begin{bmatrix} \alpha_{f,1} & \alpha_{f,2} & \cdots & \alpha_{f,r} \end{bmatrix}, \tag{5.21}$$

and $z_{f,0}$ is a snapshot of the flow field at t=10. We sort the modes of $\boldsymbol{\alpha}_f$ such that they are labeled in descending order of magnitude, $\alpha_{f,1} \geq \alpha_{f,1} \ldots \geq \alpha_{f,r}$, and the components of Ψ and Λ_f are also sorted to be in the same order as $\boldsymbol{\alpha}_f$.

These modes correspond to physical features of the flow, which are illustrated in fig. 5.4.

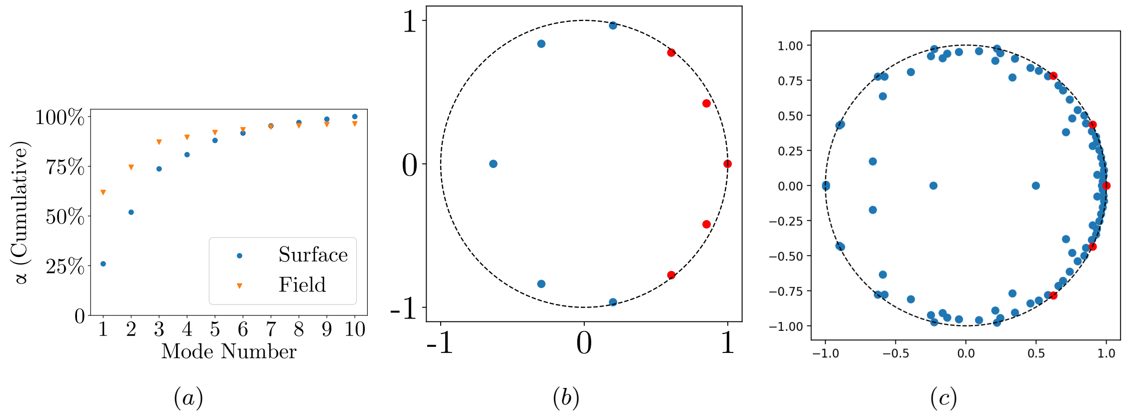


Figure 5.5: (a) The cumulative value of $\boldsymbol{\alpha}$ for the surface modes and $\boldsymbol{\alpha}_f$ for the field modes, showing convergence to greater than 85% of the flow magnitude in the first 5 modes. Eigenvalues (b) Λ and (c) Λ_f of the modes plotted in the complex plane, where the 5 corresponding to the highest magnitudes of $\boldsymbol{\alpha}$ are represented by red circles.

5.4 Estimation Approaches

Surface pressure data from simulations is stored in sets

$$\mathbb{T} = \{T^1, T^2, \dots, T^{36}\}, \tag{5.22}$$

$$\mathbb{V} = \{V^1, V^2, \dots, V^{20}\}, \tag{5.23}$$

$$\mathbb{E} = \{E^1, E^2, \dots, E^{38}\}, \tag{5.24}$$

where T^k is an array corresponding to the kth pair of training parameters, and T^k_{ij} is the pressure at the ith pressure sensor and jth time interval. The arrays V^k and E^k correspond to the validation and training (evaluation) data sets, respectively, and they also contain pressure values of the same dimension and indexed in the same manner as T^k. We define a sampling function S which acts on these sets and returns a random window of 150 timesteps of continuous data from a random element, as well as the parameters (b, d) at that element. For instance, $S(\mathbb{T}) = (T^k_{ij}, b(k), d(k))$ where k is a random variable uniformly distributed over the set $\{1, 2, \dots, 36\}$, i is the range of integers where $1 \leq i \leq 10$, and j is a range $\{j_0, j_0 + 1, \dots, j_0 + 150\}$ such that j_0 is a uniformly distributed random integer in the range $t_0 \leq j_0 \leq t_f - 150$. The initial timestep is chosen as $t_0 = 400$ to minimize the amount of transient information in the sample, and the final time is selected as $t_f = 1000$ to utilize the entire simulation time.

The estimation problem considered here is to construct an estimator that predicts (b, d) as

$$(b_p, d_p) = \mathcal{D}_\theta(\mathcal{F}_\omega(P)), \tag{5.25}$$

$$(P, b, d) = S(\mathbb{T}), \tag{5.26}$$

where $\mathcal{F}$ is a function that extracts features from the time series, and $\mathcal{D}$ is a function that maps those features to an estimate of (b, d). Both θ and ω are sets of parameters (referred to as 'weights') that support those functions. The objective is to select the parameters θ and ω such that the expected value of the mean squared error (L) in the estimation is minimized, or

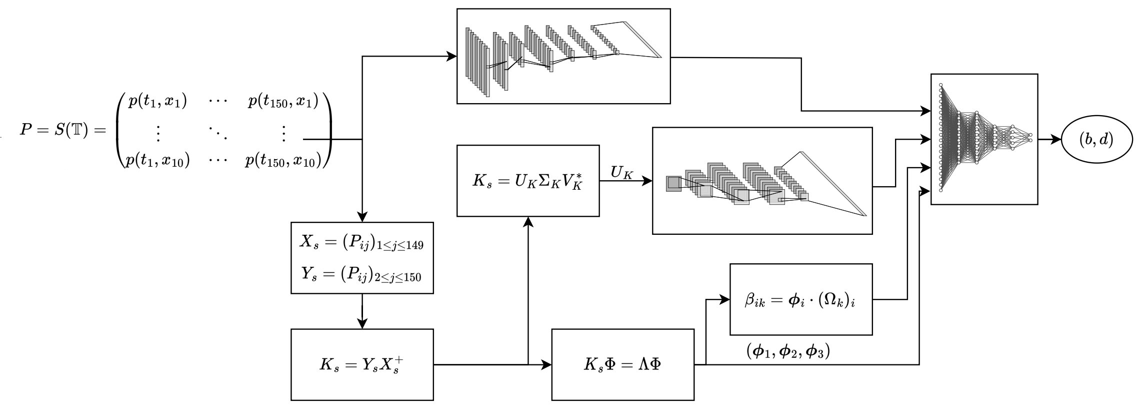
$$\min_{\theta,\omega} L_e \quad \text{where} \quad L_e = \boldsymbol{E}[(b_p - b)^2 + (d_p - d)^2)]. \tag{5.27}$$

Below, four different architectures for the feature extractor $\mathcal{F}$ are introduced, two of which are non-parametric (so $\omega = \emptyset$). After optimizing the parameters for each, the overall estimation error values are investigated to determine which feature extractor is most effective. To isolate the effect of changing $\mathcal{F}$, the architecture of $\mathcal{D}$ is kept the same for all feature extractors, though θ is re-optimized for each $\mathcal{F}$.

The function $\mathcal{D}$ is a DNN, selected due to their general utility in estimation problems. It is fully connected and uses 'ReLU' activation, with sequential layers containing 200, 100, 100, 50, and 50 nodes. The output layer has 2 nodes, representing estimates of b and d, and has a linear activation function.

5.4.1 CNN-based estimation

For time series classification problems, neural-network based parametric approaches have been found to be at the same level as the most advanced non-parametric algorithms [144]. The best suited networks for this task are recurrent neural networks such as long short-term memory (LSTM) networks, and Convolutional Neural Networks (CNNs), which extract features from time-series data by repeatedly applying a single-dimensional kernel. We use a CNN as a benchmark feature extractor, using the hyperparameters determined to be optimal for time series classification in [148], a network architecture known as the 'Time CNN', which has been benchmarked against other time series classification algorithms in [144]. Though we consider estimation problems instead



$P = S(\mathbb{T}) = \begin{pmatrix} p(t_1, x_1) & \cdots & p(t_{150}, x_1) \\ \vdots & \ddots & \vdots \\ p(t_1, x_{10}) & \cdots & p(t_{150}, x_{10}) \end{pmatrix}$
$X_s = (P_{ij})_{1\le j\le 149}$
$Y_s = (P_{ij})_{2\le j\le 150}$
$K_s = Y_s X_s^+$
$K_s = U_K \Sigma_K V_K^*$
U_K
$K_s\Phi = \Lambda\Phi$
$\beta_{ik} = \phi_i \cdot (\Omega_k)_i$
(ϕ_1, ϕ_2, ϕ_3)
(b, d)

Figure 5.6: A schematic of the estimation methods. The top row illustrates the CNN-based estimation approach. The next row down shows spectral image estimation approach. The second to bottom row shows the mode-kernel estimation approach, and the bottom row shows the direct mode estimation approach.

of classification, estimation can in a sense be viewed as a subset of parameter classification, as simply averaging the classification probability distribution on different 'bins' of parameter values yields an estimate. As a result, we expect a network architecture designed for classification to be effective at estimation. In summary, the architecture requires two layers of alternating 1D convolutional neural networks with sigmoid activation and average pooling operations. The filters have length 7, and the pooling operations operate on groups of 3 values, reducing the dimension of the latent space by roughly a factor of 3 with every application. No padding is used on the convolutional steps. The single input time-series vector is split into 6 vectors using 6 separate filters at the first step, and this is increased to 12 vectors for the second step. After the final pooling operation, the vectors are concatenated into a single feature vector of length 168, which serves as the output of $\mathcal{F}$. This procedure is visualized in the top row of fig. 5.6.

5.4.2 Direct mode estimation

Though the CNN can extract features from dynamic data, it uses a black-box approach that does not reveal any specific information about what dynamics it has identified. By contrast, the highest magnitude Koopman modes of dynamic data also form a reduced basis of the flow dynamics, which may also serve as features for further estimation. Further, a connection can be seen between the local modes and the modes of the entire fluid field, as shown in fig. 5.7. Because the modes of the larger flow field are clearly strongly affected by (b, d), this motivates the possibility that significant

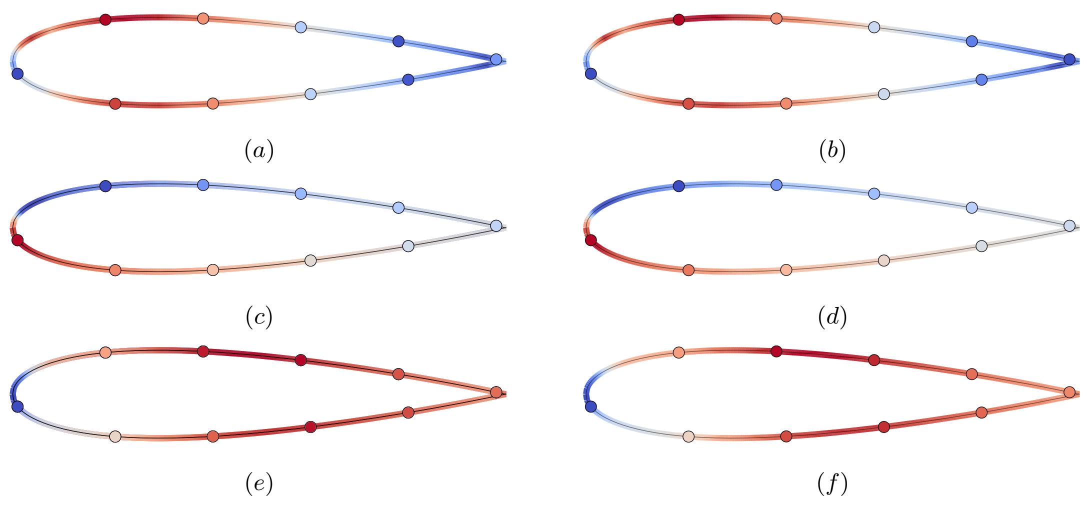


Figure 5.7: (a,c,e) Modes ϕ_1, ϕ_2, and ϕ_3, respectively of the Koopman operator computed on the 10 surface pressure measurements, with circles indicating sensor locations. The coloring between the points is determined by considering the same modes with 100 pressure sensors. (b,d,e) Modes ψ_1, ψ_2, and ψ_3 respectively of the Koopman operator of the entire flow field, interpolated onto the hydrofoil surface.

information about (b, d) has been passed from the larger fluid modes to the modes on the surface.

If modes can be identified that exist consistently and with high magnitude throughout the parameter space, it is possible to construct classifiers based on these modes directly. In the present example, there are three modes that have consistently high magnitude and similar frequencies (eigenvalues) throughout the parameter space. The information density of these nodes is shown in fig. 5.5(a), where the first 5 modes (containing to three unique modes and two conjugates) of the surface pressure data contain greater than 85% of the information needed to reconstruct the data. The eigenvalues corresponding to those dominant modes are shown in red in 5.5(b). All of the dominant modes are near the unit circle, indicating that their corresponding modes do not grow or decay rapidly with time. The small number of surface pressure measurements used introduces approximation error that causes slight deviations of the modes from the unit circle; the corresponding dominant modes of the fluid, denoted as red points in 5.5(c), are much closer to the unit circle because of the much higher number of spatial points used in the calculation. The eigenvalues of the dominant surface and field modes have similar phases: one with a phase of zero, one with a phase near 0.45 (corresponding to the vortex shedding frequency), and another with a phase near 0.90 (corresponding to the second harmonic of the vortex shedding).

Because the dominant three modes have been demonstrated to contain the majority of the information needed to reconstruct the flow, they are selected as features of the data, to be input into the DNN which performs the final estimation. However, to ensure performance, they must first be standardized. This takes two forms: standardizing their order, and standardizing their phase. The motivation for standardizing the order order is simple: if for one (b, d) pair the first mode input to the DNN corresponds to the zero phase mode, but for an adjacent (b, d) pair the first mode input to the DNN corresponds to the second harmonic, the large difference between those mode shapes due to their representing different physical phenomena conceals the subtle changes in the mode shapes that would be useful in estimating (b, d). More formally, we label modes such that for a mode $\boldsymbol{\phi}_1^1$ with parameters (b, d) and $\boldsymbol{\phi}_1^2$ with parameters $(b+\epsilon_1, d+\epsilon_2)$, for small values of (ϵ_1, ϵ_2) we expect $< \boldsymbol{\phi}_1^1, \boldsymbol{\phi}_1^2 > \approx 1$, and their corresponding eigenvalues $\frac{\boldsymbol{\lambda}_1^1}{\boldsymbol{\lambda}_1^2} \approx 1 + 0j$.

The modes are sorted by first neglecting the complex conjugates, which is done by removing from Φ, Λ, and $\boldsymbol{\alpha}$ all entries with indices i that satisfy $\Im(\lambda_i) < 0$. Modes are then labeled using the procedure

$$\begin{aligned}
\boldsymbol{\phi_1} &= \phi_i \; where \; \underset{i}{\operatorname{argmax}}(|\alpha_i|) \; s.t. \; \angle\lambda_i = 0, \\
\boldsymbol{\phi_2} &= \phi_i \; where \; \underset{i}{\operatorname{argmax}}(|\alpha_i|) \; s.t. \; 0 < \angle\lambda_i \leq 0.6, \\
\boldsymbol{\phi_3} &= \phi_i \; where \; \underset{i}{\operatorname{argmax}}(|\alpha_i|) \; s.t. \; 0.6 < \angle\lambda_i \leq 1.2.
\end{aligned}$$

The values of $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$, and $\boldsymbol{\lambda}_3$ are labeled by the same procedure.

The second inconsistency between the modes that must be standardised is their phases. This is a result of the properties of eigenvectors: an eigenvector scaled by any arbitrary complex number remains an eigenvector. As the modes are eigenvectors, the phase of any individual element of the mode vector is arbitrary, though the relative phases of different elements is not. For simplicity, we discard the phase information and use only the magnitude of the mode elements in the estimation. The feature information is concatenated into a vector

$$Z = \begin{bmatrix} |\boldsymbol{\phi}_1| & |\boldsymbol{\phi}_2| & |\boldsymbol{\phi}_3| & |\boldsymbol{\lambda}_1| & |\boldsymbol{\lambda}_2| & |\boldsymbol{\lambda}_3| & \angle\boldsymbol{\lambda}_1 & \angle\boldsymbol{\lambda}_2 & \angle\boldsymbol{\lambda}_3 \end{bmatrix} \tag{5.28}$$

where $|\phi|$ denotes the elementwise magnitude of ϕ, and as a result each element is a real scalar. The vector Z of length 36 is then the output of the feature extractor function $\mathcal{F}$.

5.4.3 Mode-kernel estimation

Because modes can be interpreted as features of the system that contain a condensed representation of the system dynamics, it is likely that on top of parametric approaches (like DNNs), non-parametric algorithms which compare the test data directly with the stored training data may also be applicable. Many such algorithms exist, but here we attempt a kernel-based approach. For each parameter pair k in the training dataset, sorted benchmark modes $[\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_3]|_k$ are identified using the procedure for exact DMD and the sorting from section 5.4.2, except using the entire simulation data T^k instead of a sample of T^k. These modes are stored in a library Ω such that $\Omega_k = [\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_3]|_k$

The estimation procedure works by estimating the similarity of the modes $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3$ from the sample $S(\mathbb{T})$ with the dictionary of benchmark modes using a kernel function. In this case the kernel function is the inner product, so the similarity between modes is defined as $\beta_{ik} =< \boldsymbol{\phi}_i, (\Omega_k)_i >$, where a value near 1 indicates a high similarity. This similarity can be used to estimate (b, d) through simple algorithms, for example the K-nearest neighbors algorithm. However, because of the proven superior performance of parametric approaches to time series [144] over non-parametric algorithms, as well as keeping consistency with the other approaches, we instead flatten β into a feature vector of length 108 to be output from $\mathcal{F}$ into $\mathcal{D}$.

5.4.4 Spectral image estimation

While in this case it is simple to order the modes to perform a direct estimation as was done in both 5.4.2 and in 5.4.3, that may not be the case for more complex fluid flows where more dominant modes are present, or for larger ranges of the estimation parameters where the modes change enough throughout the range to be not easily recognisable. The need for a human to derive the heuristics used to sort the modes for a given flow is also an obstacle to using this approach on an autonomous vehicle. Here we present a method extract information from K_s without any heuristics or requirement to sort the modes. We first extract the spectra U of the operator, where

$$K_s = U_K \Sigma_K V_K^* \tag{5.29}$$

is the singular value decomposition of K_s. The matrix U_K is then passed into a two-dimensional CNN. Because two-dimensional CNNs are most often used for image classification, this can be interpreted as treating U_K as an image. For consistency, the hyperparameters of this network are chosen to be similar to those in section 5.4.1: the convolutional steps have a filter size of 7, and max pooling is performed with a pool size of 3. However, because the operations are performed on an input array of size 10×10 instead of a vector of size 150, zero padding is required to allow applying two layers of convolution without the input becoming smaller than the filter. For the same reason, only one pooling operation is applied, located after both of the convolutional layers. After the pooling operation, the result is flattened into a feature vector also of length 108, which is then output from $\mathcal{F}$.

5.4.5 Training

Each of the four estimation approaches are trained separately, with unique weights for each. In addition, because of the risk of weight optimization finding local minima with different final loss values, a batch of 10 estimators is trained for each of the estimation approaches, with the training process of each individual estimator termed a 'run'. In total, 40 weight optimizations are performed.

Before training, a training dataset is constructed by evaluating many realizations of the sampling function,

$$(\boldsymbol{P}_i, \boldsymbol{b}_i, \boldsymbol{d}_i) = S(\mathbb{T}) \; \forall \, i \in \{1, 2, \ldots, 3600\}. \tag{5.30}$$

Validation and testing datasets are constructed in a similar manner on $\mathcal{V}$ and $\mathcal{E}$, respectively, with the validation dataset constructed of 400 realizations of $S(\mathcal{V})$ and the testing dataset constructed of 760 realizations of $S(\mathcal{E})$. The training and validation datasets are recomputed for every run, but the testing dataset is only calculated once for consistency. The predicted values are given by

$$(\boldsymbol{b}_{p,i}, \boldsymbol{d}_{p,i}) = \mathcal{D}_\theta(\mathcal{F}_\omega(\boldsymbol{P})), \tag{5.31}$$

$$\tag{5.32}$$

where $(\boldsymbol{b}_p, \boldsymbol{d}_p)$ represent vectors of predicted values of (b, d). The weights are updated such that

$$\min_{\theta,\omega} L \;\text{ where }\; L = \sum_{i=i_0}^{i_f} \left((\boldsymbol{b}_{p,i} - \boldsymbol{b}_i)^2 + (\boldsymbol{d}_{p,i} - \boldsymbol{d}_i)^2 \right), \tag{5.33}$$

which minimizes the error between predicted and actual values of (b, d) in a mean-squares sense.

Instead of performing this summation over the entire set at once, it is efficient to iteratively perform the summation over smaller batches. Here, we chose a batch size of $i_f - i_0 = 50$. This batch-based optimization allows the use of the *adam* algorithm, which calculates the parameter updates at every step using a combination of the gradient of the current batch and the 'momentum' from gradients calculated on past batches. After the entire dataset has been iterated over (known collectively as an 'epoch'), an early-stopping criterion is checked, and either the training stops or continues with a new set of batches.

The early-stopping algorithm used was first introduced in [149]. After every epoch, the loss on the validation dataset L_v is calculated. If $L_v < L_{min,i}$, where $L_{min,i}$ is the lowest validation error encountered so far in run i, then $L_{min,i} := L_v$ and the weights $\sigma_{min,i} = (\theta, \omega)$ are saved before continuing. However, if at any epoch $L_v > 1.3L_{min,i}$ and at least 200 epochs have passed, it is assessed that the network is overtrained and the training run is terminated. The representative weights of the estimation approach, σ_{opt}, are then defined as

$$\sigma_{opt} = \sigma_{min,i_{opt}} \tag{5.34}$$

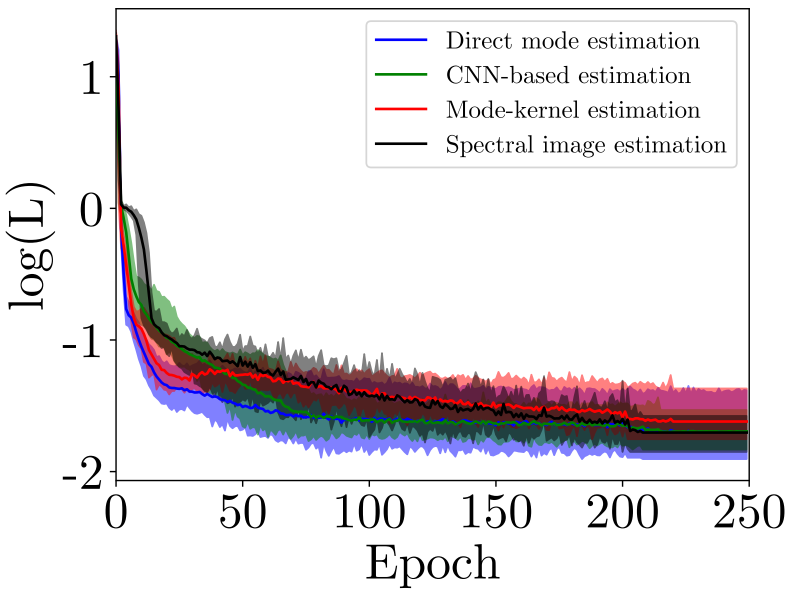
$$i_{opt} = \operatorname*{argmin}_i L_{min,i}, \tag{5.35}$$

which selects the weights form the run with the lowest loss.

