Control of the Transitional Boundary Layer

Brandt A. Belson

A DISSERTATION

PRESENTED TO THE FACULTY

OF PRINCETON UNIVERSITY

IN CANDIDACY FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE

BY THE DEPARTMENT OF

MECHANICAL AND AEROSPACE ENGINEERING

Adviser: Clarence W. Rowley

January 2014

Abstract

This work makes advances in the delay of boundary layer transition from laminar to turbulent flow via feedback control. The applications include the reduction of drag over streamline bodies (e.g., airplane wings) and the decrease of mixing and heat transfer (e.g., over turbine blades in jet engines).

A difficulty in many fields is designing feedback controllers for high-dimensional systems, be they experiments or high-fidelity simulations, because the required time and resources are too large. A cheaper alternative is to approximate the high-dimensional system with a reduced-order model and design a controller for the model. We implement several model reduction algorithms in `modred`, an open source and publicly available library that is applicable to a wide range of problems.

We use this library to study the role of sensors and actuators in feedback control of transition in the 2D boundary layer. Previous work uses a feedforward configuration in which the sensor is upstream of the actuator, but we show that the actuator-sensor pair is unsuitable for feedback control due to an inability to sense the exponentially-growing Tollmien-Schlichting waves. A new actuator-sensor pair is chosen that more directly affects and measures the TS waves, and as a result it is effective in a feedback configuration. Lastly, the feedback controller is shown to outperform feedforward controllers in the presence of unmodeled disturbances.

Next, we focus on a specific type of actuator, the single dielectric barrier discharge (SDBD) plasma actuator. An array of these plasma actuators is oriented to produce stream-wise vorticity and thus directly cancel the structures with the largest transient growth (so-called stream-wise streaks). We design a feedback controller using only experimental data by first developing an empirical input-output quasi-steady model. Then, we design feedback controllers for the model such that the controllers perform well when applied to the experiment.

Lastly, we also simulate the plasma actuators and determine a suitable numerical model for the forces they create by comparing with experimental results. This physical force model is essential to future numerical studies on delaying bypass transition via feedback control and plasma actuation.

Acknowledgements

It is only with the support of many others that I am able to present this dissertation, and I owe them all gratitude. My first thanks are to my advisor, Clancy Rowley. Over the last five years, Clancy's guidance, wisdom, and patience have greatly and unquestionably improved me as a scientist, problem solver, and writer. He encouraged me to explore my interests and learn about topics that advanced my research, and, in the process, to try to fully understand these topics. This approach has, and will continue to, serve me well.

My time as a graduate student has been graced with several terrific collaborators. From the University of Toronto, Phil Lavoie and his students Ronnie Hanson and Denis Palmeiro provided expert guidance on all things related to plasma actuators and supportive and constructive feedback on our joint work.

At Michigan State University, Ahmed Naguib and his student Kyle Bade have been great collaborators. They were gracious enough to invite me to MSU (and Kyle's spare bedroom) for a week to participate in conducting wind-tunnel experiments, which was a valuable and novel experience for a computationalist such as myself.

Thanks are due to many people at the Royal Institute of Technology of Sweden, KTH, in Stockholm. To start, they provided and assisted me with their extensive boundary layer simulation software, SIMSON. I visited the KTH Mechanics department for two months in the fall of 2011 and benefitted greatly from collaboration with Dan Henningson and Onofrio Semeraro. Additionally, I appreciate the insightful discussions on flow solver techniques with Philipp Schlatter, Peter Lenaers, and Geert Brethouwer. The people in the department made me feel welcome, particularly Miloš Ilak, Luca Brandt, George El Khoury, Jose Sharath, and the entire Floor 8 community.

Next, I'm grateful for the feedback and support of my Ph.D. committee members, Gigi Martinelli and Jeremy Kasdin. Furthermore, Marcus Hultmark and Jeremy Kasdin's suggestions as readers greatly improved this dissertation.

My undergraduate research advisors, Tim Colonius, John Dabiri, Oscar Bruno, and Randy Paffenroth played a large role in my decision to enroll in graduate school and research the topics in this dissertation. I thank them for their guidance and encouragement.

My officemates are due tremendous thanks for support, for celebration, for insights that few others could provide, and for future ideas, even if some of those ideas were to start a low-budget bus service to Princeton Junction and a high-budget submarine service from South America. Naming names, Sunil Ahuja, Zhanhua Ma, Peter Norgaard, Miloš Ilak, Steven Brunton, Lauren Padilla, Jonathan Tu, Zhonglin (Johnny) Zhang, Kevin Chen, Imène Goumiri, Carla Bahri, Scott Dawson, Anthony DeGenarro, Kunihiko (Sam) Taira, Mark Luchtenburg, Matt Williams, and Maziar Hemati. I was fortunate to have two great undergraduate advisees over the summers as well: Katelyn Meidell and Marinela Popova.

Providing balance were the soccer and softball teams, fellow MAE students, roommates, and the many other friends I made along the way. Unfortunately (and fortunately), there are far too many of you to list individually. Lastly, I would not be who I am or achieved what I have without the support and love of my family. I cannot thank them enough.

This work was supported by a grant from the National Science Foundation. This dissertation carries the number T-3272 in the records of the Department of Mechanical and Aerospace Engineering.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Boundary layers are prevalent in engineering applications. For example, the boundary layers formed by the flow of air over wind turbine blades and airplane wings play a critical role in their operation and efficiency. Similarly, the drag over automobiles and ships is largely determined by the properties of the surrounding boundary layer. In industrial processes, heat transfer and chemical mixing are influenced by the boundary layer. The flow of air over the Earth's surface is a boundary layer on a massive scale, and understanding it is critical for predicting the weather and climate.

Boundary layers are classified as laminar or turbulent. Each classification is characterized by different properties, and certain applications benefit more from one type of boundary layer than another. In the case of streamlined bodies such as airplane wings, turbine blades, and boat hulls, skin friction drag is often the largest source of drag, and so laminar flow, with its smaller spatial velocity gradients, is desirable. However, in stall conditions, flow can separate around a wing and pressure drag becomes important. Since turbulent flow is less susceptible to separation, turbulent flow is often intentionally created for such flows to reduce the total drag. Generally, turbulent flow is characterized by swirling eddies that increase mixing and transport, in contrast to laminar flow, which is stratified and smooth. In some cases, this increased mixing is advantageous, for example in expediting chemical reactions among multiple reagents. Turbulent mixing also increases heat transfer, which could be a benefit or a detriment in different applications, or even at different times within a single process.

Due to the different properties of laminar and turbulent flow, it is advantageous to be able to control the flow and choose between the two, depending on which is more desirable. This area of research falls under the broad category of flow control, and in this case is built on an understanding of the primary mechanisms underlying the physics of transition. At high enough Reynolds numbers, Re, (defined in (2.4)), disturbances can cause laminar flow to transition to turbulence, which then remains turbulent. Research continues to be conducted on how to use this understanding to delay transition and preserve laminar flow, and multiple approaches will be discussed shortly. Alternatively, creating turbulent flow is simple to achieve by disturbing laminar flow in any number of ways.

This thesis focuses on reducing drag (skin friction) and mixing by preventing laminar flow from transitioning to turbulent flow. We consider a flat wall geometry, and the flow is incompressible with zero pressure gradient. This geometry is found in many applications and,

more generally, serves as a first step towards more complicated cases encountered in practice. Further, we demonstrate useful techniques and approaches and identify key principles that carry over to practical applications.

In this thesis, we delay transition via active control, i.e., we measure flow quantities with sensors then use that output information to determine the actuation input. We examine the effects of different types of sensors and actuators on the effectiveness of the control. In simulations of 2D flow, we measure a localized velocity near the wall as the output and use localized forcing near the wall as the inputs, and both are meant to be similar to what could be used in a physical experiment or application. In the full 3D case, we both simulate the flow and collaborate with experimentalists and so we use physical, wall-mounted shear sensors as outputs and plasma actuators as inputs. Plasma actuators present unique advantages for control, creating small forces very near the wall. Some major practical advantages include that they do not alter the flow when turned off because they are very thin, they require little maintenance because there no moving parts, they can be affixed to existing wings/bodies, and their operation simply requires applying a voltage.

This research is novel for the model-based and rigorous design of controllers from both experimental and simulation data, and yields important insights into the physical fluid system for effective control of transition. The models are based on the aspects of the flow that matter from a control perspective—the input-output dynamics. That is, all of the fluid dynamics are not resolved by the model, only the components that play a role in the dynamics between the inputs and outputs. The benefit of such models is that they have much lower dimensionality than the original fluid system, and so it is cheaper (with regard to wind-tunnel time, computation, etc.) to analyze and design controllers for the models than the original system.

1.1 Previous work on transition control

There are many approaches to flow control. One is known as passive control and no measurements are made, the flow is simply manipulated either by actuators with a predefined time signal or by slightly altering the geometry to achieve a desired effect. For example, airplane wings are often outfitted with vortex generators to trip turbulence and thus prevent separation and stall. Also, many airfoil shapes are carefully designed to achieve high lift, low drag, and to avoid the detrimental effects of separation and transition.

Often, active control, sensing the flow and using this information to determine the actuation, outperforms passive control. As is the case for passive control, many active control techniques are based on fluid dynamic understanding and/or models of the most important low dimensional structures to make control design easier. For example, in [20], the authors reduce the drag in a turbulent boundary layer by 20-30% with physical insight and without a model. They test various controller schemes, including blowing/suction at the wall with an amplitude proportional to the instantaneous velocity slightly above the wall, and explain the physical mechanism for drag reduction. The physical mechanism is further explained in [36] as a "virtual wall" above the wall through which almost no fluid passes.

The work presented in [106] uses active wave cancellation to cancel Tollmien Schlichting (TS) waves that lead to transition. They perform experiments in which the TS waves are

sensed upstream by multiple sensors, and this information is used so that the downstream actuators act out of phase with the TS waves. This is demonstrated using two different types of actuation, both located at the wall. This type of control is called feedforward because the upstream sensors measure the disturbance, which informs the downstream actuation, but the sensors do *not* measure the effect of the actuation (this is feedback and discussed later). In another example of feedforward control, upstream wall-mounted sensors measure the shear stress in experiments [77]. As in other works, no model is used. Instead, when measured shear stress is above a certain threshold, the controller waits a predefined time delay before applying suction through holes in the wall. The time delay corresponds to the time it takes for the flow to convect from the sensors to the actuators. The result is a decreased amplitude of disturbances that could trigger transition, and this decrease in amplitude exists over a prolonged downstream distance.

Better controllers can be found by first finding a model of the fluid system's inputs-output behavior. The system can be an actual experiment, a high-fidelity fluid simulation, or a simplified model, as long as the inputs and outputs are well-defined. The model at least serves as an efficient testbed to try many types of ad hoc controllers or to tune gains, rather than expending large amounts of resources on large simulations or experiments. This is the approach taken in [69], in which experimental data is used to find an empirical model of the input-output relationship. This model is simplified to essentially just a time delay, and use the model to design a controller, then using the controller on the model, then on the original experiment. The result is a significant reduction of random upstream disturbances downstream of the actuators.

Beyond using models as testbeds, control theory tools can be applied to them to develop more advanced and effective controllers. The work of [55] does this by linearizing and discretizing the Navier-Stokes equations to come up with a linear input-output model of plane Poiseuille flow. This system decouples by stream-wise wavenumber, yielding a set of one-dimensional systems. The authors then analyze the poles and zeros of the systems, which helps them choose the physical locations of their outputs (shear-stress sensors) to avoid non-minimum-phase zeros, which complicate controller design. They find an effective feedback proportional-integral controller which determines the actuation (blowing/suction at the wall) and stabilizes the unstable open-loop system. The work in [13] expands on these results. Using a similar model, they use more advanced control design techniques: both optimal (H_2) and robust (H_∞) controllers. They demonstrate that they can not only stabilize the system, but increase performance and robustness to model uncertainty, as compared to the proportional-integral control used in [55]. A 3D study is done in [44] in which the model is composed of 1D systems for each pair of stream-wise and span-wise wavenumbers. Here, the controllers are again designed on a wavenumber pair basis. The controllers make use of an optimal Linear Quadratic Regulator, which provides the actuation signal for blowing/suction, and a Kalman filter to estimate the state from the shear stress measurements. The resulting controllers are effective, significantly damping a few types of disturbances.

The models in the previous works are based on approximations and simplifications to the governing equations primarily based on physical insights. The order of the original system can be reduced even when there are no further physical approximations to be made. This is done in [30], where the model is found by keeping the components which are most important for reconstructing the input-output behavior of the original system, a method

known as balanced truncation. They demonstrate this method's utility for fluid systems by applying it to Couette flow, showing that it accurately reproduces the original system. In [67], reduced-order models are also found via balanced truncation, in this case a model is found for each stream-wise wavenumber in the wall-normal direction. The authors design an H_2-optimal controller for each wavenumber based on the corresponding reduced-order model. The span-wise variations are controlled in an ad hoc matter, and the drag is reduced by about 10% from the drag in the original turbulent nonlinear boundary layer.

Balanced truncation is not computationally tractable for very large systems, but it can be approximated with a method called balanced Proper Orthogonal Decomposition (BPOD). The resulting model from BPOD is a close representation of the input-output behavior of the original high-dimensional linear system. This method is used in [7] to find a model from localized forcing inputs to localized average velocity outputs in the 2D spatially-evolving boundary layer. Linear Quadratic Gaussian (LQG) feedforward controllers are developed based on the model and are effective at canceling the growth of TS waves. This work is extended to three dimensions in [101]. Here, several independent BPOD models and LQG controllers are designed for sensor-actuator pairs aligned in the stream-wise direction where the sensor is directly upstream from the actuator (i.e. at the same span-wise position). This distributed, feedforward, scheme is effective at reducing the growth of disturbances, and is also shown to be robust to changes in flow conditions. Our work is related to these two, using BPOD and similar methods to find models which we use as the basis for control design and analysis.

Other methods exist as well. For example, in [3] a model based on the eigenfunctions of the linearized 2D Navier-Stokes is developed and used for controller design. This method has limitations though since the resulting models may not accurately approximate the effect of inputs on the outputs.

1.2 Organization and contributions

Chapter 2: The governing equations for the incompressible boundary layer are given. The mechanisms for transition are explained in terms of the linearized versions of these equations. Since the equations and their numerical solution are closely related, the numerical algorithm and software we use to simulate the flow are discussed. The main contribution here is substantial improvements to the original version of this software, particularly related to the control aspects. We also develop an extensive post-processing suite for analysis and visualization, and explain its functionality.

Chapter 3: Chapter 3 focuses on the new Python model reduction library `modred`. This library is used in this thesis, but it is also designed to be used in many other applications. Currently it is used by other members of our research group and by researchers at other universities. The beginning of the chapter (Sections 3.1, 3.2, and 3.3) also serves to introduce the model reduction techniques used in later chapters.

The author of this thesis is responsible for the majority of the concept, design, and implementation. J. H. Tu and C. W. Rowley participate in all phases of the design and J. H. Tu implements parts of the library, including the Dynamic Mode Decomposition algorithm. The work is submitted for publication in *ACM Transactions on Mathematical Software* [10].

Chapter 4: Here, we control the transitional two-dimensional boundary layer by damping the growth of TS waves. Previous work on this problem uses a feedforward controller to achieve good performance. We illustrate the difference bewtween feedforward and feedback configurations based on the relative position of the actuator and sensor. The role of the actuator and sensor on feedback control are investigated, and we show that the original actuator and sensor are poorly suited for damping the growth of TS waves. We find a better choice of actuator and sensor.

The author is responsible for conducting the simulations, model reduction, control design, and sensor and actuator analysis. O. Semeraro and D. S. Henningson provide both the software to simulate the system and useful insights into the flow physics. These results are published in *Physics of Fluids* [9].

Chapter 5: This chapter focuses on controlling bypass transition in experiments, in collaboration with R. E. Hanson and P. Lavoie at University of Toronto, and K. Bade and A. M. Naguib at Michigan Statue University. Three-dimensional stream-wise streaks of stream-wise velocity are created by roughness elements upstream on the wall to mimic the process of transition in a predetermined way. These streaks are then significantly damped via plasma actuation and feedback control.

The author's contribution is primarily in the design of a model and controller which improve the performance from the previous ad-hoc approach. R. E. Hanson and K. M. Bade conduct the wind-tunnel experiments. R. E. Hanson fabricates the plasma actuators and K. M. Bade implements of the roughness elements that disturb the laminar flow. This work is submitted for publication in *Physics of Fluids* [38].

Chapter 6: While it is useful to find models from limited experimental measurements, as done in Chapter 5, access to the entire velocity field from simulations could result in more insights and better models. Simulating plasma actuators is challenging, though, because they operate at much smaller spatial and temporal scales than those of the surrounding fluid, making it computationally inefficient to resolve both the plasma and fluid scales in the same simulation. Instead, we approximate the effect of the plasma actuators as a body force. This chapter validates our approximation by comparing simulations to experiments.

The author is responsible for conducting the simulations, adjusting the models, and validating the models against experimental measurements. Former undergraduate summer researchers K. Meidell and M. Popova also conduct some of the simulations under the author's guidance. The experimental measurements are taken by R. E. Hanson. Portions of this work are published in a conference article [**?**].

Chapter 7: We summarize our results and possible future directions for this research.

Chapter 2

Numerical simulation of boundary layers with feedback control

2.1 Governing equations

The physical system of interest is the three-dimensional incompressible boundary layer over a flat wall with zero pressure gradient, as shown in Figure 2.1. This system is governed by the incompressible Navier-Stokes equations

$$\rho \frac{\partial \mathbf{v}'}{\partial t'} = -\nabla' p' - \rho(\mathbf{v}' \cdot \nabla')\mathbf{v}' + \mu \nabla'^2 \mathbf{v}' + \mathbf{f}' \tag{2.1}$$

$$\nabla' \cdot \mathbf{v}' = 0 \tag{2.2}$$

where (2.1) is the momentum equation, (2.2) is the continuity (mass) equation, $\mathbf{v}'$ is the velocity with components $[u',\ v',\ w']$, t' is time, ∇' is the spatial derivative operator $[\frac{\partial}{\partial x'},\ \frac{\partial}{\partial y'},\ \frac{\partial}{\partial z'}]$, p' is the pressure, and $\mathbf{f}'$ is an arbitrary volume force. The quantities μ and ρ are the shear viscosity and density, respectively, and both are constant throughout space $[x',\ y',\ z']$ and time t'. The prime, $(\,)'$, denotes that these are dimensional quantities.

It is useful to non-dimensionalize these equations by characteristic length, time, and mass scales. There is no single obvious choice of length scale, and in practice many different length scales are used. In this research, we use the displacement thickness of the boundary layer, defined as

$$\delta_0^* = \int_0^\infty (U'_\infty - u'|_{x'_0}) dy' \tag{2.3}$$

where U'_∞ is the stream-wise component of the free-stream velocity and $u'|_{x'_0}$ is the stream-wise component of velocity at a particular x'_0 location. (We choose the computational inlet as the reference location.) The non-dimensional Reynolds number, Re, defined as

$$Re = \delta_0^* U'_\infty \rho / \mu, \tag{2.4}$$

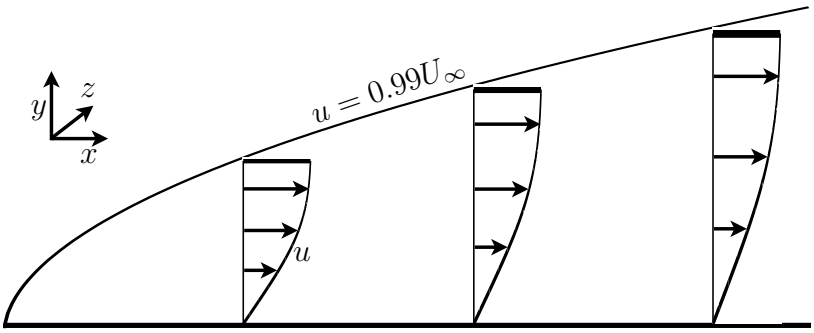


Figure 2.1: Laminar boundary layer velocity profiles. There is no variation in the z dimension.

arises as a function of the dimensional quantities. The non-dimensional variables are

$$[x, y, z] = \frac{[x', y', z']}{\delta_0^*}, \quad \mathbf{v} = \frac{\mathbf{v}'}{U'_\infty}, \quad U_\infty = 1, \quad t = \frac{t' U'_\infty}{\delta_0^*}$$
$$p = \frac{p'}{\rho U'^2_\infty}, \quad \nabla = \frac{\nabla'}{\delta_0^*}, \quad \mathbf{f} = \frac{\mathbf{f}' \delta_0^*}{\rho U'^2_\infty}.$$

Substitution into the governing equations results in

$$\frac{\partial \mathbf{v}}{\partial t} = -\nabla p - (\mathbf{v} \cdot \nabla)\mathbf{v} + \frac{1}{Re}\nabla^2 \mathbf{v} + \mathbf{f} \tag{2.5}$$
$$\nabla \cdot \mathbf{v} = 0. \tag{2.6}$$

We work with the non-dimensional form of the equations ((2.5) and (2.6)) rather than the dimensional form ((2.1) and (2.2)) throughout the rest of the chapter and the thesis. The boundary conditions are dictated by the boundary layer geometry and are discussed in Section 2.3.

2.2 Transition from laminar to turbulent flow

Fluid flows can generally be classified as either laminar, which is relatively smooth, or turbulent, which is characterized by larger velocity gradients and swirling eddies. At higher Re, fluid can undergo a transition from laminar flow to turbulent flow. This research focuses primarily on preserving laminar flow by sensing and controlling the early stages of transition. Applications include the reduction of drag and mixing over streamlined bodies, such as turbine blades and wings. In these cases, laminar boundary layers are more desirable than turbulent ones because the flow is more stratified and has less mixing. The reduced mixing also lowers the velocity gradient near the wall, resulting in lower skin friction drag. Since this is the largest source of drag in the absence of other effects such as separation, the total drag is less in laminar boundary layers.

There are multiple mechanisms by which a laminar flow can transition to turbulent flow. The first, *classical* transition, is predicted by a linear stability analysis of the governing

equations (2.5) and (2.6). The equations are simplified by the use of periodic boundary conditions in the x direction, and the parallel flow assumption. The resulting linear system is further reduced by assuming the solution has no variation in the z direction, yielding the Orr-Sommerfeld equations, which are unstable for high Re [115]. The unstable eigenvectors are known as Tollmien-Schlichting (TS) waves, and they are constant in the spanwise (z) direction and vary in the stream-wise (x) direction. TS waves can arise in the presence of very small magnitude perturbations to laminar flow, such as vibrations, surface roughness, or free-stream turbulence. The amplitude of the TS waves grows exponentially, eventually becoming large enough that nonlinear effects become important. At this point, the TS waves breakdown into secondary structures. Turbulent spots appear and eventually merge, resulting in fully turbulent flow. This type of transition is the focus of Chapter 4, where we detect TS waves and prevent their growth with active feedback control.

A second transition mechanism is called *bypass* transition because it bypasses the classical path to turbulence. In this case, larger disturbances trigger algebraic transient growth along *stable* directions. This is possible because the eigenvectors of the Orr-Sommerfeld equation are non-normal. As with any stable linear system with non-normal eigenvectors, algebraic growth can occur before the exponential decay dominates, sending solution towards the stable equilibrium. The fluid structures which undergo transient growth are elongated in the stream-wise direction and consist of stream-wise vorticity with alternating sign in the spanwise direction. The structures can also be described as a spanwise array of streaks that alternate between high and low streamwise velocity. For a purely linear system, these streaks would grow in magnitude and eventually decay, since they are stable. However, in the full Navier-Stokes equations, the streaks can grow large enough that nonlinear effects dominate, precluding their exponential decay. Instead, the streaks break down into other structures and eventually form turbulent spots and fully turbulent flow.

These streaks are fundamentally three-dimensional, as opposed to the essentially two-dimensional TS waves. The algebraic growth, as triggered by larger disturbances, is initially faster than the exponential growth of TS waves, and therefore bypass transition path is more common for larger disturbances. Further, this algebraic growth occurs at stable Re, so bypass transition can result in turbulence at subcritical (stable) Re. Bypass transition, and preventing it via active feedback control, is the focus of Chapter 5.

2.3 Numerical solution

This research uses both experiments and simulations to find effective ways to control transition and gain physical insights. The simulations provide the ability to quickly change and explore different parameters and control strategies. For example, in Chapter 4 we vary the sensor and actuator position and types over a wide range of possibilities relatively easily. This same task, in experiments, could be extremely costly. The simulations also provide the entire velocity field, making more types of analysis and modeling possible. In Chapter 6, we use this information to verify the simulation of plasma actuators and compare some flow quantities to experimental measurements.

There are many ways to simulate fluid flows with varying levels of approximations and assumptions. Here we directly numerically simulate the system, resolving all relevant length

scales, with spectral methods. In spectral methods, the solution is approximated as a finite sum of spectral basis functions that have known analytical derivatives and provide high-order spatial accuracy. Generally, spectral methods are applicable to a limited class of problems. They are applicable here because the solution is smooth (no shocks) and the domain is a simple rectangular shape that does not vary in time and is discretized on a Cartesian grid.

2.3.1 Existing flow simulation software

To simulate the flow, we use existing software called SIMSON (pseudo-spectral solver for incompressible boundary layers) that has been developed and extensively used by Dan Henningson's group at the Royal Institute of Technology Sweden (KTH) since the early 1990s [12]. It is written in Fortran 77/90 and is parallelized for distributed and shared memory architectures via MPI and OpenMP. We make several contributions to this software to perform our studies, particularly related to control (Section 2.3.2).

The algorithm it implements is described in [63] and follows a wall-normal vorticity-velocity formulation. These two quantities are first advanced via time evolution equations, then the two other components of velocity are solved for via boundary-value problems derived from the definition of vorticity, $\boldsymbol{\omega} = \nabla \times \mathbf{v}$, and the continuity equation. The pressure is eliminated when taking the curl of the momentum equation to find the vorticity equation, and it is not needed in the solution procedure. It can be solved for in post-processing, but this is unnecessary in this research.

We outline this procedure now. The equations to advance in time, after manipulation, are

$$\frac{\partial \boldsymbol{\omega}_y}{\partial t} = \frac{\partial \mathbf{N}_x}{\partial z} - \frac{\partial \mathbf{N}_z}{\partial x} + \frac{1}{Re}\nabla^2 \boldsymbol{\omega}_y \tag{2.7a}$$

$$\frac{\partial (\nabla^2 v)}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)\mathbf{N}_y - \frac{\partial^2 \mathbf{N}_x}{\partial x \partial y} - \frac{\partial^2 \mathbf{N}_z}{\partial y \partial z} + \frac{1}{Re}\nabla^4 v \tag{2.7b}$$

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right) u = \frac{\partial \boldsymbol{\omega}_y}{\partial z} - \frac{\partial^2 v}{\partial y \partial z} \tag{2.7c}$$

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right) w = -\frac{\partial \boldsymbol{\omega}_y}{\partial x} + \frac{\partial^2 v}{\partial y \partial z} \tag{2.7d}$$

where $\mathbf{N} = (\mathbf{v} \cdot \nabla)\mathbf{v}$ is the nonlinear convective term, and subscripts x, y, and z denote components of vectors.

The boundary conditions for these equations are handled in a careful way to facilitate the numerical method. The solution variables, $\mathbf{v}$ and $\boldsymbol{\omega}_y$, are expressed as Fourier series in the stream-wise and spanwise directions and as Chebyshev polynomials in the wall-normal direction. However, Fourier series require periodicity to avoid a dramatic loss of accuracy (due to Gibb's phenomenon) and the boundary layer evolves in the stream-wise direction (Figure 2.1) making it clearly aperiodic. To remedy this, a non-physical "fringe" region is appended to the downstream end of the domain [12]. In this fringe region, a body force is imposed to drive the boundary layer to the specified, laminar, inlet velocity at the outlet. In this way, the solution over the entire extended domain is artificially made to be periodic, while still satisfying the governing equations ((2.5) and (2.6)). The flow is continuously

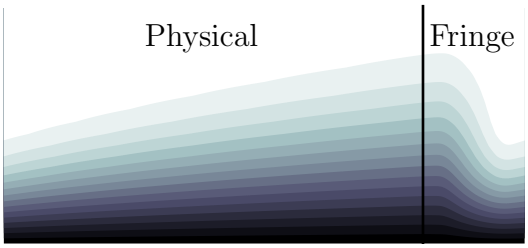
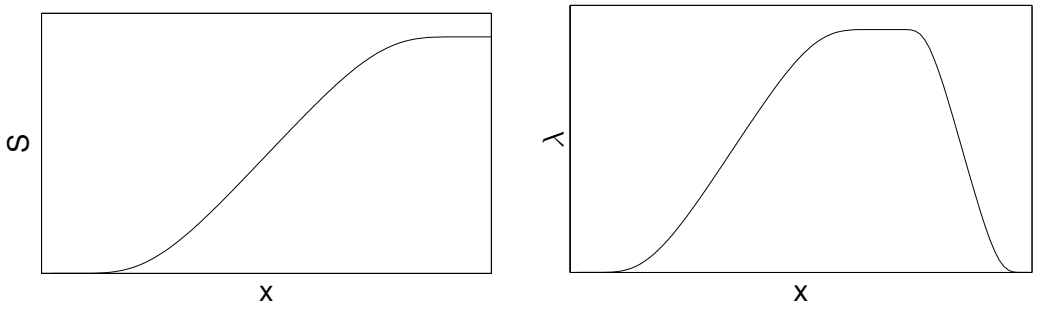


Figure 2.2: The laminar boundary layer velocity in the computational domain, including the non-physical fringe region. Contours of the stream-wise component of $\boldsymbol{\mathcal{V}}(x, y)$ are shown.



(a) $S(x)$ as in (2.8)

(b) $\lambda(x)$ as in (2.9)

Figure 2.3: Weighting functions in the fringe region.

forced over the fringe region towards a prescribed velocity profile, $\boldsymbol{\mathcal{V}}$,

$$\boldsymbol{\mathcal{V}}(x, y) = \mathbf{V}(x, y)(1 - S(x)) + \mathbf{V}(x_0, y)S(x) \tag{2.8}$$

where $S(x)$ is a smooth weighting function that increases from zero to one in the fringe region and x_0 is located at the computational inlet. Figure 2.2 shows $\boldsymbol{\mathcal{V}}$ and Figure 2.3a shows $S(x)$, see [19] for the definition of $S(x)$. The force in the fringe region acts like a proportional controller

$$\mathbf{f} = \lambda(x)(\boldsymbol{\mathcal{V}} - \mathbf{v}) \tag{2.9}$$

where the gain, $\lambda(x)$, is shown in Figure 2.3b. The shapes of $S(x)$ and $\lambda(x)$ are carefully designed and tuned to prevent the effect of the artificial force from contaminating the solution outside of the fringe region.

Returning to the boundary conditions, they are periodic in x ($\mathbf{v}|_{\text{outlet}} = \boldsymbol{\mathcal{V}}|_{\text{inlet}} = \mathbf{v}|_{\text{inlet}}$) and z. The theoretical physical boundary conditions in y are $\mathbf{v} = \boldsymbol{\mathcal{V}}$ and $\boldsymbol{\omega}_y = 0$ both at the wall and infinitely far from the wall. However, it is computationally expensive to use the infinitely far boundary condition, and so the boundary conditions are approximated as

$$\frac{\partial \mathbf{v}}{\partial y} = \frac{\partial \boldsymbol{\mathcal{V}}_y}{\partial y} \text{ at the wall and top of domain; and} \tag{2.10}$$

$$\boldsymbol{\omega}_y = 0 \text{ at the wall and top of domain.} \tag{2.11}$$

The domain is chosen to extend far enough in the wall-normal y direction to be far from the top (where $u = 0.99U_\infty$) of the boundary layer so that these approximations have limited impact lower in the boundary layer.

Table 2.1: Time discretization scheme coefficients

Type	$a^n/\triangle t$	$b^n/\triangle t$	$c^n/\triangle t$
3-stage	8/15	0	0
	5/12	-17/60	8/15
	3/4	-5/12	2/3
4-stage	8/17	0	0
	17/60	-15/68	8/17
	5/12	-17/60	8/15
	3/4	-5/12	2/3

To solve these equations, first (2.7a) and (2.7b) are advanced in time. The time discretization of the nonlinear term uses an explicit third-order-accurate Runge-Kutta scheme, while the time discretization of the linear term uses the implicit second-order-accurate Crank-Nicolson scheme. The implicit treatment of the linear term increases the length of the time step allowable for numerical stability. In general, this time discretization is written as

$$\begin{aligned} \frac{\partial \psi}{\partial t} &= N(\psi) + L(\psi) \\ \psi^{n+1} &= \psi^n + a^n N(\psi^n) + b^n N(\psi^{n-1}) + (a^n + b^n)(L(\psi^{n+1}) + L(\psi^n))/2 \end{aligned} \tag{2.12}$$

for any variable ψ, where N is a nonlinear function and L is a linear function, n is the Runge-Kutta substep, and the coefficients are given in Table 2.1. Solving these equations involves solving a linear system of equations (of Helmholtz form) due to the implicit treatment of the linear term. In Fourier space (in x and z), these systems of equations decouple by wavenumber pair into $n_x/2 \times n_z$ one-dimensional Helmholtz equations discretized in y, which are solved efficiently (for example, see [92] Chapter 3). By breaking apart homogeneous and inhomogeneous solutions for $\nabla^2 v$, boundary conditions on $\nabla^2 v$ are not explicitly needed, only on v. See Section 5 of [19] for details.

After the time advancement of $\boldsymbol{\omega}_y$ and v, (2.7c) and (2.7d) are solved at the new time step for u and w. These equations also decouple by wavenumber in x and z in Fourier space, resulting in a set of one-dimensional Poisson-like problems in the y dimension, which can be efficiently solved with spectral accuracy.

2.3.2 Contributions to flow control in SIMSON

Over the course of SIMSON's development, it's primary purpose has been to solve the fluids equations in a computationally efficient manner. The inclusion of active flow control is relatively recent. As a result, the inherited version of SIMSON has some control capabilities, but they are ad-hoc and need to be improved and extended to perform the research in this thesis. Therefore, the control aspects are reorganized, rewritten, expanded, and generally improved. More modern aspects of the Fortran language are utilized, such as built-in matrix multiplication and variable names exceeding six characters, and general software design principles are followed.

Calculation inner products

In the SIMSON, the inner products between vectors are computed as spatial integrals over the domain Ω,

$$\langle \mathbf{v}_1,\, \mathbf{v}_2 \rangle = \int \mathbf{v}_1^* \mathbf{v}_2 \, d\Omega. \tag{2.13}$$

Inner products are used for control when sensors are represented as spatial distributions. In this case, inner products are taken of sensors' spatial distributions and the velocity to find the measurement signal.

To numerically compute the inner products with discretized variables, the integral is approximated by the trapezoidal rule in the y direction in the inherited version of the code. This is only second-order accurate. We replace the trapezoidal rule weights with spectral weights to achieve high-order accuracy [37].

Further, the computation of the integral in the x and z directions is also changed. The inner products in the inherited version were taken in physical space in the x and z directions. However, the velocities are in Fourier space for most of the steps in the algorithm, and are only in physical space when computing the nonlinear term (the products are more computationally efficiently computed in physical space). In the code, this transformation to physical space and back to Fourier space happens deep in subroutines originally designed only to compute the nonlinear term and these subroutines have nothing to do with control. The inherited version contains modified versions of these subroutines which also compute inner products, but the organization and implementation is unnatural and thus error-prone. Instead, in our modified version, we move the inner product computation to a more natural location in the code with the rest of the control-related computations. The velocity is not in physical space though, and so we perform inner products in Fourier space by exploiting Plancherel's theorem in both the x and z dimensions, which states:

$$\sum_{n=0}^{N-1} v_n w_n^* = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{v}_k \tilde{w}_k^* \tag{2.14}$$

where v and w are 1D arrays composed of N complex numbers, and $\tilde{v}$ and $\tilde{w}$ are the corresponding Fourier coefficients. In 2D, this becomes

$$\sum_{i=0}^{n_x-1} \sum_{j=0}^{n_z-1} v_{i,j} w_{i,j}^* = \frac{1}{n_x n_z} \sum_{k=0}^{n_x-1} \sum_{l=0}^{n_z-1} \tilde{v}_{k,l} \tilde{w}_{k,l}^*. \tag{2.15}$$

Since the velocity is purely real, half the Fourier coefficients are complex conjugates and thus not stored for efficiency. Taking this into account in equation (2.15) yields the computationally efficient form

$$\sum_{i=0}^{n_x/2} \sum_{j=0}^{n_z-1} v_{i,j} w_{i,j}^* = \frac{1}{n_x n_z} \left(2 \sum_{k=0}^{n_x/2} \sum_{l=0}^{n_z-1} \mathrm{real}(\tilde{v}_{k,l}) \mathrm{real}(\tilde{w}_{k,l}^*) + \mathrm{imag}(\tilde{v}_{k,l}) \mathrm{imag}(\tilde{w}_{k,l}^*) + \sum_{l=0}^{n_z-1} \tilde{v}_{0,l} \tilde{w}_{0,l} \right). \tag{2.16}$$

(Note that $\tilde{v}_{0,:}$ and $\tilde{w}_{0,:}$ are purely real.)

The Fourier coefficients of the velocity are distributed among all of the processors, and so to compute the sum in (2.16) each processor computes the contributions to the total sum with the coefficients in its local memory. These contributions are summed together with a call to MPI `allreduce`.

Inclusion of general control architecture

The inherited version of SIMSON implements only one particular type of control law, Linear Quadratic Gaussian, with limited support for multiple inputs and outputs. We rewrite this for a more general class of controllers. In the new control implementation, multiple inputs and outputs of different types are supported. The inputs are distinguished, as in Figure 4.3, as either a disturbance signal (w) or a control signal determined by a control law (u). Similarly, the outputs are distinguished as either offline sensors (z) which are not available to the controller or sensor measurements (y) which are available to the controller.

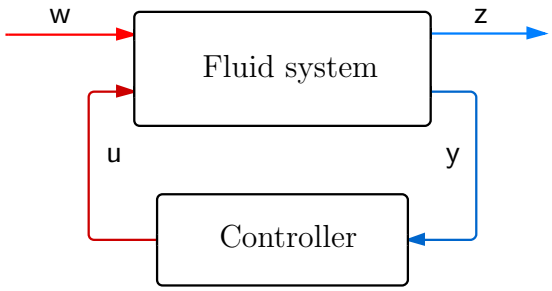


Figure 2.4: Control architecture implemented.

We implement the controller as an arbitrary continuous-time state-space system with an arbitrary number of states, independent of the way the controller was designed,

$$\begin{aligned} \frac{\partial \mathbf{q}_c}{\partial t} &= \mathbf{A}_c \mathbf{q}_c + \mathbf{B}_c \mathsf{y} \\ \mathsf{u} &= \mathbf{C}_c \mathbf{q}_c, \end{aligned} \tag{2.17}$$

where $\mathbf{q}_c$ is the internal state of the controller. In the inherited version, a more restrictive form of the controller system is used that requires a model of the open loop system with controller and observer gains. The new approach and implementation makes it easier to switch between different types of controllers, and the new implementation is particularly valuable in the study presented in Chapter 4.

2.3.3 Post-processing and visualization

SIMSON saves the velocity in raw binary format and discrete Fourier transformed in the x and z directions. While useful for some computational purposes, this format is not readily accessible by external languages or programs for visualization and analysis. Thus, an extensive suite of post-processing tools for reading, writing, visualizing, and analyzing the data is developed.

The vast majority of this code is written in Python, a general-purpose, object-oriented, and dynamically-typed programming language with similarities to Matlab. The primary functions performed in post-processing are

- Loading and saving SIMSON format binary files
- Transforming SIMSON's data between Fourier and physical space
- Loading and saving HDF5 files (useful for visualization and cross-platform compatibility)
- Converting SIMSON-format binary files to HDF5, and vice-versa
- Performing inner products
- Adding and multiplying, enabling manipulation of velocity fields as objects
- Changing units (dimensional and different non-dimensionalizations)
- Reading SIMSON parameter files and generating the grid points
- Automated unit testing of all of the above

HDF5 (hierarchical data format version 5, `http://www.hdfgroup.org/HDF5/`) is a self-describing binary format that is widely-used in high-performance computing. It is easy to work with, and can be read from and written to in Matlab, C++, Fortran, and Python. It also is not architecture dependent, as raw binary files can be.

HDF5 files can also be read by many visualization programs, such as the one we use, VisIt (`https://wci.llnl.gov/codes/visit/`). We select VisIt because it is free, widely supported, and runs on multiple cores remotely. Thus, we can run our simulations on a large remote cluster, then run VisIt on the cluster and see the graphical interface on a local machine, all without transferring the data. Since each time-sampled velocity field can be gigabytes in size, this is important.

Most one and two dimensional plotting (as well as all control design) is done locally in Matlab.

2.3.4 Portability

The inherited version of SIMSON code works only with certain compilers. In particular, it does not compile with gfortran, a common, free, fortran compiler, and the only one available on our local computers. Thus, the new version of SIMSON uses more standard practices and is compatible with Intel, PGI, and gfortran compilers.

