Control of Underactuated Fluid-Body Systems with Real-Time Particle Image Velocimetry

by

John W. Roberts, B.S., M.S.

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

Author .....
John W. Roberts, B.S., M.S.
PhD Candidate
May 18, 2012

Certified by .....
Russ Tedrake
X Consortium Associate Professor - Thesis Supervisor

Certified by .....
Annette Peko Hosoi
Associate Professor - Thesis Committee Chair

Accepted by .....
David E. Hardt
Chairman, Committee on Graduate Students

Control of Underactuated Fluid-Body Systems with Real-Time Particle Image Velocimetry

by

John W. Roberts, B.S., M.S.

Submitted to the Department of Mechanical Engineering
on May 18, 2012 in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Controlling the interaction of a robot with a fluid, particularly when the desired behavior is intimately related to the dynamics of the fluid, is a difficult and important problem. High-performance aircraft cannot ignore nonlinear stall effects, and robots hoping to fly and swim with performance matching that seen in birds and fish cannot treat fluid flows as quasi-steady. If we wish to match the level of performance seen in nature several major hurdles must be overcome, with one of the most difficult being the poor observability of the fluid state. Fluid dynamicists have long contended with this observability problem, and have used computationally intensive Particle Image Velocimetry (PIV) to gain an understanding of the fluid behavior after the fact. However, improvement in available computational power is now making it possible to perform PIV in real-time. When PIV provides real-time awareness of the fluid state it is no longer just an analysis tool, but rather a valuable sensor that can be integrated into the control loop.

In this thesis I present methods for controlling fluid-body systems in which the fluid plays a vital dynamical role, for performing real-time PIV, and for interpreting the output of PIV in a manner useful to control. The utility of these methods is demonstrated on a mechanically simple but dynamically rich experimental platform: the hydrodynamic cart-pole. This system is analogous to the well-known cart-pole system in the controls literature, but through its relationship with the surrounding fluid it captures many of the fundamental challenges of general fluid-body control tasks, including: nonlinearity, underactuation, an important and unknown fluid state and a dearth of accurate and tractable models. The first complete demonstration of closed-loop PIV control is performed on this system, and there is a statistically significant improvement in the system's ability to reject fluid disturbances when using real-time PIV for closed-loop control. These results suggest that these new techniques will push the boundaries of what we can expect a robot in a fluid to do.

Thesis Supervisor: Russ Tedrake
Title: X Consortium Associate Professor

Acknowledgments

I owe a great many people thanks for the support and and assistance I've received while the research in this thesis took shape. First, I'd like to thank my advisor, Professor Russ Tedrake, for all the guidance, mentorship and opportunities he has offered me over the past six years. The spectrum of ideas I have encountered while working with him has been broad and deep, and he has brought people together into a group of which I am proud to be a member. I have learned a great deal while working with Russ, both in the theoretical and applied realms, and I will always be grateful for time I've had in the Robot Locomotion Group.

Second, I'd like to thank my committee: Professor Domitilla Del Vecchio, Professor Doug Hart and Professor Peko Hosoi. They have all given me new perspectives on my work, and pointed me in fruitful directions for future research. I owe particular thanks to Professor Hart for loaning me a pulsed laser - without it some of the most exciting experiments could not have been performed.

Third, I want to thank my labmates. They've always been there to carry on fruitful discussions about research, offer assistance in the many fields in which they are expert, help carry out experiments, and get coffee. They have provided invaluable intellectual distraction at times, and I've learned a lot from them.

Fourth, I wish to thank my friends in Course 2. They've made my time in Cambridge more rewarding, memorable and fun than I could have ever hoped. And Wayne Staats in particular has offered commiseration during midnight thesis-writing, Clover breaks at mid-day to get me out of the lab, and countless true 16oz pints of assorted dips at his home.

Finally, I want to thank my family for their support. When I make it home I'm glad the Christmas cookie baking process has been executed on an industrial scale, and I can always count on someone picking up the phone. Thanks for everything.

Contents

Chapter 1

Introduction

Controlling the interaction of a body with a fluid – as in done by birds, fish, aircraft and myriad other systems – is a problem of great importance and subtle complexity. Birds' ability to fly through forests, maneuvering between dense obstacles at high speeds, is unmatched by current man-made technology. Similarly, experiments on fish in the lab have demonstrated the degree to their bodies are tuned to swim; not simply through their streamlined shape, but through the dynamical relationship between the mass and compliance of the fish and the forces present in the surrounding fluid [10, 52, 50].

Obtaining useful dynamical models of these systems is, however, extremely difficult. Leaving aside the particular challenges of modeling the internal dynamics of a living creature such as a fish, the behavior of Navier-Stokes when coupled to equations of motion for an immersed body is tremendously complex. Even the most advanced computational methods struggle to reproduce the dynamics of an experimental system. For some special cases such as rigid airfoils at quasi-steady operation good methods do exist [86, 92], but for many important fluid-body systems, such as small Unmanned Aerial Vehicles (UAVs) executing high-performance maneuvers, starting from first-principles produces unsatisfactory returns.

Furthermore, when interacting with a complex flow environment another problem

arises: the importance of a fluid state which can be very difficult to measure using only local sensors such as pressure taps and anemometers. If flow features such as vortices are on the same scales as the body to be controlled (e.g., neither small enough to be ignored as discrete objects nor large enough to be treated as a more global change) they can have complicated effects which are very difficult to model effectively. Contending with this sensing problem is non-trivial, and in many cases it can be difficult to determine what measurements are appropriate to make informative, and what aspects of the flow are important.

The literature has treated problems of this class with two different approaches: high-performance control of UAVs with an experimental focus that neglects fluid disturbances, or controlling fluid systems with a computational (i.e., CFD) focus that concentrates on the fluid and does not include coupling with a body. A detailed review of the work on controlling UAVs can be found below in §1.2.1, but in brief it has the following characteristics: good first-principles models coupled with strong actuators, and, in some cases, expert demonstrations of desired trajectories and inputs, allow for impressive performance on actual experimental systems. This work includes Abbeel's apprenticeship learning for an aerobatic helicopter [1], Mellinger's minimum snap trajectories for quadrotors [62] and Lupashin and D'Andrea's work on quadrotor flips [58]. On windy days, however, recorded expert maneuvers can become infeasible, and first-principles models can be completely inadequate in gusty conditions.

Alternatively, there is work on directly controlling fluid systems that consider complicated, unsteady fluid effects, but it does not generally deal with the additional difficulty of fluid-body coupling. This literature considers problems such as cavity flows [20], the vortex wake of a cylinder [12], the mixing behavior of a fluid jet [97] and the improvement of lift-to-drag ratios for stalled or near-stalled wings [72, 90]. A detailed review may again be found below in §1.2.2, but in brief this work is frequently computational in nature (i.e., not validated on an experiment), often considers only open-loop controllers [34] and the control objectives are purely functions of the flow

(e.g., suppress the instability of a cavity flow [9]). While the work in this literature does deal with directly controlling complex fluid behaviors, it does not consider the effects arising from the coupling between the fluid and a body being controlled. For example, when studying the stalled wing [72] assumes the wing is traveling in a quasi-steady domain irrespective of what the actuators do. As a result, it does not offer easy solutions to the fluid-body control problems considered here.

This thesis is concerned with designing a relatively general methodology for controlling fluid-body systems at the intersection of these two different approaches. The goal is to control an underactuated, experimental fluid-body system in the presence of intermittent fluid disturbances capable of drastically impacting the body. This can be broken up into two basic tasks:

- Control a complex, underactuated dynamical system arising from the coupled dynamics of a body and its surrounding fluid.
- Allow the body to respond to significant, intermittent fluid disturbances on the scale of the body itself.

The methodology presented here to deal with these difficulties is predicated on a very basic assumption that holds largely true despite these systems' complexity: a given interaction is relatively repeatable. For example, when attempting a certain maneuver, or when repeatedly encountering a vortex of a certain strength, the system will respond with a "sufficiently small" variation in outputs. In this context "sufficiently small" means that a *local* model can be found that is relevant within the range of the variation. This is often much easier than finding a model relevant over the full domain of operation of the system, and is more amenable to models based on general-purpose model classes[1]. This assumption is often particularly true in laboratory fluid dynamics experiments as great effort is taken to ensure repeatability in experiments,

[1]In this thesis local models were all of time-varying linear form, with free parameters motivated by a basic physical understanding of the system. Much richer and more specialized model classes could also be used, but this thesis focuses on this easily transferable model class.

even to the point of waiting several hours between experiments to allow the fluid to become quiescent[42].

Using these local models a desired trajectory for the system can be found, along with an associated feedback controller to stabilize it. This new trajectory will, if the necessary procedures outline in the subsequent chapters are followed, perform better (e.g., more precise response; better disturbance rejection) than the initial system behavior. More data can then be collected, another local model can be found, and the procedure can be repeated again until convergence.

This procedure can the be coupled with with information obtained from real-time Particle Image Velocimetry, or PIV. PIV is a sort of "motion capture for fluids" capable of providing the full velocity field of a fluid flow. Recent advances in computation have allowed it to be performed in real-time, making accurate flow fields available during run time. With this information, relevant flow disturbances can be identified and responded to even if they have not yet significantly affected the controlled body. By "seeing into the flow future of the flow" superior performance can be achieved, and even complex flow disturbances can be handled in a systematic manner. If the ultimate application is such that PIV cannot be easily deployed (e.g., a UAV flying through a forest), the information obtained by developing PIV-capable controllers can act as a form of "scaffolding," telling the control designer what fluid states are important, and what degree of knowledge of the flow is necessary to attain the benchmark level of performance achieved in the fully observable PIV case.

The efficacy of this methodology is demonstrated on a simple but dynamically rich experimental test-bed: the Hydrodynamic Cart-Pole, or HCP. The HCP is described in detail in Chap. 2, but it can be thought of as a fluid analog of a the well-known cart-pole system. The difference between the classical cart-pole and the HCP is that instead of operating against gravity in the vertical plane, the HCP operates against a fluid flow in the horizontal plane (see Fig. 1-1). The fluid attempts to make the HCP point downstream, but by moving the cart the pole can be balanced while pointing

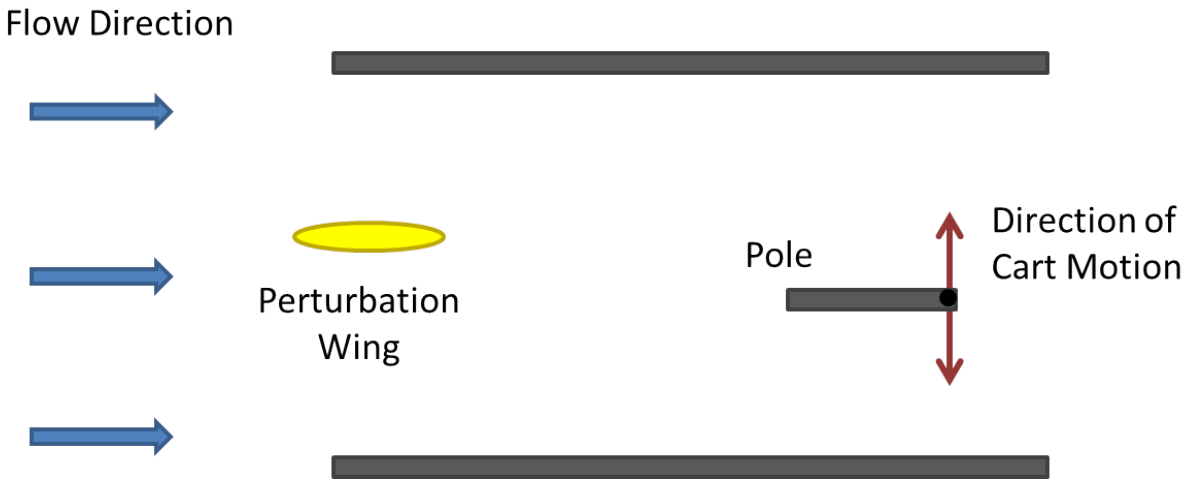


Figure 1-1: Schematic of the HCP system when looking from above (i.e,. gravity is into the page). The fluid flows from left to right while the pole (which is free to rotate) can be moved up and down by the cart. This motion of the cart is the only input to the system. The pole as shown is at the upright configuration which is passively unstable. Due to weathercock stability the pole will naturally spin 180° and point downstream if the cart is not moved to stabilize it. The yellow perturbation wing can be rotated to generate vortices which travel downstream and perturb the cart, if desired. See Chap. 2 for a full discussion of the system.

upstream. To simulate fluid disturbances a "perturbation wing" is placed upstream from the HCP. By moving this perturbation wing vortices can be shed downstream to disturb the HCP. Rejecting these disturbances effectively with the aid of PIV is the ultimate goal of this thesis, and is described in Chap. 5.

1.1 Organization of this Thesis

The remainder of thesis is organized around a means of designing controllers in the presence of these two classes of difficulties by iterative data-driven methods and real-time PIV. The chapters are as follows:

1. **Introduction -** Introduces the high-level problems and the philosophy of the solution. Describes the structure of the thesis. Presents an overview of the

relevant literature (with further citations within the text when appropriate).

2. **The Hydrodynamic Cart-Pole -** Describes the Hydrodynamic Cart-Pole (HCP); the experimental system on which the control design methodologies presented in this thesis were studied and validated. The hardware is presented and a basic, quasi-steady model of the system is developed.

3. **Control without PIV -** State observer design and basic control tasks are performed. The limitation of traditional, model-based feed-forward and feedback controllers in high-performance settings is demonstrated. A general-purpose algorithm to improve performance without building a more accurate global model is presented and its utility tested experimentally on the HCP.

4. **Real-Time PIV -** The PIV algorithm is briefly explained, and the manner in which it was implemented at real-time speeds is discussed. Filtering methods that extract relevant, low-order information from the vector fields produced by PIV are presented and their performance tested on real flow data.

5. **PIV-Enabled Control -** Real-time PIV and the iterative local-model based control design method are integrated on the HCP to reject flow disturbances. The improvement in performance over techniques without knowledge of the flow state is demonstrated experimentally.

6. **Conclusion -** The results presented in the thesis are summarized, and promising future directions are briefly outlined.

A. Implementation Details - An appendix containing details of some of the work performed in the thesis that is not central to the primary argument.

B. Robust Learning Control - An appendix containing information on a different technique that could be used to control fluid-body systems that is promising, but more dependent on an initially accurate model than the method presented

in the thesis proper. It is a relevant alternative approach, but is distinct from the methods used herein.

1.2 Previous Work

The work in this thesis builds upon the extensive literature of Iterative Learning Control and Adaptive Control (see §1.2.1) as well as work in real-time PIV processing on FPGAs and flow control (see §1.2.2). The following two sections review the relevant literature, with a focus on its relationship to the work performed in this thesis.

1.2.1 Iterative Controller Learning

Engineers attempting to design controllers for fluid-body systems frequently struggle with the difficulty of constructing a first-principles model that is accurate enough for high-performance control. "Data-driven" approaches in which either a highly-phenomenological model is built from experimental data, or in which the control policy is directly updated without ever constructing an explicit model, are an effective means of circumventing the modeling problem, and it is with these techniques that the work in this thesis has the most in common.

One of the common algorithms for improving a trajectory over time via experiment is Iterative Learning Control (ILC). ILC has a long history, and in its standard form can very effectively compensate for repeatable, structured disturbances [18]. In Bristow's definition (after Moore [63]), ILC modifies the control input (a signal) rather than the controller (a dynamical system). In this way it is distinct from the control design procedure outlined in this thesis, but the philosophy of it is very similar. Practically, the goal of ILC is to develop an open-loop input that rejects repeating disturbances while ignoring non-repeating ones. A very common (and perhaps the most common) method of performing this are so-called "PD" methods, with a form much like a PD controller [105]. These update the input based on a combination

of the measured error and the derivative of the measured error, and must be tuned by hand. This can be very difficult for a nonlinear system, particularly when it is underactuated and even the sign of the appropriate input update may be unknown.

Alternatively, methods based on inverting a model can be used, and may result in very fast convergence, though for nonlinear plants inversion can be difficult. This difficulty can be dealt with by linearizing the model around the desired behavior, and has been successful in several applications [30]. This, however, presents several difficulties for underactuated systems where the appropriate compensatory signal is difficult to find, and thus the desired behavior (and the point about which to linearize) is unclear. Furthermore, it depends upon possessing a model of the system which may be unavailable or inaccurate.

There is also relevant work in the literature on adaptive control, where both a controller and a feed-forward trajectory (not just an input) are learned from data. The most relevant examples of adaptive control of fluid-body systems take place on quadrotors, which are high-performance UAVs for which reasonably rich dynamical models are available [40], though even these models are not accurate enough to design high-performance controllers without adaptation. Control methods making use of these dynamical models have achieved impressive results [58, 61, 60, 62], but it is the goal of this thesis to not depend on an initial, relatively-accurate model.

Abbeel et al. used an adaptation method that built local models of the system near the desired behavior, but it depends upon having an initial valid, nominal trajectory to follow that is generated by having a human execute the specified task first [1, 2, 3]. Because of this dependence upon an having a "teacher" capable of performing the task at the outset, the technique is termed "apprenticeship" or "imitation" learning. For the work in this thesis I will not assume that there is a human expert who knows how to execute the desired tasks. Instead, I will assume that there is no a priori information about how attainable the specified goals are in practice, or what feasible trajectory may achieve them.

Lupashin and D'Andrea used a method in which certain "keyframes" were specified, and it was left to the quadrotor to discover a trajectory that satisfied them [59, 57]. This is a very similar formulation to that used in this thesis, but they used a simple, first-principles model to solve for the updated high-gain controller, and constrained the controller to be of a form they heuristically chose as reasonable. Because the model was available, relatively quick and robust performance could be achieved without much data from the actual system. This thesis will not assume that an accurate model exists, nor will it make use of a specialized policy parameterization designed for the task at hand. However, in the case where the model and controller parameterization are known, the method of Lupashin and D'Andrea will in general be much more data efficient than the technique outlined here.

A further set of methods for designing effective trajectories and controllers comes from the policy learning literature in Reinforcement Learning (RL) [109, 96, 45]. These techniques do not require a model, are robust to noise in the system's operation and can easily encode user intuition in the structure of the policy (as Lupashin and D'Andrea did in their method). RL techniques for learning effective trajectories on fluid-body systems have seen success in experiments, and were the basis of my previous work [78, 80, 79]. Using them to design not just trajectories but feedback controllers is an exciting possibility, and while the literature includes the possibility of learning feedback controllers in the theory, in practice it is infrequently seen. I have worked on bridging this gap with an attempt to overcome some of the difficulties in learning feedback controllers[2] by learning controllers represented in the Youla parameterization, and the direction is promising [77]. It is, however, not yet validated on an physical experimental fluid system, but can be read about in some detail in §B.

Effectively, the iterative control design method proposed in this thesis fits a certain niche within the spectrum of methods already present in the literature. It requires no model, no initial valid trajectory, and no cleverly-designed policy parameterization.

[2]Namely, the loss of stability during learning and the many local minima present in gain space.

It is not designed to follow a given trajectory (as is the case with ILC methods and apprenticeship learning), or achieve a certain state at a specified keyframe (as is the case with [57]), but rather to achieve certain performance goals such as minimizing the maximum deviation in the presence of a structured disturbance. In the general-purpose, model-starved domain studied in this thesis it is a natural fit, but in other domains some of the other techniques outlined here are more appropriate.

1.2.2 PIV and Flow-Modeling for Control

Particle Image Velocimetry (PIV; see §4.1 for a description of the technique) has been used for decades to study fluid flows in an offline setting [110, 35, 4]. Generally, despite the development of high-performance correlation techniques [36], PIV has been too computationally intensive to do at the same timescale as an experimental fluid flow. Continuing advances in computational power, however, is making it possible to perform at least basic versions of the PIV algorithm in an online setting, in which the resulting vector fields are available nearly simultaneously with the flow they represent [11].

The possibility of using this real-time PIV information in a feedback control loop has been discussed for almost a decade, with a very basic implementation in [93] in which six vectors were obtained with a delay on the order of 70ms, but no control was actually performed. In [114], a FPGA was used to obtain many more vectors still at real-time speeds, but again no control experiment was performed. In [108] it was used in a low-Reynolds number flow to control the flow speed (by controlling a pump) over a fixed airfoil in an oil tunnel, but did not actively drive the wing. This last experiment is the most similar work to that studied in this thesis, but as only the bulk flow was controlled, and not the body within the fluid, the comparisons are only superficial.

A complementary literature to that of real-time PIV is the closed-loop control of fluid flows. While this thesis concerns itself with controlling a fluid-body system,

and the flow-control literature generally concerns itself with controlling only a fluid without a coupled body, the evolution and structure of the fluid state during a fluid disturbance is critically relevant to the performance of the system. Because of this it is important to review this flow control work, even though it is not directly applicable to the coupled fluid-body control problem considered here. Note also that this review focuses on *closed-loop* flow control in which control inputs are a function of measurements of the fluid state. There is an extensive literature on active flow control that runs open loop, including work on improving the mixing in jets by actuating tiny flaps around the jet exit [97], increasing the lift-to-drag ratios of airfoils when the are nearly stalled [90] or stalled [72], and regulating leading-edge vortices on wings [34][3]. This work on open-loop control is interesting, and can be very effective at improving fluid-body systems' performance, but is not designed to make use of the novel sensing capabilities offered by real-time PIV. Thus, the remainder of this section will focus on closed-loop flow control.

1.2.2.1 Closed-Loop Flow Control

The full range of relevant fluid dynamics around the HCP are too complex to be captured by simple lumped parameter models (see Chap. 2). Thus, it is natural to consider models built on measurements of the fluid. The most prevalent method in the literature for constructing fluid models from data is the combination of Proper Orthogonal Decomposition, or POD [21] and the Galerkin projection [85].

POD is a statistical method for finding the "most important" directions in a set of multidimensional data. It does this by finding an orthonormal basis for the collected data that is aligned with the directions of greatest variation. Thus, if the data lies on a low-dimensional hyperplane POD can extract that plane, and allow the dimension of the system to be reduced while maintaining the important characteristics (i.e., the location of the data on the hyperplane). It is, in effect, an efficient means of performing

[3]Gursul also mentions a closed loop controller for leading edge vortices, but the majority of controllers in the review run open loop.

manifold learning for linear manifolds. For a system that is low-dimensional in this "linear" sense, it can be very effective at capturing the relevant low-order structure while discarding the rest of the dimensions, improving efficiency and clarity.

Once POD has been performed and relevant features have been found, a Galerkin model of the flow can be constructed in the reduced-space of the POD modes. Galerkin models for fluid systems are the result of projecting the Navier-Stokes equations onto the subspace spanned by a reduced-order model (e.g., the POD modes) to determine how the reduced states will evolve over time. A great deal of work has gone into getting the most useful features from POD for generating useful Galerkin models, as well as how to make the resulting Galkerin models capable of being used for control [98, 99, 100]. This technique is used extensively in fluid modeling and control applications to build reduced order descriptions of a fluid's evolution, particularly for finite-time or periodic flows [67]. A more data-driven approach is the "calibrated" Galerkin model [22], in which the some of the parameters which would normally come from the projection of Navier-Stokes are fit based upon collected data.

Assuming a reasonable model of the fluid has now been obtained, the problem of control can be considered. It can be very difficult to include actuation in models based on data from unactuated configurations [67], but there have been several successful applications. One such problem, and a very common one in the literature of closed-loop flow control, is the regulation of a cavity flow [20]. Examples of control in of a cavity flow include minimizing the recirculation via blowing air on the boundary [74, 75] (studied using CFD), suppressing the instability via upstream blowing [9][4] (also using CFD) and minimizing the energy in the cavity tones by using a speaker as an actuator [87] (studied on an experimental system).

Another control problem commonly encountered is the control of the wake of a circular cylinder. The modeling of this wake with POD is well-established, even in

[4]This work uses Balanced POD which is inspired by balanced truncation but less computationally expensive, and can result in fewer modes and better performance than plain POD. See [83] for more details.

complex transient regimes, and feedback controllers have been designed to successfully suppress the onset of the wake [94]. Other work on cylinder wake suppression has been done in CFD as well, including [28] and [66]. Drag reduction for the cylinder wake by periodic rotation of the cylinder has also been studied using modern POD techniques, with significant wake suppression achieved on a simulated cylinder in the laminar regime [12]. However, note that this controller was not a feedback controller (i.e., an open-loop rotation as used). There are numerous other examples of control of the wakes of cylinders in this laminar regime [37, 41, 71], but many possess different actuators or different objectives, so directly comparing their performance is difficult.

The POD-based methods and Galerkin models used in these works, however, are not easily transferred to the fluid flows around the HCP. The intuitive reason for this is that the relationship between the flow and the HCP itself is not easily captured in the framework of POD. POD does well at modeling phenomena that are low-dimensional in a "linear" sense (i.e., most behavior can be described by the superposition of several features). This property is present in cavity flows (with the dominant behavior being several superimposed cavity tones and their harmonics) and cylinder wakes (which have one dominant periodic structure that can be represented by the superposition of several modes), but the flow around the HCP is much harder to break down in this way.

The utility of the POD/Galerkin closed-loop control methodology has not been conclusively demonstrated on systems more complex than these testbeds from the literature. It has only been infrequently used on experimental systems with the associated difficulties of noisy measurements and unmodelable dynamics (though they have been applied to CFD systems using full DNS), and in a number of cases the "actuators" that are used are sometimes experimentally impractical (e.g., a body force on a portion the fluid) and often driven as much by the limitations of CFD as by practical utility (e.g., many actuators are chosen so that they can be represented as a boundary condition in the solver, hence the prevalence of blowers and cylinder

rotations as inputs). Finally, the difficulty of using CFD in problems in which the body's dynamics are coupled in a non-trivial way to the fluid (as is the case with the underactuated pole in the HCP, see Chap. 2) has prevented significant study of these methods to the domain of interest of this thesis.

As a result of these issues this thesis makes little use of complex flow modeling to improve control performance. Instead, the techniques presented here handle the fluid state's evolution implicitly, and were successful at achieving the desired tasks. However, including data-driven fluid models such as POD-based Galerkin models is an exciting direction of research, and while making them work will be challenging and require some non-trivial insight, they could ultimately improve the flexibility of the control design methods presented in this thesis, and allow them to apply to a wider range of flow conditions.