5.5 Results

The loss values during training on the validation data for the estimation approaches are shown in fig. 5.8. The mode-kernel estimation approach has the highest average loss after epoch 50, though the lowest mode-kernel estimation approach loss is still lower than the average loss for all of the other three estimation approaches at the end of training, indicating that the overall difference between the estimation approaches is small. The other three estimation approaches have very similar average losses by the end of their training. However, the direct mode estimation approach has the run with the lowest loss by a small margin.

The optimal weights σ_{opt} can be evaluated on the testing data to compare the best estimated values of (b, d) for each approach with the true values. These predicted and real values are shown in fig. 5.9. The number of unique (b, d) pairs in the testing data (38) is much less than the number of



(*b*)

Figure 5.8: Training curve for the four estimation methods. The shaded region represents the range of loss values on the validation data for the 10 networks trained for each method, and the solid lines reflect the average. The mode-kernel estimation approach consistently performs poorly, and other three methods have roughly equal average loss, with the direct mode estimation approach having the lowest minimum loss.

samples evaluated (760), leading to many different estimations of the true value distributed along a vertical line. The CNN-based estimation and spectral image estimation approaches are imprecise in their estimates of d, with a large range of predicted values arising from different samples of data associated with the same true value. By contrast, the mode and kernel-based approaches are precise, with different samples from the same simulation giving similar outputs. This is consistent with the theoretical motivation of the Koopman operator; a given system has exactly one Koopman operator $\mathcal{K}$ that maps its dynamics forward by a specific length of time, which is valid both during the transient period and during steady state. The estimated operator K_s does in practice change slightly depending on which window of data is selected, but is generally much more consistent than the time series itself. The spectral image estimation approach may be inconsistent because the SVD is not unique. We make it unique (to within a sign) by constraining the diagonal of Σ_K do be decreasing from the left to the right. However, this presents a problem for the use of U_K for estimation: very small changes in K_s can result in a reordering of Σ_K, which in turn results in a reordering of U_k. This reordering is likely responsible for the large range of estimates for d.

More quantitatively, differences between the estimation methods can be investigated by considering the loss on the testing dataset. The CNN-based estimation has the highest loss value of 0.0393, followed by the spectral image estimation at 0.0343, the direct mode estimation at 0.0340, and the mode-kernel estimation at 0.0325. The relative accuracy of the estimation methods is inconsistent with that calculated on the validation data; this inconsistency could indicate that different estimation approaches are suited to different regions of the parameter space, which is sampled differently by the testing and validation data due to the random location of samples. This may be particularly true for the kernel method, which may have difficulty with testing (b, d) values that are far from those in the training dataset.

5.6 Conclusion

We have shown that the Koopman operator can extract features amenable for estimation from a dynamic system as effectively as a state-of-the-art black-box convolutional neural network feature extractor. Multiple approaches to extract features from the Koopman operator are considered, and and both directly inputting the modes to a DNN and inputting the spectrum of the operator into a CNN were found to be as effective as applying a CNN to the time-series data itself. This motivates the use of dynamic mode decomposition in the classification and estimation of parameters for multivariate time series which are generated by dynamic systems. Future work can consider in more depth the relationship between the measurable (surface) modes and the broader modes of the fluid.

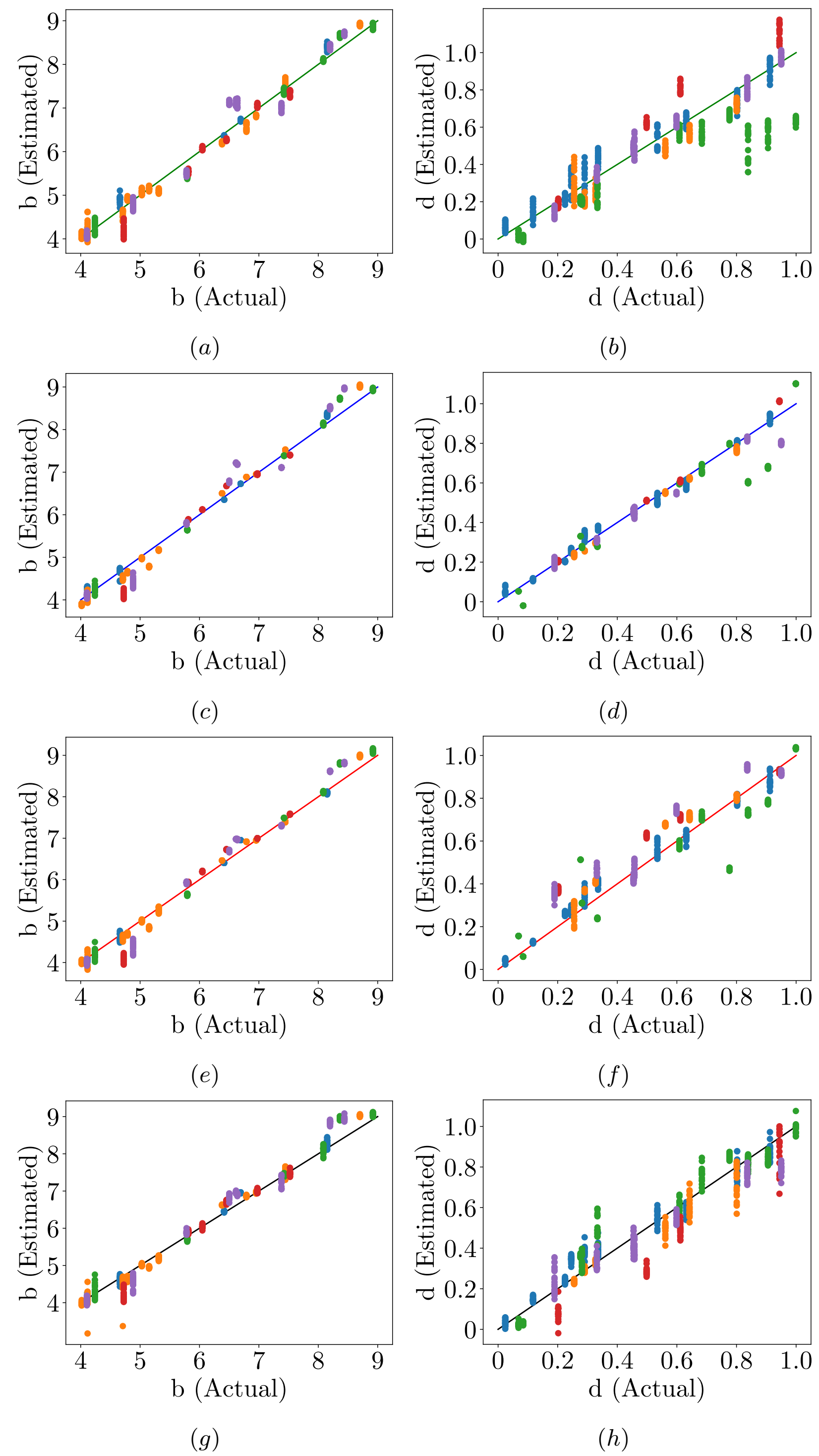


Figure 5.9: Predicted vs. real values of (a,c,e,g) b and (b,d,f,h) d using (a,b) the CNN-based estimation approach, (c,d) the direct mode estimation approach, (e,f) the mode-kernel estimation approach, and (g,h) the spectral image estimation approach. The colors of the dots correspond to the actual value of the other unknown parameter: in the plots for b, blue represents $0 \leq d \leq 0.2$, orange represents $0.2 < d \leq 0.4$, green represents $0.4 < d \leq 0.6$, red represents $0.6 < d \leq 0.8$, and purple represents $0.8 < d \leq 1.0$. For the plots estimating d, blue represents $4 \leq b \leq 5$, orange represents $5 < b \leq 6$, green represents $6 < b \leq 7$, red represents $7 < b \leq 8$, and purple represents $8 < b \leq 9$.

Chapter 6

Flow Field Reconstruction from Surface Measurements in Fluid-Structure Interaction

This chapter has been largely adapted from a paper (in review at the time of writing)

C. Rodwell, K. Sourav, and P. Tallapragada, "Feel the force: From local surface pressure measurement to flow reconstruction in fluid-structure interaction," Submitted, 2023.

6.1 Introduction

A body or structure immersed in a fluid flow is subject to hydrodynamic forces due to fluid-structure interaction. Different flow patterns in general lead to different pressure distributions on the surface of the body, and it is natural to ask if the fluid flow can be inferred solely based on pressure or velocity measurements on the surface of the immersed body. Many marine animals seemingly have an ability to at least sense and localize disturbances, if not generate detailed understanding flow patterns based on non-visual information such as pressure measurements. A well known example of such ability is the schooling of fish [177,178] which requires real-time understanding of the positions and directions of neighboring fish, and can be performed by blind fish using only the "lateral line", a sophisticated line of biological pressure and velocity sensors located along the sides of many fish [116,179]. Inspired by this, much research has been focused on developing "artificial lateral lines," which mimic biological lateral lines using artificial sensors. Some efforts using artificial lateral lines have succeeded in determining specific parameters of the flow, such as the location and movements of sources and dipoles, using analytical approaches by assuming potential flow [17,18] and by employing learning-based methods [19, 150]. In other related work; black-box neural networks have been used to classify and predict wake features using the fluid velocity field around oscillating foils in [20, 21] and variational autoencoders have been used to reconstruct flows using velocity measurements in the flow domain [180]. Shallow neural networks have also been used to reconstruct flows from sensor measurements on the surface of a body for scenarios such as flow past a cylinder such as in [181]. Neural networks have also been used to predict or classify wake features such as the Strouhal number of a wake based on solely on the kinematics of a trailing body immersed in the wake in [143, 182]. Physics-informed neural networks, have also been used to solve the inverse problem of finding the pressure distribution on a body given sparse measurements of the velocity field of the fluid surrounding the body in [183]. However, designing a more general framework for understanding the ambient flow dynamics based solely on surface measurements remains an open problem largely due to the high dimension and unsteady nature of the fluid flow.

The question addressed in this paper then is, "can the flow field around the body be reconstructed knowing only pressure measurements at a few points on a body immersed in the fluid?" We show that this can be done by a combination of dynamical systems tools and machine learning. Instead of directly reconstructing a flow field using black-box machine learning, we first show that the modal decomposition of a sparsely sampled pressure field on the surface of the body can be correlated to the modes of the fluid flow field via supervised learning by shallow neural networks. The full flow field can then be reconstructed using the identified modes. The technique used for the modal decomposition is Dynamic Mode Decomposition (DMD). We demonstrate this using two fluid-structure interaction examples, where pressure measurements on a trailing body in the wake of leading body are used to reconstruct the flow in a domain with a length scale that is many times bigger than the body length.

Dynamic Mode Decomposition (DMD) [184, 185] offers an approach to approximate an unsteady flow by modeling it as a superposition of linear modes. Because many of these modes are often insignificant to the dynamics, only a few modes can typically reconstruct the evolution of the flow with high fidelity, allowing a reduction in the temporal dimension. These modes often have physical meaning, which makes DMD a useful tool for extracting and elucidating dominant flow structures and associated dynamics [184–186]. This low dimensionality and practical usefulness raise the possibility that estimating the DMD modes of the surrounding fluid based on surface pressure measurements may be both tractable and useful.

A procedure for determining the DMD modes around a body based on surface pressures was first presented in [172], where the fluid modes at an unknown Reynolds number were selected from a dictionary of known modes (calculated for a range of Reynolds numbers) based on which modes could most accurately explain the surface measurements. A different approach was considered in [173], where the modes of the flow were known, and the correct superposition of those modes to explain the flow at a given time was selected from surface pressure measurements using a filtering approach. In this work, we instead take a parametric approach that requires no dictionary of modes to operate. This is done by training a neural network that estimates the dominant DMD modes of the flow pressure and velocity, using the DMD modes calculated only using pressure measurements on the surface as input. The idea of using a neural network with DMD modes to reconstruct the fluid field has been explored very recently in [187, 188]; however, these works use autoencoders to map the velocity of the entire fluid field data to a latent space, perform DMD in the latent space, and map the results back to the same velocity field. By contrast, in this paper DMD is performed on the surface pressure or the field velocity and pressure directly, and the mapping is from the surface DMD modes to the DMD modes of the fluid velocity field.

While the primary motivation for the problem investigated is related to sensing by fish-like underwater robots, other engineering applications are possible. Fluid-structure interaction such as in vortex induced oscillations [189], wake-induced vibrations (WIV) and forced oscillations in tandem cylinder arrangements, see for example [190–193] is of critical relevance to structural integrity in aircraft design, ship design, submersible vehicles, offshore structures and heat exchangers, see for example [194–198]. The increasing ubiquity of sensors and computing opens up possibilities for near real time sensing, estimation and control of local flow and structural response. This will require a framework of estimating flow field from onboard structures immersed in the flow.

The remainder of this paper is structured as follows: After the Introduction, Section II delves into the problem setup for wake-induced vibration (WIV) and forced oscillations, elaborating on the governing equations for both fluid dynamics and structural motion. This section also outlines the computational mesh details and studies on mesh independence and validation. Section III reviews the Dynamic Mode Decomposition (DMD) and discusses its relevance in the context of the flow reconstruction. Section IV describes the proposed method employed for flow reconstruction. Section V presents the results concerning the application of DMD and flow reconstruction on both WIV and forced oscillation systems. Finally, Section VI summarizes the key findings of this research in the Conclusions.

6.2 Numerical Simulation of Fluid-Structure Interaction

We consider two examples of fluid-structure interaction. In the first example a circular cylinder mounted on a spring-damper system is free to oscillate laterally in the wake of a stationary square prism. In the second example, two circular cylinders are forced to oscillate out of phase with varying amplitudes, aiming to increase flow complexity. A description of the numerical simulation of the two problems follows.

6.2.1 Fluid-Structure interaction simulations setup

Figure 6.1 illustrates the computational domain setup for the first problem with a downstream circular cylinder placed in the wake created by an upstream square prism. While the circular cylinder is restricted to cross-flow oscillations, the square prism remains stationary. Both bodies have the same characteristic length, 'D,' which equates to the prism's side length and the circular cylinder's diameter. Their center-to-center gap is denoted by the gap ratio (S/D) and is fixed at a value of 5, which is above the critical value suggested in a number of studies [199–202]. All simulations take place within a two-dimensional space and maintain a constant Reynolds number (Re) of 100. The downstream cylinder operates as a one-degree-of-freedom (1-DOF) mass-damper-spring system with a specified mass ratio (m^*) of 10.0 and a damping ratio (ζ) of 0.2. By keeping both Re and cylinder diameter consistent, the reduced velocity, $U^* = \frac{U}{f_n D}$ (with f_n being the natural frequency of the oscillating circular cylinder) is varied across simulations from 1 to 15 by modulating the oscillator's natural frequency.

The computational fluid domain, shown in Figure 6.1, spans a $60D \times 30D$ rectangular area, utilizing a rectangular coordinate system with the origin set at the center of the upstream body. This domain is flanked symmetrically on the top and bottom by boundaries spaced $30D$ apart, resulting

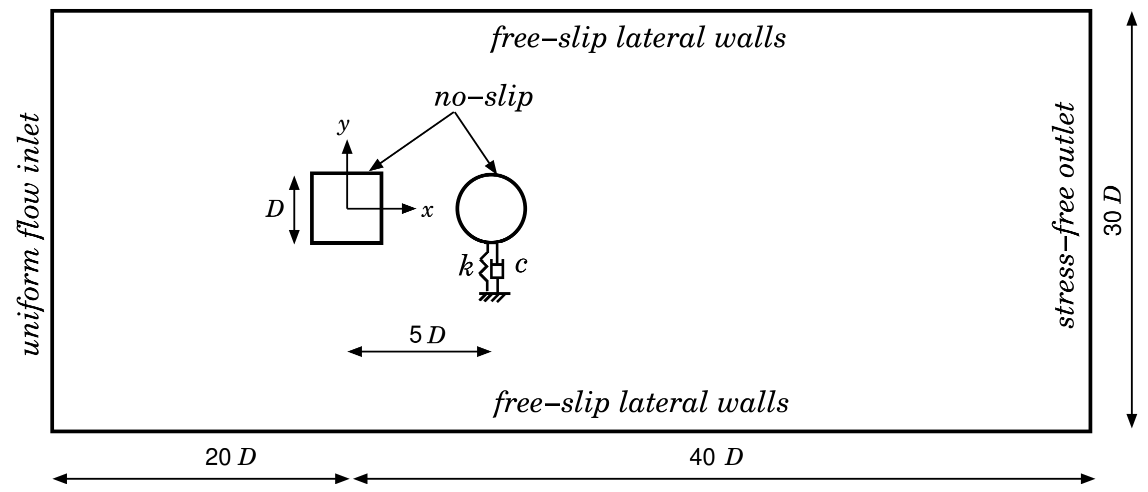


Figure 6.1: Schematic representation of the system setup for the study of wake-induced transverse oscillations of a circular cylinder, positioned $5D$ downstream of a stationary square cylinder. Both cylinders have a cross-stream length, 'D.' The circular cylinder is attached to a linear spring and damper system.

in a blockage ratio (B) of 3.33%. The distance between the lateral boundaries has little impact on the flow field around the cylinders if the blockage ratio is less than 5%, as established by [203–210]. Regardless of changes in cylinder positions or flow attributes, the fluid domain's upstream and downstream limits remain constant in all simulations at $20D$ and $40D$ from the origin, respectively. The cylinders' surfaces observe a no-slip condition, ensuring no relative motion between the fluid and the cylinder. The free-stream velocity at the upstream boundary is characterized by $u = U$ and $v = 0$. The downstream boundary enforces a zero gradient for flow velocities, facilitating the smooth exit of the fluid. The top and bottom boundaries employ a slip wall condition, defined by $\frac{\partial u}{\partial y} = 0$ and $v = 0$, mimicking a shear-free environment and mitigating interference with the flow dynamics.

Figure 6.2 is a schematic of the computational domain tailored for the study of forced transverse oscillations of two tandem circular cylinders. Located at the geometric center of the upstream circular cylinder, the Cartesian coordinate system's origin serves as a reference. The cylinders, separated by a distance of $2.5D$ (yielding $S/D = 2.5$), undergo anti-phase oscillations, a setup crafted to accentuate the intricacies of the flow. The oscillation amplitude A is systematically adjusted from 0.1 to 1.0 in increments of 0.1 while maintaining a consistent oscillation frequency for both cylinders at 1.04 rad/s. As the dimensions and boundary conditions of this computational domain mirror those of the free oscillations problem, further elaboration is omitted for conciseness. *For the sake of conciseness, discussions related to independence studies, mesh details, and validation* *will be presented exclusively for the first problem henceforth.*

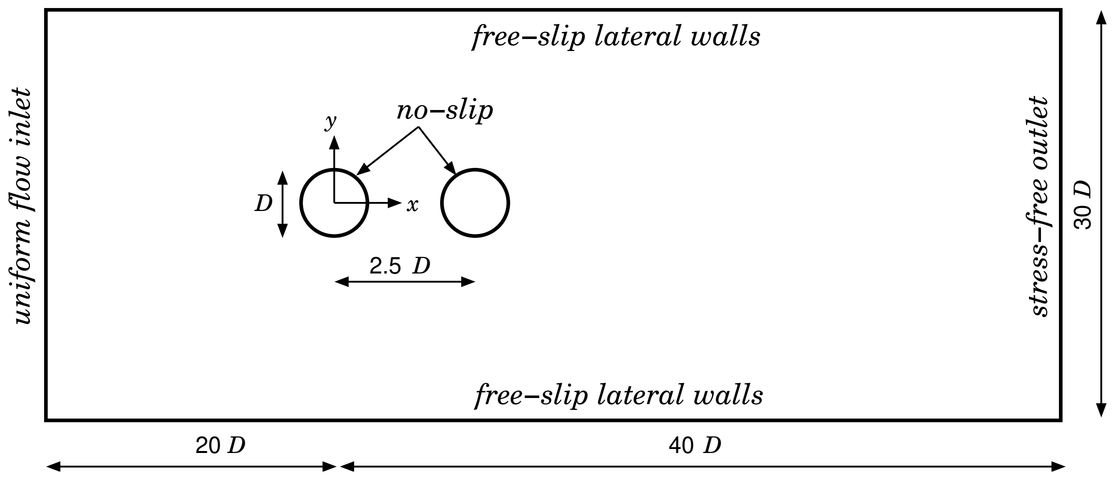


Figure 6.2: Schematic representation of the system setup for the study of forced transverse anti-phase oscillations of tandem circular cylinders, separated by a distance of $2.5D$. Both cylinders have a cross-stream length 'D'. The origin of the Cartesian coordinate system aligns with the geometric center of the upstream circular cylinder.

6.2.2 Governing Equations and Solution Methodology

The computational analyses leveraged two-dimensional direct numerical simulations performed using the open-source computational fluid dynamics (CFD) platform, OpenFOAM, accessible at `www.openfoam.org`. OpenFOAM utilizes the finite volume method for discretizing continuum mechanics problems, including the unsteady Navier-Stokes equations (6.1) and (6.2). These were discretized in conjunction with the Pressure Implicit with Splitting of Operators (PISO) algorithm. A fourth-order cubic interpolation scheme ensured high accuracy for spatial derivatives by discretizing the convective term in the equations. On the other hand, the diffusion term was discretized using a second-order linear scheme. The derivative term's temporal discretization was achieved through a blended scheme combining the second-order Crank-Nicolson scheme and the first-order Euler implicit scheme, providing a high degree of accuracy in temporal resolution while ensuring numerical stability.

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla\mathbf{p} + \mu\nabla^{\mathbf{2}}\mathbf{u}, \tag{6.1}$$

$$\nabla\cdot\mathbf{u} = \mathbf{0}. \tag{6.2}$$

The hydrodynamic forces acting on the cylinder surface are derived directly from solving the Navier-Stokes equations, subsequently triggering the vibrational response of the circular cylinder. The governing equation for the cylinder's cross-flow oscillations can be expressed as

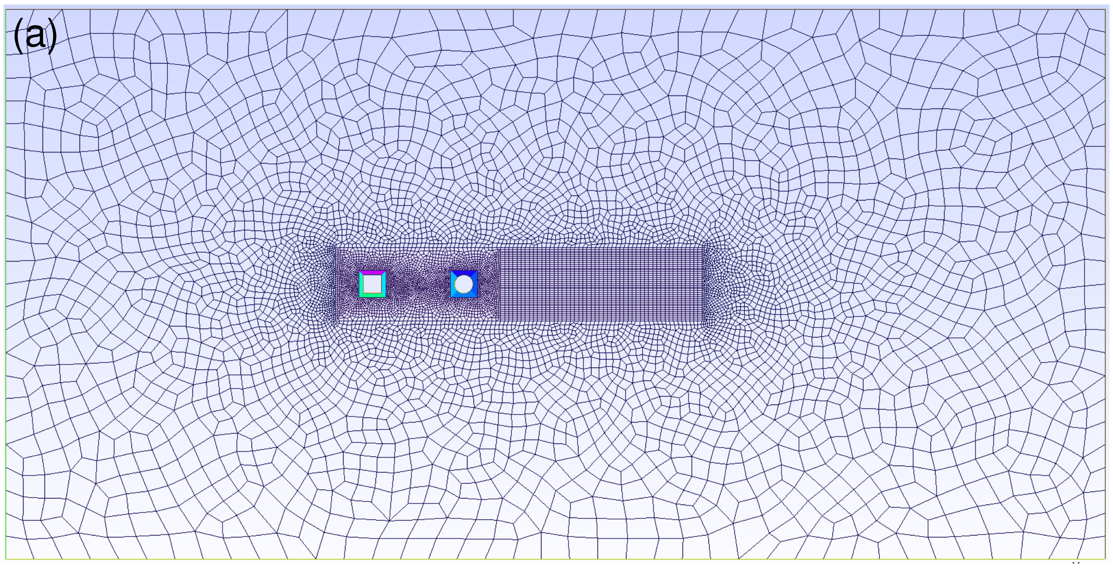
$$m\ddot{y} + c\dot{y} + ky = f_L. \tag{6.3}$$

Here, m represents the mass of the circular cylinder, and $\ddot{y}$, $\dot{y}$, and y respectively denote the cylinder's transverse acceleration, velocity, and displacement. The system damping is designated by c; the spring stiffness is represented by k, and f_L corresponds to the unsteady lift force. This equation integrates the fluid dynamics with the structural dynamics, thereby providing a comprehensive model for the study of the cylinder's oscillatory behavior within the fluid flow.