2.4 Summary

In summary, we present the governing equations of motion, the non-dimensional Navier-Stokes equations. Analysis of these equations, and experimental observation, tells us there

are two mechanisms for transition from laminar to turbulent flow: classical transition and bypass transition. The different characteristics of these two mechanisms are described in detail as they are important in future chapters. In chapter 4, we delay classical transition using a reduced-order model of the system and a feedback controller. Chapter 5 focuses on the delay of bypass transition in experiments using a quasi-steady feedback controller in which the shear stress is measured at several spanwise locations and the flow is controlled via plasma actuation. In chapter 6, we develop a body force model for plasma actuators that will be useful for future numerical studies on the control of bypass transition.

This chapter also covers the numerical algorithms and software we use to simulate the fluid system. The software is developed by D. S. Henningson's group at KTH, and is a pseudo-spectral solver called SIMSON. The version of SIMSON we are provided had limited specific control functionality. To perform the control study in Chapter 4, we add significant functionality. This functionality includes improving the accuracy of the inner product by making use of the spectral representation of velocity. Further, we add a more general control architecture that allows many types of controllers with multiple inputs and outputs, whereas the original version of SIMSON had only LQG control implemented with limited support for multiple inputs and outputs. In addition to its use in Chapter 4, SIMSON is used for larger 3D simulations of the effect of plasma actuators on the velocity in Chapter 6.

Lastly, this chapter describes a post-processing suite we use to analyze the resulting datasets from SIMSON. Among the many features of this suite, it reads and writes SIMSON binary files, visualizes the data (using VisIt), and simplifies manipulation of velocity fields. It is used heavily throughout Chapters 4 and 6.

Chapter 3

Model reduction on large data

Simulating the boundary layer, even in 2D, is computationally expensive because the system is high dimensional. The number of independent variables is the number of points at which each quantity is sampled, and is often $O(10^5)$ to $O(10^9)$. However, by isolating the important low-dimensional dynamics, we can approximate high-dimensional systems using low-dimensional models, which can be cheaply simulated, saving significant resources. This process is known as model reduction.

Model reduction is of course not unique to the problem at hand; it has uses in engineering, physics, environmental science, and biology. However, the implementations of model reduction methods tend to be for a specific problem and are only applicable to relatively small systems and datasets. Recognizing the need for model reduction in many fields and for large datasets, we write a new Python library, `modred`. It implements several important algorithms in model reduction, modal analysis, and system identification for both small and large datasets. Careful attention to software design principles are followed, including ease of use, modularity, automated unit testing, parallelization, and easy-to-read code and documentation. The library is available at `http://pypi.python.org/pypi/modred`.

The work in this chapter is submitted to the ACM Transactions on Mathematical Software journal for publication [10]. Section 3.1 reviews the different model reduction algorithms and prominent usages in different fields of study. Section 3.2 gives an overview of the modal decompositions for two different computational approaches: matrix-based and vector-based. Short code samples are presented for the implementations of these two approaches, and the advantages of each are discussed. In Section 3.3, we summarize how to find reduced-order models both based on projecting the governing equations onto existing modes and also via impulse responses, and provide more code samples. Section 3.4 describes the software design principles we employ. Section 3.5 explains the parallelization of the modal decompositions, and the theoretical and observed scaling of parallelization of `modred`.

3.1 Introduction

One method we include is the Proper Orthogonal Decomposition (POD). From an arbitrary set of data, POD yields a set of orthonormal modes which represent the dominant directions, in the L_2 sense, in the original data. The modes are ranked and typically few are necessary

to accurately reproduce the original dataset. The method originated as a way to analyze random data, and is known by various alternative names, such as principal component analysis (PCA) [90] and the Karhunen-Loève (KL) decomposition [61, 73]. In a wide range of fields, it has proven to be a useful tool for identifying important coherent structures and for developing reduced-order models via Petrov-Galerkin projection. POD has a long history of applications in fluid dynamics [5, 46, 75, 103]. POD has also been used for studying vibrations, oscillations, and microelectromechanical systems [6, 70]. In geophysics, POD modes are often called empirical orthogonal functions (EOF), and are used to observe strong climate patterns [47]. POD is also used in image processing, chemical engineering, control theory, and dynamical systems.

Balanced POD (BPOD) is a variant of POD that has desirable properties for constructing reduced-order models of large, linear input-output systems. It produces two sets of bi-orthogonal modes, and projecting the original linear system onto these modes results in a model that accurately reproduces the input-output dynamics of the original system [96]. BPOD is designed to approximate *balanced truncation* [82], a model reduction technique commonly used in control theory that has good *a priori* error bounds. BPOD is particularly useful when the original system is too large for standard control design techniques (or model reduction techniques) to be applied: the algorithm is tractable even when the state dimension is large, and control design can then be done using the reduced-order model. BPOD is applicable to any system of linear ordinary differential equations (ODEs), including discretizations of partial differential equations (PDEs). The method has been applied to several fluid dynamics systems for control purposes [2, 24, 53]. Our library includes methods to find BPOD modes and also to project a linear system onto these modes to find a reduced-order model.

Another modal decomposition technique included in `modred` is the Dynamic Mode Decomposition (DMD) [100], which identifies modes that oscillate at fixed frequencies. For systems with oscillatory dynamics, these structures may be more indicative of the flow physics than the most energetic (POD) modes, which may contain mixed frequency content. DMD has been applied to both computational and experimental data, for instance to a jet in cross-flow and the wake behind a flexible membrane [97, 100]. Connections between the DMD and Koopman operator theory suggest that DMD modes may be useful for capturing the behavior of nonlinear systems [80, 97].

Finally, `modred` also includes system identification techniques: the Eigensystem Realization Algorithm (ERA) and Observer/Kalman Filter Identification (OKID). ERA takes a series of Markov parameters (defined in (3.21)) and from them produces a reduced-order input-output model. The resulting models are theoretically equivalent to those produced by BPOD, but computing them does not require computing adjoint simulations or finding two sets of modes, making ERA computationally cheaper by orders of magnitude for large systems [58, 78]. Its primary application is control design for high-order systems. It also can be applied to experimental data since it does not require data from an adjoint system.

OKID estimates Markov parameters from arbitrary, noisy, input-output data from any linear system [59]. This is especially useful for experimental data since it is often difficult to directly measure the Markov parameters. The output from OKID is suitable for input to ERA to form a reduced-order model from experimental data [16, 49].

All of these methods are applicable in a broad range of fields, and the goal of the `modred`

library is to provide a flexible, efficient, and scalable package that implements them. The library is written in Python because Python is easy to learn, freely available, and can easily interface with existing code, including compiled C/C++ and Fortran. Users can supply functions that interact with their data (for instance, reading from files), making `modred` compatible with any type of data. `modred` is object-oriented and has been carefully designed, making it easy to use and easy to extend to new algorithms in the future. Automated tests are included for all of its functionality, and the library has already been effectively used on several datasets. The code is parallelized with MPI (using the `mpi4py` library) and demonstrates excellent scalability up to hundreds of processes. A key reason for this is the abstract vector-space approach described in Section 3.2.

3.2 Modal decompositions

In this section, we describe three algorithms for decomposing a given dataset into its dominant components. The inputs to each algorithm are *vectors*, for instance from a set $\{\mathbf{x}_i \in V\}$, where V is a vector space, and the outputs of each are *modes*, for instance from a set $\{\boldsymbol{\varphi}_j \in V\}$, usually computed as a linear combination of the $\mathbf{x}_i$, as

$$\boldsymbol{\varphi}_j = \sum_i [\mathbf{T}]_{i,j} \mathbf{x}_i, \quad j = 1, \ldots, r, \tag{3.1}$$

where $[\mathbf{T}]_{i,j}$ denotes the element in row i and column j of the matrix $\mathbf{T}$. Note that by vector, we do *not* necessarily mean a one-dimensional array—a vector could be a one-dimensional array, multi-dimensional array, or another representation. We use bold symbols to represent vectors and matrices. When $\mathbf{x}_i$ represents a frame of data taken at an instant of time i, as is very often the case, it is referred to as a *snapshot*. Here, we occasionally refer to input vectors as snapshots even when they are not necessarily sampled in time. The resulting modes serve as a low-order basis to approximate a vector via linear combinations.

The three algorithms we discuss produce modes with different properties: Proper Orthogonal Decomposition (POD) extracts the most energetic structures from the data, and the resulting modes are orthogonal; Balanced POD (BPOD) is based on input-output dynamics and produces two sets of modes that are bi-orthogonal; and Dynamic Mode Decomposition (DMD) separates structures by frequency content, producing modes that are not necessarily orthogonal. Since both the snapshots and the modes are elements of a vector space V, we refer to them as vectors.

While the modal decompositions produce different modes, the algorithms to compute the modes share the same general procedure. First, the vector space V is established by defining an inner product, vector addition, and scalar multiplication. (Of course, these operations need to satisfy the properties of a vector space.) Then a decomposition (e.g., a singular value decomposition) is done to compute the matrix of coefficients $\mathbf{T}$ in (3.1). The last step is to form the modes via equation (3.1).

We will show two different approaches to all of the algorithms. The first is based on matrix multiplications and is only suited for smaller and simpler datasets because it requires that all of the vectors be flattened into one-dimensional arrays (mapping V to $\mathbb{R}^n$ or $\mathbb{C}^n$) and be stacked into large data matrices. When the datasets are small enough to be stacked into

matrices, this matrix multiplication is computationally efficient thanks to highly-optimized matrix libraries.

However, sometimes all of the data is too large to fit entirely into a single node's memory simultaneously. For these cases, we adopt a second approach that is based solely on vector space operations and therefore does not require vectors to be flattened into one-dimensional arrays or coerced into *any* particular data structure. Instead, it requires only vector addition, scalar multiplication, and inner products between vectors. Since each operation requires only one or two vectors in memory simultaneously, this approach is ideal for larger and more complicated datasets, and is parallelized for distributed memory architectures via MPI. This vector space approach is mathematically equivalent to the matrix multiplication approach and is not novel from a mathematical point of view. However, what is novel is our recognition that implementing this approach is valuable, and then implementing it in a way that opens model reduction to a wide audience of potential users with varied problems.

In the upcoming sections we describe the steps for finding POD modes using the matrix multiplication and vector space approaches, highlighting the differences and showing code samples. We also explain the steps for finding BPOD and DMD modes, omitting some matrix multiplication steps that easily follow from the description of POD.

3.2.1 POD

Mathematically, the Proper Orthogonal Decomposition (POD) of a dataset $\{\mathbf{x}_i \in V \mid i = 1, \ldots, m_{\mathbf{x}}\}$ arises when solving for the projection $\mathbf{P}_r$ of rank r that minimizes the error

$$\text{error} = \sum_{i=1}^{m_{\mathbf{x}}} \|\mathbf{x}_i - \mathbf{P}_r \mathbf{x}_i\|^2. \tag{3.2}$$

The operation $\|\cdot\|$ is the induced norm from the inner product $\langle \cdot, \cdot \rangle$ on V (linear in the second argument, conjugate linear in the first). The projection in Equation (3.2) is written as

$$\mathbf{P}_r \mathbf{x}_i = \sum_{j=1}^{r} \langle \boldsymbol{\varphi}_j, \mathbf{x}_i \rangle \, \boldsymbol{\varphi}_j, \tag{3.3}$$

where $\boldsymbol{\varphi}_i \in V$ is the orthonormal basis of rank r such that the projection onto them minimizes the error in Equation (3.2). The elements of this basis are called the *POD modes*, and we describe two closely related and mathematically equivalent ways of deriving and computing them in the upcoming paragraphs.

The first way uses matrix representations of the vectors. It is common for a generic vector $\mathbf{x} \in V$ to be represented as a two- or three-dimensional array of data, and if V has dimension n, a vector can be "flattened" into a column vector $\hat{\mathbf{x}} \in \mathbb{R}^n$ (or $\mathbb{C}^n$). (More precisely, the n components of $\hat{\mathbf{x}}$ are the coordinates of $\mathbf{x}$ with respect to some chosen basis of V.) We can easily express the inner product on the flattened vectors as $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \hat{\mathbf{x}}_1^* \mathbf{W} \hat{\mathbf{x}}_2$, where $\mathbf{W}$ is the inner product weighting and, therefore, is a Hermitian positive definite matrix. For many physical applications, the norm is chosen so that minimizing the error corresponds physically to optimally capturing the kinetic energy.

For the present illustrative purposes, we assume $\mathbf{W}$ is the identity so it can be removed from the equations. Minimizing the error in Equation (3.2) by calculus leads to the $n \times n$

eigenvalue problem

$$\mathbf{X}\mathbf{X}^*\mathbf{\Phi} = \mathbf{\Phi}\mathbf{\Sigma} \tag{3.4}$$

where $\mathbf{X}$ has columns $\hat{\mathbf{x}}_i$

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \hat{\mathbf{x}}_1 & \hat{\mathbf{x}}_2 & \dots & \hat{\mathbf{x}}_{m_\mathbf{x}} \\ | & | & & | \end{bmatrix} \tag{3.5}$$

and $\mathbf{\Phi}$ has POD modes $\hat{\boldsymbol{\varphi}}_i$ as columns

$$\mathbf{\Phi} = \begin{bmatrix} | & | & & | \\ \hat{\boldsymbol{\varphi}}_1 & \hat{\boldsymbol{\varphi}}_2 & \dots & \hat{\boldsymbol{\varphi}}_r \\ | & | & & | \end{bmatrix} \tag{3.6}$$

Matrix $\mathbf{\Sigma}$ is diagonal and filled with the corresponding eigenvalues. The first r POD modes, ranked by largest eigenvalues, define $\mathbf{P}_r$ in Equation (3.3).

Methods that involve forming the $n \times n$ matrix $\mathbf{X}\mathbf{X}^*$ are commonly called *direct methods*, and are appropriate to use when n is small. If n is large then Equation (3.4) is computationally expensive to solve, but there is a theoretically equivalent alternative method known as the *method of snapshots* that bypasses forming $\mathbf{X}\mathbf{X}^*$, and instead forms a $m_\mathbf{x} \times m_\mathbf{x}$ matrix and finds its eigenvalues and eigenvectors [103]. The method of snapshots is well-suited for large data for which, typically, $m_\mathbf{x} \ll n$. We summarize the steps to this method below, and no longer assume that $\mathbf{W}$ is identity. For further details see [46, 103].

1. Collect and store vectors $\mathbf{x}_i \in V$, for $i = 1, \dots, m_\mathbf{x}$ from a simulation or experiment.
2. Flatten vectors into columns $\hat{\mathbf{x}}_i \in \mathbb{R}^n$ (or $\mathbb{C}^n$), where n is the dimension of V, and stack them in $\mathbf{X}$, as in Equation (3.5) so $\mathbf{X}$ is $n \times m_\mathbf{x}$.
3. Compute the $m_\mathbf{x} \times m_\mathbf{x}$ correlation matrix via $\mathbf{H} = \mathbf{X}^*\mathbf{W}\mathbf{X}$.
4. Compute the eigenvalues and eigenvectors of $\mathbf{H}$, writing $\mathbf{H}\mathbf{U} = \mathbf{U}\mathbf{\Sigma}$, where $\mathbf{\Sigma}$ is diagonal and real, and $\mathbf{U}$ is orthogonal (or unitary), since $\mathbf{H}$ is symmetric (or Hermitian). Sort the eigenvalues (and corresponding eigenvectors) by largest magnitude.
5. Select the number of modes to keep, r. Truncate the matrices, keeping the first r columns of $\mathbf{U}$ to obtain $\mathbf{U}_r$, and the first r rows and columns of $\mathbf{\Sigma}$ to obtain $\mathbf{\Sigma}_r$.
6. Compute the matrix $\mathbf{T} = \mathbf{U}_r\mathbf{\Sigma}_r^{-1/2}$.
7. Compute the matrix $\mathbf{\Phi} = \mathbf{X}\mathbf{T}$, which has modes $\hat{\boldsymbol{\varphi}}_i$ as columns as in Equation (3.6). Unflatten all $\hat{\boldsymbol{\varphi}}_i \in \mathbb{R}^n$ to obtain modes $\boldsymbol{\varphi}_i \in V$.

The code to use `modred` to find the leading ten POD modes is shown below. The variable `weights` is a 1D or 2D `numpy` array and corresponds to the inner product weight matrix $\mathbf{W}$, `vecs` is a 2D `numpy` array and corresponds to $\mathbf{X}$ that is loaded from a text file. The variables `eig_vecs`, `eig_vals`, and `modes` correspond to $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{\Phi}$, respectively, and all are `numpy` arrays.

```
vecs = numpy.loadtxt('vec_array.txt')
modes, sing_vals = modred.compute_POD_matrices_snap_method(vecs, range(10),
    inner_product_weights=weights)
```

When the data are small enough, `modred` can use a variation on the direct method that involves taking a singular value decomposition (SVD) of $\mathbf{X}$ directly. This has the numerical advantage of not "squaring up" the eigenvalues, which can cause numerical roundoff errors in modes which correspond to very small eigenvalues. In some cases, these modes are important, and so we provide another function which implements directly taking the SVD of $\mathbf{X}$, `compute_POD_matrices_direct_method`. (See the online documentation for more information on using this function.)

Now we present the vector space approach, which eliminates the need to flatten the vectors and stack them as columns of $\mathbf{X}$. In fact, the matrix $\mathbf{X}$ is never formed.

1. Collect and store vectors $\mathbf{x}_i \in V$, for $i = 1, \ldots, m_{\mathbf{x}}$ from simulations or experiments.
2. Compute each entry of the $m_{\mathbf{x}} \times m_{\mathbf{x}}$ correlation matrix $\mathbf{H}$ via $[\mathbf{H}]_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$.
3. Compute the eigenvalues and eigenvectors of $\mathbf{H}$, writing $\mathbf{HU} = \mathbf{U\Sigma}$, where $\mathbf{\Sigma}$ is diagonal and real, and $\mathbf{U}$ is orthogonal (or unitary), since $\mathbf{H}$ is symmetric (or Hermitian). Sort the eigenvalues (and corresponding eigenvectors) in descending order.
4. Select the number of modes to keep, r. Truncate the matrices, keeping the first r columns of $\mathbf{U}$ to obtain $\mathbf{U}_r$, and the first r rows and columns of $\mathbf{\Sigma}$ to obtain $\mathbf{\Sigma}_r$.
5. Compute the matrix $\mathbf{T} = \mathbf{U}_r \mathbf{\Sigma}_r^{-1/2}$.
6. Construct modes $\boldsymbol{\varphi}_j$ individually via

$$\boldsymbol{\varphi}_j = \sum_{i=1}^{m_{\mathbf{x}}} \mathbf{x}_i \, [\mathbf{T}]_{i,j}, \qquad j = 1, \ldots, r. \tag{3.7}$$

The code to find the leading ten POD modes is shown below. The first argument to `PODHandles` is a function (callable) that takes two vectors and returns their inner product. The variables `vec_handles` and `mode_handles` are lists of so-called vector handles which are responsible for loading and saving vectors to disk. In this example, the vector handles would load data from Python's "pickle" binary format. Vector handles use very little memory and allow `modred` to load/save the large vectors so they are in memory only as they are needed. This is crucial for this size of data where not all vectors can be in memory simultaneously.

```
vec_handles = [PickleVecHandle('vec%d.fmt' % i) for i in range(100)]
mode_handles = [PickleVechandle('mode%d.fmt' % i) for i in range(10)]
POD = modred.PODHandles(inner_product)
eig_vecs, eig_vals = POD.compute_decomp(vec_handles)
POD.compute_modes(range(10), mode_handles)
```

The most important difference between the two procedures is that the vector space approach involves no flattening and stacking of vectors. As a result, in step 3 of the matrix multiplication, the inner products involve the data matrix $\mathbf{X}$, but in step 2 of the vector space approach the inner products are computed one-by-one without data matrices. Similarly, step 7 of the matrix multiplication method results in a large matrix $\mathbf{\Phi}$ of modes, but step 6 of the vector space method finds each mode independently via scalar multiplication and vector addition.

These two approaches each have advantages and disadvantages, and so `modred` includes implementations of both. As previously mentioned, for smaller and simpler datasets, the matrix multiplication approach and corresponding `PODArrays` class is both more computationally efficient and easier to use. For larger and more complicated datasets, the vector space approach and corresponding `PODHandles` class is easier to use and is parallelized. See Section 3.4.1 and Section 3.5 for an in-depth discussion of these topics.

3.2.2 Balanced POD

As mentioned in the introduction, Balanced POD (BPOD) is useful for finding reduced-order models of large linear input-output systems. A general form of a discrete-time linear system is

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{A}\mathbf{x}_i + \mathbf{B}\mathsf{u}_i \\ \mathsf{y}_i &= \mathbf{C}\mathbf{x}_i \end{aligned} \tag{3.8}$$

where $\mathbf{x}_i \in V$ is the state vector, $\mathsf{u}_i \in \mathcal{U}$ is regarded as an input, and $\mathsf{y}_i \in \mathcal{Y}$ as an output. Here, $\mathcal{U}$ and $\mathcal{Y}$ are vector spaces with respective dimensions p and q (i.e., p inputs and q outputs), and $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are linear operators between the appropriate spaces. As is often the case for large systems, BPOD is most useful when p and q are much less than n (the dimension of V). The above represents a discrete-time formulation, but the equations are easily adapted to a continuous-time formulation, and the `modred` library handles both.

BPOD is an approximation of balanced truncation [82], a model reduction procedure that linearly transforms the state $\mathbf{x}_i$ such that the new coordinates are ranked in decreasing order of importance to the input-output dynamics. Truncating the lower ranked states results in a system that closely approximates the original system. We begin describing balanced truncation by defining the adjoint system that corresponds to the system in Equation (3.8),

$$\begin{aligned} \mathbf{z}_{i+1} &= \mathbf{A}^\dagger \mathbf{z}_i + \mathbf{C}^\dagger \mathbf{v}_i \\ \mathsf{w}_i &= \mathbf{B}^\dagger \mathbf{z}_i, \end{aligned} \tag{3.9}$$

where $\mathbf{z}_i \in V$, $\mathbf{v}_i \in \mathcal{Y}$, and $\mathsf{w}_i \in \mathcal{U}$. The adjoint operator, denoted $(\cdot)^\dagger$, depends on the inner product in the appropriate spaces: for instance, if $V = \mathbb{R}^n$, $\mathcal{U} = \mathbb{R}^p$, and $\mathcal{Y} = \mathbb{R}^q$ and the inner product on V is $\langle \mathbf{z}, \mathbf{x} \rangle = \mathbf{z}^* \mathbf{W} \mathbf{x}$, where $\mathbf{W}$ is Hermitian positive definite and

$$\mathbf{A}^\dagger = \mathbf{W}^{-1}\mathbf{A}^*\mathbf{W} \qquad \mathbf{B}^\dagger = \mathbf{B}^*\mathbf{W} \qquad \mathbf{C}^\dagger = \mathbf{W}^{-1}\mathbf{C}^*. \tag{3.10}$$

The balanced truncation algorithm takes the controllability and observability Gramians, $\mathbf{W}_c$ and $\mathbf{W}_o$, as inputs. As for POD, we represent $\mathbf{x}_i$ and $\mathbf{z}_i$ as column matrices $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{z}}_i$. The controllability Gramians is used to quantify the controllability of a particular state $\hat{\mathbf{x}}$, i.e., how much the state is excited by previous inputs, via $\hat{\mathbf{x}}^* \mathbf{W}_c \hat{\mathbf{x}}$. Analogously, the observability of a particular state, i.e., how much the state excites future outputs, is given by $\hat{\mathbf{x}}^* \mathbf{W}_o \hat{\mathbf{x}}$.

The Gramians can be computed by solving the corresponding Lyapunov equations. However, this can be computationally expensive, and another way is to approximate them empirically. To approximate them, we take snapshots from the impulse responses for each input for the systems in (3.8) and (3.9). We collect a total of $m_{\mathbf{x}}$ "direct" (non-adjoint) snapshots

$\mathbf{x}_i$ for $i = 1, 2, \ldots, m_{\mathbf{x}}$ and $m_{\mathbf{z}}$ adjoint snapshots $\mathbf{z}_i$ for $i = 1, 2, \ldots, m_{\mathbf{z}}$. Therefore there are $m_{\mathbf{x}}/p$ snapshots from each direct impulse response and $m_{\mathbf{z}}/q$ from each adjoint impulse response. The vectors are stacked to form $\mathbf{X}$ as in Equation (3.5) and, analogously, define an $n \times m_{\mathbf{z}}$ matrix $\mathbf{Z}$ that has columns $\hat{\mathbf{z}}_i$. The Gramians can be written in terms of these matrices as

$$\mathbf{W}_c = \mathbf{X}\mathbf{X}^\dagger \qquad \mathbf{W}_o = \mathbf{Z}\mathbf{Z}^\dagger. \tag{3.11}$$

Linearly transforming the state as $\hat{\mathbf{x}} = \mathbf{T}\hat{\mathbf{x}}_T$ also transforms the Gramians

$$\mathbf{W}_c \mapsto \mathbf{T}^{-1}\mathbf{W}_c(\mathbf{T}^{-1})^\dagger \qquad \mathbf{W}_o \mapsto \mathbf{T}^\dagger\mathbf{W}_o\mathbf{T}. \tag{3.12}$$

The *balancing* transformation is the transformation $\mathbf{T}$ that makes the Gramians equal and diagonal, $\mathbf{W}_c = \mathbf{W}_o = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ where $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$. (The diagonal entries are called the *Hankel singular values*). Such a balancing transformation is only possible if the system in Equation (3.8) is both controllable and observable. The transformation is found by solving the eigenvalue problem

$$\mathbf{W}_c\mathbf{W}_o\mathbf{T} = \mathbf{T}\boldsymbol{\Sigma} \tag{3.13}$$

with appropriately scaled eigenvectors, as explained in [26] (Proposition 4.7).

Truncating the resulting system keeps only those elements of the balanced state $\hat{\mathbf{x}}_{\mathbf{T}}$ that are most controllable and observable and thus significant in the input-output dynamics. The truncation is trivial; the first r elements of $\hat{\mathbf{x}}_{\mathbf{T}}$ and corresponding components of the balanced versions of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are retained and the rest discarded. The truncated reduced-order system has *a priori* error bounds that are generally very low—not much higher than the theoretical limit for any reduced-order system [26].

Unfortunately, for systems with a large state dimension n it becomes computationally prohibitive to compute the Gramians and solve the eigenvalue problem in Equation (3.13). BPOD approximates the reduced-order system given by balanced truncation using the method of snapshots [103]. Instead of solving the $n \times n$ eigenvalue problem in Equation (3.13), we find the SVD of the $m_{\mathbf{z}} \times m_{\mathbf{x}}$ Hankel matrix

$$\mathbf{H} = \mathbf{Z}^\dagger\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*, \tag{3.14}$$

which is computationally cheaper when $m_{\mathbf{z}}$ and $m_{\mathbf{x}}$ are smaller than n, as is typically the case. The diagonal matrix $\boldsymbol{\Sigma}$ is filled with the largest Hankel singular values. Two sets of bi-orthonormal modes, the direct (or primal) and adjoint modes, are formed via

$$\boldsymbol{\Phi} = \mathbf{X}\mathbf{V}_r\boldsymbol{\Sigma}_r^{-1/2} \qquad \boldsymbol{\Psi} = \mathbf{Z}\mathbf{U}_r\boldsymbol{\Sigma}_r^{-1/2} \tag{3.15}$$

where $\mathbf{V}_r$ and $\mathbf{U}_r$ are $\mathbf{V}$ and $\mathbf{U}$ with only the first r columns and $\boldsymbol{\Sigma}_r$ is $\boldsymbol{\Sigma}$ with only the first r rows and columns. The bi-orthonormality implies that $\boldsymbol{\Psi}^\dagger\boldsymbol{\Phi} = \mathbf{I}_r$. The i^{th} columns of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ correspond to modes, $\boldsymbol{\varphi}_i$ and $\boldsymbol{\psi}_i$, respectively. These modes are the states that are most controllable and observable, and they are ranked by index. To form the reduced-order model, the system in Equation (3.8) is projected onto these modes as described in Section 3.3.1.

Now we describe the procedure for the vector space approach for computing BPOD modes. This procedure is mathematically equivalent to the matrix approach shown above, but, as for POD, has advantages when implemented for larger datasets. For more details on BPOD, see [96].

1. Collect and store snapshots $\mathbf{x}_j$, for $j = 1, \ldots, m_{\mathbf{x}}$, and $\mathbf{z}_i$, for $i = 1, \ldots, m_{\mathbf{z}}$ from simulations.
2. Compute each entry of the Hankel matrix $\mathbf{H}$ via $[\mathbf{H}]_{i,j} = \langle \mathbf{z}_i, \mathbf{x}_j \rangle$.
3. Compute the singular value decomposition (SVD) $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.
4. Select the number of modes to keep, r. Truncate the matrices, keeping the first r columns of $\mathbf{U}$ and $\mathbf{V}$ to obtain $\mathbf{U}_r$ and $\mathbf{V}_r$, and the first r rows and columns of $\mathbf{\Sigma}$ to obtain $\mathbf{\Sigma}_r$.
5. Compute the matrices $\mathbf{T}_{\mathbf{x}} = \mathbf{V}_r \mathbf{\Sigma}_r^{-1/2}$ and $\mathbf{T}_{\mathbf{z}} = \mathbf{U}_r \mathbf{\Sigma}_r^{-1/2}$.
6. Find the direct modes via $\boldsymbol{\varphi}_j = \sum_{i=1}^{m_{\mathbf{x}}} \mathbf{x}_i [\mathbf{T}_{\mathbf{x}}]_{i,j}$ and the adjoint modes via $\boldsymbol{\psi}_j = \sum_{i=1}^{m_{\mathbf{z}}} \mathbf{z}_i [\mathbf{T}_{\mathbf{z}}]_{i,j}$ for $j = 1, \ldots, r$.

The code to perform BPOD with this approach is shown below. The primary difference from the POD case is there are now two sets of vector handles and mode handles that correspond to $\mathbf{x}_i$, $\mathbf{z}_i$, $\boldsymbol{\varphi}_i$, and $\boldsymbol{\psi}_i$.

```
BPOD = BPODHandles(inner_product)
L_sing_vecs, sing_vals, R_sing_vecs = BPOD.compute_decomp(
    direct_vec_handles, adjoint_vec_handles)
BPOD.compute_direct_modes(range(10), direct_mode_handles)
BPOD.compute_adjoint_modes(range(10), adjoint_mode_handles)
```

3.2.3 DMD

Dynamic Mode Decomposition (DMD) is useful for analyzing a series of data uniformly sampled in time, $\mathbf{x}_i = \mathbf{x}(t_i)$ for $i = 1, \ldots, m_{\mathbf{x}}$ and $t_i = i\triangle t$. The DMD modes $\boldsymbol{\varphi}_i$ and Ritz values λ_i are vectors and complex numbers, respectively, such that

$$\begin{aligned} \mathbf{x}_j &= \sum_{i=1}^{m_{\mathbf{x}}-1} \lambda_i^{j-1} \boldsymbol{\varphi}_i \qquad && j = 1, \ldots, m_{\mathbf{x}} - 1 \\ \mathbf{x}_{m_{\mathbf{x}}} &= \sum_{i=1}^{m_{\mathbf{x}}-1} \lambda_i^{m_{\mathbf{x}}-1} \boldsymbol{\varphi}_i + r \qquad && r \perp \operatorname{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_{m_{\mathbf{x}}-1}\}. \end{aligned} \tag{3.16}$$

If the vectors $\{\mathbf{x}_i \mid i = 1 \ldots, m_{\mathbf{x}}\}$ are linearly independent, such $\boldsymbol{\varphi}_i$ and λ_i always exist, and are unique [18]. We observe that the above equation is the discrete-time equivalent of an eigenvector decomposition for the evolution of a linear system, with all multiplicative constants subsumed in the scaling of the modes $\boldsymbol{\varphi}_i$. As such, the Ritz values λ_i tell us the growth rate and frequency associated with each mode. For linear systems, this information can be used to perform a stability analysis. For nonlinear systems, we can identify oscillatory structures on an attractor. DMD is not used elsewhere in this thesis and so we do not go into more details about the theory. See the works cited for examples [97, 100].

The algorithm for computing the DMD modes and Ritz values can be implemented in a way that is closely related to the POD algorithm [100]. Previous work described a new

variant that is especially suitable for our library due to its low memory requirements [111]. Here, we improve upon this algorithm by further eliminating unnecessary inner products and linear combinations (see App. A for details). Our implementation of the vector space approach to the algorithm is summarized below. The matrix multiplication approach is very similar to that of POD (both via the method of snapshots and direct SVD), and so is omitted.

1. Collect and store snapshots $\mathbf{x}_i$, for $i = 1 \ldots, m_{\mathbf{x}}$ from simulations or experiments.
2. Compute each entry of the correlation matrix via $[\mathbf{H}]_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, using all but the last snapshot (i and j have range $1, \ldots, m_{\mathbf{x}} - 1$).
3. Compute the eigenvalues and eigenvectors of $\mathbf{H}$, writing $\mathbf{HU} = \mathbf{U\Sigma}$, where $\mathbf{\Sigma}$ is diagonal and real, and $\mathbf{U}$ is orthogonal (or unitary) since $\mathbf{H}$ is symmetric (or Hermitian). Sort the eigenvalues (and corresponding eigenvectors) in descending order.
4. Define sub-matrix $\mathbf{H}' = [\mathbf{H}]_{1:m_{\mathbf{x}}-1,\, 2:m_{\mathbf{x}}-1}$ (i.e., $\mathbf{H}$ with the first column removed), making use of the previously computed correlation matrix $\mathbf{H}$.
5. Form the column matrix $\mathbf{H}''$, via $[\mathbf{H}'']_j = \langle \mathbf{x}_{m_{\mathbf{x}}}, \mathbf{x}_j \rangle$, for $j = 1, \ldots, m_{\mathbf{x}} - 1$.
6. Compute $\mathbf{M} = \mathbf{\Sigma}^{-1/2}\mathbf{U}^* \begin{bmatrix} \mathbf{H}' & \mathbf{H}'' \end{bmatrix} \mathbf{U\Sigma}^{-1/2}$. Solve the eigenvalue problem $\mathbf{MV} = \mathbf{V\Lambda}$, where the diagonal entries of $\mathbf{\Lambda}$ are the Ritz values λ_i.
7. Compute the matrix $\mathbf{T} = \mathbf{U\Sigma}^{-1/2}\mathbf{VD}$, where $\mathbf{D}$ is a diagonal matrix with diagonal $\mathbf{d} = (\mathbf{V}^*\mathbf{V})^{-1}\mathbf{V}^*\mathbf{\Sigma}^{-1/2}\mathbf{U}^*\mathbf{H}_{[1:m_{\mathbf{x}}-1,1]}$.
8. Construct modes individually via $\boldsymbol{\varphi}_j = \sum_{i=1}^{m_{\mathbf{x}}} \mathbf{x}_i \, [\mathbf{T}_{\mathbf{x}}]_{i,j}$.

The code to use `modred` to perform DMD in this fashion is shown below. The variables are analogous to the POD case, with the exception of `ritz_vals`, `mode_norms`, and `build_coeffs`, which are all `numpy` arrays and correspond to λ_i, $\|\boldsymbol{\varphi}_i\|$, and $\mathbf{T}$, respectively.

```
DMD = modred.DMDHandles(inner_product)
ritz_vals, mode_norms, build_coeffs = DMD.compute_decomp(vec_handles)
my_DMD.compute_modes(range(10), mode_handles)
```

3.3 Reduced-order models and system identification

3.3.1 Petrov-Galerkin projection for linear systems

In the case that the snapshots are taken from a linear time invariant (LTI) system such as (3.8), a reduced-order model can be found by projecting the full dynamics onto the reduced set of modes via Petrov-Galerkin projection. We outline this technique now, first for the matrix approach.

First, we approximate the full state vector $\mathbf{x} \in V$ as a linear combination of the modes $\boldsymbol{\varphi}_j$, denoted as follows:

$$\mathbf{x} \approx \sum_{j=1}^{r} q_j \boldsymbol{\varphi}_j = \mathbf{\Phi q}, \tag{3.17}$$

where $\mathbf{q} = (q_1, \ldots, q_r) \in \mathbb{R}^r$ and $\boldsymbol{\Phi} : \mathbb{R}^r \to V$. $\boldsymbol{\Phi}$ is, again, a matrix whose columns are the modes $\hat{\boldsymbol{\varphi}}_i$ and $\hat{(\cdot)}$ denotes the representation of the mode as a one-dimensional vector in $\mathbb{R}^n$.

A second set of modes is needed, $\boldsymbol{\Psi}$, which has columns $\hat{\boldsymbol{\psi}}_i$ and has the property $\boldsymbol{\Psi}^\dagger : V \to \mathbb{R}^r$, i.e.,

$$[\boldsymbol{\Psi}^\dagger \mathbf{x}]_j = \langle \boldsymbol{\psi}_j, \mathbf{x} \rangle, \quad j = 1, \ldots, r. \tag{3.18}$$

In POD and DMD, the second set of modes is the same as the first set, $\boldsymbol{\Phi} = \boldsymbol{\Psi}$. In BPOD, the second set of modes is the set of adjoint modes. Inserting (3.17) into (3.8) and multiplying by $\boldsymbol{\Psi}^\dagger$ then gives the reduced-order model

$$\begin{aligned} \mathbf{q}_{i+1} &= \mathbf{A}_r \mathbf{q}_i + \mathbf{B}_r \mathsf{u}_i \\ \mathsf{y}_i &= \mathbf{C}_r \mathbf{q}_i + \mathbf{D} \mathsf{u}_i, \end{aligned} \tag{3.19}$$

where

$$\mathbf{A}_r \equiv (\boldsymbol{\Psi}^\dagger \boldsymbol{\Phi})^{-1} \boldsymbol{\Psi}^\dagger \mathbf{A} \boldsymbol{\Phi}, \qquad \mathbf{B}_r \equiv (\boldsymbol{\Psi}^\dagger \boldsymbol{\Phi})^{-1} \boldsymbol{\Psi}^\dagger \mathbf{B}, \qquad \mathbf{C}_r \equiv \mathbf{C} \boldsymbol{\Phi}, \tag{3.20}$$

and $\mathbf{A}_r : \mathbb{R}^r \to \mathbb{R}^r$ is an $r \times r$ matrix, $\mathbf{B}_r : \mathcal{U} \to \mathbb{R}^r$, and $\mathbf{C}_r : \mathbb{R}^r \to \mathcal{Y}$. For instance, if $\mathcal{U} = \mathbb{R}^p$, then $\mathbf{B}_r$ is an $r \times p$ matrix, and similarly for $\mathbf{C}_r$. Also note that for POD and BPOD, the matrix $\boldsymbol{\Psi}^\dagger \boldsymbol{\Phi}$ is simply the identity matrix because the modes are orthonormal (for POD) or bi-orthogonal (for BPOD). In cases like these, a flag can be set in `modred` to avoid the unnecessary computation of this matrix and its inverse.

Below we summarize the steps to form the reduced-order matrices in (3.19) using the vector space approach, where the input space $\mathcal{U} = \mathbb{R}^p$ and the output space $\mathcal{Y} = \mathbb{R}^q$:

1. Advance each balancing mode, $\boldsymbol{\varphi}_i$, one time step, resulting in $\boldsymbol{\varphi}'_i = \mathbf{A}\boldsymbol{\varphi}_i$.
2. If doing a non-orthonormal projection, then compute $[\boldsymbol{\Psi}^\dagger \boldsymbol{\Phi}]_{i,j} = \langle \boldsymbol{\psi}_i, \boldsymbol{\varphi}_j \rangle$ via individual inner products, then invert it to obtain the matrix $(\boldsymbol{\Psi}^\dagger \boldsymbol{\Phi})^{-1}$.
3. Compute matrix $\mathbf{A}_r$ by first computing $\mathbf{M}$ where $[\mathbf{M}]_{i,j} = \langle \boldsymbol{\psi}_i, \boldsymbol{\varphi}'_j \rangle$. Then $\mathbf{A}_r = (\boldsymbol{\Psi}^\dagger \boldsymbol{\Phi})^{-1} \mathbf{M}$.
4. Compute matrix $\mathbf{B}_r$ by first computing $\mathbf{M}$ where $[\mathbf{M}]_{i,j} = \langle \boldsymbol{\psi}_i, \mathbf{b}_j \rangle$ and $\mathbf{b}_j$ denotes the j^{th} column of $\mathbf{B}$ (in the abstract setting, $\mathbf{b}_j = \mathbf{B}\mathbf{e}_j$, where $\mathbf{e}_j$ is a standard basis vector of $\mathbb{R}^p$). Then $\mathbf{B}_r = (\boldsymbol{\Psi}^\dagger \boldsymbol{\Phi})^{-1} \mathbf{M}$.
5. Compute matrix $\mathbf{C}_r$, where the j^{th} column is $\mathbf{C}\boldsymbol{\varphi}_j$.