Chapter 2

The Hydrodynamic Cart-Pole

Fluid-body systems, particularly at intermediate Reynolds numbers in complex flows, are very difficult to model in a manner which is both tractable and sufficiently rich to capture the full range of observed dynamical behaviors. Low-order lumped-parameter systems can capture the important dynamics for certain situations[7, 106], but must be extensively validated and often require some amount of insight to apply to new domains. The Navier-Stokes Equations, when solved numerically via Direct Numerical Simulation, (DNS) are very computationally intensive and ill-behaved, and in some cases obtaining convergence can be difficult[14, 44]. Because of this, the study of novel high-performance control techniques for fluid-body systems at intermediate Reynolds numbers must be well-grounded on a representative experimental system. This system should capture the important difficulties of control, but be experimentally convenient enough to allow different control methodologies to be easily and quickly tried-out, validated or discarded.

This chapter describes in detail the construction and modeling of the representative experimental system used in this thesis: Hydrodynamic Cart-Pole (HCP). The HCP is a testbed well-suited to studying the control ideas presented in the following chapters. The HCP is robust and can be run for long periods of time with little maintenance, and exhibits the most important difficulties of controlling fluid-body

systems. These difficulties are:

- Nonlinearity
- Underactuation
- A large and difficult to observe fluid state
- Dynamics which are difficult to usefully model

The system is also designed such that real-time Particle Image Velocimetry (PIV), a technique which allows for the fluid velocity field to be directly observed, can be used inside the feedback-control loop (see §4 for a detailed explanation).

The remainder of this chapter is divided into two sections as outlined below:

- **HCP Hardware** - The physical Hydrodynamic Cart-Pole itself
- **HCP Modeling** - Development of a quasi-steady lumped-parameter model for states near the "upright" (i.e., unstable) configuration

Further information regarding the construction and design of the water-tunnel itself may be found in §A.1, and information regarding the hardware details used to perform PIV (e.g., information on the laser and camera) are in §4.1.2 and §A.2.

2.1 HCP Hardware

All the control tasks studied in this thesis take place on the hydrodynamic cart-pole (HCP); a fluid-dynamical analog of the much-studied cart-pole problem from the controls literature [29, 95, 76]. The HCP system has a similar geometry to that classical control problem, but instead of operating in the vertical plane against gravity it operates in the horizontal plane against weathercock stability in a free stream (see Fig. 2-1). This fluid-dynamical system is governed by much more complicated dynamics than its traditional counterpart, as the behavior of the pole is dictated

primarily by the fluid forces, while the pole's own inertia (the critical force in the traditional cart-pole system) is of comparatively little importance.

The important features of the HCP's construction are listed below:

- The pole pinned about its trailing edge, where a ball-bearing connects it to the cart.
- The connection to the cart cannot apply torque (i.e., it is a simple pin joint).
- The cart can be directly controlled via an input force u.
- The angular acceleration of the pole can only be controlled by moving the cart and taking advantage of fluid and inertial coupling between them.

The most fundamental task of the HCP consists of trying to balance (i.e., stabilize) the pole at $\theta = 0$ (i.e., pointing upstream) at what is termed the "upright"[1] configuration[2]. This configuration is unstable because the center of pressure (for a stalled flat plate) is at the midpoint of the pole, and thus ahead of the pin joint. Without control, the pole will fall over and point downstream (the "downright" configuration). Stabilization of the pole is not accomplished by applying torques directly to the wing. Instead, the cart (and thus the pin joint) is moved perpendicular to the flow (see Fig. 2-2).

Moving the cart is the only input to the system, and therefore the only torques applied to wing are due to the fluid and the pole's inertia. The positions of the cart and the pole are both measured with high accuracy by encoders, making the cart-pole (but not the surrounding fluid system) fully observable. The goal is not merely to stabilize the system, but to provide a stabilizing controller with as high performance as possible (e.g., fast tracking of desired inputs and robustness to large fluid disturbances such as bluff body wakes).

[1]This configuration is termed the upright because it is analogous to the upright, unstable configuration of the traditional cart-pole system.

[2]This is approximately stabilizing an unstable equilibrium, but technically the upright is not an equilibrium as vortex shedding effects prevent any one state from having zero torque at all times.

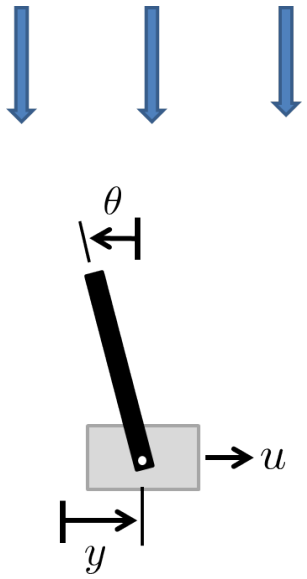


Figure 2-1: Schematic of hydrodynamic cartpole control task. Blue arrows indicate direction of fluid flow, x is the position of the cart, θ is the angle of the pole and u is the applied force (i.e., the input). The "upright" is defined as $\theta = 0$, an equilibrium made unstable by the center of pressure of the plate being ahead of its connection point (i.e., weathercock stability attempts to make the pole point downstream).

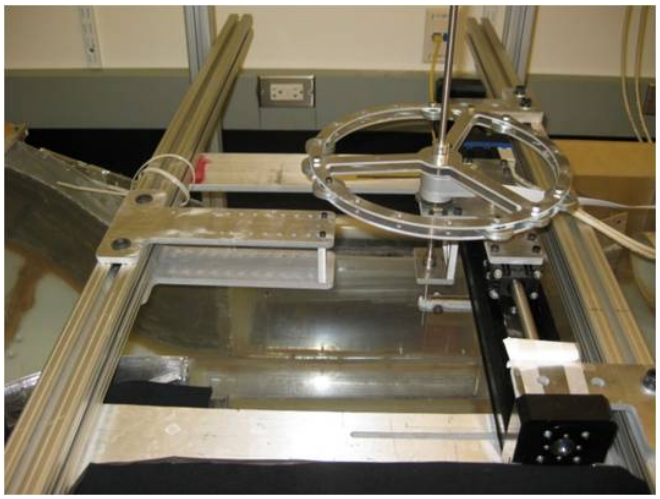


Figure 2-2: Photo of the test section of the water-tunnel setup. The water flows from left to right in this image. The black rails on the right are the linear actuator (the cart), and the circular aluminum wheel is a flywheel adding inertia to the wing (the pole), making it effectively less damped and thus more interesting to control. This disk can be removed or added to when designing the control task.

The pole is a flat plate with a rectangular cross-section of chord 9.5 cm and a thickness of 0.6 cm. The pole submerged 18 cm deep in water, and for the purposes of analysis it is treated as 2D. The water flows by the flat plate at 22 cm/s, and the available travel of the cart is 15 cm. The Reynolds number of the system, using the chord of the flat plate, is 20,000. These values are summarized in the following table, and more details on the water tunnel can be found in §A.1. The discrepancy between the test section width and the maximum cart displacement is due to hard-stops put in place to prevent damage to the test section.

Table 2.1: Dimensions and Values of HCP Parameters

Property	Value
Free Stream Speed	22 cm/s
Flat Plate Chord	9.5 cm
Flat Plate Thickness	0.6 cm
Test Section Width	22 cm
Depth of Plate into Water	18 cm
Cart Displacement Range	15 cm
Reynolds Number	20,000

The flat plate was chosen because it is a fundamental geometry, is relatively easy to study analytically, and is well-behaved optically (i.e., does not have significant lensing effects), making it the natural choice for the experiments in this thesis. See Figure 2-3 for a PIV image of the flat plate.

The cart used for the HCP is a high-force, high-speed linear actuator: a Copley STA1116 ServoTube actuator. This actuator is integrated with a linear bearing, providing a convenient package for generating linear motion. The most relevant actuator specifications may be seen in Table 2.2.

As Table 2.2 shows, the speeds and accelerations achievable by the actuator are significantly greater than those seen in the fluid (where the free stream speed is 22 cm/s). Thus, the actuator is able to move very quickly compared to the fluid; a desirable trait as the goal is to study high-performance controllers.

Table 2.2: Specification of STA1116 ServoTube Actuator

Property	Value
Peak Force	91.9 N
Peak Accel	141 m/s^2
Max Speed	4.7 m/s
Max Stroke	22 cm

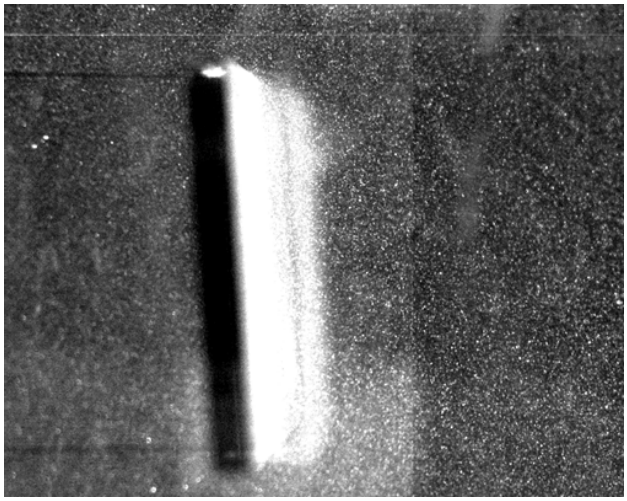


Figure 2-3: PIV photograph of flat plate immersed in water. The the bright dots are the seeding particles described in §4.

2.2 Modeling the Hydrodynamic Cart-Pole

The modeling of dynamical systems is an integral component in the vast majority control strategies, from classical bode-plot-driven design to modern H_∞-based robust control. The success of these model-based methods has been so dramatic in so many challenging domains that the limitations stemming from their dependence on a model can sometimes be forgotten. However, when attempting to apply these techniques to a fluid-body system, the difficulty inherent in obtaining a model becomes the primary obstacle to increased performance. In this section, a lumped-parameter model of the HCP is found using results from the fluid dynamics literature and experiments on the

hardware the do not require a controller[3]. This model is used to design controllers for basic tasks. The model's limitations, and a method to circumvent these limitations, is presented in §3.4.

The complete dynamics of the hydrodynamic cart-pole (HCP) are extremely nonlinear and of large state (due to the coupling with the fluid), and cannot be written down in a manner which is both complete and useful for control synthesis. However, by making a number of simplifying assumptions, a model applicable to the limited domain of operation relevant to balancing (i.e., near the upright and when the cart and pole are moving slowly) may be found. The relevant assumptions are:

- The pole acts as a flat plate (i.e., I ignore its thickness)
- The dynamics can be treated as quasi-steady (i.e., fluid state can be ignored)
- The cart is feedback linearizable
- The effect of fluid inertia can be captured by a simple added mass term
- The angular damping force is a linear function of velocity

These assumptions are clearly violated during high-performance maneuvers. However, if we wish to first simply balance the pole at the upright, a good controller based on a locally-valid model should keep the HCP in a domain in which the assumptions are approximately satisfied. Furthermore, as the HCP at the upright is open-loop unstable, obtaining an initial model from experimental data is not necessarily feasible. However, once an initial stabilizing controller is found, the system becomes much easier to study and many system-identification techniques are available for improving the model and the controller[54, 69, 103].

The dynamics of the cart can be dealt with easily as 1) the fluid forces transmitted to the cart through the pole are small compared to the mass of the cart and static friction and 2) a high-gain, high-frequency control loop exists between the motor

[3]If on finds an initial stabilizing controller without a model, it is possible to collect data on the HCP at the upright, and use this information to build a data-driven, identified model that may offer better performance. However, in general, finding a stabilizing controller without first having a basic model is a tricky proposition, and so the model presented was found without requiring an initial, stabilizing controller.

amplifier and the cart. These two factors allow the cart dynamics to be effectively feedback linearized, resulting in the dynamics:

$$\ddot{y} = u. \tag{2.1}$$

To model the dynamics of the wing (i.e., find $\ddot{\theta}$), first the quasi-steady fluid forces on a flat plate must be calculated. The coefficients for lift and drag (C_L and C_D respectively) on a fixed flat part can be written as:

$$C_L = 2\sin\alpha\cos\alpha \tag{2.2}$$

$$C_D = 2\sin^2\alpha, \tag{2.3}$$

with α representing the angle of attack (the angle between the wings major access and the fluid flow). The behavior with respect to angle of attack can be seen in Fig. 2-4).

The angle of attack α can be computed from the wing pitch angle θ and the direction of fluid flow relative to the wing ϕ. These can be computed as follows:

$$\phi = \arctan\left(\frac{-\dot{y}}{V_0}\right) \tag{2.4}$$

$$\alpha = \phi - \theta, \tag{2.5}$$

where $\dot{y}$ is the velocity of the cart and V_0 is the speed of the fluid flow in the laboratory frame.

The lift and drag forces on the plate can then be written as:

$$F_L = \frac{1}{2}\rho V^2 csC_L \tag{2.6}$$

$$F_D = \frac{1}{2}\rho V^2 csC_D, \tag{2.7}$$

with ρ the density of the fluid, $V = V_0^2 + \dot{y}^2$ representing the relative fluid velocity,

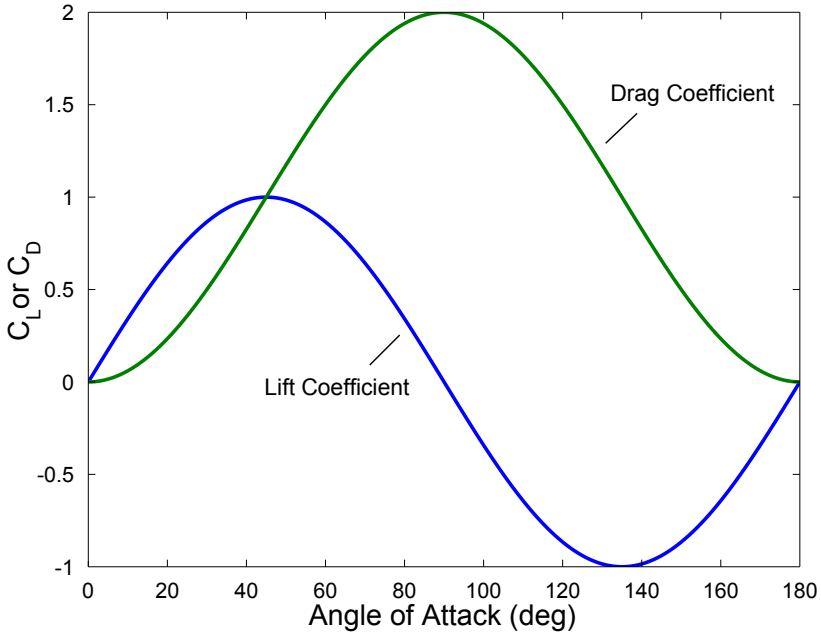


Figure 2-4: Flat plate theory lift and drag with respect to angle of attack.

c the chord length of the wing and s the span of the wing. The drag force is in the direction of fluid flow and the lift force perpendicular to the direction of fluid flow.

The torque resulting from these forces about the connection point of the flat plat can be written as:

$$\tau_{aero} = -F_L \frac{c}{2} \cos\theta - F_D \frac{c}{2} \sin\theta \tag{2.8}$$

$$= -\frac{1}{2} c^2 s \rho V^2 \sin(\alpha), \tag{2.9}$$

where θ is the angle of wing relative to the upright.

The effect of inertial torque resulting from the added mass of the fluid must then be included. The density of the pole is approximately that of the water (acrylic plastic is 18% denser than water), and is considered "thin" for the purpose of flat-plate analysis, and thus inertial forces on this plate will simply be included in the added mass calculation. The added mass for a flat plate moving in the direction of

its minor axis is given in [65] as:

$$M_{add} = \pi \left(c/2\right)^2 s\rho, \tag{2.10}$$

which can be interpreted as the mass of a fluid cylinder with a diameter equal to the length of the plate and a height equal to its span. This will result in an inertial torque equal to:

$$\tau_{inertial} = M_{add}\frac{c}{2}\cos\left(\theta\right)u, \tag{2.11}$$

where $\ddot{y} = u$ has been used.

Thus, the equations of motion for the system can be written as:

$$\ddot{y} = u, \tag{2.12}$$

$$\ddot{\theta} = \frac{\tau_{aero} + \tau_{inertial}}{I_{total}} - k_d\dot{\theta}, \tag{2.13}$$

where I_{total} is the total moment of inertia of the system (including the flywheel) and k_d is the linear angular damping coefficient. These two parameters were found via the following two experiments, neither of which required a balancing controller to perform.

The total moment of inertia I_{total} was found by assuming the holding the wing at a pitch angle of $\theta = 90^\circ$ and a speed of $\dot{y} = 0$, and measuring the initial angular acceleration of the wing. The linear angular damping coefficient k_d was found by spinning the wing by hand and fitting a decaying exponential to the resulting curve of $\dot{\theta}$ versus t.

This model is made use of in the following chapter to create a controller and an observer that allow for the upright position to be stabilized, and for some more aggressive maneuvers to be attempted.

Chapter 3

Control without PIV

Before attempting to control the HCP with PIV in the loop, a basic level of control must be achieved without knowledge of the fluid. In this operating condition disturbances to the free stream flow (e.g., vortices) and errors in the model are treated as perturbations, without understanding the structure behind them. The chapter is organized into four sections, as outlined below:

- **Model-Based Observer Design** - Design of a high-gain state observer using the model from §2.2.
- **Balancing Controller** - An LQR balancing controller for the HCP.
- **High-Speed Step Tracking** - A feed-forward controller for quickly tracking a step based upon the model from §2.2.
- **Trajectory Learning** - A data-driven trajectory improvement scheme offering higher performance than the model-based feed-forward scheme presented above.

3.1 Model-Based Observer Design

As the hardware only directly measures position and the controller methodology we wish to use (i.e., LQR. See §3.2) require velocities, an observer must be constructed

to provide as accurate velocity estimates as possible with minimum latency. The observer can also reduce measurement noise for the positions, but as we are using high-accuracy encoders for both the cart and the pole positions measurement noise is not significant for position (but can be significant for velocity as described below).

Simple discrete time differentiation of the form:

$$\dot{x}_n = \frac{x_n - x_{n-1}}{1}, \tag{3.1}$$

where ${x_n}^1 = [y\ \theta\ \dot{y}\ \dot{\theta}]^T$ is the state at time step n, is insufficiently accurate and far too noisy for the purposes of balancing, and naive filters with a large number of taps (i.e., filters that look at a long history of measurements) can result in excessive latency. A commonly used and effective form for an observer is a Luenberger Observer[56]. This observer has the form (in continuous time):

$$\dot{\tilde{x}} = f(\tilde{x}, u) + L(z - C\tilde{x}), \tag{3.2}$$

where $f(x, u)$ is a possibly nonlinear model of the system dynamics, $\tilde{x} \in \mathbb{R}^n$ is the state estimate, $z \in \mathbb{R}^m$ is the vector of measurements, $C \in \mathbb{R}^{m \times n}$ is the observation matrix and $L \in \mathbb{R}^{n \times m}$ is the matrix of observer gains. Note that the steady-state Kalman Filter is a special case of this form of observer in which the dynamics are linear and the observer gains L are in some sense optimal for a linear system.

As the assumptions made regarding noise for the Kalman filter (e.g., Gaussian and independent of state) are not valid for the HCP system there is little advantage to using the procedure for a Kalman filter to find L. Instead, the gains in L are selected such that the nominal poles for the observer (e.g., the time scale for the error dynamics) all have a real part near $-1/\epsilon$ where ϵ is a parameter chosen by the control designer. In this work, $\epsilon = 0.05$ was chosen, placing the real parts of the observer poles near $s = -20$ where s is the Laplace variable. The L to achieve this for the

[1]This means the state is organized as [cart position, pole angle, cart speed, pole speed].

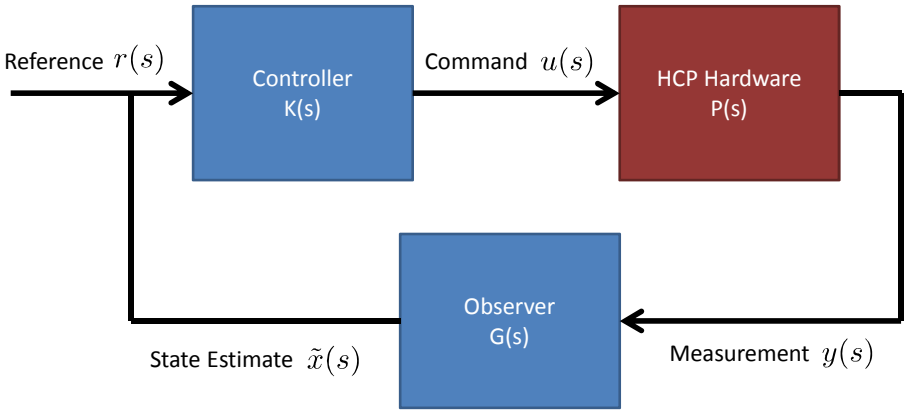


Figure 3-1: A schematic representation of the linear control loop for the HCP. When PIV is included in §5 the components needed to handle the additional sensing add complexity, but for now this simpler understanding will suffice. In general, a nonlinear observer $G(y, u)$ is used, but near the upright a linear $G(s)$ is sufficient.

HCP (when linearized about the balancing configuration) is given by:

$$L = \begin{bmatrix} 2/\epsilon & 0 \\ 0 & 2/\epsilon \\ 1/\epsilon^2 & 0 \\ 0 & 1/\epsilon^2 \end{bmatrix}, \tag{3.3}$$

with resulting observer poles at:

$$s = -20, -20, -21.1 \pm 5.687i. \tag{3.4}$$

For more detailed analysis, see §A.3.

3.2 Balancing Controller

With the nonlinear model developed in §2.2 and the observer $G(s)$ described in §3.1, all that remains to be found to complete the control loop shown in Figure 3-1 is the controller $K(s)$. One could attempt to design this controller by hand, but a natural starting place for linear control design is the Linear Quadratic Regulator (LQR). This control design technique requires a linear model of the system, and the system's full state. An estimate of the state is available through the observer $G(s)$, and the nonlinear model $\dot{x} = f(x, u)$ can be linearized to give:

$$\dot{x} = Ax + Bu \tag{3.5}$$

$$A = \partial f / \partial x \tag{3.6}$$

$$B = \partial f / \partial u, \tag{3.7}$$

thus it is reasonable to design and apply an LQR controller. This controller will have the form:

$$u = -K\tilde{x}, \tag{3.8}$$

where K is the LQR gain matrix, consisting of proportional and derivative gains (i.e., gains on position and velocity error), and $\tilde{x}$ is the output of the observer discussed in §3.1.

For a linear system, it can be shown that an LQR controller minimizes the quadratic regulator cost:

$$\text{Cost} = \frac{1}{2} \int_{t=0}^{t=\infty} \left(x^T Q x + u^T R u \right) dt, \tag{3.9}$$

where x is the system state, u is the control input and Q and R are cost matrices chosen by the designer. Choosing these matrices can sometimes be a tricky task, as the desired system behavior is rarely exactly captured by minimizing the cost shown in Eq. (3.9). However, LQR controllers often perform "well enough," particularly for

underactuated systems where hand-designing controller can be very difficult, that the disconnect between cost and desired behavior is worth accepting.

Using the cost matrices Q and R given below in the LQR design, a successful controller with gain matrix K can be found:

$$Q = \begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \tag{3.10}$$

$$R = [1] \tag{3.11}$$

$$K = \begin{bmatrix} -5.4772 & 5.2500 & 2.9238 & 2.3511 \end{bmatrix}. \tag{3.12}$$

Note that the gain on y is negative, indicating that the controller will actually move to *increase* the error in this state variable. The stabilizing effect comes about through the interplay between y and θ – a common property when controlling underactuated systems, and one of the reasons why hand-designing controllers for them is difficult.

3.2.1 Resulting Performance and Robustness

With a reasonable LQR controller designed and the system carefully centered (see §A.4) the actual performance of the system can be explored. There are two important considerations: the ability of the controller to balance with minimal deviations when the incoming flow is undisturbed (i.e., the "smooth flow" configuration; see Fig. 3-2 and Fig. 3-3) and the capacity for the controller to reject disturbances, both perturbations by hand and complex incoming flow.

The performance of the system in the smooth flow configuration can be seen in Fig. 3-4. The small oscillations seen here likely primarily a result of stiction in the actuator, but lingering centering issues are also a possibility[2]. The controller also

[2]It is tempting to attribute this oscillation to vortices shedding from the wing, but the frequency

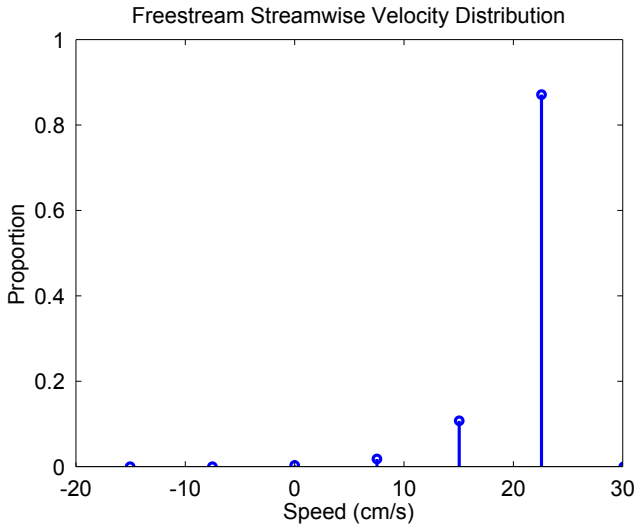


Figure 3-2: Distribution of measured velocity in the freestream in the nominal flow direction, as measured by the PIV system described in Chap. 4. Velocity vectors were median filtered to reject outliers (see Chap. 4 for more details). Due to the discrete nature of the PIV offsets and the median filtering only the discrete set of measurements plotted here are possible.

effectively rejects disturbances, as can be seen in Fig. 3-5. Perturbations by hand are quickly rejected, and the periodic wake of a cylinder (whose diameter is approximately 1/3 the chord of the pole) placed upstream from the HCP cannot cause dramatic deviations from the desired set point.