The simulations were carried out in an iterative manner, alternating between solving for the fluid field and the structural response at each time step. Initially, the velocity and pressure distributions in the fluid domain were determined, followed by the computation of the drag and lift forces through the integration of the pressure and shear stress on the cylinder surface. The calculated hydrodynamic force was then substituted into Equation 6.3, leading to the computation of the cylinder's transverse displacement (y) using an enhanced fourth-order Runge-Kutta method. This displacement information subsequently dictated updating the computational grids, thereby establishing a new mesh for the fluid field calculation in the following time step. This iterative process continued until the system's dynamic behavior reached stabilization and a sufficient number of cyclical results were accumulated. The selected time steps, denoted by Δt, for each case adhered to the Courant-Friedrichs-Lewy (CFL) condition, maintaining a number below 0.85 across the entire computational domain to ensure numerical stability and the accuracy of simulation results. Note that for the free oscillations scenario, both the fluid-flow equations (Equations 1 and 2) and the structural motion equation (Equation 3) are tackled. In contrast, the forced oscillation problem exclusively focuses on resolving the fluid-flow equations because the displacements of both cylinders are prescribed.

6.2.3 Mesh Details

As depicted in Figure 6.1, the rectangular computational fluid domain is discretized using an unstructured finite volume mesh composed of 32,434 nodes. Spatial variation in mesh density is



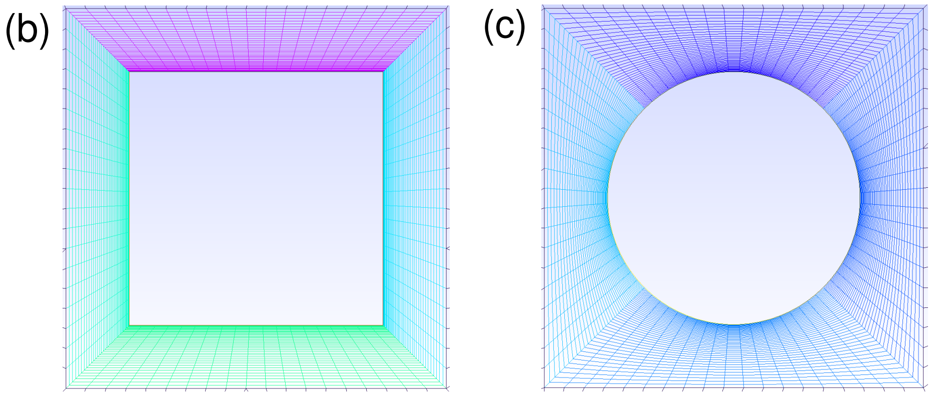


Figure 6.3: (a) The unstructured finite volume mesh in the computational fluid domain of size $60D \times 30D$. The mesh near the cylinders and the wake region is structured. (b, c) Zoomed-in view of the structured mesh near the square and circular cylinders.

employed, with higher density in regions proximal to the cylinders and coarser density towards the domain boundaries, as shown in Figure 6.3(a). Detailed views of this meshing near the upstream and downstream cylinders are shown in Figures 6.3(b) and 6.3(c), respectively, where 80 grid points define each cylinder. The cylinders are encapsulated within their respective square blocks of dimensions $1.5D \times 1.5D$ to minimize projection error during oscillations. Structured, non-uniform meshes discretize the regions between each cylinder and its enclosing square block. The initial grid line from the cylinder surface is set at a distance of $0.005D$ and extends geometrically towards the block boundaries with a progression ratio of 1.05, leading to a line segment between the cylinder and the block consisting of 44 grid points. During oscillations, the square block moves concurrently with the cylinder, preserving the internal mesh structure, while the surrounding mesh deforms in response to the transverse motion of the cylinder. To capture the wake dynamics effectively, an additional finely meshed rectangular region, characterized by dimensions $13D \times 4D$, is established in the wake of the downstream cylinder.

6.2.4 Mesh Convergence and Validation

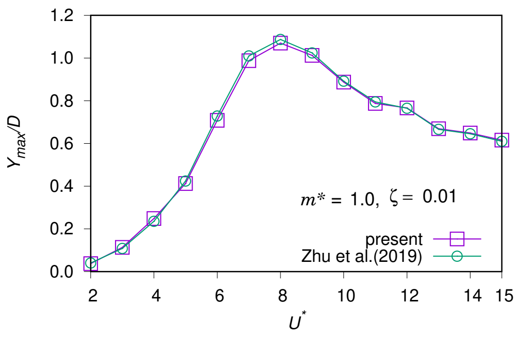
Mesh convergence tests are crucial in computational studies to ensure that the results are sufficiently independent of the mesh resolution. In this study, three different mesh densities, namely M1 (fine), M2 (medium), and M3 (coarse), were utilized. These meshes were evaluated at flow conditions defined by a Reynolds number (Re) of 100 and a reduced velocity (U^*) of 9.

Table 6.1 presents a comparative analysis of characteristic flow and vibration metrics across the three mesh densities. A close evaluation reveals distinct differences between the results from the fine mesh, M1, and those from the medium (M2) and coarse (M3) meshes. Notably, the outcomes from M2 and M3 align closely, demonstrating high consistency. This level of agreement, coupled with the limited deviation between these meshes, underscores the capability of the medium mesh, M2, to strike an optimal balance between computational efficiency and accuracy. As a result, M2 was selected as the preferred mesh for all subsequent computations.

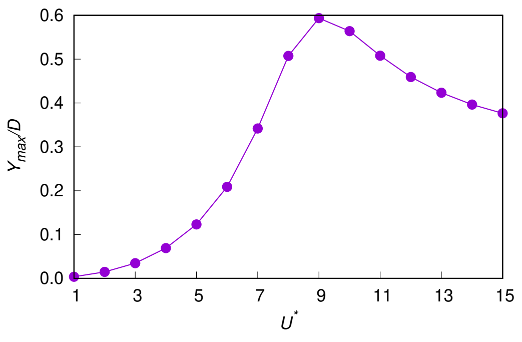
Mesh	Nodes	Y_{max}/D	$C_{l_{rms}}$	$C_{d_{avg}}$	F_y	F_{C_l}
M1	56,814	0.5988	0.6637	0.7037	0.1136	0.1136
M2	**32,434**	**0.5936**	**0.6752**	**0.7122**	**0.1152**	**0.1152**
M3	20,657	0.5346	0.7303	0.7773	0.1237	0.1237

Table 6.1: Comparison of characteristic flow and vibration quantities for three different mesh densities. The data presented is for a vibrating circular cylinder (with a mass ratio $m^* = 10$) that is free to oscillate in the wake of a stationary square cylinder exposed to uniform flow conditions at $Re = 100$ and $U^* = 9$.

The transverse oscillation response of a circular cylinder, normalized as Y_{max}/D, oscillating in the wake of a stationary square cylinder is depicted in Figure 6.4(a). This response is juxtaposed with the findings presented by [204]. The simulations were executed at a Reynolds number (Re) of 100, with the reduced velocity (U^*) spanning a range from 2 to 15. The vibrating downstream circular cylinder maintains a mass ratio (m^*) of 1 and a damping ratio (ζ) of 0.01 for the system. A close examination reveals a commendable concordance between the computed data and the results from [204]. Therefore, our projected response data for wake-induced vibrations of a circular cylinder aligns satisfactorily with the findings of [204].



(a)

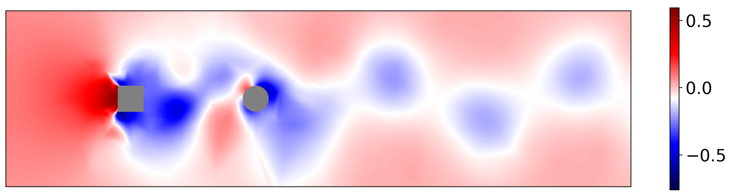


(b)

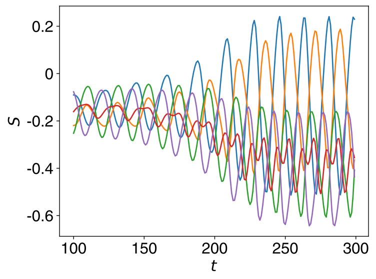
Figure 6.4: (a) Validation of the numerical model for wake-induced vibrations of a circular cylinder situated in the wake of a stationary square cylinder separated by a distance of $5D$. The normalized maximum oscillation amplitude (Y_{max}/D) of the downstream circular cylinder ($m^* = 1$) is compared with that reported by [204]. (b) Wake-induced vibrations of a circular cylinder of $m^* = 10$ at $Re = 100$ and $\zeta = 0.2$: maximum transverse oscillation amplitude (Y_{max}) normalized with D at $S/D = 5$.

6.2.5 Simulation Results

Figure 6.5(a) shows the pressure field derived from the free oscillation simulation. A low-pressure oscillating wake is located behind the square body, which is then advected past the circular cylinder. This results in periodic forcing on the surface of the cylinder, visualized in Figure 6.5(b). Once advected past the cylinder, the wake assembles into an organized vortex street.



(a)



(b)

Figure 6.5: (a) The pressure field for $U^* = 8$ at time $t = 300$, which has settled into periodic vortex shedding. (b) The pressure at 5 evenly spaced points around the cylinder for $U^* = 8$.

6.3 Dynamic Mode Decomposition

6.3.1 Koopman Operator

The Koopman operator calculation here is identical to that done in section 5.3.1, with the exception that we consider two different sets of observable functions. One maps the observations to themselves: $F = h_f(z) = z$. The resulting state vector F contains the fluid pressure, vertical velocity, and horizontal velocity at every vertex of the simulation mesh. The second observable function $S = h_s(z) \subseteq z$ also contains unlifted states at mesh vertices, but only on the vertices that lie on the surface of B, and only contains pressure measurements. In both the free and forced oscillation cases, B is defined as the downstream cylinder. The values of S can then be physically interpreted as measurements from pressure sensors on the surface of the body, which would be feasible to obtain in practical applications, whereas the field values F are likely not possible to construct outside of a controlled laboratory setting. DMD is always performed on a subset of $g(z)$, and though S is a much narrower subset than F, modes calculated on either subset are valid DMD modes of the fluid system. This motivates the possibility that the more accessible surface modes may contain information about the broader fluid system that could be useful for flow reconstruction.

6.3.2 Numerical DMD

In this section, we describe our approach to approximate the eigenvalues and eigenvectors of K on S and F from specific windows of time within the simulations. These windows contain $m \in \mathbb{N}$ snapshots, starting from snapshot $b \in \mathbb{N}$, and the window is not allowed to continue past the available data ($m+b \leq M$). The windows of data are taken from a specific simulation, defined by both its type $w \in \{1, 2\}$ (1 denoting the free oscillations and 2 denoting the prescribed oscillations), and its specific simulation parameters d, with $d = U^* \in \{1, 2, \dots, 15\}$ for $w = 1$ or $d = A \in \{0.1, 0.2, \dots, 1.0\}$ for $w = 2$. Let the set C define the set of unique tuples (m, b, w, d) that it is possible to construct under the given constraints. Here we demonstrate the DMD approach for an arbitrary $c \in C$. Because all of the quantities defined beyond this point, such as the data windows, operators, and modes, depend on c, we drop this dependence from our notation for conciseness.

The observations from the simulation can be constructed into time-shifted surface data arrays $X \in \mathbb{R}^{k_s \times (m-1)}$ and $\bar{X} \in \mathbb{R}^{k_s \times (m-1)}$, as well as field data arrays and $Y \in \mathbb{R}^{k_f \times (m-1)}$ and $\bar{Y} \in \mathbb{R}^{k_s \times (m-1)}$ (where k denotes the number of meshpoints and the subscripts s and f generally refer to quantities calculated for the surface data and broader field data, respectively), as

$$X = \begin{bmatrix} S_b & S_{b+1} & \cdots & S_{b+m-1} \end{bmatrix} \tag{6.4}$$

$$\bar{X} = \begin{bmatrix} S_{b+1} & S_{b+2} & \cdots & S_{b+m} \end{bmatrix} \tag{6.5}$$

$$Y = \begin{bmatrix} F_b & F_{b+1} & \cdots & F_{b+m-1} \end{bmatrix} \tag{6.6}$$

$$\bar{Y} = \begin{bmatrix} F_{b+1} & F_{b+2} & \cdots & F_{b+m} \end{bmatrix}. \tag{6.7}$$

The operator on the surface measurements $K_s \in \mathbb{R}^{k_s \times k_s}$ and the operator of the field measurements $K_f \in \mathbb{R}^{k_f \times k_f}$ can then be found by recognizing that these matrices can be used to reconstruct the optimization problem in eq. 5.7 as

$$K_s = \underset{A}{\operatorname{argmin}} \|\bar{X} - AX\|_F \tag{6.8}$$

$$K_f = \underset{A}{\operatorname{argmin}} \|\bar{Y} - AY\|_F, \tag{6.9}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The operators resulting from this optimization map the data snapshots forward in time with minimal error in a least-squares sense, for instance: $S_{n+1} \approx K_s S_n$ and $F_{i+n} \approx K_f F_n$. If $m - 1 \leq k_f$, in other words, the data matrices have at least as many rows as they have columns (as is the case here), then the forward prediction has only roundoff error. Our objective is to calculate the eigenvalues and eigenvectors of the K matrices. However, K can be large, and calculating the eigendecomposition of a large matrix is computationally challenging. For instance, the free oscillation case has 32,434 mesh vertices, each with three states, so K_f has $k_f^2 = 9.47 \times 10^9$ parameters, which is a challenge to store in random-access memory on most modern computers, and even more challenging to perform computations on. On the fluid field data, this challenge is avoided by first calculating the singular value decomposition (SVD),

$$Y = U\Sigma V^T, \tag{6.10}$$

where T denotes the transpose, $U \in \mathbb{R}^{k_f \times k_f}$ and $V \in \mathbb{R}^{(m-1)\times(m-1)}$ are unitary matrices and $\Sigma \in \mathbb{R}^{k_f \times (m-1)}$ is a diagonal matrix containing singular values, by convention sorted in descending order from the top left. These matrices can be truncated to improve the computational efficiency of the following steps. Truncating the matrices such that only the r largest singular values are retained can dramatically reduce the scale of the eigendecomposition with minimal loss of accuracy. Assuming $r \leq max(k, m-1)$, the truncated matrices are denoted $\tilde{\Sigma} \in \mathbb{R}^{r \times r}$, $\tilde{U} \in \mathbb{R}^{k_f \times r}$ and $\tilde{V} \in \mathbb{R}^{(m-1) \times r}$. Using the truncated SVD, matrix K_f can then be approximated as $\tilde{K}_f \in \mathbb{R}^{r \times r}$ as

$$\tilde{K}_f = \tilde{U}^T \bar{Y} \tilde{V} \tilde{\Sigma}^{-1}, \tag{6.11}$$

and its eigenvalues and eigenvectors can be computed as

$$\tilde{K}_f \Psi_l = \Lambda_f \Psi_l, \tag{6.12}$$

where $\Lambda_f \in \mathbb{C}^{r \times r}$ is a diagonal matrix containing the complex eigenvalues

$$diag(\Lambda_f) = \begin{bmatrix} \lambda_{f,1} & \lambda_{f,2} & \cdots & \lambda_{f,r} \end{bmatrix} \in \mathbb{C}^r \tag{6.13}$$

and the columns of $\Psi_l \in \mathbb{C}^{r \times r}$ contain the eigenvectors in the reduced space, which have little physical meaning at this stage. These eigenvectors can be projected back to the full space using the left singular matrix,

$$\Psi = \tilde{U} \Psi_l, \tag{6.14}$$

where the columns of $\Psi \in \mathbb{C}^{k_f \times r}$ physically correspond to modes of the fluid field

$$\Psi = \begin{bmatrix} \psi_1 & \psi_2 & \cdots & \psi_p \end{bmatrix}. \tag{6.15}$$

Using these modes, it is possible to reconstruct and extrapolate the input data as

$$F_{b+q} = \Psi \Lambda_f^q \boldsymbol{\alpha}_f, \tag{6.16}$$

where $q \in \mathbb{Z}$ and $\boldsymbol{\alpha}_f = \Psi^{-1} F_b \in \mathbb{C}^r$ is a vector of complex numbers

$$\boldsymbol{\alpha}_f = \begin{bmatrix} \alpha_{f,1} & \alpha_{f,2} & \cdots & \alpha_{f,r} \end{bmatrix} \in \mathbb{C}^r, \tag{6.17}$$

where $\alpha_{f,j}$ contains the magnitude of mode ψ_j at the initial flow snapshot F_b, and its phase encodes the phase of the mode at the same snapshot. For this reason, we refer to $|\alpha_{f,j}|$ (where $|\cdot|$ denotes the complex magnitude) as the magnitude of mode j, and to $\angle\alpha_{f,j}$ (where $\angle\cdot$ denotes the complex phase) as the phase of mode j. We will later show that the steady-state flow is often dominated by a small number of high-magnitude modes, which we can exploit to simplify the flow field estimation.

The number of simulated pressure sensors on the surface is much less than the total number of simulation nodes in the fluid, and only one state (pressure) is measured at each of those surface points (as opposed to both pressure and velocity in the field), which results in $k_s << k_f$. As a result, there is no need to compute the eigendecomposition in a reduced space, and K_s can be directly calculated without truncation as

$$K_s = \bar{X}X^{+}, \tag{6.18}$$

where + represents the Moore-Penrose Pseudoinverse. This variation of DMD is known as 'Exact DMD' [211], and is equivalent to the SVD-based method without truncation. The eigendecomposition

$$K_s\Phi = \Lambda_s\Phi, \tag{6.19}$$

reveals the fluid eigenvectors $\Phi \in \mathbb{C}^{k_s \times k_s}$ and the eigenvector matrix $\Lambda_s \in \mathbb{C}^{k_s \times k_s}$, which are composed of individual modes and eigenvectors as

$$\Phi = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_{k_s} \end{bmatrix} \tag{6.20}$$

$$diag(\Lambda_s) = \begin{bmatrix} \lambda_{s,1} & \lambda_{s,2} & \cdots & \lambda_{s,k_s} \end{bmatrix}. \tag{6.21}$$

Much like the broader flow field, the surface pressure field can be reconstructed by a superposition of modes as

$$S_{b+q} = \Phi\Lambda_s^q\boldsymbol{\alpha}_s, \tag{6.22}$$

where $\boldsymbol{\alpha}_s = \Phi^{-1}S_b \in \mathbb{C}^{k_s}$ defines the magnitude and phase of the DMD modes of the pressure in the initial time snapshot on the body in a similar manner to how $\boldsymbol{\alpha_f}$ defines them for the general flow.

This work aims to construct the global flow field F given the local pressure field S. However, mapping directly between these matrices is challenging because of the number of parameters in F. We exploit the simplicity of the underlying dynamics of the flow to simplify the problem by instead mapping the most dominant modes in Φ, selected by the magnitude of the corresponding element of $\boldsymbol{\alpha}_s$, to the most dominant modes of Ψ, determined by the magnitude of the corresponding elements of $\boldsymbol{\alpha}_f$. This allows reconstruction of the flow to a reasonable degree of accuracy using only 3 to 4 modes.

6.4 Flow Reconstruction

6.4.1 Mode Selection

For both the free and forced oscillation cases, a strategy must be developed to determine how many modes are necessary to reconstruct the flow and which modes should be used. Let the dominant flow modes which are used in the reconstruction be labeled $[\tilde{\psi}_1 \ \tilde{\psi}_2 \ \cdots \ \tilde{\psi}_a]$, and the most dominant surface modes in the reconstruction be labeled $[\tilde{\phi}_1 \ \tilde{\phi}_2 \ \cdots \ \tilde{\phi}_a]$, where $a \in \mathbb{N}$ is the number of modes selected. Similarly, let the flow field eigenvalue $\tilde{\lambda}_{f,i}$ and magnitude $\tilde{\alpha}_{f,i}$ correspond to the mode $\tilde{\psi}_i$, and the surface eigenvalue $\tilde{\lambda}_{s,i}$ and magnitude $\tilde{\alpha}_{s,i}$ correspond to $\tilde{\phi}_i$. One key challenge is to make the mapping from surface modes to flow modes consistent for different simulations and different time snapshots within the same simulation. More specifically, for two different windows c_1 and c_2 in C for the same case w (either free oscillation or forced oscillation), the inner product $< \tilde{\phi}_i(c_1), \tilde{\phi}_i(c_2) >$ and $< \tilde{\psi}_i(c_1), \tilde{\psi}_i(c_2) >$ should have magnitude near one and $\tilde{\lambda}_i(c_1) \approx \tilde{\lambda}_i(c_2)$, implying that the modes are physically similar and correspond to the same physical phenomenon. This is necessary for the neural networks to work well, as the mapping is much simpler when the changes in the modes are small and consistent. By simply ordering the modes based on their relative dominance, this criterion may not necessarily be met: as one mode overtakes another with a change in simulation parameters U^* or A, the order of the two modes would flip. To counteract this, a strategy to enforce consistency is necessary, which we tailor for each of the two cases to capture the most important modes throughout the considered ranges of U^* and A.