Code to compute the Petrov-Galerkin projection onto a set of modes with the vector space approach is shown in the code below. As in previous examples, `inner_product` is a function to take the inner product of two vectors. The variable `mode_handles[i]` corresponds to the mode $\boldsymbol{\varphi}_i$, `A_on_modes_handles[i]` to $\mathbf{A}\boldsymbol{\varphi}_i$, `B_on_basis_handles[i]` to $\mathbf{b}_i$, and `C_on_modes[j]` to $\mathbf{C}\boldsymbol{\varphi}_j$. To do so with the matrix multiplication approach is very similar and omitted.

```
Galerkin_proj = modred.LTIGalerkinProjectionHandles(
    inner_product, mode_handles)
A_reduced, B_reduced, C_reduced = Galerkin_proj.compute_model(
    A_on_mode_handles, B_on_basis_handles, C_on_modes)
```

3.3.2 OKID

The Observer/Kalman Filter Identification (OKID) estimates the Markov parameters of any system based on arbitrary and noisy input-output signals [59], and has been used in a wide range of applications. The input-output dynamics of a linear system are completely characterized by its Markov parameters, and so these are useful for other system identification methods, including the Eigensystem Realization Algorithm discussed in the next section. For the system (3.8), where we now let $\mathcal{U} = \mathbb{R}^p$ and $\mathcal{Y} = \mathbb{R}^q$, the Markov parameters, $\{\mathbf{M}_i \mid i = 1, \ldots, m\}$, are defined as

$$\mathbf{M}_0 = \mathbf{D}, \qquad \mathbf{M}_i = \mathbf{C}\mathbf{A}^{i-1}\mathbf{B} \qquad \text{for} \qquad i = 1, 2, \ldots, m. \tag{3.21}$$

The $(r, c)^{\text{th}}$ entry of $\mathbf{M}_i$ corresponds to the r^{th} output and c^{th} input and each $\mathbf{M}_i$ has dimensions $q \times p$. For large systems, the Markov parameters can be computed efficiently as impulse responses to each input individually.

Given arbitrary input-output signals, u_i and y_i, for $i = 1, \ldots, m$, a misguided approach would be to solve the matrix problem

$$\begin{bmatrix} \mathsf{y}_0 & \mathsf{y}_1 & \ldots & \mathsf{y}_m \end{bmatrix} = \begin{bmatrix} \mathbf{M}_0 & \mathbf{M}_1 & \ldots & \mathbf{M}_m \end{bmatrix} \underbrace{\begin{bmatrix} \mathsf{u}_0 & \mathsf{u}_1 & \ldots & \mathsf{u}_m \\ & \mathsf{u}_0 & \ldots & \mathsf{u}_{m-1} \\ & & \ddots & \vdots \\ & & & \mathsf{u}_0 \end{bmatrix}}_{\mathbf{U}}. \tag{3.22}$$

There are several reasons directly solving this equation is impractical. First, the matrices may be ill-conditioned. Secondly, the noise is not filtered and can significantly reduce accuracy. Lastly, it may require a long time-series of data to accurately capture the Markov parameters, increasing the computational cost.

OKID addresses all of these drawbacks. In short, the system is augmented with an asymptotically stable Kalman filter. One solves for the Markov parameters of this modified system, $\{\bar{\mathbf{M}}_i \mid i = 1, \ldots, m\}$, and then for the Markov parameters of the original system. The matrix equation which must be solved, rather than (3.22), is

$$\begin{bmatrix} \mathsf{y}_0 & \mathsf{y}_1 & \ldots \mathsf{y}_m \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{M}}_0 & \bar{\mathbf{M}}_1 & \ldots & \bar{\mathbf{M}}_m \end{bmatrix} \mathbf{V} \tag{3.23}$$

where $\mathbf{V}$ is defined later in (3.24). We reproduce the outline of the procedure below. For more details see [57].

1. Collect inputs $\{\mathsf{u}_i \mid i = 1, \ldots, m\}$ and outputs $\{\mathsf{y}_i \mid i = 1, \ldots, m\}$ as in (3.22).
2. Select the number of Markov parameters to estimate, s.
3. Construct the $\mathbf{V}$ matrix as defined below

$$\mathbf{V} = \begin{bmatrix} \mathsf{u}_0 & \mathsf{u}_1 & \mathsf{u}_2 & \ldots & \mathsf{u}_s & \ldots & \mathsf{u}_m \\ & \mathbf{v}_0 & \mathbf{v}_1 & \ldots & \mathbf{v}_{s-1} & \ldots & \mathbf{v}_{m-1} \\ & & \mathbf{v}_0 & \ldots & \mathbf{v}_{s-2} & \ldots & \mathbf{v}_{m-2} \\ & & & \ddots & \vdots & & \vdots \\ & & & & \mathbf{v}_0 & \ldots & \mathbf{v}_{m-s} \end{bmatrix}, \tag{3.24}$$

where $\mathbf{v}_i = \begin{bmatrix} \mathtt{u}_i \\ \mathtt{y}_i \end{bmatrix}$, column vectors with dimensions $(p+q)\times 1$. Matrix $\mathbf{V}$ has dimensions $(p + (p + q)s) \times (m + 1)$.

4. Solve, via least squares, for the Markov parameters of observer system (transpose of (3.23))

$$\mathbf{V}^T \begin{bmatrix} \bar{\mathbf{M}}_0 & \bar{\mathbf{M}}_1 & \dots & \bar{\mathbf{M}}_m \end{bmatrix}^T = \begin{bmatrix} \mathtt{y}_0 & \mathtt{y}_1 & \dots \mathtt{y}_m \end{bmatrix}^T \tag{3.25}$$

5. Solve for the Markov parameters of original system:

$$\mathbf{M}_0 = \mathbf{D} = \bar{\mathbf{M}}_0, \qquad \mathbf{M}_i = \bar{\mathbf{M}}_i^{(1)} - \sum_{k=1}^{i} \bar{\mathbf{M}}_k^{(2)} \mathbf{M}_{i-k} \text{ for } k = 1, 2, \dots, s$$

where for $i \geq 1$ the Markov parameters of the observer system are composed of two submatrices, $\bar{\mathbf{M}}_i = \begin{bmatrix} \bar{\mathbf{M}}_i^{(1)} & -\bar{\mathbf{M}}_i^{(2)} \end{bmatrix}$. Submatrix $\bar{\mathbf{M}}_i^{(1)}$ has dimensions $q \times p$ and $\bar{\mathbf{M}}_i^{(2)}$ has dimensions $q \times q$.

The code to estimate Markov parameters is shown below. The variable `Markovs` corresponds to $\{\mathbf{M}_i \mid i = 1, \dots, m\}$ and is a three-dimensional array with dimensions that correspond to the time-step, output, and input so that `Markovs[i]` is $\mathbf{M}_i$. Variables `inputs` and `outputs` are two-dimensional arrays such that `inputs[i]` is $\mathtt{u}_i$ and `outputs[i]` is $\mathtt{y}_i$. The last argument to `OKID` is m, the number of Markov parameters to estimate.

```
Markovs = modred.OKID(inputs, outputs, 20)
```

3.3.3 ERA

The Eigensystem Realization Algorithm (ERA) is a method for finding reduced-order input-output models from a system's Markov parameters [43, 58]. The resulting models are theoretically equivalent to those from BPOD, and therefore when a sufficiently long series of Markov parameters is taken the models are balanced [78]. The states are ordered by input-output importance and can be truncated just as in BPOD. However, ERA does not require adjoint vectors, full-state information, or the computation of modes, thereby reducing the computation time by orders of magnitude. This also makes ERA useful for finding linear input-output models from estimated Markov parameters (e.g., from OKID and experimental data) and from expensive, high-dimensional simulations. The linear models are amenable to control design.

The procedure to compute ERA models is summarized below. The Hankel matrix $\mathbf{H}$ is theoretically the same as in BPOD.

1. Collect $2N$ Markov parameters (N pairs) in the following series

$$\mathbf{M}_1,\ \mathbf{M}_2,\ \mathbf{M}_{P+1},\ \mathbf{M}_{P+2},\ \dots,\ \mathbf{M}_{(N-1)P+1},\ \mathbf{M}_{(N-1)P+2} \tag{3.26}$$

where $\mathbf{M}_i$ is defined in (3.21). The integer P is the number of time steps between pairs of Markov parameters.

2. Construct (generalized) Hankel matrices, $\mathbf{H}$ and $\mathbf{H}'$, where

$$\mathbf{H} = \begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_{P+1} & \dots & \mathbf{M}_{m_c+1} \\ \mathbf{M}_{P+1} & \mathbf{M}_{2P+1} & \dots & \mathbf{M}_{(m_c+1)P+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{M}_{m_oP+1} & \mathbf{M}_{(m_o+1)P+1} & \dots & \mathbf{M}_{(m_c+m_o)P+1} \end{bmatrix} \tag{3.27}$$

and

$$\mathbf{H}' = \begin{bmatrix} \mathbf{M}_2 & \mathbf{M}_{P+2} & \dots & \mathbf{M}_{m_c+2} \\ \mathbf{M}_{P+2} & \mathbf{M}_{2P+2} & \dots & \mathbf{M}_{(m_c+1)P+2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{M}_{m_oP+2} & \mathbf{M}_{(m_o+1)P+2} & \dots & \mathbf{M}_{(m_c+m_o)P+2} \end{bmatrix}. \tag{3.28}$$

The parameters m_c and m_o determine how much relative weight is placed on controllability and observability, respectively. Often one uses all data and lets m_c be equal to m_o, i.e., $m_c = m_o = (N-1)/2$.

3. Compute the SVD such that $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.

4. Select order of reduced-order model, r. Truncate the matrices, keeping the first r columns of $\mathbf{U}$ and $\mathbf{V}$ to obtain $\mathbf{U}_r$ and $\mathbf{V}_r$, and the first $r \times r$ block of $\mathbf{\Sigma}$ to obtain $\mathbf{\Sigma}_r$.

5. Compute reduced system matrices:

$$\begin{aligned} \mathbf{A}_r &= \mathbf{\Sigma}_r^{-1/2}\mathbf{U}_r^*\mathbf{H}'\mathbf{V}_r\mathbf{\Sigma}_r^{-1/2} \\ \mathbf{B}_r &= \text{the first } p \text{ columns of } \mathbf{\Sigma}_r^{1/2}\mathbf{V}_r^* \\ \mathbf{C}_r &= \text{the first } q \text{ rows of } \mathbf{U}_r\mathbf{\Sigma}_r^{1/2}. \end{aligned} \tag{3.29}$$

The reduced system is then the same as in (3.19).

Code to compute an ERA reduced-order model is shown. `A_reduced`, `B_reduced`, and `C_reduced` correspond to $\mathbf{A}_r$, $\mathbf{B}_r$, and $\mathbf{C}_r$. The argument `Markovs` is the same format as in OKID, a three-dimensional `numpy` array such that `Markovs[i]` is $\mathbf{M}_i$. The singular values, $\mathbf{\Sigma}$, are stored in the variable `sing_vals`.

```
ERA = modred.ERA()
A_reduced, B_reduced, C_reduced = ERA.compute_model(Markovs, 50)
sing_vals = ERA.sing_vals
```

3.4 Software design

3.4.1 Use with data of different size and complexity

Careful thought has been given to design of `modred` and the types of users and problems for which it is used. We distinguish between three cases, delineated by the size of the dataset. The first case has datasets which are smaller, meaning that they fit on a single

node, and simpler, in terms of inner products and data format. While not always true, we observe that small data and simple data often coincide, and so we consider them one case. For example, such datasets might be generated from experimental measurements or small simulations on a personal computer. Here, the matrix multiplication approach has a few advantages. Since the data fits in memory, we stack it all into data matrices and use matrix multiplication for inner products, taking advantage of the efficiency of highly optimized libraries such as BLAS, LAPACK, and Intel MKL. The added complexity of defining an inner product weighting and flattening the data into columns of a matrix is often not difficult or computationally demanding for this type of data. The classes which implement this approach are not parallelized for distributed memory and are most efficient when the `numpy` backend makes use of multiple cores.

In the second case, the dataset is larger so that data matrices such as $\mathbf{X}$ and $\mathbf{\Phi}$ are too large to fit in memory and thus a direct implementation of the steps (as in the POD section 3.2.1) is not possible. Still, the individual vectors are small enough that a few can be in a single node's memory simultaneously. Often the inner product is more complicated because the data might be multi-dimensional, defined over a non-uniform grid, and generated by a large simulation performed on computer cluster. Similarly, the data may be saved in some more complicated format. For this case, the vector space approach is ideal because each step's computations require only two individual vectors in memory simultaneously. Additionally, it preserves the mathematical abstraction between vectors (represented as objects in the sense of object-oriented programming) and functions which operate on vectors. In this approach, `modred` operates on vector objects, independent of the underlying implementation, resulting in high-level, general, modular, testable, code that is easily applied to any data format. Further, since the inner products and linear combinations require only a few vectors in memory at once, we are able to parallelize the computations for distributed memory architectures and achieve excellent scaling (see the parallelization section 3.5). It is possible to use this approach for smaller data as well, but we find it can be significantly slower than the matrix multiplication implementation.

The third and final case is datasets with vectors too large to fit in a single node's memory, and it is not yet available in `modred`. See section 3.5.4 for more details.

3.4.2 Library-wide design principles

We incorporate other elements of software design throughout `modred`. The first, and most important, is an object-oriented, high-level, easy-to-use, and flexible interface. This interface ensures that the user never needs to modify or be aware of anything deeper than the carefully-chosen inputs and outputs of each function, as shown in the code samples in previous sections. Users can focus on the science of their particular application rather than implementation details. In addition to saving users' time, the ease of use opens up model reduction to a larger audience who can use model reduction as a tool without understanding exactly how the algorithms work, just their inputs and outputs.

The library is carefully parallelized for distributed memory machines, not only for performance (as described in section 3.5), but also for modularity and usability. After a user writes a script that uses `modred` in serial, in almost all cases the same script can be used in parallel with *no changes*. The user does not need to know how to write parallel code to

run in parallel, making `modred` accessible to a wide audience. Parallel execution requires the freely available MPI Python module `mpi4py`, but the library works seamlessly in serial without it. Secondly, all of `modred`'s non-trivial parallelized functions (inner products and linear combinations) are isolated to a lower-level class used by all of the higher-level classes. This modularity greatly simplifies implementation and future extensions of `modred`.

The library has a modular organization of classes and functions. Every function's inputs and outputs are carefully chosen to be intuitive and reflect the mathematical structure. The details of the implementation are irrelevant to other classes and functions, and therefore improvements that change the implementation, for example to increase efficiency, have little, if any, effect on other code that uses `modred`, including users' scripts. This organization also enables `modred` to have comprehensive unit tests for each class method and function. This is a major advantage over "home grown" numerical packages, which typically have, at best, regression checks of a few known results. Instead, by testing all functions and steps of algorithms independently for a variety of inputs and expected outputs, the results are much more likely to be correct. This type of testing is widely employed for large projects, and it greatly improves reliability and eases debugging by isolating potential problems. In addition, we unit test all parallelized functions in serial and in parallel.

The computational overhead of `modred` is quite low since its primary purpose is to call `numpy` in the matrix multiplication case and orchestrate calls of the user-supplied functions in the vector space case. Simply put, if `numpy` and the user-supplied functions are fast, then so is `modred`. A common concern is that user-supplied functions, written in Python, are not executed fast in comparison to compiled languages such as C/C++ and Fortran. With appropriate usage of Python and `numpy` (e.g., avoiding loops), the speed limitations of Python are often acceptable. For cases where the speed limitations are severe, there are several packages which make it easy to mix compiled languages with Python, including `Cython` and `SWIG` for C/C++ and `f2py` for Fortran. Using these, one can write the inner product, loading, and saving functions in a compiled language, wrap them into a Python module, and use them with `modred`. In cases where these functions already exist, for example as part of simulation software, users can wrap the existing functions into a Python module and save themselves from tedious and mistake-prone duplication and/or translation.

3.5 Parallelization

In this section we describe how we parallelize the vector operations used in the vector space approach and corresponding implementations of POD, BPOD, DMD, and Petrov-Galerkin projections for a distributed memory architecture via MPI. There are two main steps in each decomposition: computing inner products and taking linear combinations of the input vectors to form the modes. The matrix multiplication and array-based implementations are not parallelized and operate best when `numpy` is installed with a shared memory backend such as Intel's MKL. Similarly, the system identification methods OKID and ERA are generally not computationally intensive and thus are not parallelized.

3.5.1 Inner product matrices

The computation of inner products of vectors is often the most expensive step of modal decompositions. However, it is easily parallelized since each element of $\mathbf{H}$ depends only on the inner product of two vectors, $[\mathbf{H}]_{i,j} = \langle \mathbf{z}_i, \mathbf{x}_j \rangle$. Throughout this section, the rows of $\mathbf{H}$ are indexed by i and there are n_r of them. Similarly, the columns are indexed by j and there are n_c of them.

Now we outline the parallelization of computing the inner product matrix. First, the rows and columns of $\mathbf{H}$ (and corresponding vectors $\mathbf{z}_i$ and $\mathbf{x}_j$) are assigned to the n_p processors. Thus, the first n_r/n_p rows and n_c/n_p columns are assigned to the first processor, the second n_r/n_p rows and n_c/n_p columns are assigned to the second processor, and so on.

Often there are too many vectors to fit in memory simultaneously, so we load only subsets at one time. We define the maximum number of vectors which can be in memory as n_v. Thus each subset consists of $n_v - 2$ vectors, where the 2 comes from also loading a vector corresponding to a column and to an extra spot as a buffer for MPI send and receives (upcoming). In the first, outermost, loop, each processor iterates over *subsets* of modes, of which there are $n_r/(n_p(n_v - 2))$. The processors load the vectors that correspond to their current subset of rows, $\mathbf{z}_i$.

In the first nested loop, each processor loads a single vector corresponding to a column, $\mathbf{x}_j$. At this point, $n_v - 1$ vectors are in each processor's local memory. In the second nested loop, the inner products are computed. On the first iteration, the inner products are computed between the vectors corresponding to the loaded rows, $\mathbf{z}_i$, and the newly loaded vector $\mathbf{x}_j$. On subsequent iterations, all of the processors exchange $\mathbf{x}_j$ via MPI sends and receives in a circular pattern. During the MPI send and receives, there are at most n_v vectors in each processor's local memory. Then processors compute the inner products of the loaded vectors $\mathbf{z}_i$ (unchanged) and the newly received vector $\mathbf{x}_j$, filling in new parts of a column in matrix $\mathbf{H}$. This process is repeated until the circular pattern of MPI sends and receives is completed.

After the completion of these three loops, each processor has a portion of the $\mathbf{H}$ matrix filled in. Finally, an MPI all reduce sums each processor's $\mathbf{H}$ with all others', resulting in the completed $\mathbf{H}$ matrix in each processor's local memory.

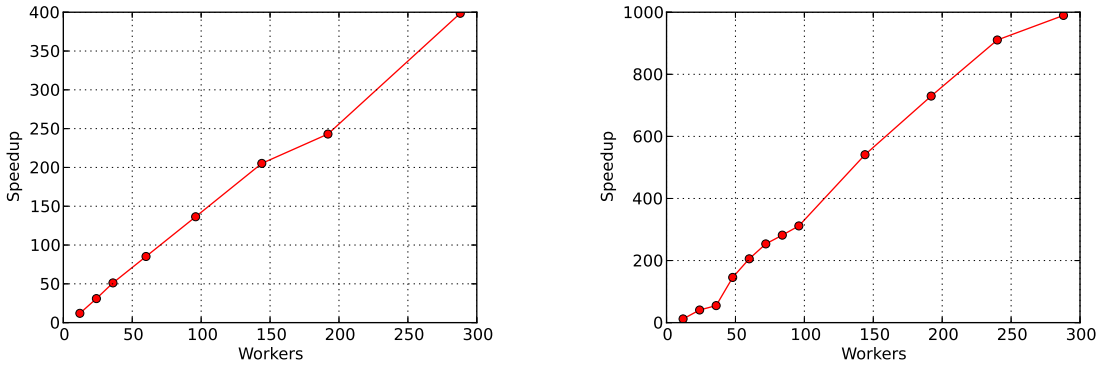
This procedure's scaling is shown in Table 3.1. Overall, it scales linearly with n_p. The number of loads has a term which scales quadratically with n_p and corresponds to n_c. This quadratic term exists because increasing n_p decreases the number of total loads. To exploit the quadratic scaling, we always use $n_c > n_r$ by switching the rows and columns when necessary. We also note that when there is a fixed amount of memory (i.e., $n_v n_p$ is constant), as is the case for multiple processors on a node, it is typically advantageous to use all available processors (also see the section 3.5.3 on hybrid parallelization).

Benchmark results are summarized in Figure 3.1a. The vectors are composed of $9,000$ random double-precision floating-point numbers, and are saved to file in Python's binary "pickle" format. The number of rows and columns is $n_r = n_c = 8000$, and the maximum number of vectors in each processor's local memory is $n_v = 5$. Typically n_v would be larger for this size vector, but we use this lower value just for demonstration. The figure shows that increasing the number of processors results in a roughly linear speedup. The slope of this speedup is greater than one, meaning increasing the number of processors is highly

Table 3.1: Scaling of inner products.

Operation	**No. operations per processor**	**Wall time scaling**
Loads	$\left(\frac{n_r}{(n_v-2)n_p}\right)\left(\frac{n_c}{n_p}\right)+\frac{n_r}{n_p}$	$O\left(\frac{n_c n_r}{n_v n_p^2}\right)+O\left(\frac{n_r}{n_p}\right)$
Send-receives	$(n_p-1)\left(\frac{n_c}{n_p}\right)\left(\frac{n_r}{(n_v-2)n_p}\right)$	$O\left(\frac{n_r n_c}{n_v n_p}\right)$
Inner products	$\frac{n_r n_c}{n_p}$	$O\left(\frac{n_r n_c}{n_p}\right)$

advantageous. This is due to the quadratic scaling in the number of loads.



(a) Inner products.

(b) Linear combinations.

Figure 3.1: Measured scaling of parallelized vector operations. "Workers" is equivalent to processors.

When $\mathbf{H}$ is symmetric, as in POD and DMD, the number of inner products to compute is nearly halved. We use a more complicated parallelization to take advantage of this, but the scaling is the same and so we omit the details.

3.5.2 Linear combinations

The second parallelized portion of the code is the computation of the modes from linear combinations of the snapshots, following (3.1). In this section we let n_s be the number of snapshots and n_m be the number of modes.

First, the snapshots and modes (not yet computed) are assigned to the n_p processors. Thus, the first n_s/n_p snapshots and n_m/n_p modes are assigned to the first processor, the second n_s/n_p bases and n_m/n_p modes are assigned to the second processor, and so on. In the outermost, first, loop, each processor computes n_v-2 modes. In the first nested loop, each processor loads one snapshot x_j from the set it is responsible for (so n_s/n_p iterations). In the second nested loop, the snapshots are summed to form the modes $\boldsymbol{\varphi}_j$. On the first iteration, the newly loaded snapshot's contribution ($\mathbf{x}_i[\mathbf{T}]_{i,j}$) is summed to the subset of

modes currently in memory. It can help to think of these contributions to the modes as "layers", and the sum of all layers is the mode. On subsequent iterations, all of the processors exchange their snapshots via MPI sends and receives in a circular pattern. During the send and receives, there are at most n_v vectors in each processor's local memory. After exchanging snapshots, each processor computes a new subset of mode layers that correspond to the newly received snapshot. This process is repeated until the circular pattern of MPI sends and receives is completed.

At the completion of first nested loop, each processor has a completed subset of modes and saves them to disk. Then a new subset of modes is begun (another iteration of the outermost loop), until all modes are computed and saved.

Table 3.2: Scaling of operations for computing linear combinations.

Operation	**No. operations per processor**	**Wall time scaling**
Loads	$\left(\frac{n_s}{n_p}\right)\left(\frac{n_m}{(n_v-2)n_p}\right)$	$O\left(\frac{n_s n_m}{n_v n_p^2}\right)$
Send-receives	$(n_p-1)\left(\frac{n_s}{n_p}\right)\left(\frac{n_m}{(n_v-2)n_p}\right)$	$O\left(\frac{n_s n_m}{n_v n_p}\right)$
Scalar multiplications	$\frac{n_s n_m}{n_p}$	$O\left(\frac{n_s n_m}{n_p}\right)$

This procedure scaling is shown in Table 3.2. The overall scaling is very similar to the scaling of the inner products. First, the scaling is linear with n_p. Again, loading (usually the most time-consuming operation) scales quadratically because additional processors reduce the total number of snapshot loads. As is also the case for inner products, when there is a fixed amount of memory, i.e., $n_v n_p$ is constant, as is the case for multiple processors on a node, it is typically advantageous to use all available processors (also see the section 3.5.3 on hybrid parallelization).

The scaling results for a dataset similar to the one used for the inner product computations are shown in Figure 3.1b. The figure shows that increasing the number of processors results in a roughly linear speedup. The slope of this speedup is greater than one, and as for the inner product case, this is due to the quadratic scaling of the loading.

3.5.3 Hybrid parallelization

The `modred` library itself is only parallelized for distributed memory (via MPI). However, it is possible to achieve speedups by using shared memory within a node. Each "processor" can be an entire node. Then, the user-supplied functions for inner products, loading, and saving can be written for a shared memory architecture (e.g., via OpenMP) and use all of the processors on each node. This approach could improve efficiency, but would depend on the type of data, the operations involved, and the particular computer hardware being used.

3.5.4 Very large data parallelization

Currently `modred` cannot operate on vectors so large that three cannot fit in a node's memory simultaneously. A work-around is to use so-called "fat" nodes – nodes with large amounts of memory, thereby increasing n_v to at least three. The algorithms' speeds are limited more by memory than processing speed, so fat nodes are generally a good choice. In the future, `modred` could be extended to operate on this data, for example by generalizing processors to be MPI communicators and allowing the user-provided vector class to itself be distributed across multiple nodes. The modular design makes this extension simpler.

3.6 Example results

In this section we show two example usages of `modred` on real problems of interest and all the intricacies associated with such real problems. The `modred` library performs well, reproducing existing model-reduction results that accurately describe the governing systems.

3.6.1 Smaller data: complex Ginzburg-Landau equation

In this case the system is the linearized complex Ginzburg-Landau (CGL) equation, a partial differential equation in time, t, and one spatial dimension, x

$$\frac{\partial q}{\partial t} = -\nu \frac{\partial q}{\partial x} + \gamma \frac{\partial^2 q}{\partial x^2} + \mu q + \mathbf{B}u \tag{3.30}$$

where q is the state and is a function of x and t, u is the input and is a function of t. Parameters ν and γ are complex constants, and parameter μ is a function of space

$$\mu = (\mu_0 - c_u^2) + \mu_2 x^2/2. \tag{3.31}$$

The parameters μ_0, μ_2, and c_u are all constants. This system displays many physical phenomena, including convection and diffusion, and thus is frequently used as a model for instabilities in fluid systems [50]. We demonstrate that our library is effective by reproducing the calculation of BPOD modes and reduced-order model of the subcritical case in previous work [52]. We use the same numerical approach as in previous work, based on spectral differentiation and Hermite polynomials, translated from the MATLAB library described in [112] to Python. All of the code is provided in the library's examples for convenience. Figure 3.2 shows the time evolution of an impulse response to the direct and adjoint systems.

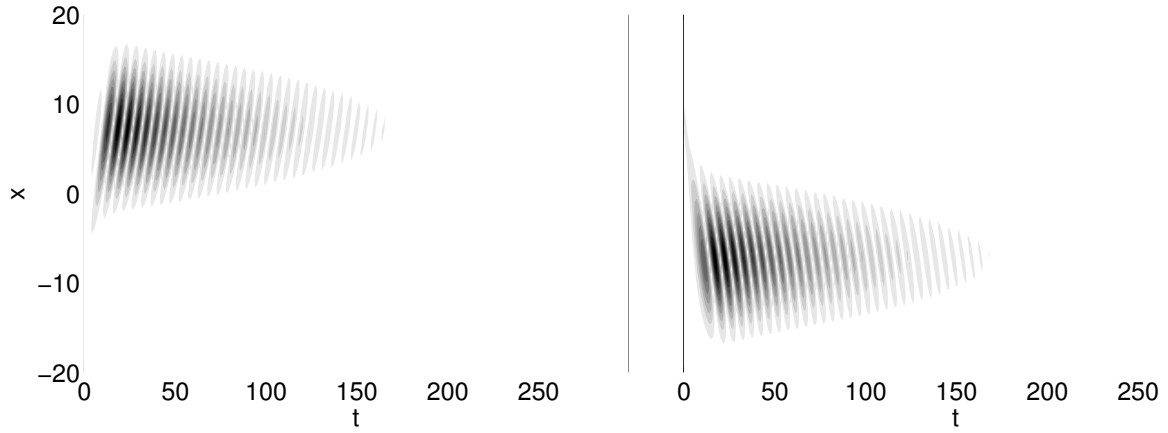


Figure 3.2: Left: Real part of snapshots of impulse response of the direct system to the optimal disturbance input. Right: Real part of snapshots of impulse response of the adjoint system to the optimal disturbance input.

The system is relatively small, with the complex solution discretized over 220 points. Thus the matrix multiplication and array implementations are appropriate. The grid is non-uniform and so there is a weighted inner product. The resulting BPOD modes are shown in Figure 3.3, and the impulse response of the reduced-order model is shown to be indistinguishable from that of the full system in Figure 3.4. The same result is shown in the original work, and so the results obtained with `modred` are consistent with peer-reviewed results.

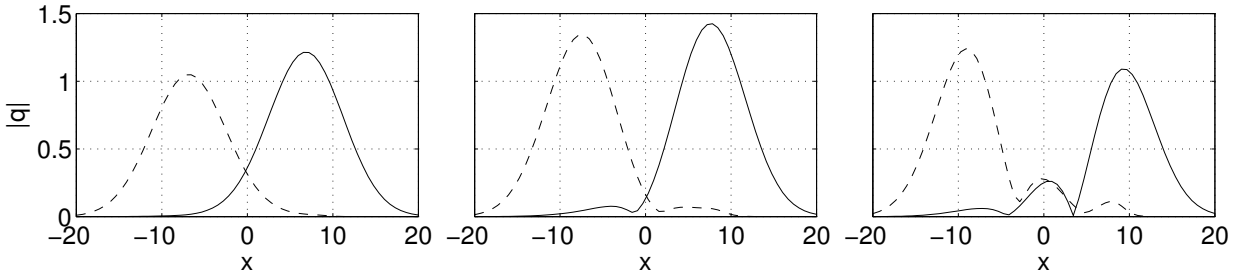


Figure 3.3: Absolute value of the BPOD direct (solid lines) and adjoint (dashed lines) modes with the optimal disturbance and a sensor centered about $x = 8.24$.

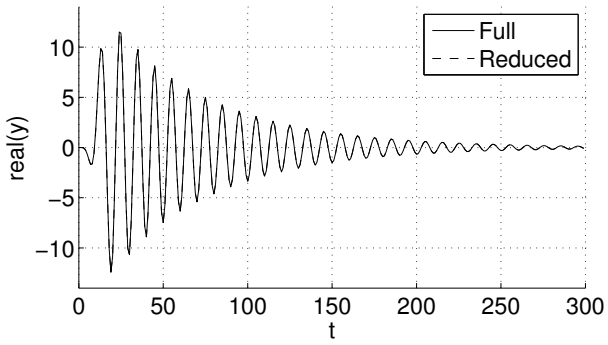


Figure 3.4: The real component of the impulse responses of full and reduced systems. The maximum difference between the two is $2.0 \cdot 10^{-6}$.

3.6.2 Larger data: boundary layer

This example is drawn from the work in Chapter 4, and is briefly summarized here. The physical system is a spatially-evolving boundary layer over a flat plate in two spatial dimensions (x and y) and is governed by the linearized incompressible Navier-Stokes partial differential equations. The application is to use feedback control to dampen the natural growth of disturbances. To do so, the controller is designed to minimize the energy in the flow, as approximated by the projection of the state onto the most energetic (POD) modes [9]. The flow is directly numerically simulated with a pseudo-spectral solver [19]. Figure 3.5 shows the snapshots from an impulse response in the inputs (located along the horizontal axis at 35 and 400). Approximately 4000 snapshots are used, each of which contains the two components of velocity at 768×101 grid points and occupies roughly two megabytes of memory.

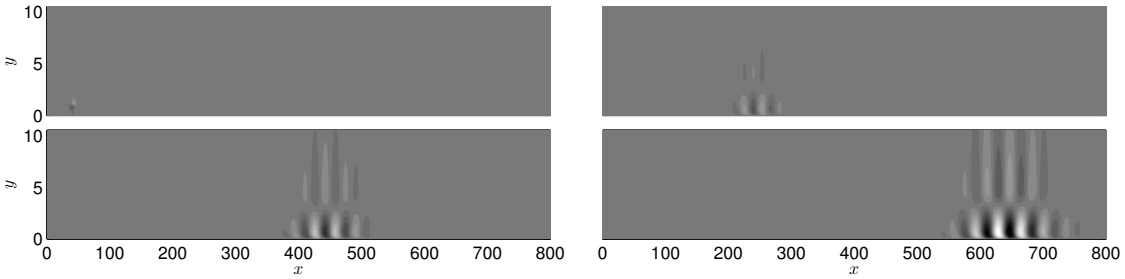


Figure 3.5: Snapshots of the response to an impulse in the disturbance input sampled at intervals of 500 convective time units.

In this case, the size of the data and computer's memory make it impossible to fit all of the vectors in memory simultaneously. Furthermore, the inner product is a complicated function involving discrete Fourier transforms and is more easily represented as a function of two vectors than as a weighted matrix. Thus the POD modes are computed using the vector space approach and the `PODHandles` class and custom vector and vector handle classes (including the inner product). The original simulation software is written in Fortran, so the

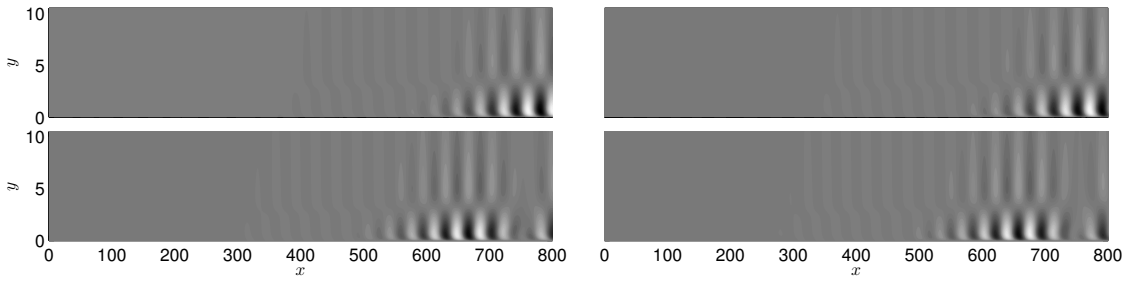


Figure 3.6: Leading four POD modes.

`f2py` package is used to wrap existing subroutines to make them accessible to Python [91]. Figure 3.6 shows the resulting POD modes.

Without `modred`, performing POD on this data would likely be accomplished by flattening and stacking all of the vectors into matrices, and performing a large matrix multiplication. However, this could require writing the inner product weights explicitly rather than as a function, which is laborious. It also would require a computer with more memory—enough to have all of the snapshots in memory simultaneously. Furthermore, the results without `modred` would be prone to many subtle errors that can go undetected even when results look reasonable, but `modred` is tested and so the results are more trustworthy.

3.7 Summary

As datasets have grown larger, so has the need for approximating them with low-dimensional models. We present a new Python library for model reduction, modal decompositions, and system identification. A few modal decomposition methods, along with Petrov-Galerkin projection of linear systems onto modes, are implemented. This library fills a need of a publicly available implementation of these algorithms.

Two implementations of the modal decomposition algorithms are provided for different categories of data. One implementation follows the common matrix multiplication approach and is suited for smaller and simpler data. The vectors are flattened and stacked into matrices to make use of highly optimized matrix multiplication algorithms. A second implementation follows a vector space approach and is well-suited for larger and more complicated data. This approach can handle any data format since the user provides a vector handle for loading and saving vector objects. Since Python is dynamically typed, the only requirements of the vector objects are to satisfy the properties of a vector, namely addition, scalar multiplication, and inner products. These vectors are loaded and saved only as needed within `modred` by user-supplied vector handles, which themselves use very little memory. As a result, `modred` can handle large datasets and is parallelized for a distributed memory architecture with favorable scaling.

The library itself is lightweight, and the computational speed is limited only by the speed of the SVDs, eigenvalue decompositions, and user-supplied functions. Additionally, it is easy for users to wrap compiled code (e.g., C/C++ and Fortran) into Python modules for increased speed. Thus the speed limitations of Python can be bypassed as necessary.

The system identification methods ERA and OKID are also implemented. These two

methods are not computationally intensive and so are not parallelized. To our knowledge, this is the first widely available implementation of these commonly used methods.

The library is written in a modular way to make it easier to use, test, and extend in the future. All of the classes and member functions are carefully designed to mirror the mathematical algorithms, promoting clarity and understanding. Each function is tested independently for a range of cases, checking sources of programming errors. Thus, when the tests pass, one can have confidence in the results. This is in contrast to most home-grown codes that are often not thoroughly tested on the level of individual functions. Instead, they only test that a few previously computed cases match (regression tests), and thus there is a chance of misunderstanding and subtle mistakes and inefficiencies.

The modular, object-oriented, design makes it easy for a user to use any or all of the aspects of `modred`. A user does not need to know all of the details of the underlying algorithm and implementation, only a few function calls, and we hope this will make model reduction a tool accessible to a broader audience.

In the future, we anticipate others will contribute additional useful model reduction techniques and collaborate with the development of the library. It is available under the "Free BSD" license at `http://pypi.python.org/pypi/modred`.

The ERA and POD algorithms implemented in the library are used in the upcoming chapter (Chapter 4) on modeling and control of the 2D boundary layer. The POD modes are used for output projection, i.e., as an approximation of the total energy in the flow. ERA is used to find a reduced-order linear model from the actuator to the outputs, which include a sensor and the projection of the velocity onto the POD modes.

Chapter 4

Selection and placement of sensors and actuators

The previous chapter describes a model reduction library. In this chapter, we use that library's implementations of the Proper Orthogonal Decomposition (POD) and Eigensystem Realization Algorithm (ERA) to help address a challenging topic in fluid dynamics and control—the effect of different actuators and sensors on a controller's performance and robustness. In particular, we consider the case of delaying transition in the 2D Blasius boundary layer.

This chapter begins, in Section 4.1, reviewing previous control studies of the boundary layer. Section 4.2 defines the physical fluids problem and the input forcing and output measurements. In Section 4.3, we recast the fluids equations as a linear time-invariant system for control, and show the equivalent block diagram. We explain the control design techniques, and how we achieve an approximation of these controllers using reduced-order models. In Section 4.4, we form the reduced-order model and observe that it accurately approximates the input-output dynamics of the original high-order system. In Section 4.5, we analyze the performance of the controllers with different actuator-sensor configurations, and explain why we need to use different actuators and sensors for an feedback controller. Lastly, we demonstrate that feedback control outperforms feedforward in the presence of unmodeled disturbances. The work in this chapter is published in *Physics of Fluids* [9].

4.1 Introduction

As discussed in Chapter 2, the drag on a streamlined body increases as the surrounding flow transitions from laminar to turbulent, and delaying this transition can increase performance in many applications, such as airplanes and turbines. Many passive control techniques have been developed to delay transition, but active control, in which actuation is based on sensor measurements, has the potential to delay transition in the presence of unknown disturbances and noisy measurements while also using less actuation energy [62]. In the present study, we focus our attention on boundary layer flows developing over a flat plate, emphasizing the influence of actuator and sensor placement on the performance and robustness of the device.

Control design for the boundary layer requires special care because the governing equa-

tions are high-dimensional and nonlinear. We begin by linearizing about the laminar convectively unstable equilibrium, since our intention is to drive the flow towards this state. Many control design techniques exist for linear systems, but they are computationally expensive to apply directly. Ideally, one would like to design a controller for the high-dimensional system, and then form a reduced-order approximation of this controller for actual implementation [4]; however, for fluids problems, computing the controller from the high-dimensional model is usually not computationally tractable. A common solution, and the one we adopt, is to first find a reduced-order model that approximates the original high-dimensional system, and then apply standard control design techniques on the model.

A variety of methods exist for finding reduced-order models for the purpose of control design. Perhaps the earliest such method is that of Aubry [5], which used Proper Orthogonal Decomposition (POD) to obtain a very low-dimensional nonlinear model capturing many relevant features of the boundary layer. These POD-based approaches have been effective for some flow control problems such as wake flows [34], but for shear flows with large non-normality, they often require physical insight or tuning to make the models accurate and amenable to control [53]. Much of the recent work on controlling boundary layers has focused on linear models, including models based on global eigenmodes [3, 69]. From a control perspective, it is most appropriate to project the governing equations onto a set of modes such that the input-output dynamics are well captured; these modes are then the most controllable and observable. As discussed in Chapter 3, a method known as the Eigensystem Realization Algorithm (ERA) [58, 78, 82] does approximately this, and yields a new set of states which are ranked by controllability and observability. Less controllable and observable states are easily truncated as desired, and there are *a priori* error bounds on the amount the truncated model differs from the original system.