3.3 High-Speed Step Tracking

Despite the controller's robust stabilization of the HCP demonstrated in §3.2, the controller's ability to track an aggressive trajectory in y while keeping the pole balanced is underwhelming, as seen in Figure 3-6A. In this figure, the cart attempts to follow a step in position (e.g., move y from 0 cm to 4 cm) while keeping the pole at the upright. Improving performance by increasing the gains, however, is not an option, as pushing the gains significantly higher than those in Eq. (3.12) results in

of such shedding is much higher than that seen in the periodic behavior of the HCP at upright.

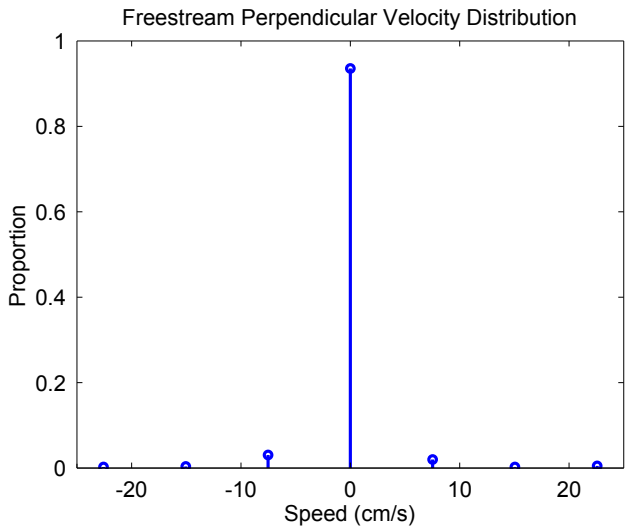


Figure 3-3: Distribution of measured velocity in the freestream perpendicular to the nominal flow direction, as measured by the PIV system described in Chap. 4. Velocity vectors were median filtered to reject outliers (see Chap. 4 for more details). Ideally these measurements would all be zero, representing no flow perpendicular to the nominal flow direction. Due to the discrete nature of the PIV offsets and the median filtering only the discrete set of measurements plotted here are possible.

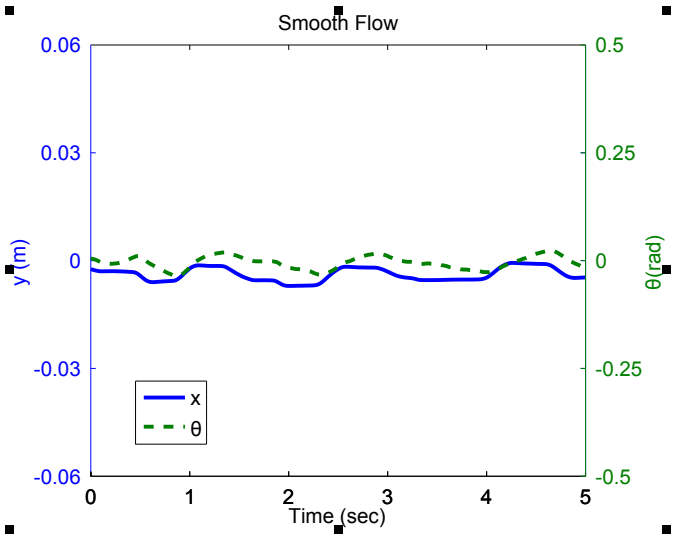


Figure 3-4: Hydrodynamic Cart-pole state (y and θ) when balancing in smooth incoming flow. The slight periodic oscillation is a result of static friction preventing very small corrections. Because of this, some error must accrue before the controller responds, resulting in the low-frequency, low-amplitude oscillation witnessed here.

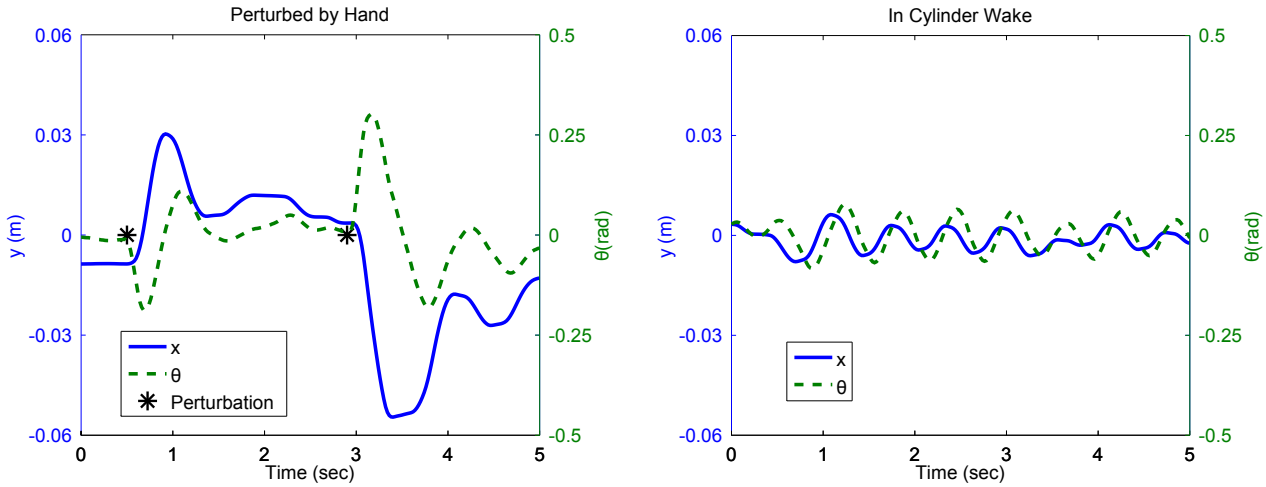


Figure 3-5: Hydrodynamic Cart-pole state (y and θ) when balancing while responding to perturbations by hand and a cylinder wake.

instability[3].

A sensible and general-purpose means of improving performance over the balancing controller (in which the desired behavior is effectively treated as a disturbance) is to design a "feed-forward" controller. A feed-forward controller does not simply respond to errors, as is done by the LQR feedback controller. Instead, a feed-forward controller pre-computes nominal control inputs and state trajectories based upon the specified task, and then uses feedback to respond to measured errors. These nominal state trajectories are then stabilized by feedback. The result is a controller with a different form than given in Eq. (3.8). Instead, it may be written as:

$$u(n) = -K(n)\left(x(n) - x_{ff}(n)\right) + u_{ff}(n), \tag{3.13}$$

where x_{ff} is the feed-forward state trajectory and u_{ff} is feed-forward control input, $K(n)$ is a time-varying gain matrix and the other terms are the same as in Eq. (3.8). The reason $K(n)$ varies with n is because the linearized model around $x_{ff}(n)$ is a

[3]The gains are increased by reducing the value of R, not by simply increasing the gains directly. This distinction is important, as even on the linearized system simply scaling the gains (e.g., by a linear scaling: $K_{new} = \alpha K_{old}$ with $\alpha > 1$) can result in instability.

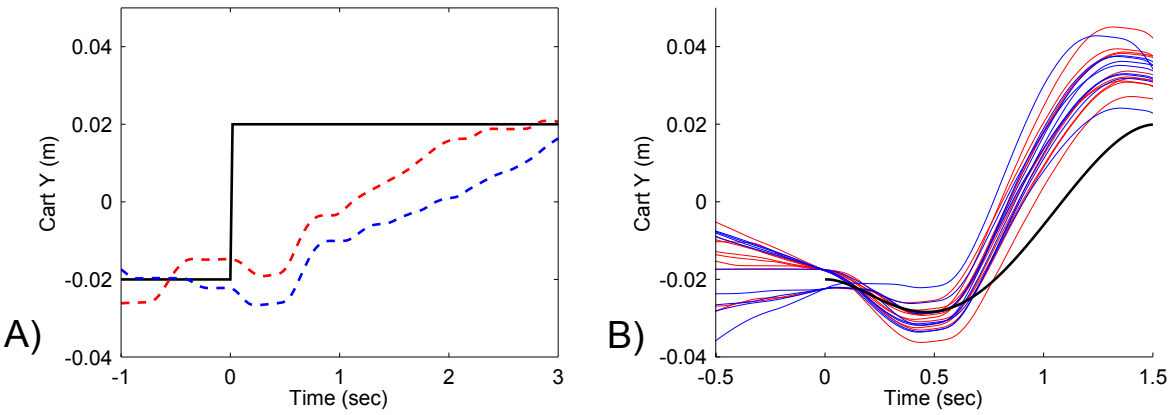


Figure 3-6: Two controllers attempting to track a step input (note that different time scales were used as the controller in B was much faster than that in A). A) Tracking performance of the balancing controller from §3.2. The slow rise time cannot be improved simply by increasing the controller gains due to the inadequacy of the model when violating the quasisteady assumptions. B) Tracking performance of feed-forward controller, in which a trajectory optimizer has found state and input trajectories based upon the quasi-steady model from §2.2.

function of n, thus the appropriate $K(n)$ changes as well. Note that the appropriate $K(n)$ for a given n depends on the model at all future $n' \geq n$, and thus must be computed starting at the end of the trajectory.

By using a feed-forward controller, the full nonlinear model from §2.2 can be used to design the control input, and the controller can take advantage of its advance knowledge of the particulars of the desired task. In general, for a nonlinear model, this will be a nonconvex optimization, and there can be some tuning involved in solving an initial feasible trajectory. Fortunately, modern trajectory design techniques (using optimization packages such as SNOPT[31, 32]) are quite robust and computationally efficient, and a feasible solution can often be found with their help. While SNOPT is a general purpose solver, trajectory optimization problems can be naturally represented in its general optimization framework. There are several forms in which the problem can be encoded, with assorted advantages and disadvantages. The most common are shooting methods [16] and direct collocation or direct transcription [104, 13].

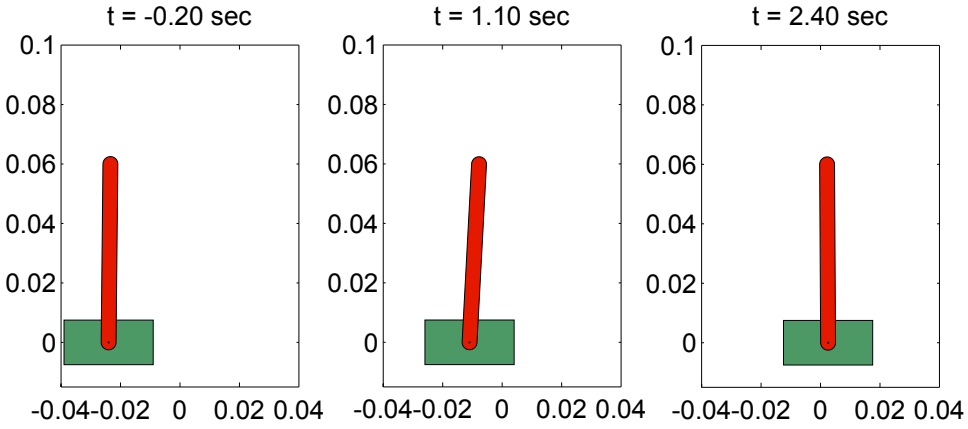


Figure 3-7: Schematic snapshots of the HCP system executing a step using the LQR controller as plotted in 3-6A. Note that even after 2.4 seconds it has not completed the step, and is never moving very aggressively.

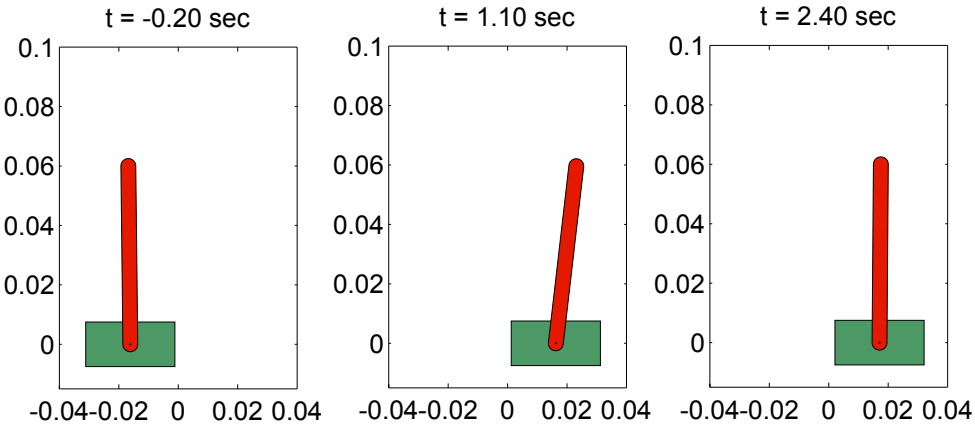


Figure 3-8: Schematic snapshots of the HCP system executing a step using the final feed-forward controller as plotted in 3-10. The initial feed-forward controller is similarly fast, but exhibits significant overshoot, which is seen in 3-6B and compared to the final controller in 3-10.

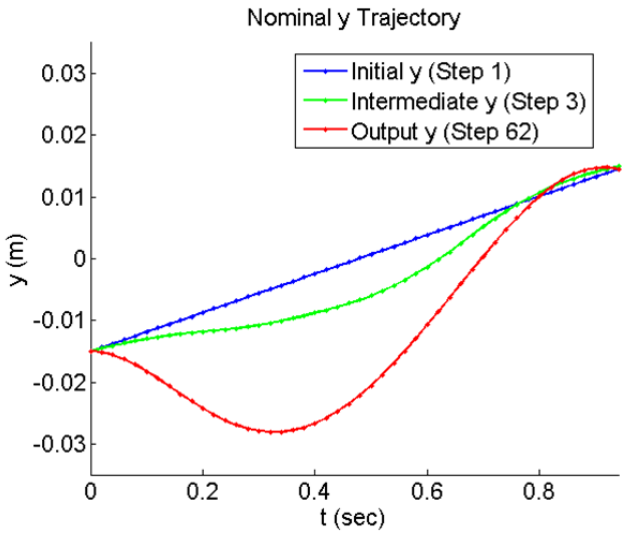


Figure 3-9: Initial "seed" trajectory, trajectory after several optimizer steps, and final trajectory, as found by SNOPT for the HCP model developed in this chapter.

To find a feed-forward trajectory for the cart-pole system I used a direct collocation routine, in which both the input tape and the state trajectory are discretized and included as variables to be optimized, and the dynamics (along with initial and final states) are imposed as constraints. The specified initial and final states are:

$$x_0 = [0.0 \;\; 0.0 \;\; 0.0 \;\; 0.0], \tag{3.14}$$

$$x_f = [0.04 \;\; 0.0 \;\; 0.0 \;\; 0.0] \tag{3.15}$$

with the state ordering as before (i.e., $[x \;\; \theta \;\; \dot{x} \;\; \dot{\theta}]$). The initial inputs (the $u(t)$) were set randomly with a small variance and the initial states were linearly interpolated between their initial and final values. Finally, a quadratic cost on input magnitude equal to that used in the balancing section was included. This formulation robustly produced a reasonable feasible trajectory at convergence. See Fig. 3-9 for an example of how the solver converges towards a feasible trajectory.

When this trajectory is executed, however, the limitations of the model come into stark relief. The system suffers from a significant amount of overshoot on average,

with the mean trajectory peaking approximately 25% above the desired point. There is also a reasonable amount of variance in behavior, though this can largely explained by variance in the initial state at which the feed-forward controller was engaged, and not due to more fundamental randomness in the system.

Improving the speed and accuracy of step tracking beyond this is a nontrivial task, as the control designer is now directly contending with the limitations of the model. For the sort of high-speed motions of the pole required for high-speed step tracking, the quasi-steady assumption used in §3.4 is significantly violated, and the fluid states, ignored in the quasi-steady model, become very important. A general technique for dealing with these issues is presented in the following section, where indirect adaptive control is used to increase the speed and accuracy of high-speed step-tracking on the HCP.

3.4 Trajectory Learning

Though the previous section introduced the feed-forward step, the resulting performance was not yet satisfactory. Due to the limitations of the quasi-steady model developed in §2.2 the actual system, when executing the step (with a stabilizing linear-time-varying controller), overshot significantly and suffered from conspicuous ringing. This is not surprising as the model made the assumption of quasi-steadiness and thus did not attempt to incorporate the fluid state. However, critically, the error experience by the system is relatively repeatable. Because of this, data-driven approaches similar to Iterative Learning Control can be investigated.

In this chapter a general-purpose iterative method will be presented that significantly improves the performance of the HCP when tracking a step. This method does not require a great deal of *a priori* information regarding the system, save assumptions of sufficient smoothness (what qualifies as sufficient will be discussed later in the chapter). The efficacy of the technique is demonstrated experimentally on the

HCP, with results appearing in §3.4.4.

3.4.1 The Proposed Algorithm

Despite the difficulties in modeling discussed in the §2.2, and the resulting apparent limits in performance (see §3.3) high-performance control is possible. The important realization is that the difficulty lies in obtaining an accurate model over a large region of state space, not in modeling the local behavior of the system near the states relevant to the current task. Thus, if one wishes to perform a specific behavior, and the system can be approximately reset to an appropriate set of valid initial conditions, a local model can be used in place of a much more complicated (and difficult to obtain) general model. By modeling only the local behavior around the current control actions, many simple but flexible model classes are sufficient even for complex tasks such as tracking a step on the HCP.

Once a suitable model class has been chosen (see §3.4.2 for a description of the model class used in this chapter) and an initial feasible controller found (e.g., a controller based on the quasisteady model described in §2.2), controller performance can be optimized by executing the following procedure until convergence (i.e., performance no longer improves):

1. Collect data using the current controller
2. Fit a model in the proposed model class
3. Optimize a proposed controller for the resulting model
4. Perturb the controller in direction of the proposed controller
5. Collect data using the new controller
6. Repeat

This procedure, which will be described in detail in the remainder of the section, allows a *local* optimization to be performed, in which each controller explores a new region of state space, and determines how the controller should be locally updated, until it reaches a controller than cannot be improved further using local methods. This fixed point represents convergence to a local optimum.

3.4.2 Fitting a Local Model

While there is a great deal of freedom for the designer when choosing a model class, this paper will focus on a powerful, general-purpose choice which is widely applicable for finite time maneuvers (e.g., the step tracking task introduced in §3.3). The proposed model class is a linear time-varying (LTV) model in discrete time. Using a discrete time model (as opposed to a continuous time one) is more appropriate, as the state is sampled by a computer at discrete time steps. The resulting model has the form:

$$x_{n+1} = A_n x_n + B_n u_n \tag{3.16}$$

where x is the state, n is the (discrete) time, u the input and A_n and B_n matrices with of appropriate sizes. Furthermore, not every entry of A_n and B_n must be fit. In fact, as only the fluid forces are substantially unknown, we need only fit the entries related to the unknown effect of the fluid on the pole; ultimately resulting in four unknown scalars for every time n. The scalars represent discrete-time interpretations of the added-mass, damping coefficient, and lift and drag of the pole.

The model can be easily fit to measured data via least-squares linear regression. The notion of sufficiently rich data for fitting a model is well-developed for linear systems, and the practical requirements for achieving this richness as outlined here. First, the controller must be executed several times from similar initial conditions (initial conditions should not be identical, and never will be due to the complexity of the fluid state). Then, the free parameters of A_n and B_n can be fit such that

the squared-error between the model proposed in Eq.(3.16) and the measured data is minimized. Rather than fitting different values for each n, A_n and B_n can be made first-order splines (i.e., piece-wise linear functions of n) with the fits weighted based upon activation of the value (i.e., high when close, low when far and zero when beyond the next knot point). This effectively enforces model smoothness over time, and results in better fits as there is more data per fit and smoother variations in the model parameters.

3.4.3 Optimizing Controller and Updating Controller

Assuming that the system has not yet converged to the desired behavior, there will be a non-zero error between the target behavior (e.g., fast tracking of a step) and the measured behavior. Using the local discrete-time model found in the previous section and standard trajectory optimization techniques, one can find a "corrective" feed-forward trajectory that cancels out a portion of the observed disparity between the desired and the observed behavior. In the case of tracking a step, if the measured data tend to overshoot the desired trajectory, a corrective trajectory that undershoots the observed behavior would be found. For models from the class described above and the suggested quadratic objective, finding this corrective trajectory is a convex optimization problem, and easily solved computationally.

The reason that only a portion of the observed error can be corrected in a single iteration is the fact that a large corrective trajectory may require large deviations from the previous feed-forward trajectory, and thus may push the system into a region of state space where the local model is invalid. Therefore, the corrective trajectory cancels out only a portion of the observed error, with the size of the correction determined by η; a parameter chosen by the user which is generally less than 1. This perturbation results in a local update in the "right" direction, and takes into account the complicated couplings between the states of an underactuated systems. If the update is appropriately sized (i.e,. not too large) the next step will result in less error

than before. This update can be written as:

$$x_{ff}^{m+1} = x_{ff}^{m} + x_{corr}^{m} \tag{3.17}$$

$$u_{ff}^{m+1} = u_{ff}^{m} + u_{corr}^{m}, \tag{3.18}$$

where m is the current iteration of the algorithm, x_{ff}^{m} and u_{ff}^{m} are the feed-forward trajectories for the current iteration, x_{corr}^{m} and u_{corr}^{m} are the corrective trajectories found in current iteration. The end of the resulting trajectory (e.g., the last several time-steps) is then smoothly blended towards the desired final state to ensure that there is no discontinuity in the desired state when the feed-forward trajectory is completed. As the algorithm converges the amount of blending can be reduced and, if the desired final state is ultimately reached, can be eliminated entirely. A new stabilizing feedback gain $K^{m+1}(n)$ can then be found for this updated trajectory by performing LQR for the current local model.

3.4.4 Performance Results

To demonstrate the effectiveness of the algorithm just suggested, it was implemented on the HCP to improve fast step tracking. The feed-forward controller performance presented in §3.3 provides both a baseline for performance and an initial controller to act as the starting point for the optimization.

The task was to execute a step of 4 cm on the HCP, starting and ending with zero velocity and the pole balanced at the upright. The initial controller is that given in §3.3, with the tracking performance shown in Fig. 3-6B. Six iterations were performed, with $\eta = .1$, and 20 trials at each step (used to provide data for fitting a local model). The result of this process can be seen in Fig. 3-10.

The improvement is dramatic and repeatable, with the initial controller resulting in the mean final state error magnitude given below as $x_{err}^{m=0}$, while after six iterations

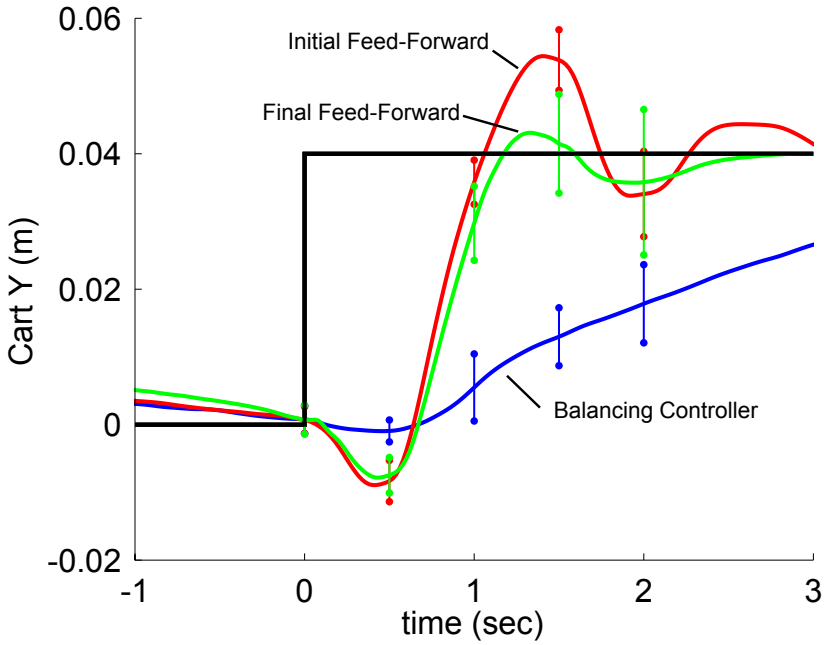


Figure 3-10: Plot showing the step tracking performance of the balancing controller, initial feed-forward controller and final feed-forward controller. The bars indicated +/- one standard deviation over 20 trials.

of the algorithm the mean final state error magnitude is that given as $x_{err}^{m=6}$.

$$x_{err}^{m=0} = [1.23 \text{ cm}, 0.1688 \text{ rad}, 2.81 \text{ cm s}^{-1}, 0.2938 \text{ rad s}^{-1}] \tag{3.19}$$

$$x_{err}^{m=6} = [0.10 \text{ cm}, 0.0365 \text{ rad}, 0.70 \text{ cm s}^{-1}, 0.0358 \text{ rad s}^{-1}] \tag{3.20}$$

The variations in the measured trajectories shown in Fig. 3-10 are largely due to the relatively broad range of initial conditions. The fact that the performance improvement is repeatable despite this variation is a demonstration of the robustness of the controller class, and the technique.

Chapter 4

Real-Time PIV

A common challenge encountered in the control of fluid-body systems is the inability to observe the state of the fluid. In many cases the dynamics of the fluid are either much slower or much faster than those of the system, and thus can be reasonably considered quasi-steady. This quasi-steady assumption greatly simplifies the problem, as the state of the fluid can be effectively separated from the state of the system. For example, very fast fluid dynamics will reach steady-state quickly enough, and any oscillations will be sufficiently high-frequency, that their transient nature can be ignored. Very slow dynamics, on the other hand, can be captured by slowly adapting the system parameters (i.e., the "time-scale separation" required for effective adaptive control is present).

However, in situations where the dynamics of the fluid are on a similar timescale to those of the body, the time-varying nature of these fluid states becomes important to the performance of the system as a whole. The need to capture the effectively infinite number of states in the fluid, combined with the highly-complex dynamics by which they evolve, is daunting. Fortunately, in many cases the behavior of the fluid component of a fluid-body system is highly structured, and relatively few states are needed to capture the relevant "first-order" behavior of the fluid. Furthermore, the dynamics of these states can be quite simple in some cases.