6.4.1.1 Free oscillations in the wake of a stationary obstacle

The relative magnitudes $\boldsymbol{\alpha}$ and eigenvalues Λ of the free oscillation simulations with $U^* = 8$ after the transient period ($b = 300$) are shown in Figure 6.6. Three dominant modes can be clearly

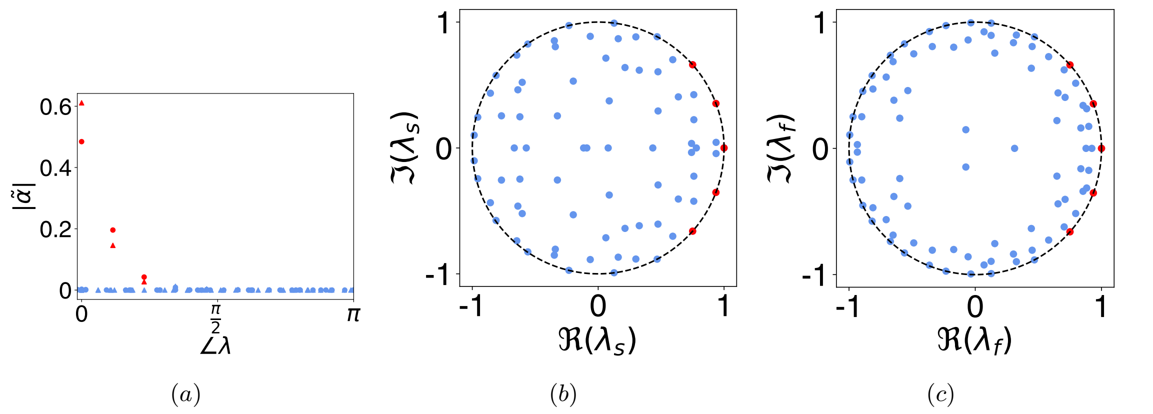


Figure 6.6: (a) The normalized magnitude ($\sum_{c \in C} |\tilde{\alpha}(c)| = 1$) of the DMD modes for the surface pressure measurements (circles) and the flow field state measurements (triangles) for $U^* = 8$ and $b = 300$. In both cases, three dominant modes (red) can be clearly seen, with the dominant modes of the field and surface having the same frequency. The eigenvalues for (b) the surface data and (c) the flow field data show that the dominant modes have an eigenvalue magnitude of roughly 1 (meaning that they do not grow or decay with time), and multiple further harmonics can be seen on the unit circle. Decaying modes within the circle are also present but have a low magnitude. The dominant eigenvalues of the flow and surface are clearly similar.

seen in Figure 6.6(a) for both the surface and flow field and are marked in red and appear to contain roughly 3/4 of the total magnitude of all of the modes. The eigenvalues corresponding to the dominant modes of the surface data and those corresponding to the dominant modes of the flow field data have very similar complex phases, indicating that they correspond to phenomena with the same frequency and likely represent the same physical phenomena. These eigenvalues are shown on the unit circle in the complex plane in Figure 6.6(b,c) with the dominant modes shown in red. Many eigenvalues are in the unit circle, corresponding to modes that decay rapidly after the initial transient period. The dominant modes all have complex magnitudes close to 1 and are on the unit circle. The phases of the eigenvalues of the two dominant modes with non-zero imaginary components can be seen to differ by almost exactly a factor of two, indicating that they are harmonics, with the higher-frequency harmonic having lower magnitude. This pattern of three dominant harmonic modes, one with zero phase and two that are harmonics, is consistent across values of U^* and across both the surface data and the flow field data. As a result, $a = 3$ modes are used for the flow reconstruction for this case. However, certain edge cases can make the mode labeling more challenging for certain windows of data.

The first labeled mode $\tilde{\psi}_1$ is always the mode corresponding to the eigenvalue with zero phase, which always has the highest magnitude of α_f. Its index can be found as

$$i_1 = \underset{i}{\operatorname{argmax}}\,(|\alpha_{f,i}|). \tag{6.23}$$

The second mode, corresponding to the first harmonic, is more challenging to identify because it is not always the mode with the second-highest magnitude. Rarely, a mode with very low frequency ($-0.001 < \angle\lambda_{f,i} < 0.001$) can be found, which can have high magnitude because of overlap with the zero-frequency mode. This low-frequency behavior is not observed in most time windows nor in simulation, so it is considered a numerical artifact and ignored. Accounting for this, the second mode can be identified as

$$i_2 = \underset{i \neq i_1}{\operatorname{argmax}}(|\alpha_{f,i}|)\ s.t.\ 0.001 < \angle\lambda_{f,i}, \tag{6.24}$$

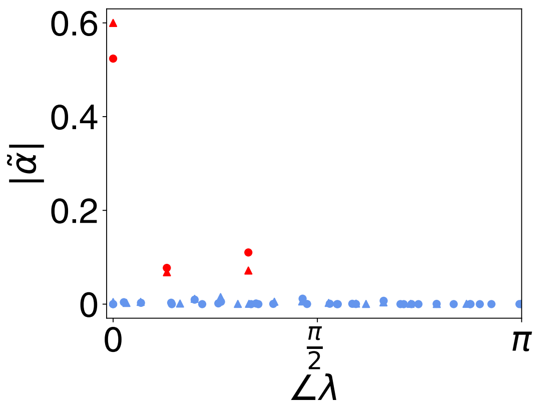
which also consistently only finds the positive complex conjugate. To consistently identify the third mode (second harmonic), an additional edge case must be considered: an additional high magnitude mode rarely appears with an eigenvalue phase very near to the phase of the eigenvalue corresponding to $\tilde{\psi}_2$, which is also considered a numerical error and neglected. The third mode index is then identified as

$$i_3 = \underset{i \notin \{i_1, i_2\}}{\operatorname{argmax}}(|\alpha_{f,i}|)\ s.t.\ 1.1\angle\lambda_{f,i_2} < \angle\lambda_{f,i}, \tag{6.25}$$

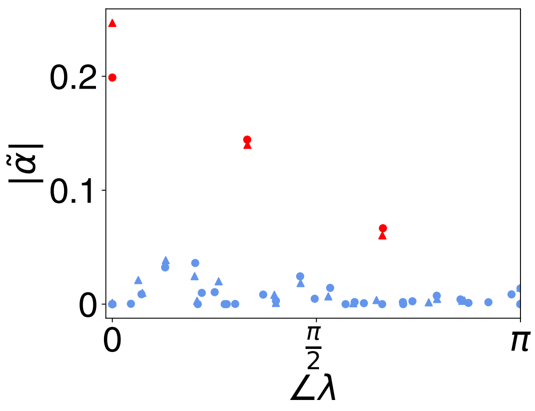
which eliminates the risk of identifying lower frequency mode as the second harmonic by considering only modes with corresponding frequency at least 1.1 times greater than the first harmonic. Knowing the indices of the modes, the modes themselves, as well as their eigenvalues and magnitudes, can be defined as $\tilde{\psi}_j = \psi_{i_j}$, $\tilde{\lambda}_{f,j} = \lambda_{f,i_j}$, and $\tilde{\alpha}_{f,j} = \alpha_{f,i_j}$ for all $j \in \{1, 2, 3\}$. The exact same procedure is applied to the surface modes to identify $\tilde{\phi}_j$, $\tilde{\lambda}_{s,j}$, and $\tilde{\alpha}_{s,j}$.

6.4.1.2 Forced oscillations of tandem cylinders

Identifying the modes in the forced oscillation case is more challenging because the dominant modes change as A varies. For instance, the mode magnitude for $A = 0.1$ and $A = 0.8$ is shown in Figure 6.7(a) and (b), respectively. At $A = 0.1$, three conjugate pairs of modes have high magnitudes in both the field and on the surface. Similar to the previous case, one has an associated phase of 0



(a)



(b)

Figure 6.7: The relative magnitude of the modes for (a) $A = 0.1$ and (b) $A = 0.8$ for the surface data (circles) and for the flow field data (triangles). The magnitude is concentrated in fewer modes for the $A = 0.1$ case, where three modes can reconstruct the flow with a low degree of lost information. The position of the three most dominant modes (red) changes as a function of A, and the magnitude is generally distributed through more modes for higher values of A.

and is the 'mean mode' roughly corresponding to the average value of the measurements. The other two modes have $\angle\lambda_{f,i}$ of 0.41 and 1.04. The phase of an eigenvalue can be converted to an angular frequency ω using the formula

$$\omega_i = \angle\lambda_{f,i}\Delta t, \tag{6.26}$$

where the time between measurements Δt is 1 for this case. The mode with phase 1.04 rad/s must correspond to a physical phenomenon with frequency $\omega = 1.04$ rad/s, the prescribed forcing frequency. This mode, labeled mode 2, corresponds with the periodic fluid behavior driven by these oscillations. The other dominant mode, which we label mode 3, has a frequency $\omega = 0.41$, which does not correspond to an integer multiple of the forcing and likely corresponds to the frequency of vortex shedding for the unforced system.

A larger number of modes with significant amplitude can be seen at the higher forcing amplitude of $A = 0.8$, shown in 6.7(b). The three highest magnitude modes exist in descending order of magnitude at $\angle\lambda_f$ of 0, 1.04, and 2.08. The former two modes can be identified by their frequencies as modes 1 and 2, respectively, which were identified for the lower forcing amplitude. The mode with $\angle\lambda_f = 2.08$ had negligible amplitude at the lower forcing amplitude, and based on its frequency, it is the second harmonic of the prescribed forcing. We label this mode as mode 4. Though its magnitude relative to the forcing modes has decreased, mode 3 can also be observed here with a phase of 0.41.

To estimate these modes, a strategy must again be developed that can identify physically consistent modes of the operators, irrespective of the parameter A. A reasonable strategy is to identify them by the phases of their associated eigenvalues as

$$i_1 = \underset{i}{\text{argmax}}(|\alpha_{f,i}|) \; s.t. \; \angle\lambda_{f,i} = 0 \tag{6.27}$$

$$i_2 = \underset{i}{\text{argmax}}(|\alpha_{f,i}|) \; s.t. \; 1.00 < \angle\lambda_{f,i} < 1.08 \tag{6.28}$$

$$i_3 = \underset{i}{\text{argmax}}(|\alpha_{f,i}|) \; s.t. \; 0.33 < \angle\lambda_{f,i} < 0.43 \tag{6.29}$$

$$i_4 = \underset{i}{\text{argmax}}(|\alpha_{f,i}|) \; s.t. \; 2.00 < \angle\lambda_{f,i} < 2.16. \tag{6.30}$$

Because of the possibility of numerical error causing the frequencies of modes 2 and 4 to vary from the known forcing frequency, those modes are identified from a small range centered on the known forcing frequency. The frequency of the vortex shedding that mode 3 corresponds to can vary as a function of A, so the range in which to search for mode 3 is determined by identifying the phase of mode 3 for a range of A values. The range of identified values is $0.35 \leq \lambda_{f,i_3} \leq 0.41$, so the range where mode 3 is searched for is the same plus a margin of 0.02.

Based on these indices, the modes themselves, as well as their eigenvalues and magnitudes, can be defined in the same manner as for the free oscillation case: $\tilde{\psi}_j = \psi_{i_j}$, $\tilde{\lambda}_{f,j} = \lambda_{f,i_j}$, and $\tilde{\alpha}_{f,j} = \alpha_{f,i_j}$. The exact same procedure is again applied to the surface modes to identify $\tilde{\phi}_j$, $\tilde{\lambda}_{s,j}$, and $\tilde{\alpha}_{s,j}$. The key difference is that here, $j \in \{1, 2, 3, 4\}$.

6.4.2 Data Normalization

To reconstruct the full flow field based on surface measurements, we need to estimate the values of $\tilde{\psi}_i$, $\tilde{\lambda}_{f,i}$, and $\tilde{\alpha}_{f,i}$ given $\tilde{\phi}_i$, $\tilde{\lambda}_{s,i}$, and $\tilde{\alpha}_{s,i}$. However, performing this mapping directly using a neural network is challenging, as the input to a traditional neural network must be a vector of real-valued features, and the values to be mapped are complex-valued. Additionally, because modes are eigenvectors, they can be multiplied by any non-zero complex number and remain a valid mode: the result of $v\tilde{\psi}_i$ for $v \in \mathbb{C}$ where $v \neq 0$ is another valid representation of mode i. During the eigendecomposition, one of these valid representations is calculated for $\lambda_{f,i}$, but it is not necessarily done in a consistent manner, so two similar modes $\psi_i(c_1)$ and $\psi_i(c_2)$ for similar data windows

$c_1, c_2 \in C$ could appear quite different only due to having different arbitrary constants. The effect of the complex constant can be decomposed into two parts: its magnitude scales the magnitude of the mode vector ψ_i, and its phase rotates the phase of every element of ψ_i in the complex plane. As a result of this rotation, the absolute phase of the elements of ψ_i holds no meaning; however, the relative phases of its elements do hold useful information about the relative timing of state oscillations at different points in the flow field.

The process of normalizing the phase of a complex-valued vector is less standardized than other forms of normalization. However, the prevalent approach involves phase rotation such that a designated reference element, often the first element in the vector, aligns with the real axis. This rotation preserves the relative phases between the points, thereby facilitating any subsequent analyses that may use that information. Letting $\tilde{\psi}_{i,1}$ denote the first element of $\tilde{\psi}_i$, the phase normalized vector $\hat{\psi}_i$ can be defined as

$$\hat{\psi}_i = \frac{\tilde{\psi}'_{i,1}}{|\tilde{\psi}_{i,1}|}\tilde{\psi}_i, \tag{6.31}$$

where $'$ denotes the complex conjugate. Because the values of $\tilde{\alpha}_{f,i}$ were constructed using the pre-normalization modes, their complex phases must be updated (in the opposite direction) as

$$\hat{\alpha}_{f,i} = \frac{\tilde{\psi}_{i,1}}{|\tilde{\psi}_{i,1}|}\tilde{\alpha}_{f,i}. \tag{6.32}$$

This normalization procedure is also applied to the surface measurements to calculate $\hat{\phi}_i$ and $\hat{\alpha}_{f,i}$.

6.4.3 Map Definition

In order to reconstruct the flow field, we require a map

$$f_c : (\hat{\phi}_i, \tilde{\lambda}_{s,i}, \hat{\alpha}_{s,i}) \rightarrow (\hat{\psi}_i, \tilde{\lambda}_{f,i}, \hat{\alpha}_{f,i}). \tag{6.33}$$

However, the mapping is performed by a dense neural network, which maps a real-valued vector to another real-valued vector. These modes must then be concatenated into a real-valued vector in order to be mapped by the neural network, which is achieved by concatenating its real and imaginary components, as well as the real and imaginary components of its corresponding magnitude,

$$x_i = \begin{cases} \begin{bmatrix} \Re(\hat{\phi}_i) & \Re(\hat{\alpha}_{s,i}) \end{bmatrix} & i = 1 \\ \begin{bmatrix} \Re(\hat{\phi}_i) & \Im(\hat{\phi}_i) & \Re(\hat{\alpha}_{s,i}) & \Im(\hat{\alpha}_{s,i}) \end{bmatrix} & i \neq 1 \end{cases} \tag{6.34}$$

$$y_i = \begin{cases} \begin{bmatrix} \Re(\hat{\psi}_i) & \Re(\hat{\alpha}_{f,i}) \end{bmatrix} & i = 1 \\ \begin{bmatrix} \Re(\hat{\psi}_i) & \Im(\hat{\psi}_i) & \Re(\hat{\alpha}_{f,i}) & \Im(\hat{\alpha}_{f,i}) \end{bmatrix} & i \neq 1. \end{cases} \tag{6.35}$$

When $i = 1$, both the mode and magnitude are real, so there is no need to store the imaginary components. The eigenvalues are conspicuously missing from these vectors; that is because their magnitudes are roughly equal because they must be near 1 in the steady state ($|\tilde{\lambda}_{f,i}| \approx |\tilde{\lambda}_{s,i}| \approx 1$), and their phases are always very similar because they describe the same physical phenomena ($\angle\tilde{\lambda}_{f,i} \approx \angle\tilde{\lambda}_{s,i}$), so the eigenvalues are mapped separately by

$$f_I : \tilde{\lambda}_{s,i} \to \tilde{\lambda}_{f,i}, \tag{6.36}$$

where f_I is the identity map. By mapping the eigenvalues separately from the other flow information in this way, the number of parameters in the neural network can be slightly reduced. The vectors are further concatenated into a longer vectors $\mathbb{X} \in \mathbb{R}^{(2a-1)(k_s+1)}$ and $\mathbb{Y} \in \mathbb{R}^{(2a-1)(k_f+1)}$ that contain information about all of the modes as

$$\mathbb{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_a \end{bmatrix} \tag{6.37}$$

$$\mathbb{Y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_a \end{bmatrix}. \tag{6.38}$$

The number of parameters in $\mathbb{X}$ and $\mathbb{Y}$ is much less than that of X and Y for large m, which makes constructing a mapping between them easier. The mapping is performed by a fully-connected dense neural network $\mathcal{N}$ as $\tilde{\mathbb{Y}} = \mathcal{N}(\mathbb{X})$, where $\tilde{\mathbb{Y}}$ is the predicted value of $\mathbb{Y}$.

The network architecture contains three hidden layers of 500, 1000, and 1500 nodes (in order from input to output), with hyperbolic tangent rectification on the hidden layers and a linear activation function on the output. More specifically, the mapping takes the form

$$\mathcal{N}(x) = W_4 \tanh(W_3 \tanh(W_2 \tanh(W_1 x + b_1) + b_2) + b_3) + b_4, \tag{6.39}$$

where the weights $W_1 \in \mathbb{R}^{500\times(2a-1)(k_s+1)}$, $W_2 \in \mathbb{R}^{1000\times 500}$, $W_3 \in \mathbb{R}^{1500\times 1000}$, $W_4 \in \mathbb{R}^{(2a-1)(k_f+1)\times 1500}$, $b_1 \in \mathbb{R}^{500}$, $b_2 \in \mathbb{R}^{1000}$, $b_3 \in \mathbb{R}^{1500}$, and $b_4 \in \mathbb{R}^{(2a-1)(k_f+1)}$. Collectively, we refer to the list containing all of these weights as σ. Each of the two simulation cases requires its own weights, so the weights corresponding to the free oscillations are labeled σ_1, and the weights corresponding to the prescribed oscillations are labeled σ_2. The weights σ_j are trained to minimize the loss

$$\sigma_j = \underset{\sigma}{\operatorname{argmin}} \sum_{c \in C_j} \|\mathcal{N}_\sigma(\mathbb{X}(c)) - \mathbb{Y}(c)\|_2, \tag{6.40}$$

where C_j denotes the set of possible time windows over all simulations for case j. This minimizes the error between the expected field mode vector and the true one in a least-squares sense.

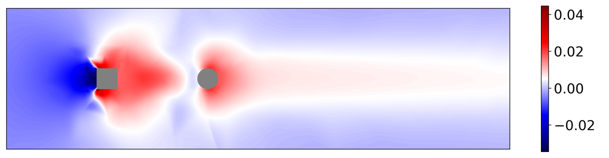
6.4.4 Map Implementation

Computational limitations make it challenging to perform this optimization over the entire set C_j, and doing so would leave no new data on which to test the effectiveness of the network. To solve both of these problems, we split C_j into three different sets: a training set $C_{j,t}$, a validation set $C_{j,v}$, and a testing set $C_{j,e}$. The training set contains either points from the free oscillation case where $U^* \in \{1, 2, \dots, 14\}$ and $b = \{210, 211, \dots, 310\}$, or points from the forced oscillation case where $A \in \{0.1, 0.2, \dots, 1.0\}$ where $A \neq 5$, with $b = \{50, 51, \dots, 90\}$. The validation set includes the same ranges of U^* or A, but has $b = \{330, 331, \dots, 340\}$ for the free oscillation case or $b = \{140, 141, \dots, 149\}$ for the forced oscillation case. This choice of initial times allows the network to be validated with little data overlap. The test data is from simulations not seen in the other datasets: in the free oscillation case, $U^* = 15$ is used to demonstrate that this procedure can extrapolate beyond the training parameter range, and in the forced oscillation case, $A = 0.5$ to demonstrate interpolation within the training parameter range. The entire steady state time window is used, $b = \{210, 211, \dots, 340\}$ for the free oscillation case and $t_0 = \{50, 51, \dots, 149\}$ for the forced oscillation case. In the free oscillation case $m = 100$, and in the forced oscillation case $m = 50$.

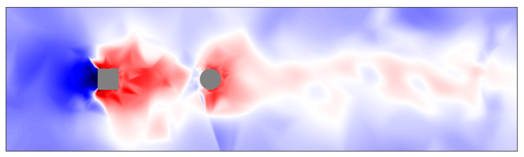
At the beginning of training, 20 snapshots from each simulation in $C_{j,t}$ are randomly selected to form a list $\boldsymbol{c}_{j,t} \subseteq C_{j,t}$, and a list of training matrices $\bar{\mathbb{X}}_t = [\mathbb{X}(\boldsymbol{c}_{j,t,1}), \mathbb{X}(\boldsymbol{c}_{j,t,2}), \ldots, \mathbb{X}(\boldsymbol{c}_{j,t,20})]$ and $\bar{\mathbb{Y}}_t = [\mathbb{Y}(\boldsymbol{c}_{j,t,1}), \mathbb{Y}(\boldsymbol{c}_{j,t,2}), \ldots, \mathbb{Y}(\boldsymbol{c}_{j,t,20})]$. Validation data $\bar{\mathbb{X}}_v$ and $\bar{\mathbb{Y}}_v$ are also constructed by the same procedure, however, using only 5 snapshots per simulation. This data is iteratively used to improve $\mathcal{N}_\sigma$ by minimizing its mean-squared training loss, given by a slightly modified version of eq. 6.40:

$$L_{t,j} = \sum_{c \in \boldsymbol{c}_{j,t}} \|\mathcal{N}_{\sigma_|}(\mathbb{X}(c)) - \mathbb{Y}(c)\|_2, \tag{6.41}$$

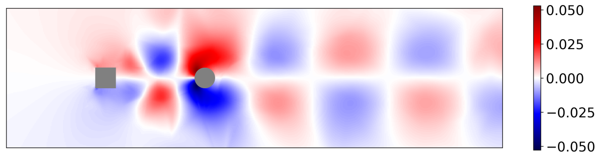
where validation loss $L_{v,j}$ is calculated by the same procedure over the validation data. Because all of the losses and weights depend on the case j, we drop it from the subscripts beyond this point for clarity. This loss is then iteratively minimized using the *adam* algorithm, which is a stochastic gradient descent algorithm with momentum. The gradient of L_t with respect to σ is calculated on a subset (batch) of the training data $\boldsymbol{C}_t$, and the weights are updated in the direction of decreasing loss with an additional 'momentum' term based on the gradients of previous batches. The batches are iterated until all of the data has been used, known as an *epoch*.



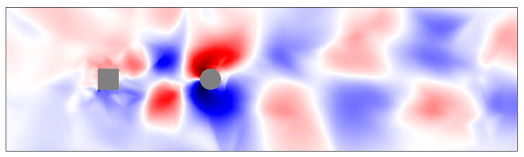
(*a*) Actual $\Re(\tilde{\psi}_1)$



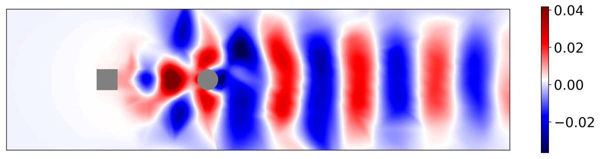
(*b*) Estimated $\Re(\tilde{\psi}_1)$



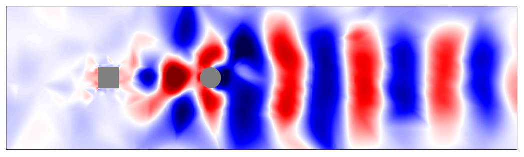
(*c*) Actual $\Re(\tilde{\psi}_2)$



(*d*) Estimated $\Re(\tilde{\psi}_2)$



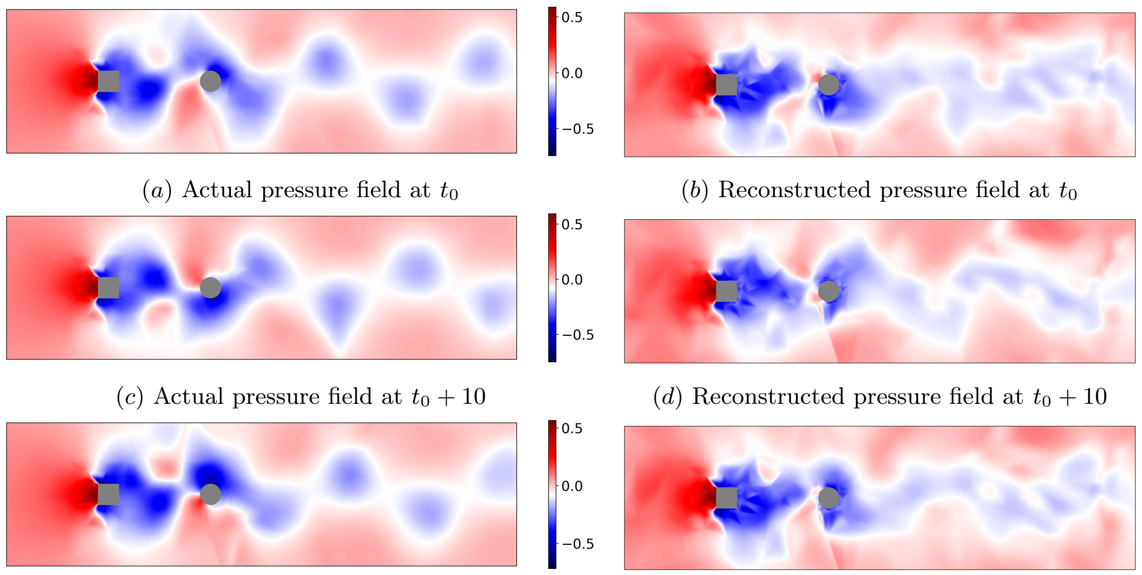
(*e*) Actual $\Re(\tilde{\psi}_3)$



(*f*) Estimated $\Re(\tilde{\psi}_3)$

Figure 6.8: The real component of the actual vs. estimated modes of the pressure field for free oscillations at $U^* = 15$. The estimated modes are qualitatively similar to the actual modes, though the estimation mapping does introduce noise especially close to the leading body.

