ADAM UCHYTIL

FEEDBACK CONTROL OF MAGNETOHYDRODYNAMIC FLOW USING DATA-DRIVEN METHODS

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

FEEDBACK CONTROL OF MAGNETOHYDRODYNAMIC FLOW USING DATA-DRIVEN METHODS

ADAM UCHYTIL

Master's Thesis

Supervised by Ing. Jiří Zemánek, Ph.D.

May 2024

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Uchytil Adam** Personal ID number: **492389**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Feedback Control of Magnetohydrodynamic Flow Using Data-Driven Methods

Master’s thesis title in Czech:

Zpětnovazební řízení magnetohydrodynamického proudění pomocí metod založených na datech

Guidelines:

This project aims to design and implement a feedback control system for manipulating magnetohydrodynamic (MHD) flow induced by a set of electrodes and electromagnets. The MHD flow control aims to achieve specific objectives, e.g., controlling the flow to achieve desired velocity patterns, improving the mixing of solutions within the system, or transporting particles by the flow.
Tasks:
- Complete and document an environment for 3D fluid dynamic simulations incorporating magnetohydrodynamic forces.
- Compare simulation results obtained from mathematical models with experimental data collected on the physical platform.
- Investigate methods for reducing the dimensionality of the mathematical model.
- Develop and implement a data-driven flow control method. Consider incorporating online learning capabilities.
- Conduct experiments using the developed control system on the physical MHD platform. If necessary, validate the control algorithm using the mathematical model.

Bibliography / sources:

[1] M. A. Mendez, A. Ianiro, B. R. Noack, and S. L. Brunton, Eds., Data-Driven Fluid Mechanics: Combining First Principles and Machine Learning. Cambridge: Cambridge University Press, 2023.
[2] S. L. Brunton and J. N. Kutz, Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, 2nd ed. Cambridge: Cambridge University Press, 2022.
[3] H. Arbabi, M. Korda and I. Mezić, "A Data-Driven Koopman Model Predictive Control Framework for Nonlinear Partial Differential Equations," 2018 IEEE Conference on Decision and Control (CDC), Miami, FL, USA, pp. 6409-6414, 2018.
[4] Bieker, K., Peitz, S., Brunton, S.L. et al. Deep model predictive flow control with limited sensor data and online learning. Theor. Comput. Fluid Dyn. 34, 577–591, 2020.

Name and workplace of master's thesis supervisor:

Ing. Jiří Zemánek, Ph.D. Department of Control Engineering FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **16.02.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Jiří Zemánek, Ph.D. Supervisor's signature	prof. Ing. Michael Šebek, DrSc. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

DECLARATION

I hereby declare that this thesis is my original work, and it has been written by me in its entirety. I have duly acknowledged all the sources of information that have been used in the thesis. Furthermore, this thesis has not been submitted for any degree in any university previously.

Prague, May 2024

Adam Uchytil

ABSTRACT

This thesis primarily deals with developing a data-driven feedback control algorithm for shaping Magnetohydrodynamic (MHD) flow, which is the flow of electrically conducting fluid in a magnetic field. The control algorithm is validated on an experimental setup comprising a tank with a water-based electrolyte, electrodes and coils for actuation, and a system for real-time flow measurement using tracing particles and a camera. Initially, the thesis presents a simulation environment for the MHD fluid flow within the experimental setup. This environment solves the incompressible Navier–Stokes equations in three dimensions using the Finite Element Method, incorporating external body forces generated by the electrodes and coils. Next, the simulation environment is utilized to collect data for developing the control algorithm. The thesis explores two control scenarios. The first scenario is simplified, where the electrodes are fixed to a constant potential, and only the coils are used for actuation. Two control algorithms are developed for the simplified scenario: a neural network-based Model Predictive Control (MPC) algorithm and a Koopman operator-based MPC algorithm. The second scenario is more complex, where both the electrodes and coils are used for actuation. The control algorithm is based on the Koopman MPC algorithm from the first scenario but extended to include the electrodes as control inputs and solved using an alternating optimization approach.

Keywords: Data-driven Control, Koopman Operator, MPC, Magnetohydrodynamic Flow, Neural Networks

ABSTRAKT

Tato práce se primárně zabývá vývojem na datech založeného zpětnovazebního řídicího algoritmu pro tvarování magnetohydrodynamického (MHD) proudění, což je proudění elektricky vodivé kapaliny v magnetickém poli. K validaci řídicího algoritmu slouží experimentální platforma, která se sestává z nádrže s elektrolytem na bázi vody, elektrod, cívek a systému pro měření proudění v realném čase pomocí obrazu z kamery a trasovacích částic. V práci je nejprve prezentováno simulační prostředí pro proudění kapaliny v experimentální platformě. Toto prostředí řeší nestlačitelné Navier-Stokesovy rovnice ve třech dimenzích pomocí metody konečných prvků a zahrnuje síly působící na kapalinu generované elektrodami a cívkami. Následně je simulační prostředí využito k získání dat pro vývoj řídicího algoritmu. Práce zkoumá dva scénáře řízení. První scénář je zjednodušený, kde jsou na elektodách udržovány konstantní potenciály a proudění je tvarováno pouze pomocí cívek. Pro tento scénář jsou vyvinuty dva řídicí algoritmy: algoritmus prediktivního řízení (MPC) založený na neuronových síti a algoritmus MPC založený na aproximaci Koopmanova operátoru. Druhý scénář je složitější, jelikož jsou k tvarování proudění využívány jak elektrody, tak cívky. Řídicí algoritmus je založen na MPC pomocí aproximace Koopmanova operátoru z prvního scénáře, ale rozšířen o potenciály na elektrodách jako řídicí vstupy a řešen pomocí alternující optimalizace.

Klíčová slova: na datech založené řízení, Koopmanův operátor, MPC, magnetohydrodynamické proudění, neuronové sítě

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Ing. Jiří Zemánek, Ph.D. His crazy idea of controlling fluid flows with electric and magnetic fields was the foundation of this thesis. I would also like to extend my heartfelt thanks to all my colleagues at our lab, especially Loi, who patiently answered my numerous questions every Friday, likely delaying the completion of his own PhD thesis by at least a year. Special thanks to Krištof, whose offbeat jokes and unique sense of humor were a constant source of joy—or despair, depending on the day. Special thanks to Šimon Pecháček, an exceptionally skilled undergraduate student who built the experimental setup. His efforts were crucial to the completion of this thesis.

I am also grateful to those who provided valuable assistance: doc. RNDr. Ing. Pavel Řezanka, Ph.D at the University of Chemistry and Technology in Prague for supplying the dangerous acids necessary for my experiments, prof. Ing. Vladimír Havlena, CSc. for suggesting a better identification signal for the development of the predictors, Ing. Jan Haidl, Ph.D. from the Academy of Sciences of the Czech Republic for providing the particles for seeding the flow, and RNDr. Jaroslav Hron, Ph.D. from the Faculty of Mathematics and Physics at Charles University for his consultation regarding the methods for simulation of the MHD flow.

Finally, I would like to thank my parents for their unwavering support throughout my studies.

CONTENTS

1 Introduction 1
- 1.1 Outline 2
- 1.2 Background 2
- 1.3 State of the Art 2
- 1.4 Contributions 3

2 Task Description 5
- 2.1 Experimental Setup 5
- 2.2 Control Task 7
- 2.3 Mathematical Description of MHD Flow 9

3 Simulation Environment 11
- 3.1 Motivation 11
- 3.2 Problem Setup 11
- 3.3 Temporal Discretization & Spliting Scheme 12
- 3.4 Spatial Discretization 14
- 3.5 Handling of the Body Force Term 14
- 3.6 Solution Strategy 16
- 3.7 Implementation 18
- 3.8 Mesh Generation 18
- 3.9 Validation using Experimental Data 19
- 3.10 Validation using COMSOL Multiphysics® 24

4 Used Methods for Identification and Control 27
- 4.1 Motivation: Model Predictive Control 27
- 4.2 Delay Embeddings 28
- 4.3 Proper Orthogonal Decomposition 28
- 4.4 Koopman Operator-Based Approaches 29
- 4.5 Deep Learning-Based Approaches 33

5 Fluid Flow Shaping by Commanding Coils and Keeping Electrodes at Fixed Potentials 35
- 5.1 Setup 35
- 5.2 Data Generation 36
- 5.3 Deep Model Predictive Control 39
- 5.4 Koopman Operator-Based Control 42
- 5.5 Results 43

6 Fluid Flow Shaping by Commanding Both Coils and Electrodes 49
- 6.1 Setup 49
- 6.2 Data Generation 50
- 6.3 Koopman MPC 51
- 6.4 Alternating Minimization Scheme 53
- 6.5 Results 55

6.6 Notes on Online Learning 59
7 Experimental Validation of Designed Control Algorithm 61
7.1 Experiment 1: Two Vortices 62
7.2 Experiment 2: Diagonal Flow 63
7.3 Experiment 3: Multiple Directional Flow 63
7.4 Experiment 4: Striped Flow 64
7.5 Discussion 65
8 Conclusions & Outlook 67
8.1 Future Work 68

Appendix
A Dense Formulation of Tracking MPC 71
B Visualization of POD Modes 73

Bibliography 75

1

INTRODUCTION

Fluid dynamics plays a pivotal role in numerous industrial, environmental, and engineering applications. Despite its significance, real-time control of fluid flows still remains a challenging task due to the inherent complexity of fluid dynamics, which is characterized by nonlinearity, high dimensionality, and, in many cases, chaotic behavior. However, in recent years, the advent of *data-driven control* algorithms has opened up new possibilities for the control of fluid flows. By utilizing models derived from data rather than first-principle models, these algorithms have the potential to mitigate the computational costs associated with traditional control methods while providing robust solutions adaptable to evolving conditions.

I explore the design of data-driven control algorithms for a unique fluid flow phenomenon known as magnetohydrodynamic (MHD) flow. MHD flow arises in conducting fluids in the presence of magnetic fields and electric currents. The interaction between the electric currents and the magnetic fields results in a force on the fluid, which in turn causes the fluid to move. I specifically deal with a flow induced in a water-based electrolyte by applying potentials to a set of electrodes submerged in the liquid and by magnetic fields generated by a set of coils. I illustrate the principle of this MHD flow in Fig. 1.1.

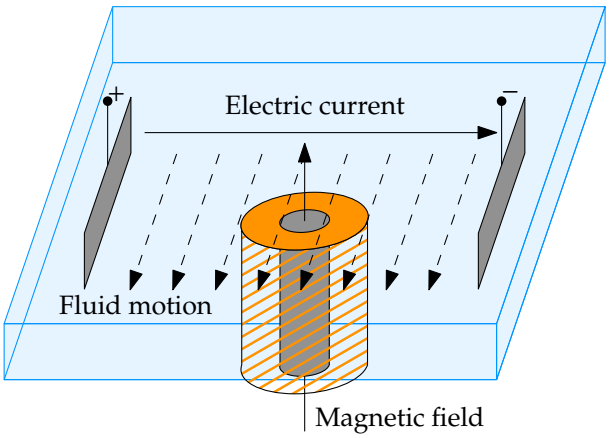


Figure 1.1: Illustration of the basic principle of MHD flow. Electrodes submerged in conducting fluid cause an electric current to flow between the electrodes. The current interacts with external magnetic field, resulting in a force on the fluid, which in turn causes the fluid to move.

GOALS

The primary goal of this thesis is to design a data-driven feedback control algorithm for the real-time shaping of MHD flows. By shaping the flow, it is meant driving the velocity field of the fuild in a designated region to some prescribed reference velocity field or achieving

close proximity to it. Subsequently, the developed control algorithm should be deployed and tested on an experimental setup to validate its efficacy.

1.1 OUTLINE

In Chapter 2, I describe the experimental setup used throughout the thesis, formally give the control objectives, and provide a brief mathematical description of the MHD flow in the setup. In Chapter 3, I present the simulation environment of the MHD flow in the experimental setup that I developed over the course of my thesis and the previous semester. In Chapter 4, I introduce the control and identification methods used in the thesis, including Model Predictive Control and surrogate models. In Chapter 5, I design control algorithm for shaping the flow by only varying the coil currents and fixing the electrodes to a constant potential. This gives an insight into the control of the MHD flow and serves as a basis for the control algorithm developed in Chapter 6, which shapes the flow by setting both the coil currents and the electrode potentials. In Chapter 7, I present the results of the experiments conducted to validate the control algorithms. Finally, in Chapter 8, I summarize the results and discuss further research directions.

1.2 BACKGROUND

For many years, the Advanced Algorithms for Control and Communications group at the Czech Technical University in Prague, which I am a part of, has pursued research in the development of control algorithms for distributed manipulation through the shaping of physical fields. Distributed manipulation involves the placement of actuators throughout a spatial domain, each exerting localized influence. These actuators collectively form a physical field, which exerts force on objects within the domain. The physical field is then shaped using control algorithms to achieve specific objectives, typically positioning objects at desired locations. This approach has been applied across various physical fields, including electric [Zemánek et al., 2018], magnetic [Zemánek, 2018], and acoustic pressure fields [Matouš et al., 2019]. Despite the diversity of fields, the primary objective remains consistent: guiding sets of objects to predetermined states. One intriguing extension of this problem involves essentially taking the limit of the number of controlled objects to infinity, thereby transitioning to continuum control. A typical continuum is a fluid, and one way to exert a force over a fluid is through the use of electric and magnetic fields, a phenomenon known as magnetohydrodynamic flow, which is the focus of this thesis.

1.3 STATE OF THE ART

Despite numerous applications of MHD flows, including fusion reactors, electromagnetic pumps, and liquid cooling systems, the work on feedback control of MHD flows has been rather limited. Authors Ren et al. [2018, 2019], Chen et al. [2020] have developed control algorithms for controlling the velocity in simple 1D channel flows. Other works like by Schuster et al. [2008], Vazquez et al. [2009] deal with mixing in 2D channel flows and

stabilization of 3D flows, respectively. These works have in common that they are based on first-principle models of the MHD flow rather than data-driven models, are limited to simple geometries, and do not experimentally validate the control algorithms.

However, recently, there has been a growing interest in data-driven control algorithms and frameworks for fluid flows. There are lot of works applying data-driven methods based on deep learning and Koopman operator theory to control fluid flows. To name a few, Arbabi et al. [2018] developed a data-driven framework for the control of nonlinear flows based on Koopman operator theory, similarly, Peitz and Klus [2019], Peitz et al. [2020], Peitz and Bieker [2023] developed frameworks for control of PDEs including fluid flows using Koopman operator theory, Morton et al. [2018] combined Koopman operator theory with deep learning to suppress vortex shedding in a cylinder wake, and Bieker et al. [2020] used deep learning based-models for control of mildly chaotic fluid flows.

1.4 CONTRIBUTIONS

The primary contribution of this thesis is demonstrating the concept of shaping an MHD flow using a set of electrodes and coils in a feedback manner with a model-free, data-driven control algorithm perfomed on a physical experimental setup. To the best of my knowledge, this has not been published before and could give rise to interesting applications. Therefore, I aim to publish the results in a scientific journal at a later date.

Secondary contribution is the development of a fast simulation environment for the MHD flow in the experimental setup, which is not necessary based on novel methods, but could be useful for future research.

2

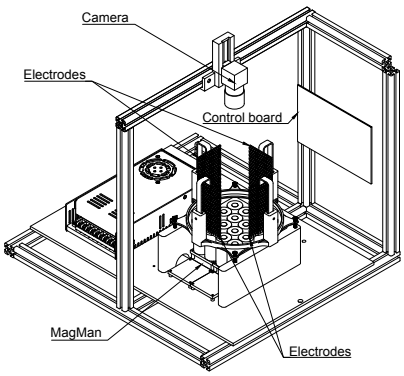
TASK DESCRIPTION

This chapter introduces and briefly describes the experimental setup used in the thesis to validate the control algorithms designed later in the thesis. The chapter further covers the formal definition of the control task and provides a mathematical description of the MHD flow within the setup.

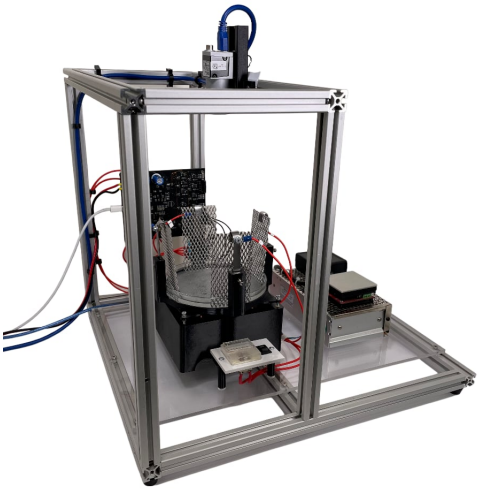
2.1 EXPERIMENTAL SETUP

The experimental setup comprises a small cylindrical tank (dish) filled with water-based electrolyte, with a set of electrodes submerged in the fluid and coils underneath the tank. To give a perspective, the tank has an inner diameter of 143 mm and a height of 16 mm. Overlooking the tank is a high-speed camera that captures the flow of the fluid seeded with small particles. The velocity field of the fluid is then obtained by processing the images captured by the camera. I present a sketch of the experimental setup in Fig. 2.3.

The setup was put together mainly by an undergraduate student, Šimon Pecháček. Therefore, for a more detailed description, I refer the reader to his thesis [Pechacek, 2024].



(a) Sketch of the experimental setup (reprint from [Pechacek, 2024]). Does not show the tank.



(b) Actual experimental setup.

Figure 2.1: Experimental setup.

2.1.1 *MagMan*

The coils are part of a system dubbed MagMan. MagMan is a platform that was previously used for distributed manipulation of steel balls via magnetic fields. The platform is now being repurposed for the control of MHD flows. MagMan is comprised of modules, each having four coils, necessary circuitry, and a communication interface. The communication interface allows for setting the current in the individual coils, both in magnitude and direction. There are four modules in total, each with a set of four coils, meaning that the platform has a total of sixteen coils. I show a single MagMan module as well as the complete platform in Fig. 2.2.

(a) Single module. (b) Complete platform.

Figure 2.2: MagMan platform (reprint from [Zemánek, 2018]).

2.1.2 *Electrode Array*

The electrodes used in the experimental platform are made of a titanium mesh with platinum coating. Currently, four electrodes are used, and they are placed in a square configuration at the border of the cylindrical tank. Each of the electrodes can be set to a specific voltage within the range of 0 V to 10 V using a custom-built power supply. I show a shot of the electrode array in Fig. 2.3.

Figure 2.3: Shot of the electrode array with the electrodes submerged in the fluid. Fluid is also seeded with particles for measurement of the velocity field.

2.1.3 *Particle Image Velocimetry*

A crucial part of the experimental setup is the measurement of the velocity field of the fluid. This is done using a technique known as Particle Image Velocimetry (PIV). The method involves seeding the surface of the fluid with small particles and capturing their flow with a camera. The velocity field is then inferred from the displacement of the particles between subsequent images. Only the surface flow at a rectangular measurement region of the tank is captured. I present a sketch of the PIV pipeline as well as the measured region in Fig. 2.4.

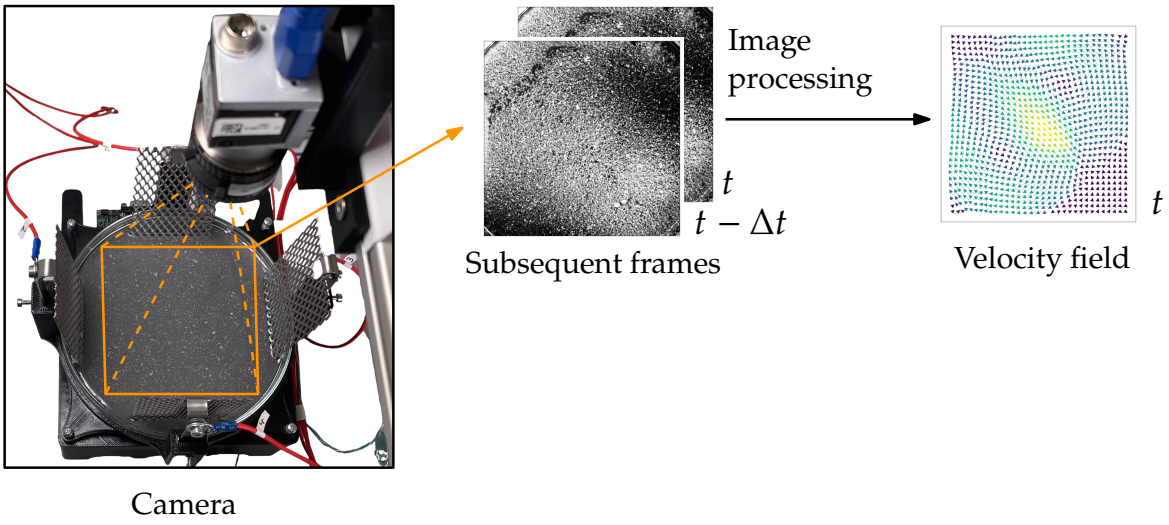


Figure 2.4: PIV pipeline. The fluid is seeded with particles. The camera captures images of a 10 cm × 10 cm region. The images are then processed to obtain the velocity field.

2.1.4 *Electrolyte*

The fluid used in the experimental setup is a water-based electrolyte. Two electrolytes were considered for the experiments: 0.6 % solution of sulfuric acid in water and 0.05 % solution of sodium sulfate in water. At these concentrations, the fluid conductivity is about $5\,\mathrm{S\,m^{-1}}$ and $4\,\mathrm{S\,m^{-1}}$, respectively.

2.2 CONTROL TASK

Having described the experimental setup, I now outline the control task. The goal is to design a feedback control algorithm that sets the potentials on the electrodes and the currents in the coils to shape the velocity field of the fluid in a designated region to a prescribed reference velocity field. However, the prescribed reference velocity field may not always be physically realizable, making this an ill-posed problem. Therefore, the aim is to achieve close proximity to the reference velocity field instead. Furthermore, I restrict myself to the two-dimensional

surface flow, as it is the only part of the flow that is measured by the PIV system. I show the high-level control loop diagram in Fig. 2.5.

This could have several practical applications. E.g., the flow could be manipulated to mix the fluid in a tank, use velocity profiles for optimal heat transfer, or transport particles in a specific manner.

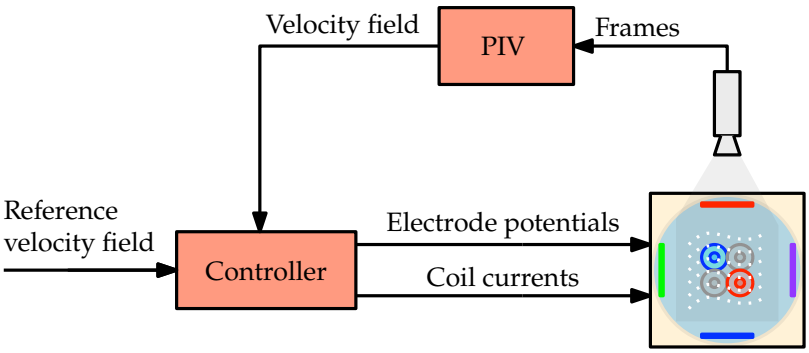


Figure 2.5: High level overview of the feedback loop.

2.2.1 *Formal Definition*

The stated objective can be formalized by minimizing the error between the actual velocity field and the reference velocity field. I propose using the square of the well-known L^2 norm of the difference between the velocity field and the reference velocity field as the error

$$e(t) = \int_{\mathcal{R}} (v_x(x,y,t) - r_x(x,y,t))^2 + (v_y(x,y,t) - r_y(x,y,t))^2 \,\mathrm{d}\eta(x,y), \tag{2.1}$$

where v_x and v_y are the x and y components of the velocity field, r_x and r_y are the corresponding components of the reference velocity field, t is time, $\mathcal{R}$ is the region of interest, and η is a measure on the region $\mathcal{R}$. The reason for using a general measure η is that it accommodates both cases of $\mathcal{R}$ being a discrete set of points or possibly collection of connected subregions. For the former, η would be the sum of Dirac measures at the appropriate points, and for the latter, η would be the Lebesgue measure.

The error evolves over time, so the actual control task is finding such electrode potentials and coil currents that minimize the *total error* over some time horizon T

$$\int_0^T e(t)\,\mathrm{d}t \to \min. \tag{2.2}$$

However, due to the discrete nature of the control algorithm and the velocity measurements, the task is not directly tractable. Therefore, I propose an approximation as typically done in practice. First, let me define the grid on which the PIV measurements are taken

$$\mathcal{M} = \{x_1, x_2, \ldots, x_{N_x}\} \times \{y_1, y_2, \ldots, y_{N_y}\}, \tag{2.3}$$

where N_x and N_y are the number of grid points in the x and y directions, respectively. Now, let me define the measure η as the sum of Dirac measures at the grid points:

$$\eta = \sum_{(x,y)\in\mathcal{M}} \delta_{(x,y)}. \tag{2.4}$$

The error w.r.t to this measure will essentially be the error computed at the grid points within the region $\mathcal{R}$, i. e., the set $\mathcal{M} \cap \mathcal{R}$. The error, therefore is

$$e(t) = \sum_{(x,y)\in\mathcal{M}\cap\mathcal{R}} \left[(v_x(x,y,t) - r_x(x,y,t))^2 + (v_y(x,y,t) - r_y(x,y,t))^2\right]. \tag{2.5}$$

This is a quantity that can be computed from the PIV measurements. Furthermore, this error is discretized in time

$$e_k = e(t_k), \quad k = 1, 2, \ldots, N_T, \tag{2.6}$$

where t_k are the discrete time steps at which the control algorithm runs, and N_T is the number of time steps in the time horizon. The task, therefore becomes

$$\sum_{k=1}^{N_T} e_k \to \min. \tag{2.7}$$

This is a cost that can be directly optimized by the control algorithm, and thus, the problem is tractable in practice, at least in the sense of local optimization.