Previous studies [7, 69, 101] have demonstrated effective control of Tollmien-Schlichting (TS) waves in the boundary layer using active control. In [27], adaptive filters and active wave cancellation were used for feedback control of a compressible boundary layer. Other studies [7, 101] used a modeling technique that is equivalent to ERA (balanced POD) [78]. Our intention is to build on these previous results by intentionally using a similar geometry in order to remove unnecessary differences and ease comparison. In previous works, the sensors were placed upstream of the actuators without rigorously exploring other choices. However, the positions of the actuator and sensor can have a significant effect on the performance and robustness of the active control. For example, for the complex Ginzburg-Landau equation (a system similar to that of bluff body wakes), it was found that the optimal actuator and sensor configuration significantly increased performance in damping the targeted structures [17].

The main contribution of this work is to analyze the effect of different actuators, sensors, and their positions on the active control of the instabilities that lead to transition in the 2D spatially evolving Blasius boundary layer. This parametric analysis can be crucial when considering implementing control in experiments and applications; indeed, numerical simulations allow us to investigate the influence of many actuator and sensor positions that would be difficult and costly in physical experiments and applications, if possible at all.

We demonstrate how the relative position of the sensor and the actuator determines the controller's properties. In particular, when the sensor is located upstream of the actuator, the arrangement results in a disturbance feedforward controller, in which the effect of the

downstream actuator is not detected by the upstream sensor due the convective nature of the boundary layer. This is the same setup used in previous works [7, 101] and we show that the best rejection of known disturbances is achieved with this configuration. However, the performance of feedforward controllers often degrades in the presence of additional disturbances and unmodeled dynamics, while feedback controllers are usually much less sensitive to these uncertainties. For this reason, a different setup may be desirable, in which the sensor does detect the effects of the actuator: i.e., a feedback configuration with the sensor downstream of the actuator.

We shall see that the original choices of actuators and sensors are ineffective in feedback configurations. By using different actuators and sensors, however, a simple proportional-integral feedback controller performs well at rejecting disturbances and is robust to unmodeled disturbances as well. We show that the feedforward controller's performance is degraded by an additional disturbance, while the feedback controller's performance is essentially unaffected.

4.2 Physical problem

The objective of active control is to delay the transition to turbulence in the two-dimensional Blasius boundary layer by suppressing the (convectively) unstable growth of propagating disturbances. The governing equations are the incompressible Navier-Stokes equations, linearized about the zero-pressure-gradient laminar base flow, $\mathbf{V}$:

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} &= -(\mathbf{V}\cdot\nabla)\mathbf{v} - (\mathbf{v}\cdot\nabla)\mathbf{V} - \nabla p + \frac{1}{Re}\nabla^2\mathbf{v} + \mathbf{f} \\ \nabla\cdot\mathbf{v} &= 0. \end{aligned} \tag{4.1}$$

Here, $\mathbf{v}$ is the deviation from the laminar base flow, and p is the pressure. The Reynolds number is defined as $Re = U\delta_0^*/\nu$ where U is the free-stream velocity, ν is the kinematic viscosity, and δ_0^* is the displacement thickness at the inlet of the computational domain. We use $Re = 1000$ in all cases and non-dimensionalize all lengths by δ_0^*. The incompressible linearized Navier-Stokes equations are convectively unstable at this Re, and the physical form of the instability is exponentially growing Tollmien-Schlichting (TS) waves. We consider the linearized equations because the controllers force the flow towards the laminar base flow, and so we expect the truncated nonlinear terms to be small. This approximation also allows us to use existing control, modeling, and analysis techniques available for linear systems.

A representative schematic of the problem is shown in Figure 4.1. Note that this arrangement is the same as in [7] for ease of comparison. The momentum equation (4.1) is forced with

$$\mathbf{f} = \mathbf{B}_{\mathsf{w}}(x,y)\mathsf{w}(t) + \mathbf{B}_{\mathsf{u}}(x,y)\mathsf{u}(t), \tag{4.2}$$

where $\mathsf{w}(t)$ is a random disturbance, sampled from a normal distribution with zero mean and unit variance, and $\mathsf{u}(t)$ is the actuator signal, provided by the controller. The terms $\mathbf{B}_{\mathsf{w}}$ and $\mathbf{B}_{\mathsf{u}}$ are the spatial distributions of the disturbance and actuator. In the case that the force is not divergence-free, then, as shown in [14], only the component of $\mathbf{f}$ that is divergence-free directly affects the velocity. The other component affects the pressure, which is not of

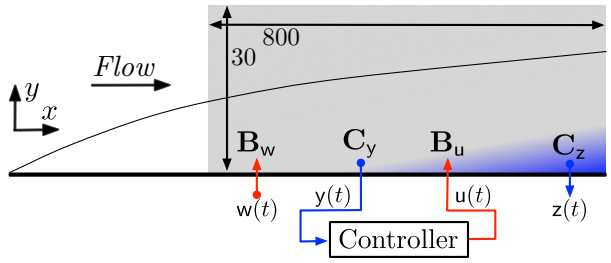


Figure 4.1: Overview of boundary layer, inputs and outputs. The grey box denotes the computational domain. The upstream disturbance is $\mathsf{w}(t)$, applied at $x_\mathsf{w} = 35$, and the control input is $\mathsf{u}(t)$, applied at $x_\mathsf{u} = 400$. The output $\mathsf{z}(t)$ is a low-order approximation of the velocity, $\mathbf{v}(t)$. Output signal $\mathsf{y}(t)$ is from a localized velocity sensor, includes noise, and its location, x_y, is varied.

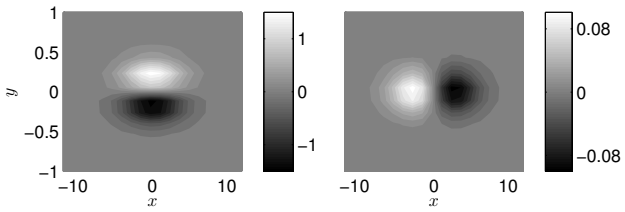


Figure 4.2: Spatial distribution of original actuator and sensor, $\mathbf{S}(0,0)$. Left: stream-wise component. Right: wall-normal component.

interest in this work. The original choice of the forcing spatial distribution is drawn from previous work [7] (called "original" because it is drawn from this previous work) and is

$$\mathbf{B}_\mathsf{w} = \mathbf{S}(35,1), \qquad \mathbf{B}_\mathsf{u} = \mathbf{S}(400,1), \qquad \text{where} \tag{4.3}$$

$$\mathbf{S}(x_0,y_0) = \begin{bmatrix} (y-y_0)\sigma_x/\sigma_y \\ -(x-x_0)\sigma_y/\sigma_x \end{bmatrix} \exp\left(-\left(\frac{x-x_0}{\sigma_x}\right)^2 - \left(\frac{y-y_0}{\sigma_y}\right)^2 \right) \tag{4.4}$$

and $\sigma_x = 4$ and $\sigma_y = 1/4$, shown in Figure 4.2. All of $\mathbf{B}_\mathsf{u}$, $\mathbf{B}_\mathsf{w}$, and $\mathbf{S}(x_0,y_0)$ are functions of x and y. Later (Section 4.5.2), we choose a different form of actuation, $\mathbf{B}_\mathsf{u}$. We fix the location of the actuator at $x_0 = 400$ and vary only the sensor position because we find that, for this flow, the relative positions of the sensor and actuator are far more important than their absolute positions (also see [15]).

The output spatial distributions are $\mathbf{C}_\mathsf{z}$ and $\mathbf{C}_\mathsf{y}$ (functions of x and y), and the corresponding output signals are given by

$$\mathsf{y}(t) = \int_\Omega \mathbf{C}_\mathsf{y} \cdot \mathbf{v}\, d\mathbf{x} + \mathsf{n}(t) \tag{4.5}$$

$$\mathsf{z}(t) = \int_\Omega \mathbf{C}_\mathsf{z} \cdot \mathbf{v}\, d\mathbf{x}, \tag{4.6}$$

where Ω is the computational domain. The original choice of $\mathbf{C}_\mathsf{y}$ uses the same localized spatial distribution as the actuator, $\mathbf{C}_\mathsf{y} = \mathbf{S}(x_\mathsf{u},1)$, where the stream-wise location of the

sensor, x_0, is a key parameter investigated in later sections. Physically, the single sensor measurement is a sum of spatial integrals of both components of velocity. The signal $\mathsf{y}(t)$ also explicitly includes noise, $\mathsf{n}(t)$, with zero mean and a variance of 0.1, a few percent of the magnitude of the noiseless signal. Our focus is not on the effect of sensor noise, hence our choice of a relatively small amount. Higher noise levels have little effect on the results presented later.

The second output signal $\mathsf{z}(t)$ is used to approximate the disturbance energy in the flow, for use in a cost function for optimal control design. The details of this method, called output projection, are explained in Section 4.3.2. The spatial support of $\mathbf{C}_\mathsf{z}$ is global but primarily downstream because the disturbance energy grows as it convects. While it is not realistic to construct a physical sensor to measure $\mathbf{C}_\mathsf{z}$, this does not preclude the resulting controllers from use in experiments because $\mathsf{z}(t)$ is not supplied to the controller; it is used only in the design of the controller.

4.3 Methods

4.3.1 Numerical flow solver

Simulations are conducted using a pseudo-spectral solver code for incompressible boundary layer flows, as previously described in Chapter 2. The domain shown in the grey box in Figure 4.1 excludes the non-physical fringe region, which extends another 200 δ_0^* to the right. The grid size is 784×101, and is chosen based on a resolution study for these boundary conditions and this $Re = 1000$.

4.3.2 Modeling and control

In this section we cast the flow control problem in terms of linear control theory. The discretized linearized Navier-Stokes equations (4.1) are expressed as a linear time-invariant system

$$\begin{aligned} \mathbf{v}_{i+1} &= \mathbf{A}\mathbf{v}_i + \mathbf{B}_\mathsf{w}\mathsf{w}_i + \mathbf{B}_\mathsf{u}\mathsf{u}_i \\ \mathsf{z}_i &= \mathbf{C}_\mathsf{z}\mathbf{v}_i \\ \mathsf{y}_i &= \mathbf{C}_\mathsf{y}\mathbf{v}_i + \mathsf{n}_i, \end{aligned} \tag{4.7}$$

where $\mathbf{A}$, $\mathbf{B}_\mathsf{w}$, $\mathbf{B}_\mathsf{u}$, $\mathbf{C}_\mathsf{z}$, and $\mathbf{C}_\mathsf{y}$ are linear operators, and the subscript denotes the discrete time step. As previously mentioned, only the divergence-free component of the inputs directly affects the velocity, and so the inputs in (4.7) are assumed to be projected onto the divergence-free space. The discrete-time formulation is used for consistency with the time-discretization of flow solvers. Note that we use the same symbols for both discrete and continuous time variables, and rely on context for distinction. We also cast the fluid system in Figure 4.1 as a generic plant and draw an equivalent block diagram in Figure 4.3.

For each position of the sensor, we design controllers to limit the growth of the disturbance energy, $\|\mathbf{v}\|^2$ with efficient use of input energy u^2. To do this on the full system is computationally expensive. Instead, we find a reduced-order model that approximates the full linearized Navier-Stokes equation and serves as the plant in Figure 4.3. Then we find controllers that are effective for the model, and apply these controllers to the original system.

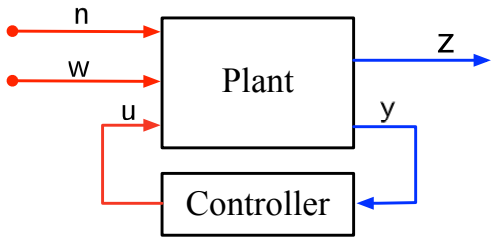


Figure 4.3: Control architecture. In Figure 4.1, the plant is the linearized Navier-Stokes equation (4.7).

For control design, the plant is simply a map from inputs u and w (the noise is relevant to the control, not the model) to outputs y and z as shown in Figure 4.3. Therefore, we seek a reduced-order model that approximates the input-output behavior of the full linearized Navier-Stokes plant; it does not necessarily need to approximate the internal state, the velocity $\mathbf{v}$. In control terminology, such a model is said to retain the most controllable and observable states and neglect the others. A state is considered highly controllable if it is easily excited by an input, and analogously, a state is considered highly observable if, when taken as an initial condition, it excites large future outputs (in the absence of inputs). We use the Eigensystem Realization Algorithm (ERA) and `modred` library, described in Chapter 3, that reduces the order of the system by truncating the (approximately) least controllable/observable states. We use a slightly modified set of steps for computing many models efficiently; see Appendix B for details.

The original state $\mathbf{v}$, and thus the disturbance energy $\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle$, is not accessible from only the reduced-order model state $\mathbf{q}$, but is important for evaluating the effectiveness of a controller. To reproduce the full velocity would require a very large $\mathbf{C}_r$ and would defeat the purpose of the model by dramatically increasing the computational cost of control design. It is appropriate to approximate $\mathbf{v}$ by its projection onto a low-order basis with a method known as output projection [96]. This basis is spanned by the POD modes, such that $\mathbf{C}_\mathsf{z}$ projects the velocity onto the POD modes, i.e. $\mathsf{z} = \mathbf{C}_\mathsf{z}\mathbf{v}$ where z are the POD mode coefficients. The disturbance energy is optimally approximated as $\|\mathsf{z}\|_2^2$. We compute the POD modes via the method of snapshots [103], and again make use of the `modred` library from Chapter 3.

The models are used for two different types of control design: H_2-optimal control and proportional-integral (PI) feedback (see a standard textbook) [104]. Many other types of control design exist, but these are selected because they are common, have clear physical meanings, and are the simplest that demonstrate our results. To facilitate both types of control design, we return to continuous time and write the plant transfer function, $\mathbf{P}(s)$, as

$$\begin{bmatrix} \mathsf{z}' \\ \mathsf{y} \end{bmatrix} = \mathbf{P}(s) \begin{bmatrix} \mathsf{w} \\ \mathsf{n} \\ \mathsf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{\mathsf{w},\mathsf{z}'}(s) & \mathbf{P}_{\mathsf{n},\mathsf{z}'}(s) & \mathbf{P}_{\mathsf{u},\mathsf{z}'}(s) \\ P_{\mathsf{w},\mathsf{y}}(s) & P_{\mathsf{n},\mathsf{y}}(s) & P_{\mathsf{u},\mathsf{y}}(s) \end{bmatrix} \begin{bmatrix} \mathsf{w} \\ \mathsf{n} \\ \mathsf{u} \end{bmatrix} \tag{4.8}$$

where $\mathsf{z}' = \begin{bmatrix} \mathsf{z} & \mathsf{u} \end{bmatrix}^T$ is used as the objective to keep small, and s is the complex frequency (the Laplace transform of time). The noise is only in the sensor, so $\mathbf{P}_{\mathsf{n},\mathsf{z}'}(s) = 0$ and $P_{\mathsf{n},\mathsf{y}}(s) = 1$. We do not place weights on z or u as our focus is primarily on the feasibility of feedback

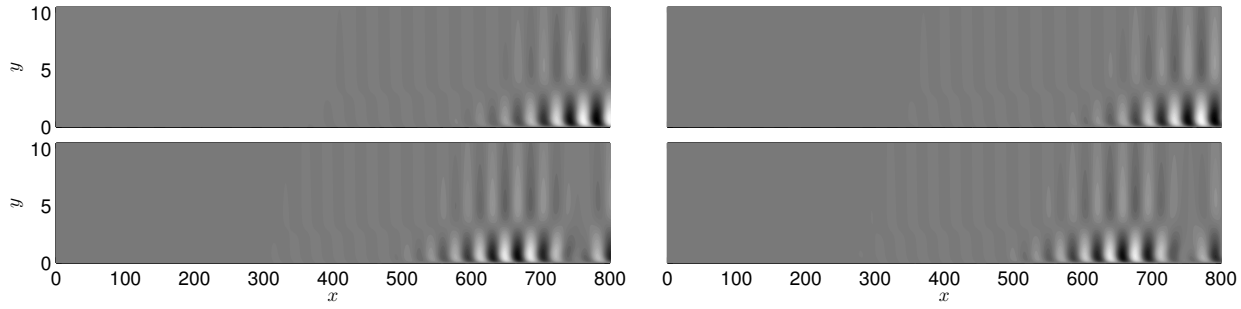


Figure 4.4: Stream-wise velocity of the leading POD modes. Dark corresponds to negative and light to positive, ranging from -0.08 to 0.08.

control, not specific performance goals. The controller transfer function is $K_{\mathsf{y},\mathsf{u}}$, $\mathsf{u} = K_{\mathsf{y},\mathsf{u}}\mathsf{y}$. The closed-loop transfer function from exogenous inputs w and n to objective z' is given by the linear fractional transform,

$$F_l(\mathbf{P}, K_{\mathsf{y},\mathsf{u}}) = \mathbf{P}_{\mathsf{wn},\mathsf{z}'} + \mathbf{P}_{\mathsf{u},\mathsf{z}'} K_{\mathsf{y},\mathsf{u}} (1 - P_{\mathsf{u},\mathsf{y}} K_{\mathsf{y},\mathsf{u}})^{-1} \mathbf{P}_{\mathsf{wn},\mathsf{y}}. \tag{4.9}$$

The first type of control, H_2-optimal, finds the controller, $K_{\mathsf{y},\mathsf{u}}$, that minimizes the cost

$$\text{cost} = \|F_l(\mathbf{P}, K_{\mathsf{y},\mathsf{u}})\|_2^2 = E(\|\mathsf{z}'\|_2^2)/E(\mathsf{w}^2 + \mathsf{n}^2), \tag{4.10}$$

where $E(\cdot)$ denotes the expected value. Physically, this is the optimal tradeoff between minimizing the disturbance energy and actuation energy. For more details, see a standard textbook [104]. Relative weights can be placed on the noise, disturbance, and objective in the cost function, and we explore this as well. The results change slightly, but the overall trends remain the same. Therefore, for clarity, we discuss only the unweighted results.

The second type of control, PI, determines the input signal, u, based on the sum of the sensor measurement and its integral: $\mathsf{u}(t) = k_P \mathsf{y}(t) + k_I \int_0^t \mathsf{y}(\tau) d\tau$, where k_P and k_I are the proportional and integral gains. The proportional term simply forces the flow proportionally to the difference in y from undisturbed laminar flow. The integral term improves the effectiveness by integrating all previous differences in y from undisturbed laminar flow. The two gains are tuned, and will be chosen to effectively reduce $E(\|\mathsf{z}'\|_2^2)$.

4.4 Model Reduction Results

The first ten POD modes capture over 90% of the disturbance energy and define $\mathbf{C}_\mathsf{z}$. Figure 4.4 shows the first four. The spatial support is concentrated downstream, where the energy has been amplified by the flow. The modes appear in pairs because of the traveling structure of the TS waves. Since the modes are real (no imaginary part), a pair is required to span the different phases.

We find that the ERA reduced-order models are accurate with $r = 70$ states. This is supported by the 70^{th} singular value being several orders of magnitude smaller than the first (Figure 4.5), and by the small amount of error between the model and DNS impulse responses shown in Figure 4.6. More states are required than in previous studies [53] due to

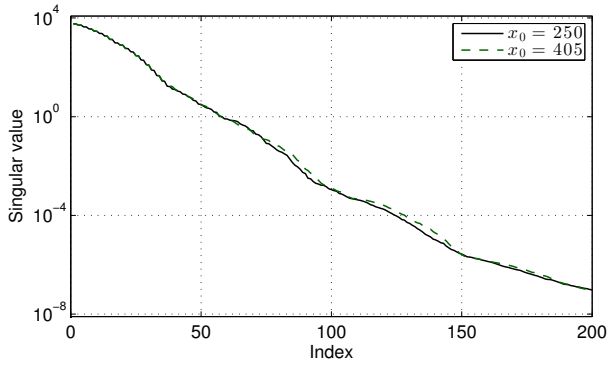


Figure 4.5: Singular values of the Hankel matrices for $\mathbf{C}_\mathsf{y}$ centered at $x_\mathsf{y} = 250$ and $x_\mathsf{y} = 405$.

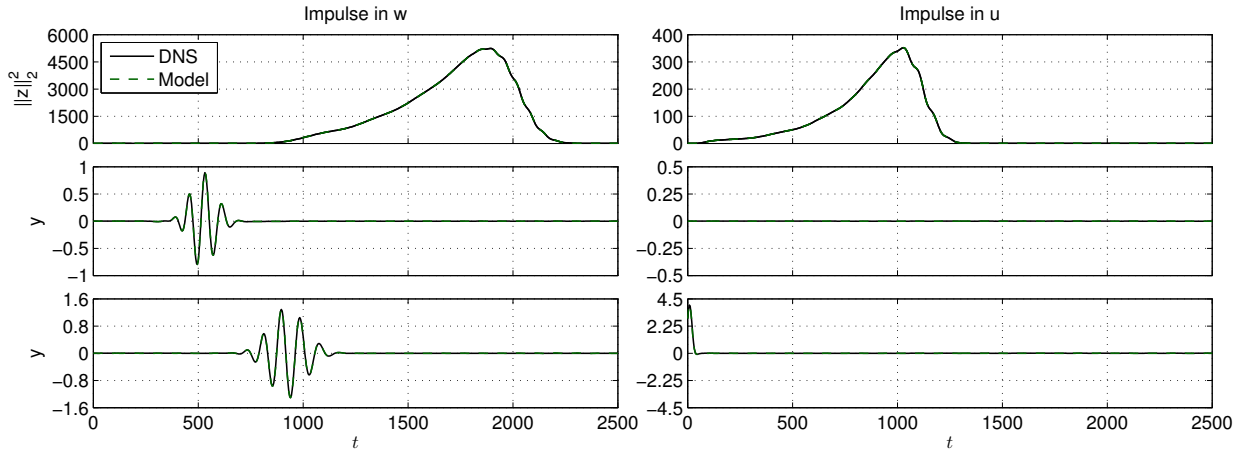
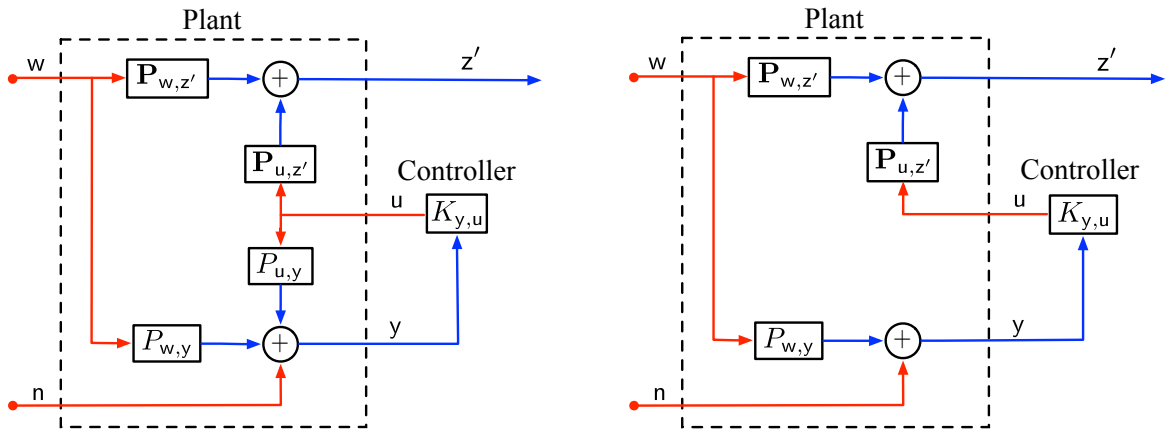


Figure 4.6: Comparison of impulse responses in both inputs to both outputs. Two sensor positions are shown. The second row has $\mathbf{C}_\mathsf{y}$ centered at $x_\mathsf{y} = 250$ and the third row has it centered at $x_\mathsf{y} = 405$. The maximum difference between the full system (DNS) and model amongst all plots is $9.1 \cdot 10^{-3}$.

the large time delays present in this system. The delays exist because the spatially localized inputs in w and u generate perturbations that must convect significantly downstream before the output signals are non-zero. Approximations (such as Padè approximations) of time delays typically need to be high order if the delay is large, as is the case here.

In Figure 4.6, showing input u to output y, the signal is essentially zero because the sensor, $\mathbf{C}_\mathsf{y}$, is upstream of the actuator, $\mathbf{B}_\mathsf{u}$. This highlights an important difference between two classes of actuator-sensor configurations; if the sensor measures the effect of the actuator, then this is a feedback configuration since the controller has information about its effectiveness fed back to it. The flow is highly convective, so this can only occur when the sensor is very near or downstream of the actuator. Conversely, if the sensor does not sense the effect of the actuator, then this is a feedforward control configuration. Even though the flow is incompressible and all effects are technically global, we observe that the effect of the actuator is negligible at points significantly upstream of the actuator. This distinction is ex-



(a) Feedback, $P_{u,y} \neq 0$.

(b) Feedforward, $P_{u,y} = 0$.

Figure 4.7: Comparison of feedforward and feedback actuator-sensor configurations. The dashed box denotes the plant, similarly to Figure 4.3.

plained mathematically in [116], where, within the output feedback control framework, this special case is referred to as disturbance feedforward controller, because only the disturbance is sensed.

The two categories of control types are depicted as block diagrams in Figure 4.7, which are equivalent to the first (Figure 4.3), but broken into four components. The feedback loop in Figure 4.7a between u and y does not exist in the disturbance feedforward case. In previous works [7, 101] the sensor is upstream of the actuator (centered at $x_{\mathsf{y}} = 300$), resulting in a disturbance feedforward control configuration.

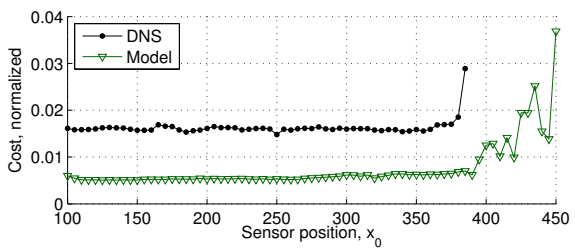
The difference between feedforward and feedback control is more than semantics – the two types of control have fundamentally different properties. For many systems, feedback controllers offer advantages over feedforward controllers, such as increased effectiveness in the presence of plant uncertainties and unknown disturbances. In the next section, we examine the effects of sensor position and feedforward versus feedback control.

4.5 Improved actuators and sensors

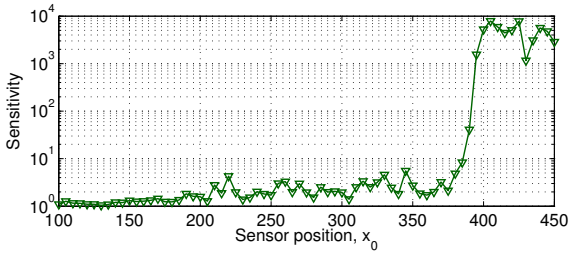
4.5.1 Original actuators and sensors

We begin by varying the position (x_{y}) of the sensor $\mathbf{C}_{\mathsf{y}}$. For each sensor position, the positions of $\mathbf{B}_{\mathsf{w}}$, $\mathbf{B}_{\mathsf{u}}$, and $\mathbf{C}_{\mathsf{z}}$ are unchanged, and we form a new ERA model from existing snapshots (see Appedix B). We compute an H_2-optimal controller for each model and apply it to the full linearized Navier-Stokes system. The resulting performance of the controllers as a function of sensor position is shown in Figure 4.8a.

First, we focus on the sensor positions in feedforward arrangements where $\mathbf{C}_{\mathsf{y}}$ is centered at $x_0 < 390$. Figure 4.8a shows that these sensor locations result in controllers that reduce the cost (Equation (4.10)) to less than 2% of the uncontrolled cost when applied to both the model and the full system (DNS). It is clear that any sensor position upstream of the actuator results in the same good performance. Physically, feedforward is effective because



(a) Cost of H_2-optimal controlled simulations normalized by the uncontrolled cost, $\|F_l(\mathbf{P}, K_{\mathsf{y},\mathsf{u}})\|_2^2 / \|\mathbf{P}_{\mathsf{wn},\mathsf{z}'}\|_2^2$.



(b) Infinity norm of sensitivity function, $\|S\|_\infty$, for H_2-optimal controllers.

Figure 4.8: The actuator is centered at $x_0 = 400$. The controlled full system is unstable for $\mathbf{C}_\mathsf{y}$ centered at $x_0 \geq 390$.

the TS waves retain their structure as they amplify and convect from the sensor to the actuator. This is effectively approximated by the H_2-optimal controller and the TS waves are damped by the controller as they convect by the actuator.

The feedforward-controlled full system is excited by a stochastic disturbance and sensor noise, and the input and output signals are shown in Figure 4.9. This case has the sensor centered at $x_\mathsf{y} = 250$, and is representative of all feedforward configurations. The controller effectively drives the approximate energy towards zero and uses relatively low levels of control, u, so the cost (Equation (4.10)) is low. The time delay of about 2000 time units between when control is turned on and when the disturbance energy begins decreasing is due to the convective nature of the boundary layer. The time delay is 2000 time units because the domain is long ($800\delta_0^*$), and the convection speed in this near-wall region is less than one. The same time delay is observed when the existing disturbed flow downstream of the sensor exits the domain. There is no need to try other actuators, sensors, or control design techniques for feedforward control because this choice is very effective and it has been explored before [7, 69].

Figure 4.8a also shows that the controller is more effective when applied to the model than the full system, and this is because the controller was computed specifically for the model. The slight difference between the input-output behavior of the full system and model results in slightly degraded performance. Since this is feedforward control, the stability is unaffected by this small error in approximating the plant.

However, when the sensor is further downstream and in the feedback regime, i.e. where the effect of $\mathbf{B}_\mathsf{u}$ is sensed by $\mathbf{C}_\mathsf{y}$ for $x_\mathsf{y} \geq 390$, this small difference results in instability for the full controlled system. Thus the full system's controlled cost is infinite and omitted from Figure 4.8a. The instability is due to a lack of robustness, and is an issue only for feedback controllers in which the stability can be changed by the controller. Robustness can be quantified as the infinity norm of the sensitivity transfer function $S(s)$,

$$S(s) = \frac{1}{1 - P_{\mathsf{u},\mathsf{y}}(s) K_{\mathsf{y},\mathsf{u}}(s)} \tag{4.11}$$

where, for good stability margins, one generally seeks $\|S\|_\infty$ to be less than 2.0 [104]. Figure 4.8b shows $\|S\|_\infty$ versus sensor location. For upstream sensors, robustness is a non-issue

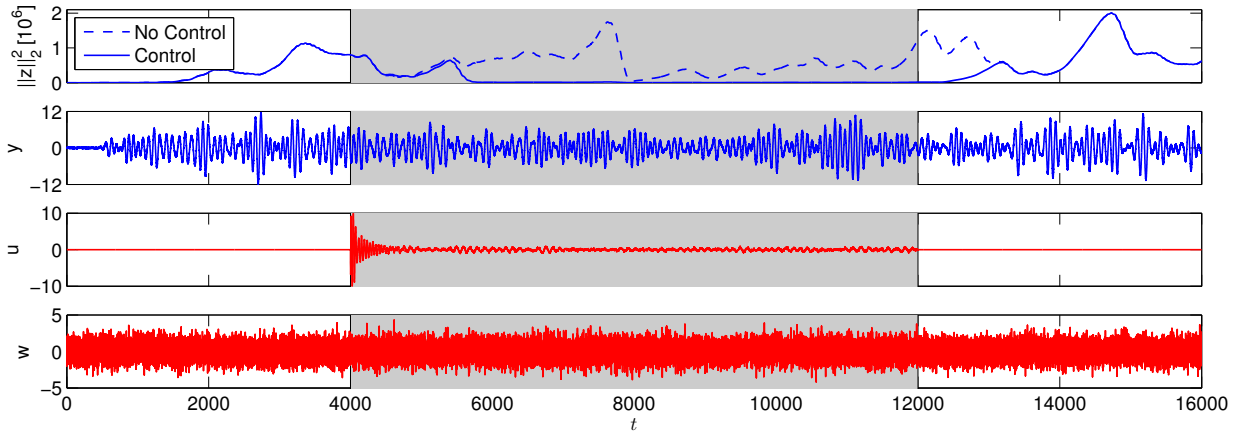


Figure 4.9: Input and output signals of H_2-optimal controlled full system with sensor $\mathbf{C}_\mathsf{y}$ centered at $x_\mathsf{y} = 250$ (feedforward). The control is on from $t = 4000$ to 12000, the grey region.

($P_{\mathsf{u},\mathsf{y}}(s) \approx 0$ and deviations from $S(s) \approx 1$ are negligible). When feedback from u to y exists and $P_{\mathsf{u},\mathsf{y}}$ becomes significantly non-zero, $\|S\|_\infty$ is much greater than 2.0 and even small errors between the model and the full system results in instability.

Physically, the actuation influences the flow field and sensor slightly out of phase with how it is modeled. A measure of robustness related to $\|S\|_\infty$ is the phase margin, the amount of allowable phase error before instability. Typically it is designed to be a minimum of 45° but for these feedback controllers it is very low – less than 0.01°. Therefore, when the effect of the actuation is fed back to the controller, it actuates with a phase that adds to the disturbance, and instability results. Sensor noise tends to expedite this process, but instability is present in the absence of noise. The same low levels of phase error exist in feedforward controllers, but because the error is not fed back, control is not destabilizing.

Generally, poor robustness has a variety of root causes. To identify the root causes here, we divide the feedback configurations into two cases: when the sensor is far downstream of the actuator (roughly $x_\mathsf{y} > 415$) and when the sensor is closer to the actuator (roughly $390 \leq x_\mathsf{y} \leq 415$). In the first case, the sensor measures the effect of the actuator on the flow after a significant time delay due to the highly convective nature of the boundary layer at this Re. This results in right-half plane (RHP) zeros in the transfer function from actuator to sensor, $P_{\mathsf{u},\mathsf{y}}$. RHP zeros are problematic in control design because, following the "weighted sensitivity integral" [104], the frequency at which they exist is the approximate maximum bandwidth, or the maximum frequency that can be controlled with good performance and robustness. Therefore, RHP zeros at low frequencies place severe restrictions on the tradeoff between performance and robustness. Figure 4.10 shows that for the sensor at $x_0 = 450$, the most restrictive (minimum) RHP zero is at $s = 0.03 \pm 0.067i$, or a frequency of $|s| = 0.073$. In [7], it is shown that the disturbance is amplified at frequencies up to approximately $|s| = 0.12$, thus requiring a bandwidth of at least 0.12, significantly higher than 0.073. Thus it is impossible to find controllers with good performance and robustness for this actuator-sensor pair.

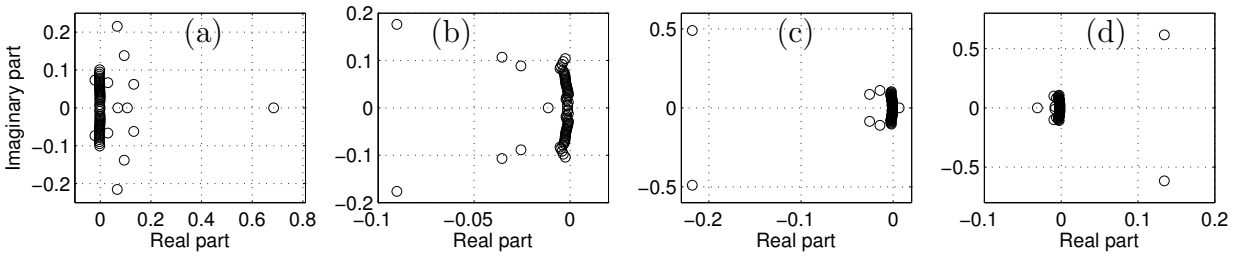


Figure 4.10: Zeros of transfer function $P_{\mathsf{u},\mathsf{y}}$ for four cases. (a): original actuator and sensor at $x_{\mathsf{y}} = 450$ with minimum RHP zeros at $s = 0.03 \pm 0.067i$; (b): original actuator and sensor at $x_{\mathsf{y}} = 405$ with no RHP zeros; (c): original actuator and point sensor at $x_{\mathsf{y}} = 405$ with minimum RHP zero at $s = 0.0065$; (d): new actuator and point sensor at $x_{\mathsf{y}} = 405$ with minimum RHP zeros at $s = 0.13 \pm 0.62i$.

A physical interpretation is the sensor measures flow structures that convected past the actuator at a previous time. The flow structures convecting over the actuator at any time cannot be approximated well by the sensor's outdated information. Thus the control signal cannot be chosen so as to cancel the flow structures convecting over the actuator. This time delay is present for any choice of localized actuator and sensor. Therefore, no controller can perform well and have good robustness, and we restrict our focus to the second case: feedback configurations with the sensor near the actuator.

Specifically, we focus on a feedback configuration with the sensor centered at $x_{\mathsf{y}} = 405$. In this case, as we show over the remainder of this section and the next, the cause of the controller's lack of robustness is not a time delay but a property of this actuator-sensor pair. We begin by noting that H_2-optimal controllers have no guaranteed stability margins [25] and in practice tend to lack robustness. A natural first thought, then, is that while H_2-optimal controllers are not robust, other controllers using this actuator-sensor pair may be robust and perform well. This notion is not supported by the controllers we try. First, methods to recover robustness from an H_2-optimal controller, such as loop-transfer recovery, are not effective here because there are zeros near the imaginary axis (Figure 4.10), which can result in highly oscillatory dynamics and ineffective control [104].

In an effort to achieve better robustness, we consider tuning simple PI controllers. The frequency response of $P_{\mathsf{u},\mathsf{y}}$ is shown in the Bode plot in Figure 4.11. Since TS wave disturbances exist at low frequencies, we want the controller to respond aggressively to low frequency signals. Thus the loop transfer function ($P_{\mathsf{u},\mathsf{y}} K_{\mathsf{y},\mathsf{u}}$) should have a high magnitude (gain) at low frequencies. A simple proportional controller ($k_P = 1$ and $k_I = 0$) achieves this, since $P_{\mathsf{u},\mathsf{y}}$ is already large at low frequencies. With such a controller, the bandwidth (frequency at which the magnitude is 0 dB) is greater than the required 0.12 and the phase margin (phase at the same frequency) is greater than 45°. We find that this controller is robust but does not significantly reduce the disturbance energy as compared to the uncontrolled case, $\|F_l(\mathbf{P}, K_{\mathsf{y},\mathsf{u}})\|_2^2 / \|\mathbf{P}_{\mathsf{wn},\mathsf{z}'}\|_2^2 \approx 0.9$. The controller effectively forces y to be nearly zero, but the disturbance energy is relatively unaffected, shown in Figure 4.12. For other choices of gains k_P and k_I, the results are nearly unchanged; the sensor signal is forced to zero and the energy is relatively unaffected. Forcing the sensor measurement to zero would be desirable if the measurement was correlated with the structures one wants to diminish,

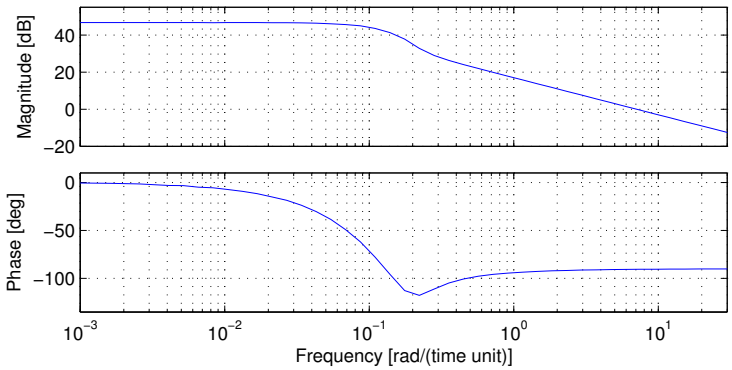


Figure 4.11: Frequency response (Bode plot) of $P_{\mathsf{u},\mathsf{y}}$ with original actuator and sensor centered at $x_0 = 405$.