This chapter focuses on a relatively general situation that possesses these simplifying qualities: the shedding of a vortex off of a rotating flat plate. This situation is used to illustrate the basics of the PIV algorithm, and the manner in which it was implemented to achieve real-time speeds. This situation does not involve a control task, though it does involve real-time tracking of non-trivial fluid states capturing the location and strength of the vortex. Note that the question of tracking fluid states in real-time opens up many interesting questions surrounding the design of causal filters for tracking fluid states; something which has not been considered in situations where PIV is performed offline and acausal filters can be used.

The remainder of this chapter is organized around this problem. The first section gives an overview of the PIV algorithm, using the a flow dominated by a single large vortex as an example. The second section presents the real-time vortex tracking problem, with associated tracking results. This second section will also present an effective general purpose method for finding relevant features in a PIV flow field: spatial match filtering. The third section discusses a means of improving the efficiency of tracking via sparsification; i.e., selecting the "most informative" measurements in a principled way so that less informative measurements may be omitted without compromising tracking performance. This reduces overall computational burden and decreases the latency of the PIV data.

4.1 PIV Algorithm

Particle Image Velocimetry (PIV)is a technique used extensively to aid the study of fluid dynamics by providing a quantitative description of the fluid velocity field. In rough terms, it does this by seeding the fluid with small reflective particles and tracking the motion of the fluid by following the motion of groups of particles[1]. There are, however, a number of details that must be considered if this is to be performed

[1]There is a similar technique, Particle Tracking Velocimetry or PTV, that tracks individual particles themselves.

successfully. There exists a wide body of literature explaining PIV and associated improvements and variants of the general scheme (see [36, 4, 91, 35]), but in this section a brief overview will be presented along with details of the real-time, parallelized version implemented for this thesis.

4.1.1 Overview of PIV

Consider a simple flow of interest: a single, largely isolated vortex. In this work, this vortex is generated by spinning the pole of the HCP, which results in the shedding of a large vortex from the edge of the plate without significantly disturbing the remainder of the fluid. For a schematic of the situation, see Fig. 4-1. While the existence of the vortex is clear to the eye, a quantitative description of the flow can be obtained via PIV.

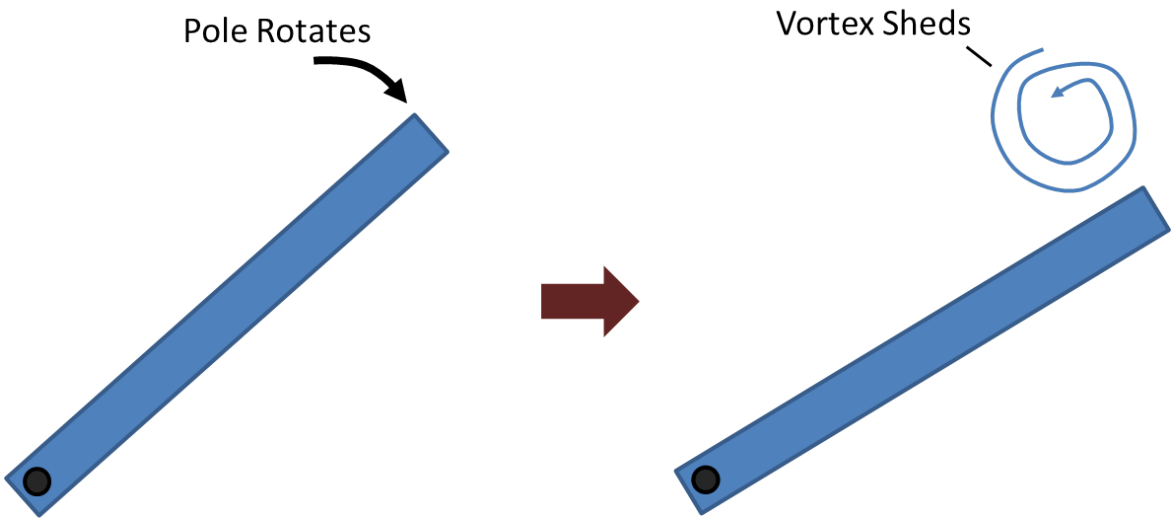


Figure 4-1: A schematic of the vortex shedding process, in which the pole of the HCP is rotated to shed a single, dominant vortex that can be identified by eye.

The two images in Fig. 4-2 were taken 33 ms apart, with the particles moving only a small distance between frames. The displacement between the two can be found by taking small regions of pixels in the first image and measuring their "distance" from

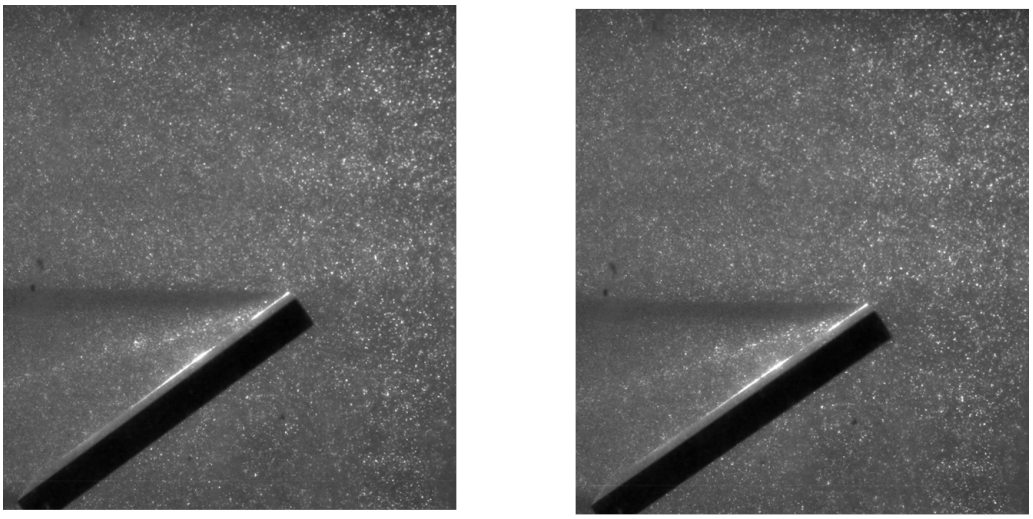


Figure 4-2: Two consecutive images of the single vortex shedding off the rotating wing. PIV operates by comparing small subareas of the image of the images and finding what displacement produces the best match, thus giving the displacement of the fluid.

regions of pixels in the second image. The minimum of this distance is where the two image regions match up best, and is thus the most likely displacement for that region of the flow. By repeating this process for many subregions across the first image, many vectors can be found, providing a vector which describes the motion of the fluid throughout the image. This is clearly not foolproof, and depends upon appropriately sized regions, appropriately dense particle seeding, an appropriate concept of distance, reasonably two-dimensional flow in the plane of the laser and an appropriate time between images (i.e., the "interframe time"). In practice, however, PIV can work very well.

The size of the pixel regions, seeding density and the interframe time are often set by rules of thumb, but a common and robust distance metric that can often be used

is a function referred to as the "error correlation," and is written as:

$$\text{Distance}(\Delta X, \Delta Y) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(g(X_i, Y_j) - g'(X_i + \Delta X, Y_j + \Delta Y) \right)^2, \quad (4.1)$$

where the region of interest is size N-by-M, $g(\cdot, \cdot)$ is the first image, $g'(\cdot, \cdot)$ is the second image, (X_i, Y_j) is the pixel location in the first image, and $(\Delta X, \Delta Y)$ is the offset to the second image. The minimum of this function over $(\Delta X, \Delta Y)$ gives the displacement of the fluid between the frames, and thus the local fluid velocity. Executing this process (and performing a simple outlier-rejection step) produces vector fields with enough quality to be highly useful, as seen in Fig. 4-3.

4.1.2 PIV Hardware

The PIV system (see Fig. 4-4 and Fig. 4-5) used for these experiments was designed around the need for real-time processing at rates sufficient to be useful in control. The hardware used for imaging and illumination (i.e., the camera and the laser) are fairly standard, but the camera was chosen with particular care for its imaging and data rates. These rates had to be fast enough to ensure that the relevant fluid dynamics could be effectively captured, and the resulting images had to be offloaded to processing computer fast enough that latency would not be a significant issue (i.e., the images must be streamed to the processing computer in real-time).

4.1.3 Computation

If one considers the computational demands of the algorithm, it is clear that a very large number of operations are required to produce a vector field. Because of this, PIV has often been used in an offline setting where images are captured and stored and processed after the experiment has been run. This is useful for understanding a system and for modeling purposes, but the information cannot be used in a control loop as it

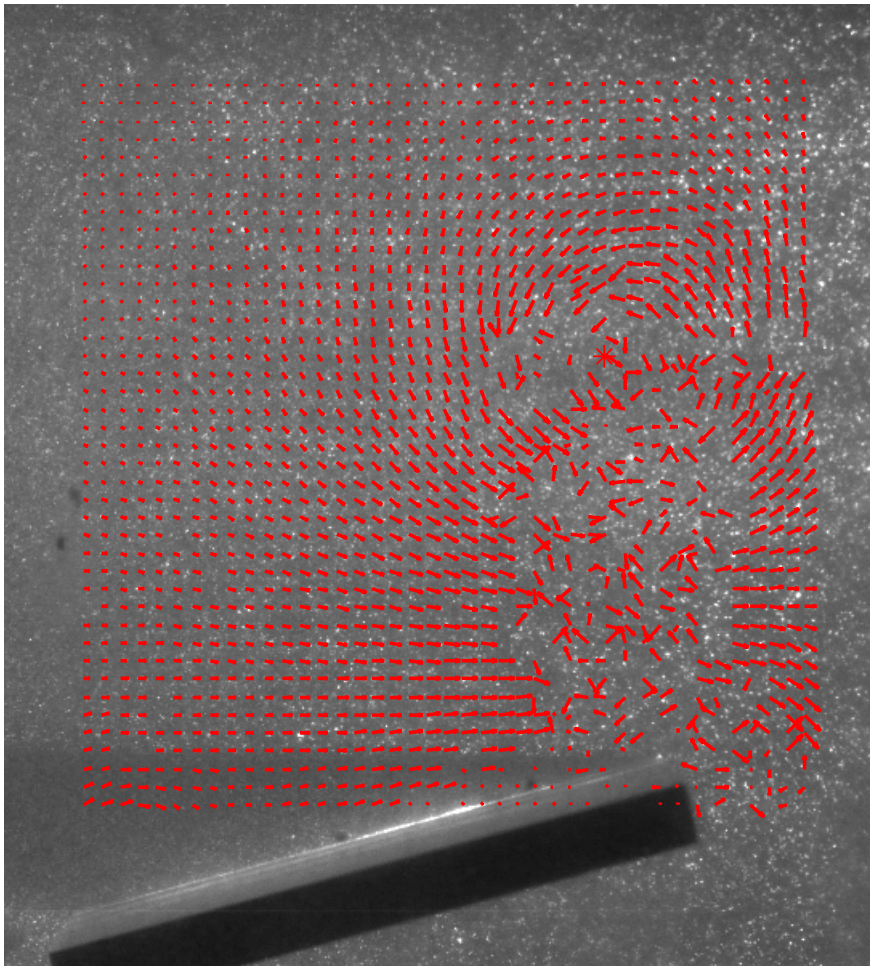


Figure 4-3: The vector field of the shed vortex superimposed over one of the images used to generate it. Note that despite faulty vectors in the center of the vortex and between the vortex and the wing itself, the vortex structure is clearly visible.

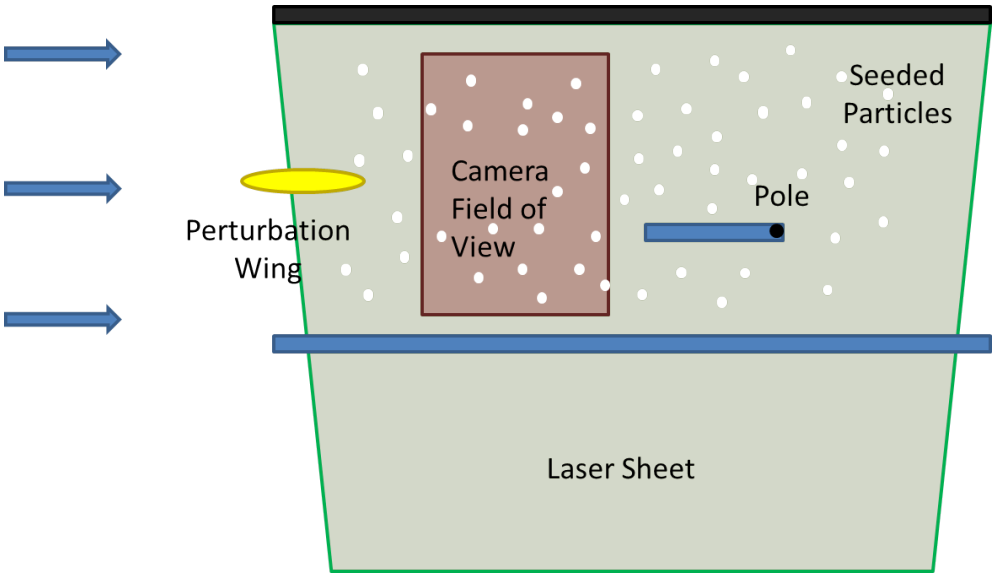


Figure 4-4: Schematic of a top-down view of the test section when performing PIV. Flow is from left to right, and the flat plate as shown is in the "upright" configuration. The perturbation wing is used to generate the fluid perturbations used in the Chapter 5. The laser sheet is produced by passing a laser through a cylindrical lens, and the seeding particles are $50\mu m$ polystyrene beads.

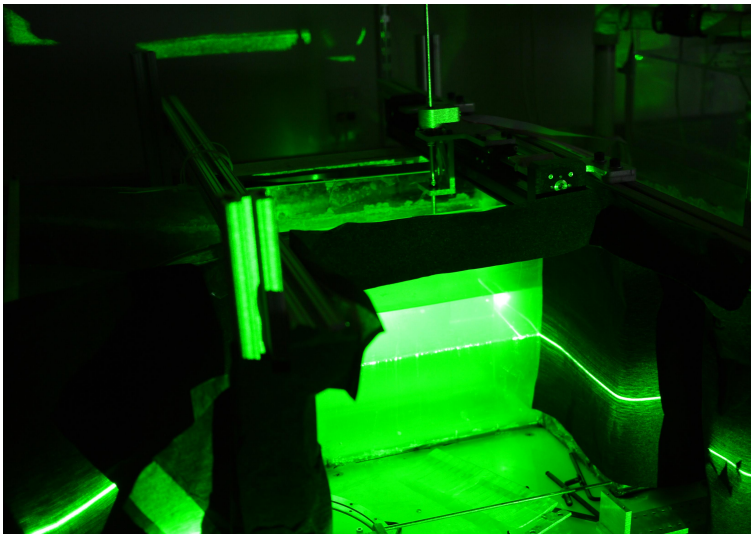


Figure 4-5: The PIV hardware in operation. The green laser sheet illuminates the seeded particles in the flow, while a camera images from below, providing the actual PIV images.

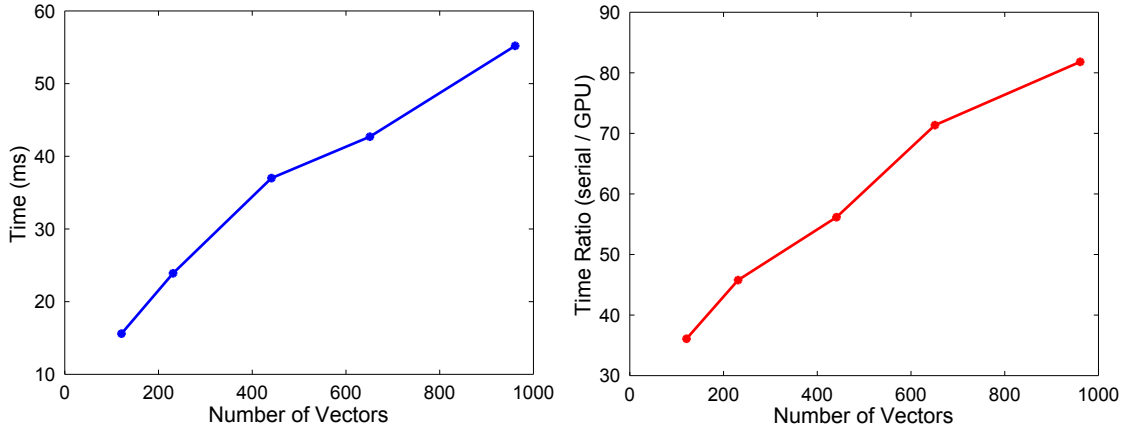


Figure 4-6: Total time to perform a set of correlations using the GPU implementation of PIV, and ratio of serial computation time to GPU time. Masks are 21-by-21 pixels and vectors are maxima over a 31-by-31 grid of offsets. At 15 image pairs per second, 66.7 ms are available between PIV calls. The serial computation time is very linear with the number of vectors, while the time on the GPU is approximately linear with some variation as seen above. About 30ms of the time spent in this computation is simply moving the image onto the GPU (which, given the implementation used here, must be performed independent of the number of vectors desired).

is unavailable at runtime. However, the structure of the problem (many independent comparisons and minimizations) is very naturally parallelized. Modern GPUs are theoretically capable of calculation rates of over a teraflop (trillion of floating point operations per second), completing very demanding calculations in milliseconds. With this newly available computational capability, PIV can now be used within a control loop to provide the state of the fluid to the controller. The GPU used for this thesis is a NVIDA GeForce GTX 570 with 480 cores. Fig. 4-6 demonstrates the enabling potential of GPU-based PIV implementations.

4.2 Vortex Tracking

Tracking an isolated vortex in otherwise largely stationary flow (i.e., no free stream) is non-trivial problem that is nonetheless easy to understand and formulate. If we wish

to track the center of a large, isolated vortex (as described in the previous section), we must find a robust way to turn the large number of noisy vectors produced by PIV into something more abstract, and useful: a low-dimensional description of the vortex location and strength. Due to the noisiness and outlier level of the vector fields obtained by real-time PIV, the most common approach of finding vortices in PIV - computing the curl of the vector field and thus local vorticity - is a non-starter.

Fortunately, theoretical models of vortices are available that capture many of the important structural aspects of the flow field in a relatively large region around an isolated vortex (i.e., away from its center). Potential flow theory provides a model of an inviscid vortex which can be quite accurate away from the vortex center. Viscous flow theory, on the other hand, suggests that the core of a vortex should rotate like a rigid body. By blending these two theories together in a manner consistent with experiment a reasonable *a priori* model of a vortex can be developed. This model is effectively a smoothed Rankine vortex [5], and can be used for spatial match filtering in which a filter searches for the structure of a vortex (in terms of its velocity field) in the vector field provided by PIV. The remainder of this section will be devoted to explaining the fluid model behind the model vortex, detailing the implementation of the match filter, and demonstrating its performance on real flow data.

4.2.1 Vortex Modeling

The basic idea behind the model vortex is to create a "smoothed" Rankine vortex. A standard Rankine vortex is defined by the following function for tangential velocity:

$$u_\theta(r) = \begin{cases} \Gamma r/(2\pi R^2) & : r \leq R \\ \Gamma/(2\pi r) & : r > R \end{cases}, \tag{4.2}$$

where Γ is the vortex strength, r is the distance from the vortex center and R is the radius at which the viscous, rigidly rotating vortex transitions to an ideal, inviscid vortex. Thus justification for the model is pragmatic: an inviscid model of vortices

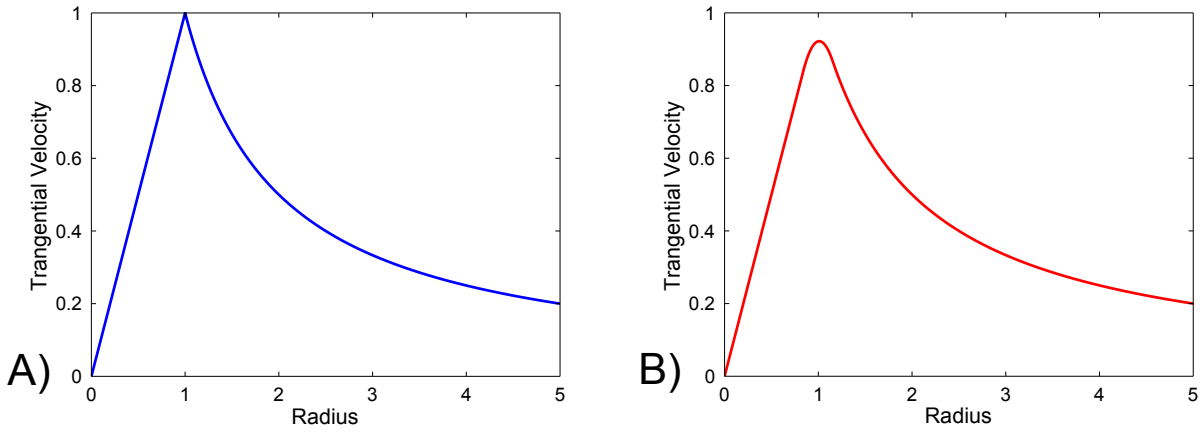


Figure 4-7: A) Tangential velocity versus radius for a standard Rankine vortex model. B) Tangential velocity versus radius for a smoothed Rankine vortex model.

works very well away from the vortex center, but due to the high (and ultimately infinite) shear rates predicted by an ideal inviscid vortex, the inviscid model cannot be used near the vortex center. However, viscous forces will tend to make the center of the vortex rotate as a rigid body (a mode in which the shear rates are zero), and thus by patching in this rigidly rotating core a reasonable model of a vortex for all radii can be produced. Fig. 4-7A plots this velocity profile against r.

This model could be sufficient on its own to produce a match filter, but when simulating this vortex's behavior the sharp change in velocity produced apparently unphysical results. This unphysicality it not particularly surprising as the sharp change is not realistic. To solve this problem, a smoothed Rankine vortex was created by blending the viscous center and the inviscid medium-to-far field with a cubic polynomial. This spline provides continuity of the function and its first derivative. The resulting tangential velocity profile is given by:

$$u_\theta(r) = \begin{cases} \Gamma r/(2\pi R^2) & : r \leq R_{viscous} \\ ar^3 + br^2 + cr + d & : R_{viscous} < r < R_{inviscid} \\ \Gamma r/(2\pi r) & : r \geq R_{inviscid} \end{cases}, \tag{4.3}$$

where $R_{viscous}$ is the radius at which the viscous model stops, $R_{inviscid}$ is the radius at which the inviscid vortex begins, and a, b, c, and d are parameters fit to match the values and derivatives of the function at $R_{viscous}$ and $R_{inviscid}$. This results in a velocity profile of the form seen in Fig. 4-7B.

4.2.2 Match Filter Construction

The match filter is then constructed by searching for a *spatial* signal matching this shape (i.e., a local vector field matching shape predicted by the model). Because the model predicts axial symmetry, the velocity vectors are converted into polar (i.e., $(r, \theta$ coordinates). Then, due to the relatively significant proportion of outliers that are present, the vectors are segmented by distance (e.g., all vectors from between 70 and 80 pixels from the candidate center are grouped together) and the median radial and tangential velocity for each segment is computed (see Fig. 4-8). These medians as a function of radius are then compared to that predicted by the smoothed Rankine vortex model, with the minimum irrotational vortex radius $R_{inviscid}$ specified by the user and the vortex strength Γ either specified by the user or optimized online to find the strength that allows for the best fit.

This is in contrast to the manner in which match filters are often used in signal processing, in which a temporal signal is matched. The distinction, however, is not a fundamental one. An example of an experimental image with associated vector field can be seen in Fig. 4-3, where to the eye the center of the vortex is quite clear.

A realization, however, when one views the vortex vector fields seen in practice, is that the central rigidly rotating core is not well resolved. With some thought this is not surprising, though it is subtle. The core of the vortex is rotational, and the particles rotate much like a rigid body. With the cross-correlation method employed here (described in §4.1) this sort of motion is difficult to detect as regions of pixels are offset and compared without rotation or shear. While one could in principle include rotation and shear in the cross correlation process, the computational burden

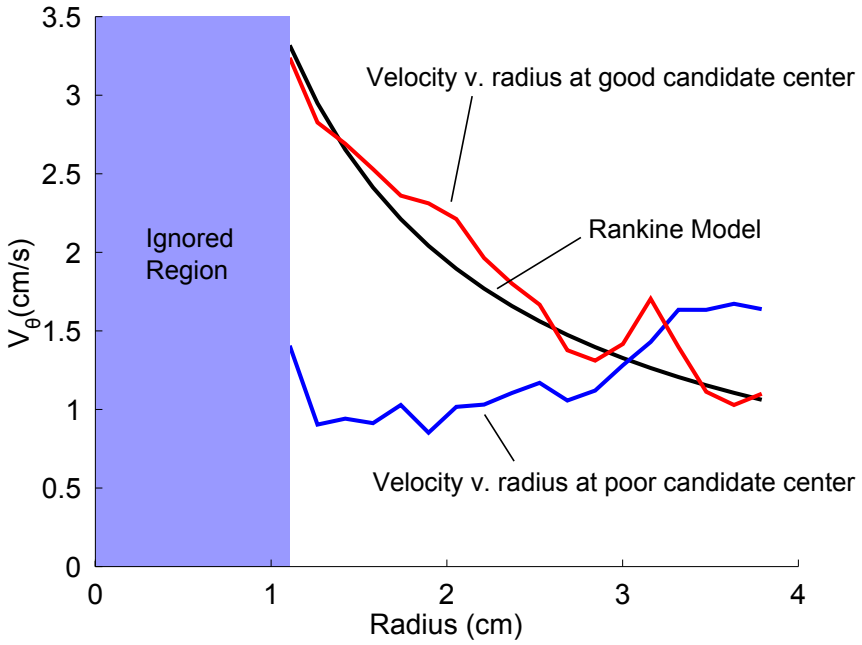


Figure 4-8: Tangential velocity versus radius for two candidate centers, one 100 pixels (1.58 cm) from the other. The black line is the structure predicted by the Rankine vortex model, and the ignored region is not studies as it is at radii less than $R_{inviscid}$.