We use an early stopping algorithm from [149] to know when to terminate the training before overtraining can occur. After every 10th epoch, the loss L_v on the validation data is calculated. If



(a) Actual pressure field at t_0

(b) Reconstructed pressure field at t_0

(c) Actual pressure field at $t_0 + 10$

(d) Reconstructed pressure field at $t_0 + 10$

(e) Actual pressure field at $t_0 + 20$

(f) Reconstructed pressure field at $t_0 + 20$

Figure 6.9: The actual vs. reconstructed pressure field for the case of the wake induced free oscillation of the cylinder at $U^* = 15$ at times t_0 (a,b) , $t_0 + 10$ (c,d), and $t_0 + 20$ (e,f).

$L_v < L_{min}$, where L_{min} is the lowest validation error yet recorded, then $L_{min} := L_v$ and $\sigma_{opt} := \sigma$, where σ_{opt} are the weighs corresponding to L_{min}. However, if $L_v > 1.2L_{min}$, and at least 100 epochs have passed, then it is assessed that the network is overtrained and the training is terminated. The weights σ_{opt} are then used to reconstruct the flow for case j.

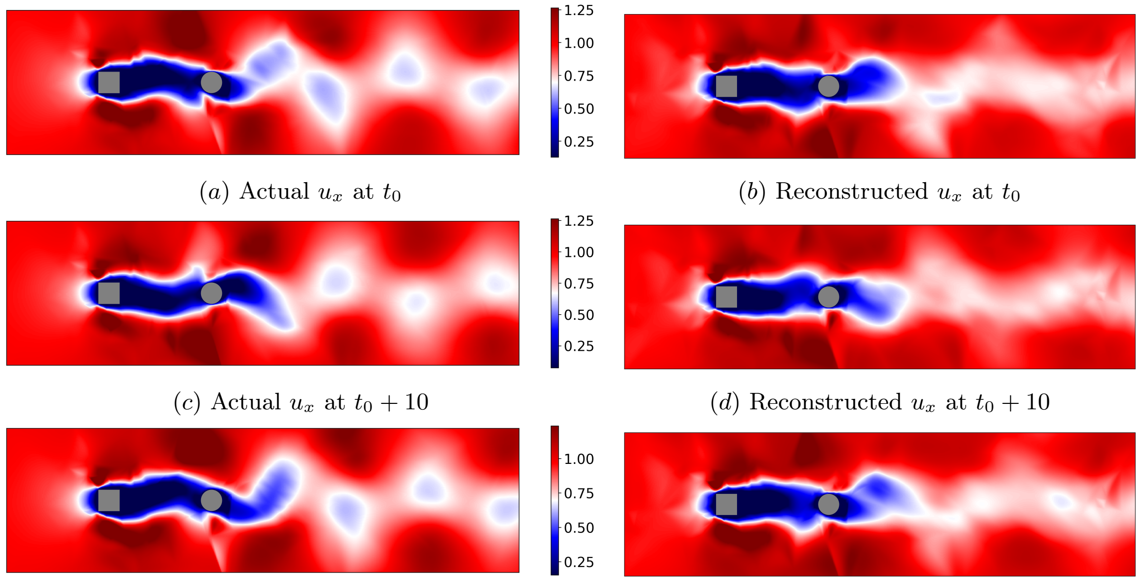
From the mapping, the value of $\tilde{\mathbb{Y}}_c$ is known, which allows extracting the estimated values of $\hat{\psi}_{i,e}$ and $\hat{\alpha}_{f,i,e}$ (combining the estimates of the real and imaginary components), and the estimated flow field eigenvalues $\hat{\lambda}_{f,i,e}$ are known because it is assumed that $\tilde{\lambda}_{f,i,e} = \tilde{\lambda}_{s,i}$. Using these values, it is possible to reconstruct an approximation of Y using a similar reconstruction to that in 6.16:

$$\tilde{F}_{b+q} = \sum_{i=1}^{a} \hat{\psi}_{i,e} \tilde{\lambda}_{f,i,e}^{q} \hat{\alpha}_{f,i,e}, \tag{6.42}$$

where $\tilde{F}$ is the predicted flow field.

6.5 Results

The modal characteristics of pressure fields obtained from the surface modes for the free oscillation problem are illustrated in Figure 6.8. Overall, the reconstructed pressure field provides



(a) Actual u_x at t_0 (b) Reconstructed u_x at t_0

(c) Actual u_x at $t_0 + 10$ (d) Reconstructed u_x at $t_0 + 10$

(e) Actual u_x at $t_0 + 20$ (f) Reconstructed u_x at $t_0 + 20$

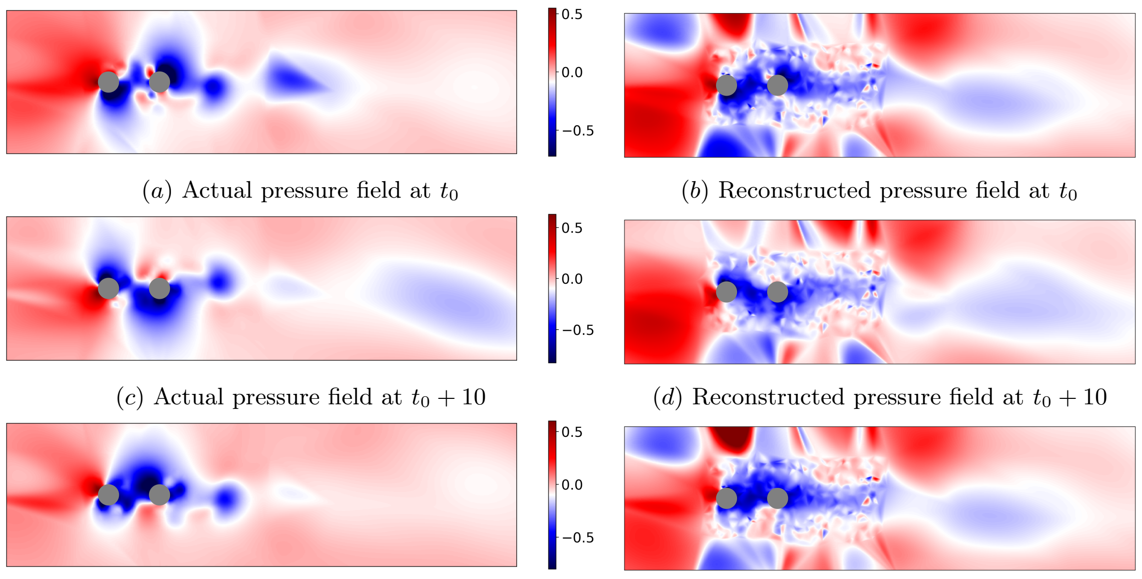
Figure 6.10: The actual vs. reconstructed horizontal velocity (u_x), field for the case of the wake induced free oscillation of the cylinder at $U^* = 15$ at times t_0 (a,b) , $t_0 + 10$ (c,d), and $t_0 + 20$ (e,f).

a high-fidelity representation of the essential characteristics of each mode, with only a moderate degree of error. Notably, the pressure oscillations observed downstream of the cylinder, indicative of vortex wake advection, are accurately captured in the reconstruction. This suggests that these modes may possess sufficient accuracy to properly reconstruct the entire flow field.

The reconstructed pressure and velocity fields are delineated in Figures 6.9 and 6.10, respectively. The pressure field reconstruction generally exhibits a high level of accuracy, although the mapping fuses the discrete low-pressure zones induced by individual vortices into a single, heterogeneous low-pressure region. Importantly, the fidelity of the reconstruction appears to be temporally invariant within the evaluated 20-second time frame. This is despite this duration being ample for the advection of approximately three vortex pairs downstream. Notably, the predominant source of error emanates from the initial modal estimates rather than the time-projection of these modes. A consistent vortex wake smoothing effect is observed in the velocity field. The reconstruction demonstrates heightened accuracy in the regions proximate to the cylinder, which is congruent with the methodology that employs surface data from the cylinder for the reconstruction process.

Reconstructing the forced oscillation case, where information about the flow is distributed across multiple modes, presents increased computational challenges. As depicted in Figures 6.11

and 6.12, the reconstructed pressure and velocity fields exhibit certain idiosyncrasies. Notably, numerical noise is discernible in the rectangular region characterized by high mesh density adjacent to the bodies. The low-pressure oscillations localized on and between the cylinders are reconstructed with the highest fidelity, which is consistent with the data collection region serving as the basis for the flow field reconstruction. Comparative analysis suggests that the velocity field is reconstructed with greater accuracy than the pressure field. However, it should be highlighted that features located at a greater distance from the cylinders are generally subject to lower reconstruction accuracy than those in closer proximity.



(a) Actual pressure field at t_0 (b) Reconstructed pressure field at t_0

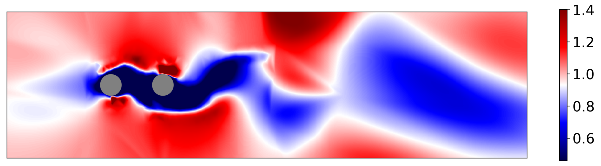
(c) Actual pressure field at $t_0 + 10$ (d) Reconstructed pressure field at $t_0 + 10$

(e) Actual pressure field at $t_0 + 10$ (f) Reconstructed pressure field at $t_0 + 20$

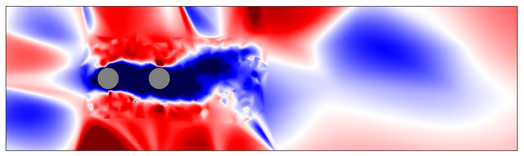
Figure 6.11: The actual vs. reconstructed pressure field is depicted for the case of forced oscillations of two tandem cylinders, with an oscillation amplitude of $A = 0.5$ and a frequency of $\omega = 1.04\,\mathrm{rad/s}$, at times t_0 (a,b), $t_0 + 10$ (c,d), and $t_0 + 20$ (e,f).

6.6 Conclusion

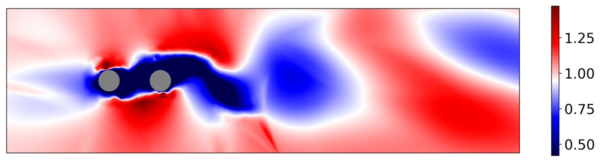
In this study, we have established that Dynamic Mode Decomposition (DMD) can be effectively utilized to estimate the modes of a flow field using data gathered exclusively from the surface of an immersed body. Moreover, by mapping both the magnitude and phase of these modes, we have successfully reconstructed the corresponding velocity and pressure fields. Our methodology has been demonstrated on two distinct fluid-body interaction scenarios: one involving free oscillations



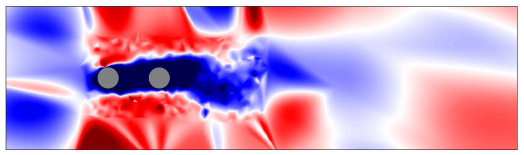
(a) Actual u_x at t_0



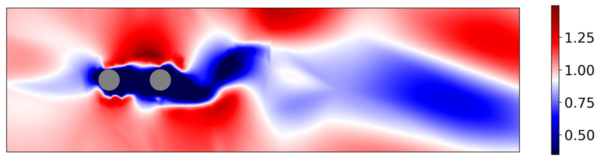
(b) Reconstructed u_x at t_0



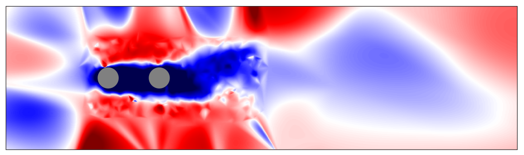
(c) Actual u_x at $t_0 + 10$



(d) Reconstructed u_x at $t_0 + 10$



(e) Actual u_x at $t_0 + 20$



(f) Reconstructed u_x at $t_0 + 20$

Figure 6.12: The actual vs. reconstructed horizontal velocity (u_x) field is depicted for the case of forced oscillations of two tandem cylinders, with an oscillation amplitude of $A = 0.5$ and a frequency of $\omega = 1.04\,\mathrm{rad/s}$, at times t_0 (a,b), $t_0 + 10$ (c,d), and $t_0 + 20$ (e,f).

in the wake of a cylinder and the other encompassing forced oscillations. The approach has been demonstrated to be versatile, exhibiting applicability across both categories of fluid dynamics problems. These findings carry significant implications for underwater robotics, offering the potential to facilitate advanced features such as obstacle avoidance and optimal motion planning through an enhanced understanding of the surrounding fluid environment.

Chapter 7

Conclusions

In this research, we have developed a range of techniques to utilize the dynamics of underwater robots to improve their sensing and control capability, with the goal of improving their autonomy to better utilize the aquatic environments that cover 90% of the planet. While this work has already demonstrated that our proposed frameworks can solve new problems and meet or exceed the performance state-of-the-art approaches, there are opportunities to extend this work further.

Our work on controls for underwater robots has shown that extremely simplified models that capture the key physics can be a useful step in a curriculum to train an RL agent on a complex system. Though we only show this with the Chaplygin sleigh as a simplified model of a swimmer, many fields have comparable simplified models that could extend this framework to many different problems. Further work on this framework may consider the case where the number of inputs to the RL must increase throughout the curriculum. It is also worth proving that this curriculum can be applied in experiment, perhaps changing the common sim-to-real strategy to a reduced model-to-sim-to-real framework.

Our work on sensing using body kinematics has shown that often-discarded body kinematic data can be a useful part of a robot's sensory suite, and further that increasing the complexity of the robot's internal dynamics can result in better sensing. This was tested in a body of two rigid links, but it is unclear what the diminishing returns of increased complexity are. Testing this concept in a flexible body (similar to a real fish) and perhaps with flexible fins, may reveal that proprioceptive sensing can continue to scale well with complexity. Also of interest is the potential interference between actuation and sensing: If the body were actuated, would it still be able to capture the motion induced by the fluid flow over the motion resulting from its actuation, and how could the forcing be designed to minimize interference, or perhaps even improve the sensing capabilities?

Using the Koopman operator for sensing, both for localization of an obstacle and sensing the general fluid flow, using dynamic pressure sensor measurements has been shown to be an effective strategy in simulation. Further, it was found that the Koopman modes can achieve parity with a state-of-the-art CNN for the purpose of feature extraction. Of course, much more research has gone into parametric approaches to time series classification compared to the Koopman operator for time series classification, should more research effort be directed to the latter it entirely possible that it could overtake purely parametric approaches for high-dimensional time series with underlying dynamics. It is important to note that, while this framework was developed and tested on an underwater robot, the framework can in principle be generalized to any high-dimensional time series which is generated by a dynamic system, which includes many problems of broad interest such as fault detection or prediction of financial markets. Further work may also investigate generating the Koopman operator from multiple data sources, for instance simultaneously using the lateral line data and kinematic data.

Appendices

Appendix A Two-link Sleigh Equations

This section provides further details for the derivation in chapter 3. The mass matrix used in the Lagrangian and the equation of motion (3.3) is

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} m & 0 & -m_2\,\epsilon\,l\,\sin\theta_1 & -m_2\,(1-\epsilon)\,l\,\sin\theta_2 \\ 0 & m & m_2\,\epsilon\,l\,\cos\theta_1 & m_2\,(1-\epsilon)\,l\,\cos\theta_2 \\ -m_2\,\epsilon\,l\,\sin\theta_1 & m_2\,\epsilon\,l\,\cos\theta_1 & m_2\,\epsilon^2\,l^2 + I_1 & m_2\,\epsilon\,(1-\epsilon)\,l\,\cos(\theta_2-\theta_1) \\ -m_2\,(1-\epsilon)\,l\,\sin\theta_2 & m_2\,(1-\epsilon)\,l\,\cos\theta_2 & m_2\,\epsilon\,(1-\epsilon)\,l^2\,\cos(\theta_2-\theta_1) & m_2\,(1-\epsilon)^2\,l + I_2 \end{bmatrix}. \tag{1}$$

Note that the mass matrix is used in the derivation before the rescaling, so the parameters here have not been rescaled. In the dynamical system (3.5) $\boldsymbol{\mathcal{N}}\dot{\boldsymbol{\xi}} = \boldsymbol{g}(\boldsymbol{\xi}) + \boldsymbol{f}(t)$, the inertia-like tensor

$$\boldsymbol{\mathcal{N}} = \begin{bmatrix} 1 & -\sin\delta\,(\epsilon-1)^2 & -\sin\delta\,(\epsilon-1)^2 & 0 \\ -\sin\delta\,(\epsilon-1)^2 & \boldsymbol{\mathcal{N}}_{2,2} & \boldsymbol{\mathcal{N}}_{2,3} & 0 \\ -\sin\delta\,(\epsilon-1)^2 & \boldsymbol{\mathcal{N}}_{2,3} & \boldsymbol{\mathcal{N}}_{3,3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where

$$\boldsymbol{\mathcal{N}}_{2,2} = 4\,\epsilon\,(\epsilon-1)^2\cos(\delta) - 4\,\epsilon^3 + (48\,\gamma+7)\epsilon^2 + (-48\,\gamma-3)\epsilon + 16\,\gamma + 1 \tag{3}$$

$$\boldsymbol{\mathcal{N}}_{2,3} = -(\epsilon-1)(16\,\epsilon^2\gamma - 2\,\epsilon^2\cos(\delta) - 32\,\epsilon\,\gamma + 2\,\epsilon\,\cos(\delta) + \epsilon^2 + \gamma\,r^2 + 16\,\gamma - 2\,\epsilon + 1 \tag{4}$$

$$\boldsymbol{\mathcal{N}}_{3,3} = -\,(\epsilon-1)\left(16\,\epsilon^2\gamma + \epsilon^2 - 32\,\epsilon\,\gamma - 2\,\epsilon + \gamma\,r^2 + 16\,\gamma + 1\right). \tag{5}$$

For the special case where $c = 0$, $A = 0$, fixed points can be defined for this system by solving $\boldsymbol{\mathcal{N}}^{-1}\,\boldsymbol{g} = \boldsymbol{0}$. For nonsingular $\boldsymbol{\mathcal{F}}$, this simplifies to $\boldsymbol{g} = \boldsymbol{0}$.

At a fixed point $\dot{\boldsymbol{\xi}} = \boldsymbol{0}$ and $\dot{\delta} = \omega_2 = 0$, requiring that any solution to 3.5 must result in $\boldsymbol{g} = 0$. Both $\boldsymbol{g}_1 = 0$ and $\boldsymbol{g}_2 = 0$ can be solved by setting

$$\delta^* = \cos^{-1}\left(\frac{\epsilon\,(\epsilon-2)}{\epsilon^2 - 2\,\epsilon + 1}\right), \tag{6}$$

which gives a turning gait, or

$$\omega_1^* = 0, \tag{7}$$

which results in straight line motion. The equation $\boldsymbol{g}_4 = 0$ is satisfied if $\omega_2 = 0$. The remaining equation $\boldsymbol{g}_3 = 0$ can be satisfied for infinitely many adjacent (u, ω) pairs, resulting in non-isolated fixed points extending away from the constant energy manifold. Evaluating the dynamics reduced to the energy manifold results in isolated fixed points which can then be analyzed for stability. This reduction is performed by defining the system energy

$$E = \mathcal{T}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathcal{V}(\boldsymbol{q}) \tag{8}$$

which, rewritten in $\boldsymbol{\xi}$ coordinates, is

$$E' = \frac{E}{m\,l^2} = \frac{1}{2}\left(u_x^2 + b\,u_x + c\right) \tag{9}$$

for

$$b = m_2'(\epsilon - 1)(\omega_1 + \omega_2)\sin\delta \tag{10}$$

$$\tag{11}$$

and

$$c = \left((I_1' + \frac{m_1'\epsilon^2}{4} + m_2'\epsilon^2)\omega_1^2 + \left(I_2' + \frac{m_2'(\epsilon-1)^2}{4}\right)(\omega_1 + \dot{\delta})^2 - m_2'(\epsilon - 1)(\omega_1 + \omega_2)(\omega_1 \epsilon \cos\delta)\right). \tag{12}$$