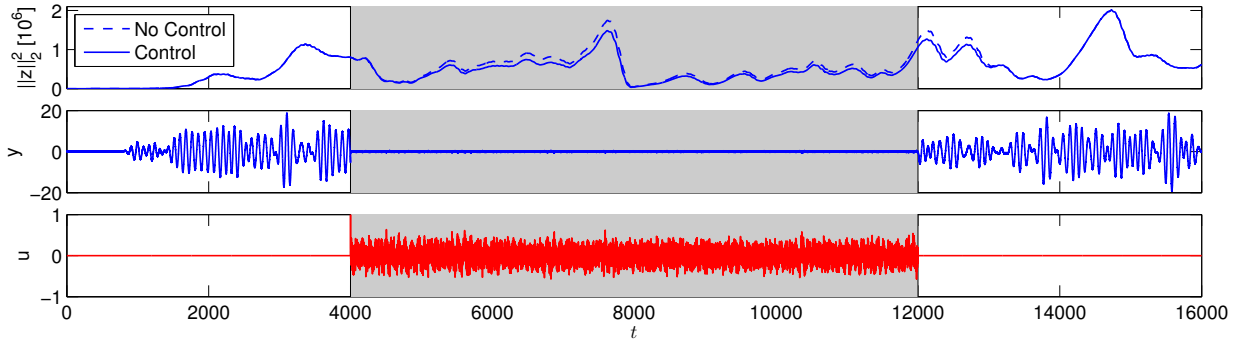


Figure 4.12: Input and output signals of proportional feedback control of the full system with the original actuator and original sensor centered at $x_{\mathsf{y}} = 405$ (feedback). The control is on from $t = 4000$ to 12000 in the grey region. The disturbance signal, w, is the same as in Figure 4.9.

as it often is. However, in this case the actuation only deforms the TS waves to be a slightly different fluid structure – a structure that is poorly observed by these sensors. The controller effectively cancels only the observable component of the deformed TS wave structure, but that component contains little disturbance energy. Put another way, the deformed TS waves are strongly controllable by the disturbance input, but weakly observable by the sensor.

The poor observability of the deformed TS waves (after actuation) is attributed to this sensor, which measures a linear combination of localized stream-wise and wall-normal velocities (Figure 4.2). Figure 4.13 shows the effect of the actuation on the stream-wise velocity. The plot on the left shows the uncontrolled TS wave, and the plot on the right shows the deformed TS wave has two peaks of roughly equal magnitude. These peaks align with those in the stream-wise component of the sensor's spatial distribution (Figure 4.2), resulting in a nearly zero sensor measurement (Equation 4.6). Only this linear combination of velocities is nearly zero, and the deformed TS waves are not significantly damped, continuing to grow downstream.

In the next section, we show that different actuator-sensor pairs have different properties that make feedback control effective. Thus the reason we do not find a feedback controller

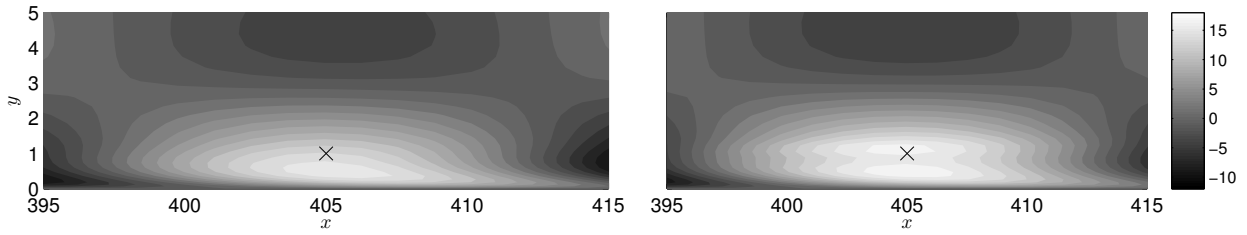


Figure 4.13: Instantaneous stream-wise velocity around sensor location $x_{\mathsf{y}} = 405$, denoted by a cross. Left: Uncontrolled. Right: Proportionally controlled with original actuator-sensor pair, at time $t = 4876$. Even though the sensor measurement is nearly zero (Figure 4.12), the overall velocity field shows little change.

for this sensor-actuator pair with good performance and robustness is attributable to this pair and the poor observability of the deformed TS waves.

4.5.2 New actuators and sensors for feedback

The previous actuator-sensor pair resulted in poor feedback controllers since the TS waves, deformed by actuation, were poorly observable. By choosing new actuators and sensors, we overcome this and design an effective feedback controller.

We begin by simplifying the sensor from the spatial distribution given in (4.3) to a point sensor so that $\mathbf{C}_{\mathsf{y}}$ measures only the stream-wise, dominant, component of velocity at $x_{\mathsf{y}} = 405$ and $y_{\mathsf{y}} = 1$. We choose this sensor because it simplifies the input-output dynamics for easier interpretation and controller design. It measures the TS wave structure more directly than the original sensor does. Further, it is only a slight modification of the original sensor, and it allows us to achieve our primary goal of comparing feedforward and feedback control. Although a point sensor is impossible in experiments, the point sensor is likely to be closer approximation to an experimental sensor than the original sensor because the original sensor averages over a wide stream-wise distance. Further changes to the sensor (or actuator) could potentially give even better performance, robustness, or increased ease of practical implementation, but optimizing the sensor is not our primary goal here.

At first we do not change the actuators from those previously described. The new ERA model is as accurate as those for the original sensors. We find though that the resulting model has a transfer function from input u to output y with a real right-half-plane zero at frequency $s = 0.0065$ (see Figure 4.10), which again is less than the required minimum bandwidth of 0.12. Thus it is impossible to find a feedback controller with good performance and robustness for this actuator-sensor pair.

We chose a different actuator that directly and immediately affects the TS waves, and does not result in a plant with RHP zeros at low frequencies (disturbance input, $\mathbf{B}_{\mathsf{w}}$, is unchanged). We use a Gaussian distribution that forces only in the stream-wise direction, since this is the dominant direction of the flow, and the stream-wise disturbance velocity is significantly larger than the wall-normal component:

$$\mathbf{B}_{\mathsf{u}} = \left(\exp\left(-\left(\frac{x-x_0}{\sigma_x}\right)^2 - \left(\frac{y-y_0}{\sigma_y}\right)^2\right),\ 0\right). \tag{4.12}$$

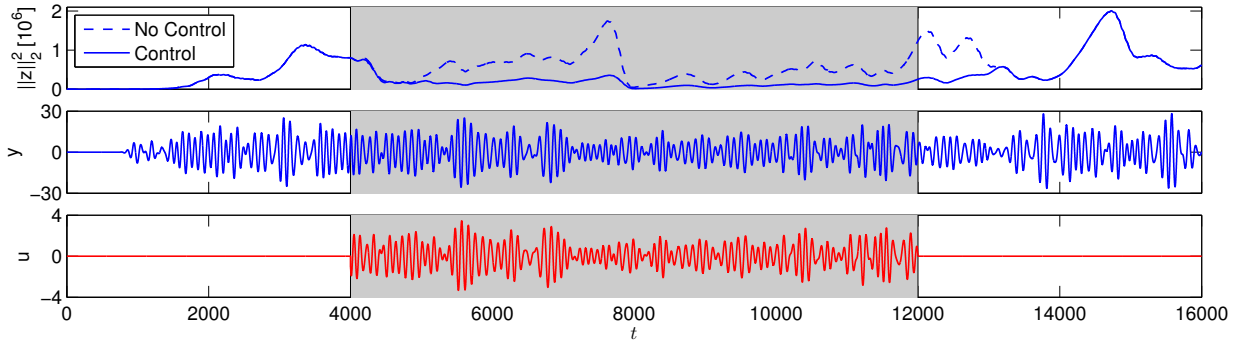


Figure 4.14: Input and output signals of PI feedback controlled full system with the new actuator, Equation (4.12), and a stream-wise velocity point sensor centered at $x_{\mathsf{y}} = 405$ (feedback). The control is on from $t = 4000$ to 12000, the grey region. The disturbance signal, w, is the same as in Figure 4.9.

where, as before, $x_0 = x_{\mathsf{u}} = 400$ and $y_0 = 1$. By directly opposing the growth of the unstable TS waves, we expect this actuator to have the simple and predictable effect of reducing the magnitude of the TS waves. In contrast, more complicated actuators can create more complicated actuator-flow interactions, which can result in undesirable behavior (such as RHP zeros) in the input-output system, as seen with the previous choice of actuator. Figure 4.10 shows that there are RHP zeros at $s = 0.13 \pm 0.62i$, but $|s| = 0.63$ is greater than the required bandwidth, 0.12, and so the RHP zeros are not problematic. ERA models with $r = 70$ states are again accurate. Generally, the choice of actuator and sensor requires physical insight. A rigorous and general study would be valuable, but is not our intention. Instead, we use knowledge about the flow to find a simple choice that is effective for feedback control (as will be shown), and generally advocate this type of approach. Other choices of actuator and sensor can be effective; for example, wall-normal actuation has been used in related studies [13, 74].

The H_2-optimal feedback controllers using this actuator-sensor pair suffers the same drawbacks as the original pair – high performance but with unacceptably poor robustness. Again, lack of robustness is a common drawback of H_2-optimal controllers and methods to recover robustness are not applicable due to many zeros near the imaginary axis and in the RHP [104].

However, PI control has good performance and robustness. We begin by choosing gains k_P and k_I that give an acceptable phase margin and high magnitude loop gain at low frequencies. Tuning the gains results in a robust controller with $\|S\|_\infty = 1.5$, less than the maximum guideline value of 2.0, and a phase margin of 70°, above the minimum guideline value of 45°. Due to improved robustness, the control is not destabilizing when applied to the original full system. The approximate disturbance energy, $\|\mathsf{z}\|_2^2$, versus time is shown for the full system in Figure 4.14, and the overall cost (Equation (4.10)) is less than 25% of the uncontrolled case. While the performance is not as good as feedforward's, which reduces the cost to about 2% of the uncontrolled case, it is an effective controller. More interestingly, it is robust to plant perturbations and unknown disturbances. More advanced robust control design techniques might further improve the performance of feedback control.

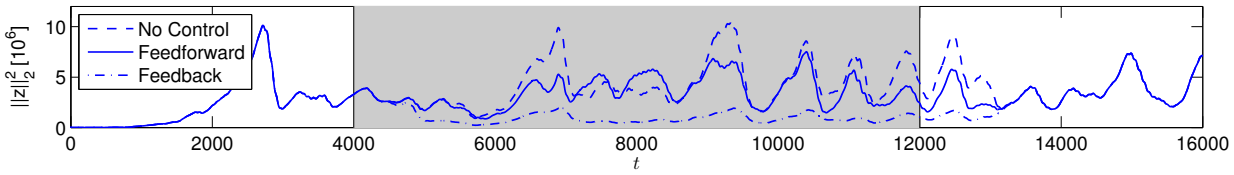


Figure 4.15: Comparison of the effectiveness of feedforward and feedback controllers when a second, unmodeled, disturbance is introduced at $x = 300$. The feedforward controller uses the original actuator and sensor, and the sensor is given by $\mathbf{S}(250, 1)$. The feedback controller uses the new actuator and point sensor at $x_y = 405$.

4.5.3 An unmodeled disturbance's effect

To concretely demonstrate an instance where feedback is preferable to feedforward, we include a second random disturbance that is unaccounted for in the control design. Generally speaking, any disturbance downstream of the actuator is impossible to damp since the flow structures cannot be sensed and then influenced by the actuator because the flow is highly convective. However, it is possible, and desirable, for any disturbance upstream of the actuator to be damped by the controller.

We place an additional, unmodeled, disturbance at $x = 300$ (defined by $\mathbf{S}(300, 1)$) and the corresponding disturbance signal has a variance of 5.0. This is downstream of the feedforward sensor at $x_y = 250$. Thus the new disturbance's effect is not sensed and the feedforward controller is ineffective, only reducing the cost to 82% of the uncontrolled cost. The time signals are shown in Figure 4.15. One could, of course, place the feedforward sensor downstream of the new disturbance, for example at $x = 350$, but this is missing the issue. The location of the unmodeled disturbance is unknown during the control design, and, as mentioned before, we desire a controller that damps any disturbance upstream of the actuator.

A feedback controller, with a sensor downstream of the actuator, can damp any unmodeled disturbance upstream of the actuator. In our example, we find that the feedback controller reduced the cost to 20% of the uncontrolled case. This is much better than the 82% achieved by the feedforward controller, and also similar to the case with only the modeled disturbance (25%). Overall we find that larger unmodeled disturbances worsen the feedforward controller's performance, but have almost no effect on the feedback controller's performance.

4.6 Summary

We use the Eigensystem Realization Algorithm (as described in Chapter 3) to model the 2D linearized Blasius boundary layer flow, then design controllers for the reduced-order models and apply them to the original high-order system. We investigate the role of placement and choice of the actuator and sensor on performance, stability and robustness of the closed-loop. Due to the highly convective nature of the system, an upstream sensor cannot significantly sense the effect of the downstream actuator, even though this is incompressible flow. Thus the relative positions of the sensor and actuator dictate whether control is feedforward or

feedback. We confirm the physical intuition that a feedforward configuration performs best, and that the particular upstream location is relatively unimportant.

However, feedforward controllers have many significant drawbacks. Their performance depends on the accuracy of the model, and so they are ineffective in the presence of unmodeled disturbances and perturbations. Therefore, feedback configurations are attempted. Our first finding is that the original choice of actuator and sensor was poor for feedback, both with H_2-optimal and proportional-integral (PI) controllers. The Tollmien-Schlichting waves are deformed by the actuation in such a way that they were poorly observable due to the choice of sensor, which outputs a localized linear combination of velocity components. The H_2-optimal controllers perform well on the model, but are not robust. The PI controllers are robust and significantly reduce the sensor signal, y, but perform poorly at reducing the disturbance energy. For feedback, this strict tradeoff between performance and robustness renders this actuator-sensor pair ineffective.

We change the sensor to be a simple point sensor of the stream-wise velocity. This actuator-sensor pair results in a transfer function from actuator to sensor, $P_{\mathsf{u},\mathsf{y}}$, which has a right-half-plane (RHP) zero at low frequency. This places severe limitations on performance and robustness, preventing any controller from being developed.

We then change the actuator to be a Gaussian distributed force in the stream-wise direction so $P_{\mathsf{u},\mathsf{y}}$ has no RHP zeros. The H_2-optimal controllers are again not robust. However, the PI controller both performs well and is robust, reducing the objective cost to about 25% of the uncontrolled cost. Thus feedback control is shown to be effective for the boundary layer.

Feedback configurations where the sensor is far downstream of the actuator have large time delays which also cause RHP zeros in the transfer function $P_{\mathsf{u},\mathsf{y}}$, and this severely limits the tradeoff between performance and robustness. Thus, for any localized actuator and sensor, feedback control can be effective only when the sensor is near the actuator.

We demonstrate that feedback control outperforms feedforward control in the presence of unmodeled disturbances. A second disturbance is introduced downstream of the feedforward sensor. The feedforward controller reduces the cost to only 82% of the uncontrolled case, while the feedback controller reduces the cost to 20% of the uncontrolled case. Thus for cases where the unmodeled disturbances are known to be small, feedforward control is a good choice; otherwise, feedback control is the better choice. This flow is only convectively unstable (not absolutely unstable), and so the LTI system (4.7) is stable, and feedforward control alone can potentially perform well. However, for absolutely unstable flows, such as the wake behind a bluff body, the LTI system has unstable poles, and feedback control is necessary, since feedforward control cannot change the pole locations.

In the upcoming chapter (Chapter 5), we control 3D bypass transition directly from experimental data. As in the 2D TS-wave case, the sensors are placed downstream of the actuators to achieve feedback control and better reject unmodeled disturbances. In contrast to TS waves, which travel in the stream-wise direction, stream-wise streaks do not strongly vary in the stream-wise direction. This allows us to use a simpler quasi-steady control law in which the actuation updates only after the sensor measures a steady state.

Chapter 5

Experimental models and control

While the previous chapter focuses on controlling classical transition in computations, this chapter focuses on controlling bypass transition in experiments. We replicate the stream-wise streak structures characteristic of bypass transition with a spanwise array of cylindrical roughness elements. Plasma actuators, a particularly useful and practical type of actuator, are arranged into a spanwise array to generate similar streaky structures, but of opposite phase in order to attenuate the disturbance. To facilitate the design of a feedback controller, we develop an input-output model of the system where the inputs physically correspond to the roughness elements and plasma actuators. The output, computed from a spanwise array of wall-mounted shear stress sensors, is carefully chosen to target the specific spanwise wavenumber of the disturbance.

We limit our attention to quasi-steady control in which the controller updates are slower than the convective time scale, thus ignoring most transient effects. The feedback controller performs well, attenuating the stream-wise streaks both in the vicinity of the sensors and further downstream. The controller remains effective for a range of off-design flow conditions, such as when the free-stream velocity is varied.

5.1 Introduction

In recent years, there has been substantial progress in the use of model-based linear feedback control in computational flow studies [13, 62]. Notably, in [45], linear optimal control theory is applied to both classical and bypass transition. Important achievements using model reduction to reduce computational cost of control design are given in [8, 53, 101]. Simulations are best suited for these types of studies, however, it is crucial to bridge the gap between the successful control strategies in simulations that have access to the entire velocity field and experiments that have limited sensing and actuation.

Experimental boundary-layer control demonstrations are rare, but a few are successful. For example, in [93], a linear controller based on stochastically estimated transfer functions between the inputs and outputs is designed to reduce drag in a turbulent boundary layer. The feedback-feedforward controller reduces the mean wall shear stress measurements by 7%. In [54], a laminar boundary layer is disturbed with a stationary vortex pair and is effectively controlled via a synthetic-jet actuator that creates a vortex pair of opposite sign. In another

work, [77], distributed suction attenuates streaks in a laminar boundary layer caused by free-stream turbulence. The growth of the low-speed streaks is inhibited 40 boundary layer thicknesses downstream. In a numerical simulation of a similar system, it is shown that it may be possible to delay transition using an extensive spanwise sensor-actuator array and improve effectiveness using a more sophisticated controller [76]. However, as mentioned in [76], there are difficulties applying suction with solenoid valve arrangements, and future efforts should develop durable, flexible, small, and inexpensive actuators.

Possessing many of these properties, single-dielectric-barrier-discharge (SDBD) plasma actuators, herein referred to simply as plasma actuators, offer several major practical advantages: they have no moving parts, can be flush-mounted to the wall, and are relatively easy and inexpensive to construct. In previous work, plasma actuators are used to create stream-wise forces that attenuate TS waves in a laminar boundary layer under an adverse pressure gradient [35] and prevent separation [51]. Plasma actuators can also create forces in the spanwise direction. When two are separated in the spanwise direction and create spanwise forces in opposite directions, a stream-wise vortex pair is generated. Arrays of such actuator pairs are used to induce transition [95] and maintain attached flow [60, 65, 99]. Another configuration applies alternating spanwise forcing in a turbulent boundary layer [21], and another creates jet vectoring [11]. Independent of the configuration, plasma actuators operate by weakly ionizing the air near the actuator (termed plasma), which then experiences a force due to the electric field. This results in a body force on the surrounding fluid that is exploited for flow control purposes. Chapter 6 presents results on the operation and modeling of these actuators. In-depth reviews [23] and [83] explain the operation, physics, and applications.

In [42], a spanwise array of plasma actuators controls transient growth modes in a Blasius boundary layer. The actuators are arranged to generate stream-wise oriented, counter-rotating vortices, similar to those of [95], but with smaller magnitude forces such that streaks are of comparable amplitude to the targeted disturbance. It is also shown that the control effectiveness could be greatly affected by the actuator array geometry, which influences the energy contained in spatial frequencies of the flow response. For all actuators employed, the magnitude of the structure that has the targeted spanwise wavenumber is typically reduced by over 95%. Later results show that the frequency and amplitude of voltage driving the actuator also influence the energy contained in each spatial frequency [41, 88].

The goal of this study is to implement a feedback controller that inhibits transient growth in the Blasius boundary layer. The control objective is to minimize the disturbance by reducing the energy in the single spanwise wavenumber related to the streaks, as measured by the wall shear stress sensors. We create streaks at a known location using arrays of cylindrical roughness elements as also done in [33, 66, 113, 114]. This provides a tractable first problem before the more complex case of streaks at unknown locations as would be caused by free-stream turbulence.

To achieve this goal, we first develop a novel monotonic control objective as part of an empirical model of the flow's response to forcing. This model is then used to design and analyze a proportional-integral controller before implementing it in experiments. The effectiveness of the feedback controller is studied for both a steady and slowly time varying disturbance (i.e., where the time scale of variation is much larger than the convective time scale). Although transient flow effects are not addressed, we highlight issues that have

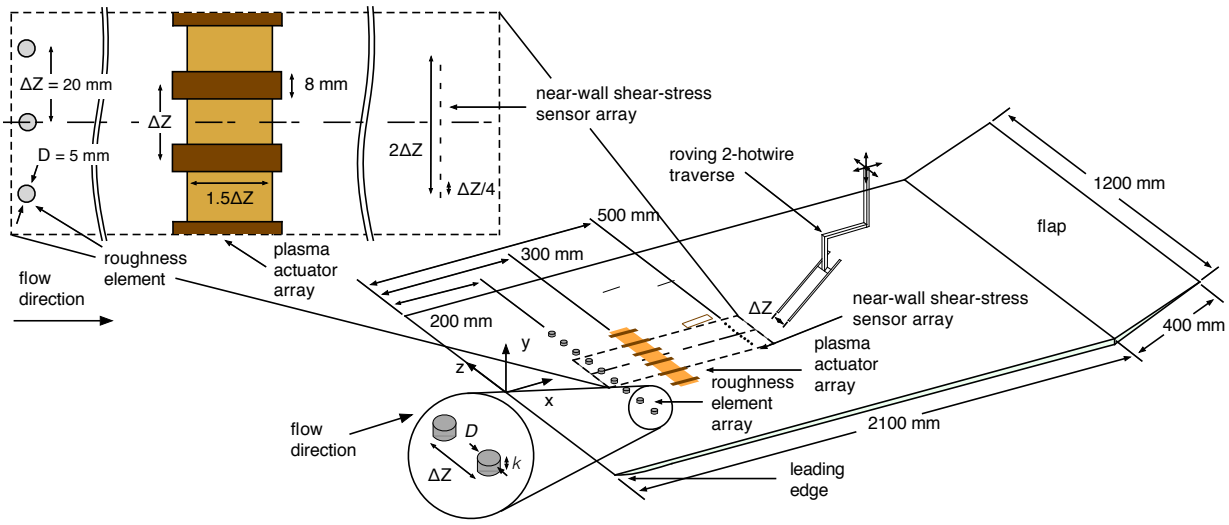


Figure 5.1: Schematic of the experimental arrangement.

not been examined previously and are critical for control at faster time scales. These include aliasing in the measurements (due to practical limitations on the number of sensors), near-wall measurements providing limited information about the flow farther from the wall, plasma actuator nonlinearity at the low voltage levels needed, and off-design performance of controllers designed with empirical input-output models.

5.2 Experimental Details

The wind tunnel (at the University of Toronto) has a working section that is 1.2 m × 0.8 m and 5 m long. The free-stream turbulence intensity is less than 0.05% of the average free-stream velocity, $U_\infty = 5$ m/s. A schematic of the boundary layer plate is shown in Figure 5.1. The plate is equipped with an asymmetric leading edge geometry designed to minimize the adverse pressure gradient over the region of interest. (See [39].) The location of the stagnation line is controlled using a flap at the downstream end of the plate.

Measurements of the stream-wise flow velocity, u are made by two hot-wire probes comprised of a 5 μm diameter copper-plated tungsten wire with an active length of approximately 1 mm. The hot-wire probes operate using a constant temperature anemometer with an overheat ratio of 1.5. Each velocity measurement consists of an average of 5 seconds of data sampled at 5 kHz. The total uncertainty of the velocity measurements is within ±1.1%. Errors are calculated using standard uncertainty analysis methodologies, see for example [81] and [109]. The resolution of the traverse system is at least 2.5 μm and the minimum precision of the traverse, based on the lead screw accuracy, is ±1 μm over a span of 10 mm. Each velocity profile consists of measurements of the stream-wise velocity at 45 wall-normal locations over the entire height of the boundary layer.

5.2.1 Base Flow

To verify the base flow is laminar, we compare measurements with the Blasius solution, and, as shown in Figure 5.2, they have good agreement. The virtual leading edge is found to be 21 mm downstream of the geometric leading edge based on measurements of the displacement thickness, and is shown in Figure 5.2, where $\hat{x}$ denotes distance from the virtual leading edge. (For information on the virtual leading edge, see [98].) The laminar boundary layer has almost no pressure gradient because the shape factor, $H_{12} = \delta^*/\theta$, where δ^* is the displacement thickness and θ is the momentum thickness, remains near the Blasius value of 2.59 (Figure 5.2c).

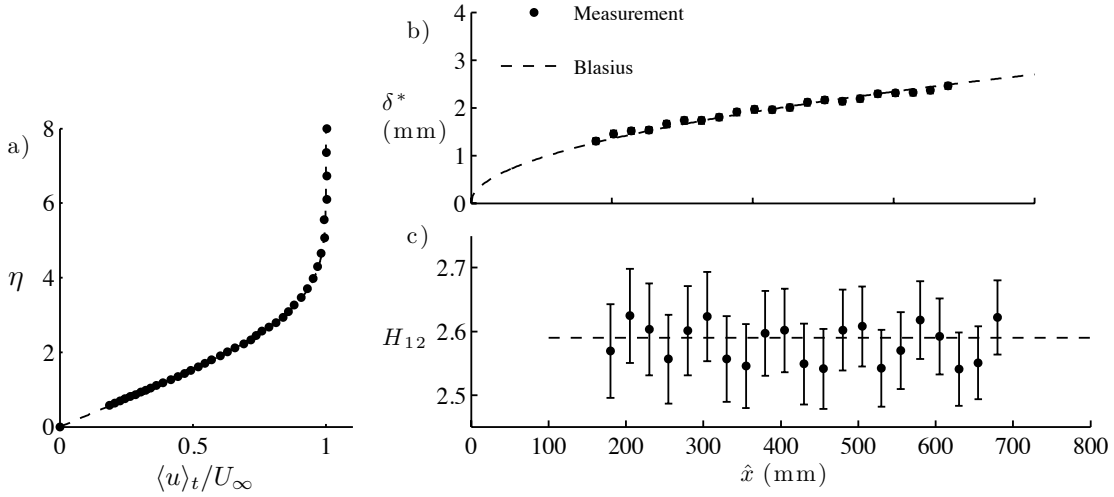


Figure 5.2: (a) Comparison of a measured boundary layer profile, where $\langle\,\rangle_t$ denotes the time average. (b) Comparison of the measured displacement thickness with the Blasius solution. (c) Stream-wise variation of the shape factor, H_{12}. The dashed line corresponds to the Blasius solution, $H_{12} = 2.59$.

5.2.2 Control System Elements

Roughness Elements

The disturbance is generated by an array of cylindrical roughness elements. These arrays generate streaks of spanwise periodic low- and high-stream-wise-velocity, replicating those seen in bypass transition [33, 113, 114]. The array consists of nine cylindrical roughness elements with heights we vary from a wall-flush position ($k = 0$) to a maximum height of $k = 1.75$ mm.

Plasma Actuator

The actuator consists of 1 μm thick copper electrodes deposited on a 0.2 mm thick borosilicate glass dielectric layer as shown in Figure 5.3. The manufacturing method of these

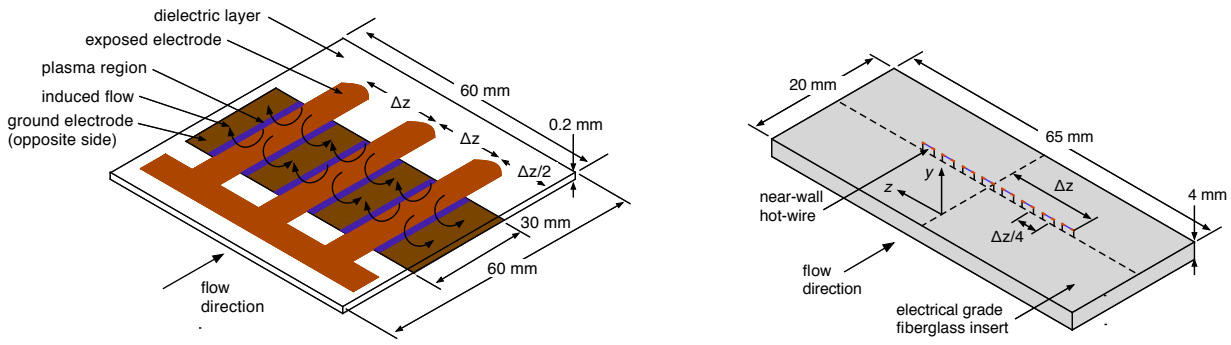


Figure 5.3: Schematic of spanwise plasma actuator (left) and shear-stress sensor array plug (right).

actuators is outlined in [48]. Two of the 60 mm square actuator tiles, each with three surface mounded high voltage (HV) electrodes, are placed side-by-side to produce the spanwise array of six evenly spaced HV electrodes shown in Figure 5.1. The electrodes are spaced $\Delta z = 20$ mm apart, the same spacing as the roughness elements, and the width of the exposed electrode is 8 mm. The width and spacing of the exposed electrodes are chosen to most reduce the energy of the disturbance streaks [42].

Wall-Shear-Stress Sensors

A spanwise array of wall-mounted hot-wire sensors are used to measure the stream-wise shear stress and thereby provide feedback information about the streaks [84]. A schematic of the sensor arrangement is shown in Figure 5.3. The sensors are spaced uniformly $\Delta z/4$ apart from one another over $2\Delta z = 40$ mm at $x = 500$ mm ($\hat{x} = 479$ mm). The arrangement provides four measurements per fundamental disturbance wavelength ($\lambda_z = \Delta z = 20$ mm). Therefore, the Nyquist wavenumber corresponds to a wavelength of $\Delta z/2$ and the presence of modes higher than the first harmonic of the fundamental produces aliasing (examined in Section 5.4.2). The sensors are arranged such that the first, middle, and last sensors are directly in-line with the roughness elements.

Each hot-wire is mounted across a pair of the support prongs (0.2 mm diameter jewellers' broaches) protruding 1 mm from the plug surface, which is within the linear region of the boundary layer. The sensing elements consists of 5μm diameter 1 mm long tungsten wires that are shouldered by copper plated regions.

5.2.3 Control System

The measurements are averaged over 0.05 s to filter high-frequency noise, and are then fed to the controller, which computes the actuator input voltage. Control iterations are made every $dt = 0.5$ s to allow for the flow to reach equilibrium.

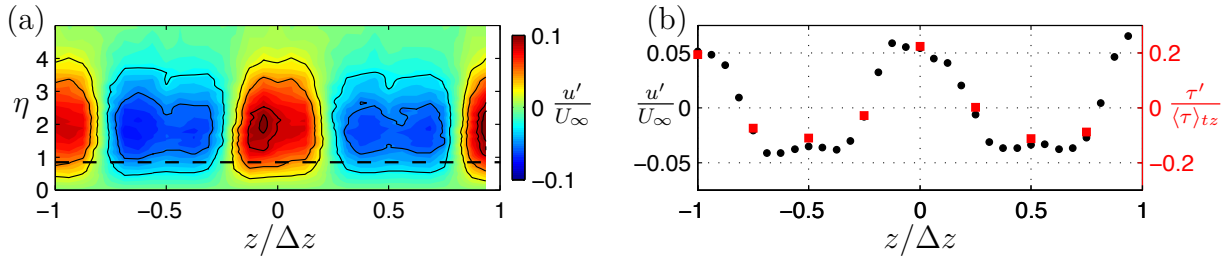


Figure 5.4: (a) Stream-wise velocity due to roughness elements at height $k = 1.5$ mm (unavailable to controller). (b) Spanwise profile of the velocity at $\eta = 0.85$ (unavailable to controller) and shear stress (available to controller).

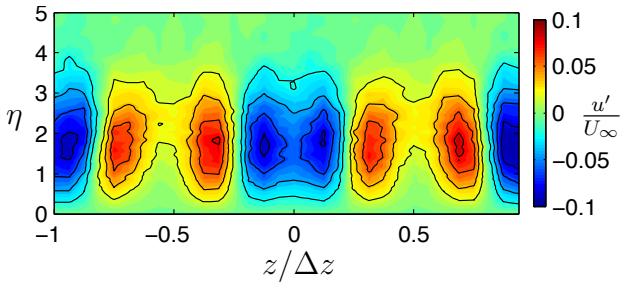


Figure 5.5: Stream-wise velocity due to plasma actuation at $V_{pp} = 5.1$ kV.

5.3 Empirical Model and Controller Design

The control objective is to cancel the growth of stream-wise streaks triggered by the roughness elements by generating counteracting structures with the plasma actuator array. To do so, we devise an empirical input-output model, then devise a controller for this model. The inputs are the roughness elements and plasma actuators and the output is a function of the shear stress measurements, as in Figure 5.1.

5.3.1 Measurements available

Only certain measurements are available from the experiments for use in developing an input-output model. We measure the effect of the roughness elements alone on the y-z plane of stream-wise velocity and shear stress at $x = 490$ mm, at a variety of different deployment heights k. An example of this data is shown in Figure 5.4, where u' is the difference of the stream-wise velocity difference from the Blasius value. Only the coarsely sampled shear stress measurements are available to the controller; the other measurements are only available to develop a model.

We also measure the effect of the plasma actuator alone on the velocity at the same y-z plane. Shear stress measurements are not available for the plasma actuator case. As shown by [42], this configuration will cause the fundamental disturbance, with wavelength defined by the spacing of the HV electrodes $\lambda_z = \Delta z$, to be of opposite phase to that produced by the roughness element array. An example of this data is shown in Figure 5.5. We note that below $V_{pp} = 3.2$ kV, the actuator does not consistently cause plasma formation.

5.3.2 Choice of Output

In choosing the output, the spatial mode of the velocity that corresponds to the streak is of primary interest. This can be isolated in frequency space as the component of velocity with wavelength $\lambda_z = \Delta z$. We refer to this component, or mode, by its wavenumber, $\kappa = 1$, where the relationship between wavenumber and wavelength is $\lambda_z = \Delta z/\kappa$. Both the roughness element and plasma actuator arrays create structures that have most of their energy in $\kappa = 1$. These $\kappa = 1$ modes have a fixed spanwise phase, so the streaks created by the roughness elements can only be affected in $\kappa = 1$ at the fixed phase of the plasma actuators. With all of this in mind, a suitable output for the model is related to the complex coefficient of $\kappa = 1$ from the Discrete Fourier Transform (DFT) of the velocity, averaged over the boundary layer. Further, we are only concerned with the component of that coefficient that has the same phase as the plasma actuator array because the other components are uncontrollable. We call this component of the DFT coefficient the "projected velocity," φ_{Cu}. Since velocity measurements are only available offline, we use an analogous quantity as the output, the projected shear stress $\varphi_{C\tau}$, and find that is proportional to φ_{Cu}. (We denote the shear stress as τ and the difference from the Blasius value as τ'.)

Now we define the output $\varphi_{C\tau}$ mathematically. The shear stress measurements are Discrete Fourier Transformed, and the $\kappa = 1$ coefficient is denoted $\tilde{\tau}_1$. This coefficient is normalized by the average shear stress (roughly equal to that of the Blasius boundary layer) at the same stream-wise location, yielding $\tilde{\tau}_1' = \tilde{\tau}_1/\langle\tau\rangle_{tz}$. Similarly, the Fourier coefficient for $\kappa = 1$ of the velocity is $\langle\tilde{u}_1'\rangle_y$, averaged over the wall-normal direction to result in a single complex number. These complex numbers physically represent the phase and magnitude of the $\kappa = 1$ Fourier coefficient. The phase that can be controlled by the plasma actuators is the angle of complex number $\langle\tilde{u}_1'\rangle_y$ which is nearly identical to that of $\tilde{\tau}_1'$. Normalizing $\langle\tilde{u}_1'\rangle_y$ gives

$$\tilde{C}_u = \frac{\langle\tilde{u}_1'\rangle_y}{\|\langle\tilde{u}_1'\rangle_y\|}, \tag{5.1}$$

where the velocities on the right hand side of (5.1) result from plasma actuation (not the roughness elements). Considering complex numbers as two-dimensional vectors, the projected shear output can be written as a dot product,

$$\text{output} = \varphi_{C\tau} = \text{dot}(\tilde{C}_u, \tilde{\tau}_1') = \text{real}(\tilde{C}_u)\cdot\text{real}(\tilde{\tau}_1') + \text{imag}(\tilde{C}_u)\cdot\text{imag}(\tilde{\tau}_1'). \tag{5.2}$$

A similar quantity, projected velocity, is defined as

$$\varphi_{Cu} = \text{dot}(\tilde{C}_u, \tilde{u}_1'). \tag{5.3}$$

Due to the spanwise arrangement of the roughness elements relative to the plasma actuators, the coefficient for $\kappa = 1$ caused by only the roughness element array has a negative value of $\varphi_{C\tau}$ and φ_{Cu}, whereas the one caused only by the plasma actuators is positive. The goal of the controller can now be simply stated as driving $\varphi_{C\tau}$ towards zero by canceling the effect of the roughness elements.

This choice of output, $\varphi_{C\tau}$, is better suited for control purposes than the spanwise power spectrum of τ' used previously in [40]. The power spectrum yields information only about the strength of the spanwise sinusoidal shear stress variation without providing an indication

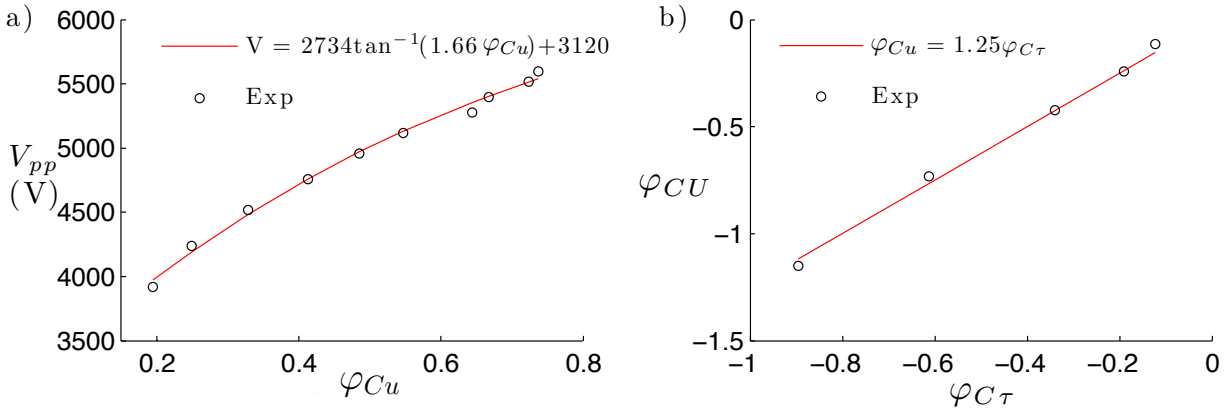


Figure 5.6: (a) Relationship between the actuator voltage and projected velocity with the inverse tangent fit. (b) The linear relationship between the projected velocity and shear for the disturbance cased by the roughness element array.

of the spanwise spatial phase of the disturbance. Therefore, an *ad hoc* treatment is required to detect the direction of the controller in [40]. The underlying issue is that controllers are simpler to devise when the output varies monotonically with the input. The output used in this work, $\varphi_{C\tau}$, varies monotonically with the input voltage, as shown in Figure 5.6, while the power spectrum is quadratic—not monotonic since $\varphi_{C\tau}$ can be negative.

5.3.3 Empirical Flow Model

We empirically find an input-output model from V_{pp} to $\varphi_{C\tau}$ from the data, and, for now, set aside the effect of the roughness elements. First, the relationship between the actuator voltage input and the intermediate variable, projected velocity φ_{Cu}, is fit with an inverse tangent function, as shown in Figure 5.6(a). In practice, any monotonic function that fits the observed data is acceptable for the purposes of this model. Then, the relationship between the projected velocity, φ_{Cu}, and shear, $\varphi_{C\tau}$, is determined from measurements of the disturbance caused by the roughness elements, and found to be approximately linear, as shown in Figure 5.6(b). The array of roughness elements is deployed from 0.75 to 1.75 mm in increments of 0.25 mm. From these empirical fits, the following relationships are derived:

$$\varphi_{Cu} = m \cdot \varphi_{C\tau}, \tag{5.4a}$$

$$V_{pp} = c_1 \cdot \tan^{-1}(c_2 \cdot \varphi_{Cu}) + c_3, \tag{5.4b}$$

where $m = 1.25$, $c_1 = 2734$, $c_2 = 1.66$, and $c_3 = 3120$. Together, these equations define the relationship between the input V_{pp} and the output $\varphi_{C\tau}$.

5.3.4 Linearization and Control Model

The nonlinearity makes it difficult to perform control analysis directly [1]. Since the form of the nonlinearity is known, the terms are regrouped into a modified plant which takes a new

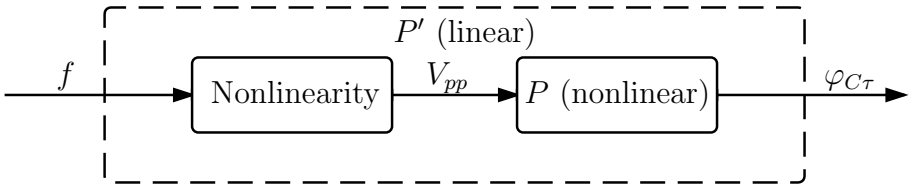


Figure 5.7: The linear plant, P', takes input f, internally converts that via Equation (5.5) to V_{pp}, then outputs $\varphi_{C\tau}$ via Equation (5.6).

input f, given by

$$f \equiv \tan\left(\frac{V_{pp} - c_3}{c_1}\right). \tag{5.5}$$

Equation (5.4) is then rewritten in terms of f,

$$\varphi_{C\tau} = \frac{f}{m \cdot c_2}, \tag{5.6}$$

which is a *linear* relationship between the input f and output $\varphi_{C\tau}$. Figure 5.7 is an illustration of the modified block diagram which takes the input f, and outputs $\varphi_{C\tau}$.