2.2.2 *Choice of the region $\mathcal{R}$*

Now is a good time to elaborate on the choice of the region $\mathcal{R}$. Prescribing the reference velocity field over the entire measured surface and thus minimizing the error across the whole surface can lead to undesirable behavior. E. g., consider a scenario where the reference velocity field comprises of a vortex at the center of the surface while the velocity field is zero everywhere else. The prevalence of the zero velocity field in the reference would likely prevent the vortex from forming, as that actually might be the optimum of the presented error. Therefore, I propose prescribing the velocity only within a specific region of interest, $\mathcal{R}$, and allowing the controller to determine the velocity for the rest of the surface. In the case of the example, the reference velocity field would be prescribed only within the region where the vortex is desired, and the controller would determine the velocity field elsewhere.

2.3 MATHEMATICAL DESCRIPTION OF MHD FLOW

The platform is subject to a MHD flow known as inductionless MHD flow. Inductionless MHD flow, as the name suggests, describes the flow of a conducting fluid where the magnetic field induced by the current in the fluid is negligible compared to the external magnetic field. The determination of whether MHD flow can be considered inductionless hinges on a

dimensionless quantity dubbed the magnetic Reynolds number, denoted as $\mathrm{R_m}$, and defined as

$$\mathrm{R_m} = \mu\sigma V L. \tag{2.8}$$

Here, μ is the magnetic permeability of the fluid, σ is the fluid conductivity, V is the characteristic fluid velocity, and L is the characteristic length of the system.

For the experimental setup—employing a water-based electrolyte—the relative magnetic permeability (μ_r) of the fluid corresponds to that of water, i. e., $\mu_r = 1$, giving $\mu = \mu_0\mu_r \sim 1\,\mu\mathrm{H\,m^{-1}}$. Additionally, the fluid conductivity was measured to be $\sigma \sim 1\,\mathrm{S\,m^{-1}}$, the observed velocity in our setup is $V \sim 1\,\mathrm{cm\,s^{-1}}$, while the characteristic length of the system is $L \sim 10\,\mathrm{cm}$. Substituting theses values, the magnetic Reynolds number for the system is $\mathrm{R_m} \sim 10^{-9}$, which is significantly less than unity. This result justifies the assertion that the flow can indeed be regarded as inductionless.

2.3.1 *Equations of Magnetohydrodynamics*

Furthermore, I assume the fluid is incompressible. This together with the inductionless assumption, implies the physics phenomena can be described by the incompressible Navier–Stokes equations with an external body force term—the Lorentz force. These Navier–Stokes equations read

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v} - \nu\nabla^2\mathbf{v} + \nabla p = \frac{1}{\rho}\mathbf{F}, \tag{2.9a}$$

$$\nabla\cdot\mathbf{v} = 0, \tag{2.9b}$$

where $\mathbf{v}$ is the velocity field, p is the pressure, ν is the kinematic viscosity, ρ is the density, and $\mathbf{F}$ is the Lorentz force. The Eq. (2.9a) is called the momentum equation, and the Eq. (2.9b) is called the continuity equation. Further, the Lorentz force is given by the cross product of the current density $\mathbf{J}$ and the external magnetic field $\mathbf{B}$, as per the Lorentz force law

$$\mathbf{F} = \mathbf{J}\times\mathbf{B}. \tag{2.10}$$

The current density in the fluid is then described by Ohm's law

$$\mathbf{J} = \sigma(\mathbf{E} + \mathbf{v}\times\mathbf{B}), \tag{2.11}$$

where σ is the conductivity, and $\mathbf{E}$ is the electric field in the fluid. Expanding the Lorentz force using the Ohm's law gives

$$\mathbf{F} = \sigma\mathbf{E}\times\mathbf{B} + \sigma\,(\mathbf{v}\times\mathbf{B})\times\mathbf{B}. \tag{2.12}$$

The magnitude of the second term in the above equation can be bounded by

$$\sigma\|(\mathbf{v}\times\mathbf{B})\times\mathbf{B}\|_2^2 \le \sigma\|\mathbf{v}\|_2\|\mathbf{B}\|_2^2, \tag{2.13}$$

From the experimental measurements, I know that $\|\mathbf{B}\|_2 \sim 10\,\mathrm{mT}$ and $\|\mathbf{v}\|_2 \sim 1\,\mathrm{cm\,s^{-1}}$, making the second term to be $\sim 1\,\mu\mathrm{N\,m^{-3}}$, which is negligible compared to the first term, which is $\sim 1\,\mathrm{mN\,m^{-3}}$. Therefore, it can be readily omitted, and the Lorentz force can be approximated as

$$\mathbf{F} = \sigma\mathbf{E}\times\mathbf{B}. \tag{2.14}$$

3

SIMULATION ENVIRONMENT

In this chapter, I present the simulation environment I developed to efficiently simulate the MHD flow in the experimental setup. The simulation environment is designed to solve the incompressible Navier-Stokes equations in the presence of the body force generated by the electric and magnetic fields. The development of the simulation environment stretched over several months, beginning in the last winter semester. Therefore, this chapter describes the work not only from the period of the thesis but also from the previous semester.

3.1 MOTIVATION

For the purposes of developing a feedback control methodology for the MHD flow, I required a fluid flow simulation framework with few specific capabilities. First and foremost, the simulation environment should be capable of operating within a closed loop, i. e., allowing me to evaluate the current solution at each timestep and, based on that evaluation, adjust the body forces for the next timestep. The second is the ability to run fast, preferably close to real-time, to facilitate the high data requirements of data-driven control methods. Unfortunately, neither the free Computational Fluid Dynamics (CFD) software I encountered nor the commercial software available to me were completely suitable for this task. As an example, popular open-source CFD software is OpenFOAM, which, by default, does not support arbitrary body forces and coupling with external software. On the other hand, while COMSOL Multiphysics® offers a closed-loop capability through its LiveLink™ for Simulink add-on, it only supports scalar inputs and outputs, and the simulation time significantly increases when operating in this mode. As a result, I opted to develop my own simulation environment. Moreover, as the programming language, I selected Julia because it is highly performant, while being easy to use.

3.2 PROBLEM SETUP

The simulation environment is designed to solve the incompressible Navier–Stokes equations in the following form

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\nabla p + \nu \nabla^2 \mathbf{v} + \frac{1}{\rho}\mathbf{F} \text{ in } \Omega, \tag{3.1a}$$

$$\nabla \cdot \mathbf{v} = 0 \text{ in } \Omega, \tag{3.1b}$$

where Ω is a given 3D domain, $\mathbf{v}$ is the velocity field, p is the pressure field, ν is the kinematic viscosity, ρ is the fluid density, and $\mathbf{F}$ is the body force. The body force is the force exerted by the electric and magnetic fields

$$\mathbf{F} = \sigma \mathbf{E} \times \mathbf{B}, \tag{3.2}$$

where σ is the conductivity of the liquid, $\mathbf{E}$ is the electric field, and $\mathbf{B}$ is the magnetic field. Moreover, I also consider a Dirichlet boundary condition for the velocity field

$$\mathbf{v} = \mathbf{g} \text{ on } \partial\Omega, \tag{3.3}$$

where $\mathbf{g}$ comprises the following

$$\mathbf{v} = \mathbf{0} \text{ on } \Gamma_{\text{walls}}, \tag{3.4}$$

i. e., the velocity is zero at the walls, and zero flux of the velocity through the surface of the domain

$$\mathbf{v} \cdot \mathbf{n} = \mathbf{0} \text{ on } \Gamma_{\text{surface}}. \tag{3.5}$$

The sets Γ_{walls} and Γ_{surface} represent the walls and the surface of the domain, with $\mathbf{n}$ being the normal vector to the surface, and $\partial\Omega = \Gamma_{\text{walls}} \cup \Gamma_{\text{surface}}$.

As pressure only appears in the Navier–Stokes equations in the form of its gradient and no outflow boundary conditions are imposed, it is determined up to a constant. Therefore, I need to impose a condition to fix this constant. I choose the condition that the average pressure over the domain is zero, i. e.,

$$\int_\Omega p \, \mathrm{d}\Omega = 0. \tag{3.6}$$

3.3 TEMPORAL DISCRETIZATION & SPLITING SCHEME

I handle the temporal discretization of the problem by utilizing the Backward Difference Formula of the second order (BDF2). BDF2 is a well-established multi-step implicit method known for its unconditional stability and second-order accuracy, making it a popular choice in the field of CFD. I apply the BFD2 in conjunction with the Incremental Pressure Correction Scheme (IPCS) in the rotational form, as shown in the work by Guermond et al. [2006]. IPCS weakly decouples the velocity and pressure computations, therefore breaking down the computation of each timestep into more manageable and efficiently computable sub-problems. This decoupling strategy is crucial, as solving the fully coupled incompressible Navier-Stokes equations results in so-called saddle point problem, which is notoriously difficult to solve efficiently.

IPCS is a member of the family of projection methods. Generally, these methods exploit the fact that in incompressible isothermal flows, the pressure is not a state quantity, but rather a Lagrange multiplier enforcing the continuity equation. One timestep of the projection methods usually consists of the following three steps:

1. Solve the momentum equation Eq. (3.1a) with some a priori guess for the pressure. Ignore the continuity equation Eq. (3.1b).

2. Compute a posterior pressure field that would make the velocity field from the previous step incompressible.

3. Using the posterior pressure field project the velocity field found by the first step to make it divergence-free.

However, the final velocity is just a projection. It does not satisfy the boundary conditions. From the point of error analysis, both the velocity from the first and last step are equally good approximations of the true solution. Thus, there is no need to prefer the velocity from the last step over the first one as the output of the simulation. This incentive is further supported by the fact that in the IPCS algorithm, the final step can be easily eliminated, effectively reducing the scheme to two steps. This is the approach I take in my simulation environment. Thus, the scheme consists of the following two steps.

3.3.1 *Velocity Step*

The first step is so-called *Velocity Step*, where the current value of the velocity field $\mathbf{v}_k$, i. e., the velocity field at the time $t = k\Delta t$, is computed. The velocity is computed from the knowledge of the previous two values of the velocity field $\mathbf{v}_{k-1}, \mathbf{v}_{k-2}$, the pressure $p_{k-1}, p_{k-2}, p_{k-3}$, and the current value of the body force $\mathbf{F}_k$. The step is realized by solving the following PDE problem

$$\frac{3\mathbf{v}_k - 4\mathbf{v}_{k-1} + \mathbf{v}_{k-2}}{2\Delta t} + [(2\mathbf{v}_{k-1} - \mathbf{v}_{k-2}) \cdot \nabla]\mathbf{v}_k - \nu\nabla^2\mathbf{v}_k + \frac{1}{3}\nabla\left(7p_{k-1} - 3p_{k-2} + p_{k-3}\right) = \frac{1}{\rho}\mathbf{F}_k \text{ in } \Omega, \tag{3.7a}$$

$$\mathbf{v}_k = \mathbf{g} \text{ on } \partial\Omega, \tag{3.7b}$$

which arises from the discretization of the momentum equation using the BDF2 method. To linearize the problem, I approximate the convection term $(\mathbf{v} \cdot \nabla)\mathbf{v}$ using extrapolated velocity

$$(\mathbf{v}_k \cdot \nabla)\mathbf{v}_k \approx [(2\mathbf{v}_{k-1} - \mathbf{v}_{k-2}) \cdot \nabla]\mathbf{v}_k.$$

3.3.2 *Pressure Step*

The second and the last step is the computation of the pressure p_k at the current timestep k. This is achieved by solving the Pressure Poisson Problem

$$-\nabla^2\left(\tilde{p}_k - p_{k-1}\right) = -\frac{3}{2\Delta t}\nabla \cdot \mathbf{v}_k \text{ in } \Omega, \tag{3.8a}$$

$$\frac{\partial\left(\tilde{p}_k - p_{k-1}\right)}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega, \tag{3.8b}$$

where $\mathbf{n}$ is the normal vector to the boundary of the domain, and $\tilde{p}_k$ is the tentative pressure field, to which the rotational projection

$$p_k = \tilde{p}_k - \nu\nabla \cdot \mathbf{v}_k, \tag{3.9}$$

is applied to obtain the final pressure field p_k.

3.4 SPATIAL DISCRETIZATION

I approach the spatial discretization of the problem using the conventional Galerkin Finite Element Method (FEM). This decision primarily stems from the prevalence of FEM packages within the Julia ecosystem, contrasting with the availability of Finite Volume Method and Finite Differences Method alternatives.

FEM also has the advantage of being easily adaptable to complex geometries. Generally, It can also handle arbitrary meshes on these geometries, but I prefer using structured meshes with hexahedral elements, as they have fewer degrees of freedom and, therefore, require fewer computational resources.

I utilize quadratic polynomials to approximate the velocity space and linear polynomials for the pressure space, forming what is commonly referred to as Taylor–Hood elements or Q2-Q1 discretization. This selection is deliberate, as it adheres to the well-known inf-sup condition, ensuring numerical stability. However, it should be noted that when the flow becomes advection-dominated at high Reynolds numbers, the standard Galerkin method with Taylor–Hood elements can still suffer spurious oscillations in the solution. In such cases, stabilization techniques like the Streamline Upwind Petrov–Galerkin, Galerkin Least-Squares, or Discontinuous Galerkin methods can be employed.

The spatial discretization leads to the differential operators being approximated by matrices. I do not provide the explicit form of these matrices for the brevity of the exposition, as their construction is well-known in the literature. If the reader is interested in the details, I recommend consulting the work by Elman et al. [2005] or the documentation of different FEM packages, such as the Julia package Ferrite.jl[1].

3.5 HANDLING OF THE BODY FORCE TERM

The body force generated by the interaction of the electric field in the fluid and the external magnetic field is a crucial part of the simulation. However, it is computationally expensive to simulate the electric and magnetic fields together with the fluid flow. Fortunately, the fields can be precomputed and stored in a file, as they are not dependent on the velocity and pressure fields.

I approach the generation of the body force by constructing a basis of simulated electric and magnetic fields. These simulated fields are then used to compute the total fields in the domain by linearly combining the basis with the corresponding electrode and coil commands. Specifically, assuming n_{el} electrodes, the electric field is expressed as

$$\mathbf{E} = \sum_{i=1}^{n_{\text{el}}} \phi_i \mathbf{E}_i, \tag{3.10}$$

1 https://ferrite-fem.github.io/Ferrite.jl/stable/

where ϕ_i denotes the command for ith electrode, basically a voltage scaling factor, and $\mathbf{E}_i$ is the field generated by the ith electrode at 1 V while all the other electrodes grounded. Similiary, assuming n_{coils} coils, the magnetic field is expressed as

$$\mathbf{B} = \sum_{i=1}^{n_{\text{coils}}} \psi_i \mathbf{B}_i,$$

where ψ_i is the command for ith coil, acting as current scaling factor, and $\mathbf{B}_i$ is the field generated by the ith coil when subjected to a reference current of 440 mA, with all other coils turned off. It is noteworthy that due to the nonlinear BH curve of the ferromagnetic material composing the coil cores, the magnetic fields of individual cores are not superimposable. However, the nonlinearity can be partially alleviated by considering

$$\psi = f(i), \tag{3.11}$$

where i is the coil current, and $f(i)$ is the nonlinear scaling function, which was experimentaly determined by Zemánek [2018] for MagMan platform as:

$$f(i) = 0.889 \operatorname{atan}\left(2.5\,|i| + 5.38 i^2\right) \operatorname{sign}(i). \tag{3.12}$$

Furthermore, the cross interaction of the coils can be neglected, as the coils are placed far enough from each other, and the magnetic field decays rapidly with distance.

The total body force acting on the fluid is then given as:

$$\mathbf{F} = \sigma \sum_{i=1}^{n_{\text{el}}} \sum_{j=1}^{n_{\text{coils}}} \phi_i \psi_j \mathbf{E}_i \times \mathbf{B}_j. \tag{3.13}$$

3.5.1 *Implementation*

I utilize COMSOL Multiphysics® to conduct simulations of the electromagnetic field, employing a high-fidelity model that encompasses the entire platform. However, COMSOL Multiphysics® presents a challenge when it comes to exporting solutions, as it only allows for pointwise evaluations of the fields.

In my experience, simply evaluating the fields on a uniform grid embedded in the domain and then linearly interpolating them on the grid for arbitrary point evaluation does not yield accurate results. To overcome this limitation, I export the fields evaluated in the nodes of the COMSOL mesh. This approach provides the most accurate representation of the solution. However, I conduct the fluid simulations on a different (arbitrary) mesh, which may not align with the COMSOL mesh. Consequently, I need to interpolate the fields from the COMSOL mesh to the fluid mesh, specifically from the nodes of the COMSOL mesh to the quadrature points of the fluid mesh. To achieve this, I employ the Euclidean distance-based Nearest Neighbor interpolation, which I illustrate in Fig. 3.1.

Algorithmically, the Nearest Neighbor search is done by constructing a KD tree from the COMSOL nodes and then querying the tree for the nearest node to the quadrature point. The value of the field at the nearest node is then assigned to the quadrature point.

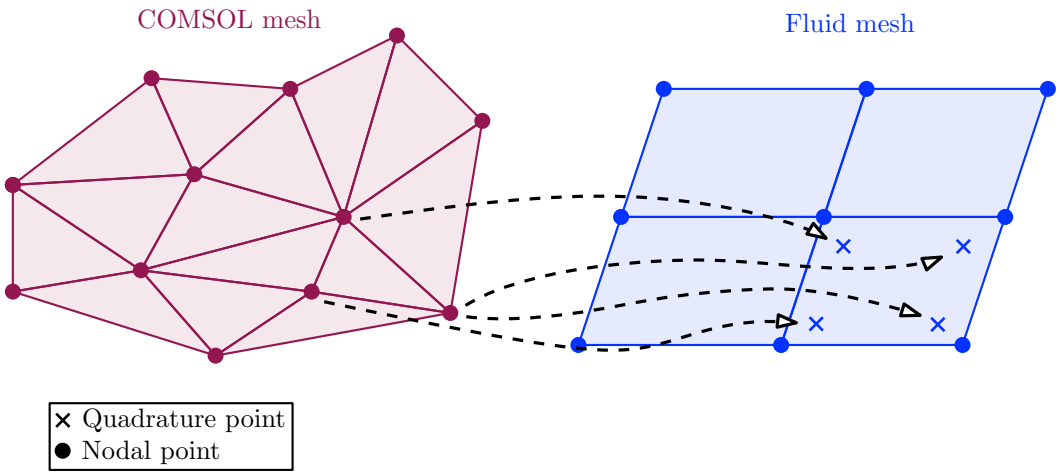


Figure 3.1: Illustration of the Nearest Neighbor interpolation of the fields generated by COMSOL Multiphysics® to the quadrature points of the fluid mesh. The two meshes spatially represent the same domain.

3.6 SOLUTION STRATEGY

In the following section, I outline the solution strategy employed to efficiently solve the arising linear sub-problems obtained from the temporospatial discretization of the simulation timestep. Notation-wise, symbols $\mathbf{v}, \mathbf{p}$ and $\mathbf{F}$ from this section onwards do not denote the approximated functions, but rather the corresponding vectors of values of the Degrees of Freedom (DOFs) from the FEM discretization, i. e., $\mathbf{v}, \mathbf{F} \in \mathbb{R}^{\mathrm{DOF}_v}$ and $\mathbf{p} \in \mathbb{R}^{\mathrm{DOF}_p}$, where DOF_v and DOF_p are the number of DOFs in the velocity and pressure space, respectively.

3.6.1 *Velocity Step*

The discrete form of the Velocity Step is

$$\frac{1}{2\Delta t}\mathbf{M}\left(3\mathbf{v}_k - 4\mathbf{v}_{k-1} + \mathbf{v}_{k-2}\right) + \mathbf{C}(\mathbf{v}_{k-1}, \mathbf{v}_{k-2})\mathbf{v}^k + \nu\mathbf{K}\mathbf{v}_k - \frac{1}{3}\mathbf{G}\left(7\mathbf{p}_{k-1} - 3\mathbf{p}_{k-2} + \mathbf{p}_{k-3}\right) = \frac{1}{\rho}\mathbf{F}_k + \text{b.c. terms}, \tag{3.14}$$

where $\mathbf{M}$ is the (velocity) mass matrix, $\mathbf{K}$ is the stiffness matrix, $\mathbf{C}(\mathbf{v}_{k-1}, \mathbf{v}_{k-2})$ is the convection matrix, $\mathbf{G}$ is the gradient matrix. These matrices arise from discretization of their corresponding differential operators using the FEM. The b.c. terms are the terms arising from the prescribed boundary conditions. The step can be rewritten as a system of linear equations

$$\mathbf{A}\mathbf{v}_k = \mathbf{b}, \tag{3.15}$$

where

$$\mathbf{A} = \frac{3}{2\Delta t}\mathbf{M} + \nu\mathbf{K} + \mathbf{C}(\mathbf{v}_{k-1}, \mathbf{v}_{k-2}), \tag{3.16}$$

$$\mathbf{b} = \frac{1}{\rho}\mathbf{F}_k + \frac{1}{3}\mathbf{G}\left(7\mathbf{p}_{k-1} - 3\mathbf{p}_{k-2} + \mathbf{p}_{k-3}\right) + \frac{1}{2\Delta t}\mathbf{M}\left(4\mathbf{v}_{k-2} - \mathbf{v}_{k-3}\right) + \text{b.c. terms}. \tag{3.17}$$

I solve Eq. (3.15) by standard restarted GMRES method with Incomplete LU preconditioner.

3.6.2 *Pressure Step*

Next step, is the solution of the system arising from the Pressure Poisson Equation

$$\mathbf{R}\tilde{\mathbf{p}}_k = \frac{3}{2\Delta t}\mathbf{D}\mathbf{v}_k + \mathbf{R}\mathbf{p}_{k-1} + \text{b.c. terms}. \tag{3.18}$$

Note that for matrix $\mathbf{D}$—which represents the discretization of the divergence operator—it holds that $\mathbf{D} = -\mathbf{G}^\mathsf{T}$. Additionally, matrix $\mathbf{R}$, resulting from the discretization of the Laplace operator, remains constant throughout the simulation runtime. Therefore, I precompute its LU factorization. This allows me to just employ forward and backward substitution during the actual simulation to solve the system efficiently.

Lastly, the rotational projection is applied,

$$\mathbf{p}_k = \tilde{\mathbf{p}}_k - \nu\mathbf{D}\mathbf{v}_k. \tag{3.19}$$

3.6.3 *Pressure Mean Value Constraint*

Additionally, I should elaborate on how the constraint for the zero mean value of the pressure is enforced. The mean value of the pressure is computed by the following quadrature

$$\int_\Omega p \, \mathrm{d}\Omega \approx \sum_{i=1}^{\mathrm{DOF}_p} w_i p_i, \tag{3.20}$$

where p_i are the individual DOFs of the pressure field, and w_i are the corresponding quadrature weights. This implies the mean value of the pressure is zero if and only if

$$\sum_{i=1}^{\mathrm{DOF}_p} w_i p_i = 0. \tag{3.21}$$

Furthermore, the previous sum can be used to express the pressure DOF p_j as a linear combination of the other pressure DOFs p_i, i. e.,

$$p_j = -\sum_{\substack{i=1 \\ i\neq j}}^{\mathrm{DOF}_p} \frac{w_i}{w_j} p_i. \tag{3.22}$$

Thus, I can eliminate the pressure DOF p_j from my computations. The question remains, which pressure DOF to eliminate? I need to make sure that w_j is nonzero. Therefore, I choose the pressure DOF p_j with the largest (absolute) quadrature weight w_j

$$j = \underset{i=1,\ldots,\mathrm{DOF}_p}{\operatorname{argmax}} |w_i| . \tag{3.23}$$

3.7 IMPLEMENTATION

I implemented the simulation environment in Julia using the FEM toolbox *Ferrite.jl*[2]. Ferrite is a low-level FEM toolbox, facilitating the user with tools for computing quadrature rules, FEM interpolations, evaluating shape functions and their derivatives in the finite element space, enforcing boundary conditions, etc.

Where possible, the assembly of the FEM matrices and vectors is done in a parallelized manner, utilizing the Julia's native threading capabilities (*Base.Threads*). This significantly speeds up the computation, especially for large meshes. The GMRES solver is provided by the *Krylov.jl*[3] package and the Incomplete LU preconditioner by the *IncompleteLU.jl*[4] package. The direct LU factorization solver is provided by the *SparseArrays.jl*[5], which is interfacing the *SuiteSparse*[6] library.

The simulation environment itself is constructed as a Julia package and is available on CTU FEE Gitlab [7] including the example of its usage.

3.8 MESH GENERATION

I generate the mesh using the open-source mesh generator *Gmsh*[8], specifically its Julia interface *Gmsh.jl*[9]. The mesh is a cylinder, corresponding to the dish's interior in the experimental setup, with a diameter of 143 mm and a height based on the fluid level in the experimental setup. The mesh is structured, consisting of hexahedral elements. I refine the mesh near the walls and the surface of the domain to capture the boundary layers accurately. I show the generated mesh in Fig. 3.2.