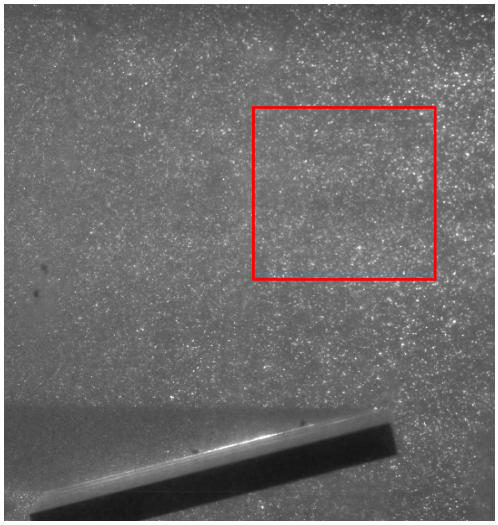


Figure 4-9: Image to which match filter was applied. Red box bounds area off candidate centers. This bounded area is limited primarily by computational considerations, and thus depending upon the available time and computational capability the search can be expanded. Given a reasonable initial guess of the vortex center, however, the region shown here is more than sufficient.

would be greatly increased as computing rotations of a region requires many floating point multiplications, whereas the method used here simply performs fewer floating point subtractions. Still, these more complex and power correlations are possible, and interesting future work could consist of attempts to make them fast enough to be used in real-time, perhaps through reasonable approximation schemes.

The cross-correlation method performed here can track the structure of the vortex at medium distance quite well (outside the rigid core, but still near enough to show a clear signal). This region does not have rotation (according to the model) but does exhibit some shear. This shear is relatively minor in the data, however, and even without including it in the cross-correlation procedure useful vector fields can be found. The performance of this match filtering can be seen in Figs. 4-9 and 4-10.

The peak at the center in this figure is clear, but not extremely "sharp", just simply picking the maximum is not the most accurate (nor the most robust) option. However, the appropriate center can be found in a robust and highly accurate manner

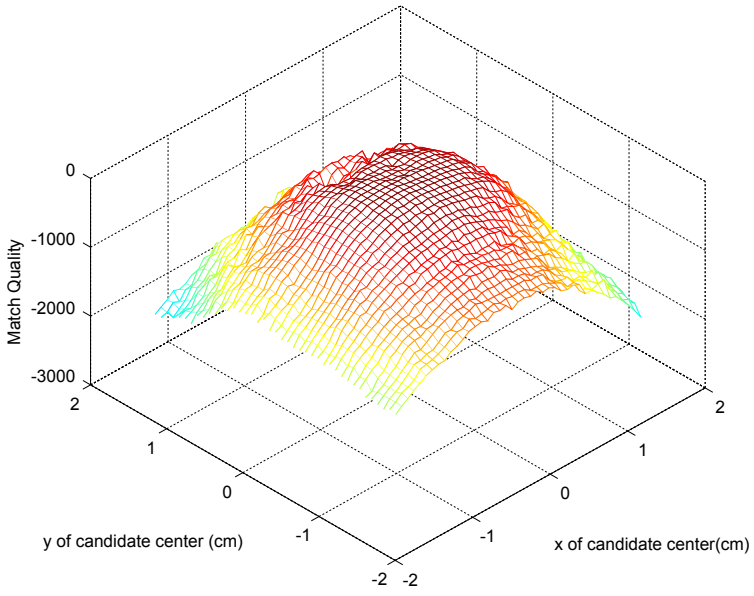


Figure 4-10: Match filter output around vortex for different candidate centers for image shown in Fig. 4-9.

with a very minor additional computational cost by fitting an elliptic paraboloid on $\mathbb{R}^2$ to the match filter returns. This paraboloid is represented algebraically by:

$$f(x,y) = p_1x^2 + p_2xy + p_3y^2 + p_4x + p_5y + p_6 = PQ(x,y), \tag{4.4}$$

where P is the row matrix $P = [p_1, p_2, p_3, p_4, p_5, p_6]$ and $Q(x,y)$ is the column matrix of monomials $Q(x,y) = [x^2, xy, y^2, x, y, 1]^T$. If there are N match filter returns $g_i = (x_i, y_i, z_i)$ where x_i and y_i are the center offsets and z_i is the filter return, the goal is to minimize:

$$\text{Error} = \sum_{n=1}^{N} \left(PQ(x_i, y_i) - g_i\right)^2. \tag{4.5}$$

This minimization is equivalent to a least-squares program, producing the best fit parameters p_i^{opt}. The location of the maximum of this this parabola (i.e., the best candidate center), refered to as $[x^{opt}, y^{opt}]$ can computed by solving the linear system:

$$\begin{bmatrix} 2p_1^{opt} & p_2^{opt} \\ p_2^{opt} & 2p_3^{opt} \end{bmatrix} \begin{bmatrix} x^{opt} \\ y^{opt} \end{bmatrix} = \begin{bmatrix} -p_4^{opt} \\ -p_5^{opt} \end{bmatrix}, \tag{4.6}$$

which can be solved analytically to give:

$$x^{opt} = \frac{p_2p_5 - 2p_3p_4}{4p_1p_3 - p_2^2} \tag{4.7}$$

$$y^{opt} = \frac{p_2p_4 - 2p_1p_5}{4p_1p_3 - p_2^2}, \tag{4.8}$$

with the superscripts on the p_i^{opt} left out for clarity. Through this procedure highly accurate fits can be found. One caveat is that if the region searched over is significantly larger than the vortex this paraboloid structure may be present over the full region of candidate centers. In that case the paraboloid much be fit to subregions small enough to exhibit this shape (i.e., aroudn the size of the vortex itself). The result is a robust, fast and precise means of tracking the center of the vortex, with the performance described in §4.2.3.

4.2.3 Tracking Performance

Using the filter outlined above, a vortex center can now be tracked through time. Fig. 4-11 shows the proposed fit from the match filter over the vector field of the image. Clearly these fits are reasonable and approximately correct, but it is also clear that not all fits are precisely at the point which, to the eye, appears to be the vortex center.

This "noise" in the fits is a result of the imperfect nature of the Rankine model (e.g., at certain times the vortex is not completely axisymmetric; likely due to interactions with the plate and the test section walls), and outliers and errors in the PIV vector field. As a result, the behavior of the system when tracking a vortex center over time exhibits a non-physical "jaggedness" as the center fit evolves through time. This can be seen in Fig. 4-12.

4.2.3.1 Luenberger Observer for Vortex Center

A solution to this, however, can be found using the same method through which an observer was constructed in §3.1. Assuming that the center behaves as an unforced, undamped, second-order system, a simple linear discrete-time model can be constructed, with the form:

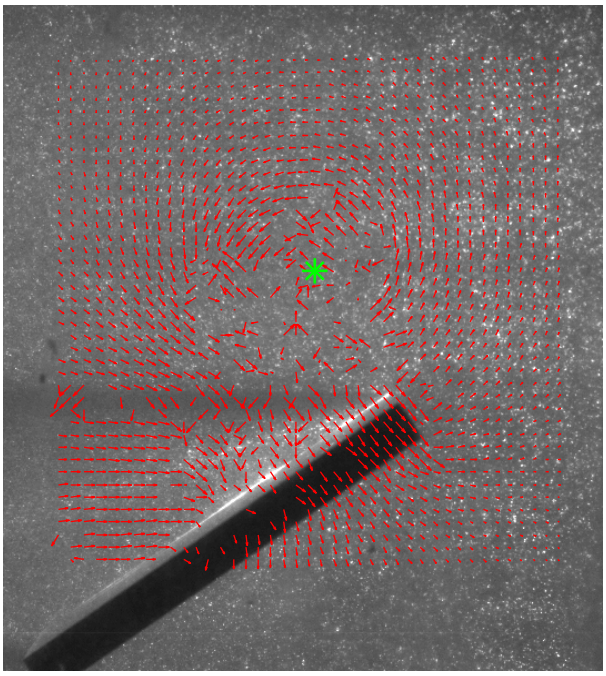
$$x_c(n+1) = x_c(n) + \dot{x}_c(n)\Delta t \tag{4.9}$$

$$y_c(n+1) = y_c(n) + \dot{y}_c(n)\Delta t \tag{4.10}$$

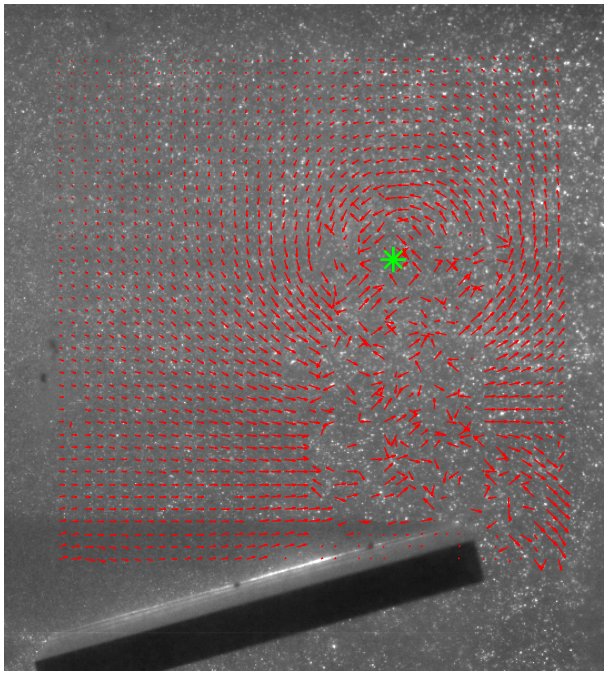
$$\dot{x}_c(n+1) = \dot{x}_c(n) \tag{4.11}$$

$$\dot{y}_c(n+1) = \dot{y}_c(n) \tag{4.12}$$

where (x_c, y_c) is the location of the center of the vortex. This model was used to construct a Luenberger observed, with a gain matrix L found by following the same procedure as that in §3.1, then converting the observer to discrete-time via a Zero-Order Hold continuous-to-discrete transformation [68].



Field 1



Field 2

Figure 4-11: Two examples of center fits (green stars) overlaid over images and associated PIV vector fields. Despite the regions of poor fit performance, the centers all appear very reasonable given the vector field.

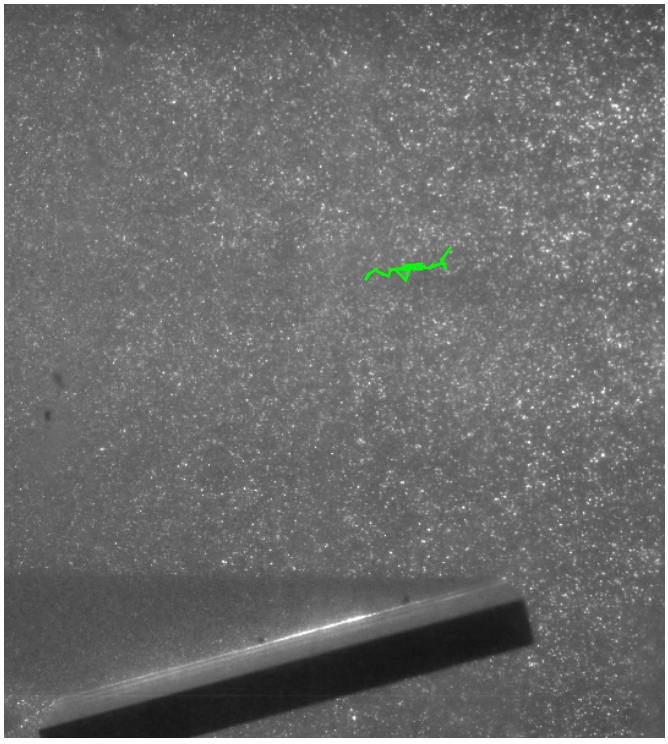


Figure 4-12: Profile of fit centers of vortex overlaid over last image in the tracked set. The true vortex should move smoothly (though not necessarily straight), but due to errors in the fits it appears to jump around. This issue is dealt with via a Luenberger observer on the vortex center.

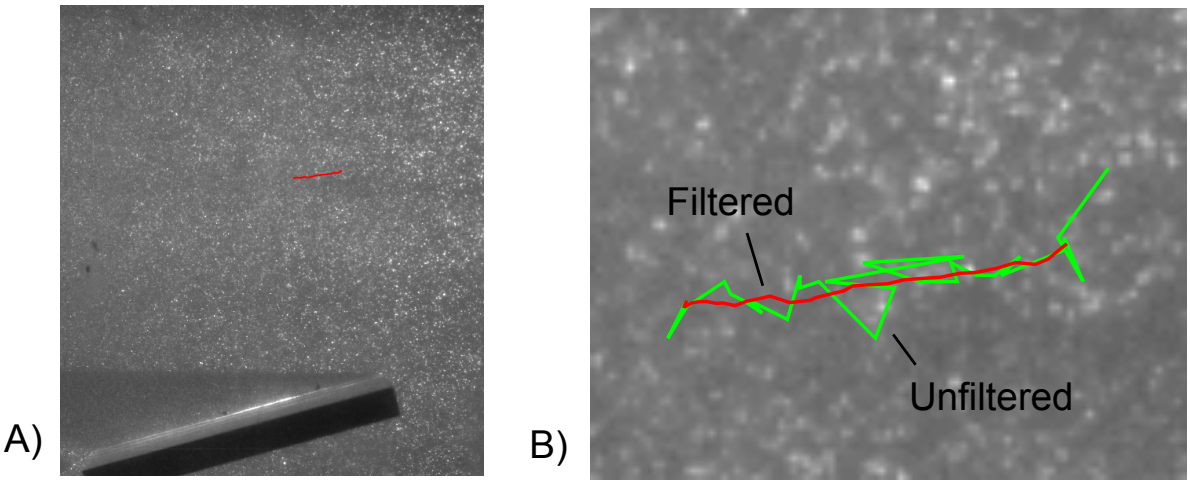


Figure 4-13: A) Profile of fit centers of vortex using a Luenberger observer overlaid over last image in the tracked set. B) A comparison of the tracking performance of the observer-based tracking versus the pure measurements shown in Fig. 4-12.

The measurements to this observer are the best fit centers from the match filter described above. With an appropriate choice of the epsilon in L more accurate and physically plausible center tracking can be performed. See Fig. 4-13A for this observer's performance, and Fig. 4-13B for a comparison to the pure measurement-based tracking seen in Fig. 4-12.

4.3 Sparsely-Sampled Tracking

The match filter presented above, while effective at tracking an isolated vortex, depends upon a relatively large number of cross correlations. In some sense, this is desirable because of the noise level of each individual vector. However, there is still the question of how to weight the different vectors that go into the estimate (i.e., design a feature that is in some sense optimal). In the section above a reasonable radius range was selected by the designer and the degree to which the measured velocity matched the predicted velocity was computed for each radius segment and averaged; should all radius be treated equally, and if not how *should* they be weighted?

Similarly, do all the correlations used in the filter above have to be performed to obtain a reasonable estimate of the center? If certain radii are not particularly informative not only could they be omitted from the match filter; the vectors used to compute those radii could be ignored. By feeding back this information into the PIV processor itself many fewer calculations would be needed and speed could be improved without sacrificing performance.

The possibility of this "closed-loop" sensing is very exciting, and this section will only present preliminary results relevant to that task. The remainder of the section will concern itself with how to formulate the sparsification and feature-selection problem in a computationally useful and principled way, how well the resulting sparse filter is able to reproduce the results of the match filter described above, and the ultimate tracking performance achieved. There is a great deal of work than can be done in this direction, and there are many promising avenues of attack. The method proposed here is certainly one such avenue, but is by no means the only one.

Unsurprisingly, this question of sparse sensing has been considered before in different contexts. A relevant literature to consider is that of Compressed Sensing [24]. Compressed Sensing takes many forms, but when dealing with noisy measurements it is primarily concerned with obtaining a sparse solution to a least-squares problem. It does this by approximating the goal of sparsity with the L1-norm (as opposed to the L0-norm). The L1-norm does not directly encode sparsity (directly optimizing the L0-norm is NP-Hard in the relevant cases [64]), but does tend to produce somewhat sparse solutions. For some situations Compressed Sensing can allow the user to compute significantly fewer vectors through PIV while sacrificing little in terms of detection accuracy.

The basic formulation for compressed sensing is the L1-regularized least squares problem. Eq. (4.13) below gives the mathematical description of this problem:

$$x = \underset{x}{\operatorname{argmax}} \left(||Ax - y||_2^2 + \lambda ||x||_1 \right), \tag{4.13}$$

where $||\cdot||_2$ is the L2-norm (i.e., euclidean norm), $||\cdot||_1$ is the L1-norm (i.e., sum of the absolute values), A and y are from data, x is the result of the optimization and λ is a parameter set by the user to control the balance between sparsity and accuracy. This is a more complicated problem to solve than traditional least-squares, and requires iteration to come to a solution, but it remains convex [46]. Therefore, it can be solved efficiently and the global optimum can be obtained.

This methodology was used to determine sparse center weights for the vortex center-tracking task. The vector x contains the weights given to each radius when trying to compute how good the proposed center is. The matrix A contains the fit performance at each radius for all proposed centers, and y contains the fit results from the dense computation. The goal then is to find an x which is able to reproduce the results presented above (i.e., the densely sampled results) while being as sparse as possible (i.e., contain as many zeros as possible without seriously compromising performance). Sparsity is desirable because if a weight in x is zero the associated cross correlations need not be performed, greatly reducing the computation burden. With this in place either faster frame rates could be obtained or more vectors could be computed in the same amount of time.

After sparsification 60% fewer cross-correlations had to be performed to track the center, as compared to the dense sampling case[2]. The tracking performance did not suffer significantly despite this significant reduction in the number of measurements, as can be seen in Fig. 4-14 and Fig. 4-15. Extending these ideas to other flow situations and configurations is an interesting direction for future research, and one which could result is significantly improved performance in future PIV experiments.

[2]The sparsification achieved for a single guessed center was higher, but because computations no longer needed for a certain center may be needed for an adjacent center the overall sparsification was approximately 60%.

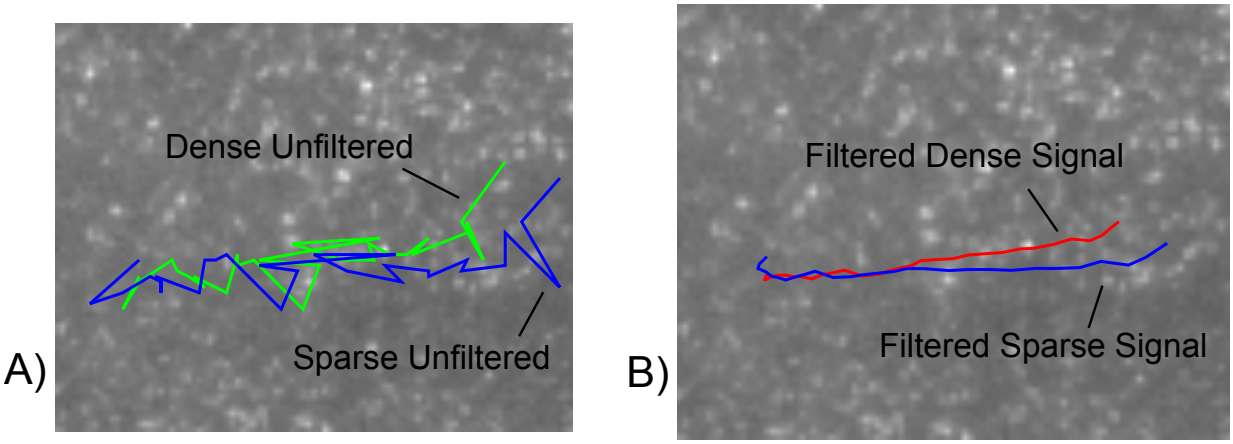


Figure 4-14: A) Comparison of the measured centers for both the dense and the sparse center trackers. B) A comparison of the tracking performance when using an observer of the dense and sparse center trackers.

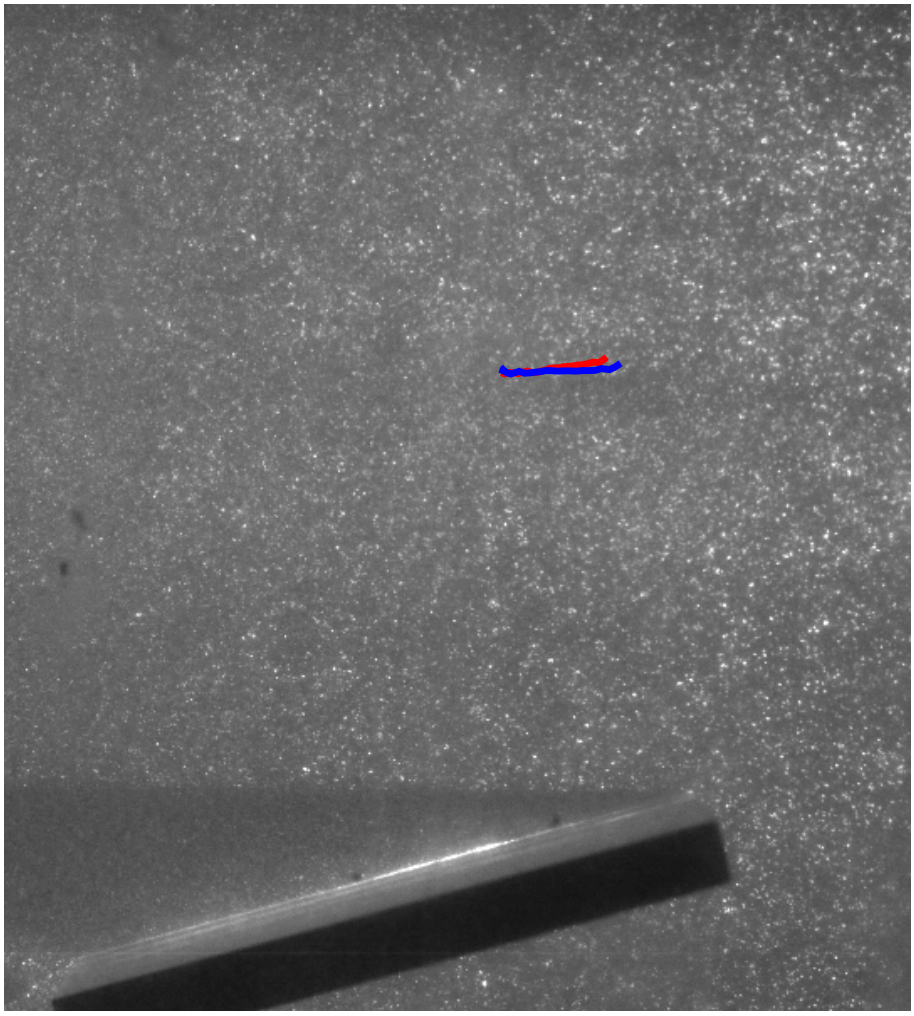


Figure 4-15: "Zoomed-out" view of the center tracking performance of the dense and sparse trackers. Clearly the two techniques produce similar results on the scale of the full vortex and the image.

Chapter 5

PIV-Enabled Control

The previous two chapters have introduced both a method for optimizing fluid-body systems without the need for an explicit model, and a procedure for quickly performing PIV and extracting from it relevant information about the features of a flow. In this chapter these two techniques are combined and used to greatly reduce the magnitude of the perturbation caused by the HCP passing through a vortex.

The experimental setup is modified to include a servo-driven wing (referred to as the "perturbation wing") upstream from the HCP, as seen in Fig. 5-1. By moving this wing, a vortex can be produced upstream of the HCP, after which the vortex will advect downstream to the HCP. This wing is controlled by the experimenter, and thus the timing and size of the perturbation can be accurately controlled. Also, as the vortex is produced at a known time the performance of the PIV filter (e.g., the delay before PIV recognizes a vortex is present and how often false positives or false negatives occur) can be measured. If the system performs well in this controlled context it can also be employed in situations where the time of the vortex shedding is unknown, such as in the wake of several cylinders which can produce a chaotic vortex street, or in the wake of another body not directly controlled (e.g., a human or animal controlled system).

Two control experiments were studied, both of which involved minimizing the

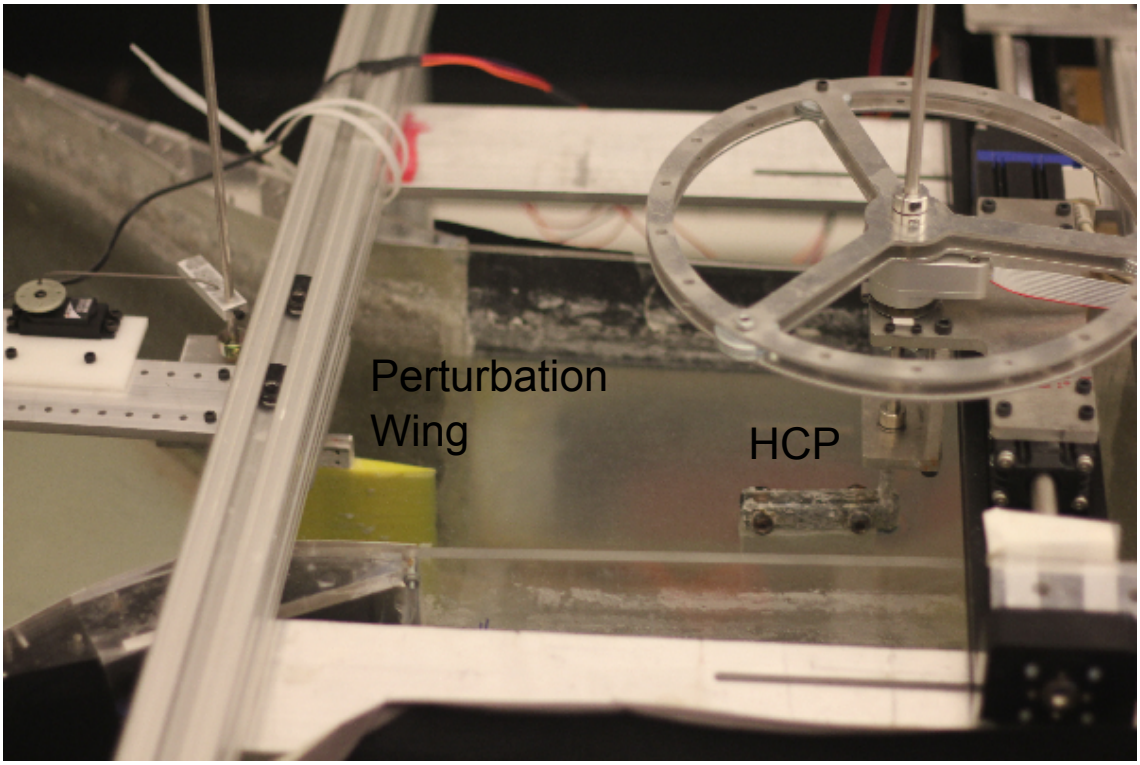


Figure 5-1: Picture of the test section with the servo-driven wing (yellow) in place upstream of the HCP.

deviation of the HCP from the desired balancing configuratino as it passed through a vortex generated by the perturbation wing. The first of these tasks involved minimizing the θ deviation via trajectory optimization, while the second minimized the cart position (y) deviation using the same procedure. In both situations the methodology presented in §3 was followed. First, an LQR controller was used to balance in the presence of a vortex. A model was built around the mean of these perturbation trajectories, and a compensatory trajectory and input were found and added to the nominal (in LQR's case, the nominal is a simple zero trajectory). When this was executed the deviation was significantly smaller, as discussed in § 5.2 and § 5.3. The PIV filtering used to detect the vortex was a basic match filter, designed to provide a clear signal with minimal latency. The design and performance of the match filter are characterized in the following section.