Up to this point, the effect of the roughness elements has been set aside. That effect is now included and assumed to act in the same way as the plasma actuators so that the input to P' is the sum $f + d$, where d is the input that corresponds to the roughness elements. The disturbance signal is shown as part of the closed-loop block diagram in Figure 5.8. This is a common way to include disturbances for control purposes, and is physically motivated in this case because the plasma actuators are specifically chosen to create the same streak structures as the roughness elements. The time evolution of the output is given as

$$\varphi_{C\tau}^{i+1} = \frac{f^i + d^i}{c_2 \cdot m}, \tag{5.7}$$

where i is the discrete time step. The time step corresponds to the time between measurements, chosen to allow the flow to reach an equilibrium, and is $dt = 0.5$ seconds. In comparison, the convective time scale, taken to be the time for the free-stream to travel from the actuators to the sensors, is 0.04 seconds. The left-hand side of the equation is a time step later because the measurement is taken one time step after the input is applied. The system can be expressed in discrete time as a state-space system with one state,

$$\begin{aligned} x_{P'}^{i+1} &= 0 \cdot x_{P'}^{i} + \frac{f^i + d^i}{m \cdot c_2} \\ \varphi_{C\tau}^{i} &= x_{P'}^{i}, \end{aligned} \tag{5.8}$$

where the state, $x_{P'}$, is only an intermediate variable. Equation (5.8) is the empirical input-output linear model of the plant, P', and facilitates the design of a controller, K.

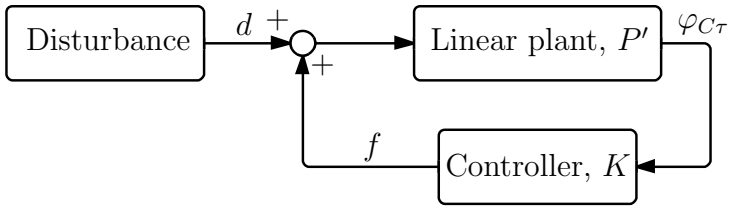


Figure 5.8: Control scheme with the output defined and a linear plant.

5.3.5 PI Controller Design

In this section we design a proportional-integral-derivative (PID) controller for the model, as depicted in Figure 5.8. Proportional-integral-derivative (PID) controllers are widely used in feedback control for their good performance, robustness properties, and simplicity. The integral term ensures that the system can reach the target value, whereas the derivative term typically improves closed-loop stability. A proportional-integral (PI) controller is chosen for this work because the derivative term is highly sensitive to noise and model uncertainty. Moreover, the plant has only one state, so PI control is equivalent to pole placement.

To choose the PI controller gains, the performance and robustness are analyzed. A measure of performance is how quickly the controller drives $\varphi_{C\tau}$ to zero, whereas robustness is the ability of the controller to perform well in off-design conditions. The controller system, K, evolves as

$$x_K^{i+1} = x_K^i + dt\varphi_{C\tau}^i, \tag{5.9}$$

$$f^i = -K_I x_K^i - K_P \varphi_{C\tau}^i, \tag{5.10}$$

where x_K is the time integral of $\varphi_{C\tau}$ needed for integral feedback. The proportional and integral gains are denoted by K_P and K_I, respectively. The time between controller updates, as previously defined, is $dt = 0.5$ s. The plant and controller systems are combined to yield the state-space equations of the controlled system

$$\begin{pmatrix} x_{P'}^{i+1} \\ x_K^{i+1} \end{pmatrix} = \begin{bmatrix} \frac{-K_P}{c_2 \cdot m} & \frac{-K_I}{c_2 \cdot m} \\ dt & 1 \end{bmatrix} \begin{pmatrix} x_{P'}^i \\ x_K^i \end{pmatrix} + \begin{bmatrix} \frac{1}{c_2 \cdot m} \\ 0 \end{bmatrix} d^i, \tag{5.11}$$

$$\varphi_{C\tau}^i = \begin{bmatrix} \frac{1}{c_2 \cdot m} & 0 \end{bmatrix} \begin{pmatrix} x_{P'}^i \\ x_K^i \end{pmatrix}. \tag{5.12}$$

For the best performance, the state should decay to zero as quickly as possible. For this discrete time system, this implies the eigenvalues of the 2×2 matrix in (5.11) should be zero, i.e., we want a so-called dead beat controller. It is easily shown that $K_P = K_I/dt = c_2 \cdot m$ makes both eigenvalues zero.

Robustness is considered by the infinity norm of the sensitivity function, $\|S\|_\infty$, which is a measure of robustness to plant model uncertainty (such as would be caused by a change in free-stream velocity), and $\|\cdot\|_\infty$ is the infinity norm. The sensitivity function is given by

$$S \equiv \frac{1}{1 + P'K}, \tag{5.13}$$

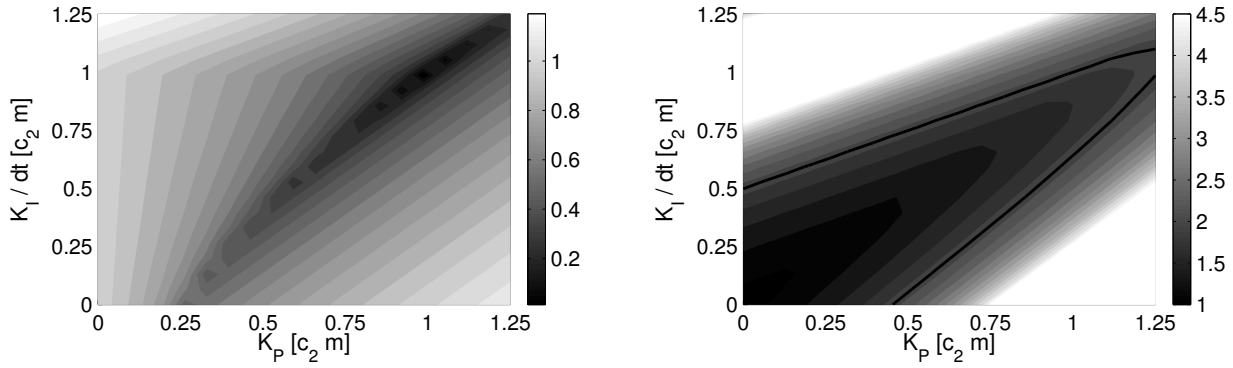


Figure 5.9: The maximum eigenvalues of the 2×2 matrix in (5.11) with control gains (left), and the value of the infinity norm of the sensitivity function (right). The solid line indicates $\|S\|_\infty = 2$.

A general guideline is for the value of $\|S\|_\infty$ to be small, typically in the range of 1.3 to 2. The contour plots in Figure 5.9 demonstrate the performance (maximum eigenvalue) and robustness ($\|S\|_\infty$) for a range of PI controller gains. The best performing K_P and K_I (zero eigenvalues) result in $\|S\|_\infty = 2$. For convenience, we define $K_P = K_I/dt = G_c \cdot c_2 \cdot m$ where G_c is the gain coefficient. To accommodate a larger degree of uncertainty in the model of the plant we set $G_c = 0.5$, yielding $\|S\|_\infty = 1.33$.

Results using various values of G_c are shown in Figure 5.10. At iteration 0, the system is undisturbed. At the next iteration, a disturbance corresponding with the maximum value considered in the experiments is applied. Control is applied in all subsequent iterations. The case with the best performance corresponds with $G_c = 1$. Further increases in the gain result in oscillation of the output. When $G_c = 0.5$, the controller effectively reduces $\varphi_{C\tau}$ to zero after only a few iterations. Further decrease of the gain slows the controller's response.

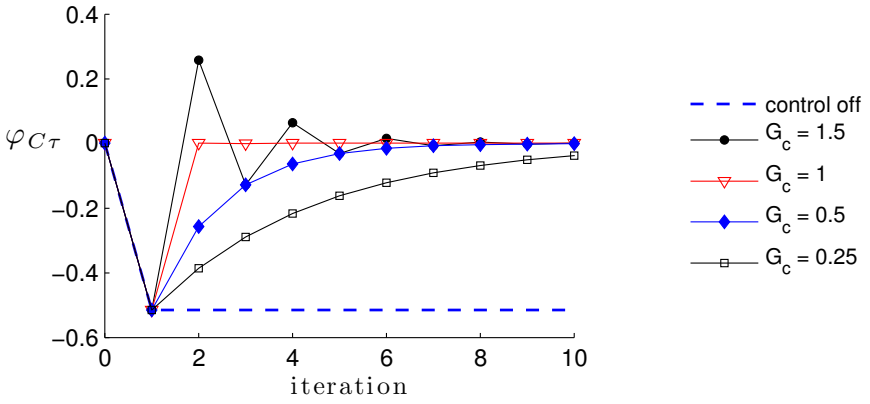


Figure 5.10: Output of the feedback controlled model (not experiment) plant, demonstrating the variation in the response of the PI closed-loop controller to a typical steady disturbance.

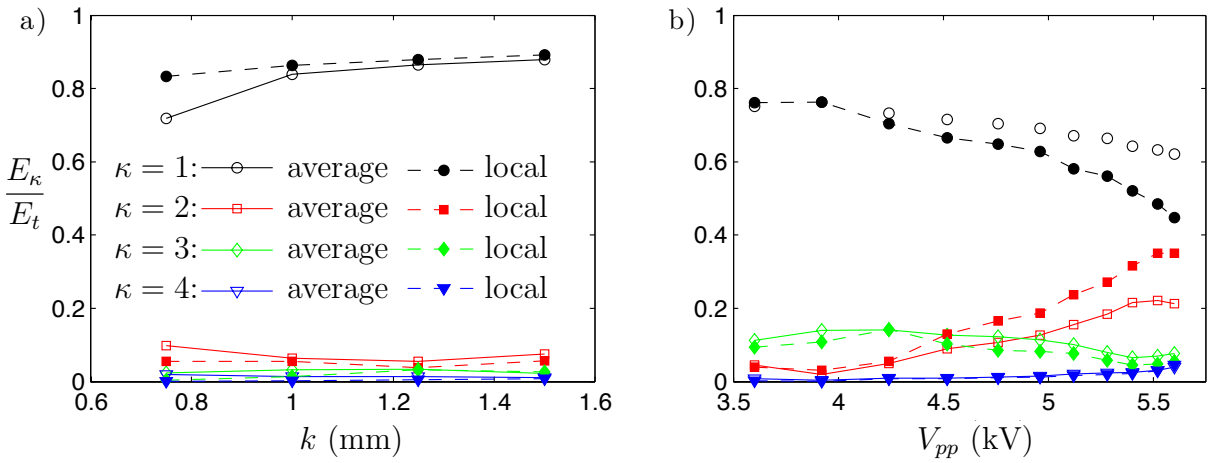


Figure 5.11: The variation of the energy contained in wavenumbers from the near-wall velocity measurements at $\eta = 0.85$, and the boundary layer averaged measurements. (a) For the roughness element array, and (b) for the actuator.

5.4 Limitations of wall shear stress sensors

This section discusses the limitations of the shear stress sensors by comparing to measurements over the entire thickness of the boundary layer.

5.4.1 Comparison with measured velocity planes

Now we quantify how representative the near-wall measurements are of the disturbance higher in the boundary layer. To do so, we compare the energy in the fundamental and first three harmonics obtained from velocity measurements at 1 mm above the wall ($\eta = 0.85$) to that obtained by integrating the energy measured at all heights within the boundary layer.

$$\frac{E_\kappa}{E_t} = \frac{(\phi(u'))_\kappa}{\sum_{i=1}^{8}(\phi(u'))_i}, \tag{5.14}$$

where E_κ is the energy in the wavenumber κ, and E_t is the energy integrated over all wavenumbers, $\phi(u')$ is the power spectrum of u' over all wavenumbers, and the subscript κ denotes the value of the power spectrum corresponding to wavenumber κ. (For reference, a non-dimensional quantity that is sometimes used is $\beta_\kappa = 2\pi\delta/(\kappa\Delta z)$, where $\delta = \sqrt{\nu x/U_\infty}$ is the Blasius boundary layer thickness.) The boundary layer averaged disturbance is evaluated over the thickness of the boundary layer,

$$\langle(\phi(u'))_\kappa\rangle_y = \frac{\int_0^{5\delta}(\phi(u'))_\kappa \, \mathrm{dy}}{5\delta}. \tag{5.15}$$

The comparison of the near-wall and average energy ratios for different wavenumbers is shown in Figure 5.11(a) for the roughness array disturbance, and (b) for the actuator. These figures show distinctly different trends. As the roughness element deployment height increases, the local and averaged energy ratios reach approximately the same constant value.

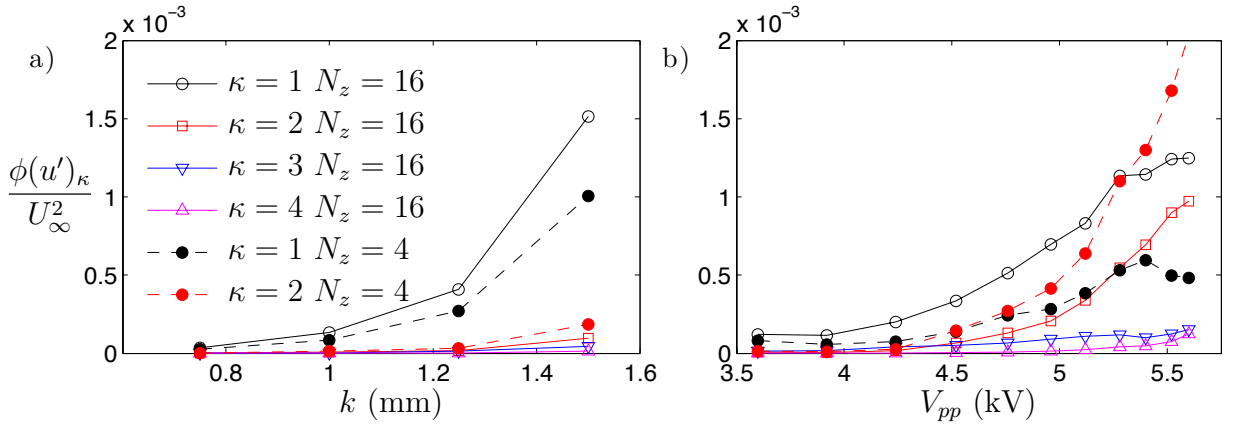


Figure 5.12: From $N_z = 16$ velocity measurements over Δz, the energies in the first four wavenumbers are plotted and have no aliasing. The energies in the first two wavenumbers are plotted with $N_z = 4$ velocity measurements over Δz, and have aliasing. The flow is disturbed in (a) by roughness elements and actuated in (b) by plasma actuators.

In contrast, as voltage to the plasma actuators increases, the local and averaged energy ratios differ. These results imply that at higher voltages the near-wall sensors under predict the boundary-layer averaged energy. They also imply that the roughness elements and plasma actuators induce slightly different spatial distributions in the flow, as seen before in Figures 5.4 and 5.5.

As we will see later in Section 5.5.4, this under prediction of shear stress results in slight over actuation when higher voltages are needed. In short, the shear stress measurement is driven to zero effectively by the controller, but the true contribution from the plasma actuator to shear stress is larger, and thus the controller slightly over actuates.

5.4.2 Aliasing due to limited spanwise resolution

Due to hardware limitations, we employ four stream-wise shear stress measurement locations per fundamental disturbance wavelength. For this arrangement, the smallest resolved wavelength is $\Delta z/2$, and the corresponding (Nyquist) wavenumber is $\kappa = 2$. The contribution of higher, unresolved, wavenumbers alters the apparent (i.e., measured) content of the first two modes. The influence of this aliasing on the shear stress measurements is found by comparing more densely sampled velocity data (from the roving hot-wire) at the same height above the wall as the shear sensor ($\eta = 0.85$).

Figure 5.12 shows the energy contained in the first four modes calculated by downsampling the 16 points per Δz to 4 points per Δz. These results are compared against the alias-free modal energy content obtained without downsampling. More aliasing error is present when measuring the reaction of the flow to roughness elements than to plasma actuators, as shown in Figure 5.12(b). Significant aliasing occurs due to the large energy in the unresolved third and fourth modes.

In summary, Sections 5.4.1 and 5.4.2 show that the effect of the actuator is measured by the shear sensor array as being less energetic than if it is measured over the entire boundary

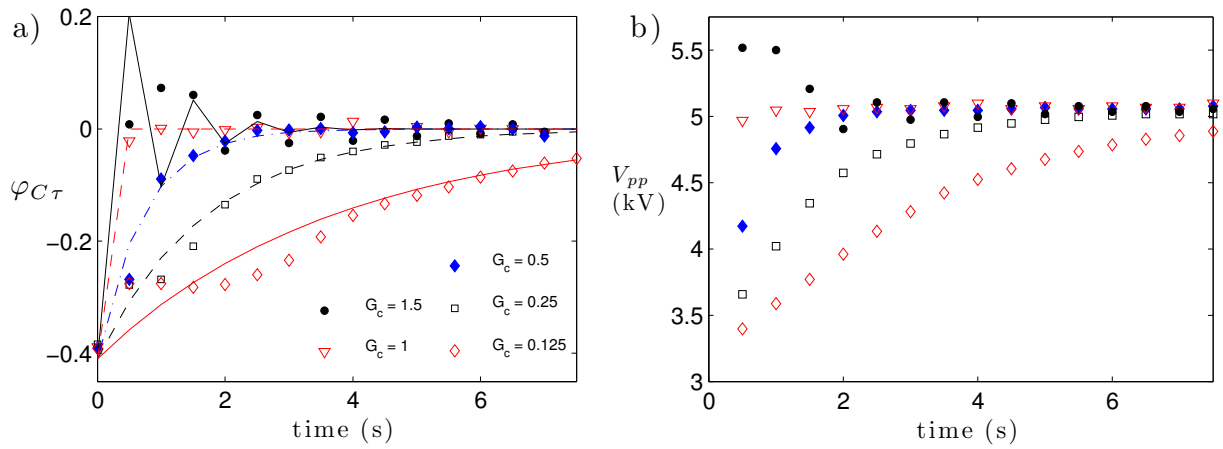


Figure 5.13: (a) Time variation of the control output $\varphi_{C\tau}$ for $k = 1.25$ mm and $U_\infty = 5.0$ m/s using different G_c. Markers represent the experimental measurements, and lines depict results from the simulations based on (5.11) and (5.12). (b) Corresponding plasma actuator voltage.

layer. Further, it is shown that the plasma actuators effect on the $\kappa = 1$ mode (corresponding to a wavelength of Δz) is concentrated nearer the wall than is the effect of the roughness elements. It is also shown that aliasing is more significant for the actuator disturbance. The implications of these results are discussed in Section 5.5.4.

5.5 Experimental Flow Control Results

Experimental results regarding the performance and robustness of feedback control are discussed in this section. The effects of control for a range of gain values and free-steam velocities are considered, including values different from the value at which the input-output models are obtained. In the process, notable plasma actuator characteristics are exposed and discussed.

5.5.1 Effect of Controller Gain

The gains of the controller, K_P and K_I, affect the performance of the closed-loop controlled system. As shown in Figure 5.9, when $K_P = K_I/dt$ good performance and robustness are achieved and therefore they are kept equal. Recall from Section 5.3.5 that $K_P = K_I/dt = c_2 \cdot m$ optimizes performance, but $K_P = K_I/dt = G_c \cdot c_2 \cdot m$, where $G_c = 0.5$ (the gain coefficient), improves robustness to uncertainty in the input-output models and flow conditions. Figure 5.13(a) shows the control results with roughness elements deployed to a height of $k = 1.25$ mm, $U_\infty = 5$ m/s, and several G_c.

Larger values of G_c lead to damped oscillation of the control response, as shown for $G_c = 1.5$. Lowering G_c so it is less than one leads to an over-damped response, as shown in Figure 5.13(a). The experiments deviate from the model for $G_c < 0.5$ due to the non-monotonic actuator behavior in experiments at low voltage, which is discussed next.

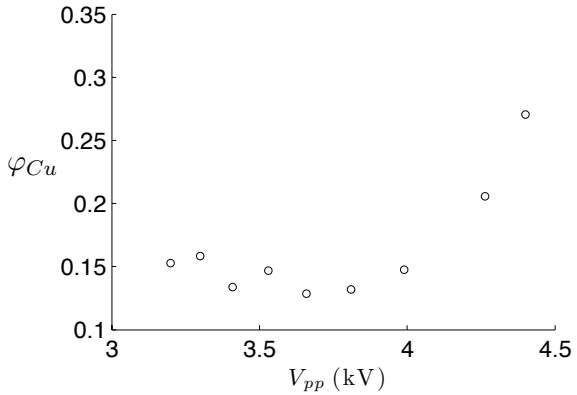


Figure 5.14: Variation of φ_{Cu} with low actuator excitation voltages.

In Figure 5.13(b), the case of $G_c = 0.125$ has actuator voltages below 3.9 kV, the lowest voltage we use in developing the model. During the first few iterations, the actuator has a larger effect than the model predicts. This is seen in in Figure 5.13(a) where the output $\varphi_{C\tau}$ is significantly closer to zero than it is for the model. This is caused by non-monotonic behavior of the actuator at lower V_{pp} values.

The non-monotonic behavior is clearly shown in Figure 5.14. For voltages less than 3.2 kV, the electric field is insufficient to ionize air and produce a body force. As voltage increases from 3.2 and 3.8 kV, the strength of actuation, as quantified by $\varphi_{C\tau}$, *decreases*. Further increases in voltage result in *increasing* $\varphi_{C\tau}$, as one expects. This low range of voltage frequency and amplitude is not addressed in the literature, which instead focuses on larger amplitudes. The work [88] shows that lowering the actuator voltage can shift the location of the maximum disturbance velocity downstream. The shift of this location is significantly greater for smaller voltages, corroborating the low-voltage behavior we see.

Nonetheless, the feedback controller is effective, as shown in Figure 5.13(a), because it is designed to be robust to unmodelled behavior. This is a clear improvement over the *ad hoc* controller in [40] that is destabilizing when lower voltages are needed. Note though that further decreases in the free-stream velocity would destabilize this controller as well.

5.5.2 Effect of Aliasing

When the roughness element is higher, at $k = 1.375$ mm, and $U_\infty = 5$ m/s, the aliasing contributes to destabilizing the closed-loop system. This is shown in Figure 5.15. The controller gains are $K_P = K_I/dt = 0.5 \cdot c_2 \cdot m$, as they are when control is effective at lower roughness height $k = 1.25$ mm (Figure 5.13). For $k = 1.375$ mm, the controller increases the actuator voltage beyond 5.3 kV, and in this regime, as seen in Figure 5.12(b), increasing voltage *decreases* the measured energy in the $\kappa = 1$ mode, and thus $\varphi_{C\tau}$, due to aliasing from the $\kappa = 3$ mode. The controller compensates by increasing the voltage, which only further decreases $\varphi_{C\tau}$, and is therefore destabilizing.

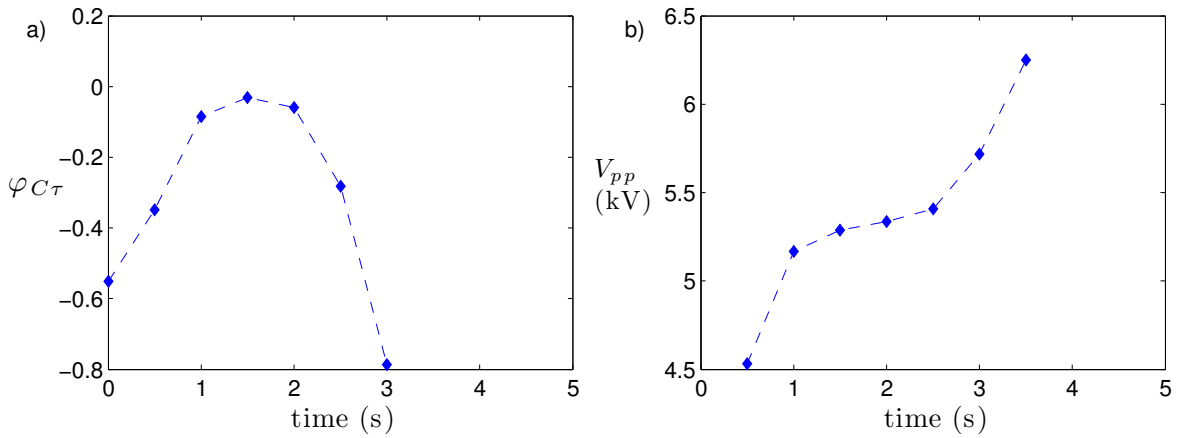


Figure 5.15: Time evolution of the control objective and input voltage for $k = 1.375$ and $U_\infty = 5$ m/s.

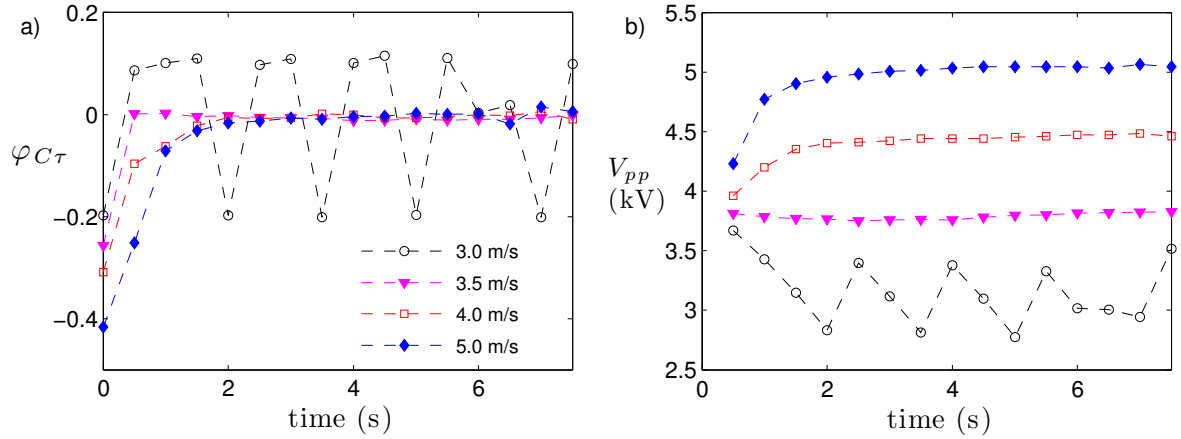


Figure 5.16: Influence of varying the free-stream velocity on the controller's effectiveness for $k = 1.25$ mm.

5.5.3 Effect of Free-Stream Velocity

Robustness to off-design conditions is shown by controlling the flow at different free-stream velocities, 3, 3.5, 4, and 5 m/s. The controller gains are $K_P = K_I/dt = 0.5 \cdot c_2 \cdot m$, as in Section 5.3.5. The results for two different roughness element heights ($k = 1.25$ mm and 1.375 mm) are shown in Figures 5.15 and 5.16.

Reducing the free-stream velocity reduces the magnitude of the disturbance and, thus, the required actuation voltage. For $k = 1.25$ mm and $U_\infty = 3.5$, 4, and 5 m/s, the controller quickly drives $\varphi_{C\tau}$ towards zero as shown in Figure 5.16(a). When $U_\infty = 3$ m/s, the voltages are low, in the range of the inverse behavior previously described and shown in Figure 5.14. Thus, when the controller compensates for a positive $\varphi_{C\tau}$ by reducing the voltage, $\varphi_{C\tau}$ either further increases at the next iteration, or, if the voltage is below 3.2 kV, the plasma does not form at all and $\varphi_{C\tau}$ reduces significantly. This pattern is shown in Figure 5.16(a).

Results for $k = 1.375$ mm are shown in Figure 5.15. For the free-stream velocities of 3,

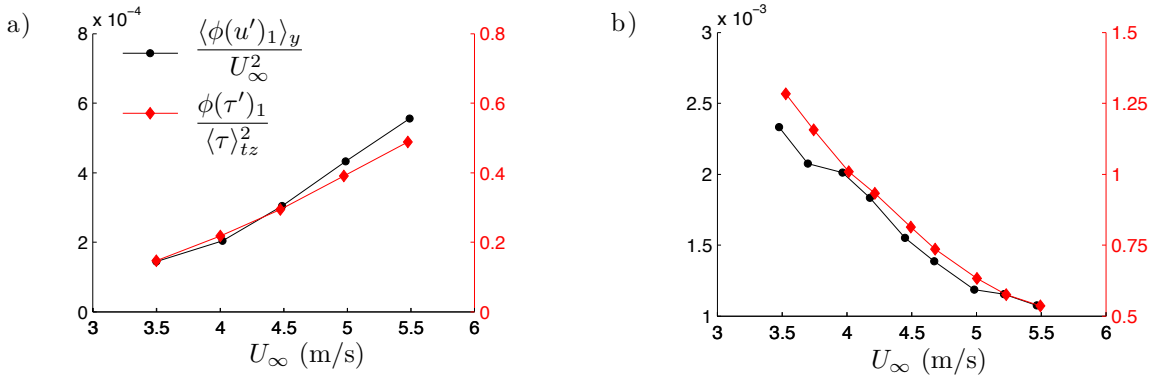


Figure 5.17: The energy in the $\kappa = 1$ mode, as measured near the wall and over the entire boundary layer and as a function of free-stream velocity. The response to roughness elements is shown in (a) and the response to plasma actuators in (b). (There is no feedback control in these plots.)

3.5, and 4 m/s, the controller drives $\varphi_{C\tau}$ towards zero in a few iterations, as shown in Figure 5.16(b). When $U_\infty = 5$ m/s, the controller increases the actuator voltage beyond 5.3 kV, and in this regime, as seen in Figure 5.12(b), increasing voltage *decreases* the measured energy in the $\kappa = 1$ mode, and thus $\varphi_{C\tau}$, due to aliasing from the $\kappa = 3$ mode. The controller compensates by increasing the voltage, which only further decreases $\varphi_{C\tau}$, and is therefore destabilizing, just as it is for aliasing.

The effects of the roughness elements and plasma actuators on the flow change in opposite ways with respect to changes in the free-stream velocity. Increased free-stream velocity *increases* the magnitude of the streaks generated by the roughness elements, but increased free-stream velocity *decreases* the magnitude of the counter-streaks generated by the plasma actuators. This is demonstrated in Figure 5.17. In the case of the actuator, constant power is imparted on the flow over the actuator length and an increase in free-stream velocity decreases the residence time of the fluid over the actuator, thereby decreasing the total energy imparted on a fluid particle.

5.5.4 Characterization of the Controlled Flow

In this section, we examine the effectiveness of the control over the entire boundary layer thickness, not just near the wall. Figure 5.18(a) shows the uncontrolled flow and Figure 5.18(b) the controlled flow after reaching a steady state with a voltage of approximately 5 kV ($k = 1.25$ mm in both). The corresponding wall-normal energy distributions are shown in Figures 5.18(c & d) for the uncontrolled and controlled flow, respectively. The energy in the targeted mode 1 is substantially reduced. The remaining energy is located predominantly in the lower half of the boundary layer and in the third spanwise mode. This distribution of energy is similar to that created by only the actuator, as in Figure 5.11(a), and occurs because the roughness element array produces almost no disturbance in the $\kappa = 3$ mode at the measurement plane (Figure 5.11(b)).

The targeted $\kappa = 1$ mode is isolated via band-pass spatial filtering of u'/U_∞ shown in

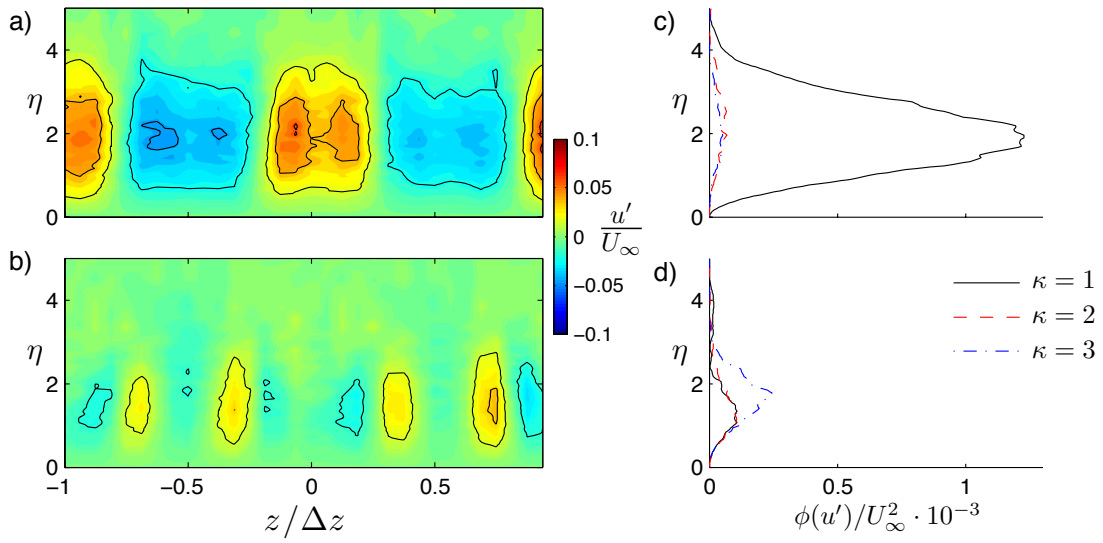


Figure 5.18: Stream-wise velocity for (a) the uncontrolled flow with $k = 1.25$ mm, and (b) the controlled flow with $V_{pp} = 5.0$ kV. Wall-normal disturbance energy profiles are shown for the first three modes for (c) the uncontrolled and (d) the controlled flow.

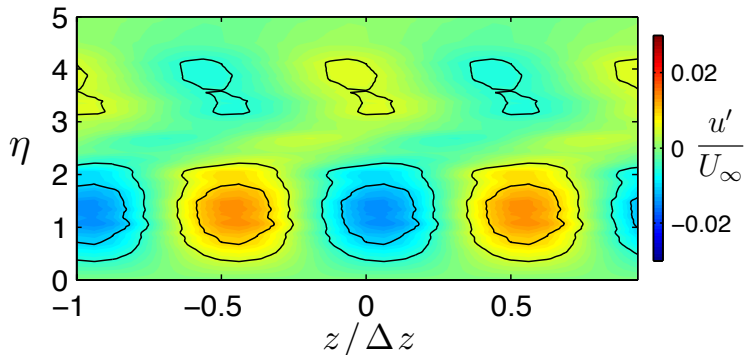


Figure 5.19: Targeted $\kappa = 1$ mode of the stream-wise velocity of the controlled flow.

Figure 5.19. Approximately 75% of the residual energy in the $\kappa = 1$ mode is contained in the near-wall region, $\eta < 2.5$ where there are residual streaks (e.g., a low-speed streak at $z/\Delta z = 0$). For $\eta > 2.5$, however, the residual streaks have opposite sign.

The cause for this behavior stems from earlier observations in Section 5.4.2. The wall-based measurement of the $\kappa = 1$ mode energy is smaller than the measurement made over the entire boundary layer height. This error is attributed to aliasing, which is more significant for the actuator than the roughness disturbance, as shown in Figure 5.12. Based on these measurements, the actuation is slightly too strong over the height of the boundary layer. Since the plasma actuators generate more response in the $\kappa = 1$ mode in the near-wall region ($\eta < 2.5$) and less far from the wall ($\eta > 2.5$) than do the roughness elements (as suggested by Figure 5.11), we see the distribution in Figure 5.12. Again, despite these effects, the control output indicates that the disturbance is attenuated (i.e., $\varphi_{C_\tau} \approx 0$), because the signal is based only on shear-stress measurements near the wall.

5.5.5 Stream-wise Evolution of the Control Effect

In the previous sections, the control results are presented for a single measurement plane at one stream-wise location. In this section, the effect of the control on the transient growth of streaks is examined over multiple stream-wise locations. Three different flow conditions are examined. In the first two, the free-stream velocity is 4 and 5 m/s and $k = 1.25$ mm. For the third case, the free-stream velocity is 4 m/s and $k = 1.375$ mm.

For the case with $U_\infty = 5$ m/s and $k = 1.25$ mm, contour plots of u'/U_∞ for the uncontrolled flow are shown in Figure 5.20. As the flow developed, energy in higher modes decays and past $x = 500$ mm almost all that remains is the $\kappa = 1$ mode. The measurements of the controlled flow are shown in Figure 5.21. The measurement planes upstream of the plasma actuators are nearly identical to the uncontrolled case, and so they are omitted. The controller effectively eliminates the energy in the targeted mode 1, and so it is predominantly higher modes that remain downstream, consistent with the results shown in Figure 5.18.

The effectiveness of the control over the measurement region is quantified by the energy in $\kappa = 1$ as $\overline{\phi(u')}_1/U_\infty^2$, as shown in Figure 5.22. The actuator attenuated over 90% of the energy contained in the target mode downstream of the measurement plane ($x = 500$ mm). Further upstream, the controller is less effective, while further downstream the structures exponentially decay.

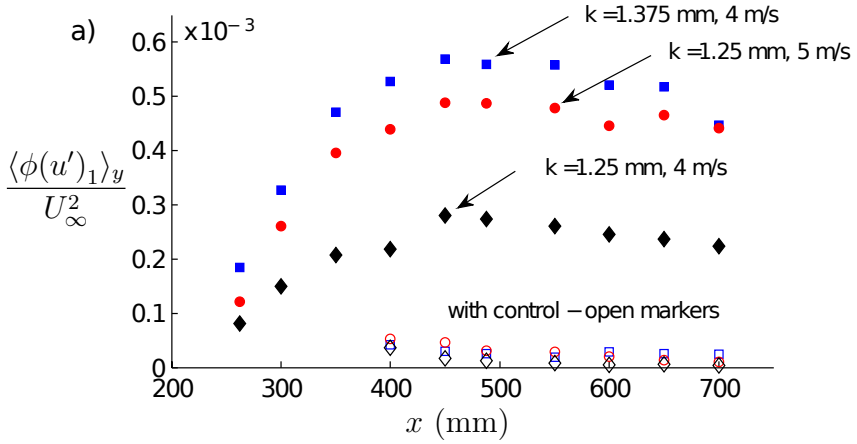


Figure 5.22: Effect of control as a function of downstream location.

5.5.6 Continuous Free-Stream Velocity Perturbation

We study the ability of the controller to reject the disturbance as the free-stream velocity slowly varies in time. We categorize the variation of the free-stream velocity as slowly time varying because the time scale of the variation is three orders of magnitude larger than the period of time it takes for flow to convect from the actuators to the sensors.

The free-stream velocity is varied sinusoidally between 4 m/s and 5 m/s with period $T_w = 50$ s. To see how well the controller performs in response to different disturbance frequencies, one would naturally vary T_w, but T_w is fixed due to wind tunnel constraints. Instead, we vary the time step dt so the ratio dt/T_w is 0.01, 0.02, 0.04 and 0.08 (dt is 0.5, 1.0, 2.0, and

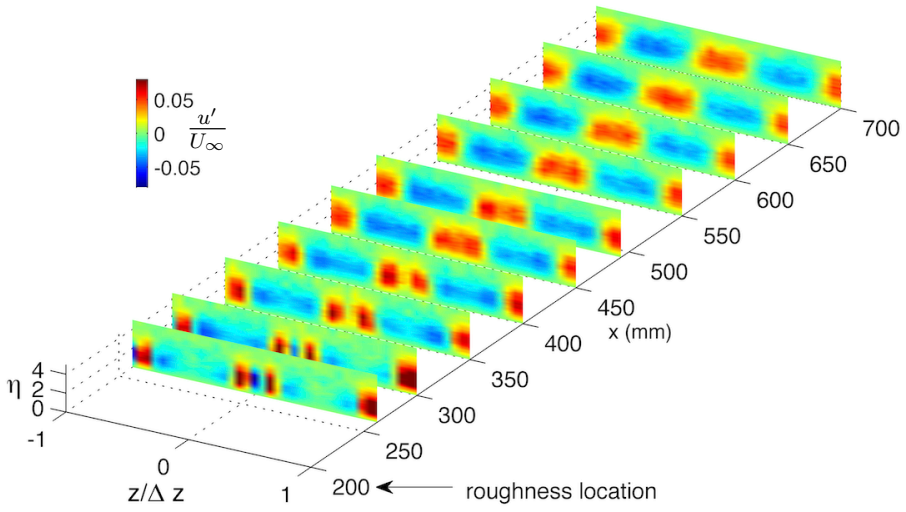


Figure 5.20: Stream-wise evolution of the disturbance caused by a roughness array with $k = 1.25$ mm and $U_\infty = 5$ m/s.