2 https://github.com/Ferrite-FEM/Ferrite.jl
3 https://github.com/JuliaSmoothOptimizers/Krylov.jl
4 https://github.com/haampie/IncompleteLU.jl
5 https://github.com/JuliaSparse/SparseArrays.jl
6 https://github.com/DrTimothyAldenDavis/SuiteSparse
7 https://gitlab.fel.cvut.cz/aa4cc/mhd/MHDSim.jl
8 https://gmsh.info/
9 https://github.com/JuliaFEM/Gmsh.jl

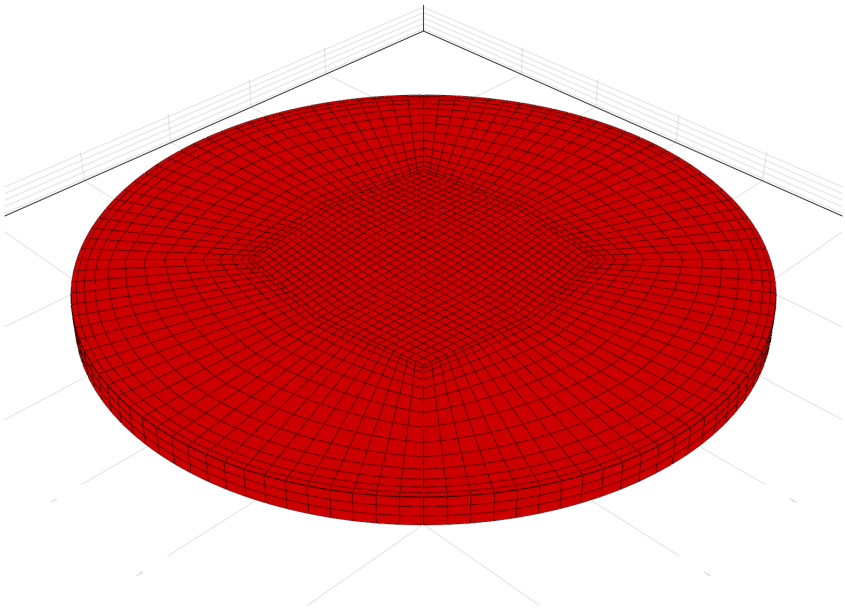


Figure 3.2: Used mesh.

3.9 VALIDATION USING EXPERIMENTAL DATA

In the following section, I compare the outputs from the simulation environment with the experimental data obtained from our experimental setup. The experimental data consist of surface velocity fields on a 32×32 grid of points, corresponding to the $10\,\mathrm{cm} \times 10\,\mathrm{cm}$ surface area of the domain. These velocity fields were obtained using PIV with 85 µm polymethyl methacrylate (PMMA) particles dispersed over the surface of the fluid.

I present the results of three different scenarios, each with a different flow configuration. The first scenario is a simple one-directional flow, the second scenario is a flow induced by multiple coils, and the third scenario showcases an oscillatory behavior. All three experiments were conducted with the same electrolyte, a 0.5 % solution of sulfuric acid in water with a conductivity of $5\,\mathrm{S\,m^{-1}}$, and the fluid height of 8 mm. The electric field was generated by two electrodes, which were placed on opposite sides of the domain. The magnetic field was generated by four coils below the center of the domain. I present the coil and electrode configurations for each experiment in Fig. 3.4, Fig. 3.6, and Fig. 3.8, respectively. The coils were excited with a current of ±440 mA, and the electrodes were kept at the potentials of 10 V and 0 V, respectively.

Scenario 1: Simple One Directional Flow

In this scenario, a solitary coil is used to actuate the flow, resulting in a mostly straightforward unidirectional flow pattern flow pattern with slight vortices created by the interaction of the flow with the domain walls. I show both the experimental and simulated velocity fields at different time instances in Fig. 3.3.

Scenario 2: Flow Induced by Multiple Coils

In this scenario, the flow is induced by exciting all four coils simultaneously with maximum current in the same direction. This results in a more intricate flow pattern characterized by prominent lateral vortices alongside a streamlined flow in the center. The comparison between the experimental and simulated velocity fields is illustrated in Fig. 3.5.

Scenario 3: Oscillations

The final scenario aims to highlight oscillatory flow patterns. The flow is induced by activating two diagonally adjacent coils using opposing currents. Once more, the comparison between the experimental and simulated velocity fields is presented in Fig. 3.7. The fluid patterns observed in the experiment recur periodically.

Discussion

The simulation environment effectively captures the general structure of the flow patterns, meaning their shape and direction. However, it tends to overestimate velocity magnitudes by up to 30 %, leading to slightly accelerated dynamics. Identifying the precise source of this deviation is challenging due to various factors affecting the experimental setup's accuracy. These include slight unevenness in the setup's level, the dish's imperfect cylindrical shape, and the fluid's initial state not being entirely still. Additionally, the PIV technique itself can influence the flow, as the PMMA particles sometimes form a crust on the fluid's surface, slowing down the flow. Moreover, the simulation involves simplifications such as disregarding the temperature dependence of fluid properties, the non-superimposability of coil magnetic fields, surface tension effects, and the absence of mesh electrodes within the domain or other possible factors.

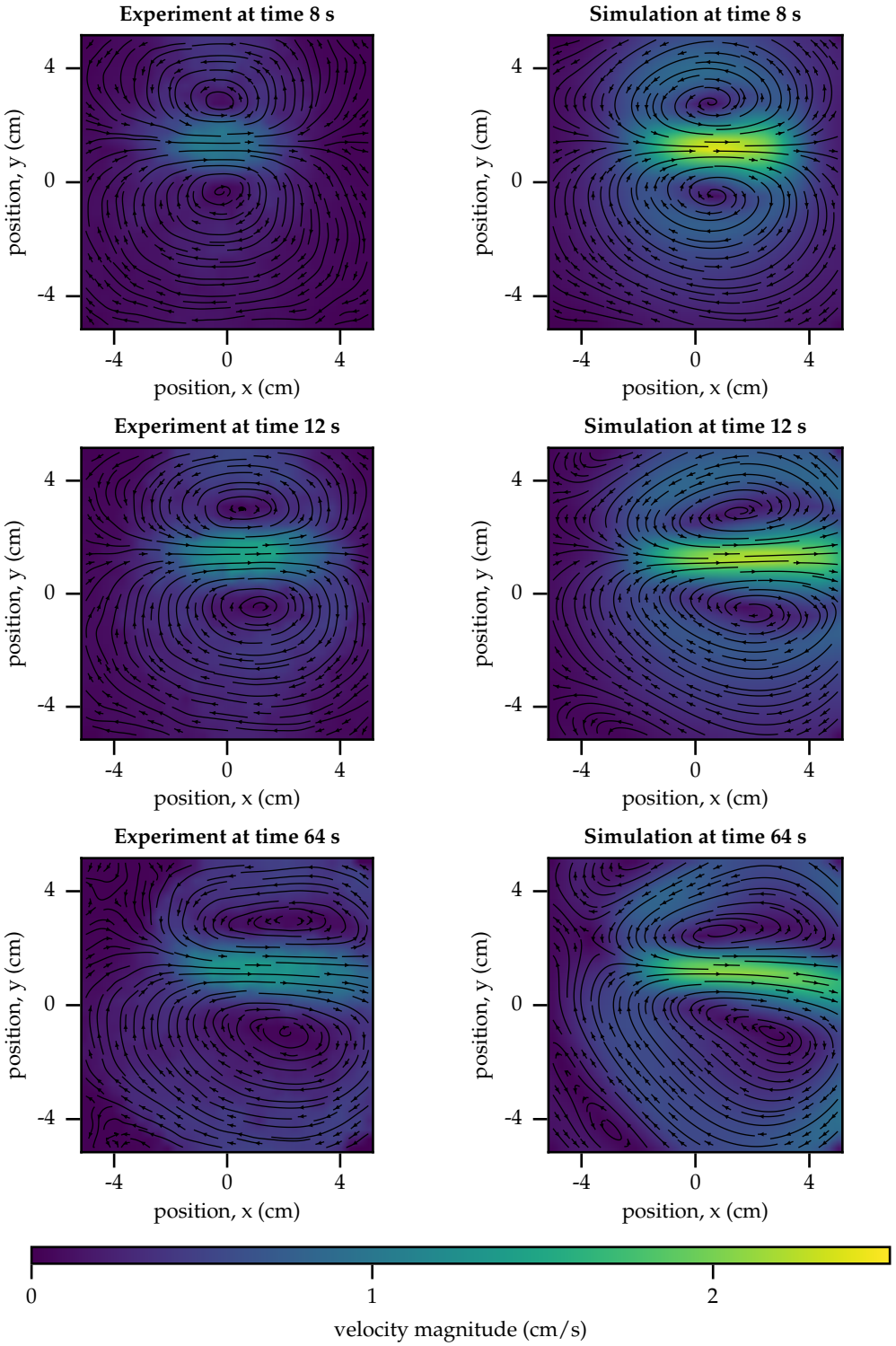


Figure 3.3: Scenario 1. Comparison of the velocity field obtained by PIV in the experimental setup (left column) and the velocity field produced by my simulation environment (right column).

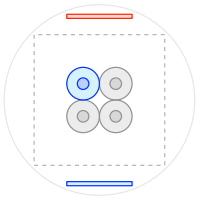


Figure 3.4: Configuration of the electrodes and coils in Scenario 1.

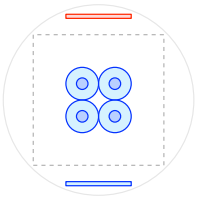


Figure 3.6: Configuration of the electrodes and coils in Scenario 2.

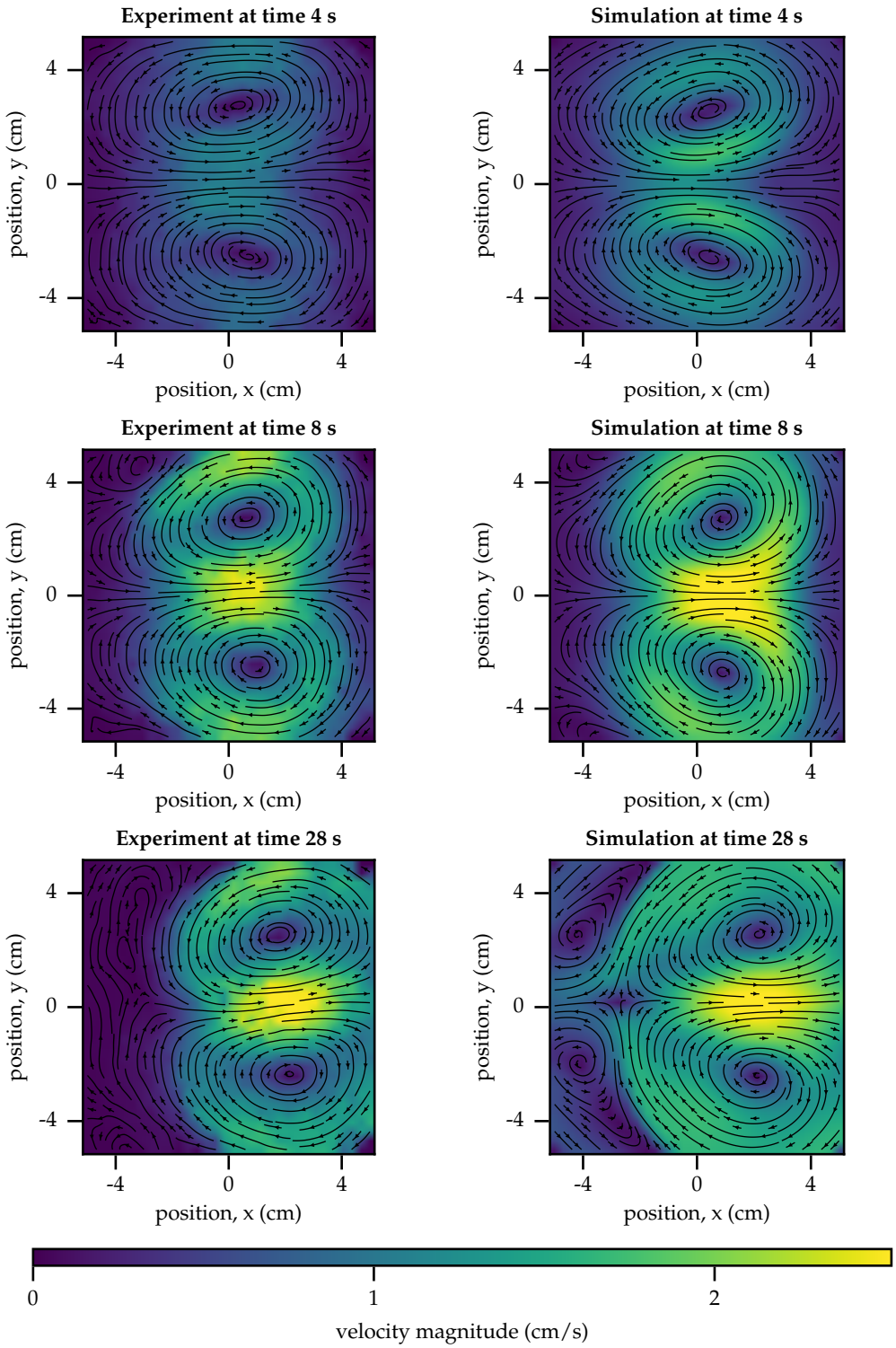


Figure 3.5: Scenario 2. Comparison of the velocity field obtained by PIV in the experimental setup (left column) and the velocity field produced by my simulation environment (right column).

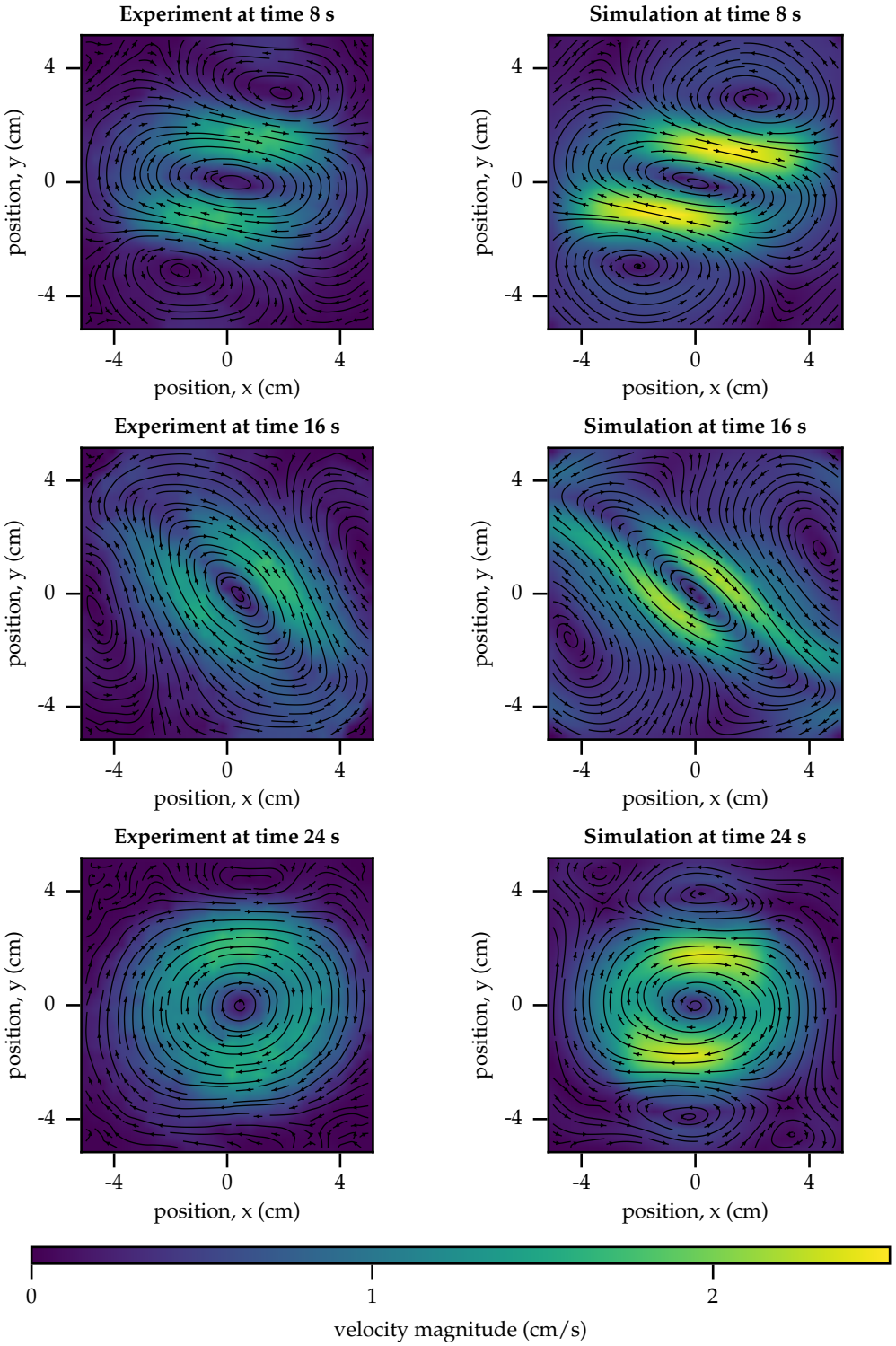


Figure 3.7: Scenario 3. Comparison of the velocity field obtained by PIV in the experimental setup (left column) and the velocity field produced by my simulation environment (right column).

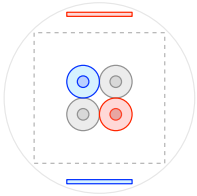


Figure 3.8: Configuration of the electrodes and coils in Scenario 3.

3.10 VALIDATION USING COMSOL MULTIPHYSICS®

Lastly, I compare the solutions obtained by my simulation environment with the solutions obtained by a commercial CFD software, COMSOL Multiphysics®. The COMSOL simulation encompasses the entire platform, including the fluid flow and the electric and magnetic fields, while the simulation environment only solves the fluid flow with the pre-computed basis of electric and magnetic fields from COMSOL.

There are further differences in the numerical methods used to solve the fluid flow which I summarize in Table 3.1. To validate the simulation environment, I compare the velocity fields obtained in the same configuration as in Scenario 3. The comparison is illustrated in Fig. 3.9. The flow fields obtained by the two methods are in good agreement. There may be slight time discrepancies, as the comparison at time $t = 24\,\mathrm{s}$ suggests, but the overall flow patterns are consistent.

The computational performance of the two methods is compared in Table 3.2. My simulation environment solves the fluid flow in almost one-tenth of the time required by COMSOL Multiphysics®. This is due to the simpler numerical methods used in the simulation environment, the structured mesh, and the decoupled solution strategy. It should be noted that the COMSOL simulation has a slightly higher number of DOFs, as the mesh is tetrahedral and automatically generated.

Table 3.1: Comparison of the simulation parameters for the simulation environment and COMSOL Multiphysics®.

Parameters	COMSOL Multiphysics®	Simulation Environment
Number of fluid DOFs	265438	209577
Mesh	Tetrahedral	Hexahedral
Method	Galerkin/Least-Squares	Galerkin
Spliting Scheme	—	IPCS
Temporal Scheme	BDF1/BDF2	BDF2
Timestep	Dynamic	0.25 s

Table 3.2: Comparison of the computational performance of the simulation environment and COMSOL Multiphysics®.

Metric	COMSOL Multiphysics®	Simulation Environment
Solve time (EM fields)	14 min 5 s	—
Solve time (CFD)	59 min	6 min 41 s

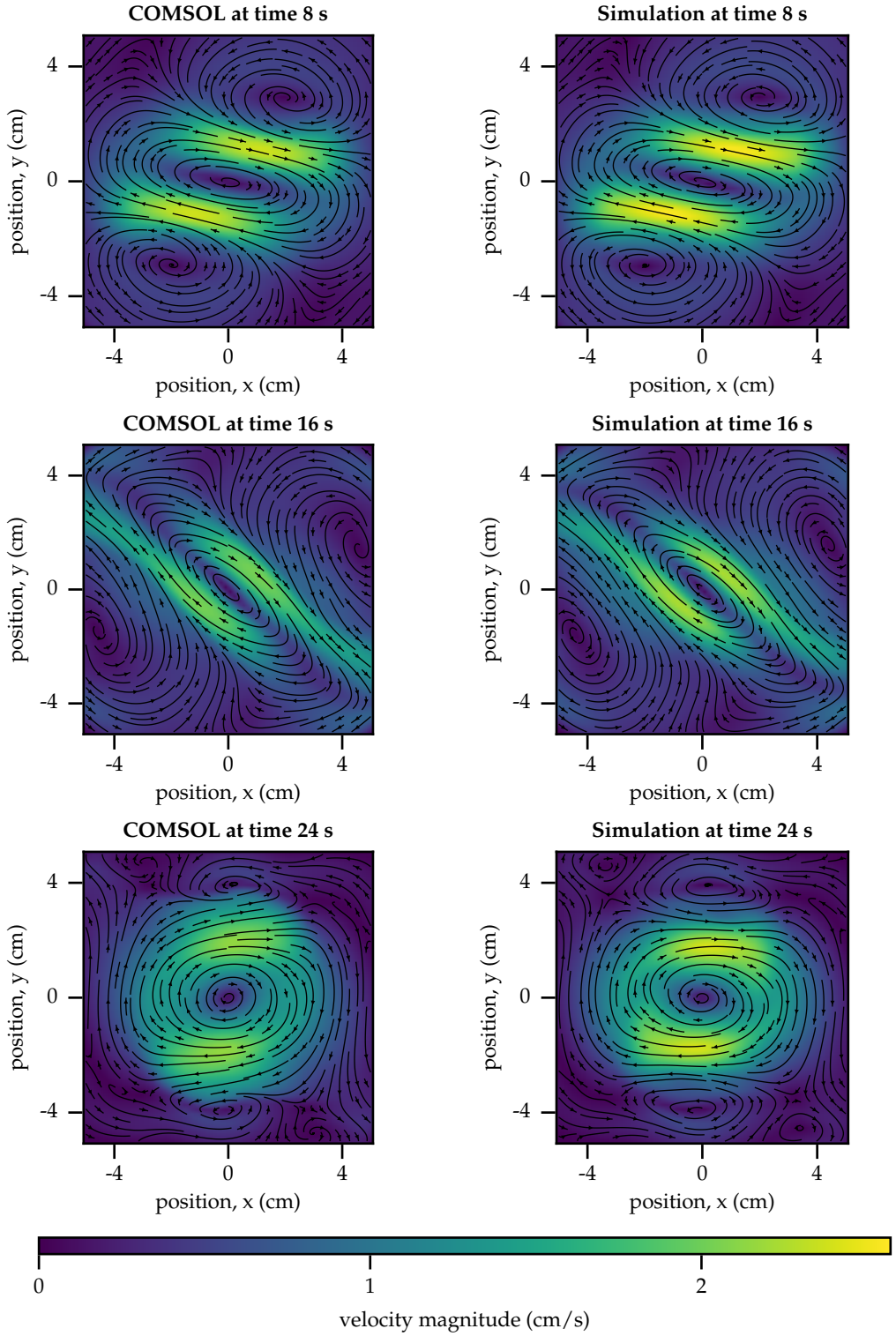


Figure 3.9: Comparison of the flow fields produced by COMSOL Multiphysics® (left column) and my simulation environment (right column). The situation is the same as in Experiment 3.

4

USED METHODS FOR IDENTIFICATION AND CONTROL

This chapter serves as an overview of the identification and control methods used in the thesis while also providing a brief overview of the relevant literature. First, I introduce the concept of surrogate modeling, motivating it with the Model Predictive Control framework. Then, I further elaborate on methods for constructing data-driven surrogate models and their use in the Model Predictive Control framework.

4.1 MOTIVATION: MODEL PREDICTIVE CONTROL

With the rise of computational power, real-time optimization-based controllers have become increasingly popular, as they can handle complex systems with nonlinear dynamics and constraints. However, why should I be interested in optimization-based controllers for the problem at hand? The answer dwells in the fact that I want to minimize the error between the desired flow field and the actual flow field, which is an optimization problem.

A concrete example of an optimization-based controller is Model Predictive Control (MPC). Given a measured or estimated state $\mathbf{x}(t)$ at time t, MPC computes the optimal control input $\mathbf{u}(t)$ by solving the following optimization problem:

$$\underset{\{\mathbf{u}_k\}_{k=0}^{N_\mathrm{p}-1},\{\mathbf{x}_k\}_{k=1}^{N_\mathrm{p}}}{\text{minimize}} \quad J\left(\{\mathbf{x}_k\}_{k=1}^{N_\mathrm{p}}, \{\mathbf{u}_k\}_{k=0}^{N_\mathrm{p}-1}\right), \tag{4.1a}$$

$$\text{subject to} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, \dots, N_\mathrm{p} - 1, \tag{4.1b}$$

$$\mathbf{x}_k \in \mathcal{X}, \quad k = 1, \dots, N_\mathrm{p}, \tag{4.1c}$$

$$\mathbf{u}_k \in \mathcal{U}, \quad k = 0, \dots, N_\mathrm{p} - 1, \tag{4.1d}$$

$$\mathbf{x}_0 = \mathbf{x}(t), \tag{4.1e}$$

where J is the cost function, $\mathbf{f}$ is the system dynamics, $\mathcal{X}$ and $\mathcal{U}$ are the state-space and input-space constraints, respectively, and N_p is the prediction horizon. The control input $\mathbf{u}(t)$ applied to the system is obtained as the first element of the optimal control sequence $\left\{\mathbf{u}_k^\star\right\}_{k=0}^{N-1}$. The entire process is repeated at the next timestep, with the state $\mathbf{x}(t)$ updated to the new measured or estimated state.

4.1.1 *Surrogate Modeling*

It is necessary to solve the optimization problem Eq. (4.1) in real-time within the control timestep. This is a challenging task, particularly for highly dimensional nonlinear systems. Consider, for instance, system dynamics $\mathbf{f}$ arising from the spatio-temporal discretization of a PDE, such as the FEM-based scheme discussed in Chapter 3. In such cases, the state may

exhibit high dimensionality coupled with nonlinear dynamics, rendering the optimization problem intractable.