5.1 Real-Time PIV Filtering Performance

The goal of the real-time filter presented in this section is to identify a vortex as early as possible, ideally just as it enters the frame. Because of this constraint the center tracking filter presented in the previous chapter is not necessarily the highest performing. Instead, to minimize latency, a characteristic shift in the cross-flow velocity (i.e., a shift away from the mean flow) is matched to the measured PIV velocity field. When this characteristic shift first appears it is taken as a signal that a vortex is entering the frame.

An example of the smooth-flow (i.e., no vortex) flow field can be seen in Fig. 5-2A. The match filter should return a small value (poor match) in this operating mode. In contrast, Fig. 5-2B shows a characteristic velocity field when a vortex is about to enter. The clear velocity shift near the top of the frame is the "giveaway," and it is here that we will focus.

The match filter operates by comparing the measured vector field it is given in

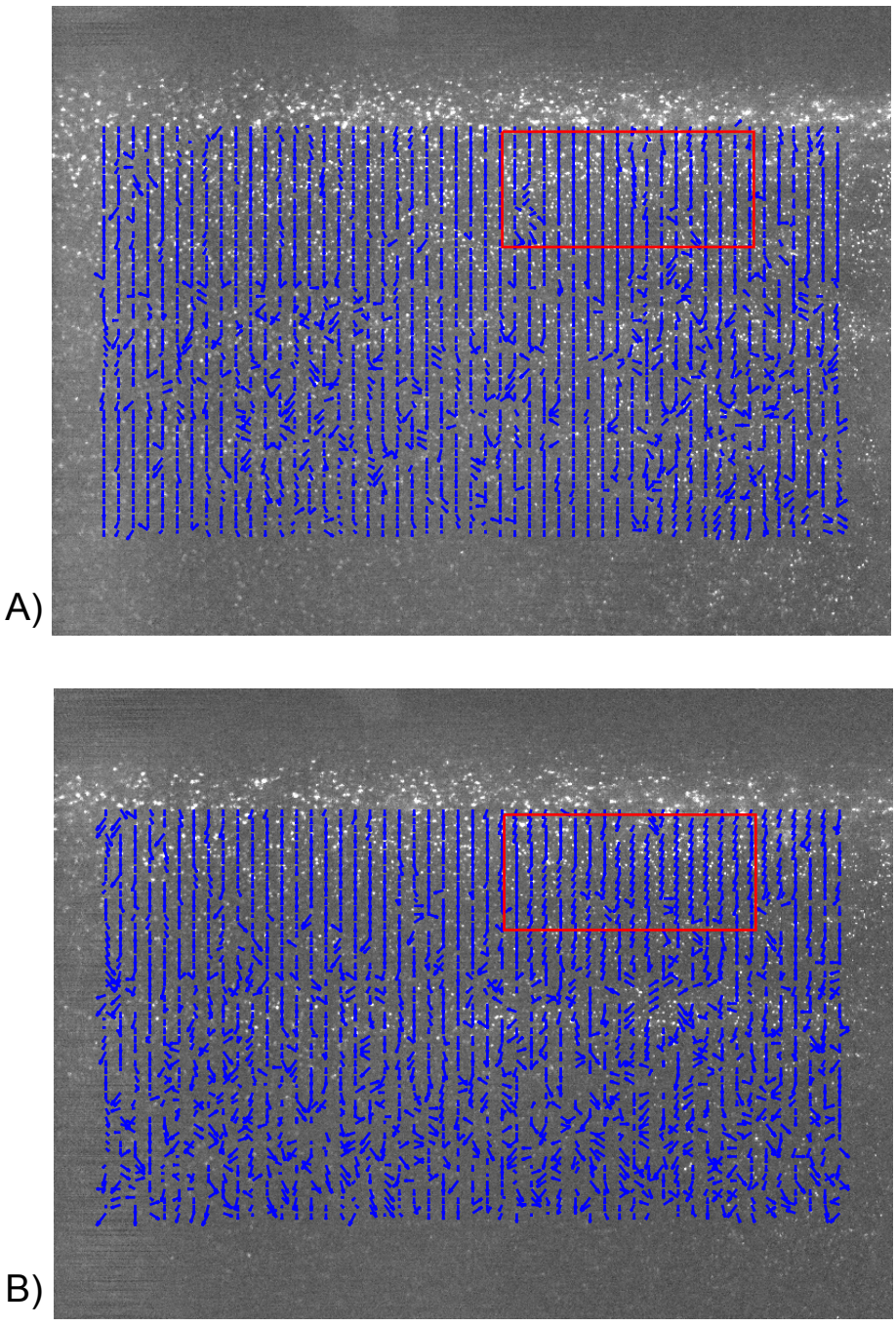


Figure 5-2: A) Sample vector field from a flow (flow direction from top to bottom) undisturbed by a vortex. Note that the vectors are pointed straight down. B) Sample vector field from a flow (flow direction from top to bottom) with a a vortex entering from the top right. Note that the vectors are pointing down and to the left.

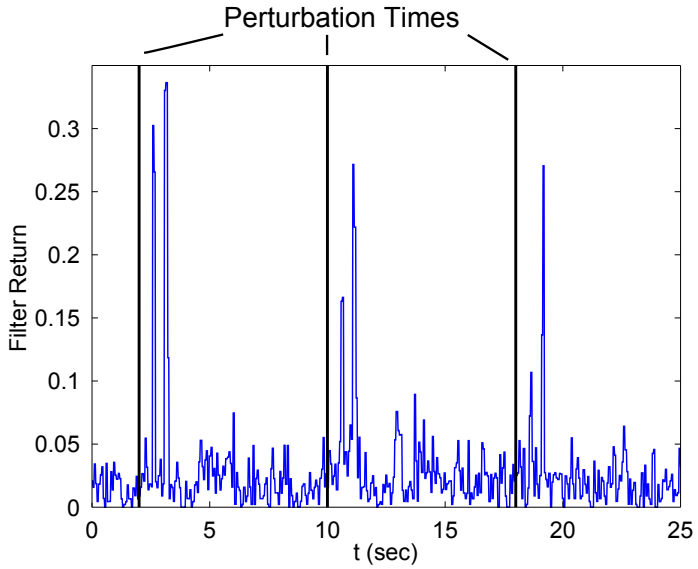


Figure 5-3: A time history of the match filter output as vortices are produced via the perturbation wing. The times at which the wing starts moving (i.e., the beginning of perturbation and the creation of a vortex) are shown as black vertical lines. The detection delay to the first of the double peaks following a perturbation is approximately 0.6 seconds – it is this peak we wish to detect. The third perturbation shows why detection is not perfect: the first peak is not significantly higher than the noise, thus as the threshold approaches this level false positives will begin to occur. If the threshold is set well above the noise this peak will not be located, and thus false negatives will occur. This is the trade-off that must be considered, and that is represented in Fig. 5-4.

real-time by PIV and comparing it to a vector field designed to be similar that in Fig. 5-2B[1]. This comparison is performed by taking the difference between the desired vector field and the measured vector field, and counting the number of vectors that differ by only one or zero pixels. The proportion of vectors which differ by this amount is computed, and this ratio (between 0 and 1) is considered the output of the filter. See Fig. 5-3 to see a trace of this output compared to times of perturbations. As the figure shows, the filter responds to the vortex being generated approximately 0.6 seconds after the perturbation begins.

[1]The filter is not based on any one "detection" case, but rather on the average behavior of several

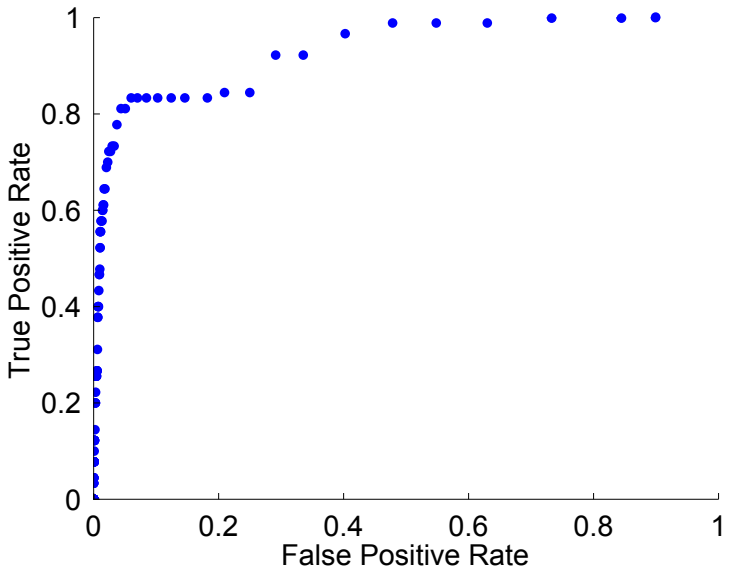


Figure 5-4: Receiver Operating Characteristic curve for the match filter presented in this section. By change the threshold at which a detection occurs the designer can move along this curve and select the appropriate trade-off for the problem at hand.

To decide when the filter's output reaches a level sufficient to be considered a "detection" a threshold can be chosen. After a detection no second detection is possible for a certain amount of time (set at 2.5 seconds) to prevent repeated detections close together from the same vortex. The appropriate threshold depends on the relative costs of false positives and false negatives. Thus, it must be chosen by the designer based upon an understanding of the system. To aid in this selection a Receiver Operating Characteristic (ROC) curve can be constructed [26]. An ROC curve shows how the proportion of false positives and false negatives evolve as the threshold is changed, and it makes the trade-offs clear. It may also be used as a filter design tool as optimizing the ROC to provide the best available discrimination can be seen as the goal of the match filter design. Fig. 5-4 shows the ROC curve for the θ compensation task described below.

5.2 Deviation in θ Compensation

For the experiments in this section a vortex was generated 15 cm ahead of the HCP's leading edge (i.e., the trailing edge of the perturbation wing was 15 cm in front of the leading edge of the HCP at the upright). For each perturbation the perturbation wing was driven from an angle of attack of 0° to approximately 30° then back to 0° with a triangle wave profile. This process took 0.8 seconds, and from the end of one perturbation to the beginning of the next the HCP was allowed to settle for 7.2 seconds to allow its return to its approximate steady state behavior. The size and duration of the perturbation was selected to produce a deviation of the appropriate size when stabilized by the LQR controller (i.e., significant, but not so large as to make the system regularly exceed its travel[2]).

Fig. 5-5 and Fig. 5-6 show sample trajectories of the LQR controller being perturbed by a vortex generated in the manner described above. These trajectories are synchronized to the time at which PIV could recognize the vortex (0.6 seconds after the perturbation wing *starts* moving, or 0.2 seconds before it finishes its profile). Note that some of the trajectories appear to begin a small dip around $t = 0$, while others do not begin to respond until well after, at approximately 0.5 seconds. Thus, some trajectories feel the effect of the moving perturbation wing at the same time PIV detects it[3], while others fail to see a clear signal until 0.5 seconds later (see Fig. 5-6).

As a point of comparison for PIV an optimized controller (using the same optimization scheme as the PIV controller as detailed in §3) which detects the vortex 0.5 seconds after PIV would is developed and tested as well. This delay of 0.5 seconds is very optimistic as to avoid false positives additional latency would have to be introduced to distinguish an actual vortex from the variation witnessed in the normal

[2]Travel in y (i.e., cart position) is in practice the most strict constraint in the system, and when the system fails it is because the cart "runs into the wall."

[3]The latency of the PIV system: capturing the images, moving them from the camera to the computer and performing the processing results in a delay of approximately 100ms that could be reduced with hardware changes, but this chapter will refer to the time at which the controller itself is aware as the time of PIV detection.

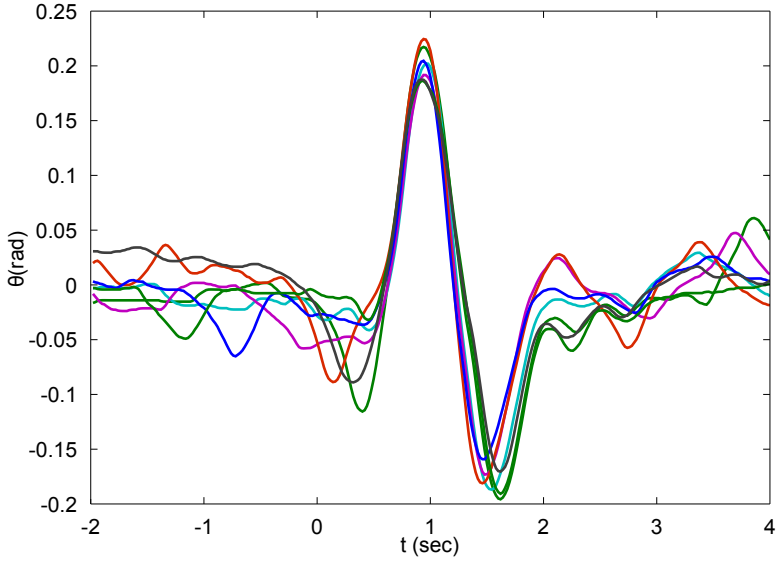


Figure 5-5: A sample of LQR-controller perturbation trajectories in θ. Note that given the variation during normal operation, detection is difficult until the vortex has significantly perturbed the system. See Fig. 5-6 for a close up on the time at which the perturbation is just beginning.

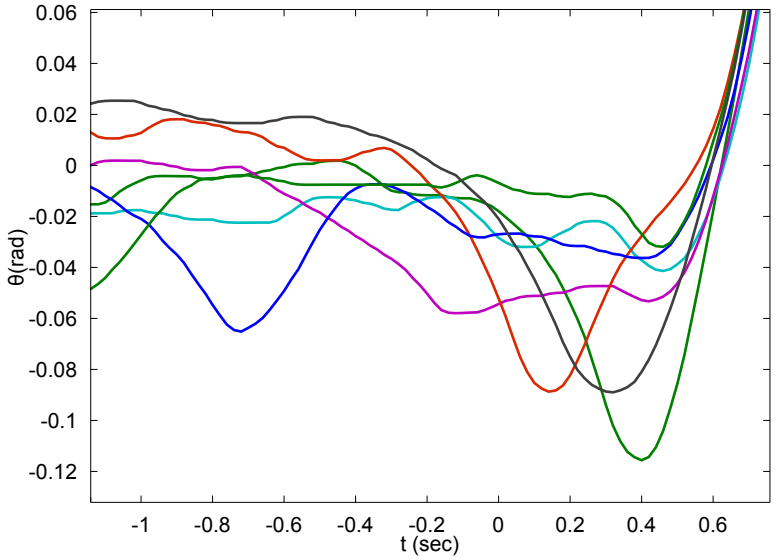


Figure 5-6: A sample of LQR-controller perturbation trajectories in θ. Note that given the variation during normal operation, detection is difficult until the vortex has significantly perturbed the system.

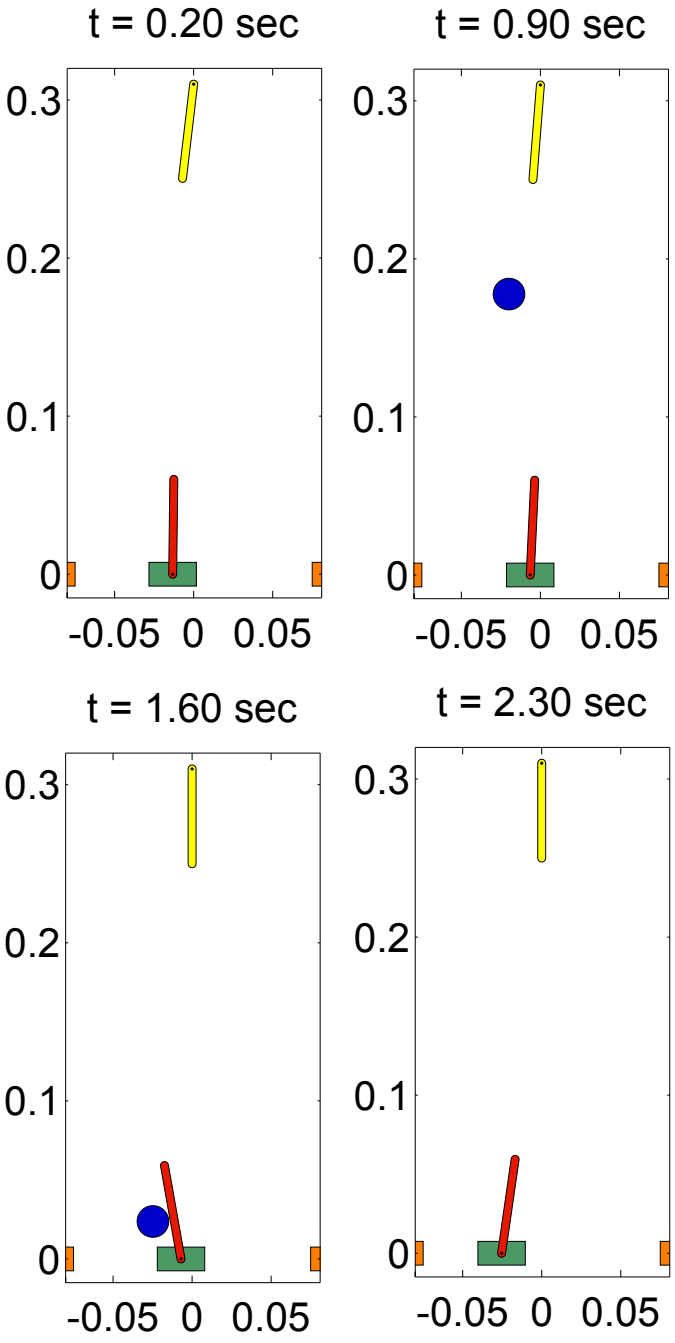


Figure 5-7: A series of snapshots of the LQR-controller responding to vortex perturbations. Compare to Fig. 5-10 and Fig. 5-13 to see the improvement achieved via the PIV-enabled controller when attempting to minimize θ (pole) and y (cart) deviation respectively.

balancing operation of the HCP. This controller was given awareness of the vortex by allowing it to "cheat." In other words, after beginning to move the servo an appropriate delay was introduced (i.e., the mean time to PIV detection plus 0.5 seconds) and a "detection" signal was given to the controller. This removed any complexities involving possible false positives or negatives when evaluating this competing controller. The comparison of a controller using PIV, a controller using a delayed detection signal, and the LQR controller is presented in Fig. 5-8, and again in Fig. 5-9 showing the repeatability of the trajectories via the measured standard deviation as well.

To give a more clear picture of what individual trajectories look like under the PIV controller, see Fig. 5-11. In this figure three individual trajectories are plotted, demonstrating the degree of variability in performance and the gap between LQR and PIV performance. As is quite clear, while there is reasonable inter-trial variation in behavior and performance the PIV controller outperforms handily.

5.3 Deviation in y Compensation

Similar to the previous section, a vortex was again generated 15 cm ahead of the HCP's leading edge (i.e., the trailing edge of the perturbation wing was 15 cm in front of the leading edge of the HCP at the upright). For each perturbation the perturbation wing was driven from an angle of attack of 0° to approximately 40°, then back to 0° in a triangle wave profile. This process took 1.0 seconds, and from the end of one perturbation to the beginning of the next the HCP was allowed to settle for 7.0 seconds to allow its return to its approximate steady state behavior. The size and duration of the perturbation is larger than that used for the θ tests, and was again sized to achieve a significant disturbance without causing an excessive number trajectories to exceed the bounds on y.

These trajectories are synchronized to the time at which PIV could recognize the

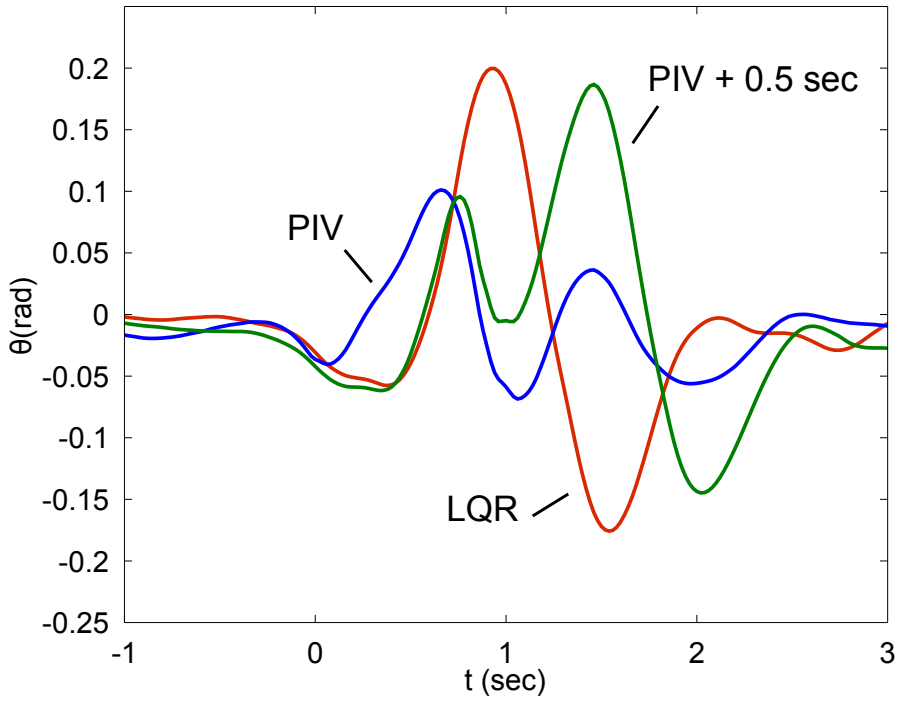


Figure 5-8: Comparison of mean θ perturbation trajectories for the LQR controller, a controller recognizing the vortex via PIV and a controller whose awareness of the vortex is delayed 0.5 seconds after PIV. Trajectories are synchronized to the time PIV would detect detect the vortex (i.e., t = 0 above is when PIV would first recognize a vortex was incoming). While the mean trajectory begins responding before this time, the effect is very subtle and does not robustly appear in individual trajectories (see Fig. 5-5). A delay of 0.5 seconds is an optimistic detection delay in practice.

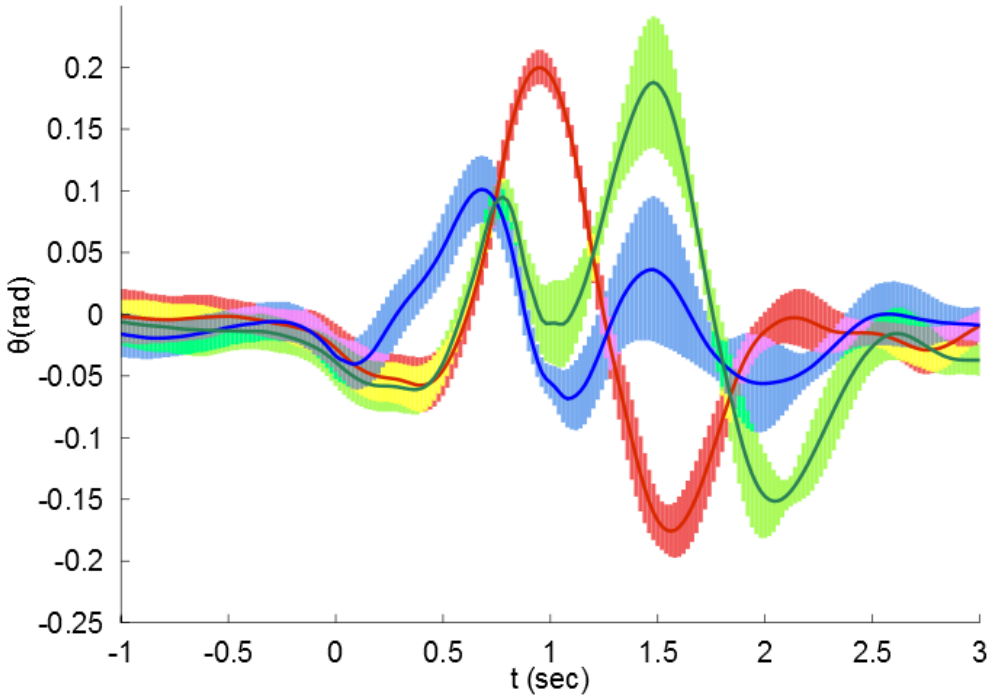


Figure 5-9: The same comparison as in Fig. 5-8, but with shading indicating plus/minus one standard deviation around the mean.