With energy rescaled, we now once again drop the $'$ superscripts and work only with the rescaled variables. The longitudinal velocity can be written as a function of $(\omega_1, \omega_2, \delta)$ and the energy. The reduced dynamical system is then obtained from the last three equations of (3.5) where u_x is substituted by a function of the other three state variables, $u_x(\omega_1, \omega_2, \delta; E)$. Suppose $\boldsymbol{H}(\boldsymbol{\xi}) = \boldsymbol{\mathcal{N}}^{-1}\boldsymbol{h}(\boldsymbol{\xi}) \in \mathbb{R}^4$. Setting $\boldsymbol{H}_r(\boldsymbol{\xi}; E) = [\boldsymbol{H}_2(\boldsymbol{\xi}), \boldsymbol{H}_3(\boldsymbol{\xi}), \boldsymbol{H}_4(\boldsymbol{\xi})]^T$, which excludes the first component of $\boldsymbol{H}(\boldsymbol{\xi})$, the reduced dynamical system is

$$\dot{\boldsymbol{\xi}}_r = \boldsymbol{H}_r(\boldsymbol{\xi}_r; E) \tag{13}$$

where $\boldsymbol{\xi}_r = (\omega_1, \omega_2, \delta)$. We then define the fixed points as a function of energy by substituting $u_x(\omega_1, \omega_2, \delta, E)$ into $\boldsymbol{g}_3 = 0$, which gives an expression which can be solved for ω_1^* as

$$\omega_1^* = \pm \frac{\sqrt{2}}{2\, m_2} \sqrt{A \left(B \pm \sqrt{C} \right)} \tag{14}$$

for

$$\begin{aligned}
A &= \Big(\Big(\epsilon^2 - \frac{\epsilon}{2} + \frac{1}{4} \Big) m_2 - \frac{3\,\epsilon^2}{4} + I_1 + I_2 \Big) ((\cos(\delta))^2 - m_2\, \epsilon\, (-1+\epsilon) \cos(\delta) + \epsilon^2)^{-1} (-1+\epsilon)^{-2} \\
B &= ((-\delta^2 \alpha^2 + 2\, E) \epsilon^2 + (2\, \delta^2 \alpha^2 - 4\, E) \epsilon - \delta^2 \alpha^2 + 2\, E) {m_2}^2 (\cos(\delta))^2 \\
&\quad + (-2\, \alpha^2 \delta\, \epsilon^2 + 4\, \alpha^2 \delta\, \epsilon - 2\, \alpha^2 \delta) {m_2}^2 \sin(\delta) \cos(\delta) + (4\, \alpha^2 \delta\, \epsilon^2 - 4\, \alpha^2 \delta\, \epsilon) m_2 \sin(\delta) \\
C &= ({m_2}^2 (-1+\epsilon)^2 \Big(\Big(-\frac{\delta^2}{2} + \delta \Big) \alpha^2 + E \Big) \Big(\Big(-\frac{\delta^2}{2} - \delta \Big) \alpha^2 + E \Big) (\cos(\delta))^2 \\
&\quad - 2\, \alpha^2 (m_2\, (-1+\epsilon)(-1/2\, \delta^2 \alpha^2 + E) \sin(\delta) - 2\, \alpha^2 \delta\, \epsilon) m_2\, (-1+\epsilon) \delta\, \cos(\delta) \\
&\quad + 4\, \alpha^2 \delta\, (m_2\, (-1+\epsilon)(-1/2\, \delta^2 \alpha^2 + E) \epsilon\, \sin(\delta) - \alpha^2 \delta\, (-1/4\, {m_2}^2 (-1+\epsilon)^2 \\
&\quad + (\epsilon^2 - \epsilon/2 + 1/4) m_2 + 1/4\, \epsilon^2 + I_1 + I_2))) (\cos(\delta))^2 {m_2}^2 (-1+\epsilon)^2.
\end{aligned}$$

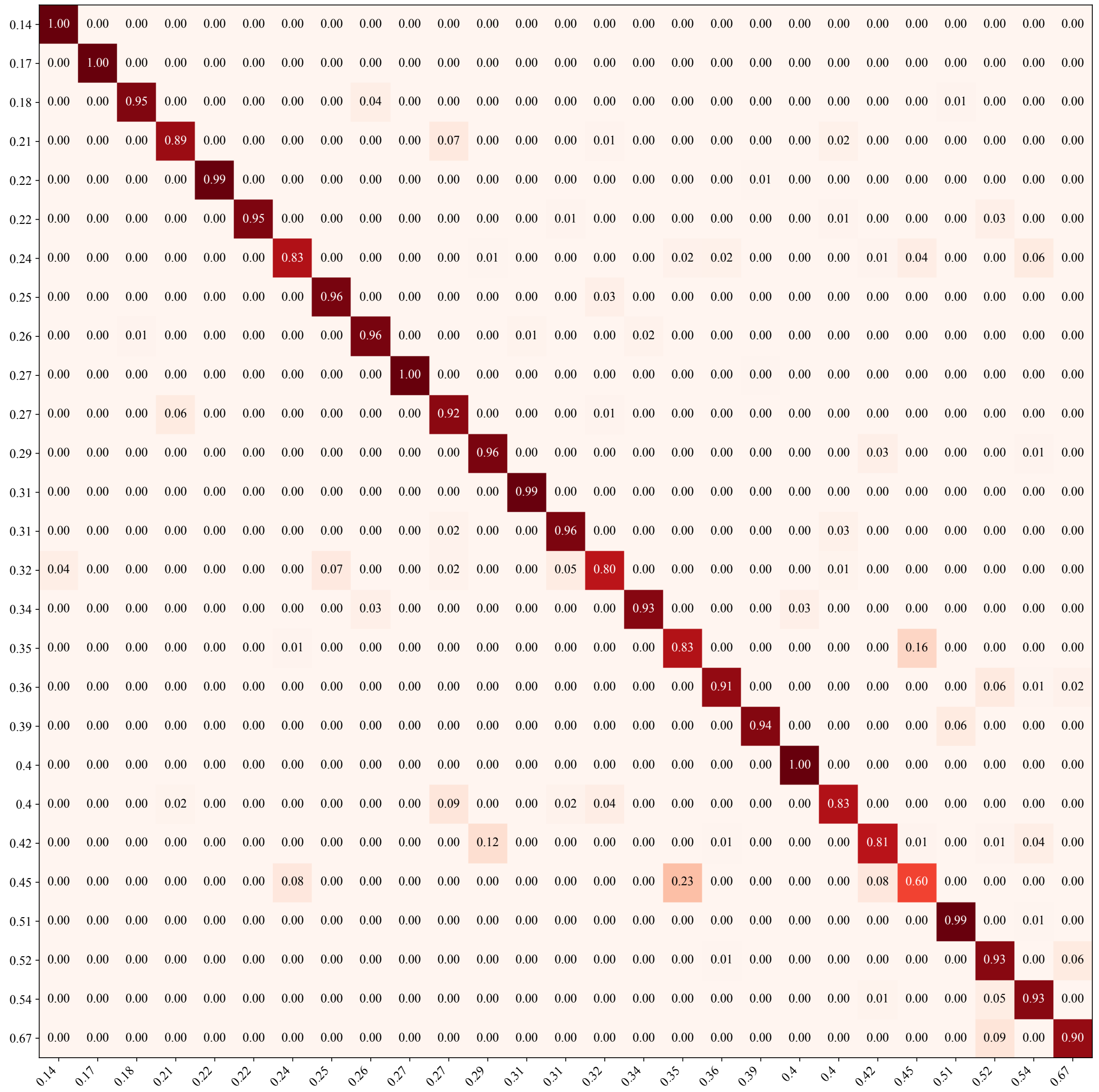
Through straightforward manipulation of equation 9 we can find

$$u_x = \frac{1}{2} \left(-b \pm \sqrt{b^2 - 4\,(c - 2E)} \right). \tag{15}$$

Substituting $b = b(\omega_1 = \omega_1^*)$ and $c = c(\omega_1 = \omega_1^*)$ into 15 then yields u_x^*.

Appendix B Kinematic Classification Confusion Matrices

Below are the confusion matrices for classification fo the Strouhal number using the body kinematics, corresponding to the work in chapter 4.



	0.14	0.17	0.18	0.21	0.22	0.22	0.24	0.25	0.26	0.27	0.27	0.29	0.31	0.31	0.32	0.34	0.35	0.36	0.39	0.4	0.4	0.42	0.45	0.51	0.52	0.54	0.67
0.14	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.17	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.18	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
0.21	0.00	0.00	0.00	0.89	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00
0.22	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.22	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.03	0.00	0.00
0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.83	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.02	0.02	0.00	0.00	0.00	0.01	0.04	0.00	0.00	0.06	0.00
0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.26	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.01	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.27	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.01	0.00
0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
0.32	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.02	0.00	0.00	0.05	0.80	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.34	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.35	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.83	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.00	0.00	0.00
0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.01	0.02
0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.94	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.02	0.04	0.00	0.00	0.00	0.00	0.00	0.83	0.00	0.00	0.00	0.00	0.00	0.00
0.42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.81	0.01	0.00	0.01	0.04	0.00
0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.00	0.00	0.00	0.00	0.08	0.60	0.00	0.00	0.00	0.00
0.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.01	0.00
0.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00	0.06
0.54	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.05	0.93	0.00
0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.90

Figure 1: Confusion matrix for the classification of the Strouhal number using the coupled head kinematics. A clear diagonal can be seen that remains roughly consistent for the entire range of St.

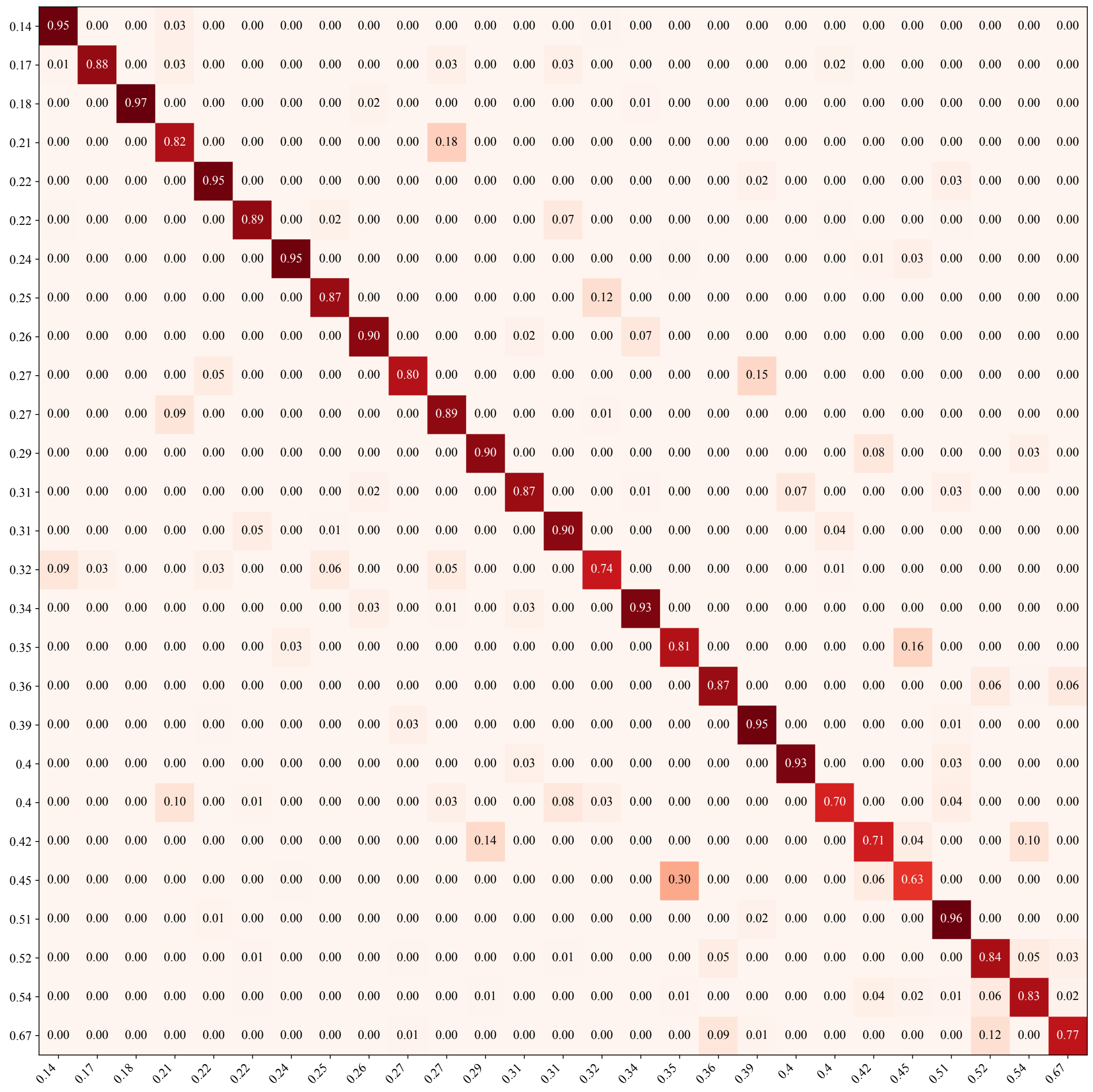
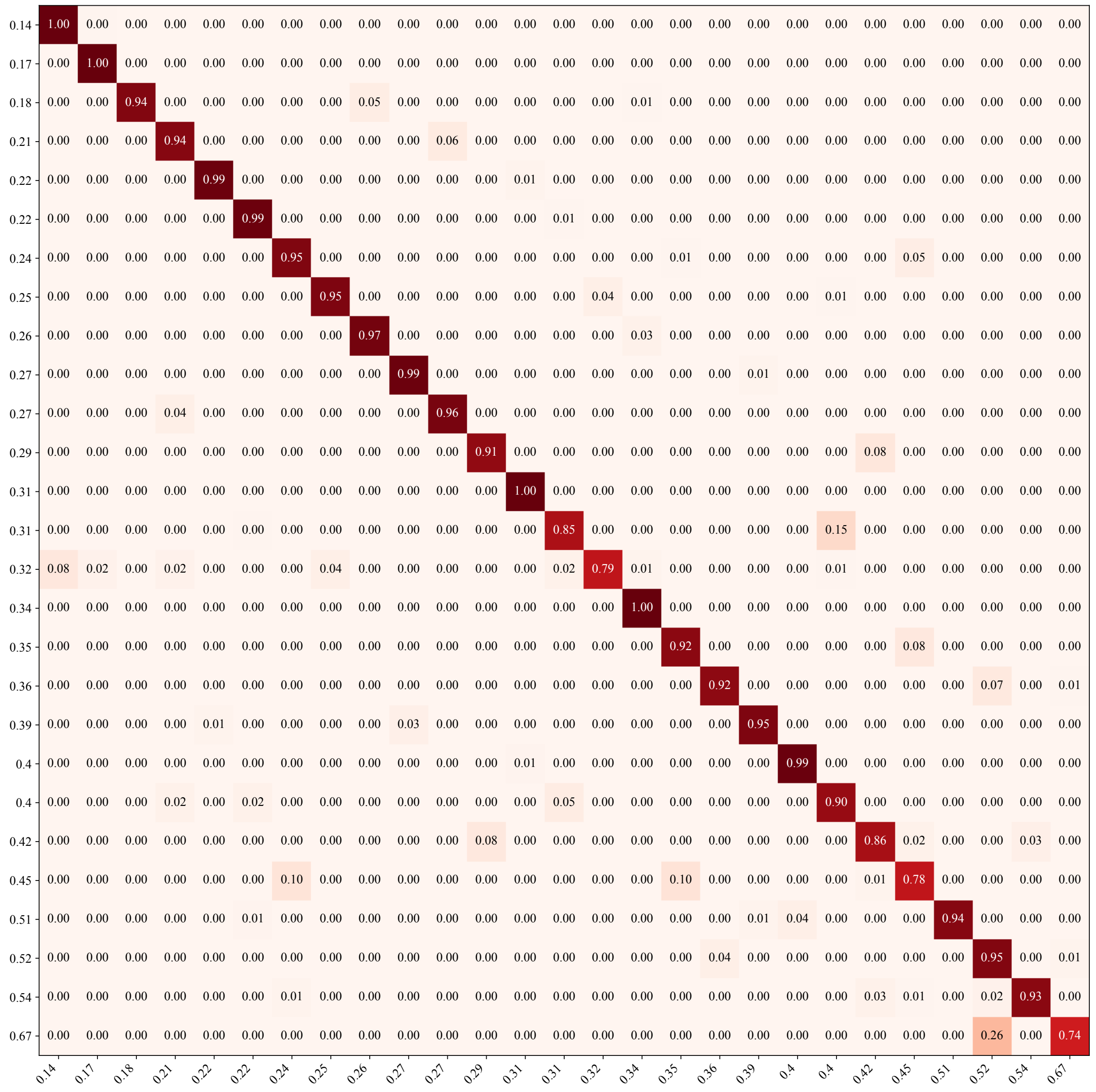


Figure 2: Confusion matrix for the classification of the Strouhal number using the coupled tail kinematics.



	0.14	0.17	0.18	0.21	0.22	0.22	0.24	0.25	0.26	0.27	0.27	0.29	0.31	0.31	0.32	0.34	0.35	0.36	0.39	0.4	0.4	0.42	0.45	0.51	0.52	0.54	0.67
0.14	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.17	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.18	0.00	0.00	0.94	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.21	0.00	0.00	0.00	0.94	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.22	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.22	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00
0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.27	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00
0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.85	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00	0.00
0.32	0.08	0.02	0.00	0.02	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.02	0.79	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.34	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00
0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.01
0.39	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00
0.42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.86	0.02	0.00	0.00	0.03	0.00
0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.01	0.78	0.00	0.00	0.00	0.00
0.51	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.00	0.00	0.00	0.94	0.00	0.00	0.00
0.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.01
0.54	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.01	0.00	0.02	0.93	0.00
0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.26	0.00	0.74

Figure 3: Confusion matrix for the classification of the Strouhal number using combined coupled head and coupled tail kinematics. The additional data allows better classification accuracy than using either set of kinematic data alone.

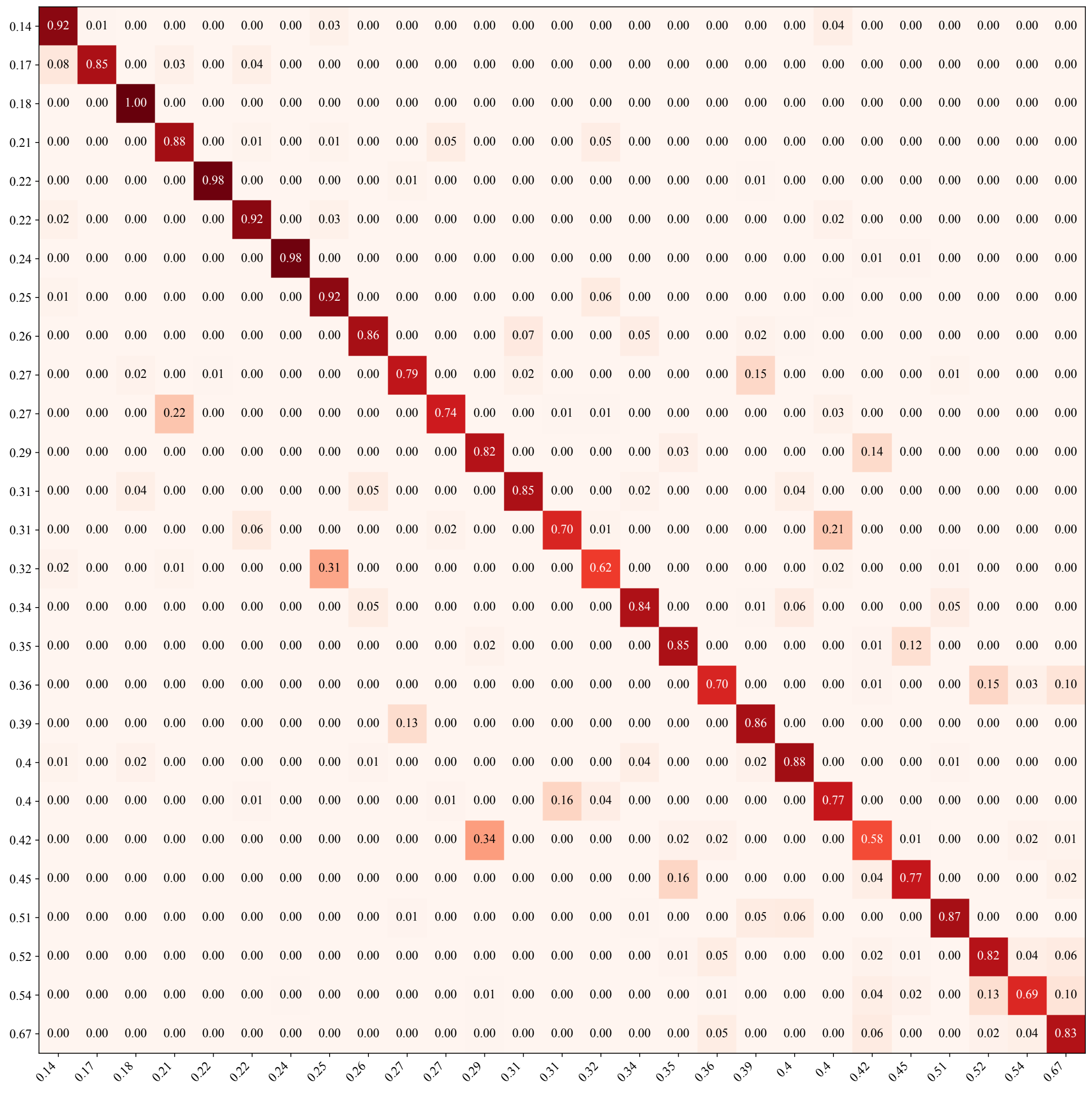
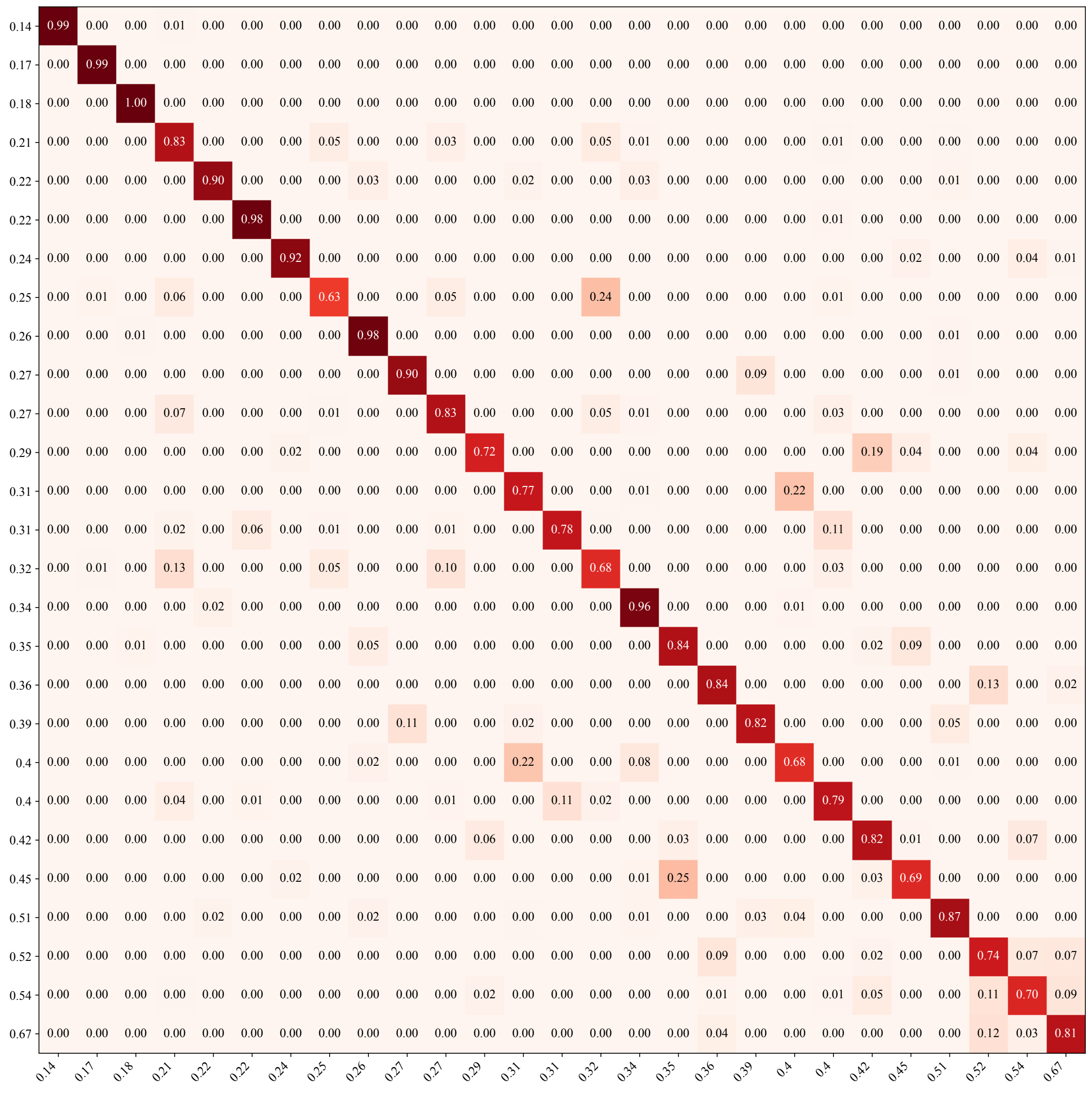
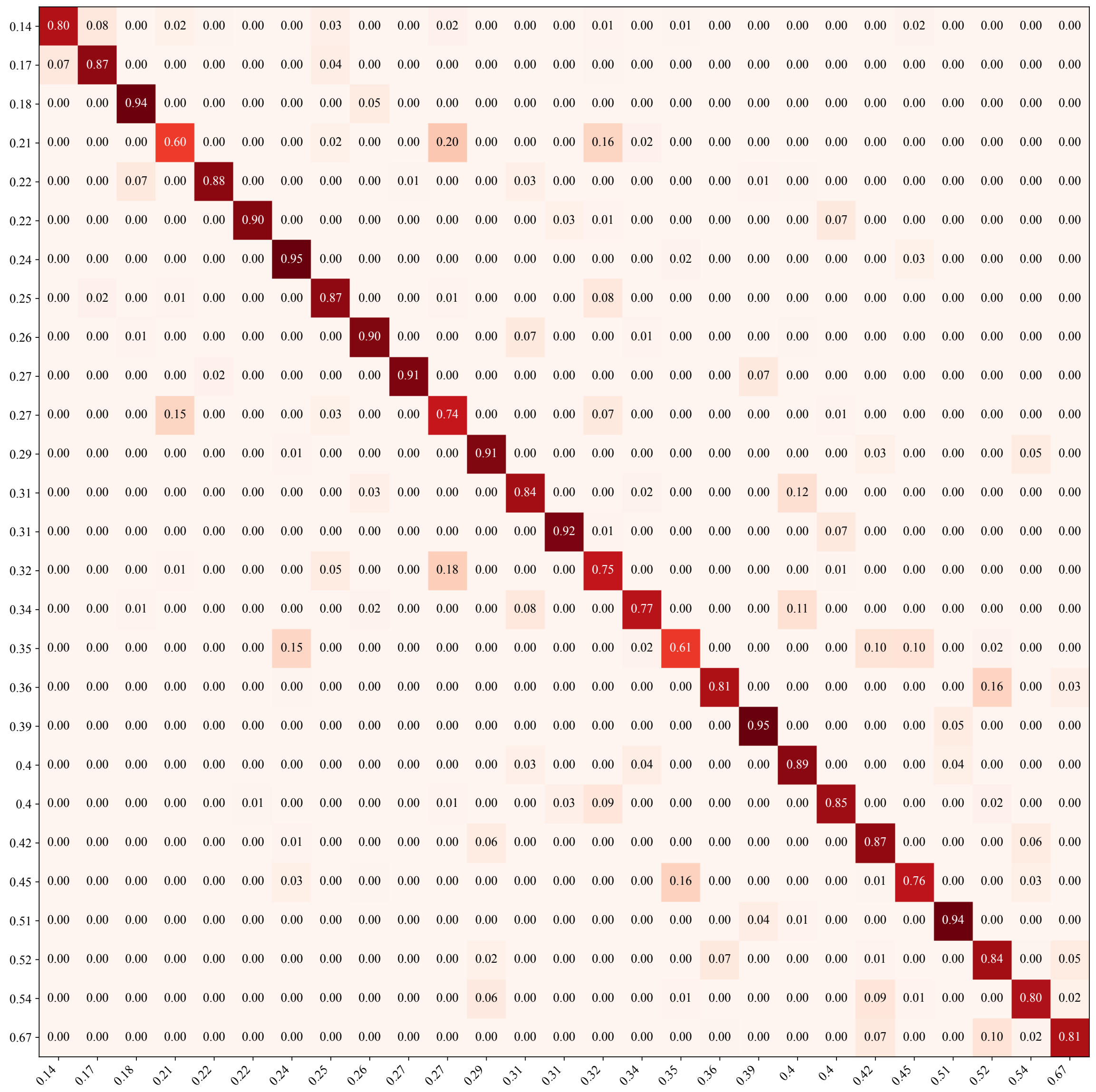