Fortunately, the optimization problem is solved on a finite horizon, requiring only a prediction of the system behavior over a time window of length N_p. Thus, the idea of replacing the original system dynamics in Eq. (4.1) with a *surrogate model* arises. The surrogate model should give an accurate prediction of the system behaviour over the prediction horizon while making the optimization problem in Eq. (4.1) much more manageable. Methods used for constructing such surrogate models, as well as their use in the MPC framework, are the subject of the following sections.

4.2 DELAY EMBEDDINGS

It may be expensive or outright impossible to measure or estimate the entire state of such a complex system as fluid flow. Moreover, I am interested in controlling only the flow field that I am able to measure, i. e., the flow in the top fluid layer. Thus, it may be ill-advised to construct the surrogate model with the full state, as the input-output dynamics can be simpler than the full state-space dynamics. Fortunately, by means of Takens' embedding theorem [Takens, 1981] and its forced variant [Stark et al., 1997], the input-output dynamics can be reconstructed from the past measurements and the past inputs. The idea is to construct the state vector $\mathbf{x}_k$ at time k from the time series of d past measurements $\mathbf{y}_{k-d+1}, \dots, \mathbf{y}_k$ and $d-1$ past control-inputs $\mathbf{u}_{k-d}, \dots, \mathbf{u}_{k-1}$ as

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{y}_{k-d+1}^\mathsf{T} & \dots & \mathbf{y}_k^\mathsf{T} & \mathbf{u}_{k-d}^\mathsf{T} & \dots & \mathbf{u}_{k-1}^\mathsf{T} \end{bmatrix}^\mathsf{T}, \tag{4.2}$$

where $d \geq 1$ is the embedding dimension, left to be determined. The reconstructed state $\mathbf{x}_k$ is then used to construct the surrogate model.

4.3 PROPER ORTHOGONAL DECOMPOSITION

A well-established method for constructing surrogate models, namely that of a reduced-order, is the Proper Orthogonal Decomposition (POD), also called the Principal Component Analysis (PCA) in the statistics community or the Karhunen-Loève expansion in the stochastic processes community. POD was first introduced by Lumley [1967] to study coherent structures in turbulent flow and later popularized by Sirovich [1987]. In POD, the evolution of some spatio-temporal quantity of interest, such as the flow field, is approximated by a linear combination of a few spatial modes with time-varying coefficients, i. e.,

$$\mathbf{y}(\mathbf{r}, t) \approx \sum_{i=1}^{r} a_i(t) \boldsymbol{\xi}_i(\mathbf{r}), \tag{4.3}$$

where $\mathbf{y}(\mathbf{r}, t)$ is the quantity of interest, $\mathbf{r}$ is the spatial coordinate, t is the time, r is the number of modes, $a_i(t)$ are the time-varying coefficients, and $\boldsymbol{\xi}_i(\mathbf{r})$ are the spatial modes.

The POD modes are usually obtained in the following data-driven manner. Starting with a series of snapshots of the quantity of interest at different time instances $t_1, t_2, \ldots, t_N$, and spatial coordinates $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M$, organized into a snapshot matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}(\mathbf{r}_1, t_1) & \ldots & \mathbf{y}(\mathbf{r}_1, t_N) \\ \vdots & \ddots & \vdots \\ \mathbf{y}(\mathbf{r}_M, t_1) & \ldots & \mathbf{y}(\mathbf{r}_M, t_N) \end{bmatrix}, \tag{4.4}$$

POD modes are extracted by performing Singular Value Decomposition (SVD) on the snapshot matrix $\mathbf{Y}$

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}, \tag{4.5}$$

then selecting the first r left singular vectors as the POD modes denoted as

$$\boldsymbol{\xi}_i = \mathbf{U}_i, \quad i = 1, \ldots, r. \tag{4.6}$$

The choice of the number of modes r typically involves analyzing the decay of the singular values σ_i of the matrix $\mathbf{Y}$. One common approach is to select the number of modes that collectively capture a certain percentage of the total energy, such as 99%.

The obtained modes can then be used to project the original equations of motion onto a low-dimensional subspace, essentially obtaining a reduced-order model for the evolution of the temporal coefficients. Many control strategies have been developed based on POD, see, e.g., [Bergmann et al., 2005, Hinze and Volkwein, 2005, Peitz et al., 2019].

4.4 KOOPMAN OPERATOR-BASED APPROACHES

A prominent class of system models are linear models. They are simple, interpretable, and host a plethora of mature control design methods. Linear models typically come from linearizing the system dynamics about an equilibrium point, but the validity region of such models is usually small. Here, I present a class of linear models that are not obtained by linearizing the system dynamics about an equilibrium point but acting as a global linearization of the system, making them much more suitable for the task at hand. First, let me consider the system dynamics in the form of

$$\mathbf{x}^+ = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad \mathbf{x} \in \mathcal{X}, \quad \mathbf{u} \in \mathcal{U}, \tag{4.7}$$

where $\mathcal{X}$ is the state-space of the system and $\mathcal{U}$ is the input-space of the system. Moreover, consider the output equation of the system in the form of

$$\mathbf{y} = \mathbf{g}(\mathbf{x}). \tag{4.8}$$

Given a state $\mathbf{x}_k$ at time k, the goal is to predict the output $\mathbf{y}_{k+i}$ over some prediction horizon N_p. To this end, I define an artificial dynamical system, so-called *Koopman predictor*, in the form of

$$\mathbf{z}_0 = \mathbf{\Psi}(\mathbf{x}_k), \tag{4.9a}$$

$$\mathbf{z}_{i+1} = \mathbf{A}\mathbf{z}_i + \mathbf{B}\mathbf{u}_{k+i}, \quad i = 0, \ldots, N_\mathrm{p} - 1, \tag{4.9b}$$

$$\hat{\mathbf{y}}_{k+i} = \mathbf{C}\mathbf{z}_i, \quad i = 1, \ldots, N_\mathrm{p}, \tag{4.9c}$$

where $\mathbf{z} \in \mathbb{R}^N$ is the lifted state, $\boldsymbol{\Psi} : \mathcal{X} \to \mathbb{R}^N$ is the lifting map, $\mathbf{A} \in \mathbb{R}^{N \times n}$, $\mathbf{B} \in \mathbb{R}^{N \times m}$, $\mathbf{C} \in \mathbb{R}^{p \times N}$ are the system matrices, N is the dimension of the lifted state space, m is the dimension of the input space, and p is the dimension of the output space. Finally, the predicted output is denoted as $\hat{\mathbf{y}}_i$. The existence of such a system is justified by the Koopman operator theory, which I will now briefly introduce.

4.4.1 *The Koopman Operator*

The idea of the Koopman Operator essentially boils down to capturing the dynamics of a nonlinear system by a linear yet infinite-dimensional operator. It dates back to the works of Koopman [1931] and Koopman and v. Neumann [1932] but has gained popularity in the control community only relatively recently. To set the stage, I now quickly go over the operator's definition. Consider an autonomous, unforced dynamical system in the form of

$$\mathbf{x}^+ = \mathbf{f}(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X}. \tag{4.10}$$

Let me call a function $g : \mathcal{X} \to \mathbb{R}$ an *observable* of the system. The Koopman operator $\mathcal{K}$ is then defined on the space of observables as:

$$\mathcal{K}g = g \circ \mathbf{f}, \tag{4.11}$$

where $\circ$ denotes the composition of functions. Even though the underlying dynamical system is nonlinear, the Koopman operator is always linear, albeit generally infinite-dimensional.

The definition of the Koopman operator can also be extended to controlled systems in the form of Eq. (4.7). This extension is achieved by introducing so-called *extended state-space*, i.e, the state-space augmented with the of all possible control input sequences, i.e.,

$$\mathcal{S} = \mathcal{X} \times \ell(\mathcal{U}), \tag{4.12}$$

where

$$\ell(\mathcal{U}) = \left\{ \{\mathbf{u}_k\}_{k=0}^{\infty} \mid \mathbf{u}_k \in \mathcal{U} \right\}.$$

Let me now consider the observable defined on the extended state-space, i. e., $g : \mathcal{S} \to \mathbb{R}$. The Koopman operator for controlled systems then reads:

$$\mathcal{K}g\left(\mathbf{x}, \{\mathbf{u}_k\}_{k=0}^{\infty}\right) = g\left(\mathbf{f}(\mathbf{x}, \mathbf{u}_0), \{\mathbf{u}_k\}_{k=1}^{\infty}\right). \tag{4.13}$$

The main idea of the Koopman operator-based approaches is to essentially try to find a good finite set of observables $g_1, \ldots, g_N$, compose them into the feature map

$$\boldsymbol{\Psi}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) & \ldots & g_N(\mathbf{x}) \end{bmatrix}^{\mathsf{T}}, \tag{4.14}$$

and find the system Eq. (4.9) or similar, as the finite-dimensional approximation of the Koopman operator in Eq. (4.13).

This is a valid approach because when the set of observables spans an invariant subspace, meaning for every $g \in \operatorname{span}\{g_1, \ldots, g_N\}$ it holds that

$$\mathcal{K}g \in \operatorname{span}\{g_1, \ldots, g_N\}, \tag{4.15}$$

then, the Koopman operator can be represented exactly by a finite-dimensional matrix. Even if the observables do not span an invariant subspace, it is still possible to look for the set of observables that approximate the Koopman operator well, i. e., the set of observables that minimizes the error in the approximation.

4.4.2 *Extended Dynamic Mode Decomposition*

Probably the most well-known method for constructing Koopman predictors is the Extended Dynamic Mode Decomposition (EDMD), first introduced by Williams et al. [2015]. EDMD is a data-driven method for approximating the Koopman operator with proven convergence in the unforced case in the data and dimensionality limit by Korda and Mezić [2018a].

The method goes as follows. Given a series of state measurements $\{\mathbf{x}_k\}_{k=0}^{N_\mathrm{d}}$ and control inputs $\{\mathbf{u}_k\}_{k=0}^{N_\mathrm{d}-1}$, which are assumed to be related by the system dynamics Eq. (4.7) and provided the lifting function $\boldsymbol{\Psi}$, EDMD computes the best-fit linear system matrices $\mathbf{A}$ and $\mathbf{B}$ as a solution to

$$\underset{\mathbf{A},\mathbf{B}}{\text{minimize}} \sum_{k=0}^{N_\mathrm{d}-1} \|\boldsymbol{\Psi}(\mathbf{x}_{k+1}) - \mathbf{A}\boldsymbol{\Psi}(\mathbf{x}_k) - \mathbf{B}\mathbf{u}_k\|_2^2 . \tag{4.16}$$

Moreover, the output matrix $\mathbf{C}$ may be obtained as the minimizer of the following optimization problem

$$\underset{\mathbf{C}}{\text{minimize}} \sum_{k=0}^{N_\mathrm{d}} \|\mathbf{x}_k - \mathbf{C}\boldsymbol{\Psi}(\mathbf{x}_k)\|_2^2 . \tag{4.17}$$

These optimization problems have closed-form solutions. Stacking the lifted states into matrices

$$\mathbf{Z} = \begin{bmatrix} \boldsymbol{\Psi}(\mathbf{x}_0) & \boldsymbol{\Psi}(\mathbf{x}_1) & \dots & \boldsymbol{\Psi}(\mathbf{x}_{N_\mathrm{d}-1}) \end{bmatrix}, \quad \mathbf{Z}^+ = \begin{bmatrix} \boldsymbol{\Psi}(\mathbf{x}_1) & \boldsymbol{\Psi}(\mathbf{x}_2) & \dots & \boldsymbol{\Psi}(\mathbf{x}_{N_\mathrm{d}}) \end{bmatrix}, \tag{4.18}$$

and the control inputs into a matrix

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \dots & \mathbf{u}_{N_\mathrm{d}-1} \end{bmatrix}, \tag{4.19}$$

the matrices $\mathbf{A}$ and $\mathbf{B}$ can then be obtained as

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} = \mathbf{Z}^+ \begin{bmatrix} \mathbf{Z} \\ \mathbf{U} \end{bmatrix}^\dagger , \tag{4.20}$$

where † denotes the Moore–Penrose pseudoinverse. The matrix $\mathbf{C}$ matrix can then be obtained in an analogous manner.

The predictor's performance depends heavily on the choice of the lifting map $\boldsymbol{\Psi}$, i. e., the choice of observables. There are many possible choices for the lifting map, typically the observables are chosen to form a basis of some functional space, but other choices are possible, e. g., Mamakoukas et al. [2019] proposed to use the observable in form of the higher order derivatives of the underlying nonlinear functions, Arbabi et al. [2018] used delay embeddings, which itself are a form of nonlinear lifting.

It is of note that when considering $\boldsymbol{\Psi}(\mathbf{x}) = \mathbf{x}$, the EDMD presented here reduces to the standard Dynamic Mode Decomposition (DMD) in the unforced case and Dynamic Mode Decomposition with Control (DMDc) in the forced case. These algorithms were introduced by Schmid [2010] and Proctor et al. [2016], respectively. Moreover, the original DMD and DMDc algorithms first project the data onto a low-dimensional subspace spanned by the dominant POD modes, thus giving a reduced-order model.

4.4.3 *Other Methods*

There exist other approaches of approximating the Koopman operator, which I do not use in this thesis. However, I will briefly mention them here. Notable are approaches that use neural networks to simultaneously learn the lifting function and the linear system matrices, e. g., [Li et al., 2020, Fan et al., 2022, Mandić et al., 2023]. Korda and Mezić [2020] constructed both the lifting function and the system matrices as a solution of an optimization problem. Lastly, Peitz and Klus [2019] constructed a surrogate model for a PDE system with a switched control input using a set of Koopman operators, each corresponding to a different input mode this approach was later extended to interpolating between different Koopman operators in [Peitz et al., 2020].

4.4.4 *Koopman Model Predictive Control*

Koopman Model Predictive Control, shortly Koopman MPC, introduced by Korda and Mezić [2018b], is a way of exploiting Koopman predictors for the purposes of MPC. Consider the standard linear MPC problem with the system dynamics given by the Koopman predictor in Eq. (4.9), i. e.,

$$\begin{aligned}
\underset{\{\mathbf{u}_k\}_{k=0}^{N_\mathrm{p}-1},\{\mathbf{z}_k\}_{k=1}^{N_\mathrm{p}}}{\text{minimize}} \quad & J\left(\{\mathbf{z}_k\}_{k=1}^{N_\mathrm{p}},\{\mathbf{u}_k\}_{k=0}^{N_\mathrm{p}-1}\right), && \text{(4.21a)}\\
\text{subject to} \quad & \mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{B}\mathbf{u}_k, \quad k = 0,\dots,N_\mathrm{p}-1, && \text{(4.21b)}\\
& \mathbf{E}_k\mathbf{z}_k + \mathbf{F}_k\mathbf{u}_k \leq \mathbf{b}_k, \quad k = 1,\dots,N_\mathrm{p}, && \text{(4.21c)}\\
& \mathbf{z}_0 = \boldsymbol{\Psi}(\mathbf{x}_0), && \text{(4.21d)}
\end{aligned}$$

where J is a convex quadratic cost function, $\mathbf{A}$ and $\mathbf{B}$ are the Koopman system matrices, $\mathbf{E}_k$ and $\mathbf{F}_i$ are the state and input constraints, and $\mathbf{b}_k$ are the constraint bounds. This problem is a convex quadratic program, which means that it can be solved efficiently by specialized solvers, and the solution is guaranteed to be a global optimum. Furthermore, the problem can be reformulated into a so-called *dense* formulation, i. e., a quadratic program only in terms of the control inputs, meaning that the dimension of the lifted state does not affect the computational complexity of the problem.

4.5 DEEP LEARNING-BASED APPROACHES

Deep learning-based approaches represent the surrogate models using Neural Networks (NNs). The main advantage of NNs is that they can approximate highly nonlinear functions, which is particularly useful for systems with complex dynamics. However, they are also black box models, meaning that they are hard to interpret, and they are typically data-hungry, meaning that they require a large amount of data to learn the dynamics properly. Recently, there has been a surge of interest in reducing the data requirements of NN-based models by embedding symmetries, conservation laws, and other physical constraints into the model, see, e.g., Raissi et al. [2019].

There exist many works dealing with learning NN-based surrogate models from data, see, e.g., [Parish and Carlberg, 2020, Regazzoni et al., 2019, Fresca et al., 2021]. However, they usually do not focus on the learning model for the purposes of control.

4.5.1 *Deep Model Predictive Control*

Deep Model Predictive Control, shortly DeepMPC, is the use of the NN predictor in the MPC framework. The optimization problem arising from DeepMPC is typically non-convex due to the presence of the NN model, and thus is generally hard to solve. Despite this, DeepMPC has shown promising results in many applications across different fields, e.g., Lenz et al. [2015] used DeepMPC for robotic foodcutting, Baumeister et al. [2018] used DeepMPC for self-tuning of mode-locked lasers, Bieker et al. [2020] controlled a chaotic fluid flow using DeepMPC and more recently, Lugagne et al. [2024] used DeepMPC to control gene expression in thousands of single cells simultaneously.

5

FLUID FLOW SHAPING BY COMMANDING COILS AND KEEPING ELECTRODES AT FIXED POTENTIALS

This chapter focuses on designing control methods to shape the fluid flow field by adjusting individual coil currents while keeping the electrode potentials fixed. This approach simplifies the control problem by eliminating the inherent nonconvexity caused by the multiplication of coil and electrode commands in the body force term. I elaborate on the concept of this nonconvexity and tackle the full control problem in the following chapter, Chapter 6. Despite the simplification, this task still allows for a fairly rich set of different fluid flow patterns to be achieved.

Briefly going over the chapter, I first describe the setup of the control problem. Following this, I detail the process of generating data for the purpose of identifying the data-driven models. Subsequently, I present the design of controllers based on the DeepMPC and Koopman MPC methods. Concluding the chapter, I present the performance of both control methods.

Notes on the Chapter

Chronologically, my initial focus was on developing the Koopman MPC-based approach, which showed promise due to its resulting MPC problem being convex, allowing for rapid and optimal solutions. However, this approach failed to maintain the desired flow shape as the learned model struggled to capture system's steady-state behavior. Consequently, I devised the DeepMPC approach, which showcased improved performance, primarily attributed to its training methodology involving the minimization of multi-step prediction error. Yet, after refining the identification signal to induce varied steady states better, I observed the Koopman MPC approach achieving competitive performance akin to the DeepMPC approach.

5.1 SETUP

The setup is illustrated in Fig. 5.1. Four electrodes and four coils are used, giving $n_{\text{el}} = 4$ and $n_{\text{coil}} = 4$. The electrodes are kept at a fixed potentials, employing commands (equivalent to potentials in Volts, see definition in Section 3.5) of $\phi_1 = 10$, $\phi_2 = 5$, $\phi_3 = 5$, and $\phi_4 = 0$. The coils are considered to be the actuators, and thus, the coil commands from the control input

$$\mathbf{u} = \begin{bmatrix} \psi_1 & \psi_2 & \psi_3 & \psi_4 \end{bmatrix}^{\mathsf{T}}. \tag{5.1}$$

Furthermore, they are subject to box constraints

$$-1 \leq \psi_i \leq 1, \quad i = 1, \ldots, 4. \tag{5.2}$$

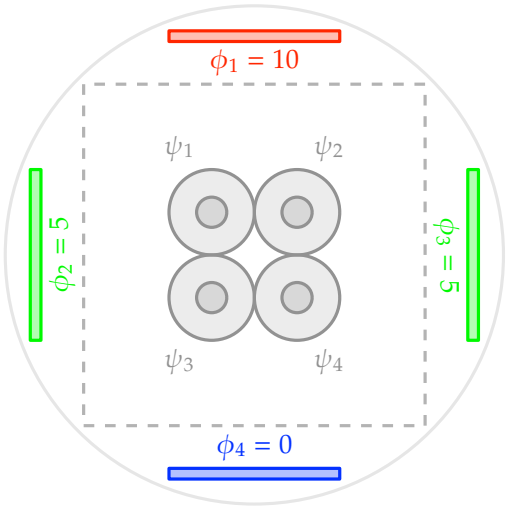


Figure 5.1: The coil based control setup. Dashed lines denote the 10 cm × 10 cm measurement domain. The electrodes are kept at fixed potentials, thus are colored. The coils are the actuators, and are shaded. The indexes of ϕs and ψs denote the assignment of the coil commands and the electrode commands to the coils and electrodes.

This more generally corresponds to the control input constraints

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}. \tag{5.3}$$

These constraints arise from the current limitation of the coils, which is ±440 mA.

The available measurements are the 2D top velocity field evaluated on a uniform 64 × 64 grid inside the 10 cm × 10 cm measurement domain highlighted in Fig. 5.1. The measurements are stacked into a vector $\mathbf{y}$, defined as

$$\mathbf{y} = \left[v_{1,1,x}, v_{1,1,y}, \ldots, v_{64,64,x}, v_{64,64,y}\right]^{\mathsf{T}}, \tag{5.4}$$

where $v_{i,j,x}$ and $v_{i,j,y}$ are the x and y components of the velocity at the ith grid row at the jth grid column, respectively.

5.2 DATA GENERATION

I seek to learn a model of the system dynamics from data. To this end, I employ my simulation environment to generate a dataset of different input-output trajectories, where each trajectory is a sequence of input-output pairs of the form

$$\left\{(\mathbf{u}_0, \mathbf{y}_0), (\mathbf{u}_1, \mathbf{y}_1), \ldots, (\mathbf{u}_{N-1}, \mathbf{y}_{N-1})\right\}, \tag{5.5}$$

where N is the length of the trajectory.

I always start the simulation from zero initial pressure and velocity. Therefore, the dynamics is completely driven by the identification signal. I run the simulation using the timestep of $\Delta t = 0.25$ s. However, I subsample the measurements to $\Delta t = 0.5$ s as the system dynamics

is much slower than this sampling rate. This gives me a longer time horizon in the MPC for the same computational cost. The total amount of the trajectories is 6, and length of each trajectory is $N = 22001$, giving the single trajectory length of 11 000 s. Next, I describe the identification signal used to generate the data, the symmetry exploited to double the trajectory amount, and the dimensionality reduction of the measurements.

5.2.1 *Identification Signal*

I found that it is necessary to have a signal that can drive the system into different steady-states and keep it there for a certain time, as otherwise, the inferred models are not able to capture the steady-state behavior of the system, and consequently, the control methods cannot keep the flow in the desired shape.

To this end, I designed the following pseudorandom identification signal

$$\mathbf{u}_{k+1} = \begin{cases} \mathbf{u}_k & \text{if } q_k \leq p, \\ \mathbf{s}_{k+1} & \text{if } q_k > p, \end{cases} \tag{5.6}$$

where p is the probability of the input staying the same, $q_k \sim U([0,1])$ is an uniformly distributed random variable, and $\mathbf{s}_k \sim U\left([-1,1]^4\right)$ is an uniformly distributed random vector within the bounds of the input space. Moreover, I initialize the sequence with $\mathbf{u}_0 = \mathbf{0}$. The probability p can be adjusted to control the expected time between changes in the input signal in the following way.

I now describe how to derive p for a given expected time between changes in the input signal. Whether the input signal changes or not is determined by the following binary random variable

$$c_k = [\![q_k > p]\!], \tag{5.7}$$

where $[\![\cdot]\!]$ is the Iverson bracket, which is equal to 1 if the condition inside the brackets is true, and 0 otherwise. The total number of changes in the input signal in the first N steps is then given by

$$C_N = \sum_{k=1}^{N} c_k. \tag{5.8}$$

Taking the expectation of the above equation gives

$$\mathbb{E}[C_N] = \sum_{k=1}^{N} \mathbb{E}[c_k] = \sum_{k=1}^{N} \mathbb{P}(q_k > p) = \sum_{k=1}^{N} 1 - p = N(1-p). \tag{5.9}$$

For the expected time between changes in the input signal, τ, the sampling period, Δt, the following relationship holds

$$\tau = \frac{N\Delta t}{C_N + 1}. \tag{5.10}$$

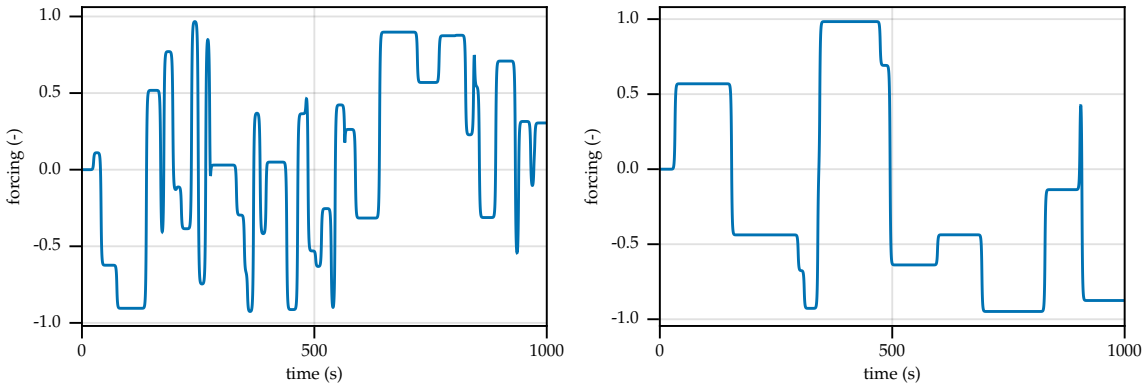
By substituting the expression for C_N into the above equation, doing some algebraic manipulations, and solving for p, I obtain

$$p = \frac{N+1}{N} - \frac{\Delta t}{\tau}. \tag{5.11}$$

For large enough N, the above relationship can be approximated as

$$p \approx 1 - \frac{\Delta t}{\tau}. \tag{5.12}$$

Furthermore, to obtain a bandlimited smooth signal, I apply a first-order low pass noncausal filter to the identification signal. I present examples of the identification signal for different values of τ in Fig. 5.2. The subplot Fig. 5.2a corresponds to the value of $\tau = 25\,\mathrm{s}$, and the subplot Fig. 5.2b corresponds to the value of $\tau = 80\,\mathrm{s}$, which is the value I used to generate the data.