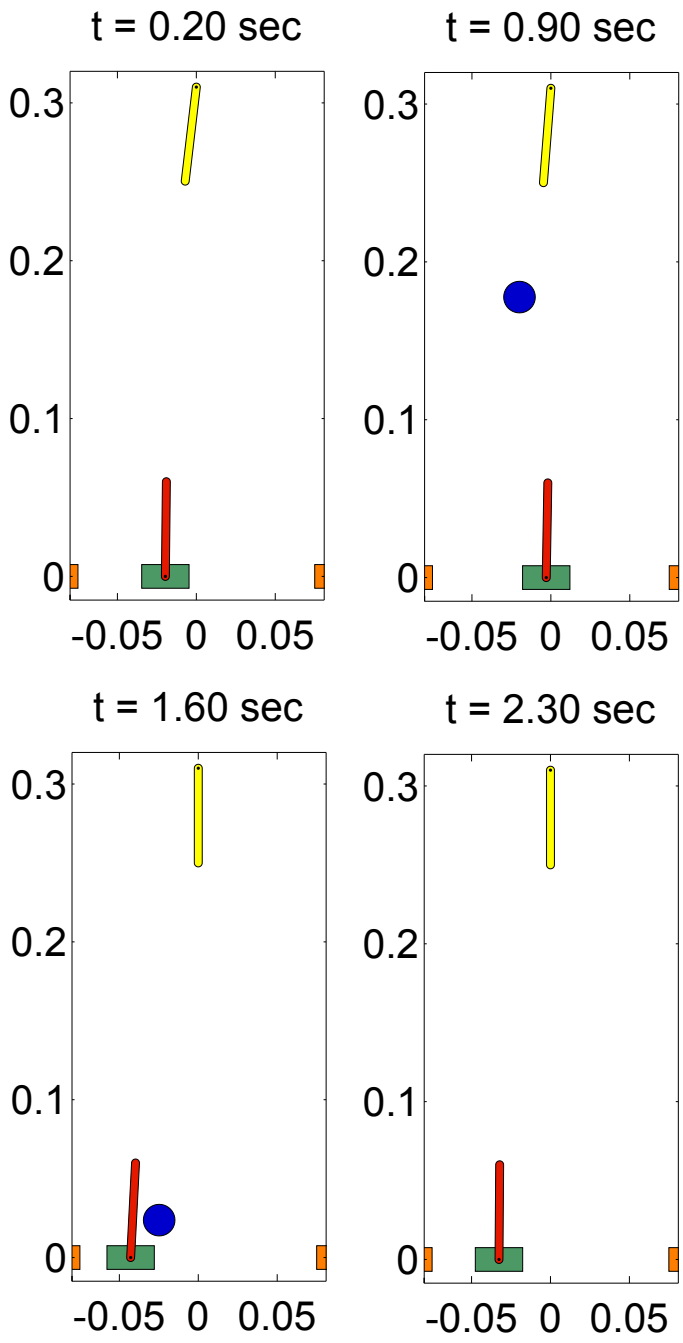


Figure 5-10: A series of snapshots of the PIV-enabled controller responding to vortex perturbations and attempting to minimize θ deviation. Contrast maximum angular pole deviation with the LQR controller in Fig. 5-7.

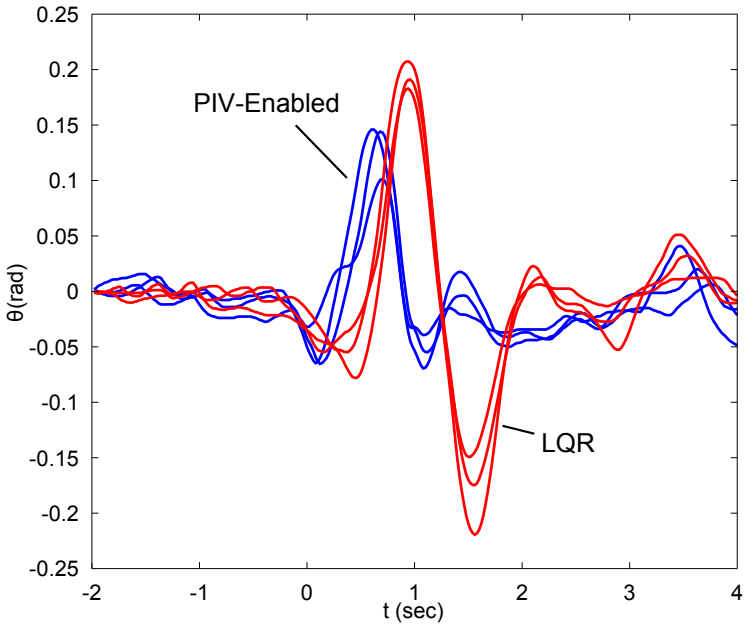


Figure 5-11: Three θ trajectories for LQR and PIV-Enabled controllers. The improvement in performance is clear. In both cases, the three trajectories closest to $\theta = 0$ at the start of the window were chosen. The fact that despite the clear variation in initial conditions the PIV-Enabled controller reliably outperforms LQR demonstrates the robustness of the corrective maneuver.

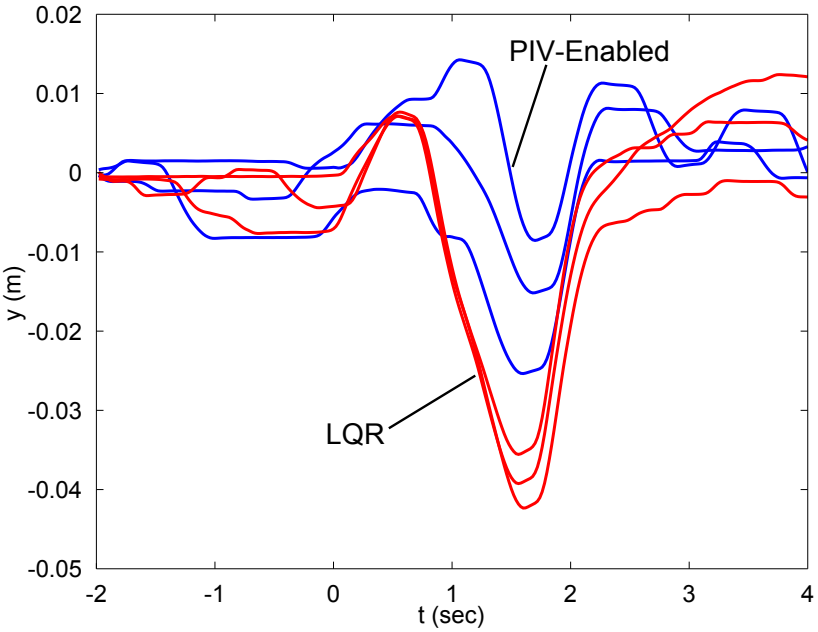


Figure 5-12: Three y trajectories for LQR and PIV-enabled controllers. The improvement in performance is clear. In both cases, the three trajectories closest to $y = 0$ at the start of the window were chosen. The fact that despite the clear variation in initial conditions the PIV-Enabled controller reliably outperforms LQR demonstrates the robustness of the corrective maneuver.

vortex (0.6 seconds after the perturbation wing *starts* moving, or 0.4 seconds before it finishes its profile). Note that some of the trajectories appear to dip just around $t = 0$, while others do not respond until well after, at approximately 0.5 seconds (as was the case in the previous section). Fig. 5-12 shows several sample trajectories for both the PIV-Enabled controller and LQR. Again, the variance is nontrivial but performance is robustly better.

Another useful perspective on the the advantage provided by the PIV-enabled controller over both LQR and a controller with only local awareness of the vortex is presented in Fig. 5-14. In this figure histograms of maximum y deviation for the different controllers are plotted. Fig. 5-14A shows a comparison between deviation between PIV-enabled and LQR, while Fig. 5-14B compares PIV-enabled and the

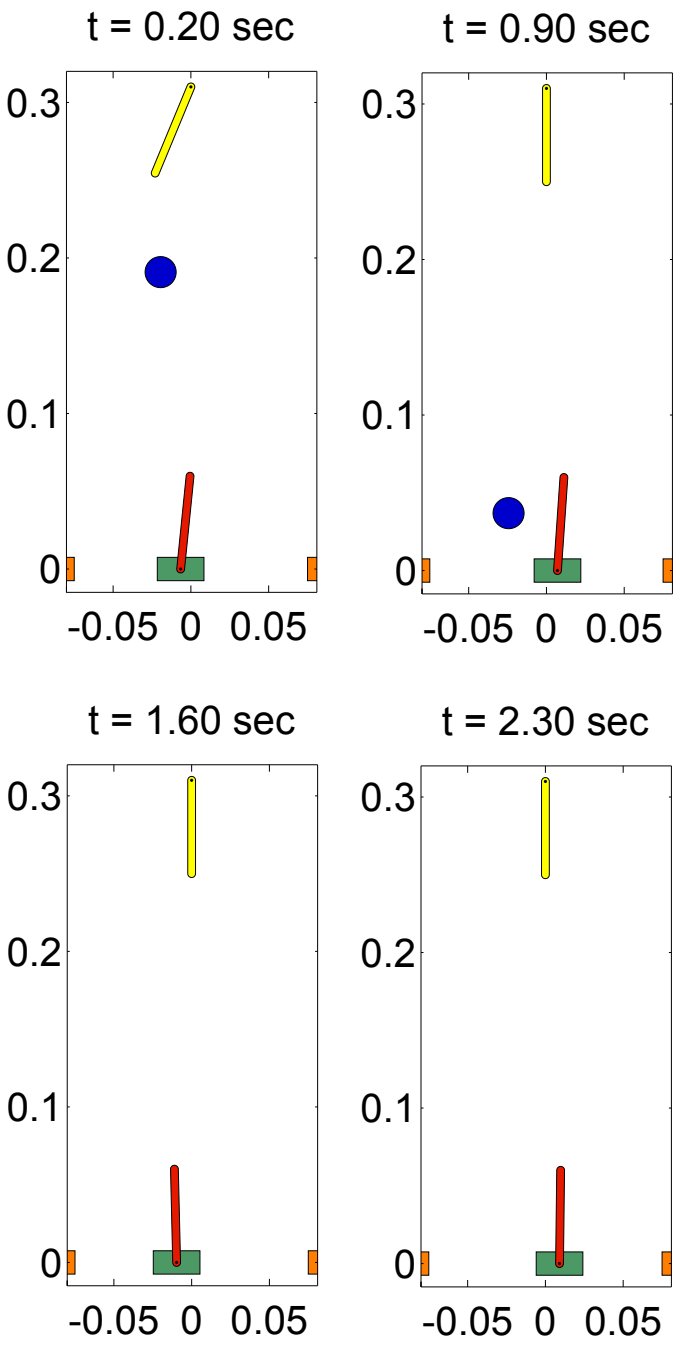


Figure 5-13: A series of snapshots of the PIV-enabled controller responding to vortex perturbations and attempting to minimize y (i.e., cart) deviation. Contrast maximum cart deviation with the LQR controller in Fig. 5-7.

delayed controller. In both cases PIV allows for significantly improved performance over both alternatives. The delayed controller outperforms LQR but underperforms the PIV-enabled case, as would be expected.

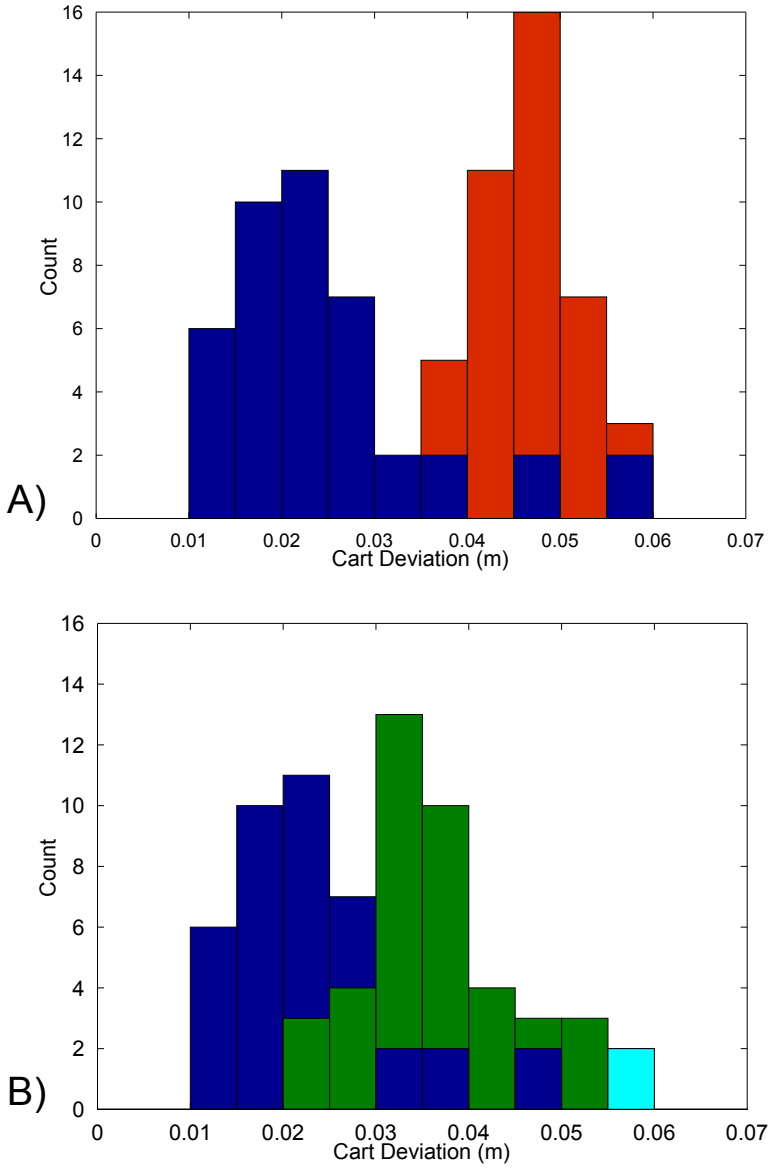


Figure 5-14: Histogram of maximum deviations in cart-position after a vortex perturbation for the PIV-enabled controller (blue) comapred against the LQR controller (red), and the delayed controller (green). A) PIV vs. LQR: The performance is dramatically improved by the use of a feed-forward controller with vortex detection enabled by PIV. B) PIV vs. Delayed: The cyan bar signifies an equal number of counts for both. A delay of 0.5 seconds after PIV is again a very optimistic number for when the presence of a vortex can be identified by local sensing alone. The degrading level of performance when this delay is present demonstrates the importance of recognizing the vortex before it is encountered.

Chapter 6

Conclusion

This thesis has presented two complementary methods of controlling fluid-body systems, and shown how the can be combined to offer improved performance and a better understanding of the nature of the control problems at hand. The method presented in Chapter 3 dealt with designing high-performance controllers for underactuated systems without good models. Chapter 4 introduced the real-time PIV system used, and the basic filtering methodology through which vector fields could be converted into low-dimensional information useful to control. Finally, Chapter 5 introduced these two techniques working together to improve performance on a complex, underactuated fluid system interacting with fluid disturbances. These results offer a "proof of concept" about how to usefully couple data-driven control design techniques with real-time PIV to achieve higher-performance.

The remainder of this chapter will concern itself with discussing the interpretation of the control results presented earlier in the thesis, and suggesting avenues of possibly fruitful further research. A great many interesting problems suggest themselves now that the system is fully operational and offering useful results. The next three sections will discuss exciting possibilities, organized by problem to which they are most related: 1) Data-Driven Control Without PIV, 2) Real-Time PIV Filtering and 3) PIV-Enabled Control.

6.1 Data-Driven Control

As discussed in the section of previous work (see §1.2.1) a significant amount of effort has been spent on designing controllers for fluid-body systems, with different techniques appropriate to different situations. The methodology presented in Chapter 3 is well-suited to the HCP system presented here, and it was used in a relatively general-purpose way without significant engineering insight being required. It would be interesting, however, to evaluate the technique with more specialized models and policy classes, blending it with the work of Lupashin and D'Andrea [57]. For a system with more states and actuators, it could be that a subspace of the state space is relatively easy to model and control, and thus more model-based methods could be effective to regulate some of the dynamics, while the data-driven iterative method of this thesis could compensate for those parts too difficult to represent conveniently as a model or "squash" with high-gain control.

An alternative direction is to study how policy parameterizations more structured than the discrete-time state, input and gain trajectories used here would perform. One issue that can appear when using these vary naive policy classes is that over the course of several iterations high-frequency content appears in the signals, possibly degrading performance. In the this thesis this was not a problem when the updates were made relatively small, but in general the issue could appear, particularly if many iterations are required. Therefore, a limited dimensional policy class that precludes high-frequency content (e.g., a spline[1]) could be effective, as the optimization could be performed over sets other than discrete-time linear signals. In many cases this will prevent the use of very computationally efficient formulations such the quadratic programs used here, but in practice the problems can often be reasonably solved for at least a local solution.

[1] A truncated fourier representation could seem natural here as well, though in practice I have found splines to be better behaved and more useful.

6.2 Real-Time PIV Filtering

There are many possible directions of research in real-time filtering for PIV. These can be considered in three sections, organized as follows:

6.2.1 Improvements to the real-time PIV itself

6.2.2 Improvements to the extraction of relevant information from a PIV vector field

6.2.3 Closing the loop between information from vector field and measurements of the vector-field

6.2.1 Improving Real-Time PIV

There is a wealth of literature about performing PIV as efficiently as possible (see §1.2.2), with frequency domain methods a common approach which I have yet to investigate. The implementation is significantly more complicated than that used for real-time PIV in this thesis, but many aspects of the algorithm can still be parallelized and further speed up could result. These could allow for denser vector fields and less latency before the field is available for filtering. Also, taking into account rotation and shear of the fluid could result in better matches and more accurate vector fields, but at the cost of reduced speed. Dealing with this trade-off is an interesting research topic, and in many situations it could be worthwhile to perform the extra computation.

Another interesting possibility is to modify the PIV algorithm to make it more specialized for the filtering task at hand. If a certain match-filter is to be applied to the output, the implementation of PIV can be optimized to obtain the necessary information as quickly as possible with minimal computation cost. For example, if a match filter largely ignores an aspect or region of the vector field, it may be more efficient to simply not compute it. One must take care, however, about shrinking the number of comparisons performed between subregions, as fewer comparisons makes

it more likely at match between regions will happen by chance rather than because of actual match quality, and thus more false positives may result.

6.2.2 Improving Extraction

There are many possible means of designing effective filters on the vector field beyond the match filter methods used here. The filters used in this thesis were constructed based upon models of the fluid (i.e., a Rankine vortex) and by hand. Obtaining features directly from data is an exciting possibility, and tools exist to make it possible. Methods such as Proper Orthogonal Decomposition (POD) allow relevant features to be extracted automatically from data, so long as the can be represented in a linear fashion. Other automatic feature learning tools exist, and some are well-suited to control applications, such as balanced truncation [84].

It is also exciting to consider further the possibility of designing sparse features that look at only a limited subset of the available PIV vectors, and at means of blending measurements that are taken in some "optimal" way. As seen in §4.3 sparse features can in certain cases largely reproduce the performance of a dense feature. Because of this it is sometimes possible to greatly decrease the number of cross-correlations necessary by not generating the PIV vectors upon which the sparse feature does not depend. This decreases the computational burden while maintaining filter performance. Furthermore, incorporating the measurements that are taken (e.g., giving more weight to lower variance measurements) is another interesting, rich problem to be explored. The sparsification technique presented in §4.3 is a good starting point, but superior classification and greater sparsification could certainly be achieved with further research.

6.2.3 Closed-Loop Sensing

The most exciting potential improvement to PIV performance, however, is to couple the feature tracking component with the PIV processing itself. For example, when

tracking a vortex center the current estimate of the vortex location could be sent to the PIV system. The PIV system could then perform denser correlations in the regions that are likely to be the most informative. This is related to the compressed sensing work referenced in §6.2.2, as the information provided by the current estimate of the fluid state could be combined with the L1-norm sparsification procedure to generate sparse, principled sampling distributions optimized for the current fluid state.

The advantages of this are obviously significant, and a number of interesting problems are raised. An example is how to select an efficient sparse sampling distribution while maintaining robustness to errors in the estimate. If the vortex is not where the current estimate places it, how well can the filter correct its error, and at what cost? These are interesting questions, and ones only now being raised because of the feasibility of closed-loop PIV-enabled control.

6.3 PIV-Enabled Control

PIV-enabled control is an exciting topic, and one for which the necessary computational capabilities are just coming on line. Its most obvious applicability to problems of practical engineering comes in the form of a "scaffold" towards deployable controllers that do not ultimately depend on PIV measurements to perform. In this context PIV can be performed in the lab to provide the controller with a fully-observable problem in which control design is much easier and more natural. The question then involves discovering how much of the information relevant for control can be obtained via local sensors such as anemometers and pressure taps.

By first making use of a PIV-enabled controller, the control designer can directly respond to flow disturbances without depending upon local sensing. Once a high-performing controller has been designed in this more intuitive and "easy" space, local sensors can be correlated with the actions of the PIV controller. There is a possibility that a structure or signal in the flow that was otherwise difficult to discover the

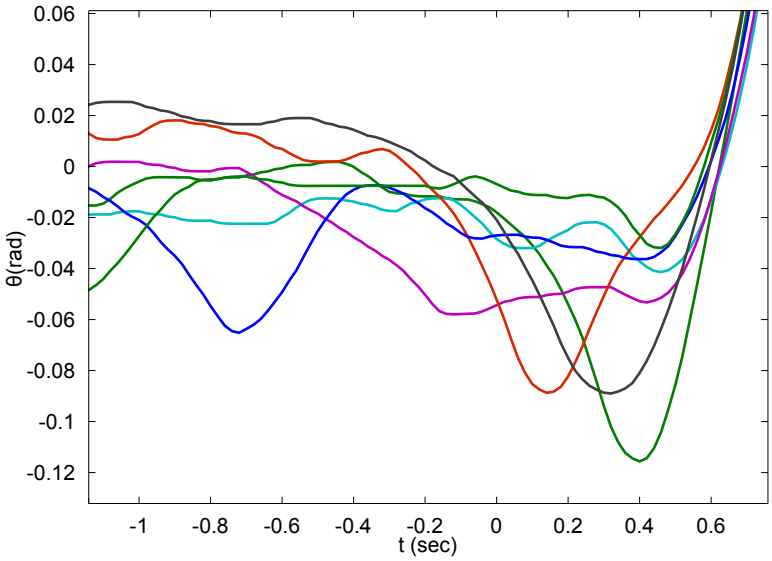


Figure 6-1: A sample of LQR-controller perturbation trajectories in θ. Note that while the variation during normal operation makes detection difficult until the vortex has significantly perturbed the system, some trajectories do see witness an effect in the flow earlier than would otherwise be expected. Identifying the cause of this disturbance, and characterizing it more robustly could result in improved performance without PIV.

importance of could be found by connecting it with the PIV controller's actions. For example, in the PIV-enabled control task presented here there is occasionally a signal in the θ measurement that arrives at the same time as PIV detects the vortex. This can be seen in Fig. 5-6, which is reproduced here for convenience as Fig. 6-1.

While the variation during normal operation makes detection difficult until the vortex has significantly perturbed the system, some trajectories exhibit a signal suggesting the presence of a vortex earlier than expected. If the cause of this early perturbation could be discerned, it may be identified and used to achieve a level of performance without PIV approaching the performance available with PIV, though likely with inferior detection performance (i.e., more false negatives or false positives). Furthermore, the PIV-enabled controller can act as a benchmark in this setting, giving in some sense an "upper bound" on performance, by allowing the control design to

treat the system as fully observable. In a sense, if a controller attaining an acceptable level of performance cannot be obtained with PIV in the loop it may be impractical to achieve that level of performance in the field no matter what sensors are available.

In addition to using the PIV-enabled controller as a scaffold for building a controller that requires only local, traditional sensors there is the possibility that a system of engineering interest could be controller during actual operation through a PIV-enabled controller. PIV can be performed in combustion chambers of jet engines and power plants [107]. In certain modes of operation vortex formation or instability can be critical to performance [89], with models difficult to come by and simulation difficult [27]. PIV is used to study these combusting flows [49] and analyze how the interaction between the flow and the reaction can affect efficiency. Passive and open-loop active control approaches have been explored [43], but the possibility of actively controlling the combustion in a deployed system through real-time PIV measurements is exciting, though several technical hurdles must first be overcome.

6.4 Concluding Remarks

The possibilities offered by real-time PIV-enabled controllers are myriad, and many interesting questions arise out of the need to obtain flow information relevant to control quickly and robustly. This need for causal filtering in PIV is new, with the real-time requirement imposing constraints that are not present when PIV is used in its traditional setting.

By looking at PIV as a sensor in a control loop as opposed to a means of studying fluid dynamics makes clear the numerous research questions that are unanswered, each of which could provide not only superior performance for a controlled fluid-body system but a better understanding of how such systems behave and interact. The difficulty in modeling these systems does not preclude controlling them, and through controlling them new insight can be gained.

Ultimately, many threads of research can be carried out on how to perform real-time PIV more efficiently and robustly; how to focus the available computation on the features of the flow that are relevant to control; how to design controllers that take advantage of the information gained by PIV; and how to correlate PIV-enabled controllers with controllers using only local, deployable sensing. All of these are exciting problems and all of them have several fruitful avenues of attack. It is my hope that the results presented here will quickly be surpassed by more sophisticated filters and controllers operating on more complex systems, and that the knowledge gained will allow fluid-body systems to do things never before possible while granting insight into a challenging and important aspect of the behavior of fluids.

Appendix A

Implementation Details

This appendix contains great detail of some of the engineering and design decisions made in the process of executing this thesis. They are relegated to this appendix because they are not central to the primary argument, but are included to be useful to anyone wishing to reproduce these results or continue using the setup used for these experiments.

A.1 The Water-Tunnel

Steven Proulx (a former lab staff member) and I built the water-tunnel (see Fig. A-1 and Fig. A-2) specifically to enable the convenient study of control for fluid-body systems. For its size (2.4m x 1m) it has a wide test section (25 cm) to facilitate the study of a body which must move within the test section. The water tunnel has a free surface and thus the depth at the test section is not fixed, but in practice it is run with a depth of 20cm. It can achieve flow rates of over 20cm/s in the test section, while maintaining uniform flow. Using pure water at room temperature (20 C), the Reynolds number of a body in the test section is:

$$Re = \frac{UL}{\nu} = \frac{0.20m/s \cdot 0.050m}{1.004 \cdot 10^{-6} m^2/s} = 1.2 \cdot 10^4, \tag{A.1}$$

where U is mean fluid velocity in the test section, L is a representative length scale (chosen here to be 20% of the test section width which is about the size of the bodies studied in this thesis) and ν is kinematic viscosity. Thus, this setup allows fluid-body control problems at a Reynolds numbers on the order of 10,000 to be studied.