Figure 4: Confusion matrix for the classification of the Strouhal number using the kinematics of the fixed assembly.



	0.14	0.17	0.18	0.21	0.22	0.22	0.24	0.25	0.26	0.27	0.27	0.29	0.31	0.31	0.32	0.34	0.35	0.36	0.39	0.4	0.4	0.42	0.45	0.51	0.52	0.54	0.67
0.14	0.99	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.17	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.18	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.21	0.00	0.00	0.00	0.83	0.00	0.00	0.00	0.05	0.00	0.00	0.03	0.00	0.00	0.00	0.05	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.22	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.02	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
0.22	0.00	0.00	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.04	0.01
0.25	0.00	0.01	0.00	0.06	0.00	0.00	0.00	0.63	0.00	0.00	0.05	0.00	0.00	0.00	0.24	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.26	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
0.27	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.01	0.00	0.00	0.83	0.00	0.00	0.00	0.05	0.01	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.72	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.19	0.04	0.00	0.00	0.04	0.00
0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.77	0.00	0.00	0.01	0.00	0.00	0.00	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.31	0.00	0.00	0.00	0.02	0.00	0.06	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.78	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00	0.00
0.32	0.00	0.01	0.00	0.13	0.00	0.00	0.00	0.05	0.00	0.00	0.10	0.00	0.00	0.00	0.68	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
0.34	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.35	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.84	0.00	0.00	0.00	0.00	0.02	0.09	0.00	0.00	0.00	0.00
0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.84	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.02
0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.82	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.22	0.00	0.00	0.08	0.00	0.00	0.00	0.68	0.00	0.00	0.00	0.01	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.04	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.11	0.02	0.00	0.00	0.00	0.00	0.00	0.79	0.00	0.00	0.00	0.00	0.00	0.00
0.42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.82	0.01	0.00	0.00	0.07	0.00
0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.25	0.00	0.00	0.00	0.00	0.03	0.69	0.00	0.00	0.00	0.00
0.51	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.03	0.04	0.00	0.00	0.00	0.87	0.00	0.00	0.00
0.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.00	0.02	0.00	0.00	0.74	0.07	0.07
0.54	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.05	0.00	0.00	0.11	0.70	0.09
0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.03	0.81

Figure 5: Confusion matrix for the classification of the Strouhal number using the kinematics of the uncoupled head.



	0.14	0.17	0.18	0.21	0.22	0.22	0.24	0.25	0.26	0.27	0.27	0.29	0.31	0.31	0.32	0.34	0.35	0.36	0.39	0.4	0.4	0.42	0.45	0.51	0.52	0.54	0.67
0.14	0.80	0.08	0.00	0.02	0.00	0.00	0.00	0.03	0.00	0.00	0.02	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
0.17	0.07	0.87	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.18	0.00	0.00	0.94	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.21	0.00	0.00	0.00	0.60	0.00	0.00	0.00	0.02	0.00	0.00	0.20	0.00	0.00	0.00	0.16	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.22	0.00	0.00	0.07	0.00	0.88	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.22	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.01	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00
0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00
0.25	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.87	0.00	0.00	0.01	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.26	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.07	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.27	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.27	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.03	0.00	0.00	0.74	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.05	0.00
0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.84	0.00	0.00	0.02	0.00	0.00	0.00	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.01	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00
0.32	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.05	0.00	0.00	0.18	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.34	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.08	0.00	0.00	0.77	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.35	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.61	0.00	0.00	0.00	0.00	0.10	0.10	0.00	0.02	0.00	0.00
0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.81	0.00	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.03
0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.04	0.00	0.00	0.00	0.89	0.00	0.00	0.00	0.04	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.03	0.09	0.00	0.00	0.00	0.00	0.00	0.85	0.00	0.00	0.00	0.02	0.00	0.00
0.42	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.87	0.00	0.00	0.00	0.06	0.00
0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.00	0.00	0.00	0.01	0.76	0.00	0.00	0.03	0.00
0.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.01	0.00	0.00	0.00	0.94	0.00	0.00	0.00
0.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.01	0.00	0.00	0.84	0.00	0.05
0.54	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.09	0.01	0.00	0.00	0.80	0.02
0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.10	0.02	0.81

Figure 6: Confusion matrix for the classification of the Strouhal number using the kinematics of the uncoupled tail.

Bibliography

[1] G. V. Lauder, "Fish locomotion: recent advances and new directions," *Annual review of marine science*, vol. 7, pp. 521–545, 2015.

[2] M. S. Triantafyllou, G. D. Weymouth, and J. Miao, "Biomimetic survival hydrodynamics and flow sensing," *Annual Review of Fluid Mechanics*, vol. 48, no. 1, 2016.

[3] G. V. Lauder, P. G. A. Madden, J. L. Tangorra, E. Anderson, and T. V. Baker, "Bioinspiration from fish for smart material design and function," *Smart Material Structures*, vol. 20, Sept. 2011.

[4] M. S. Triantafyllou, G. S. Triantafyllou, and D. K. P. Yue, "Hydrodynamics of fishlike swimming.," *Annual Reviews of Fluid Mechanics*, vol. 32, pp. 33–53, 2000.

[5] M. S. Triantafyllou and G. Triantafyllou, "An efficient swimming machine.," *Scientific American*, vol. 272, no. 3, p. 64, 1995.

[6] J. Zhu, C. White, D. K. Wainwright, V. Di Santo, G. V. Lauder, and H. Bart-Smith, "Tuna robotics: A high-frequency experimental platform exploring the performance space of swimming fishes," *Science Robotics*, vol. 4, no. 34, p. eaax4615, 2019.

[7] C. H. White, G. V. Lauder, and H. Bart-Smith, "Tunabot flex: a tuna-inspired robot with body flexibility improves high-performance swimming," *Bioinspiration & Biomimetics*, vol. 16, no. 2, p. 026019, 2021.

[8] E. Kelasidi, P. Liljeback, K. Y. Pettersen, and J. T. Gravdahl, "Innovation in underwater robots: Biologically inspired swimming snake robots," *IEEE Robotics & Automation Magazine*, vol. 23, no. 1, pp. 44–62, 2016.

[9] F. Boyer, M. Porez, A. Leroyer, and M. Visonneau, "Fast dynamics of an eel-like robot—comparisons with navier–stokes simulations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1274–1288, 2008.

[10] Z. Chen, S. Shatara, and X. Tan, "Modeling of biomimetic robotic fish propelled by an ionic polymer-metal composite caudal fin," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 3, pp. 448–459, 2010.

[11] Z. Chen, T. I. Um, and H. Bart-Smith, "Bio-inspired robotic manta ray powered by ionic polymer–metal composite artificial muscles," *International Journal of Smart and Nano Materials*, vol. 3, no. 4, pp. 296–308, 2012.

[12] J. Shintake, V. Cacucciolo, H. Shea, and D. Floreano, "Soft biomimetic sh robot made of dielectric elastomer actuators," *Soft Robotics*, vol. 5, no. 4, pp. 466–474, 2018.

[13] A. D. Marchese, C. D. Onal, and D. Rus, "Autonomous soft robotic sh capable of escape maneuvers using fluidic elastomer actuators," *Soft Robotics*, vol. 1, no. 1, pp. 75–87, 2014.

[14] T. Chen, O. R. Bilal, K. Shea, and C. Daraio, "Harnessing bistability for directional propulsion of soft, untethered robots," *Proceedings of the National Academy of Sciences*, vol. 115, no. 22, pp. 5698–5702, 2018.

[15] B. Pollard and P. Tallapragada, "An aquatic robot propelled by an internal rotor," *IEEE/ASME Transaction on Mechatronics*, vol. 22, no. 2, pp. 931–939, 2017.

[16] B. A. Free, J. Lee, and D. A. Paley, "Bioinspired pursuit with a swimming robot using feedback control of an internal rotor," *Bioinspiration and Biomimetics*, vol. 15, no. 3, p. 035005, 2020.

[17] W.-K. Yen, C.-F. Huang, H.-R. Chang, and J. Guo, "Localization of a leading robotic fish using a pressure sensor array on its following vehicle," *Bioinspiration & Biomimetics*, vol. 16, no. 1, p. 016007, 2020.

[18] A. T. Abdulsadda and X. Tan, "Underwater tracking of a moving dipole source using an artificial lateral line: algorithm and experimental validation with ionic polymer–metal composite flow sensors," *Smart Materials and Structures*, vol. 22, no. 4, p. 045010, 2013.

[19] B. J. Wolf, J. van de Wolfshaar, and S. M. van Netten, "Three-dimensional multi-source localization of underwater objects using convolutional neural networks for artificial lateral lines," *Journal of the Royal Society Interface*, vol. 17, no. 162, p. 20190616, 2020.

[20] B. Colvert, M. Alsalman, and E. Kanso, "Classifying vortex wakes using neural networks," *Bioinspiration & biomimetics*, vol. 13, no. 2, p. 025003, 2018.

[21] B. L. R. Ribeiro and J. Franck, "A machine learning approach to classify kinematics and vortex wake modes of oscillating foils," *AIAA AVIATION 2021 FORUM*, 2021.

[22] R. Sutton, "The bitter lesson," *Incomplete Ideas (blog)*, vol. 13, no. 1, 2019.

[23] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[24] C. Rodwell, *The Frequency-Amplitude Response of a Class of Nonholonomic Systems.* PhD thesis, Clemson University, 2020.

[25] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[26] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[27] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*, pp. 651–673, PMLR, 2018.

[28] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2019.

[29] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979, 2019.

[30] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *2018 International Conference on Robotics and Automation (ICRA)*, 2018.

[31] H. S. Choi and et.al., "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward," *Proceedings of the National Academy of Sciences*, vol. 118, no. 1, 2020.

[32] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017.

[33] H. Tang, J. Rabault, A. Kuhnle, Y. Wang, and T. Wang, "Robust active flow control over a range of reynolds numbers using an artificial neural network trained through deep reinforcement learning," *Physics of Fluids*, vol. 32, no. 5, p. 053605, 2020.

[34] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, and E. Hachem, "A review on deep reinforcement learning for fluid mechanics," *Computers & Fluids*, vol. 225, p. 104973, 2021.

[35] S. Verma, G. Novati, and P. Koumoutsakos, "Efficient collective swimming by harnessing vortices through deep reinforcement learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 23, pp. 5849–5854, 2018.

[36] H. Yu, B. Liu, C. Wang, X. Liu, X.-Y. Lu, and H. Huang, "Deep-reinforcement-learning-based self-organization of freely undulatory swimmers," *Physical Review E*, vol. 105, no. 4, p. 045105, 2022.

[37] Q. Wang, Z. Hong, and Y. Zhong, "Learn to swim: Online motion control of an underactuated robotic eel based on deep reinforcement learning," *Biomimetic Intelligence and Robotics*, vol. 2, no. 4, p. 100066, 2022.

[38] I. Mandralis, P. Weber, G. Novati, and P. Koumoutsakos, "Learning swimming escape patterns for larval fish under energy constraints," *Physical Review Fluids*, vol. 6, no. 9, p. 093101, 2021.

[39] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *26th International Conference on Machine Learning*, 2009.

[40] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," *arXiv:2101.10382*, 2021.

[41] D. Weinshall, G. Cohen, and D. Amir, "Curriculum learning by transfer learning: Theory and experiments with deep networks," *arXiv:1802.03796*, 2018.

[42] F. Ren, J. Rabault, and H. Tang, "Applying deep reinforcement learning to active flow control in weakly turbulent conditions," *Physics of Fluids*, vol. 33, no. 3, p. 037121, 2021.

[43] S. Childress, *Mechanics of swimming and flying.* Cambridge University Press, 1981.

[44] P. Tallapragada, "A swimming robot with an internal rotor as a nonholonomic system," *Proceedings of the American Control Conference, 2015*, 2015.

[45] P. Tallapragada and S. D. Kelly, "Integrability of velocity constraints modeling vortex shedding in ideal fluids," *Journal of Computational and Nonlinear Dynamics*, vol. 12, no. 2, p. 021008, 2016.

[46] B. Pollard, V. Fedonyuk, and P. Tallapragada, "Swimming on limit cycles with nonholonomic constraints," *Nonlinear Dynamics*, vol. 97, no. 4, p. 2453 – 2468, 2019.

[47] V. Fedonyuk and P. Tallapragada, "Path tracking for the dissipative chaplygin sleigh," in *2020 American Control Conference (ACC)*, 2020.

[48] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, R. Collobert, and J. Weston, "Deterministic policy gradient algorithms," in *31st International Conference on Machine Learning*, 2014.

[49] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations*, 2016.

[50] J. M. Osborne and D. V. Zenkov, "Steering the chaplygin sleigh by a moving mass," in *Proceedings of the American Control Conference*, 2005.

[51] L. M. Milne-Thomson, *Theoretical Hydrodynamics.* Dover, 1996.

[52] V. Fedonyuk and P. Tallapragada, "Sinusoidal control and limit cycle analysis of the dissipative chaplygin sleigh," *Nonlinear Dynamics*, 2018.

[53] J. Katz and A. Plotkin, *Low-Speed Aerodynamics.* Cambridge University Press, 2001.

[54] L. L. Erickson, "Panel Methods - An Introduction," *NASA Technical Paper*, 1990.

[55] P. Watts and F. Fish, "The influence of passive, leading edge tubercles on wing performance," in *Proc. Twelfth Intl. Symp. Unmanned Untethered Submers. Technol*, Auton. Undersea Syst. Inst. Durham New Hampshire, 2001.

[56] J. M. Anderson, K. Streitlien, D. Barrett, and M. S. Triantafyllou, "Oscillating foils of high propulsive efficiency," *Journal of Fluid mechanics*, vol. 360, pp. 41–72, 1998.

[57] M. Wolfgang, J. Anderson, M. Grosenbaugh, D. Yue, and M. Triantafyllou, "Near-body flow dynamics in swimming fish," *Journal of Experimental Biology*, vol. 202, no. 17, pp. 2303–2327, 1999.

[58] K. W. Moored, "Unsteady three-dimensional boundary element method for self-propelled bio-inspired locomotion," *Computers & Fluids*, vol. 167, pp. 324–340, 2018.

[59] S. Chakravarty and D. Samanta, "Numerical simulation of a one-dimensional flexible filament mimicking anguilliform mode of swimming using discrete vortex method," *Physical Review Fluids*, vol. 6, no. 3, p. 033102, 2021.

[60] F. Ayancik, K. Moored, and F. E. Fish, "Disentangling the relation between the planform shape and swimming gait in cetacean propulsion," in *2018 Fluid Dynamics Conference*, p. 2914, 2018.

[61] B. Pollard, *Improving Swimming Performance and Flow Sensing by Incorporating Passive Mechanisms.* PhD thesis, Clemson University, 2020.

[62] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[63] W. Wang and G. Xie, "Cpg-based locomotion controller design for a boxfish-like robot," *International Journal of Advanced Robotic Systems*, vol. 11, no. 6, p. 87, 2014.

[64] J. Yu, Z. Wu, M. Wang, and M. Tan, "Cpg network optimization for a biomimetic robotic fish via pso," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 9, pp. 1962–1968, 2015.

[65] J. M. Snider *et al.*, "Automatic steering methods for autonomous automobile path tracking," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[66] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," tech. rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[67] J. Lee, B. Free, S. Santana, and D. A. Paley, "State-feedback control of an internal rotor for propelling and steering a flexible fish-inspired underwater vehicle," in *Proceedings of the American Control Conference*, pp. 2011–2016, IEEE, 2019.

[68] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[69] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2219–2225, IEEE, 2006.

[70] D. Silver, A. Huang, C. Maddison, and et.al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, p. 484–489, 2016.

[71] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[72] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015.

[73] J. J. Videler and D. Weihs, "Energetic advantages of burst-and-coast swimming of fish at high speeds," *Journal of Experimental Biology*, vol. 97, no. 1, pp. 169–178, 1982.

[74] Q. Zhong, J. Zhu, F. E. Fish, S. J. Kerr, A. M. Downs, H. Bart-Smith, and D. B. Quinn, "Tunable stiffness enables fast and efficient swimming in fish-like robots," *Science Robotics*, vol. 6, no. 57, 2021.

[75] C. Rodwell and P. Tallapragada, "Induced and tunable multistability due to nonholonomic constraints," *Nonlinear Dynamics*, vol. 108, no. 3, pp. 2115–2126, 2022.

[76] V. Fedonyuk and P. Tallapragada, "The dynamics of a chaplygin sleigh with an elastic internal rotor," *Regular and Chaotic Dynamics*, vol. 24, no. 1, pp. 114–126, 2019.

[77] L. Meng, R. Kang, D. Gan, G. Chen, L. Chen, D. T. Branson, and J. S. Dai, "A mechanically intelligent crawling robot driven by shape memory alloy and compliant bistable mechanism," *Journal of Mechanisms and Robotics*, vol. 12, no. 6, 2020.

[78] Y. Tang, Y. Chi, J. Sun, T. Z. Huang, O. H. Maghsoudi, A. Spence, J. Zhao, H. Su, and J. Yin, "Leveraging elastic instabilities for amplified performance: Spine-inspired high-speed and high-force soft robots," *Science Advances*, vol. 6, no. 19, 2020.

[79] H. Zhang, J. Sun, and J. Zhao, "Compliant bistable gripper for aerial perching and grasping," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1248–1253, IEEE, 2019.

[80] G. Wan, Y. Liu, Z. Xu, C. Jin, L. Dong, X. Han, J. X. J. Zhang, and Z. Chen, "Tunable bistability of a clamped elastic beam," *Extreme Mechanics Letters*, vol. 34, p. 100603, 2020.

[81] A. Cazzolli, D. Misseroni, and F. D. Corso, "Elastica catastrophe machine: theory, design and experiments," 2020.

[82] Z. Meng, W. Chen, T. Mei, Y. Lai, Y. Li, and C. Q. Chen, "Bistability-based foldable origami mechanical logic gates," *Extreme Mechanics Letters*, vol. 43, p. 101180, 2021.

[83] L. L. Howell, *Compliant Mechanisms.* Wiley, New York, 2001.

[84] T. Greigarn and M. Cavusoglu, "Pseudo-rigid-body model and kinematic analysis of mri-actuated catheters," in *2015 IEEE International Conference on Robotics and Automation*, 2015.

[85] V. K. Venkiteswaran and H. J. Su, "A three-spring pseudorigid-body model for soft joints with significant elongation effects," *Journal of Mechanisms and Robotics*, vol. 8, no. 6, 2016.

[86] R. K. Katzschmann, C. D. Santina, Y. Toshimitsu, A. Bicchi, and D. Rus, "Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 454–461, IEEE, 2019.

[87] H. Zhang, B. Zhu, and X. Zhang, "Origami kaleidocycle-inspired symmetric multistable compliant mechanisms," *Journal of Mechanisms and Robotics*, vol. 1, no. 11, p. 011009, 2019.