(a) $\tau = 25\,\mathrm{s}$ (b) $\tau = 80\,\mathrm{s}$

Figure 5.2: One dimensional example of the identification signal for different values of τ.

5.2.2 *Exploiting Symmetry*

The system contains a symmetry with respect to the y-axis, allowing for the input-output pairs to be mirrored across the y-axis. This symmetry is exploited to double the dataset's size and improve the generalization of the learned models. I attempt to illustrate the symmetry in Fig. 5.3.

5.2.3 *Dimensionality Reduction*

I reduce the dimensionality of the trajectories by projecting them onto the dominant POD modes. I visualize the first six POD modes in Appendix b, specifically in Fig. b.1.

I obtain the POD modes using the SVD-based method, as I described in Chapter 4. The selection criterion for the number of modes is based on the energy captured by the modes, and I select the number of modes that capture about 90 % of the energy of the measurements,

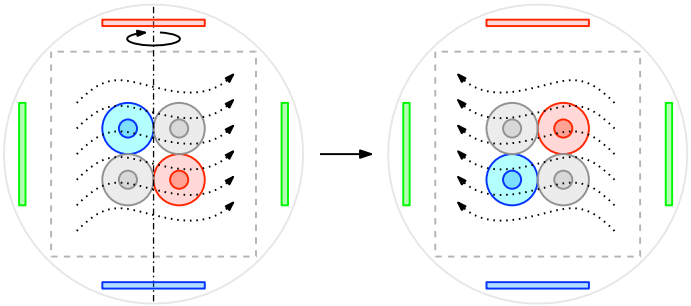


Figure 5.3: Illustration of the symmetry with respect to the y-axis.

which in this case is $r = 300$. The projection itself is done by first stacking the individual POD modes in the following matrix:

$$\Xi = \left[\xi_1, \ldots, \xi_r\right], \tag{5.13}$$

and then computing the reduced measurements as

$$\tilde{\mathbf{y}} = \Xi^{\mathsf{T}}\mathbf{y}. \tag{5.14}$$

5.3 DEEP MODEL PREDICTIVE CONTROL

The first type of control method I consider is DeepMPC. The idea is to learn a model of the system dynamics in the form of a NN and use this model in an MPC scheme. Specifically, the NN, represented by $\mathbf{f}_{\text{NN}}$, predicts the next state $\hat{\mathbf{x}}^+$ based on the current state $\mathbf{x}$ and control input $\mathbf{u}$, parameterized by θ, i. e.,

$$\hat{\mathbf{x}}^+ = \mathbf{f}_{\text{NN}}(\mathbf{x}, \mathbf{u}; \theta). \tag{5.15}$$

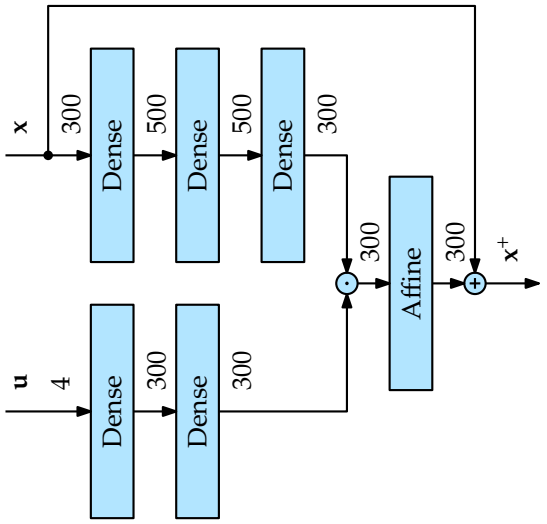
Employing the delay embeddings discussed in Chapter 4, I take the state to be equivalent to the POD projected velocity measurements from the previous section, i. e.,

$$\mathbf{x} = \tilde{\mathbf{y}}. \tag{5.16}$$

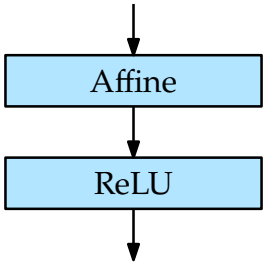
While using a single measurement as the state may seem unconventional, it proves sufficient for capturing system dynamics in this case. Moreover, it reduces the state space dimensionality, resulting in a smaller neural network and faster training and optimization processes.

5.3.1 *Architecture*

I illustrate the architecture of the NN as well as the input and output shapes of its layers in Fig. 5.4a. The network is composed of multiple fully connected (Dense) layers, each of these layers is composed of an affine transformation followed by a non-linear activation function, in my case the ReLU. The structure of a Dense layer is illustrated in Fig. 5.4b.



(a) Network architecture. The numbers denote the sizes of the layers.



(b) Structure of a Dense Layer.

Figure 5.4: Neural network predictor.

The network contains a multiplicative nonlinearity, more precisely, a Hadamard product between the embeddings of the state and the control input just before being passed to the final layer. The original system does not explicitly contain this type of nonlinearity. However, it helps the network to learn the system dynamics better. Moreover, the whole network has a residual structure, where the input to the network is added to the output of the last layer. The Euler-type integrators inspire this structure. More specifically, when the first order ODE of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{5.17}$$

is integrated using the explicit Euler method, the state-transition map takes the form of

$$\mathbf{x}^+ = \mathbf{x} + \Delta t \mathbf{f}(\mathbf{x}) = \mathbf{x} + \mathbf{g}(\mathbf{x}), \tag{5.18}$$

which is precisely the structure of the neural network.

5.3.2 *Training*

I employ a two-stage training procedure, simplified from the method described by Bieker et al. [2020]. Essentially, in the first stage, a one-step-ahead predictor is trained, while in the second stage, the model is trained in a recurrent manner over the entire prediction horizon. The first stage is conducted to avoid the vanishing or exploding gradient problem that arises when training recurrent neural networks. The second stage is conducted to train the network to predict accurately over the entire prediction horizon in the same manner as the MPC scheme will use it.

Thus, the first stage is done by minimizing the mean squared error of a one-step ahead prediction, i. e., the loss function for one sample $(\mathbf{x}, \mathbf{u}, \mathbf{x}^+)$ reads

$$\mathcal{L}(\theta) = \|\mathbf{f}_{\text{NN}}(\mathbf{x}, \mathbf{u}; \theta) - \mathbf{x}^+\|_2^2. \tag{5.19}$$

In the second stage, the network is trained in a recurrent manner by minimizing the mean squared error over the entire prediction horizon N_p. This is done by standard backpropagation using a time algorithm. I depict the unrolling of the network in Fig. 5.5.

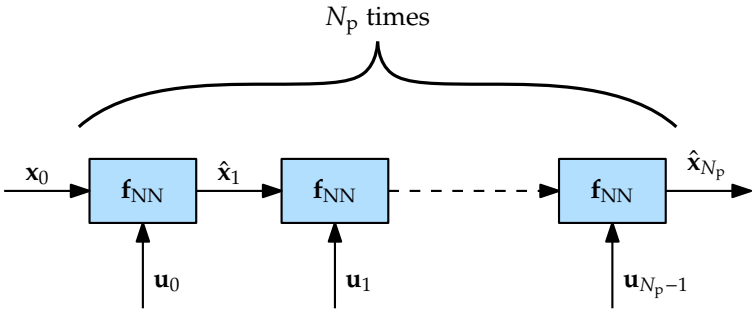


Figure 5.5: Unrolled recurrent neural network.

The loss function for the entire prediction horizon reads

$$\mathcal{L}_{\text{RNN}}(\theta) = \frac{1}{N_\text{p}} \sum_{k=1}^{N_\text{p}} \|\hat{\mathbf{x}}_k - \mathbf{x}_k\|_2^2, \tag{5.20}$$

where

$$\hat{\mathbf{x}}_{k+1} = \mathbf{f}_{\text{NN}}(\hat{\mathbf{x}}_k, \mathbf{u}_k; \theta), \quad k = 1, 2, \ldots, N_\text{p} - 1, \tag{5.21}$$

$$\hat{\mathbf{x}}_1 = \mathbf{f}_{\text{NN}}(\mathbf{x}_0, \mathbf{u}_0; \theta). \tag{5.22}$$

The network is trained stochastically using minibatches and the ADAM optimizer. For the used training hyperparameters, I refer the reader to Table 5.1. The parameters are the same for both stages of training.

Table 5.1: Training hyperparameters.

Hyperparameter	Value
Learning Rate	0.001
Batch Size	256
Number of Epochs	1000

5.3.3 *Model Predictive Control*

I adopt a standard formulation of a tracking MPC scheme as can be found, e. g., in [Borrelli et al., 2017]. The arising optimization problem reads

$$\begin{aligned}
\underset{\{\mathbf{u}_k\}_{k=0}^{N_\mathrm{p}-1}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{e}_{N_\mathrm{p}}^\mathsf{T}\mathbf{Q}\mathbf{e}_{N_\mathrm{p}} + \frac{1}{2}\Delta\mathbf{u}_0^\mathsf{T}\mathbf{R}\Delta\mathbf{u}_0 + \frac{1}{2}\sum_{k=1}^{N_\mathrm{p}-1}\mathbf{e}_k^\mathsf{T}\mathbf{Q}\mathbf{e}_k + \Delta\mathbf{u}_k^\mathsf{T}\mathbf{R}\Delta\mathbf{u}_k, && \text{(5.23a)}\\
\text{subject to} \quad & \mathbf{x}_{k+1} = \mathbf{f}_\mathrm{NN}(\mathbf{x}_k, \mathbf{u}_k; \boldsymbol{\theta}), \quad k = 0, 1, \ldots, N_\mathrm{p} - 1, && \text{(5.23b)}\\
& \mathbf{e}_k = \mathbf{r}_k - \mathbf{C}\mathbf{x}_k, \quad k = 1, 2, \ldots, N_\mathrm{p}, && \text{(5.23c)}\\
& \mathbf{u}_\mathrm{min} \leq \mathbf{u}_k \leq \mathbf{u}_\mathrm{max}, \quad k = 0, 1, \ldots, N_\mathrm{p} - 1, && \text{(5.23d)}\\
& \Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}, \quad k = 1, 2, \ldots, N_\mathrm{p} - 1, && \text{(5.23e)}\\
& \mathbf{x}_0 \text{ given.} && \text{(5.23f)}
\end{aligned}$$

In this context, $\mathbf{e}_k$ represents the tracking error, $\{\mathbf{r}_k\}_{k=1}^{N_\mathrm{p}}$ denotes the reference trajectory of the tracked quantity, $\Delta\mathbf{u}_k$ stands for the control rate, $\mathbf{Q}$ and $\mathbf{R}$ are the matrices weighting the tracking error and the control rate, respectively, and $\mathbf{u}_\mathrm{min}$ and $\mathbf{u}_\mathrm{max}$ are the lower and upper bounds on the control input. The matrix $\mathbf{C}$ serves as the output matrix, determining the tracked quantity. In the special case, when

$$\mathbf{C} = \boldsymbol{\Xi}, \tag{5.24}$$

the tracked quantity becomes the entire velocity field, as the output matrix projects the state back to the space of the original measurements. I solve this problem using the Sequential Quadratic Programming Solver from the *NLopt* library[1]. The gradients are computed using the automatic differentiation library *ReverseDiff*[2].

5.4 KOOPMAN OPERATOR-BASED CONTROL

The second type of control method I consider is Koopman MPC. The idea is to learn a model of the system dynamics by approximating the Koopman operator and then use this model in an MPC scheme. I start by identifying the system dynamics in form of the Koopman predictor as described in Chapter 4, i. e.,

$$\begin{aligned}
\mathbf{z}_0 &= \boldsymbol{\Psi}(\mathbf{x}_k), && \text{(5.25a)}\\
\mathbf{z}_{i+1} &= \mathbf{A}\mathbf{z}_i + \mathbf{B}\mathbf{u}_{k+i}, \quad i = 0, \ldots, N_\mathrm{p} - 1, && \text{(5.25b)}\\
\hat{\mathbf{y}}_{k+i} &= \mathbf{C}\mathbf{z}_i, \qquad\qquad i = 1, \ldots, N_\mathrm{p}. && \text{(5.25c)}
\end{aligned}$$

Here, I do not have to take the dimensionality of the Koopman state $\mathbf{z}$ into account, as it will be eliminated by the dense formulation of the Koopman MPC problem. Therefore, I still employ the delay embeddings of the POD projected velocity measurements as the state.

1 https://github.com/stevengj/nlopt
2 https://github.com/JuliaDiff/ReverseDiff.jl

However, I do not consider just a single measurement as in the case of the DeepMPC, but the past 4 measurements and the past 3 inputs, i. e.,

$$\boldsymbol{\Psi}(\mathbf{x}_k) = \begin{bmatrix} \tilde{\mathbf{y}}_{k-3}^\mathsf{T} & \tilde{\mathbf{y}}_{k-2}^\mathsf{T} & \tilde{\mathbf{y}}_{k-1}^\mathsf{T} & \tilde{\mathbf{y}}_{k}^\mathsf{T} & \mathbf{u}_{k-3}^\mathsf{T} & \mathbf{u}_{k-2}^\mathsf{T} & \mathbf{u}_{k-1}^\mathsf{T} \end{bmatrix}^\mathsf{T}. \tag{5.26}$$

The dimensionality of the Koopman state is 1212.

The velocity measurements in the original space can then be reconstructed from the Koopman state by selecting the corresponding values and multiplying them by the POD modes $\boldsymbol{\Xi}$. This relation gives the $\mathbf{C}$ matrix in the Koopman predictor for predicting the entire velocity field. Lastly, I obtain the matrices $\mathbf{A}$ and $\mathbf{B}$ of the Koopman predictor by applying the pseudoinverse-based DMD algorithm in Eq. (4.20) to all the trajectories in the dataset. The found matrix $\mathbf{A}$ has all the eigenvalues inside the unit circle, thus the Koopman predictor is stable.

I should note that for finding the predictor, it was vital to do the dimensionality reduction of the measurements using POD, as the solution of the DMD algorithm would take a long time (upwards of hours) otherwise, and the found predictory was badly conditioned.

5.4.1 *Model Predictive Control*

Having formed the Koopman predictor, I construct the Koopman MPC problem. Again, I use the standard tracking MPC formulation. However, the Koopman predictor now gives the system dynamics, i. e.,

$$\underset{\{\mathbf{u}_k\}_{k=0}^{N_\mathrm{p}-1}}{\text{minimize}} \quad \frac{1}{2}\mathbf{e}_{N_\mathrm{p}}^\mathsf{T}\mathbf{Q}\mathbf{e}_{N_\mathrm{p}} + \frac{1}{2}\Delta\mathbf{u}_0^\mathsf{T}\mathbf{R}\Delta\mathbf{u}_0 + \frac{1}{2}\sum_{k=1}^{N_\mathrm{p}-1} \mathbf{e}_k^\mathsf{T}\mathbf{Q}\mathbf{e}_k + \Delta\mathbf{u}_k^\mathsf{T}\mathbf{R}\Delta\mathbf{u}_k, \tag{5.27a}$$

$$\text{subject to} \quad \mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{B}\mathbf{u}_k, \quad k = 0, 1, \ldots, N_\mathrm{p} - 1, \tag{5.27b}$$

$$\mathbf{e}_k = \mathbf{r}_k - \mathbf{C}\mathbf{z}_k, \quad k = 1, 2, \ldots, N_\mathrm{p}, \tag{5.27c}$$

$$\mathbf{u}_\mathrm{min} \leq \mathbf{u}_k \leq \mathbf{u}_\mathrm{max}, \quad k = 0, 1, \ldots, N_\mathrm{p} - 1, \tag{5.27d}$$

$$\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}, \quad k = 0, 1, \ldots, N_\mathrm{p} - 1, \tag{5.27e}$$

$$\mathbf{z}_0 \text{ given by Eq. (5.26)}, \tag{5.27f}$$

$$\mathbf{u}_{-1} \text{ given}. \tag{5.27g}$$

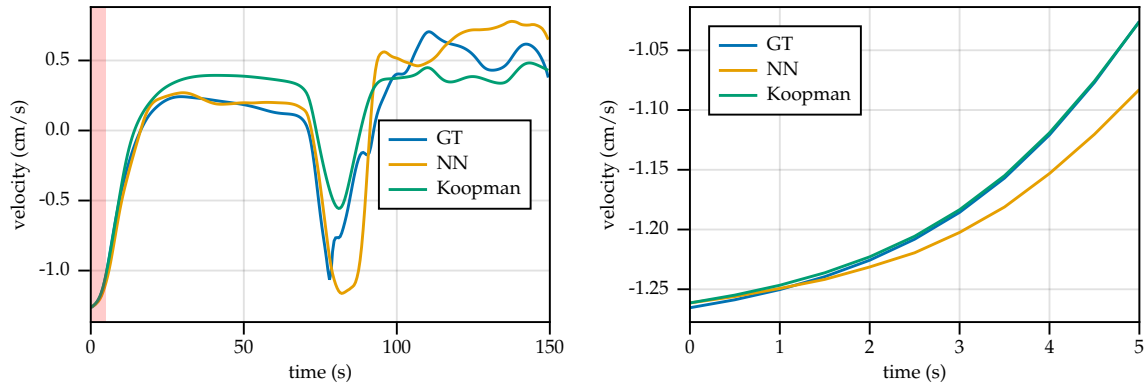
The matrices $\mathbf{Q}$ and $\mathbf{R}$ are the same as in the DeepMPC case, altough they should be positive semidefinite, and the matrices $\mathbf{A}$ and $\mathbf{B}$ are obtained from the Koopman predictor. The matrix $\mathbf{C}$ can be, again, constructed to express the quantity to be following the reference $\{\mathbf{r}_k\}_{k=1}^{N_\mathrm{p}}$. Furthermore, I solve this MPC problem in the dense formulation using the OSQP solver [Stellato et al., 2020].

5.5 RESULTS

In this section, I present the performance of the developed control methods as well as the prediction performance of the identified predictors.

5.5.1 *Prediction Performance*

I showcase the typical prediction performance of the predictors over a validation trajectory in Fig. 5.6. The figure shows the evolution of a x velocity at a certain measurement point. It shows both the long-term prediction and the prediction over the MPC prediction horizon of $N_\mathrm{p} = 10$ steps, or 5 s.



(a) Long-term prediction.

(b) Prediction on the prediction horizon.

Figure 5.6: Comparison of the prediction performance of the identified predictors. Red region denotes the prediction horizon.

The performance of the predictors is comparable. However, the Koopman predictor is slightly more accurate in the short-term prediction, while the NN predictor captures the close-to-steady-state behavior of the system more accurately.

5.5.2 *Control Performance*

Here, I present the control performance of the developed control methods. Using both controllers, I was, in simulations, able to achieve the desired flow in all four primary directions, i. e., up, down, left, and right. This was accomplished by setting the reference velocity over a certain region $\mathcal{R}$ of the domain to a constant desired value, e. g., $v_x = 0\,\mathrm{cm\,s^{-1}}, v_y = 1\,\mathrm{cm\,s^{-1}}$ for the up direction.

Practically, this involves constructing the output matrix $\mathbf{C}$ for the predictors, ensuring that the output represents the velocity field at the measurement points within the region $\mathcal{R}$, and taking the reference $\mathbf{r}$ to be the prescribed velocity field. I visualize the reference velocity fields together with the control regions for the cases of the up and right directions in Fig. 5.11. Furthermore, I take the $\mathbf{Q}$ and $\mathbf{R}$ in the cost functions of both MPCs to be

$$\mathbf{Q} = \frac{2 \cdot 10^3}{N_\mathcal{R}} \mathbf{I}_{2N_\mathcal{R} \times 2N_\mathcal{R}}, \quad \mathbf{R} = \mathbf{I}_{n_\mathrm{coil} \times n_\mathrm{coil}}, \tag{5.28}$$

where $\mathbf{I}$ is an identity matrix, and by $N_\mathcal{R}$ I denote the number of measurement points that lie within the region $\mathcal{R}$.

Up Direction

To assess the performance of the controllers, I study one of these directions, the up direction, in detail. I conduct a 160 s simulation for each controller, where at the 10 s mark, the reference velocity is set to $v_x = 0\,\mathrm{cm\,s^{-1}}, v_y = 1\,\mathrm{cm\,s^{-1}}$. Both of the simulations start from the same initial condition, i. e., zero pressure and velocity. I give the plots that display the spatially averaged velocity field across the control region in Fig. 5.8, the tracking error in Fig. 5.9, and the control inputs for the DeepMPC and Koopman MPC in Fig. 5.10. The tracking error is normalized by the initial error, and is defined as:

$$\mathrm{error}_k = \frac{\|\mathbf{y}_k - \mathbf{r}\|_2}{\|\mathbf{y}_0 - \mathbf{r}\|_2}, \tag{5.29}$$

where $\mathbf{y}_k$ is the actual velocity field at the k-th time step, and $\mathbf{r}$ is the reference velocity field. Moreover, I present the velocity fields at the end of the simulation in Fig. 5.7.

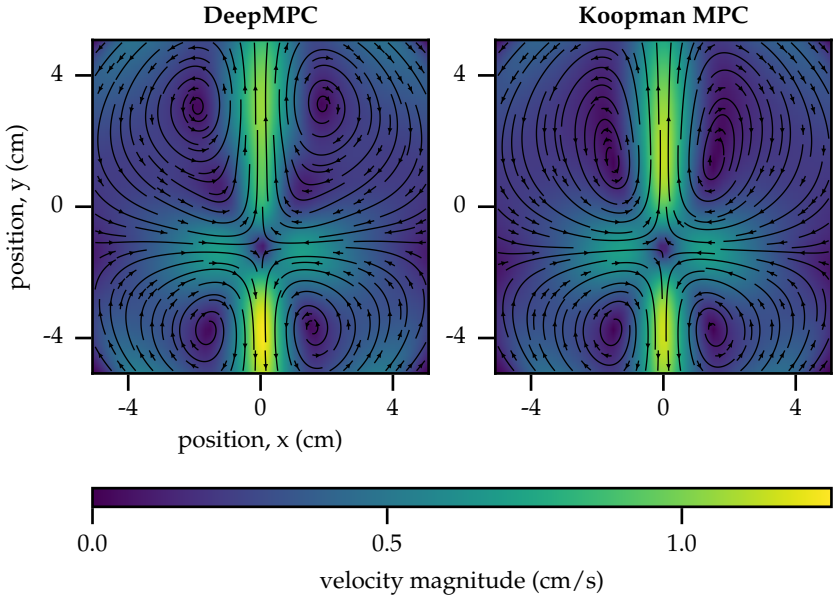
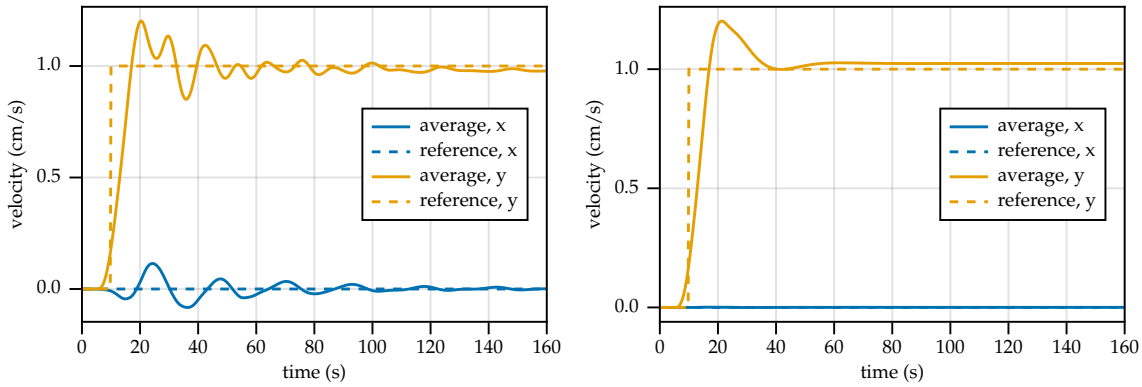
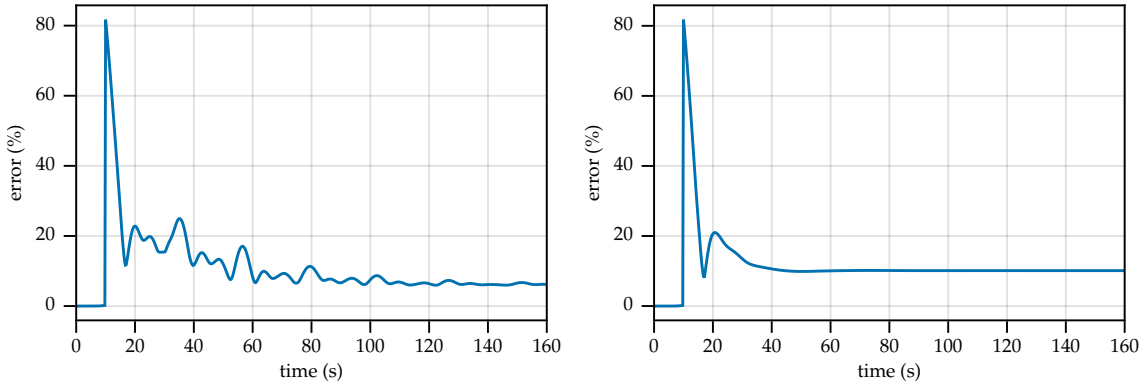


Figure 5.7: Velocity fields at the end of the simulation for the up direction.