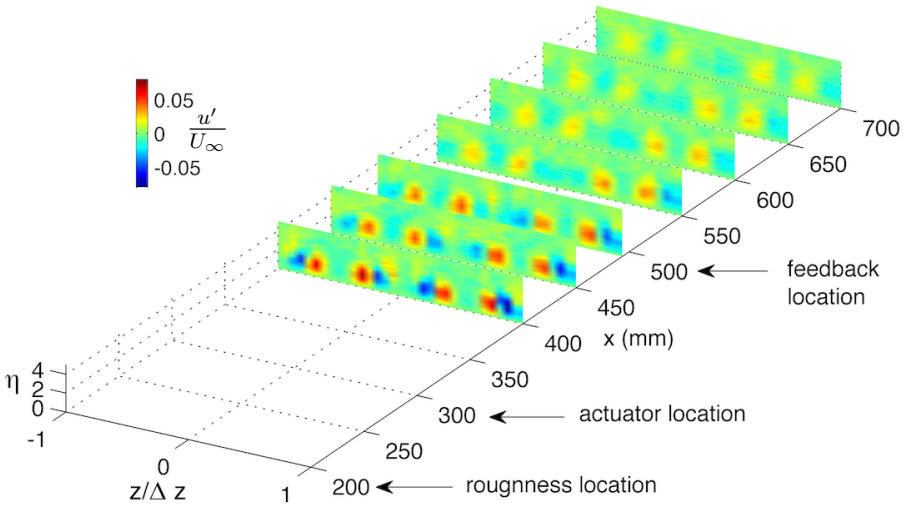


Figure 5.21: Stream-wise evolution of the flow shown in Figure 5.20 with control by the plasma actuator operated at 5.05 kV.

4.0 s). However, this has the additional effect of changing the control gains. We keep the ratio of K_I/dt constant (Equation (5.10)) so that $K_I/dt = K_p = G_c c_2 m$ and $G_c = 0.5$. This keeps the measure of robustness constant at $|S|_\infty = 1.33$, an acceptable level, for each dt, but the performance varies. The experimental results for each of the four cases are shown in Figure 5.23. We also find the frequency response of the open loop gain, $P'K$, as shown in Figure 5.24. There is a different Bode plot for each value of dt, and since the error in rejecting disturbances is $|\frac{1}{1+P'K}|$ and $|P'K|$ is large at the frequency of the velocity variation, $2\pi/T_w = 0.13$ Rad/sec, the disturbances are rejected effectively. At smaller dt the controller responds faster, and the gain is higher at every frequency, resulting in better performance. Controllers with smaller dt are also more robust, as seen from the gain and phase margins. This supports what we see in the experiments where the deviations from $\varphi_{C\tau} = 0$ are minimal for all four controller update frequencies. As the ratio of dt/T_w is increased, the ability of the control system to respond to the changing disturbance decreases. Still, the controller is effective in all of these cases, and thus is practical for applications that have similar levels of free-stream velocity variation.

5.6 Summary

We control bypass transition in the Blasius boundary layer using a model-based feedback controller in experiments. The choice of feedback controllers is motivated by the results in Chapter 4, in which unmodeled disturbances degrade the performance of purely feedforward controllers. The stream-wise streaks of spanwise periodic low- and high-velocity characteristic to bypass transition are created with an array of cylindrical roughness elements upstream. Further downstream we use a spanwise array of plasma actuators, arranged to generate streaks of opposite spanwise phase, to control the flow and attenuate the disturbance. Feedback measurements are taken with a spanwise array of wall-mounted shear-stress sensors, located downstream of the plasma actuators. A novel control scheme is formulated to reduce the energy in the particular spanwise wavenumber corresponding to the streaks which undergo transient growth and lead to transition. Experimental data is used to find an empirical quasi-steady input-output model of the experiments, where the inputs are the roughness elements and plasma actuators, and the output is related to measured shear stress and targeted spanwise frequency. This model is used to design and analyze a proportional-integral controller prior to implementation in the wind tunnel.

The controller is quite effective; for *on-design* conditions, the energy of the targeted mode is reduced by 94% at the stream-wise location of the feedback sensors. The total disturbance energy of the boundary layer is reduced by 74% to 86%, leaving room for improvement with more intricate actuator configurations or targeting multiple spatial frequencies. The controller is also shown to attenuate the disturbance both upstream and downstream of the measurement plane. To check the robustness of the controller, we operate it in off-design conditions by varying the free-stream velocity. The controller remains highly effective, even for slowly-time-varying disturbances, supporting its use in applications that have similarly slowly time-varying conditions.

The effectiveness of this controller demonstrates that the use of near-wall sensing, feedback, and plasma actuators are all well-suited to the control of bypass transition. The next

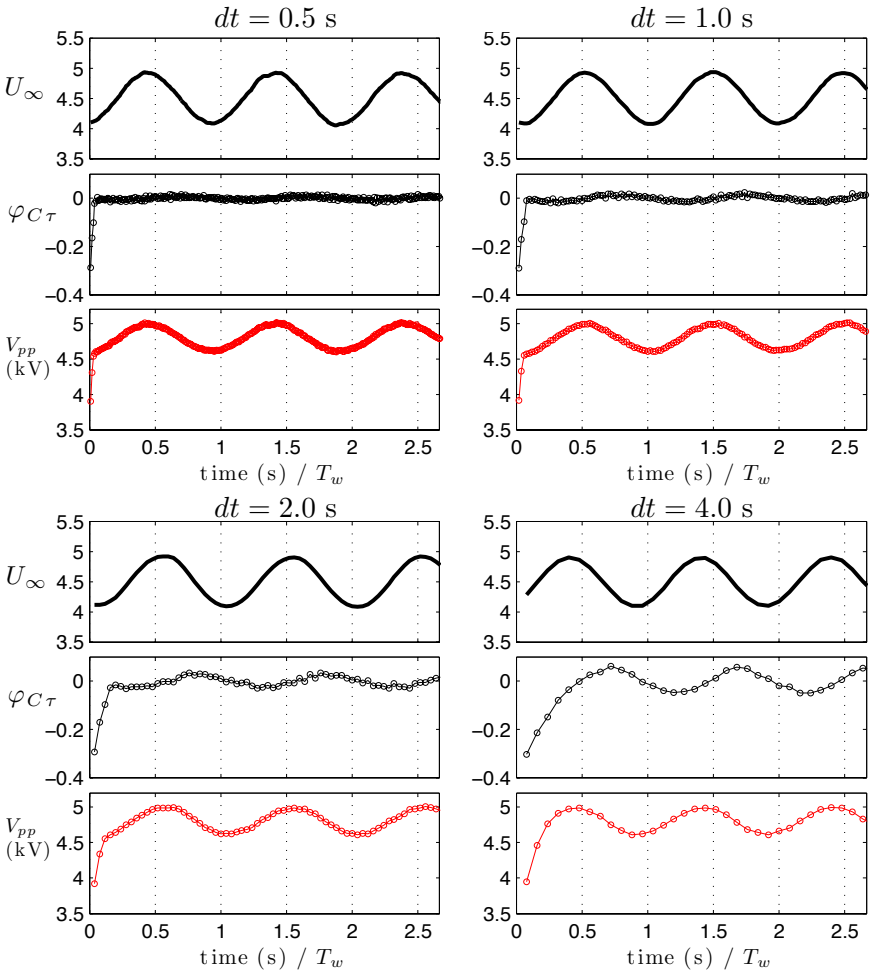


Figure 5.23: Effectiveness of control when the free-stream velocity is continuously varied between 4 and 5 m/s with $T_w = 50$ s, and the controller time step is varied as $dt = 0.5$, 1, 2, and 4 s. In each subplot, the free-stream velocity is shown at the top with the corresponding control objective $\varphi_{C\tau}$ in the middle and actuator voltage at the bottom. The roughness element height is $k = 1.25$ mm.

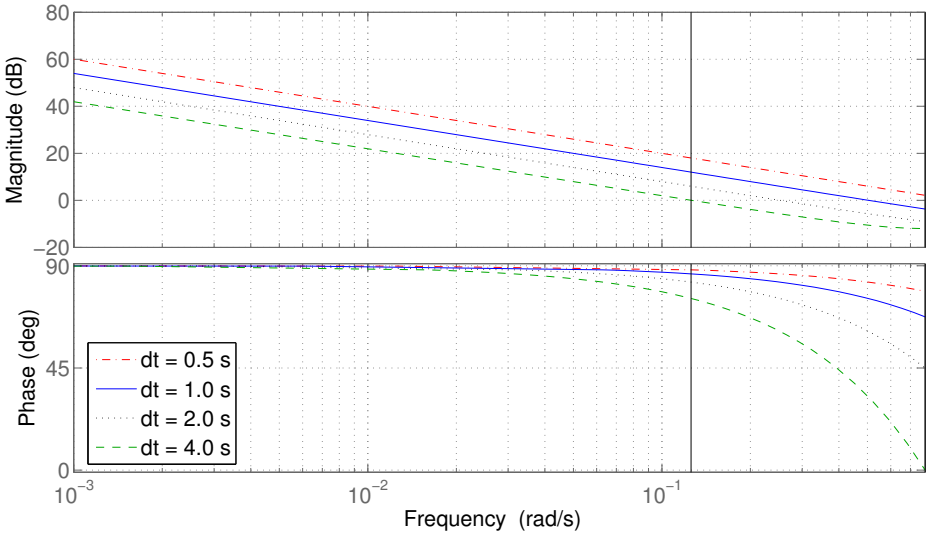


Figure 5.24: Frequency response of loop gain $P'K$. The vertical line marks the frequency at which the free stream velocity is varied, 0.13 Rad/s.

logical step is aimed at the development of low-order dynamic models of the pulsed actuator and disturbance input. This work addresses issues that are relevant to this next step, including the implementation of near-wall sensing and a suitable control objective. Another future direction is to combine feedforward and feedback control to increase the bandwidth of the controller. In addition, the optimal placement of actuators and sensors remains unknown, and is best approached with numerical simulations.

The next chapter (Chapter 6) is on progress towards such a 3D numerical study of actuator and sensor optimization. Such a numerical study requires the inclusion of plasma actuators in a fluid mechanics simulation, which is made challenging by the much smaller length and time scales inherent to the plasma actuators relative to the surrounding fluid dynamics. In Chapter 6, we develop and validate plasma body force models based on approximations to the underlying physics.

Chapter 6

Plasma actuator body force models

The previous chapter discusses feedback control using plasma actuators in experiments. The control is only quasi-steady and is based entirely on steady-state experimental measurements. The next logical step is to control the transients. Controller design is easier with simulations than with experiments where we have access to the entire velocity field rather than a limited set of measurements. Simulations also allow for an in-depth analysis of sensor and actuator arrangements that could result in better performing and more robust controllers. Varying sensors, actuators, and other parameters in experiments takes significantly more time and resources than it does in simulations. Thus, it is desirable to simulate the system for controller design.

This research finds a physical model for plasma actuators that is easily understood, computationally inexpensive, and simple to include in fluid simulations. The model approximates the force from the plasma actuators on the surrounding fluid. The main contributions of this chapter are choosing a suitable model for simulations and control studies, adapting it to our geometry, and validating that model across a range of voltages. The simulations and experiments, both in the steady-state and transients, are close to agreeing, but would likely need to be improved to design a controller in a simulation and apply it to an experiment.

In the next section (6.1), we review notable body force models. We choose a model for this work in Section 6.2, and modify the boundary conditions to suit the plasma actuator geometry we use. The resulting force is calculated and applied in simulations in Section 6.3 where we fit the tunable parameters against experimental measurements. In Section 6.4, we investigate the validity of the force models by comparing transient behavior in simulations in analogous experiments.

6.1 Introduction

Plasma actuators, introduced in the previous chapter (5), consist of two electrodes separated by a dielectric material as in Figure 6.1. Due to the intense electric field produced by an alternating voltage source (of order kV and kHz), a local region of ionized air is produced above the dielectric surface. The electrons and ions in the plasma are accelerated by the electric field, colliding with neutral particles. The net effect, over a full excitation period, is an electromechanical coupling to the fluid flow that causes a wall-jet away from the exposed

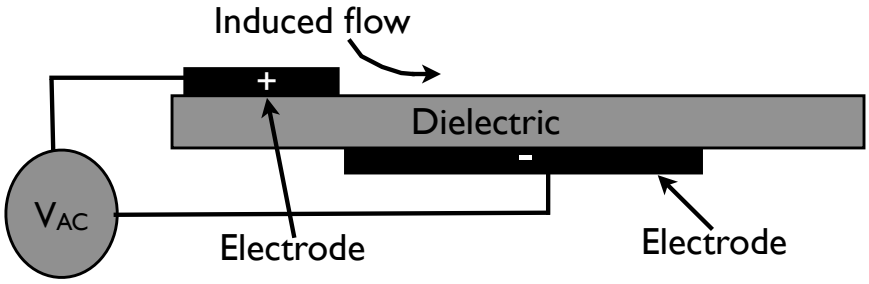


Figure 6.1: Typical plasma actuator geometry.

electrode [32, 64, 94, 95]. The surrounding fluid is subsequently forced as well. Heating also has an effect, however, at the low velocities in this research (free-stream 5 m/s), the primary effect is the force [22, 29, 31].

In this work we only use alternating, sinusoidally time-varying, voltage. Other time series, such as nano-second pulses, can produce different results, and this is an active area of research [71, 85]. Since the voltage varies sinusoidally, there are two half-cycles. In the forward-discharge half-cycle, the exposed electrode has a negative potential difference, electrons collect over the covered electrode, and there is some discrepancy in the literature on whether the force on the surrounding fluid acts primarily towards [31] or away from the exposed electrode [32, 64]. In the backward-discharge half-cycle, the force is an order of magnitude larger and directed away from the exposed electrode [31]. Since the time scale of these half-cycles is much smaller than that of the surrounding fluid, only the average is relevant for our purposes, and so the uncertainty of the forward-discharge half-cycle will be safely ignored.

There are several approaches to simulating plasma actuators. The first involves simulating them from first principles and resolving all of spatial and time scales, keeping track of individual species, ions, and recombinations. For well-designed simulations, the result is an accurate description of the forces that can lead to insights about the physics governing the plasma and the forces (for example, [71, 72, 86]). The drawback is that the spatial scales are on the order of micrometers and the time scales nano-seconds, which is far smaller and shorter than those associated with the surrounding fluid's dynamics. Conducting a plasma simulation within a fluid simulation would be computationally infeasible, and furthermore, wasteful, since the small scales do not affect the fluid and that is our primary focus.

Instead, a practical way to include the effect of plasma actuation in fluid simulations is to model the force, averaged over a full period and average out the smallest spatial variations that have no influence on the fluids. There are many such models in the literature, each with advantages and disadvantages. In [89], an effort is made to compare notable models from several different categories, and it is shown that they all give different results and require considerable tuning to give results that do not violate experimental observations.

The simplest type of model assumes a spatial force distribution and has a few adjustable parameters to tune the strength and shape so it better approximates a particular plasma actuator. A notable model in this category is proposed in [102], which prescribes a linearly-

varying spatial force distribution with its peak at the corner of the exposed electrode. Its most desirable feature is simplicity; no computations are necessary. Despite its simplicity, it is effective for the original authors' purposes. Among its drawbacks are that it does not explicitly depend on the geometry of the plasma actuator, for example, the height of the exposed electrode or length of the covered one. The multiple parameters allow for flexibility and can be fit as functions of the geometry and applied voltage, but there are many of them, making such a procedure potentially ambiguous, time-consuming, and highly dependent on the particular plasma actuator, therefore requiring many independent fits. A second is proposed in [105], and consists of an analytical approximation of the governing equations. The electric field varies inversely proportionally with distance from the corner of the exposed electrode. We find that adapting this model to the conditions we are interested in results in a spatial distribution of force that has too small an area.

Another type of model is based on approximations to the governing Maxwell's equations. After manipulating the equations, generally there are boundary value problems to solve for the electric potential (which is directly related to the electric field), and the charge density. More approximations must be made to determine the boundary conditions, and it is here that tunable parameters appear. The first such model is proposed in [108]. The boundary conditions for the electric potential are straightforward, with the exposed electrode having the value of the applied voltage. The charge density is prescribed to vary as a half-Gaussian distribution, decaying from a maximum at the corner of the exposed electrode. This gives rise to two tunable parameters: the maximum charge density and the rate at which the Gaussian decays (i.e., the Gaussian spatial distribution's width). We use this model later because it gives force fields that appear roughly physical with only two parameters that have clear physical meanings. A minor drawback, though, is that it predicts equal force in both half-cycles, something we know to contradict experiments but we also know the half-cycles are insignificant with respect to the surrounding fluid. A modification to this model is given in [56] that slightly changes the boundary conditions on the charge density. The charge density is assumed to vary sinusodially with the voltage, with a maximum value throughout the entire covered electrode. This works well for the geometry in the original work, but as we show later, does not adapt well to our slightly different actuator geometry.

Another, more complicated, model of this type involves many virtual sub-circuits running from the dielectric surface to above the exposed electrode [79, 87]. Solving a series of ODEs, one for each sub-circuit, gives boundary conditions for the electric potential on the dielectric surface. Then a similar boundary value problem to that solved in [108] is solved. The charge density is taken to be proportional to this electric potential, and the force is calculated from these two quantities. One of the advantages of this model is the original authors show that it accurately reproduces the maximum force generated is proportional to $V_{pp}^{7/2}$. The disadvantages are that there are multiple parameters related to the sub-circuits that are difficult to define and, one could argue, are further removed from the physics than in other models.

A final model of this type, given in [68], combines the previous two. The sub-circuits are used to find boundary conditions on the charge density as well as the electric potential. This replaces approximating the charge density as a Gaussian or as being constant in the covered electrode, as in [108] and [56], and replaces the approximation that the electric potential and charge density are proportional as in [79, 87]. The advantage is the charge density is found

explicitly, but the same drawbacks of the model in [79, 87] are present.

6.2 Model and adaptations to our geometry

The simplifications of the spatial distribution of the electric potential, charge density, and force fields in the models in [102, 105] result in approximations that are too inaccurate for our use. The multiple parameters in the models of [79, 87] makes them more difficult to use without expert knowledge to guide the tuning of each parameter. Therefore, we decide to use a model of the type in the works [56, 107, 108], which provide accurate approximations with only a few simple physical parameters. Both models begin with the same approximations of the governing equations, and we reproduce that here.

First, the body force is given by

$$\mathbf{f} = \rho_c \mathbf{E}. \tag{6.1}$$

Faraday's law of induction gives us

$$\nabla \times \mathbf{E} = -\mu_0 \frac{\partial \mathbf{H}}{\partial t} \approx 0, \tag{6.2}$$

where the magnetic field $\mathbf{H}$ has a very small derivative in time for this plasma application (and μ_0 is the permeability of free space). Since the curl of the gradient is zero, we know that there must exist an electric potential Φ such that

$$\mathbf{E} = -\nabla \Phi. \tag{6.3}$$

Gauss's law gives

$$\nabla \cdot (\epsilon \mathbf{E}) = \rho_c \tag{6.4}$$

where $\epsilon = \epsilon_0 \epsilon_r$ is the permittivity, ϵ_0 the permittivity of free space, and ϵ_r the relative permittivity of the material. Substituting for $\mathbf{E}$ from Equation (6.3) gives

$$\nabla \cdot (\epsilon \nabla \Phi) = -\rho_c. \tag{6.5}$$

An approximation for the charge density in terms of the electric potential (given in [28]) is

$$\rho_c / \epsilon_0 = (-1/\lambda_d^2)\Phi \tag{6.6}$$

where λ_d is the Debye length, which is the characteristic length over which a charge carrier's effect is significant. As in original works [56, 107], we approximate the Debye length as a constant $\lambda_d = 0.17$ mm, despite the fact that it varies both spatially and temporally. A physical justification for this approximation is lacking in the previous works, but the resulting model accurately reproduces the actuators' effect on the surrounding fluid, and thus the results lend support that the approximation has at least some validity for the purpose of an approximate force model. The differential equation for Φ (6.5) is linear, and so we can use superposition to split the solution into two components

$$\Phi = \phi + \psi. \tag{6.7}$$

This is useful because, as we will show, it allows us to split Equation (6.5) into two independent equations for charge density and electric potential. To arrive at these two equations, we first make an approximation. Assuming the Debye length is small and the charge density on the surface of the electrodes is small relative to the charge density in the interior, as is the case here, then we can approximate that the electric field is primarily due to the potential difference between the electrodes:

$$\mathbf{E} \approx -\nabla\phi. \tag{6.8}$$

Another result of this approximation is that the charge density in Equation (6.5) (Gauss's Law) is approximately zero, yielding

$$\nabla \cdot (\epsilon \nabla\phi) = 0. \tag{6.9}$$

Subtracting Equation (6.9) from (6.5) yields

$$\nabla \cdot (\epsilon \nabla\psi) = -\rho_c. \tag{6.10}$$

Next, we manipulate (6.6) and (6.7) into

$$\psi = -(\rho_c \lambda_d^2/\epsilon_0 + \phi) \tag{6.11}$$

and substitute into (6.10) to give

$$\nabla \cdot (\epsilon_r \nabla(-\rho_c \lambda_d^2/\epsilon_0 - \phi)) = -\rho_c/\epsilon_0 \tag{6.12}$$

which simplifies to

$$\nabla \cdot (\epsilon_r \nabla\rho_c) = \rho_c/\lambda_d^2. \tag{6.13}$$

After solving Equations (6.9) and (6.13), the force is given by

$$\mathbf{f} = -\rho_c \nabla\phi. \tag{6.14}$$

Now we consider the boundary conditions. In the original works, the plasma actuator geometry is similar to that in Figure 6.1, but we use a different geometry shown in Figure 6.2. In [56], the boundary condition for charge density is $\rho_c = \rho_c^{\max} h(t)$, where $h(t) = \sin(2\pi f t)$ and f is the frequency, on the covered electrode. When applied to our geometry, the resulting charge density spatial distribution is not physical. It has no variation in the horizontal (z) direction, but the charge density should be largest where the electric field is largest—near the corners of the exposed electrode. Instead, we use the model in [108], in which the boundary condition for the charge density is a half-Gaussian along the surface of the dielectric, defined by $G(z, \sigma) = \exp\left(-\left(\frac{z-z_0}{2\sigma}\right)^2\right)$. Adapting these boundary conditions to our geometry gives more physical results for the potential and charge density. Figure 6.2 shows the boundary conditions, and it is clear that the charge density is largest near the corners of the exposed electrode, as expected.

We solve the governing equations ((6.13) and (6.9)) subject to the boundary conditions in Figure 6.2. Numerically, we approximate the derivatives with second-order finite-differences on a uniform Cartesian grid in both the z and y dimensions. The solution is found iteratively using the Successive Over Relaxation (SOR) method, a standard variant of Jacobi iteration

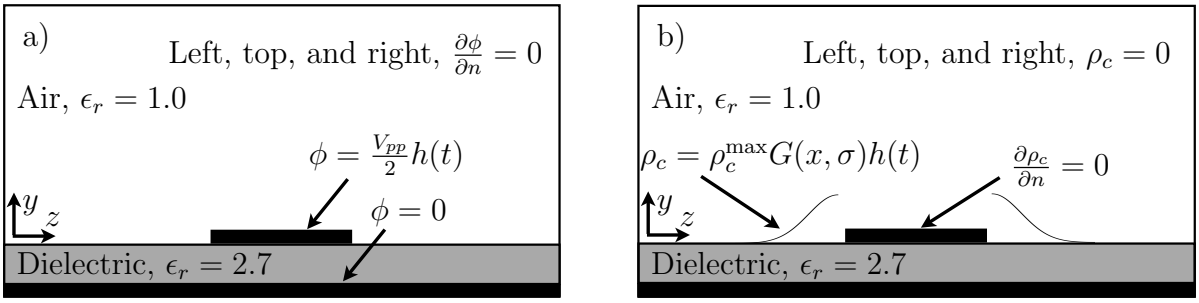


Figure 6.2: Boundary conditions for the model that we adapt for our geometry. Function $h(t) = \sin(2\pi f t)$ and $G(z, \sigma) = \exp\left(-\left(\frac{z - z_0}{2\sigma}\right)^2\right)$.

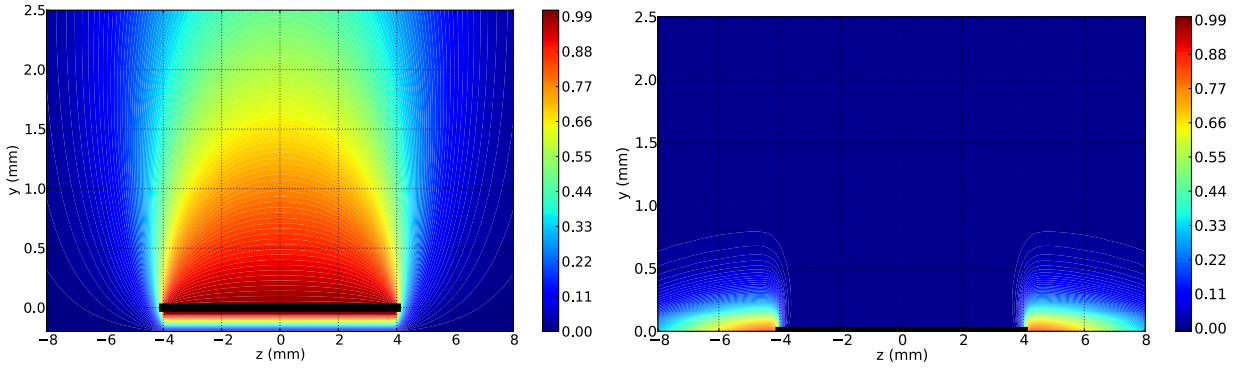


Figure 6.3: Potential (left) and charge density (right) solutions, normalized by $V_{pp}/2$ and $\rho_c^{\max}$ respectively.

with faster convergence. The accuracy is sufficient given the level of approximations already included in the model.

The resulting electric potential ϕ and charge density ρ_c solutions are shown in Figure 6.3. The charge density has its peaks around the corners of the exposed electrodes, as expected, and appears to have reasonable decay away from these peaks. The strongest electric field is also at the corners of the exposed electrode where the electrons and ions are concentrated in experiments [31]. Lastly, the forces, calculated from Equation (4.2) are shown in Figure 6.4.

The spatial distribution of the force is fixed and its magnitude is determined by the two tunable parameters—the maximum charge density $\rho_c^{\max}$ and width σ.

6.3 Fit parameters

In this section, we find the tunable parameters for the model by comparing simulations to previous experiments. In particular, we apply an AC voltage and compare measurements at a single y-z plane at a downstream location after the flow has reached equilibrium. We do not consider transients when tuning the parameters, but instead compare the transients of the models and experiments as a form of validation.

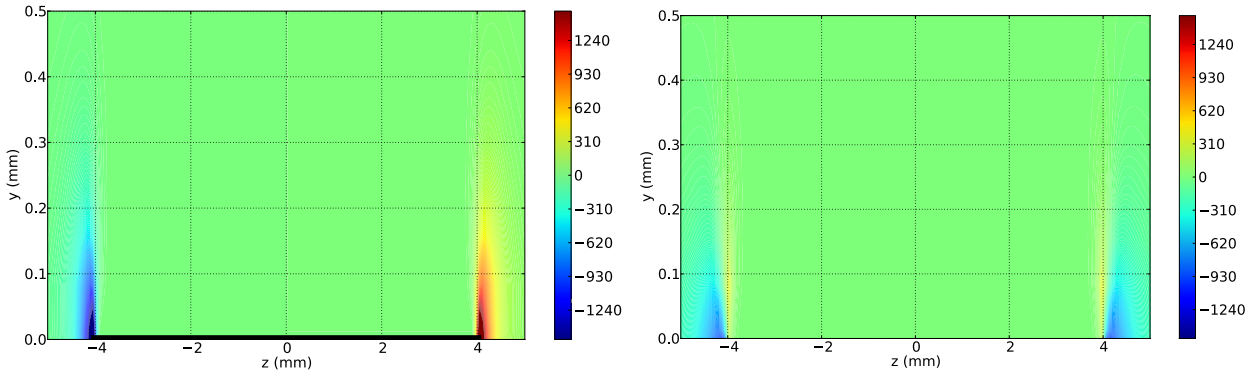


Figure 6.4: Force components, in units of $\frac{\text{Newtons}}{\rho_c^{\max} V_{pp}/2}$.

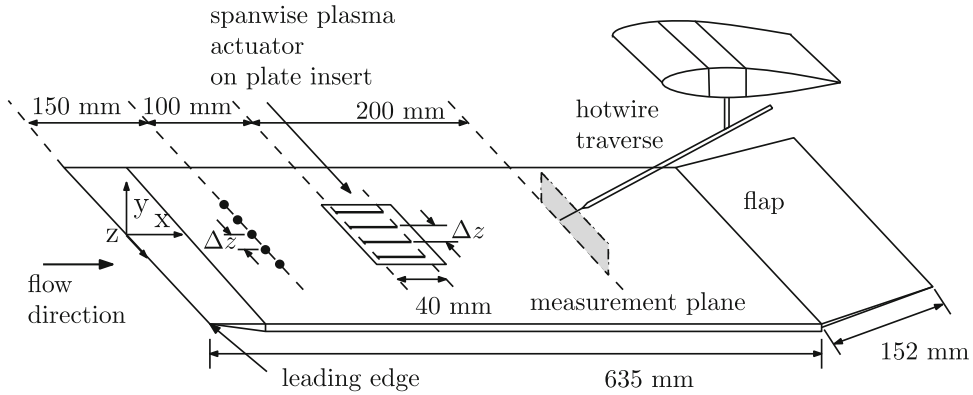


Figure 6.5: Geometry of plasma actuators and boundary layer.

The experiments are conducted by Ronald Hanson at the University of Toronto. The geometry is similar to that in Chapter 5, except the upstream roughness elements are not used—they are retracted and flush to the wall. The conditions of the computational study are based on the experimental setup used by [42] as shown in Figure 6.5. Summarizing, the plasma actuator array is located 250 mm from the geometric leading edge, but the virtual leading edge is a further 21 mm downstream (due to a pressure gradient), so, when comparing to simulations, the plasma actuator array is 229 mm downstream from the virtual leading edge. (See [110] chapter 12 for information on the use of a virtual leading edge.) The actuators extend 40 mm in the stream-wise x direction and are spaced $\Delta z = 20$ mm apart from each other. At these conditions, $\mathrm{Re}_{\delta^*} = U\delta^*/\nu = 530$, where $\delta^* = 1.59$ mm is the displacement thickness at the plasma actuators, $U = 5$ m/s is the free-stream velocity, and $\nu = 1.5 \cdot 10^{-5}$ m^2/s is the kinematic viscosity of air. Measurements of the stream-wise velocity are taken of entire y-z planes at several stream-wise locations with hot wires. The measurements are phase-averaged to reduce high-frequency variations.

The flow is simulated using the software described in Chapter 2. The computational domain in this case is chosen to start 200 mm from the (virtual) leading edge and extends 670 mm, about 190 mm beyond the last measurement plane at $x = 679$ mm. This ensures

that the fringe region is sufficiently far downstream to not interfere with the region of interest.

One of the purposes of this force model is for future use in real-time control of bypass transition. The controller would vary the input voltage, and so it is important to find a relationship between the input voltage and the two parameters $\rho_c^{\max}$ and σ. We do not vary the voltage frequency here, since it is harder to do with commonly used lab equipment, and keep it fixed at $f = 1.5$ kHz. As previously mentioned, these two parameters are dependent on the particular characteristics of the actuators in use. They are not specified by the model, and therefore they are found empirically. We fit these two parameters by comparing the steady-state stream-wise velocity produced by the plasma actuator array in experiments at a y-z plane located at $x = 479$ mm from the virtual leading edge to analogous simulation data. The parameters are iteratively adjusted based on the maximum velocity amplitude and streak width until the simulations closely match the experiments. An example of the comparison of experiments and simulations is shown in Figure 6.6. The relationship between V_{pp} and $\rho_c^{\max}$ is shown in Figure 6.7. We find that the same value of $\sigma = 2.0$ mm is appropriate for the entire range of voltages, significantly simplifying the tuning process. This may or may not be a good approximation for a wider range of voltages. It is worth noting that it may be possible and preferable to more directly measure these parameters in experiments with sensors specifically dedicated to this purpose.

The linear relationship between V_{pp} and $\rho_c^{\max}$ is not the same as in previous works. In [28], they find $V_{pp}^{7/2}$ to be proportional to the maximum induced velocity U_{induced}. The quantity U_{induced} is proportional to the maximum force, and the maximum force predicted by the model we use is proportional to $V_{pp}\rho_c^{\max}$. Therefore, the linear relationship we find in Figure 6.7 predicts a quadratic relationship between V_{pp} and U_{induced}, rather than the $7/2$ relationship in [28]. However, our experiments are at lower voltages than those in [28], and it is possible that at higher voltages the plasma actuators we use would also adhere to the $7/2$ relationship. For that to happen, the relationship in Figure 6.7 would have to change to be $V_{pp}^{5/2} \propto \rho_c^{\max}$.

Figure 6.6 shows that, qualitatively, the spatial difference between the simulations and experiments in Figure 6.6 is relatively small. The structure of the streaks is a close match with the streaks of nearly the same magnitude and spatial distribution. More quantitatively, however, Figures 6.6c and 6.6f show that there is error and it is primarily due to the spanwise misalignment of the streaks. One possible cause of this error is inaccuracies in the model. Another is inexact knowledge of where the experimental measurements are taken in the spanwise direction. This error would likely result in destabilizing feedback control if a controller developed exclusively from simulation results was applied to an experiment because the two systems react significantly differently to actuation.

A simple way to eliminate this spanwise error would be to move the plasma actuators in the spanwise direction in the simulations until the error is minimized. We do not do this because such an approximation ceases to be an analysis of the simulation of the model and the experiments, instead becoming a matter of curve-fitting. While useful for achieving results, our intention is to illustrate the accuracy of the model, and so we leave it to future practitioners to do such curve-fitting if no first-principle improvements to the model can be found.

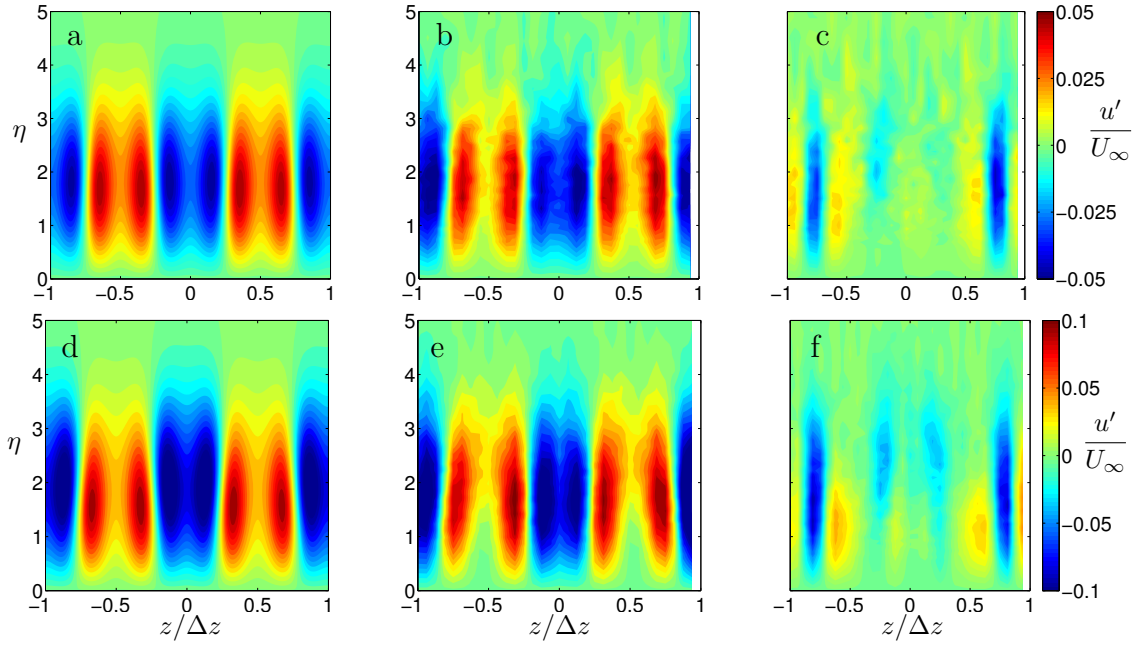


Figure 6.6: The steady-state effect of plasma actuators on a downstream plane, $x = 479$ mm, for two representative voltage levels. The top row corresponds to $V_{pp} = 4.52$ kV with a) simulation ($\rho_c^{\max} = 5.0 \cdot 10^{-4}$ C/m^3), b) experiment, and c) error, the difference between the simulation and experiment. Analogously, the bottom row corresponds to $V_{pp} = 5.52$ kV with d) simulation ($\rho_c^{\max} = 9.3 \cdot 10^{-4}$ C/m^3), e) experiment, and f) error.

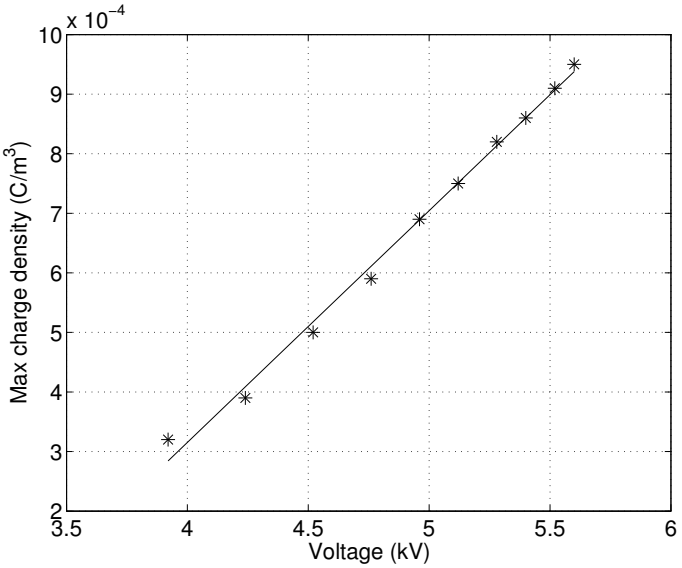


Figure 6.7: Linear fit of $\rho_c^{\max}$ as a function of V_{pp}. Each data point represents an iterative tuning of $\rho_c^{\max}$ that makes the simulation closely match the experiment, as in Figure 6.6. The fit is given by $\rho_c^{\max}[C/m^3] = 3.89 \cdot 10^{-4} V_{pp}[kV] - 1.24 \cdot 10^{-3}$.

6.4 Validation with transients

To validate the model, we compare the transient response to plasma actuation at a stream-wise location $x = 487$ mm. These comparisons are purposefully not used in finding the model and fits so that they can be used as a form of validation. The actuator array is given a sinusoidal voltage at a fixed frequency of 1.5 kHz and magnitude of 5.0 kV for 0.1 seconds, then no voltage. The results are shown in Figure 6.8, which has y-z planes of stream-wise velocity sampled in time.

The results are very similar to the steady case shown previously in Figure 6.6, and this is the best possible result. The dynamic behavior is accurately reproduced in the simulations, and only the spanwise misalignment error is significant. The high-speed fluid higher in the boundary layer causes the plasma actuation to be first measured higher in the boundary layer, and this effect is present in the simulations just as in the experiments. The first two rows of Figure 6.8 demonstrate this. The simulation and model are also accurate both when the plasma actuator is turned on and when it is turned off, which is important because the plasma actuators can have residual effects for a short period of time after the voltage is turned off. There is a small amount of error, as seen in the last row of Figure 6.8 near $z/\Delta z = 0.2$, but we do not expect it to be significant in future control design. Lastly, we anticipate that eliminating the spanwise error in the steady velocity field, as we discuss in Section 6.3, would also eliminate the spanwise error in the transients seen in Figure 6.8.

6.5 Summary

In this chapter, we begin by reviewing the different methods by which plasma actuators can be included in fluid simulations. The most appropriate choice for our purposes of controlling an incompressible boundary layer is a body force model. There are several of these models in the literature, which we discuss, and settle on the model in [108] because its results give a reasonable spatial force distribution. Further, it has only two parameters that need to be tuned, and these parameters have clear physical meanings. Thus, this model's simplicity, accuracy, and physical meaning make it an appealing choice.

The plasma actuators in the original paper do not have the same geometrical arrangement as the array of actuators we use, and therefore we modify the boundary conditions for our geometry. Additionally, the model's two parameters, the magnitude ρ_{max} and the shape of the Gaussian distribution σ of the charge density, are tuned for our particular plasma actuator. We do this by simulating the downstream effect of the plasma actuators given a sinusoidal voltage with constant magnitude. We compare the steady-state velocity with analogous measurements from experiments. Iteratively, the two parameters are varied until the simulations closely approximate the experiments.

The two parameters can, in general, vary with the voltage magnitude, and we find these relationships. The shape of the charge density boundary condition, σ, does not vary significantly with voltage, and so we approximate it as a constant. The other parameter, $\rho_c^{\max}$, varies linearly with voltage. This relationship is different than the commonly used $U_{\text{induced}} \propto V^{7/2}$ [28], however, that relationship is found empirically at higher voltages than we use. Therefore, it is possible that our results are consistent.