[88] B. Zhu, X. Zhang, H. Zhang, J. Liang, H. Zang, H. Li, and R. Wang, "Design of compliant mechanisms using continuum topology optimization: a review," *Mechanism and Machine Theory*, vol. 143, p. 103622, 2020.

[89] A. Cazzolli, F. D. Corso, and D. Bigoni, "Flutter instability and ziegler destabilization paradox for elastic rods subject to non-holonomic constraints," vol. 88, no. 3, p. 031003, 2021.

[90] S. A. Chaplygin, "On the theory of the motion of nonholonomic systems : The reducing multiplier theorem.," *Translated version in Regular and Chaotic Dynamics*, 2008.

[91] A. V. Borisov and I. S. Mamaev, "On the history of the development of the nonholonomic dynamics," *Regular and Chaotic Dynamics*, vol. 7, no. 1, pp. 43–47, 2002.

[92] A. M. Bloch, *Nonholonomic Mechanics and Control.* Springer Verlag, 2003.

[93] S. D. Kelly, M. J. Fairchild, P. M. Hassing, and P. Tallapragada, "Proportional heading control for planar navigation: The chaplygin beanie and fishlike robotic swimming," in *Proceedings of the American Control Conference*, 2012.

[94] V. Fedonyuk and P. Tallapragada, "Sinusoidal control and limit cycle analysis of the dissipative chaplygin sleigh," *Nonlinear Dynamics*, pp. 1–12, 2018.

[95] C. dynamics of the Chaplygin sleigh with a passive internal rotor, "V. fedonyuk and p. tallapragada," *Nonlinear Dynamics*, 2018.

[96] J. Náprstek and C. Fischer, "Appell-gibbs approach in dynamics of non-holonomic systems," in *Nonlinear Systems* (M. Reyhanoglu, ed.), ch. 1, Rijeka: IntechOpen, 2018.

[97] J. Náprstek and C. Fischer, *Non-holonomic Systems in View of Hamiltonian Principle*, pp. 3–25. 01 2021.

[98] P. Tallapragada and S. D. Kelly, "Integrability of velocity constraints modeling vortex shedding in ideal fluids," *Journal of Computational and Nonlinear Dynamics*, 2016.

[99] B. Pollard, V. Fedonyuk, and P. Tallapragada, "Swimming on limit cycles with nonholonomic constraints," *Nonlinear Dynamics*, 2019.

[100] J. Lee, B. Free, S. Santana, and D. A. Paley, “State-feedback control of an internal rotor for propelling and steering a flexible fish-inspired underwater vehicle,” in *2019 American Control Conference (ACC)*, pp. 2011–2016, IEEE, 2019.

[101] L. A. Pars, *A Treatise on Analytical Dynamics.* Ox Bow Press, 1965.

[102] A. M. Bloch, P. S. Krishnaprasad, J. E. Marsden, and R. M. Murray, “Nonholonomic Mechanical Systems with Symmetry,” *Archive for Rational Mechanics and Analysis*, vol. 136, pp. 21–99, 1996.

[103] A. Bloch, M. Reyhanoglu, and N. McClamroch, “Control and stabilization of nonholonomic dynamic systems,” *IEEE Transactions on Automatic Control*, vol. 37, no. 11, pp. 1746–1757, 1992.

[104] S. K. Soltakhanov, M. P. Yushkov, and S. A. Zegzhda, *Mechanics of non-holonomic systems.* Springer, Berlin Heidelberg, 2009.

[105] I. A. Bizyaev, A. V. . Borisov, and S. P. Kuznetsov, “The chaplygin sleigh with friction moving due to periodic oscillations of an internal mass,” *Nonlinear Dynamics*, pp. 1–16, 2019.

[106] S. Portugal, T. Hubel, J. Fritz, S. Heese, D. Trobe, , B. Voelkl, S. Hailes, A. M. Wislon, and J. R. Usherwood, “Upwash exploitation and downwash avoidance by flap phasing in ibis formation flight.,” *Journal of Fluid Mechanics*, vol. 505, p. 399–402, 2014.

[107] D. Beal, F. Hover, M. Triantafyllou, J. Liao, and G. Lauder, “Passive propulsion in vortex wakes,” *Journal of Fluid Mechanics*, vol. 549, pp. 385–402, 2006.

[108] F. Fish and G. Lauder, “Passive and active flow control by swimming fishes and mammals,” *Annu. Rev. Fluid Mech.*, vol. 38, pp. 193–224, 2006.

[109] E. D. Tytell and G. V. Lauder, “The hydrodynamics of eel swimming: I. wake structure,” *Journal of Experimental Biology*, vol. 207, no. 11, pp. 1825–1841, 2004.

[110] I. K. Bartol, M. Gharib, P. W. Webb, D. Weihs, and M. S. Gordon, “Body-induced vortical flows: a common mechanism for self-corrective trimming control in boxfishes,” *Journal of Experimental Biology*, vol. 208, no. 2, pp. 327–344, 2005.

[111] F. Fish and G. Lauder, “Not just going with the flow,” *American Scientist*, vol. 101, no. 2, pp. 114–123, 2013.

[112] M. Gazzola, M. Argentina, and L. Mahadevan, “Gait and speed selection in slender inertial swimmers,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 112, no. 13, pp. 3874–3879, 2015.

[113] M. F. Platzer, K. D. Jones, J. Young, and J. C. S. Lai, “Flapping wing aerodynamics: progress and challenges,” *AIAA journal*, vol. 46, no. 9, pp. 2136–2149, 2008.

[114] K. Pohlmann, F. W. Grasso, and T. Breithaupt, “Tracking wakes: The nocturnal predatory strategy of piscivorous catfish,” *Proceedings of the National Academy of Sciences*, vol. 98, no. 13, pp. 7371–7374, 2001.

[115] K. Pohlmann, J. Atema, and T. Breithaupt, “The importance of the lateral line in nocturnal predation of piscivorous catfish,” *Journal of Experimental Biology*, vol. 207, no. 17, pp. 2971–2978, 2004.

[116] T. J. Pitcher, B. L. Partridge, and C. S. Wardle, “A blind fish can school,” *Science*, vol. 194, no. 4268, pp. 963–965, 1976.

[117] J. Liao, D. Beal, G. Lauder, and M. Triantafyllou, "The kármán gait: novel body kinematics of rainbow trout swimming in a vortex street," *Journal of experimental biology*, vol. 206, no. 6, pp. 1059–1073, 2003.

[118] J. C. Liao, "A review of fish swimming mechanics and behaviour in altered flows," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 362, no. 1487, pp. 1973–1993, 2007.

[119] J. C. Liao, "The role of the lateral line and vision on body kinematics and hydrodynamic preference of rainbow trout in turbulent flow," *Journal of Experimental Biology*, vol. 209, no. 20, pp. 4077–4090, 2006.

[120] S. P. Windsor, S. E. Norris, S. M. Cameron, G. D. Mallinson, and J. C. Montgomery, "The flow fields involved in hydrodynamic imaging by blind mexican cave fish (astyanax fasciatus). part i: open water and heading towards a wall," *Journal of Experimental Biology*, vol. 213, no. 22, pp. 3819–3831, 2010.

[121] B. Free and D. A. Paley, "Model-based observer and feedback control design for a rigid joukowski foil in a karman vortex street," *Bioinspiration & biomimetics*, 2017.

[122] R. Venturelli, O. Akanyeti, F. Visentin, J. Ježov, L. D. Chambers, G. Toming, J. Brown, M. Kruusmaa, W. M. Megill, and P. Fiorini, "Hydrodynamic pressure sensing with an artificial lateral line in steady and unsteady flows," *Bioinspiration & biomimetics*, vol. 7, no. 3, p. 036004, 2012.

[123] T. Salumäe and M. Kruusmaa, "Flow-relative control of an underwater robot," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 469, no. 2153, p. 20120671, 2013.

[124] F. D. Lagor, L. D. DeVries, K. Waychoff, and D. A. Paley, "Bio-inspired flow sensing and control: Autonomous rheotaxis using distributed pressure measurements," *Journal of Unmanned System Technology*, vol. 1, no. 3, pp. 78–88, 2013.

[125] D. F. Gomez, F. Lagor, P. B. Kirk, A. Lind, A. R. Jones, and D. A. Paley, "Unsteady dmd-based flow field estimation from embedded pressure sensors in an actuated airfoil," in *AIAA Scitech 2019 Forum*, p. 0346, 2019.

[126] A. Abdulsadda and X. Tan, "Nonlinear estimation-based dipole source localization for artificial lateral line systems," *Bioinspiration & biomimetics*, vol. 8, no. 2, p. 026005, 2013.

[127] A. T. Abdulsadda and X. Tan, "An artificial lateral line system using ipmc sensor arrays," *International Journal of Smart and Nano Materials*, vol. 3, no. 3, pp. 226–242, 2012.

[128] A. Qualtieri, F. Rizzi, G. Epifani, A. Ernits, M. Kruusmaa, and M. De Vittorio, "Parylene-coated bioinspired artificial hair cell for liquid flow sensing," *Microelectronic Engineering*, vol. 98, pp. 516–519, 2012.

[129] Y. Yang, J. Chen, J. Engel, S. Pandya, N. Chen, C. Tucker, S. Coombs, D. L. Jones, and C. Liu, "Distant touch hydrodynamic imaging with an artificial lateral line," *Proceedings of the National Academy of Sciences*, vol. 103, no. 50, pp. 18891–18895, 2006.

[130] X. Zheng, W. Wang, M. Xiong, and G. Xie, "Online state estimation of a fin-actuated underwater robot using artificial lateral line system," *IEEE Transactions on robotics*, vol. 36, no. 2, pp. 472–487, 2020.

[131] G. Dehnhardt, B. Mauck, and H. Bleckmann, "Seal whiskers detect water movements," *Nature*, vol. 394, no. 6690, pp. 235–236, 1998.

[132] A. R. Hardy, B. M. Steinworth, and M. E. Hale, "Touch sensation by pectoral fins of the catfish," *Proceedings of the Royal Society B*, 2016.

[133] R. Williams, N. Neubarth, and M. E. Hale, "The function of fin rays as proprioceptive sensors in fish," *Nature Communications*, vol. 4, p. 1729, 2013.

[134] B. R. Aiello, M. W. Westneat, and M. E. Hale, "Mechanosensation is evolutionarily tuned to locomotor mechanics," *Proceedings of the National Academy of Sciences*, vol. 114, p. 4459–4464, 2017.

[135] B. R. Aiello, A. R. Hardy, M. W. Westneat, and M. E. Hale, "Fins as mechanosensors for movement and touch-related behaviors," *Integrative and Comparative Biology*, vol. 58, no. 5, p. 844–859, 2018.

[136] L. Li, D. Liu, J. Deng, M. J. Lutz, and G. Xie, "Fish can save energy via proprioceptive sensing," *Bioinspiration & Biomimetics*, vol. 16, p. 056013, aug 2021.

[137] B. Pollard and P. Tallapragada, "Passive appendages improve the maneuverability of fish-like robots," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 4, pp. 1586–1596, 2019.

[138] B. Colvert, M. Alsalman, and E. Kanso, "Classifying vortex wakes using neural networks," *Bioinspiration & biomimetics*, vol. 13, no. 2, p. 025003, 2018.

[139] M. Alsalman, B. Colvert, and E. Kanso, "Training bioinspired sensors to classify flows," *Bioinspiration & biomimetics*, vol. 14, no. 1, p. 016009, 2018.

[140] M. Wang and M. S. Hemati, "Detecting exotic wakes with hydrodynamic sensors," *Theoretical and Computational Fluid Dynamics*, vol. 33, pp. 235–254, 2019.

[141] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 2019.

[142] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa, "Next generation reservoir computing," *Nature Communications*, vol. 12, p. 5564, 2021.

[143] B. Pollard and P. Tallapragada, "Learning hydrodynamic signatures through proprioceptive sensing by bioinspired swimmers," *Bioinspiration & Biomimetics*, vol. 16, no. 2, p. 026014, 2021.

[144] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[145] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[146] G. K. Taylor, R. L. Nudds, and A. L. R. Thomas, "Flying and swimming animals cruise at a strouhal number tuned for high power efficiency," *Nature*, vol. 425, pp. 707–711, 2003.

[147] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, pp. 267–285, Springer, 1982.

[148] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[149] L. Prechelt, "Early stopping-but when?," in *Neural Networks: Tricks of the trade*, pp. 55–69, Springer, 1998.

[150] C. Rodwell and P. Tallapragada, "Embodied hydrodynamic sensing and estimation using koopman modes in an underwater environment," in *2022 American Control Conference (ACC)*, pp. 1632–1637, IEEE, 2022.

[151] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?': Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–44, Association for Computing Machinery, 2016.

[152] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, pp. 1–46, 07 2015.

[153] T. J. Pitcher, B. Partridge, and C. S. Wardle, "A blind fish can school," *Science*, vol. 194, no. 4268, pp. 963–965, 1976.

[154] H. Bleckmann and R. Zelick, "Lateral line system of fish," *Integrative zoology*, vol. 4, no. 1, pp. 13–25, 2009.

[155] L. Sirovich, "Turbulence and the dynamics of coherent structures. part 1: Coherent structures," *Quart. Appl. Math*, vol. 45, no. 3, pp. 561–571, 1987.

[156] P. J. H. G. Berkooz and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual Review of Fluid Mechanics*, vol. 25, no. 1, pp. 539–575, 2003.

[157] R. Everson and L. Sirovich, "Karhunen–loève procedure for gappy data," *J. Opt. Soc. Am. A*, vol. 12, no. 8, pp. 1657–1664, 1995.

[158] M. D. T. Bui-Thanh and K. E. Willcox, "Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition.," *AIAA Journal*, vol. 42, no. 8, p. 1505–1516, 2004.

[159] K. E. Willcox, "Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition.," *Computers & Fluids*, vol. 35, no. 2, pp. 208–226, 2006.

[160] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *Journal of fluid mechanics*, vol. 641, pp. 115–127, 2009.

[161] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[162] E. M. B. Q. Li, F. Deitrich and I. G. Kevrekidis, "Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator," *CHAOS*, vol. 27, p. 103111, 2017.

[163] S. Otto and C. Rowley, "Linearly recurrent autoencoder networks for learning dynamics," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 1, p. 558–593, 2019.

[164] J. N. K. K. Champion, B. Lusch and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *PNAS*, vol. 116, no. 45, p. 22445–22451, 2019.

[165] A. Y. M. Raissi and G. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[166] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual Review of Fluid Mechanics*, vol. 52, p. 2020, 2019.

[167] K. M. J. L. Callaham and S. L. Brunton, "Robust flow reconstruction from limited measurements via sparse representation," *Physical Review Fluids*, vol. 4, no. 2, p. 103907, 2019.

[168] B. R. N. S. Brunton and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual Review of Fluid Mechanics*, vol. 52, no. 1, pp. 477–508, 2020.

[169] M. Alsalman, B. Colvert, and E. Kanso, "Training bioinspired sensors to classify flows," *Bioinspiration & biomimetics*, vol. 14, no. 1, p. 016009, 2018.

[170] B. Colvert, M. Alsalman, and E. Kanso, "Classifying vortex wakes using neural networks," *Bioinspiration & biomimetics*, vol. 13, no. 2, p. 025003, 2018.

[171] B. Pollard and P. Tallapragada, "Sensing and classification of ambient vortex wake from the kinematics of a bioinspired swimming robot using neural networks," in *Proceedings of the Dynamic Systems Conference*, 2020.

[172] I. Bright, G. Lin, and J. N. Kutz, "Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements," *Physics of Fluids*, vol. 25, no. 12, p. 127102, 2013.

[173] J. M. Lidard, D. Goswami, D. Snyder, G. Sedky, A. R. Jones, and D. A. Paley, "Data-driven estimation of the unsteady flowfield near an actuated airfoil," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 10, pp. 2279–2287, 2019.

[174] J. M. Lidard, D. Goswami, D. Snyder, G. Sedky, A. R. Jones, and D. A. Paley, "Output feedback control for lift maximization of a pitching airfoil," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 3, pp. 587–594, 2021.

[175] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[176] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, p. 4950, 2018.

[177] L. Li, M. Nagy, J. M. Graving, J. Bak-Coleman, G. Xie, and I. D. Couzin, "Vortex phase matching as a strategy for schooling in robots and in fish," *Nature Communications*, vol. 11, no. 1, p. 5408, 2020.

[178] S. J. Cooke, J. N. Bergman, W. M. Twardek, M. L. Piczak, G. A. Casselberry, K. Lutek, L. S. Dahlmo, K. Birnie-Gauvin, L. P. Griffin, J. W. Brownscombe, *et al.*, "The movement ecology of fishes," *Journal of Fish Biology*, vol. 101, no. 4, pp. 756–779, 2022.

[179] G. Liu, A. Wang, X. Wang, P. Liu, *et al.*, "A review of artificial lateral line in sensor fabrication and bionic applications for robot fish," *Applied Bionics and Biomechanics*, vol. 2016, 2016.

[180] P. Dubois, T. Gomez, L. Planckaert, and L. Perret, "Machine learning for fluid flow reconstruction from limited measurements," *Journal of Computational Physics*, vol. 448, p. 110733, 2022.

[181] N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, and J. N. Kutz, "Shallow neural networks for fluid flow reconstruction with limited sensors," *Proceedings of the Royal Society A*, vol. 476, no. 2238, 2020.

[182] C. Rodwell, B. Pollard, and P. Tallapragada, "Proprioceptive wake classification by a body with a passive tail," *Bioinspiration & Biomimetics*, vol. 18, no. 4, p. 046001, 2023.

[183] M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Deep learning of vortex-induced vibrations," *Journal of Fluid Mechanics*, vol. 861, pp. 119–137, 2019.

[184] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of Fluid Mechanics*, vol. 656, pp. 5–28, 2010.

[185] P. J. Schmid, L. Li, M. P. Juniper, and O. Pust, "Applications of the dynamic mode decomposition," *Theoretical and Computational Fluid Dynamics*, vol. 25, pp. 249–259, 2011.

[186] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics*, vol. 641, pp. 115–127, 2009.

[187] H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian, "Deep neural networks for nonlinear model order reduction of unsteady flows," *Physics of Fluids*, vol. 32, no. 10, 2020.

[188] B. Zhang, "Nonlinear mode decomposition via physics-assimilated convolutional autoencoder for unsteady flows over an airfoil," *Physics of Fluids*, vol. 35, no. 9, 2023.

[189] C. H. Williamson and R. Govardhan, "Vortex-induced vibrations," *Annual Review of Fluid Mechanics*, vol. 36, pp. 413–455, 2004.

[190] M. Zdravkovich, "Flow induced oscillations of two interfering circular cylinders," *Journal of Sound and Vibration*, vol. 101, no. 4, pp. 511–521, 1985.

[191] N. Mahir and D. Rockwell, "Vortex formation from a forced system of two cylinders. part i: Tandem arrangement," *Journal of Fluids and Structures*, vol. 10, no. 5, pp. 473–489, 1996.

[192] J. R. Meneghini, F. Saltara, C. d. L. R. Siqueira, and J. Ferrari Jr, "Numerical simulation of flow interference between two circular cylinders in tandem and side-by-side arrangements," *Journal of Fluids and Structures*, vol. 15, no. 2, pp. 327–350, 2001.

[193] S. Mittal and V. Kumar, "Flow-induced oscillations of two cylinders in tandem and staggered arrangements," *Journal of Fluids and Structures*, vol. 15, no. 5, pp. 717–736, 2001.

[194] H. Jing, X. Min, X. He, and Y. Yang, "Wake-induced interactive vibrations of two tandem cables with a center-to-center distance of 2d," *Ocean Engineering*, vol. 266, p. 113259, 2022.

[195] Z.-S. Chen, S. Wang, and X. Jiang, "Effect of wake interference on vibration response of dual tandem flexible pipe," *Ocean Engineering*, vol. 269, p. 113497, 2023.

[196] M. O. Awadallah, C. Jiang, and O. el Moctar, "Numerical study into the impact of fixed upstream cylinder diameter ratios on vibration of leeward tandem cylinders," *Ocean Engineering*, vol. 285, p. 115367, 2023.

[197] G. R. d. S. Assi, P. W. Bearman, N. Kitney, and M. Tognarelli, "Suppression of wake-induced vibration of tandem cylinders with free-to-rotate control plates," *Journal of Fluids and Structures*, vol. 26, no. 7-8, pp. 1045–1057, 2010.

[198] P. Li, L. Liu, Z. Dong, F. Wang, and H. Guo, "Investigation on the spoiler vibration suppression mechanism of discrete helical strakes of deep-sea riser undergoing vortex-induced vibration," *International Journal of Mechanical Sciences*, vol. 172, p. 105410, 2020.

[199] M. Zdravkovich and D. Pridden, “Interference between two circular cylinders; series of unexpected discontinuities,” *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 2, no. 3, pp. 255–270, 1977.

[200] G. Papaioannou, D. Yue, M. Triantafyllou, and G. Karniadakis, “On the effect of spacing on the vortex-induced vibrations of two tandem cylinders,” *Journal of Fluids and Structures*, vol. 24, no. 6, pp. 833–854, 2008.

[201] D. Kumar, K. Sourav, and S. Sen, “Steady separated flow around a pair of identical square cylinders in tandem array at low reynolds numbers,” *Computers & Fluids*, vol. 191, p. 104244, 2019.

[202] W. Yang and M. A. Stremler, “Critical spacing of stationary tandem circular cylinders at re=100,” *Journal of Fluids and Structures*, vol. 89, pp. 49–60, 2019.

[203] K. Sourav and S. Sen, “Transition of viv-only motion of a square cylinder to combined viv and galloping at low reynolds numbers,” *Ocean Engineering*, vol. 187, p. 106208, 2019.

[204] H. Zhu, C. Zhang, and W. Liu, “Wake-induced vibration of a circular cylinder at a low reynolds number of 100,” *Physics of Fluids*, vol. 31, no. 7, 2019.

[205] H. Zhu, H. Ping, R. Wang, Y. Bao, D. Zhou, and Z. Han, “Flow-induced vibration of a flexible triangular cable at low reynolds numbers,” *Physics of Fluids*, vol. 31, no. 5, 2019.

[206] H. Zhang, L. Zhou, P. Deng, and T. K. Tse, “Fluid–structure-coupled koopman mode analysis of free oscillating twin-cylinders,” *Physics of Fluids*, vol. 34, no. 9, 2022.

[207] K. Sourav, P. K. Yadav, P. Tallapragada, and D. Kumar, “Simultaneous streamwise and cross-stream oscillations of a diamond oscillator at low reynolds numbers,” *Physics of Fluids*, vol. 34, no. 6, 2022.

[208] D. Kumar and K. Sourav, “Vortex-induced vibrations of tandem diamond cylinders: A novel lock-in behavior,” *International Journal of Mechanical Sciences*, vol. 255, p. 108463, 2023.

[209] S. Sen and S. Mittal, “Free vibration of a square cylinder at low reynolds numbers,” *Journal of Fluids and Structures*, vol. 27, no. 5-6, pp. 875–884, 2011.

[210] K. Sourav, D. Kumar, and S. Sen, “Vortex-induced vibrations of an elliptic cylinder of low mass ratio: Identification of new response branches,” *Physics of Fluids*, vol. 32, no. 2, 2020.

[211] K. Taira, S. L. Brunton, S. T. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, “Modal analysis of fluid flows: An overview,” *Aiaa Journal*, vol. 55, no. 12, pp. 4013–4041, 2017.