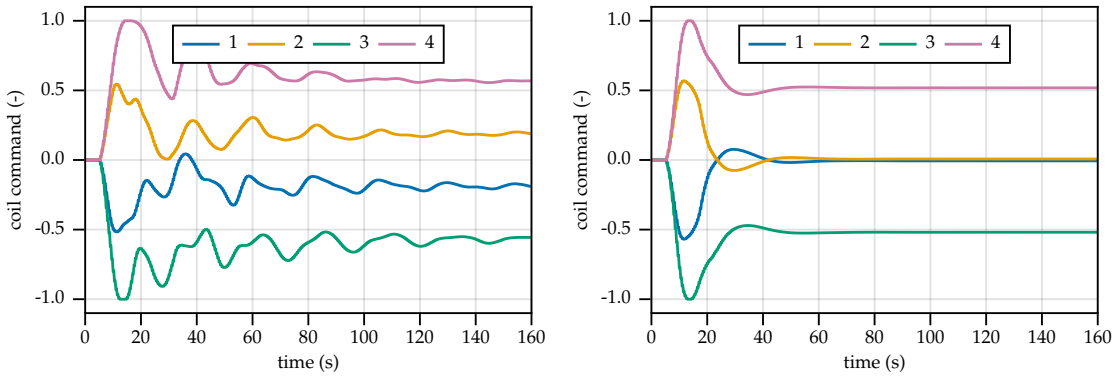
(a) DeepMPC (b) Koopman MPC

Figure 5.8: Comparison of spatially averaged velocity field for the DeepMPC and Koopman MPC.



(a) DeepMPC (b) Koopman MPC

Figure 5.9: Comparison of tracking error for the DeepMPC and Koopman MPC.



(a) DeepMPC (b) Koopman MPC

Figure 5.10: Comparison of control inputs for the DeepMPC and Koopman MPC.

From the point of minimizing the tracking error, the performance of both controllers seems to be comparable, as both controllers achieve the desired flow in the upward direction. The DeepMPC achieves the final error of about 6 % while the Koopman MPC achieves the final error of about 10 %. This appears to be in order with the prediction performance of the predictors, where the NN predictor seems to be more accurate when close to the steady state.

However, from the point of the quantity of the response and the control inputs, there is a clear difference between the two controllers. The DeepMPC seems to be driving the system in much more oscillatory manner, as evident from Fig. 5.8 and Fig. 5.9. Moreover, notice that in Fig. 5.10, the Koopman MPC puts the coils 1 and 2 to zero, while the DeepMPC does not. Since coils 1 and 2 are unnecessary to achieve the desired flow in the upward direction, the Koopman MPC demonstrates greater efficiency in this case.

Table 5.2: Comparison of MPC computation times for the DeepMPC and Koopman MPC.

Method	Average	Maximum	Minimum
DeepMPC	320 ms	731 ms	85 ms
Koopman MPC	219 µs	297 µs	193 µs

Furthermore, I compare the computational times[3] of both controllers in Table 5.2. The Koopman MPC is about 1000 times faster than the DeepMPC, which makes it viable for deployment on the real system. Unfortunately for the DeepMPC, it overruns the control timestep of 500 ms in the worst case, which makes it unsuitable for real-time control.

However, I am certain that the computational time of the DeepMPC could be improved upon by not completely relying on the automatic differentiation for the computation of the gradients. Currently, the gradients are computed by the automatic differentiation, which is okay for the network itself. However, the Jacobian matrix of the criterion w.r.t. the system output could be computed analytically.

3 The computation times are measured on the MacBook Pro 14" (2023) with the M2 Max chip.

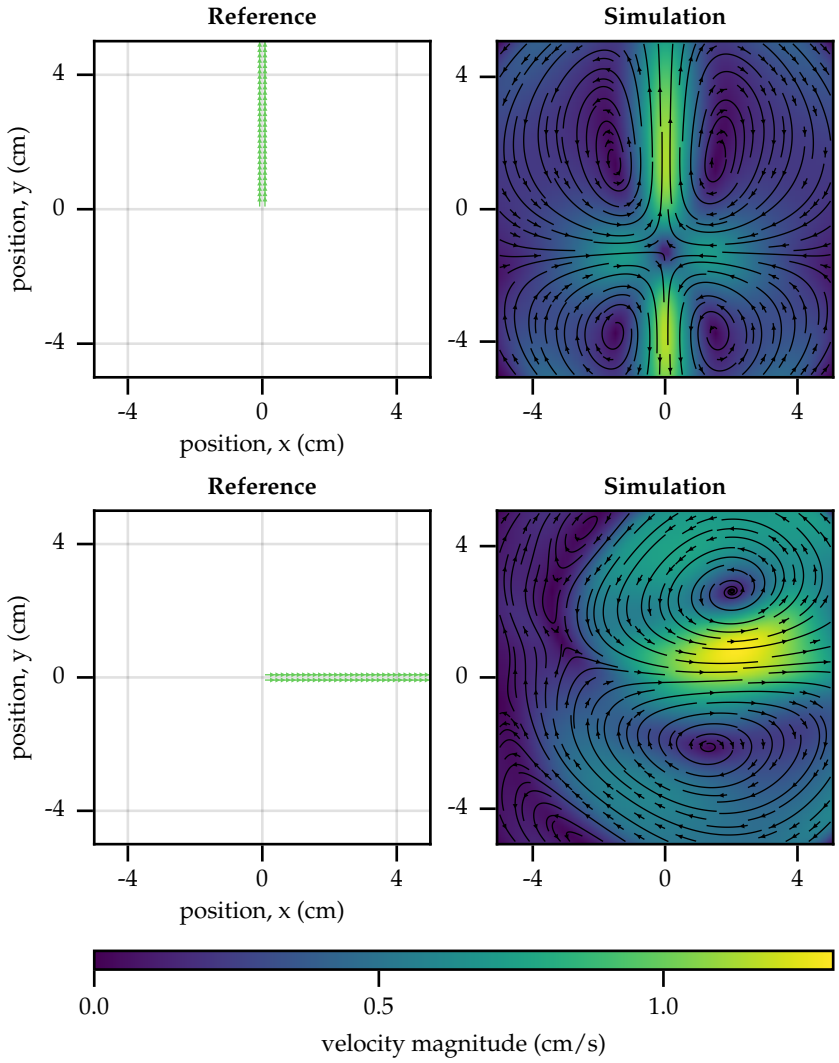


Figure 5.11: References for the up and right directions (left column) and their corresponding simulated velocity fields (right column). The arrows in the reference are placed at the measurement points, where the velocity field is prescribed. References for the left and down directions are analogous.

6

FLUID FLOW SHAPING BY COMMANDING BOTH COILS AND ELECTRODES

This chapter aims to design a control strategy for the full problem of driving the fluid flow of a conductive liquid to a desired shape by setting both the coil currents and the electrode potentials.

6.1 SETUP

Similarly to the previous chapter, I start by introducing the control setup. The setup is essentially the same as in the previous chapter but with the addition of the electrode commands as control inputs. This means that the control inputs are now the coil commands of the four middle coils:

$$\mathbf{u}_\psi = \begin{bmatrix} \psi_1 & \psi_2 & \psi_3 & \psi_4 \end{bmatrix}^\mathsf{T}, \tag{6.1}$$

and the electrode commands of the four peripheral electrodes

$$\mathbf{u}_\phi = \begin{bmatrix} \phi_1 & \phi_2 & \phi_3 & \phi_4 \end{bmatrix}^\mathsf{T}. \tag{6.2}$$

Thus, $n_{\text{coil}} = 4$ and $n_{\text{el}} = 4$. The assignment of the coil commands and the electrode commands to the coils and electrodes is shown in Fig. 6.1. Furthermore, the commands are bounded by the following constraints

$$0 \leq \phi_i \leq 10, \quad i = 1, 2, 3, 4, \tag{6.3}$$

and

$$-1 \leq \psi_i \leq 1, \quad i = 1, 2, 3, 4. \tag{6.4}$$

Generally, the constraints take the form of

$$\mathbf{u}_{\psi,\min} \leq \mathbf{u}_\psi \leq \mathbf{u}_{\psi,\max}, \quad \mathbf{u}_{\phi,\min} \leq \mathbf{u}_\phi \leq \mathbf{u}_{\phi,\max}. \tag{6.5}$$

Again, there are available 2D measurements of the top velocity field in a $10\,\text{cm} \times 10\,\text{cm}$ region denoted by the dashed line in Fig. 6.1. However, the measurements are taken on a grid of 32×32 points, half of the grid used in the previous chapter. This because the real platform uses this resolution of the PIV measurements. The measurements are stacked into a vector:

$$\mathbf{y} = \begin{bmatrix} v_{1,1,x}, v_{1,1,y}, v_{2,1,x}, v_{2,1,y}, \ldots, v_{32,32,x}, v_{32,32,y}, \end{bmatrix} \tag{6.6}$$

where $v_{i,j}$ is the x or y component of the velocity at the ith row and the jth column of the grid.

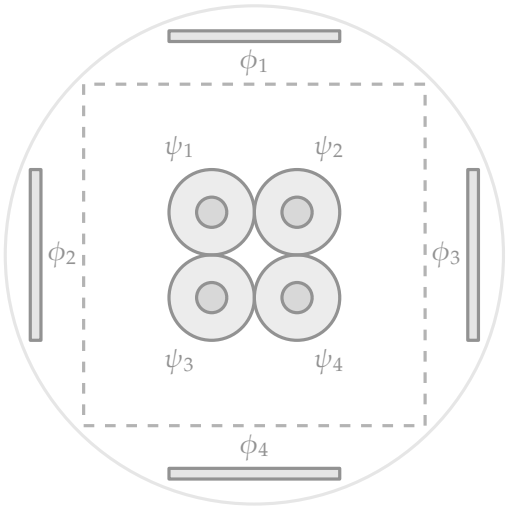
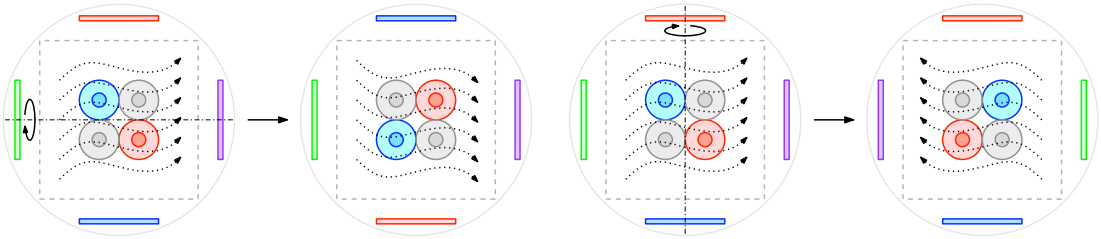


Figure 6.1: Full control setup. The dashed lines denote the 10 cm × 10 cm measurement region. The indexes of ϕs and ψs denote the assignment of the coil commands and the electrode commands to the coils and electrodes.

6.2 DATA GENERATION

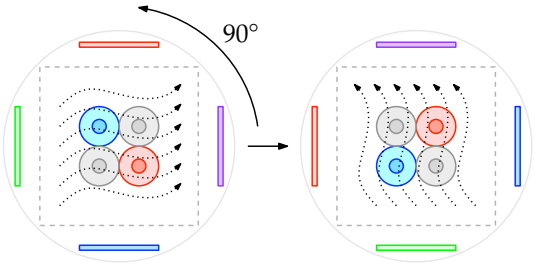
I use the same data as in the previous chapter. However, I augment them using symmetries of the full control problem. This is possible due to the full control problem containing many more symmetries than the reduced one. In addition to symmetry about the y axis, I can exploit the symmetry about the x axis and also rotational symmetries. I illustrate the symmetries in Fig. 6.2.

This allows me to take the six trajectories from the previous and generate the full dataset of 48 trajectories. The length of trajectories, therefore, stays the same as in the previous chapter, i.e., a total of 22001 samples, with the timestep of 0.5 s resulting in the length of 11 000 s. Again, I reduce the dataset's dimensionality by projecting it on its dominant POD modes. I showcase the first six POD modes of the dataset in Appendix b, specifically in Fig. b.2. The truncation criterion stays the same as in the previous chapter, i.e., the cumulative energy of the modes is at least 90% of the total energy, leading to 512 modes.



(a) Symmetry about the x axis.

(b) Symmetry about the y axis.



(c) Rotational symmetry.

Figure 6.2: Symmetries of the full control problem.

6.3 KOOPMAN MPC

The success of Koopman MPC from the previous chapter begs the question of whether the same approach can be used to solve the full control problem. The answer is affirmative. However, there are some caveats, which I discuss in the following sections and subsections.

6.3.1 *Rank-One Control Input Constraint*

Now, let me explain what I referred to in the last chapter as inherent nonconvexity arising from the multiplication of the coil commands and the electrode commands in the body force term. Recall that the Lorentz Force term of the Navier–Stokes equations is

$$\mathbf{F} = \sigma \mathbf{E} \times \mathbf{B} = \sum_{i=1}^{n_{\text{el}}} \sum_{j=1}^{n_{\text{coil}}} \phi_i \psi_j \sigma \mathbf{E}_i \times \mathbf{B}_j. \tag{6.7}$$

The Lorentz Force term enters into the Navier Stokes equations in a linear fashion, but it is not the actual control input. The control inputs are the coil commands ψ, and the electrode commands ϕ. Whenever the coil or electrode commands are fixed, the term becomes linear in the other set of commands. However, the term is nonlinear when both sets of commands are allowed to vary. This breaks the assumption of linearity in the control inputs of the Koopman predictor.

Therefore, neither of the commands can be considered as control inputs. Instead, let me introduce a new virtual control variable, $\mathbf{u} \in \mathbb{R}^{n_{\text{el}} n_{\text{coil}}}$, which is the product of the coil and electrode commands, more precisely

$$\mathbf{u} = \operatorname{vec}\left(\mathbf{u}_\psi \mathbf{u}_\phi^\mathsf{T}\right), \tag{6.8}$$

where vec is the vectorization operator, which stacks the columns of a matrix into a vector. Then, the Lorentz Force term becomes

$$\mathbf{F} = \sum_{i=1}^{n_{\text{el}} n_{\text{coil}}} u_i \sigma \mathbf{E}_{(i-1) \operatorname{div} n_{\text{coil}}+1} \times \mathbf{B}_{(i-1) \bmod n_{\text{coil}}+1}, \tag{6.9}$$

where mod is the modulo operator and div is the integer division operator.

The control input $\mathbf{u}$ is now linear in the Lorentz Force term. However, it still must be decomposable into the coil and electrode commands, meaning it must satisfy Eq. (6.8) for some $\mathbf{u}_\phi$ and $\mathbf{u}_\psi$. This is a rank-one constraint on the control input, which is nonconvex.

6.3.2 *Koopman Predictor*

Based on the former argument, the Koopman predictor has the same form as in the previous chapter but with the virtual control input $\mathbf{u} = \operatorname{vec}\left(\mathbf{u}_\psi \mathbf{u}_\phi^\mathsf{T}\right)$ instead of the coil commands

$$\mathbf{z}_0 = \boldsymbol{\Psi}(\mathbf{x}_k), \tag{6.10a}$$
$$\mathbf{z}_{i+1} = \mathbf{A}\mathbf{z}_i + \mathbf{B}\mathbf{u}_{k+i}, \quad i = 0, \ldots, N_\text{p} - 1, \tag{6.10b}$$
$$\hat{\mathbf{y}}_{k+i} = \mathbf{C}\mathbf{z}_i, \qquad i = 1, \ldots, N_\text{p}. \tag{6.10c}$$

The Koopman state can be taken the same as in the previous chapter, i.e., the delay embedded POD projected fluid flow measurements

$$\boldsymbol{\Psi}(\mathbf{x}_k) = \begin{bmatrix} \tilde{\mathbf{y}}_{k-3}^\mathsf{T} & \tilde{\mathbf{y}}_{k-2}^\mathsf{T} & \tilde{\mathbf{y}}_{k-1}^\mathsf{T} & \tilde{\mathbf{y}}_{k}^\mathsf{T} & \mathbf{u}_{k-3}^\mathsf{T} & \mathbf{u}_{k-2}^\mathsf{T} & \mathbf{u}_{k-1}^\mathsf{T} \end{bmatrix}^\mathsf{T}. \tag{6.11}$$

This time the dimensionality of the Koopman state is 2060. The control input $\mathbf{u}$ is the rank-one control input as defined in Eq. (6.8). The Koopman predictor matrices $\mathbf{A}$, $\mathbf{B}$ are computed using the EDMD algorithm on the symmetrized dataset from Section 6.2. The matrix $\mathbf{A}$ has all the eigenvalues inside the unit circle, which means that the Koopman predictor is stable.

6.3.3 *Model Predictive Control*

The MPC problem is similar to the one in the previous chapter, but with the penalization of the rates of both types of commands and the rank-one constraint on the virtual control input $\mathbf{u}$. The full control MPC is therefore

$$\begin{aligned}
\underset{\{\mathbf{u}_{\phi,k}\}_{k=0}^{N_\mathrm{p}-1},\{\mathbf{u}_{\psi,k}\}_{k=0}^{N_\mathrm{p}-1}}{\text{minimize}} \quad & J\left(\{\mathbf{u}_{\phi,k}\}_{k=0}^{N_\mathrm{p}-1},\{\mathbf{u}_{\psi,k}\}_{k=0}^{N_\mathrm{p}-1},\{\mathbf{e}_k\}_{k=1}^{N_\mathrm{p}}\right), && \text{(6.12a)}\\
\text{subject to} \quad & \mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{B}\mathbf{u}_k, \quad k = 0,1,\ldots,N_\mathrm{p}-1, && \text{(6.12b)}\\
& \mathbf{e}_k = \mathbf{r}_k - \mathbf{C}\mathbf{z}_k, \quad k = 1,2,\ldots,N_\mathrm{p}, && \text{(6.12c)}\\
& \mathbf{u}_k = \operatorname{vec}\left(\mathbf{u}_{\phi,k}\mathbf{u}_{\psi,k}^\mathsf{T}\right), \quad k = 0,1,\ldots,N_\mathrm{p}-1, && \text{(6.12d)}\\
& \mathbf{u}_{\phi,\min} \le \mathbf{u}_{\phi,k} \le \mathbf{u}_{\phi,\max}, \quad k = 0,1,\ldots,N_\mathrm{p}-1, && \text{(6.12e)}\\
& \mathbf{u}_{\psi,\min} \le \mathbf{u}_{\psi,k} \le \mathbf{u}_{\psi,\max}, \quad k = 0,1,\ldots,N_\mathrm{p}-1, && \text{(6.12f)}\\
& \Delta\mathbf{u}_{\phi,k} = \mathbf{u}_{\phi,k} - \mathbf{u}_{\phi,k-1}, \quad k = 0,1,\ldots,N_\mathrm{p}, && \text{(6.12g)}\\
& \Delta\mathbf{u}_{\psi,k} = \mathbf{u}_{\psi,k} - \mathbf{u}_{\psi,k-1}, \quad k = 0,1,\ldots,N_\mathrm{p}, && \text{(6.12h)}\\
& \mathbf{u}_{\phi,-1} \text{ given}, && \text{(6.12i)}\\
& \mathbf{u}_{\psi,-1} \text{ given}, && \text{(6.12j)}\\
& \mathbf{z}_0 \text{ given by Eq. (6.11)}, && \text{(6.12k)}
\end{aligned}$$

where the criterion J is

$$\begin{aligned}
J\left(\{\mathbf{u}_{\phi,k}\}_{k=0}^{N_\mathrm{p}-1},\{\mathbf{u}_{\psi,k}\}_{k=0}^{N_\mathrm{p}-1},\{\mathbf{e}_k\}_{k=1}^{N_\mathrm{p}}\right) &= \frac{1}{2}\mathbf{e}_{N_\mathrm{p}}^\mathsf{T}\mathbf{Q}\mathbf{e}_{N_\mathrm{p}} + \frac{1}{2}\Delta\mathbf{u}_{\phi,0}^\mathsf{T}\mathbf{R}_\phi\Delta\mathbf{u}_{\phi,0} + \frac{1}{2}\Delta\mathbf{u}_{\psi,0}^\mathsf{T}\mathbf{R}_\psi\Delta\mathbf{u}_{\psi,0}\\
&\quad + \frac{1}{2}\sum_{k=1}^{N_\mathrm{p}-1}\mathbf{e}_k^\mathsf{T}\mathbf{Q}\mathbf{e}_k + \Delta\mathbf{u}_{\phi,k}^\mathsf{T}\mathbf{R}_\phi\Delta\mathbf{u}_{\phi,k} + \Delta\mathbf{u}_{\psi,k}^\mathsf{T}\mathbf{R}_\psi\Delta\mathbf{u}_{\psi,k},
\end{aligned} \tag{6.13}$$

where $\mathbf{Q}$ is the tracking error weight, $\mathbf{R}_\phi$ is the electrode commands rate weight, and $\mathbf{R}_\psi$ is the coil commands rate weight.

6.4 ALTERNATING MINIMIZATION SCHEME

To simplify the notation, I denote the sequence of the commands over the prediction horizon as

$$U_a = \{\mathbf{u}_{a,k}\}_{k=0}^{N_\mathrm{p}-1}, \quad a \in \{\phi,\psi\}. \tag{6.14}$$

The nonconvex rank-one constraint on the virtual control input $\mathbf{u}$ makes the optimization problem Eq. (6.12) hard to solve, even locally. However, the problem still retains a structure that can be exploited. Essentially, the problem becomes convex in U_ψ for a fixed U_ϕ, and vice versa. This suggests an alternating minimization scheme, where the optimization problem is solved iteratively by fixing one set of commands and optimizing the other set. Furthermore, when the single set of commands is fixed, the problem becomes a standard Koopman MPC

problem developed in the previous chapter, which can be solved extremely efficiently in the dense formulation using a specialized quadratic programming solver.

6.4.1 *Algorithm*

Let me now rephrase the Koopman MPC problem from the previous chapter, more specifically Eq. (5.27), in a slightly more general form, which will be used in the alternating minimization scheme. The Koopman MPC problem reads

$$\begin{aligned}
\underset{U_a}{\text{minimize}} \quad & \frac{1}{2}\mathbf{e}_{N_\mathrm{p}}^\mathsf{T}\mathbf{Q}\mathbf{e}_{N_\mathrm{p}} + \frac{1}{2}\Delta\mathbf{u}_{a,0}^\mathsf{T}\mathbf{R}\Delta\mathbf{u}_{a,0} + \frac{1}{2}\sum_{k=1}^{N_\mathrm{p}-1}\mathbf{e}_k^\mathsf{T}\mathbf{Q}\mathbf{e}_k + \Delta\mathbf{u}_{a,k}^\mathsf{T}\mathbf{R}_a\Delta\mathbf{u}_{a,k}, && \text{(6.15a)}\\
\text{subject to} \quad & \mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{B}_{a,k}\mathbf{u}_k, \quad k = 0, 1, \ldots, N_\mathrm{p}-1, && \text{(6.15b)}\\
& \mathbf{e}_k = \mathbf{r}_k - \mathbf{C}\mathbf{z}_k, \quad k = 1, 2, \ldots, N_\mathrm{p}, && \text{(6.15c)}\\
& \mathbf{u}_{a,\min} \leq \mathbf{u}_k \leq \mathbf{u}_{a,\max}, \quad k = 0, 1, \ldots, N_\mathrm{p}-1, && \text{(6.15d)}\\
& \Delta\mathbf{u}_{a,k} = \mathbf{u}_{a,k} - \mathbf{u}_{a,k-1}, \quad k = 0, 1, \ldots, N_\mathrm{p}-1, && \text{(6.15e)}\\
& \mathbf{z}_0 \text{ given by Eq. (6.11)}, && \text{(6.15f)}\\
& \mathbf{u}_{a,-1} \text{ given}, && \text{(6.15g)}
\end{aligned}$$

where $a \in \{\phi, \psi\}$. This formulation covers both cases of fixing the electrode commands and optimizing over the coil commands and vice versa. The only significant difference is the time-varying input matrix $\mathbf{B}_{a,k}$, which is either $\mathbf{B}_{\phi,k}$ or $\mathbf{B}_{\psi,k}$. These time-varying input matrices can be constructed from the other set of commands, and the constant input matrix $\mathbf{B}$ of the full Koopman predictor as

$$\mathbf{B}_{\psi,k} = \mathbf{B}\left(\mathbf{u}_{\phi,k} \otimes \mathbf{I}_{n_\mathrm{coil}\times n_\mathrm{coil}}\right), \quad k = 0, 1, \ldots, N_\mathrm{p}-1, \tag{6.16}$$

in case of fixing the electrode commands and optimization over the coil commands, and

$$\mathbf{B}_{\phi,k} = \mathbf{B}\left(\mathbf{I}_{n_\mathrm{el}\times n_\mathrm{el}} \otimes \mathbf{u}_{\psi,k}\right), \quad k = 0, 1, \ldots, N_\mathrm{p}-1, \tag{6.17}$$

in the other case. Moreover, $\otimes$ is the Kronecker product. By this construction, the minimizer of the problem in Eq. (6.15) is the same as the minimizer of the problem in Eq. (6.12) when the appropriate set of commands is fixed.

I show the pseudocode of the alternating minimization scheme in Section 6.4.1. The algorithm requires an initial guess for one set of commands. Here, I provide the algorithm for the case when the initial guess is provided for the electrode commands. The other case is similar.

For the termination criterion, I suggest using the change in the electrode commands, i.e., the RMSE of the change in the electrode commands between two consecutive iterations:

$$\sqrt{\frac{1}{N_\mathrm{p}}\sum_{k=0}^{N_\mathrm{p}-1}\left\|\mathbf{u}_{k,\phi}^{(n)} - \mathbf{u}_{k,\phi}^{(n-1)}\right\|_2^2} \leq \varepsilon, \tag{6.18}$$

where ε is the tolerance.