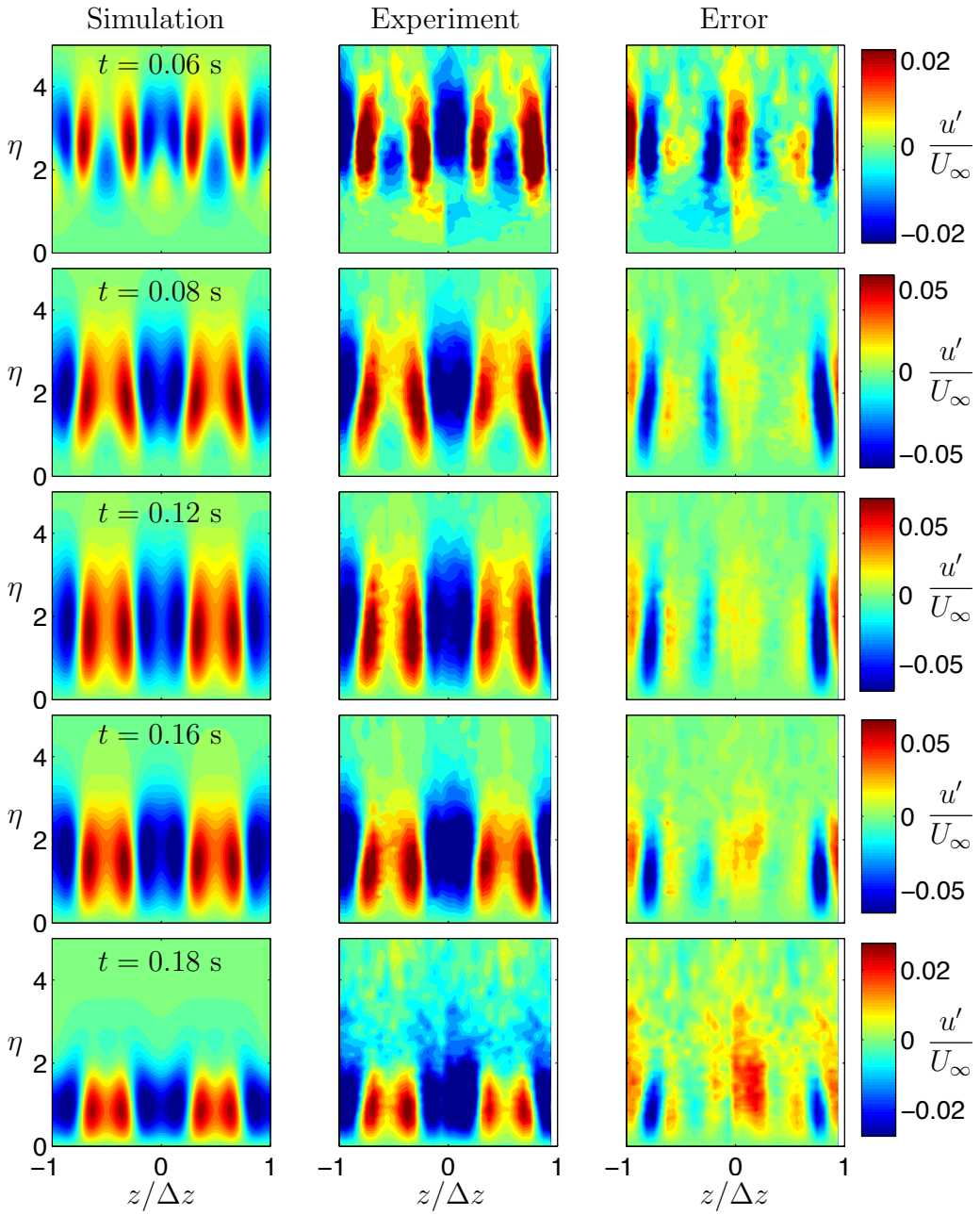


Figure 6.8: Comparison of transient response in simulations and experiments. Each row corresponds to a different time. The three columns correspond to the results from simulation (model), the experiment, and the difference (error) between the first two.

As a form of validation, we compare the transient, dynamic, behavior of the simulations of plasma actuators to experiments. The transients are important for real-time control since an effective and fast-acting controller would operate almost entirely on the transient behavior. We see that the error present in the transient velocities, a small mismatch in the spanwise direction, is also present in the steady-state velocities. This is the only significant error in the transients, further supporting the use of this model in control studies.

In the future, this model can be used as part of developing controllers from simulations, where it is simpler and cheaper to vary parameters such as the actuator and sensor positions and type of sensors. The simulations also provide more information for analyzing performance and robustness. Then, the well-designed controllers can be applied to experiments.

Chapter 7

Conclusions and future work

7.1 Conclusions

This thesis advances the modeling and control of transitional boundary layers using both simulations and experiments. We make four main contributions: a model reduction library, an analysis of the role of actuators and sensors in delaying classical transition, canceling the growth of streaks in a wind tunnel, and simulating plasma actuators' action on the surrounding fluid as a body force.

The first main contribution is the model reduction Python library `modred` that implements several important modal decompositions and system identification algorithms, including POD, BPOD, DMD, Galerkin projection for LTI systems, OKID, and ERA. Previously, practitioners wrote their own implementations that were tightly coupled to their dataset and were often not rigorously tested. Our library is designed to work with a wide range of datasets and is unit tested to improve reliability.

The library has different interfaces and implementations that are most appropriate for different sized data. For smaller data there is a simple Matlab-like interface—generally a single function call is all that is required. For datasets that are too large be loaded into memory simultaneously, there is an object-oriented interface that allows users to provide functions that interact with their specific data. For this case, `modred` is parallelized for distributed memory architectures with MPI. Other features include online documentation, easy-to-use interfaces, and extendability to new algorithms.

The second contribution is an analysis of the role of actuators and sensors and their positions in the control of TS waves in the 2D Blasius boundary layer. We show that sensors upstream of the actuators constitute a feedforward configuration, and having the sensors downstream of the actuators is a feedback configuration. Previous work uses feedforward configurations, but we seek feedback configurations since they have the potential to better reject unknown disturbances. A few combinations of different types of actuators, sensors, and control design techniques are tried for feedback configurations. One sensor poorly measures the TS waves after they are deformed by the actuation, and so is ill-suited for this control problem. Other combinations are found to have right-half-plane zeros that fundamentally limit a controller's robustness and performance. By choosing an actuator that directly and immediately affects the disturbance TS waves and a sensor which measures the TS waves, we

find a controller with good performance and robustness. The previous feedforward controller is ineffective in the presence of an unmodeled disturbance, but the new feedback controller with different actuators and sensors is quite effective, reducing the control objective by 75%.

The third contribution is quasi-steady control of bypass transition in a wind tunnel. Incoming flow is disturbed by roughness elements emanating from the wall, creating stream-wise streaks of high and low stream-wise velocity that are inherent to bypass transition. Further downstream plasma actuators create stream-wise streaks of opposite phase, thus attenuating their growth. Wall-mounted shear stress sensors measure the effectiveness of the plasma actuators and their signals are fed to a controller to determine the level of actuation. The controller updates slower than the convective time scales, and so the control is quasi-steady. To design the controller, we first empirically model the relationship from the inputs (the roughness elements and plasma actuators) to the outputs (shear stress measurements) using experimental data. Careful consideration is paid to the choice of output to facilitate the control design and to target the specific span-wise wavenumber of the stream-wise streaks. The nonlinearity in the original input-output model is inverted, resulting in a linear input-output model that is easier to use in control design. We design a simple PI controller that performs well, both in on-design flow conditions and off-design flow conditions such as sinusoidally varying free-stream velocity.

The last main contribution is in modeling plasma actuators for fluid simulations. The motivation is that it is beneficial to try different types of sensors and positions and to analyze the effectiveness of corresponding controllers, and this iterative process is easier in simulations. However, plasma actuators operate at smaller length and time scales than the surrounding fluid, making direct simulation computationally prohibitive and wasteful. Instead, we use a physics-based model that averages over the time and space, yielding forces on scales that are appropriate for the fluid dynamics. The body force model we use has two parameters that depend on the particular plasma actuator setup (geometry, materials, etc.), and the driving voltage. We empirically fit these parameters as a function of voltage magnitude using the steady results measured in the wind tunnel. Then we validate that the transients in the simulations with those in the experiments. The transients match reasonably well, and so this model could be a good choice for future numerical flow control studies.

7.2 Future work

- **Eliminate dataset size limitations in `modred`.** The model reduction library can only be used when individual data elements (vectors) are small enough that at least three can fit in a single node's memory simultaneously. Eliminating this limitation would allow `modred` to operate on the very high-dimensional datasets that stand to benefit the most from model reduction. Such an improvement is possible by generalizing the so-called MPI workers in `modred` to not be individual processors, but MPI communicators (groups of processors). User-supplied functions would be called by one MPI communicator and have access to all of its processors. Thus, users would write functions that utilize all of the communicators processors. Careful analysis of the scaling of speedup versus processors is necessary before such changes are made.

- **Actuator and sensor choice for high-dimensional dynamical systems.** To find an actuator-sensor pair that effectively and robustly delayed the growth of TS waves in Chapter 4, we utilize knowledge of the physics and control theory. Even with this knowledge, it takes some experimenting with different types of actuators and sensors before we find a good combination. Progress on simplifying and generalizing the process of choosing actuators, sensors, and their positions for large systems would be useful in *many* applications.

- **Perform time-resolved control on bypass transition in experiments.** Extending the quasi-steady plasma-actuator control from Chapter 5 to controlling transients would result in high performance in the presence of time-varying disturbances. To capture the transient input-output behavior, a different approach to finding empirical input-output models is needed. One approach is to use OKID and ERA, and both are available in `modred`.

- **Develop controllers for bypass transition with simulations.** Simulations of the experimental setup would allow for a relatively cheap simulation to guide control strategies to implement in experiments. Unfortunately, simulating the roughness elements as they rise and fall is difficult (varying boundary conditions) and could require a numerical approach that is more flexible than the pseudo-spectral approach of SIMSON. We consider an alternative pressure-correction approach on a collocated grid with compact finite difference spatial derivatives in the stream-wise and wall-normal directions. Some progress towards this is made but the work is incomplete and not included.

Bibliography

[1] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[2] S. Ahuja and C. W. Rowley. Feedback control of unstable steady states of flow past a flat plate using reduced-order estimators. *J. Fluid Mech.*, 645:447–478, 2010.

[3] E. Åkervik, J. Hœpffner, U. Ehrenstein, and D. S. Henningson. Optimal growth, model reduction and control in a separated boundary-layer flow using global eigenmodes. *J. Fluid Mech.*, 579:305–314, 2007.

[4] B. D. O. Anderson and Y. Liu. Controller reduction: concepts and approaches. *IEEE T. Automat. Contr.*, 34(8):802–812, August 1989.

[5] N. Aubry, P. Holmes, J. L. Lumley, and E. Stone. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *J. Fluid Mech.*, 192:115–173, 1988.

[6] M. F. A. Azeez and A. F. Vakakis. Proper Orthogonal Decomposition (POD) of a class of vibroimpact oscillations. *J. Sound Vib.*, 240(5):859–889, 2000.

[7] S. Bagheri, L. Brandt, and D. S. Henningson. Input-output analysis, model reduction and control of the flat-plate boundary layer. *J. Fluid Mech.*, 620:263–298, 2009.

[8] S. Bagheri and D. S. Henningson. Transition delay using control theory. *Philos. T. Roy. Soc. A*, 369:1365–1381, 2011.

[9] B. A. Belson, R. E. Hanson, D. Palmeiro, P. Lavoie, K. Meidell, and C. W. Rowley. Comparison of plasma actuators in simulations and experiments for control of bypass transition. AIAA Paper 2012-1141, 50th AIAA Aerospace Sciences Meeting and Exhibit, January 2012.

[10] B. A. Belson, O. Semeraro, C. W. Rowley, and D. S. Henningson. Feedback control of instabilities in the 2d Blasius boundary layer: the role of sensors and actuators. *Phys. Fluids*, 25(5):054106, 2013.

[11] B. A. Belson, J. H. Tu, and C. W. Rowley. A parallelized model reduction library [submitted]. *ACM T. Math. Software*, 2013.

[12] N. Bénard, J. Jolibois, E. Moreau, R. Sosa, G. Artana, and G. Touchard. Aerodynamic plasma actuators: A directional micro-jet device. *Thin Solid Films*, 516(19):6660–6667, 8 2008.

[13] F. P. Bertolotti, Th. Herbert, and P. R. Spalart. Linear and nonlinear stability of the Blasius boundary layer. *J. Fluid Mech.*, 242:441–474, 1992.

[14] T. R. Bewley and S. Liu. Optimal and robust control and estimation of linear paths to transition. *J. Fluid Mech.*, 365:305–349, 1998.

[15] T. R. Bewley, R. Temam, and M. Ziane. A general framework for robust control in fluid mechanics. *Physica D*, 138:360–392, 2000.

[16] L. Brandt, D. Sipp, J. O. Pralits, and O. Marquet. Effect of base-flow variation in noise amplifiers: the flat-plate boundary layer. *J. Fluid Mech.*, 687:503–528, November 2011.

[17] S. L. Brunton, C. W. Rowley, and D. R. Williams. Linear unsteady aerodynamic models from wind tunnel measurements. AIAA Paper 2011-3581, 41st AIAA Fluid Dynamics Conference and Exhibit, June 2011.

[18] K. K. Chen and C. W. Rowley. H2 optimal actuator and sensor placement in the linearised complex Ginzburg-Landau system. *J. Fluid Mech.*, 681:241–260, June 2011.

[19] K. K. Chen, J. H. Tu, and C. W. Rowley. Variants of Dynamic Mode Decomposition: Boundary condition, Koopman, and Fourier analyses. *J. Nonlinear Sci.*, 22(6):887–915, March 2012.

[20] M. Chevalier, P. Schlatter, A. Lundbladh, and D. S. Henningson. Simson: A pseudo-spectral solver for incompressible boundary layer flows. Technical report, KTH, 2007.

[21] H. Choi, P. Moin, and J. Kim. Active turbulence control for reduction in wall-bounded flows. *J. Fluid Mech.*, 262:75–110, 1994.

[22] K.-S. Choi, T. Jukes, and R. Whalley. Turbulent boundary-layer control with plasma actuators. *P. R. Soc. Lond. A Mat.*, 369:1443–1458, April 2011.

[23] T. C. Corke, C. L. Enloe, and S. P. Wilkinson. Dielectric barrier discharge plasma actuators for flow control. *Annual Review of Fluid Mechanics*, 42:505–529, 2010.

[24] T. C. Corke, M. L. Post, and D. M. Orlov. Single dielectric barrier discharge plasma enhanced aerodynamics: physics, modeling and applications. *Exp. Fluids*, 46:1–26, 2009.

[25] G. Dergham, D. Sipp, and J.-C. Robinet. Accurate low dimensional models for deterministic fluid systems driven by uncertain forcing. *Phys. Fluids*, 23:094101, 2011.

[26] J. C. Doyle. Guaranteed margins for LQG regulators. *IEEE T. Automat. Contr.*, 23(4):756–757, August 1978.

[27] G. E. Dullerud and F. Paganini. *A Course in Robust Control Theory*, volume 36 of *Texts in Applied Mathematics*. Springer, 2000.

[28] M. Engert, A. Pätzold, R. Becker, and W. Nitsche. Active cancellation of Tollmien-Schlichting instabilities in compressible flows using closed-loop control. IUTAM Symposium on flow control and MEMS, 2008.

[29] C. L. Enloe, T. E. McLaughlin, R. D. VanDyken, K. D. Kachner, E. J. Jumper, T. C. Corke, M. Post, and O. Haddad. Mechanisms and responses of a single dielectric barrier plasma actuator: Geometric effects. *AIAA Journal*, 42(3):595–604, 2004.

[30] Z. Falkenstein and J. Coogan. Microdischarge behaviour in the silent discharge of nitrogen-oxygen and water-air mixtures. *J. Phys. D Appl. Phys.*, 30:817–825, 1997.

[31] B. F. Farrell and P. J. Ioannou. Accurate low-dimensional approximation of the linear dynamics of fluid flow. *J. Atmos. Sci.*, 58:2771–2789, 2001.

[32] G. I. Font. Boundary-layer control with atmospheric plasma discharges. *AIAA J.*, 44(7):1572–1578, 2006.

[33] M. Forte, J. Joliboism, F. Moreau, G. Touchard, and M. Cazalens. Optimization of a dielectric barrier discharge actuator by stationary and non-stationary measurements of the induced flow velocity: application to flow control. AIAA Paper 2006-2863, 3rd AIAA Flow Control Conference, 2006.

[34] J. H. M. Fransson and L. Brandt. Experimental and theoretical investigation of the nonmodal growth of steady streaks in a flat plate boundary layer. *Phys. Fluids*, 16(10):3627–3638, October 2004.

[35] E. A. Gillies. Low-dimensional control of the circular cylinder wake. *J. Fluid Mech.*, 371:157–178, 1998.

[36] S. Grundmann and C. Tropea. Experimental transition delay using glow-discharge plasma actuators. *Exp. Fluids*, 42(4), 2007.

[37] E. P. Hammond, T. R. Bewley, and P. Moin. Observed mechanisms for turbulence attenuation and enhancement in opposition-controlled wall-bounded flows. *Phys. Fluids*, 10(9):2421–2423, 1998.

[38] A. Hanifi, P. J. Schmid, and D. S. Henningson. Transient growth in compressible boundary layer flow. *Phys. Fluids*, 8(3):826–837, 1996.

[39] R. E. Hanson, K. M. Bade, B. A. Belson, P. Lavoie, A. M. Naguib, and C. W. Rowley. Feedback control of slowly-varying transient growth by an array of plasma actuators [submitted]. *Phys. Fluids*, 2013.

[40] R. E. Hanson, H. P. Buckley, and P. Lavoie. Aerodynamic optimization of the flat-plate leading edge for experimental studies of laminar and transitional boundary layers. *Exp. Fluids*, 53:863–871, June 2012.

[41] R. E. Hanson, P. Lavoie, K. M. Bade, and A. M. Naguib. Steady-state closed-loop control of bypass boundary layer transition using plasma actuators. AIAA Paper 2012-1140, 50th AIAA Aerospace Sciences Meeting and Exhibit, 2012.

[42] R. E. Hanson, P. Lavoie, and A. M. Naguib. Effect of plasma actuator excitation for controlling bypass transition in boundary layers. AIAA Paper 2010-1091, 48th AIAA Aerospace Sciences Meeting and Exhibit, January 2010.

[43] R. E. Hanson, P. Lavoie, A. M. Naguib, and J. F. Morrison. Transient growth instability cancellation by a plasma actuator array. *Exp. Fluids*, 49(6):1339–1348, 2010.

[44] B. L. Ho and R. E. Kalman. Effective construction of linear state-variable models from input/output data. Proc. Third ann. Allerton Conf. on Circuit and System Th., 1965.

[45] M. Högberg, T. R. Bewley, and D. S. Henningson. Linear feedback control and estimation of transition in plane channel flow. *J. Fluid Mech.*, 481:149–175, 2003.

[46] M. Högberg and D. S. Henningson. Linear optimal control applied to instabilities in spatially developing boundary layers. *J. Fluid Mech.*, 470:151–179, November 2002.

[47] P. Holmes, J. L. Lumley, and G. Berkooz. *Turbulence, coherent structures, dynamical systems, and symmetry*. Cambridge Univ. Press, Cambridge, UK, 1996.

[48] J. R. Holton. *An introduction to dynamic meteorology, Volume 1*. Academic Press Inc, Burlington, MA, 4th edition, 2004.

[49] N. Houser, L. Gimeno, R. E. Hanson, T. Goldhawk, T. Simpson, and P. Lavoie. Microfabrication of dielectric barrier discharge plasma actuators. *Submitted to the Journal of Sensors and Actuators A*, 2013.

[50] K. H. Hsieh, M. W. Halling, and P. J. Barr. Overview of vibrational structural health monitoring with respresentative case studies. *J. Bridge. Eng.*, 11(6):707–715, 2006.

[51] P Huerre and P A Monkewitz. Local and global instabilities in spatially developing flows. *Annu. Rev. Fluid Mech.*, 22:473–537, 1990.

[52] L. S. Hultgren and D. E. Ashpis. Demonstration of separation delay with glow-discharge plasma actuators. AIAA Paper 2003-1025, 41st AIAA Aerospace Sciences Meeting and Exhibit, January 2003.

[53] M. Ilak, S. Bagheri, L. Brandt, C. W. Rowley, and D. S. Henningson. Model reduction of the nonlinear complex Ginzburg-Landau equation. *SIAM J. Appl. Dyn. Sys.*, 9(4):1284–1302, 2010.

[54] M. Ilak and C. W. Rowley. Modeling of transitional channel flow using balanced proper orthogonal decomposition. *Phys. Fluids*, 20:034103, March 2008.

[55] S. A. Jacobson and W. C. Reynolds. Active control of streamwise votices and streaks in boundary layers. *J. Fluid Mech.*, 360:179–211, 1998.

[56] S. S. Joshi, J. L. Speyer, and J. Kim. A system theory approach to the feedback stabilization of infinitesimal and finite-amplitude disturbances in plane Poiseuille flow. *J. Fluid Mech.*, 332:157–184, 1997.

[57] D. A. Reasor Jr., R. P. LeBeau Jr., and Y. B. Suzen. Unstructured grid simulations of plasma actuator models. AIAA Paper 2007-3973, 37th AIAA Fluid Dynamics Conference and Exhibit, June 2007.

[58] J.-N. Juang. *Applied System Identification*. Prentice Hall, Upper Saddle River, NJ, 1994.

[59] J.-N. Juang and R. S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *J. Guid. Control Dynam.*, 8(5):620–627, 1985.

[60] J.-N. Juang, M. Phan, L. G. Horta, and R. W. Longman. Identification of observer/Kalman filter Markov parameters: theory and experiments. NASA Tech. Mem. 104069, June 1991.

[61] T. Jukes and K. Choi. Dielectric-barrier-discharge vortex generators: characterisation and optimisation for flow separation control. *Exp. Fluids*, 52:329–345, 2012.

[62] K. Karhunen. Zur spektraltheorie stochasticher prozesse. *Ann. Acad. Sci. Fennicae*, 34, 1946.

[63] J. Kim and T. R. Bewley. A linear systems approach to flow control. *Annu. Rev. Fluid Mech.*, 39:383–417, 2007.

[64] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully-developed channel flow at low Reynolds-number. *J. Fluid Mech.*, 177:133–166, April 1987.

[65] W. Kim, H. Do, M. Mungal, and M. Cappelli. On the role of oxygen in dielectric barrier discharge actuation of aerodynamic flows. *Applied Physics Letters*, 2007.

[66] A. V. Kozlov and F. O. Thomas. Bluff-body flow control via two types of dielectric barrier discharge plasma actuation. *AIAA Journal*, 49:1919–1931, September 2011.

[67] P. Lavoie, A. Naguib, and J. Morrison. Transient growth induced by surface roughness in a blasius boundary layer. In *Proceedings of ICTAM*. ICTAM, Adelaide, Australia, August 2008.

[68] K. H. Lee, L. Cortelezzi, J. Kim, and J. Speyer. Application of reduced-order controller to turbulent flows for drag reduction. *Phys. Fluids*, 13(5):1321–1330, 2001.

[69] S. Lemire and H. D. Vo. Reduction of fan and compressor wake defect using plasma actuation for tonal noise reduction. *Journal of Turbomachinery*, 133, 2011.

[70] Y. Li and M. Gaster. Active control of boundary-layer instabilities. *J. Fluid Mech.*, 550:185–205, 2006.

[71] Y. C. Liang, W. Z. Lin, H. P. Lee, S. P. Lim, K. H. Lee, and H. Sun. Proper Orthogonal Decomposition and its applications – part 2: model reduction for MEMS dynamical analysis. *J. Sound Vib.*, 256(3):515–532, 2002.

[72] A. V. Likhanskii, M. N. Shneider, S. O. Macheret, and R. B. Miles. Modeling of dielectric barrier discharge plasma actuators driven by repetitive nanosecond pulses. *Phys. Plasmas*, 14, 2007.

[73] A. V. Likhanskii, M. N. Shneider, S. O. Macheret, and R. B. Miles. Modeling of dielectric barrier discharge plasma actuator in air. *J. Appl. Phys.*, 103, 2008.

[74] M. Loève. *Probability Theory*. Springer-Verlag, New York, NY, 1955.

[75] N. R. Losse, R. King, M. Zengl, U. Rist, and B. R. Noack. Control of Tollmien-Schlichting instabilities by finite distributed wall actuation. *Theor. Comp. Fluid Dyn.*, 25:167–178, 2011.

[76] J. L. Lumley. The structure of inhomogeneous turbulence. In A. M. Yaglom and V. I. Tatarski, editors, *Atmospheric Turbulence and Wave Propogation*, pages 166–178. Nauka, Moscow, 1967.

[77] F. Lundell, A. Monokrousos, and L. Brandt. Feedback control of boundary layer bypass transition: experimental and numerical progress. AIAA Paper 2009-612, 47th AIAA Aerospace Sciences Meeting and Exhibit, January 2009.

[78] R. Lundell. Reactive control of transition induced by free-stream turbulence: an experimental demonstration. *J. Fluid Mech.*, 585:41–71, 2007.

[79] Z. Ma, S. Ahuja, and C. W. Rowley. Reduced order models for control of fluids using the Eigensystem Realization Algorithm. *Theor. Comp. Fluid Dyn.*, 25(1):233–247, June 2011.

[80] B. Mertz and T. C. Corke. Time-dependent dielectric barrier discharge plasma actuator modeling. AIAA Paper 2009-1083, 47th AIAA Aerospace Sciences Meeting and Exhibit, January 2009.

[81] I. Mezič. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlin. Dynam.*, 41:309–325, 2005.

[82] R. J. Moffat. Describing the uncertainties in experimental results. *Exp. Therm. Fluid Sci.*, 1:3–17, January 1988.

[83] B. C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE T. Automat. Contr.*, 26(1):17–32, February 1981.

[84] E Moreau. Airflow control by non-thermal plasma actuators. *J. Phys. D Appl. Phys.*, 40:605–636, February 2007.

[85] A. M. Naguib, J. F. Morrison, and T. A. Zaki. On the relationship between the wall-shear-stress and transient-growth disturbances in a laminar boundary layer. *Phys. Fluids*, 22, 2010.

[86] D. F. Opaits, M. N. Shneider, and R. B. Miles. Electrodynamic effects in nanosecond-pulse-sustained long dielectric-barrier-discharge plasma actuators. *Appl. Phys. Lett.*, 94(6), 2009.

[87] D. F. Opaits, M. N. Shneider, R. B. Miles, A. V. Likhanskii, and S. O. Macheret. Surface charge in dielectric barrier discharge plasma actuators. *Phys. Plasmas*, 15, 2008.

[88] D. M. Orlov. *Modelling and simulation of single dielectric barrier discharge plasma actuators*. PhD thesis, Notre Dame, 2006.

[89] L. Osmokrovic. Receptivity of laminar boundary layers to spanwise-periodic forcing by an array of plasma actuators. Masters, University of Toronto, 2012.

[90] D. Palmeiro and P. Lavoie. Comparitive analysis on single-dielectric-barrier-discharge plasma actuator models. Proc. 7th Symposium on Turbulence and Shear Flow Phenomena, June 2011.

[91] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philos. Mag.*, 2(6):559–572, 1901.

[92] P. Peterson. F2PY: a tool for connecting Fortran and Python programs. *Int. J. Comp. Sci.*, 4(4):296–305, 2009.

[93] R. Peyret. *Spectral Methods for Incompressible Viscous Flow*, volume 148. Springer, 2002.

[94] R. Rathnasingham and K. S. Breuer. Active control of turbulent boundary layers. *J. Fluid Mech.*, 495:209–233, November 2003.

[95] J. R. Roth and D. M. Sherman. Boundary layer flow control with a one atmosphere uniform glow discharge surface plasma. AIAA Paper 98-0328, AIAA Aerospace Sciences Meeting and Exhibit, 1998.

[96] J. R. Roth, D. M. Sherman, and S. P. Wilkinson. Electrohydrodynamic flow control with a glow-discharge surface plasma. *AIAA J.*, 38(7):1166–1172, 2000.

[97] C. W. Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. *Int. J. Bifurcat. Chaos*, 15(3):997–1013, 2005.

[98] C. W. Rowley, I. Mezič, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. *J. Fluid Mech.*, 641:115–127, December 2009.

[99] W. S. Saric. Boundary-layer stability and transition. In C. Tropea, A. Yarin, and J. Foss, editors, *Springer Handbook of Experimental Fluid Mechanics*, chapter 12.3, pages 886–896. Springer, Berlin-Heidelberg, 2007.

[100] D. M. Schatzman and F. O. Thomas. Turbulent boundary layer separation control with plasma actuators. AIAA Paper 2008-4199, 4th Flow Control Conference, June 2008.

[101] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.*, 656:5–28, August 10 2010.

[102] O. Semeraro, S. Bagheri, L. Brandt, and D. S. Henningson. Feedback control of three-dimensional optimal disturbances using reduced-order models. *J. Fluid Mech.*, 677:63–102, 2011.

[103] W. Shyy, B. Jayaraman, and A. Andersson. Modeling of glow discharge-induced fluid dynamics. *Journal of Applied Physics*, 92(11):6434–6443, 2002.

[104] L. Sirovich. Turbulence and the dynamics of coherent structures, parts I–III. *Q. Appl. Math.*, 3:561–590, 1987.

[105] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. John Wiley and Sons, 2nd edition, 2005.

[106] V. R. Soloviev and V. M. Krivstov. Phenomenological model of the body force induced by surface dielectric barrier discharge. AIAA Paper 2011-157, 49th AIAA Aerospace Sciences Meeting and Exhibit, January 2011.

[107] D. Sturzebecher and W. Nitsche. Active cancellation of Tollmien-Schlichting waves instabilities on a wing using multi-channel sensor actuator systems. *Int. J. Heat Fluid Fl.*, 24(4):572–583, 2003.

[108] Y. B. Suzen, P. G. Huang, and D. E. Ashpis. Numerical simulations of flow separation control in low-pressure turbines using plasma actuators. AIAA Paper 2007-937, 45th AIAA Aerospace Sciences Meeting and Exhibit, January 2007.

[109] Y. B. Suzen, P. G. Huang, J. D. Jacob, and D. E. Ashpis. Numerical simulations of plasma based flow control applications. AIAA Paper 2005-4633, 35th AIAA Fluid Dynamics Conference and Exhibit, 2005.

[110] J. R. Taylor. *Introduction to Error Analysis*. University Science Books, 2nd edition, 1997.

[111] C. Tropea, A. L. Yarin, and J. F. Foss. *Springer Handbook of Experimental Fluid Mechanics*. Springer, Berlin, 2007.

[112] J. H. Tu and C. W. Rowley. An improved algorithm for balanced POD through an analytic treatment of impulse response tails. *J. Comput. Phys.*, 231(16):5317–5333, 2012.

[113] J. A. C. Weideman and S. C. Reddy. A MATLAB differentiation matrix suite. *ACM T. Math. Software*, 26(4):465–519, December 2000.

[114] E. B. White. Transient growth of stationary disturbances in a flat plate boundary layer. *Phys. Fluids*, 14:4429–4439, December 2002.

[115] E. B. White, J. M. Rice, and F. G. Ergin. Receptivity of stationary transient disturbances to surface roughness. *Phys. Fluids*, 17(6), 2005.

[116] F. M. White. *Viscous Fluid Flow*. McGraw-Hill, 2006.

[117] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, 1996.

Appendix A

DMD algorithm derivation

The work in this appendix is a joint effort with Jonathan Tu. There are a number of algorithms for computing DMD modes, each an improvement on its predecessor. While the companion-matrix formulation provides a clear analogy to Koopman spectral analysis [97], the SVD-based algorithm given in [100] is more computationally stable. A low-memory variant of the latter algorithm requires as few as two vectors in memory at any given time, making it suitable for large data [111]. It also eliminates a large number of unnecessary inner products by reusing elements of the correlation matrix in later computations (about a 50% reduction). Here, we improve on the low-memory algorithm by identifying and eliminating additional redundancies, avoiding costly inner products and linear combinations of snapshots. Specifically, whenever possible we replace operations involving snapshots with equivalent operations using small matrices. For large data, this is much more efficient, as the small matrices can be stored in memory and manipulated easily, whereas the snapshots require careful memory management.

The original low-memory DMD algorithm is summarized below.

1. Collect and store snapshots $\mathbf{x}_i$, for $i = 1 \ldots, m_{\mathbf{x}}$.

2. Compute each entry of the correlation matrix via $[\mathbf{H}]_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, using all but the last snapshot (i and j have range $1, \ldots, m_{\mathbf{x}} - 1$).

3. Compute the eigenvalues and eigenvectors of $\mathbf{H}$, writing $\mathbf{HU} = \mathbf{U\Sigma}$, where $\mathbf{\Sigma}$ is diagonal and real, and $\mathbf{U}$ is orthogonal (or unitary) since $\mathbf{H}$ is symmetric (or Hermitian). Sort the eigenvalues (and corresponding eigenvectors) in descending order.

4. Construct the POD modes individually via $\boldsymbol{\psi}_j = \sum_{i=1}^{m_{\mathbf{x}}} \mathbf{x}_i \, [\mathbf{U\Sigma}^{-1/2}]_{i,j}$.

5. Define the sub-matrix $\mathbf{H}' = [\mathbf{H}]_{1:m_{\mathbf{x}}-1,2:m_{\mathbf{x}}-1}$ (i.e., $\mathbf{H}$ with the first column removed), making use of the previously computed correlation matrix.

6. Compute each entry of $\mathbf{H}''$ via $[\mathbf{H}'']_i = \langle \mathbf{x}_i, \mathbf{x}_{m_{\mathbf{x}}} \rangle$.

7. Compute $\mathbf{M} = \mathbf{\Sigma}^{-1/2}\mathbf{U}^* \begin{bmatrix} \mathbf{H}' & \mathbf{H}'' \end{bmatrix} \mathbf{U\Sigma}^{-1/2}$ and solve the eigenvalue problem $\mathbf{MV} = \mathbf{\Lambda V}$.

8. Construct intermediate (unscaled) modes individually via $\tilde{\boldsymbol{\varphi}}_j = \sum_{i=1}^{m_{\mathbf{x}}} \boldsymbol{\psi}_i [\mathbf{V}]_{i,j}$.

9. Compute the elements of $\mathbf{d}'$ via $[\mathbf{d}']_i = \langle \tilde{\boldsymbol{\varphi}}_i, \mathbf{x}_1 \rangle$.

10. Compute the vector $\mathbf{d} = (\mathbf{V}^*\mathbf{V})^{-1}\mathbf{d}'$.

11. Scale the DMD modes: $\boldsymbol{\varphi}_i = [\mathbf{d}]_i \tilde{\boldsymbol{\varphi}}_i$.

(Note that we have changed from the notation used in [111] to better match that used in this work.)

In the above algorithm, there are five steps (2, 4, 8, 9, 11) that require manipulations of snapshots (or equivalently large data). Many of these computations can be combined or avoided completely. To see this, we first note that the "intermediate" modes in step 9 can be written in terms of the original vectors as

$$\begin{aligned}
\tilde{\boldsymbol{\varphi}}_j &= \sum_{i=1}^{m_{\mathbf{x}}} \boldsymbol{\psi}_i [\mathbf{V}]_{i,j} \\
&= \sum_{i=1}^{m_{\mathbf{x}}} \sum_{k=1}^{m_{\mathbf{x}}} \mathbf{x}_k \, [\mathbf{U}\boldsymbol{\Sigma}^{-1/2}]_{k,i} [\mathbf{V}]_{i,j} \\
&= \sum_{k=1}^{m_{\mathbf{x}}} \mathbf{x}_k \sum_{i=1}^{m_{\mathbf{x}}} [\mathbf{U}\boldsymbol{\Sigma}^{-1/2}]_{k,i} [\mathbf{V}]_{i,j} \\
&= \sum_{k=1}^{m_{\mathbf{x}}} \mathbf{x}_k [\mathbf{U}\boldsymbol{\Sigma}^{-1/2}\mathbf{V}]_{k,j}.
\end{aligned} \tag{A.1}$$

We achieve a savings by moving $\mathbf{x}_k$ outside of one of the sums; to compute this linear combination, the snapshots only have to be loaded and summed once. Previously, the algorithm required two independent linear combinations (steps 4 and 8).

We achieve further savings by eliminating unnecessary inner products from the computation of $\mathbf{d}'$. Substituting for $\tilde{\boldsymbol{\varphi}}_i$, we find

$$\begin{aligned}
[\mathbf{d}']_i &= \langle \tilde{\boldsymbol{\varphi}}_i, \mathbf{x}_1 \rangle \\
&= \left\langle \sum_{j=1}^{m_{\mathbf{x}}} \mathbf{x}_j [\mathbf{U}\boldsymbol{\Sigma}^{-1/2}\mathbf{V}]_{j,i}, \, \mathbf{x}_1 \right\rangle \\
&= \sum_{j=1}^{m_{\mathbf{x}}} [\mathbf{U}\boldsymbol{\Sigma}^{-1/2}\mathbf{V}]_{j,i} \langle \mathbf{x}_j, \mathbf{x}_1 \rangle.
\end{aligned}$$

The inner products $\langle \mathbf{x}_j, \mathbf{x}_1 \rangle$ are simply elements of $\mathbf{H}$, which has already been computed in step 2. This eliminates the m inner products required in step 9. As such, we can write

$$\mathbf{d}' = \mathbf{V}^*\boldsymbol{\Sigma}^{-1/2}\mathbf{U}^*[\mathbf{H}]_{1:m_{\mathbf{x}}-1,1}, \tag{A.2}$$

which is a product of small matrices. We can then compute $\mathbf{d}$ as

$$\begin{aligned}
\mathbf{d} &= (\mathbf{V}^*\mathbf{V})^{-1}\mathbf{d}' \\
&= (\mathbf{V}^*\mathbf{V})^{-1}\mathbf{V}^*\boldsymbol{\Sigma}^{-1/2}\mathbf{U}^*[\mathbf{H}]_{1:m_{\mathbf{x}}-1,1},
\end{aligned} \tag{A.3}$$

again using only previously computed matrices.

Finally, combining the above results, we can write

$$\boldsymbol{\varphi}_j = \sum_{i=1}^{m_{\mathbf{x}}} \mathbf{x}_i \left[\mathbf{T}_{\mathbf{x}}\right]_{i,j}, \tag{A.4}$$

where

$$\mathbf{T}_{\mathbf{x}} = \mathbf{U}\boldsymbol{\Sigma}^{-1/2}\mathbf{V}\mathbf{D}, \tag{A.5}$$

and $\mathbf{D} = \mathrm{diag}(\mathbf{d})$, a diagonal matrix with main diagonal $\mathbf{d}$. Thus we can compute the scaled DMD modes using a single linear combination operation; the original low-memory algorithm required computing intermediate modes, computing a scaling, and then scaling the modes. In summary, this improved formulation reduces the total number of linear combinations by half and the total number of inner products by m. (We note that because the algorithm requires $O(m^2)$ total inner products, the latter is a relatively minor savings.)

Appendix B

Efficient calculation of many ERA models

We make two slight modifications to the procedure from Section 4.3.2 for increased accuracy and computational efficiency. The first accounts for the different amplitudes of the output signals. The magnitude of output y is much smaller than that of z, and so ERA tends to weight the observability space in favor of z, resulting in the truncation of states that improve the accuracy of y, but y is vital for the controller. To remedy this, we normalize all of the outputs before applying ERA, then adjust the resulting model. Specifically, we collect the Markov parameters, then divide all y_i by their maximum, $\mathsf{y}_{\max}$ and all z_i by the single maximum of all ten signals, $z_{\max}$. We then perform the rest of the ERA procedure, which provides the reduced-order model matrices specified in Equation 3.29. Then $\mathbf{C}_{\mathsf{y},r} \mapsto \mathbf{C}_{\mathsf{y},r} \cdot \mathsf{y}_{\max}$ and $\mathbf{C}_{\mathsf{z},r} \mapsto \mathbf{C}_{\mathsf{z},r} \cdot z_{\max}$.

The second modification increases the computational efficiency of calculating of many ERA reduced-order models. Each model depends on $\mathbf{B}$ and $\mathbf{C}$, so each sensor location requires a new model. However, in this work the inputs are mostly kept fixed and the position of the outputs is varied, i.e. we primarily change $\mathbf{C}_{\mathsf{y}}$. Instead of simulating an impulse response and collecting the series of Markov parameters $\mathbf{C}\mathbf{A}^i\mathbf{B}$ for each choice of $\mathbf{C}_{\mathsf{y}}$, we collect the full snapshots $\mathbf{A}^i\mathbf{B}$. Then, for each choice of $\mathbf{C}_{\mathsf{y}}$, we compute $\mathbf{C}\mathbf{A}^i\mathbf{B}$ as post-processing. This means we do only two impulse responses for each $\mathbf{B}$ (one per input) rather than two impulses per sensor location. The same series of snapshots, $\mathbf{A}^i\mathbf{B}$, is also used when finding the POD modes.