Algorithm 1 Alternating Minimization Scheme for solving Eq. (6.12)

Require: $\mathbf{z}_0$, $\mathbf{u}_{\phi,-1}$, $\mathbf{u}_{\psi,-1}$, initial guess: $U_\phi^{(0)}$
  $n \leftarrow 0$
  repeat
    $U_\psi^{(n+1)} \leftarrow$ Solve Eq. (6.15) fixing $U_\phi^{(n)}$
    $U_\phi^{(n+1)} \leftarrow$ Solve Eq. (6.15) fixing $U_\psi^{(n+1)}$
    $n \leftarrow n+1$
  until termination
  return $U_\phi^{(n)}$, $U_\psi^{(n)}$

6.4.2 *Convergence*

The optimization problem Eq. (6.15) is nonconvex, and thus the alternating minimization scheme, can at best find a local minimum. This begs the question of convergence. To shine some light on this question, let me give the following well-known result of Grippo and Sciandrone [2000]:

Result 1 *Consider the following optimization problem*

$$\underset{\mathbf{x}\in X,\mathbf{y}\in Y}{\text{minimize}}\, f(\mathbf{x},\mathbf{y}), \tag{6.19}$$

where $f(\mathbf{x},\mathbf{y})$ is a continuously differentiable function over the set $X\times Y$, where X and Y are closed and convex sets. Assuming that every problem of the alternating minimization scheme

$$\mathbf{x}^{(n+1)} = \underset{\mathbf{x}\in X}{\operatorname{argmin}}\, f(\mathbf{x},\mathbf{y}^{(n)}), \quad \mathbf{y}^{(n+1)} = \underset{\mathbf{y}\in Y}{\operatorname{argmin}}\, f(\mathbf{x}^{(n+1)},\mathbf{y}), \tag{6.20}$$

has a solution, then every limit point of the sequence $\left\{\left(\mathbf{x}^{(n)},\mathbf{y}^{(n)}\right)\right\}$ is a stationary point of the original problem.

The result suggests that the alternating minimization scheme will converge to a stationary point of the original problem.

6.5 RESULTS

In this section, I present the performance of the developed control algorithm. Similarly, to the previous chapter, I construct a reference flow at certain points in the measured region. The penalty matrices are the same in all examples, i.e.,

$$\mathbf{Q} = \frac{10^4}{N_\mathcal{R}}\mathbf{I}_{2N_\mathcal{R}\times 2N_\mathcal{R}}, \quad \mathbf{R}_\phi = \frac{1}{10}\mathbf{I}_{n_\text{el}\times n_\text{el}}, \quad \mathbf{R}_\psi = \mathbf{I}_{n_\text{coil}\times n_\text{coil}}, \tag{6.21}$$

where $N_\mathcal{R}$ is the number of points in the measured region, where the reference flow is prescribed. The prediction horizon is $N_\text{p} = 10$, or 5 s. In the first timestep, the alternating

minimization scheme is initialized with random electrode commands sampled from a uniform distribution within the input constraints. The electrode commands from the previous timestep are used as the initial guess in the following timesteps.

Being able to set both the coil and electrode commands opens up more possibilities for achievable flow shapes. E. g., the flows do not have to be just in the x or y direction but can be rotated or have a more complex shape. I showcase two such flows: two rotated vortices and a diagonal flow. The simulations are started from the same initial condition, i. e., zero initial velocity and pressure. The reference is set at the time $t = 0$ and is kept constant throughout the simulation. The length of each simulation is 90 s.

6.5.1 *Two Rotated Vortices*

I start with a reference of two vortices rotated by 45°, which are together with the flow field at the final time of the simulation shown in Fig. 6.3a, and the evolution of the tracking error is shown in Fig. 6.3b. As the figures show, the control algorithm successfully creates the two vortices. The final tracking error achieved is 32 %. Note that the reference is not really physically realizable. Therefore the error is not expected to be zero.

Moreover, I present the statistics[1] of the MPC solution time and the number of iterations of the alternating minimization scheme in Table 6.1. The solution time is well below the 500 ms of the control timestep, which is a good sign. The number of iterations is also low, which suggests that the alternating minimization scheme converges quickly.

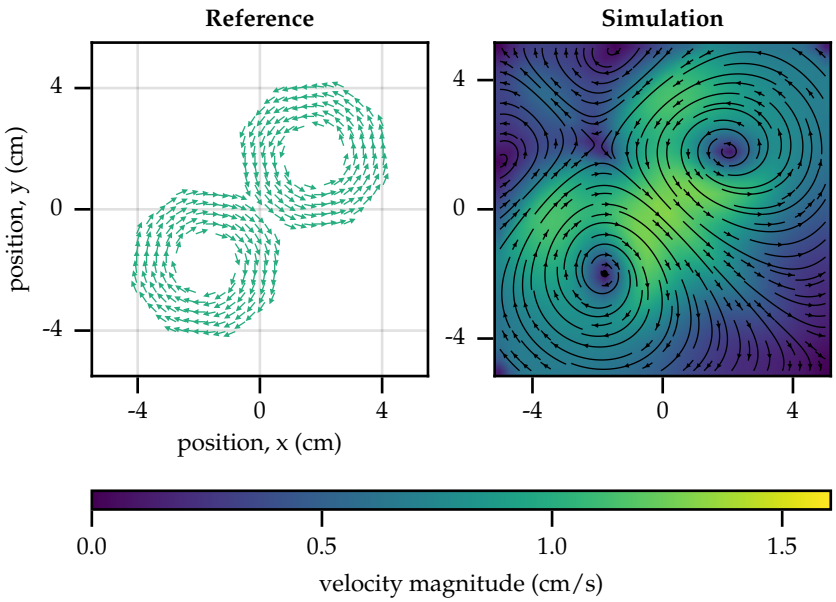
Table 6.1: Statistics of the MPC solution time and iterations of the alternating minimization scheme for the two rotated vortices.

Metric	Average	Maximum	Minimum
Solution time	17 ms	36.6 ms	8.7 ms
Iterations	3.4	7	1

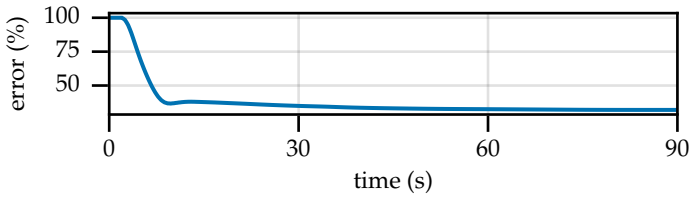
6.5.2 *Diagonal Flow*

The second example is a diagonal flow, which is shown in Fig. 6.4a, and the evolution of the tracking error is shown in Fig. 6.4b. The control algorithm shapes the flow into a diagonal shape. However the tracking error is higher than in the previous example, reaching 70 % at the end of the simulation. It appears there is a large discrepancy between the magnitude of the reference and the achieved flow, which is about twice as large in the maximum. This may be a sign of a local minimum, which the alternating minimization scheme has converged to. However, I must again note that the reference is not physically realizable, and therefore, the error is not expected to be zero.

1 The computation times are measured on the MacBook Pro 14" (2023) with the M2 Max chip.



(a) Reference and the velocity field at the final time.

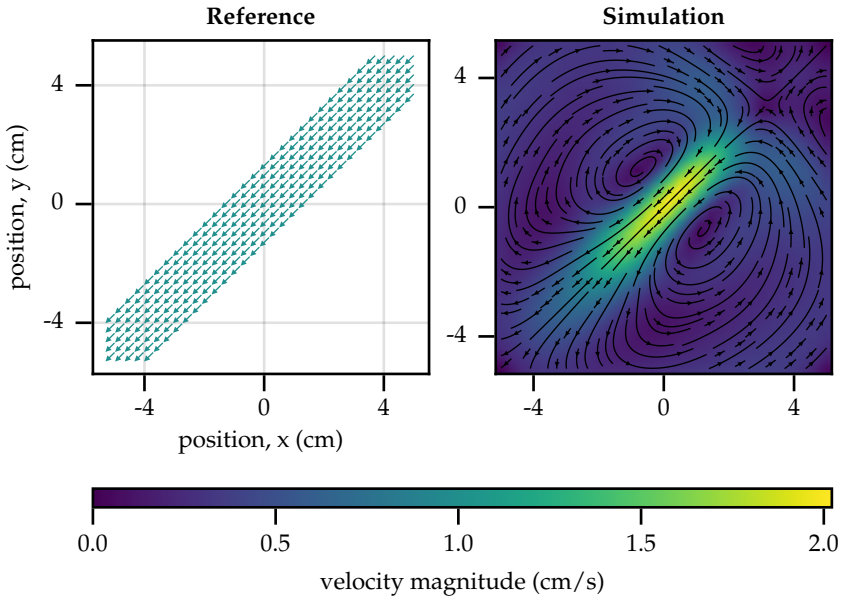


(b) Evolution of the reference tracking error.

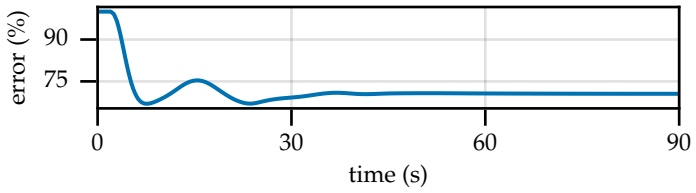
Figure 6.3: Two rotated vortices.

Table 6.2: Statistics of the MPC solution time and iterations of the alternating minimization scheme for the diagonal flow.

Metric	Average	Maximum	Minimum
Solution time	13.3 ms	35.1 ms	3.6 ms
Iterations	3	8	1



(a) Diagonal flow.



(b) Evolution of the reference tracking error.

Figure 6.4: Diagonal flow.

6.6 NOTES ON ONLINE LEARNING

The official thesis assignment includes the consideration of incorporating online learning into the control algorithm. After consulting with my supervisor, we have decided to postpone the implementation of online learning to future work due to time constraints.

However, I can share my thoughts on how online learning could be integrated into the control algorithm. Online learning involves updating the Koopman matrices, $\mathbf{A}$ and $\mathbf{B}$, in real time as data is collected during the execution of the control algorithm. I suggest using a minibatch approach for the updates, where the matrices are revised every 100 or so samples using, for example, the ADAM optimizer. The challenge is that the dense formulation of the Koopman MPC must be entirely recomputed with each update of the $\mathbf{A}$ matrix. This process takes several seconds, making real-time applications infeasible. A potential solution is to employ a parallel processing environment, where the predictor is updated, and the dense formulation of the Koopman MPC problem is recomputed in the background while the control algorithm continues to run.

7

EXPERIMENTAL VALIDATION OF DESIGNED CONTROL ALGORITHM

In this chapter, I detail the experiments performed to validate the control algorithm designed for shaping MHD flows. Specifically, the control algorithm from Chapter 6 is applied to shape the flow within the experimental setup outlined in Chapter 2. Additionally, the complete control loop diagram is provided in Fig. 7.1, and the parameters of the experiments are listed in Table 7.1.

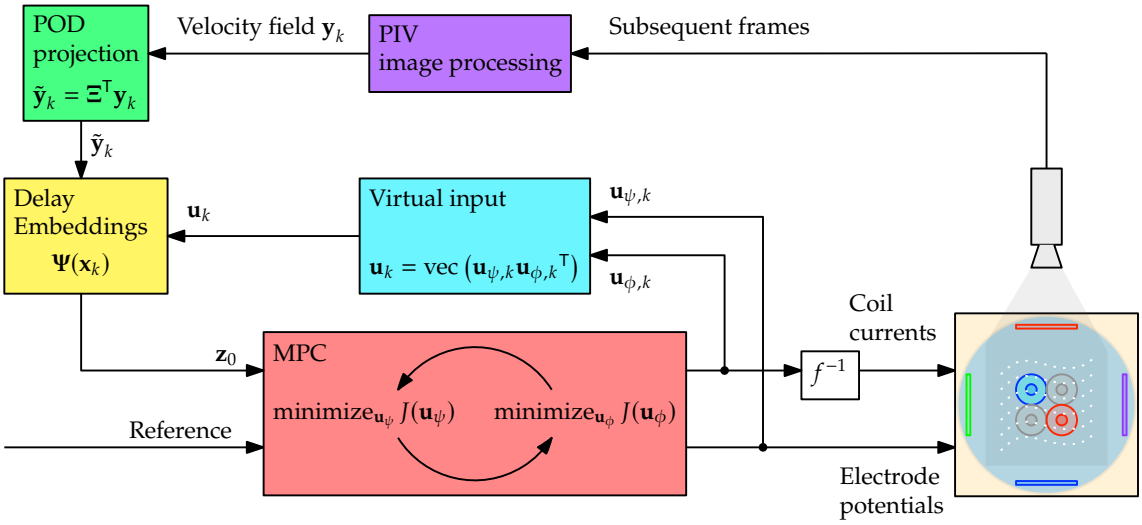


Figure 7.1: Signal flow diagram of the control loop. Function f^{-1} maps the coil commands to the coil currents and is the inverse of f introduced in Eq. (3.12).

Table 7.1: Parameters of the experiments.

Parameter	Value
Seeding	85 µm PMMA particles
PIV resolution	32×32 (10 cm $\times$ 10 cm)
Electrolyte	0.6 % sulfuric acid in water
Fluid height	8 mm

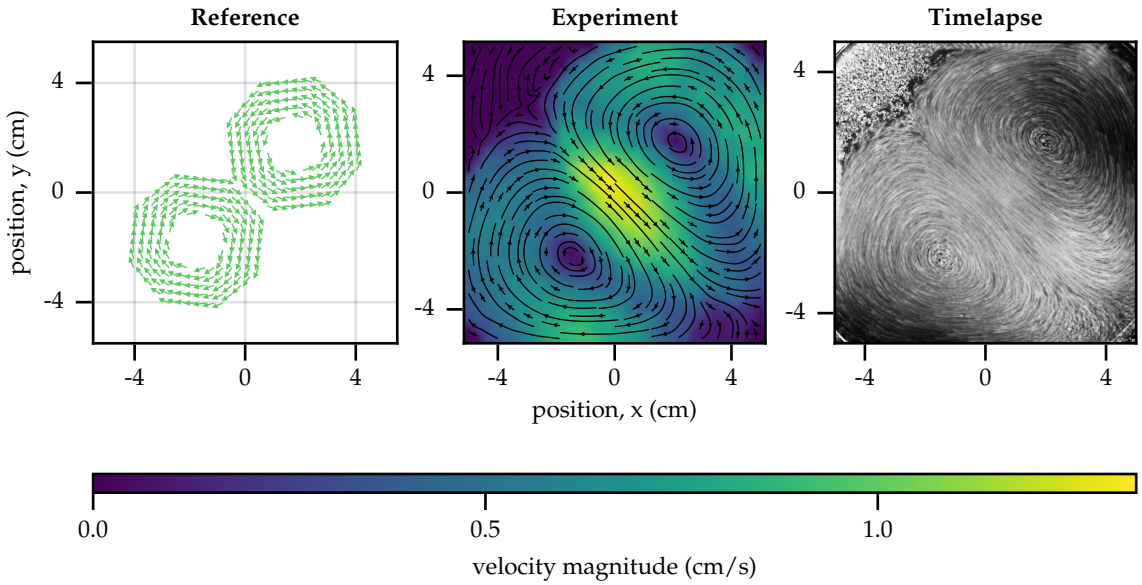
The goal of each experiment is to shape the flow into a desired reference flow. Each experiment is conducted in the same way: the flow is at rest initially, and the controller is turned on for the duration of the experiment, which is 87 s. For each experiment, I provide the prescribed reference flow, the flow at the end of the experiment, and the long exposure

shot of the flow. Moreover, for qualitative evaluation, I use the normed tracking error from Eq. (5.29).

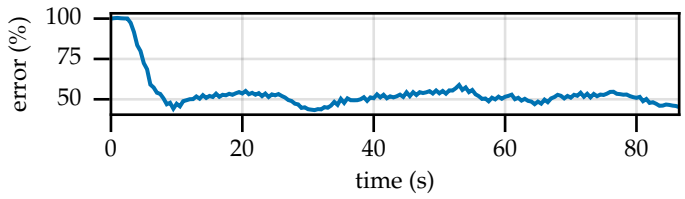
I provide the results of four experiments in the following sections. The first two experiments use the same reference flows as in the previous chapter so the reader can compare the results of the simulation and the experiment. The last two experiments use different reference flows to show the controller's additional capabilities.

7.1 EXPERIMENT 1: TWO VORTICES

The first experiment is shaping the flow into two vortices. I present the results of the experiment in Fig. 7.2. As the figures show, the controller is able to shape the flow into the desired shape, and the tracking error converges to about 45 %. Compare that to the simulation results in Fig. 6.3, where the tracking error converges to about 32 %.



(a) Reference (left), velocity field (middle), and long exposure shot of the flow at the end of the experiment (right).

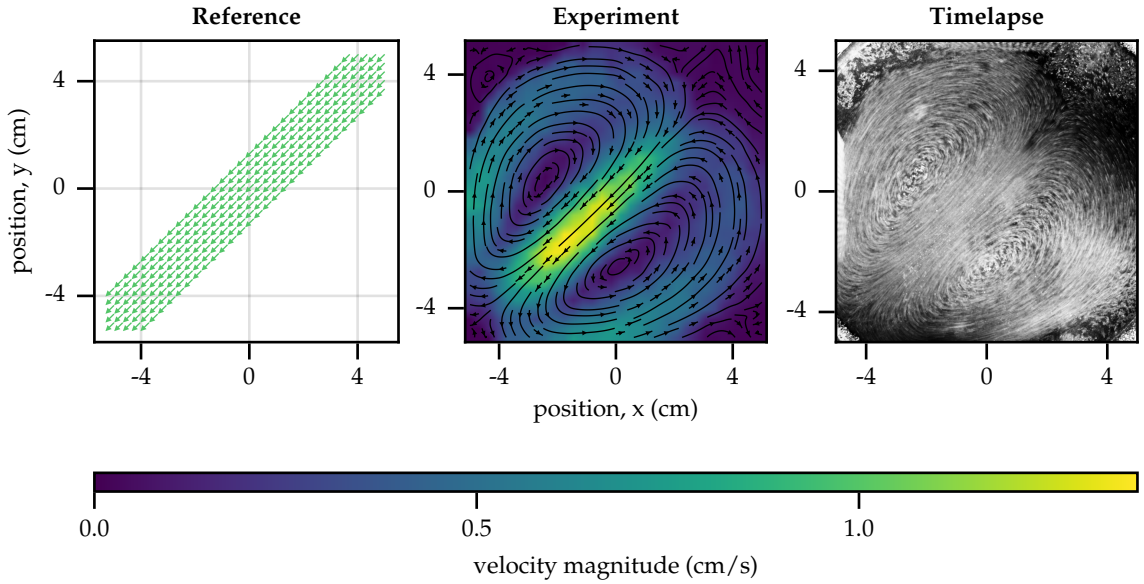


(b) Evolution of the tracking error.

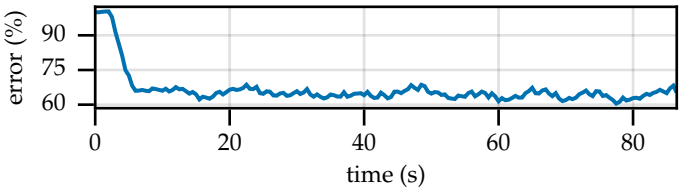
Figure 7.2: Experiment 1: Two Vortices.

7.2 EXPERIMENT 2: DIAGONAL FLOW

The second experiment consists of creating a diagonal flow. I show the experiment results in Fig. 7.3. The controller is able to shape the flow into the desired shape, and the tracking error converges to about 65 % of the initial error. Compare that to the simulation results in Fig. 6.4, where the tracking error reaches 70 %.



(a) Reference (left), velocity field (middle), and long exposure shot of the flow at the end of the experiment (right).

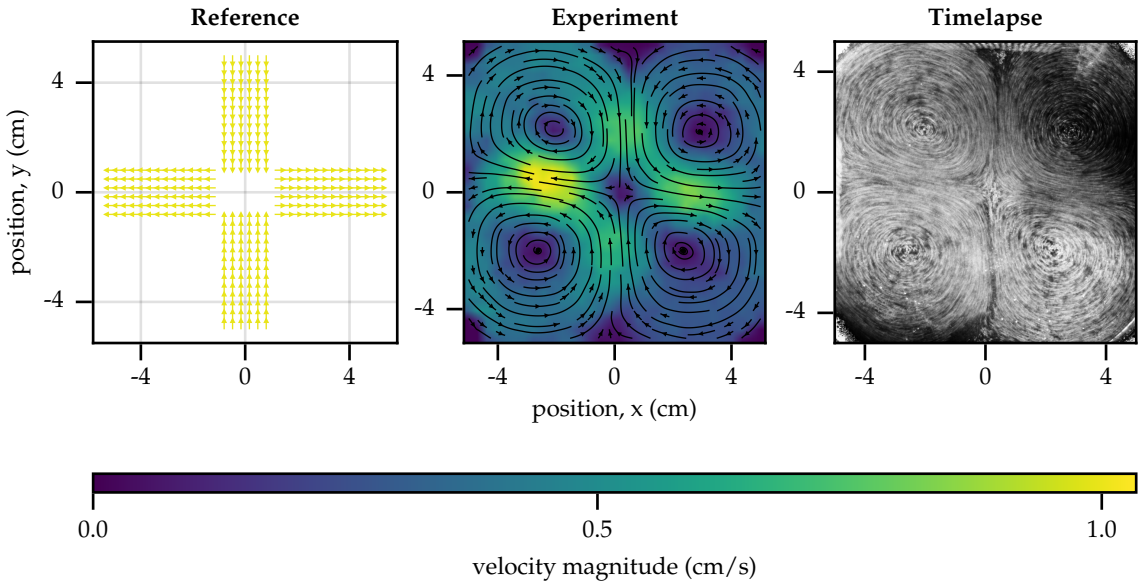


(b) Evolution of the tracking error.

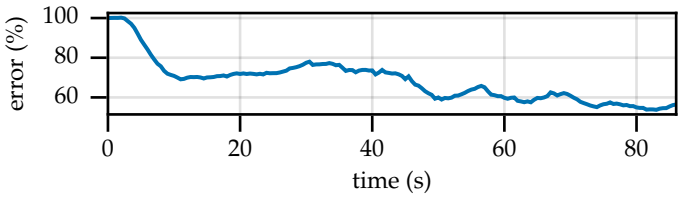
Figure 7.3: Experiment 2: Diagonal Flow.

7.3 EXPERIMENT 3: MULTIPLE DIRECTIONAL FLOW

The third experiment consists of creating flows in multiple directions, which in turn lead to four vortices. I show the results of the experiment in Fig. 7.4. The controller, again, shapes the flow close to the desired shape and decreases the error to about 55 % of the initial error at the end of the experiment.



(a) Reference (left), velocity field (middle), and long exposure shot of the flow at the end of the experiment (right).

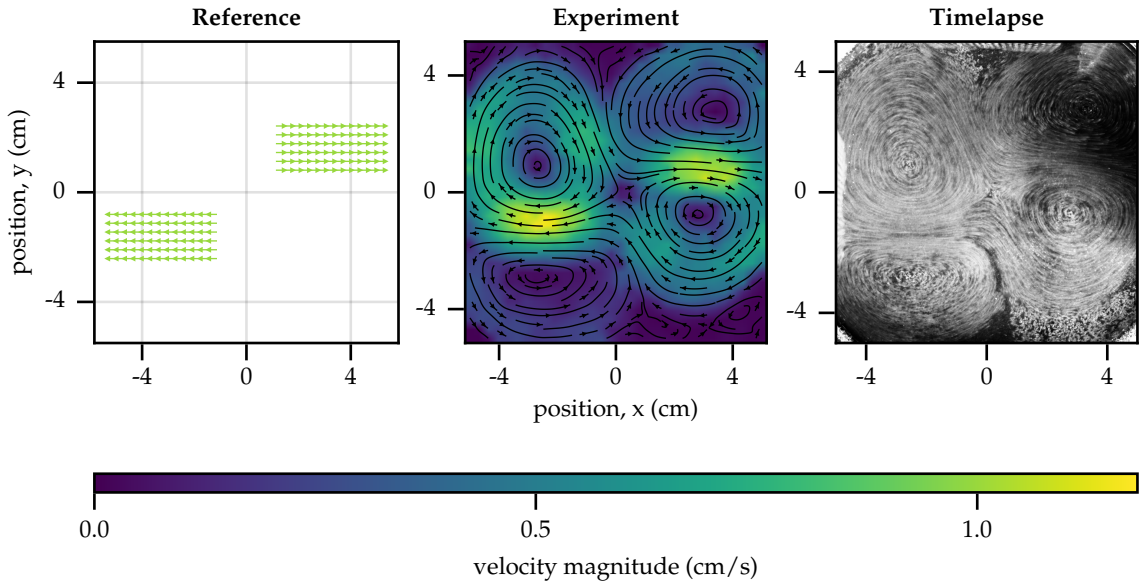


(b) Evolution of the tracking error..

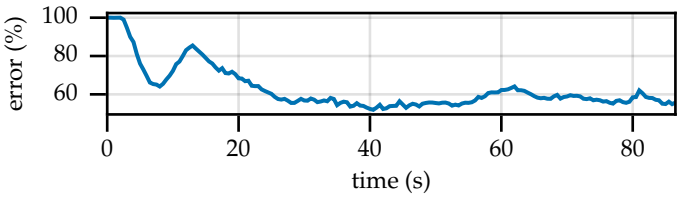
Figure 7.4: Experiment 3: Multiple Directional Flow.

7.4 EXPERIMENT 4: STRIPED FLOW

The last experiment consists of creating a flow of two stripes offseted from the origin. I show the results of the experiment in Fig. 7.5. The controller successfully shapes the flow into the desired shape and decreases the tracking error to about 56 % of the initial error at the end of the experiment.



(a) Reference (left), velocity field (middle), and long exposure shot of the flow at the end of the experiment (right).



(b) Evolution of the tracking error..

Figure 7.5: Experiment 4: Striped Flow

7.5 DISCUSSION

At the end of the experiments, the fields were close to the desired shapes, at least in terms of fluid patterns and flow directions. The tracking error decreased throughout the experiments, indicating that the controller effectively guided the flow toward the desired shape. However, there are indications that the controller could be improved. Oscillations were present in all experiments, and in the final experiment, the error significantly increased at the beginning. This is likely due to two main factors: significant noise in the measurements and the predictor being trained on simulation data that do not perfectly match the experimental data. Additionally, further error reduction might be possible, but this is difficult to determine without a comprehensive reachability analysis of the system, which is beyond the scope of this thesis.

8

CONCLUSIONS & OUTLOOK

In this thesis, I aimed to develop a data-driven feedback control algorithm for shaping MHD flows. This algorithm was to be validated using an experimental setup comprising a tank with a water-based electrolyte, electrodes, and coils for actuation and a PIV system for flow measurement. I approached this task in the following steps:

In Chapter 2, I described the experimental setup, outlined the control objectives, and provided a brief mathematical description of the MHD flow.

In Chapter 3, I presented the simulation environment of the MHD flow in the experimental setup that I developed during my thesis and the previous semester. The simulation environment solves the incompressible Navier-Stokes equations in three dimensions using the Finite Element Method with an Incremental Pressure Correction Scheme. Additionally, I validated the simulation environment by comparing its results with those from experiments conducted using the setup. The simulation results showed good agreement with the experimental results. Furthermore, I validated the simulation environment using the commercial computational fluid dynamics software COMSOL Multiphysics. My simulation environment was able to conduct the simulation approximately ten times faster.

In Chapter 4, I introduced the methods used for constructing the control algorithm, including MPC and techniques for building surrogate models.

In Chapter 5, I designed two control algorithms for shaping the flow using only the coils while fixing the electrodes at a constant potential. The first algorithm was a DeepMPC, which utilized a deep neural network to predict future states of the flow. The second algorithm was a Koopman MPC, which used a linear system approximating the Koopman operator to predict future states. These models were constructed from data generated by the simulation environment. Both controllers successfully shaped the flow into the desired configurations and exhibited similar performance in terms of tracking error. However, the Koopman MPC outperformed the DeepMPC in computational efficiency by approximately 1000 times.

In Chapter 6, I developed a control algorithm for shaping the flow using both the coils and the electrodes. This algorithm was based on the Koopman MPC algorithm from Chapter 5, but extended to include the electrodes as control inputs, resulting in a nonconvex MPC problem. However, the problem had an exploitable structure. I solved the problem by alternately solving the MPC problems from the previous chapter, once with fixed electrode potentials and once with fixed coil currents. This approach rapidly converged to a stationary point of the nonconvex problem, enabling the control algorithm to shape the flow into the desired patterns.

Lastly, in Chapter 7, I conducted experiments on the experimental setup to validate the control algorithm developed in Chapter 6. The control algorithm successfully shaped the flow close to the desired configurations in all experiments, albeit with some discrepancies.

8.1 FUTURE WORK

Firstly, designing a more effective initialization scheme for the MPC problem beyond the current random initialization would be highly beneficial. A systematic approach to initialization could significantly improve the algorithm's efficiency and reliability. Additionally, training the predictor directly on experimental data could enhance the control algorithm's performance. However, this approach presents significant challenges. It requires a substantial amount of data, and the data from the experiments are highly noisy. DMD-based methods are particularly sensitive to noise, complicating the training process. Incorporating the system's symmetries directly into the predictor could potentially reduce data requirements and mitigate some of these issues.

Exploring other tasks beyond flow shaping presents another promising direction. For instance, optimizing fluid mixing in the tank would involve developing a new cost function, such as maximizing the flow's vorticity. Alternatively, the task of transporting objects within the tank could be considered. This is exceptionally challenging due to the disturbance of flow measurements caused by the object's presence, necessitating a highly adaptive control algorithm or a robust observer.

Expanding the experimental platform is another avenue for future work. Integrating more electrodes and coils would enable the generation of more complex flow patterns. Furthermore, if the electrodes were more localized, it could open up opportunities for the development and exploration of distributed control algorithms. This would allow for more precise and varied control over the fluid dynamics within the tank.

APPENDIX

a

DENSE FORMULATION OF TRACKING MPC

In this appendix, I present the dense formulation of the tracking MPC algorithm. It was left out in the main text for brevity and clarity of the exposition.

The minimizer of problems in Eq. (5.27) and Eq. (6.15) can be efficiently computed using the dense formulation of the MPC algorithm. This dense formulation reads

$$\underset{\mathbf{U}}{\text{minimize}} \quad \frac{1}{2}\mathbf{U}^\mathsf{T}\mathbf{H}\mathbf{U} + \mathbf{f}^\mathsf{T}\mathbf{U} \tag{a.1a}$$

$$\text{subject to} \quad \mathbf{U}_{\min} \leq \mathbf{U} \leq \mathbf{U}_{\max}, \tag{a.1b}$$

$$\tag{a.1c}$$

where $\mathbf{U} = \begin{bmatrix}\mathbf{u}_0^\mathsf{T} & \mathbf{u}_1^\mathsf{T} & \dots & \mathbf{u}_{N_\mathrm{p}-1}^\mathsf{T}\end{bmatrix}^\mathsf{T}$ is the vector of control inputs, $\mathbf{H}$ is the Hessian matrix, $\mathbf{f}$ is the gradient vector, and $\mathbf{U}_{\min}$ and $\mathbf{U}_{\max}$ are the bounds on the control inputs. The Hessian matrix $\mathbf{H}$ and the gradient vector $\mathbf{f}$ can be expressed as

$$\mathbf{H} = \bar{\mathbf{A}}^\mathsf{T}\bar{\mathbf{C}}^\mathsf{T}\bar{\mathbf{Q}}\bar{\mathbf{C}}\bar{\mathbf{A}} + \bar{\mathbf{R}}, \quad \mathbf{f}^\mathsf{T} = \begin{bmatrix}\mathbf{z}_0^\mathsf{T}\mathbf{G} \\ -\mathbf{u}_{-1}^\mathsf{T}\hat{\mathbf{R}} \\ -\bar{\mathbf{r}}^\mathsf{T}\mathbf{F}\end{bmatrix}, \tag{a.2}$$

where

$$\mathbf{F} = \bar{\mathbf{Q}}\bar{\mathbf{C}}\bar{\mathbf{A}}, \quad \mathbf{G} = \hat{\mathbf{A}}^\mathsf{T}\bar{\mathbf{C}}^\mathsf{T}\bar{\mathbf{Q}}\bar{\mathbf{C}}\bar{\mathbf{A}}, \tag{a.3a}$$

$$\hat{\mathbf{A}} = \begin{bmatrix}\mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{N_\mathrm{p}}\end{bmatrix}, \quad \hat{\mathbf{R}} = \begin{bmatrix}\mathbf{R}^\mathsf{T} \\ \mathbf{0} \\ \vdots \\ \mathbf{0}\end{bmatrix}^\mathsf{T}, \quad \bar{\mathbf{r}} = \begin{bmatrix}\mathbf{r}_0 \\ \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_{N_\mathrm{p}}\end{bmatrix}, \tag{a.3b}$$

$$\bar{\mathbf{Q}} = \begin{bmatrix}\mathbf{Q} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q}\end{bmatrix}, \quad \bar{\mathbf{R}} = \begin{bmatrix}-\mathbf{R} & 2\mathbf{R} & -\mathbf{R} & \cdots & \mathbf{0} \\ \mathbf{0} & -\mathbf{R} & 2\mathbf{R} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{R} & \mathbf{R}\end{bmatrix}. \tag{a.3c}$$

$$\bar{\mathbf{C}} = \begin{bmatrix}\mathbf{C} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{C} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}\end{bmatrix}, \quad \bar{\mathbf{A}} = \begin{bmatrix}\mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}\mathbf{B} & \mathbf{B} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{N_\mathrm{p}-1}\mathbf{B} & \mathbf{A}^{N_\mathrm{p}-2}\mathbf{B} & \cdots & \mathbf{B}\end{bmatrix}. \tag{a.3d}$$

b

VISUALIZATION OF POD MODES

In this appendix, I visualize the first 6 dominant POD modes of the dataset from Chapter 5 and the symmetrized dataset from Chapter 6 in Fig. b.1 and Fig. b.2, respectively. These figures were omitted from the main text to maintain conciseness and readability.

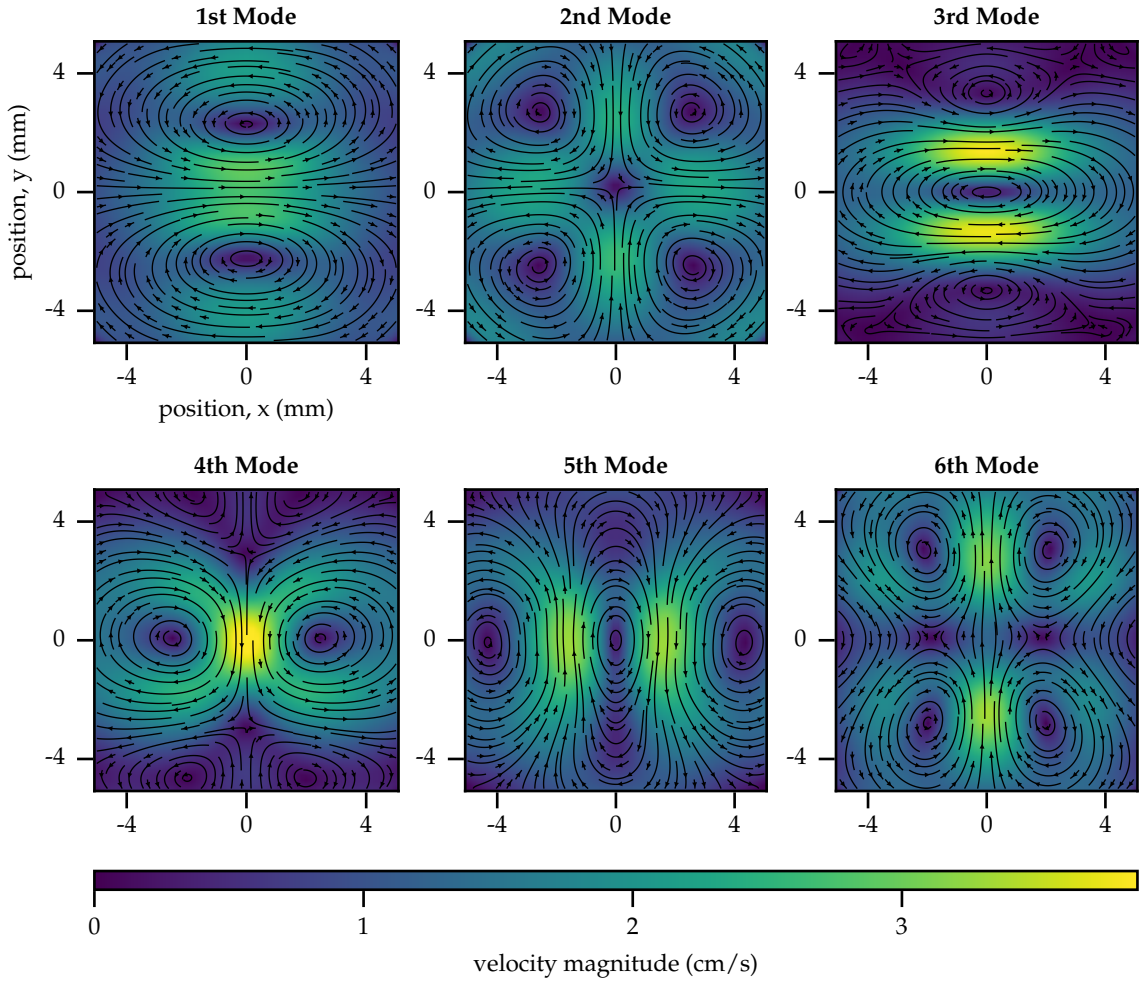


Figure b.1: First six POD modes of the dataset from Chapter 5.

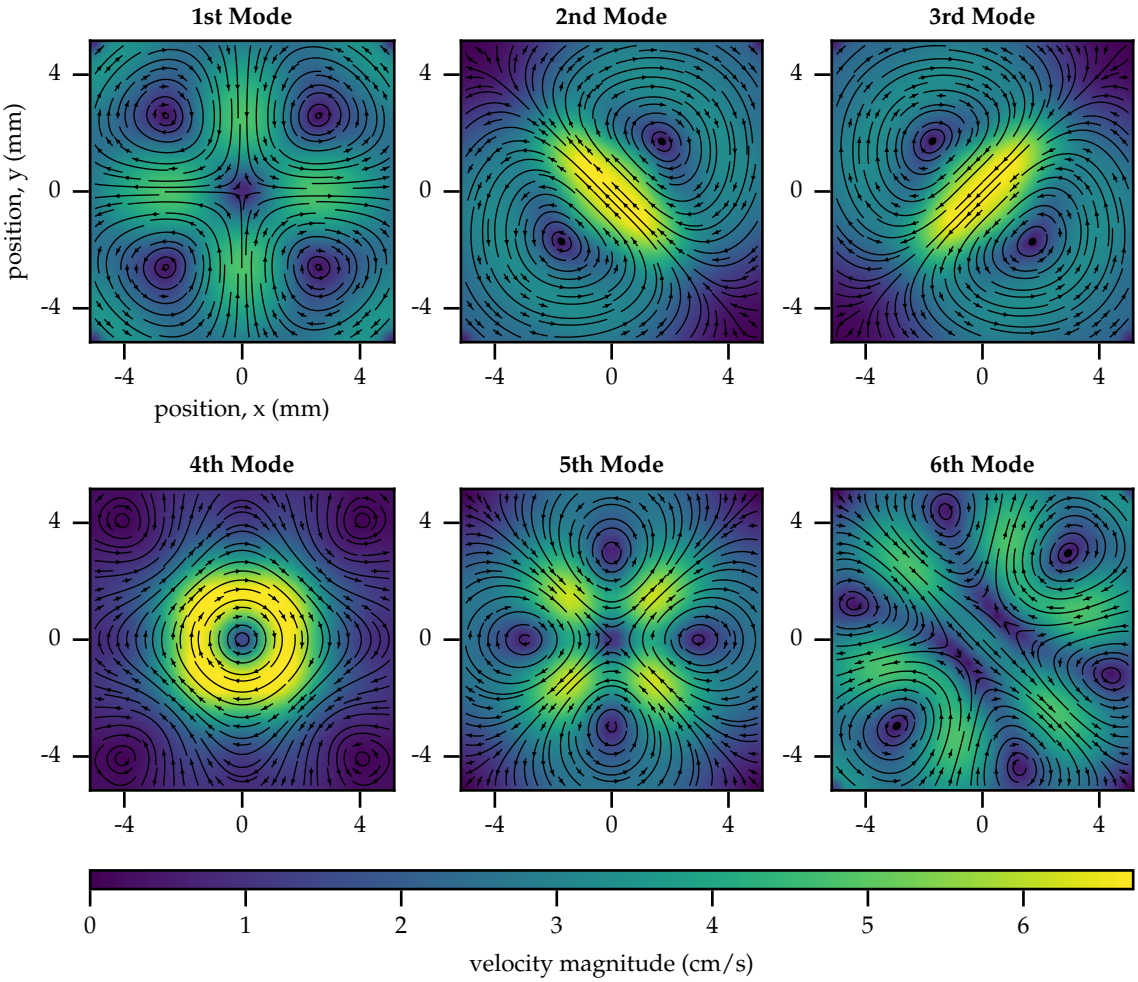


Figure b.2: First six POD modes of the symmetrized dataset from Chapter 6.

BIBLIOGRAPHY

H. Arbabi, M. Korda, and I. Mezić. A Data-Driven Koopman Model Predictive Control Framework for Nonlinear Partial Differential Equations. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6409–6414, Dec. 2018. doi: 10.1109/CDC.2018.8619720.

T. Baumeister, S. L. Brunton, and J. N. Kutz. Deep learning and model predictive control for self-tuning mode-locked lasers. *JOSA B*, 35(3):617–626, Mar. 2018. ISSN 1520-8540. doi: 10.1364/JOSAB.35.000617.

M. Bergmann, L. Cordier, and J.-P. Brancher. Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model. *Physics of Fluids*, 17(9): 097101, Aug. 2005. ISSN 1070-6631. doi: 10.1063/1.2033624.

K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz. Deep model predictive flow control with limited sensor data and online learning. *Theoretical and Computational Fluid Dynamics*, 34(4):577–591, Aug. 2020. ISSN 1432-2250. doi: 10.1007/s00162-020-00520-4.

F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, Cambridge, United Kingdom ; New York, NY, USA, 1st edition edition, July 2017. ISBN 978-1-107-01688-0.

T. Chen, Z. Ren, G. Lin, Z. Wu, and B.-L. Ye. Real-time computational optimal control of an MHD flow system with parameter uncertainty quantification. *Journal of the Franklin Institute*, 357(5):2830–2850, Mar. 2020. ISSN 0016-0032. doi: 10.1016/j.jfranklin.2019.12.013.

H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*. Oxford University Press, Oxford, UNITED KINGDOM, 2005. ISBN 978-0-19-152378-6.

F. Fan, B. Yi, D. Rye, G. Shi, and I. R. Manchester. Learning Stable Koopman Embeddings. In *2022 American Control Conference (ACC)*, pages 2742–2747, June 2022. doi: 10.23919/ACC53348.2022.9867865.

S. Fresca, L. Dede', and A. Manzoni. A Comprehensive Deep Learning-Based Approach to Reduced Order Modeling of Nonlinear Time-Dependent Parametrized PDEs. *Journal of Scientific Computing*, 87(2):61, Apr. 2021. ISSN 1573-7691. doi: 10.1007/s10915-021-01462-7.

L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss–Seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, Apr. 2000. ISSN 0167-6377. doi: 10.1016/S0167-6377(99)00074-7.

J. L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44):6011–6045, Sept. 2006. ISSN 0045-7825. doi: 10.1016/j.cma.2005.10.010.

M. Hinze and S. Volkwein. Proper Orthogonal Decomposition Surrogate Models for Nonlinear Dynamical Systems: Error Estimates and Suboptimal Control. In P. Benner, D. C. Sorensen, and V. Mehrmann, editors, *Dimension Reduction of Large-Scale Systems*, pages 261–306, Berlin, Heidelberg, 2005. Springer. ISBN 978-3-540-27909-9. doi: 10.1007/3-540-27909-1_10.

B. O. Koopman. Hamiltonian Systems and Transformation in Hilbert Space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, May 1931. doi: 10.1073/pnas.17.5.315.

B. O. Koopman and J. v. Neumann. Dynamical Systems of Continuous Spectra. *Proceedings of the National Academy of Sciences*, 18(3):255–263, Mar. 1932. doi: 10.1073/pnas.18.3.255.

M. Korda and I. Mezić. On Convergence of Extended Dynamic Mode Decomposition to the Koopman Operator. *Journal of Nonlinear Science*, 28(2):687–710, Apr. 2018a. ISSN 1432-1467. doi: 10.1007/s00332-017-9423-0.

M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, July 2018b. ISSN 0005-1098. doi: 10.1016/j.automatica.2018.03.046.

M. Korda and I. Mezić. Optimal Construction of Koopman Eigenfunctions for Prediction and Control. *IEEE Transactions on Automatic Control*, 65(12):5114–5129, Dec. 2020. ISSN 1558-2523. doi: 10.1109/TAC.2020.2978039.

I. Lenz, R. Knepper, and A. Saxena. DeepMPC: Learning Deep Latent Features for Model Predictive Control. In *Robotics: Science and Systems XI*. Robotics: Science and Systems Foundation, July 2015. ISBN 978-0-9923747-1-6. doi: 10.15607/RSS.2015.XI.012.

Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba. Learning Compositional Koopman Operators for Model-Based Control. In *Eighth International Conference on Learning Representations*, Apr. 2020.

J.-B. Lugagne, C. M. Blassick, and M. J. Dunlop. Deep model predictive control of gene expression in thousands of single cells. *Nature Communications*, 15(1):2148, Mar. 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-46361-1.

J. Lumley. The Structure of Inhomogeneous Turbulence. *Yaglom, A.M. and Tartarsky*, pages 166–177, 1967.

G. Mamakoukas, M. Castano, X. Tan, and T. Murphey. Local Koopman Operators for Data-Driven Control of Robotic Systems. In *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. ISBN 978-0-9923747-5-4. doi: 10.15607/RSS.2019.XV.054.

L. Mandić, N. Mišković, and D. Nad. Learning globally linear predictors using deep Koopman embeddings with application to marine vehicles. *IFAC-PapersOnLine*, 56(2):11596–11601, Jan. 2023. ISSN 2405-8963. doi: 10.1016/j.ifacol.2023.10.464.

J. Matouš, A. Kollarčík, M. Gurtner, T. Michálek, and Z. Hurák. Optimization-based Feedback Manipulation Through an Array of Ultrasonic Transducers. *IFAC-PapersOnLine*, 52(15): 483–488, Jan. 2019. ISSN 2405-8963. doi: 10.1016/j.ifacol.2019.11.722.

J. Morton, A. Jameson, M. J. Kochenderfer, and F. Witherden. Deep Dynamical Modeling and Control of Unsteady Fluid Flows. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

E. J. Parish and K. T. Carlberg. Time-series machine-learning error models for approximate solutions to parameterized dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 365:112990, June 2020. ISSN 0045-7825. doi: 10.1016/j.cma.2020.112990.

S. Pechacek. *Control of Magnetohydrodynamic Flow: Integration of Electronics and Measurement*. Bachelor's thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, 2024.

S. Peitz and K. Bieker. On the universal transformation of data-driven models to control systems. *Automatica*, 149:110840, Mar. 2023. ISSN 0005-1098. doi: 10.1016/j.automatica.2022.110840.

S. Peitz and S. Klus. Koopman operator-based model reduction for switched-system control of PDEs. *Automatica*, 106:184–191, Aug. 2019. ISSN 0005-1098. doi: 10.1016/j.automatica.2019.05.016.

S. Peitz, S. Ober-Blöbaum, and M. Dellnitz. Multiobjective Optimal Control Methods for the Navier-Stokes Equations Using Reduced Order Modeling. *Acta Applicandae Mathematicae*, 161(1):171–199, June 2019. ISSN 1572-9036. doi: 10.1007/s10440-018-0209-7.

S. Peitz, S. E. Otto, and C. W. Rowley. Data-Driven Model Predictive Control using Interpolated Koopman Generators. *SIAM Journal on Applied Dynamical Systems*, 19(3):2162–2193, Jan. 2020. doi: 10.1137/20M1325678.

J. L. Proctor, S. L. Brunton, and J. N. Kutz. Dynamic Mode Decomposition with Control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, Jan. 2016. doi: 10.1137/15M1013857.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, Feb. 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045.

F. Regazzoni, L. Dedè, and A. Quarteroni. Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics*, 397:108852, Nov. 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.07.050.

Z. Ren, Z. Zhao, Z. Wu, and T. Chen. Dynamic Optimal Control of a One-Dimensional Magnetohydrodynamic System With Bilinear Actuation. *IEEE Access*, 6:24464–24474, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2830768.

Z. Ren, C. Xu, Z. Wu, and X. Liu. Optimal tracking control of flow velocity in a one-dimensional magnetohydrodynamic flow. *Engineering Optimization*, 51(1):1–21, Jan. 2019. ISSN 0305-215X. doi: 10.1080/0305215X.2018.1426759.

P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, Aug. 2010. ISSN 1469-7645, 0022-1120. doi: 10.1017/S0022112010001217.

E. Schuster, L. Luo, and M. Krstić. MHD channel flow control in 2D: Mixing enhancement by boundary feedback. *Automatica*, 44(10):2498–2507, Oct. 2008. ISSN 0005-1098. doi: 10.1016/j.automatica.2008.02.018.

L. Sirovich. Turbulence and the dynamics of coherent structures. I. Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987. ISSN 0033-569X, 1552-4485. doi: 10.1090/qam/910462.

J. Stark, D. S. Broomhead, M. E. Davies, and J. Huke. Takens embedding theorems for forced and stochastic systems. *Nonlinear Analysis: Theory, Methods & Applications*, 30(8):5303–5314, Dec. 1997. ISSN 0362-546X. doi: 10.1016/S0362-546X(96)00149-6.

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, Dec. 2020. ISSN 1867-2957. doi: 10.1007/s12532-020-00179-2.

F. Takens. Detecting strange attractors in turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence, Warwick 1980*, pages 366–381, Berlin, Heidelberg, 1981. Springer. ISBN 978-3-540-38945-3. doi: 10.1007/BFb0091924.

R. Vazquez, E. Schuster, and M. Krstic. A Closed-Form Full-State Feedback Controller for Stabilization of 3D Magnetohydrodynamic Channel Flow. *Journal of Dynamic Systems, Measurement, and Control*, 131(041001), Apr. 2009. ISSN 0022-0434. doi: 10.1115/1.3089561.

M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, Dec. 2015. ISSN 1432-1467. doi: 10.1007/s00332-015-9258-5.

J. Zemánek. *Distributed Manipulation by Controlling Force Fields through Arrays of Actuators*. PhD thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, 2018.

J. Zemánek, T. Michálek, and Z. Hurák. Phase-shift feedback control for dielectrophoretic micromanipulation. *Lab on a Chip*, 18(12):1793–1801, June 2018. ISSN 1473-0189. doi: 10.1039/C8LC00113H.