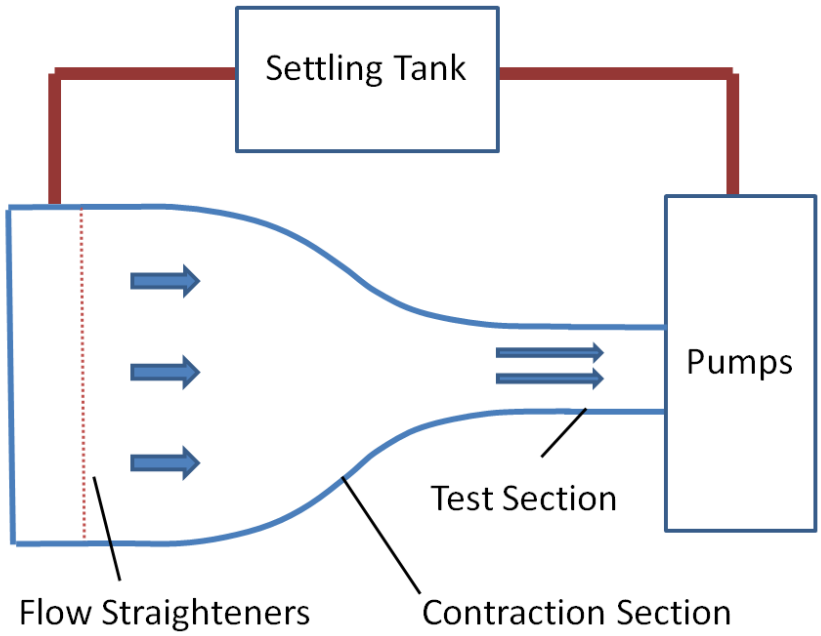


Figure A-1: Schematic of the water-tunnel hardware. The pumps circulate water through pipes into the settling tank, after which it spills over and flows through a flow straightening section consisting of three screens and a honeycomb of small-diameter tubes to eliminate any vorticity. The water then goes through a contraction section to smoothly accelerate the flow to 20cm/s in the test section, after which it returns to the pumps.

A.1.1 Flow Straightening

I desired a reasonably uniform flow in the test section for the control experiments, but did not require the flow to be as carefully regulated as would be required to study the fluid dynamics of the immersed bodies themselves in detail. Thus, I designed a reasonably effective flow straightening section without attempting to minimize every source of nonuniformity in the flow. The first stage in this straightening section is the

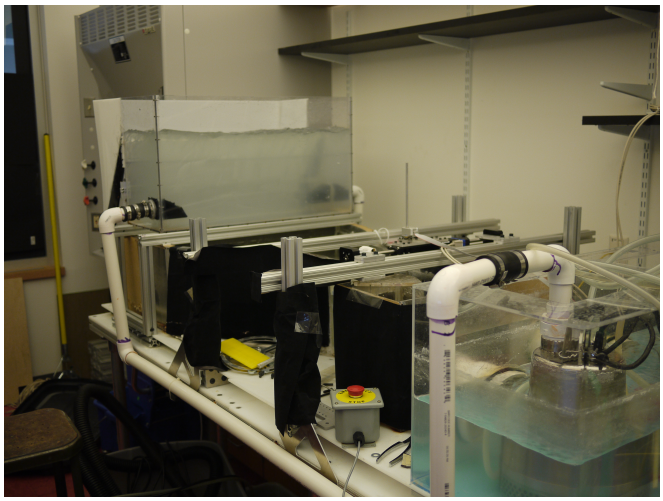


Figure A-2: Photo of the water-tunnel hardware. The upper left shows the settling tank filled of water as it is fed by the stainless steel pump. Note the black fabric used to prevent any unwanted reflections of the laser sheet back into the lab.

settling tank, into which the pumps discharge. The water entering this tank has a high speed, and the settling tank allows much of this energy to be dissipated. The water moves travels from this tank to the rest of the straightening section by a spillover, causing it to be relatively uniform across the water tunnel's width. After this spillover, the flow is straightened using three wire meshes and an aluminum honeycomb. The meshes act to reduce velocity variation over across the width of the water tunnel, and dampen out small-length scale vorticity. The goal of the honeycomb is eliminate medium scale vortices before entering the contraction section. This can be shown to be the case for a test section flow rate of 20cm/s (the maximum in these experiments) by the following calculation.

The volume flow rate in the test section is:

$$\dot{V} = w \cdot h \cdot U = 0.25m \cdot 0.20m \cdot 0.20m/s = 0.010m^3/s, \tag{A.2}$$

where w is width of test section, h is height of test section and U is mean fluid velocity in the test section. From this, U_s, the mean velocity in the straightening

section (ignoring the cross-section of the tube walls, a reasonable approximation for our setup), can be calculated as:

$$U_s = \frac{\dot{V}}{w_s \cdot h} = \frac{0.010m^3/s}{0.80m \cdot 0.20m} = 0.063m/s, \tag{A.3}$$

with w_s the width of test section and the height h the same as in the test section (seen in practice to be a good approximation).

The honeycomb tubes, which are hexagonal, have a hydraulic diameter of 1.0 cm. Thus, the Re in the tube, using the hydraulic diameter as a length scale and water at 20 C, is:

$$Re_{tube} = \frac{0.063m/s \cdot 0.010m}{1.004 \cdot 10^{-6}} = 630, \tag{A.4}$$

less than $Re = 2000$ which is generally considered the point at which flow in a pipe begins a transition to turbulence, thus the flow in the straightener is laminar. The entrance length for flow in a laminar pipe is a dimensionless number defined as:

$$EL_{laminar} = l_e/d, \tag{A.5}$$

where l_e is the length to a fully developed velocity profiled and d is the pipe diameter. $EL_{laminar}$ can be approximated as:

$$EL_{laminar} = 0.06 \cdot Re = 37.8, \tag{A.6}$$

for this situation, giving an entrance length of 38 cm. The straightening honeycomb is only 20 cm long, so fully-developed flow is not entirely achieved in the straightener tubes, but at half the entry length slug flow has developed and a sufficient degree of straightening has be achieved to provide uniform incoming flow.

Figure A-3: The high-QE camera used in PIV: a pco.1600 CCD.

A.2 PIV Hardware

This section describes the details of the hardware used to perform PIV.

A.2.1 Camera

I use a PCO.1600 cooled-CCD, high-Quantum Efficiency (high-QE) camera (see Fig. A-3) for imaging. The camera has a resolution of 1600x1400, but to increase the signal-to-noise ratio and increase the frame rate, I use 2x2 binning resulting in images that are 800x600. Using this setting I can easily capture single images at 32 frames per second, when using an exposure of approximately 2.5ms-5ms. This fast exposure has been seen in practice to minimize blurring while maintaining sufficient contrast. Alternatively, I can capture image pairs at 16 frames per second, allow less frequent sampling at lower interframe times. This feature makes it possible to observe much faster flows. The magnification of the lens is such that 1 pixel = 0.158 mm, or equivalently, 6.33 pixels per mm.

The camera is placed below the test section, imaging upwards. Because of this, the risk of leaked water dripping onto the camera and lens must be considered. To protect the camera, a box was built out of aluminum, with a top of optical quality glass (see Fig. A-4). The glass seals against a rubber gasket, effectively preventing

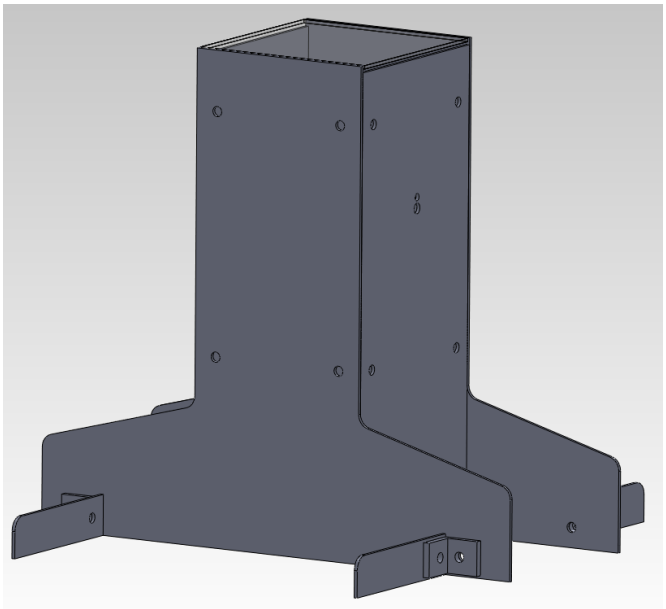


Figure A-4: SolidWorks rendering of box built to mount camera, and protect it from any droplets of water leaked from the test section.

any water leaking from the test section from damaging the camera. Finally, the extra layer of glass has been seen to not significantly reduce the brightness of the particles when imaged, an important feature as we do not have a great deal of contrast "to spare".

A.2.2 Laser and Optics

For illumination I have used two lasers: a LaVision 2 Watt continuous wave (CW) for lower speeds and a 100mJ double pulsed laser for higher speeds. Both lasers have the same wavelength: green at a wavelength of 532nm. The wavelength of these lasers is matched with the high-QE wavelength range of our camera, ensuring maximum efficiency and contrast. The continuous wave laser is cheaper and simpler, but it lacks the beneficial strobing effect of the pulsed laser, which eliminates the concerns regarding exposure time. However, for some of the experiments pursued in this thesis/footnoteIn the PIV chapter the vortex tracking results use the CW

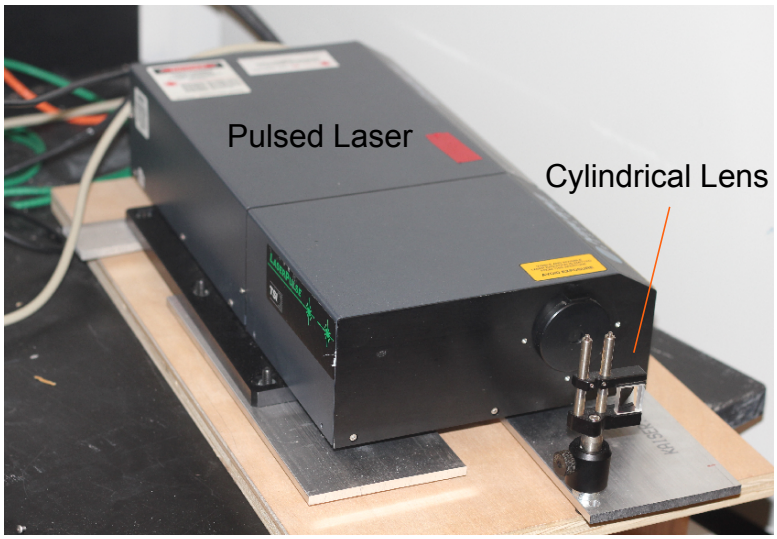


Figure A-5: Pulsed laser and optics.

laser, while the PIV-enabled control chapter the pulsed laser was used. it was more convenient.

The laser beam was turned into a sheet via a cylindrical lens which was rigidly mounted to the laser itself, limiting the opportunity for lens alignment issues (which can be both experimentally inconvenient and unsafe). The lens was a ThorLabs anti-reflectivity coated cylindrical lens with a focal length of -38.1mm (part number LK1325L1-A). The anti-reflectivity coating was chosen to match our laser frequency, minimizing the amount of light lost when passing through the lens.

A.2.3 Particle Seeding

I seed the water with 50μm polymer particles from Dantec Dynamics (part number PSP-50). These particles are approximately neutrally buoyant (in fact they are very slightly denser than water), and thus remain suspended easily when mixed via the pumps. The images produced are of sufficient quality that PIV software can generally process them quite easily (see Fig. A-6 and Fig. A-7 for an example of the images obtained and the vector fields resulting). Smaller particles (10μm glass spheres, also

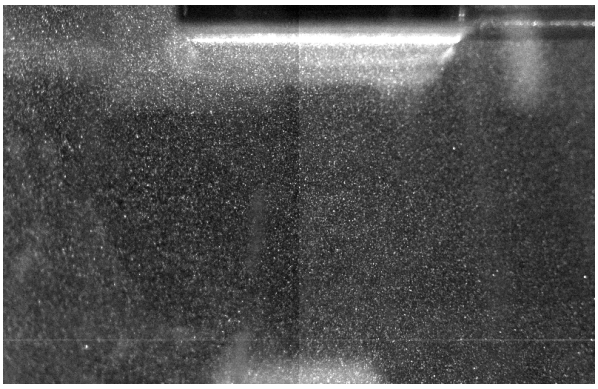


Figure A-6: Unprocessed PIV image of stagnation flow on setup (flow is vertical against a bluff body).

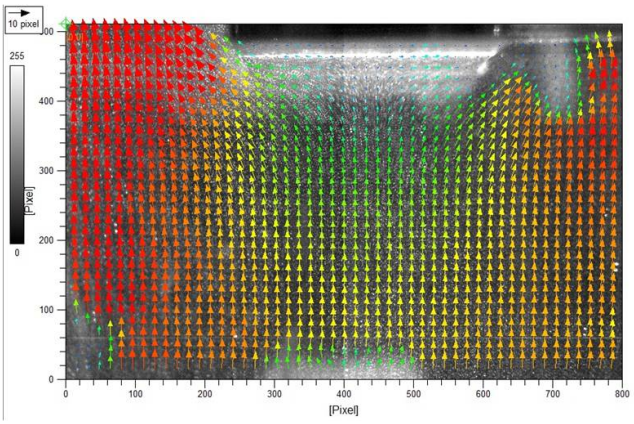


Figure A-7: Processed PIV image of stagnation flow on the water-tunnel setup (the algorithm struggles in the corners due to the edges of the image being out of focus).

from Dantec) were also tried, but did not scatter enough light to produce clear images with a short exposure.

A.3 Observer Gain Selection

The poles for the error dynamics of the observer in the case of a linear system are given by the eigenvalues of the matrix:

$$M = A - LC, \tag{A.7}$$

and thus are dependent on the system dynamics present in A and the measurement matrix C. However, first consider a second-order mechanical system with $n/2$ degrees of freedom, n states, and only integrator dynamics, as given in A_s below:

$$A_s = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}, \tag{A.8}$$

where $I \in \mathbb{R}^{n/2 \times n/2}$ is the identity matrix. If only the positions are measurable (as is the case in the HCP) the measurement matrix C is:

$$C = \begin{bmatrix} I & 0 \end{bmatrix}, \tag{A.9}$$

where $I \in \mathbb{R}^{n/2 \times n/2}$ is again the identity matrix. In the case of a 2 degree of freedom system, the observer gain matrix L:

$$L = \begin{bmatrix} 2/\epsilon & 0 \\ 0 & 2/\epsilon \\ 1/\epsilon^2 & 0 \\ 0 & 1/\epsilon^2 \end{bmatrix}. \tag{A.10}$$

places all the poles at exactly at at $-1/\epsilon$. For the actual dynamics of the HCP the poles will not be precisely at this location. However, so long as ϵ is sufficiently small the importance of the lower two rows of the matrix A become negligible and the real parts of the poles (i.e., the parts that determines convergence rate) are all very near

$-1/\epsilon$ relative to their distance from 0.

The value of ϵ used in this work is $\epsilon = .05$. Note that because the system is observable an L could be chosen that places the observer poles at any location, but in practice there are bandwidth limits on the observer beyond which tracking becomes very poor and possibly unstable.

For the HCP system, the linearized model gives observer poles at:

$$s = -20, -20, -21.1 \pm 5.687i, \tag{A.11}$$

where s is the laplace variable.

A.4 Centering the Balance Controller

Another feature of many underactuated systems, and balancing systems in particular, is a strong sensitivity to appropriate "zeroing." In other words, if the controller believes the equilibrium state of the controller to be slightly different from the true equilibrium, performance can rapidly degrade. Intuitively this occurs, because when the controller is at what it believes the equilibrium to be the system is actually causing it to accelerate away from the true center. Because the controller is unaware of this, it waits longer to take a corrective action, resulting in a larger ultimate deviation to correct the error than was necessary. Similarly, when at the true upright the controller takes an action to push it towards what it believes to be the center, and an unnecessary error is created.

To demonstrate this effect, the balancing controller designed above was executed on the HCP with two different zero points, one 1.5° greater than the other. While this disparity is small enough that it cannot easily be seen when setting up the system, is has a noticeable effect on performance, as demonstrated in Fig. A-8.

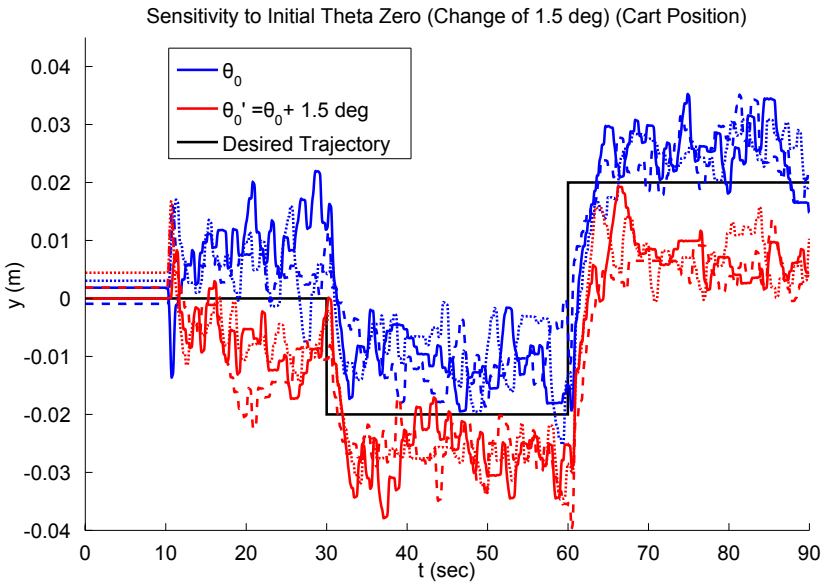


Figure A-8: Six trajectories of the HCP with two different centers, different by only 1.5°. This dramatic sensitivity to such a small angular offset demonstrates the importance of obtaining an accurate zero, either through careful calibration or through online adaptation of the center point.

A.4.1 Adapting the Center

To alleviate this problem, an estimate of the true system zero can be produced online as the system executes. The important caveat in producing this online estimate is that it must adapt at a slower timescale that the system itself, both in terms of the system's intrinsic dynamics (e.g., the fluid timescales) and the timescale of the controller. In this sense, adapting to calibration error in the centering is effectively adding integral term, though in the estimator rather than the controller. Furthermore, because we have a roughly accurate model of the system available from § 2.2, we can do better than simply integrate the error.

First we must examine the torque equations presented above, but with an included angular offset term, θ_0:

$$\tau_{aero} = -\frac{1}{2}c^2 s\rho V^2 \sin(\alpha - \theta_0) \tag{A.12}$$

$$\tau_{inertial} = \frac{1}{2}M_{add}c\cos(\theta + \theta_0)\, u \tag{A.13}$$

$$\ddot{\theta} = \frac{\tau_{aero} + \tau_{inertial}}{I_{total}} - k_d\dot{\theta} \tag{A.14}$$

$$\tag{A.15}$$

From these we can solve for θ_0 as a function of the current state and angular acceleration. By assuming θ_0 is small (but not making that assumption for the measured angle θ), we find that:

$$\theta_0 = \frac{1}{\cos\alpha}\left(\sin\alpha - \frac{\ddot{\theta}I_{total} - \tau_{inertial} + I_{total}k_d\dot{\theta}}{-\frac{1}{2}c^2 s\rho V^2}\right) \tag{A.16}$$

This requires us to measure $\ddot{\theta}$, which we must obtain by double-differentiating position measurements, resulting in a noisy estimate. However, as the timescale of the adaptation must be slower than the system timescale anyway, passing the estimated θ_0 through a low-pass filter resolves the problem, and allows us to compensate for

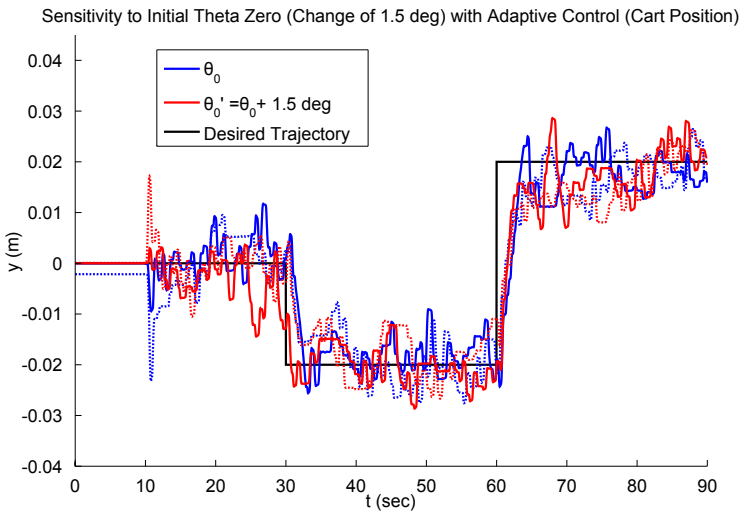


Figure A-9: Six trajectories of the HCP with the center being adapted online in the method described in §3.2. Note that despite the calibration disparity, both subsets of trajectories have means that are very close to the specified trajectory, and to each other.

calibration error. The success of this technique can be seen in Fig. A-9, in which the same angular error of 1.5° that caused difficulty in Fig. A-8 is canceled out effectively.

Appendix B

Robust Learning Control

This thesis has focused on iterative, data-driven local-model based control for solving fluid-body control problems. As evidenced in the previous chapters the data-driven model approach is very powerful, and can be successful in relatively complex control domains. However, a significantly different approach to controlling fluid-body systems that has also shown promise on complex fluid systems is Reinforcement Learning, or RL.

RL is in many ways a much larger departure from traditional model-based control than the other work in this thesis, as in contrast to the results presented in the remainder of this thesis RL does not require even a local model to be fit to measured data. Instead, it can directly learn the control policy for the fluid-body system without ever explicitly attempting to model the system's behavior. This approach worked well in my Master's Thesis [79] when finding an energetically-optimal trajectory for a flapping wing. The flapping wing, however, was not open-loop unstable, and the feedback controller used track the nominal trajectory was fixed at the beginning of the experiment based upon standard loop-shaping design techniques. This is in contrast to the HCP problem in which a poor controller can result in a failure event (e.g., falling over), and in which the controllers were updated automatically during the iteration.

Using RL controller-design methods without risking a failure during learning is a non-trivial problem, but not an insuperable one. The work presented in this appendix deals with learning a controller for a system with uncertain dynamics[1] while guaranteeing that no controller used during the learning process will cause the system to go unstable. The system used as an example in this appendix is not a fluid-body system (building an accurate model of such system, along with bounds on performance, is often a thesis project in itself), but it is a system that can be driven unstable by a poor choice of controller. There is still much further research to be done in this domain, but the work presented in the following sections present a good starting point for stability-guaranteed RL.

B.1 Introduction

Reinforcement Learning offers a very general set of methods for optimization which has been used successfully in numerous controls applications, particularly in robotics. Most uses of learning which have been successful on hardware involved learning an open-loop trajectory or control tape (see, for example [3, 47, 81, 88]). These trajectories can then be wrapped in stabilizing controllers designed using more traditional methods, such as Model Predictive Control [19] or the Linear Quadratic Regulator (LQR). More rarely, feedback policies themselves have been learned directly on hardware (e.g., [48, 101]), but not in situations in which instability greatly disrupts the learning process[2].

There is less work focusing on RL for feedback design when the system and task are such that instability can be an issue even as the policy converges. This has prevented the use of RL in high-performance tasks near the edge of stability, such as

[1] The uncertainty must be bounded for the techniques to work, with the class of uncertainties and the level of conservative similar to that seen in work on Robust Control.

[2] Feedback policies have been developed for stability critical systems through the use of Iterative LQR (see, for example, [2]), but the only policies which can result are those produced as a byproduct of iLQR , i.e., an LQR policy about the converged-to trajectory.

learning with hardware in-the-loop when the hardware could be damaged through the use of an unstable controller. This lack of work is not because RL is ill-suited to this domain of high-performance, low-stability-margin control, but because what seem to be the most natural parameterizations actually behave poorly in this domain.

In this chapter we will study four parameterizations of linear feedback controllers and examine their performance on an example learning task in which instability can be a significant issue. The task is to quickly reach with a flexible manipulator to catch a ball, with learning performed using REINFORCE. While the manipulator is modeled as an open-loop stable linear system, it is underactuated, lacks full-state information and can easily be driven unstable through the choice of an inappropriate controller.

Two of the most natural-seeming parameterizations (state feedback gains with a fixed observer and a feedback controller transfer function) do not guarantee stability, and suffer both from non-convexity of the set of stabilizing controllers and from small changes in parameters causing a policy to go from high-performance to unstable. The other two parameterizations studied (LQR cost matrices with a fixed observer and the Youla Parameterization (YP)) *do* guarantee stability. The LQR cost matrix parameterization can offer high-performance, high-bandwidth controllers in situations where there is no delay, the cost function is of a form similar to the quadratic cost assumed by LQR, and the noise is not excessively structured. However, the Youla Parameterization offers a richer set of controllers, allowing for better performance when the cost function is significantly non-quadratic or when the noise is structured but non-Gaussian.

The remainder of this chapter is organized as follows:

- **Section II** An introduction to the four parameterizations for linear feedback control studied in this chapter: Feedback Controller Transfer Function, State Feedback Gains with a Fixed Observer, LQR Cost Matrices with a Fixed Observer and the Youla Parameterization

- **Section III** A description of the flexible arm ball-catching dynamics and task
- **Section IV** A presentation of the REINFORCE method used for learning in this chapter
- **Section V** An analysis of the various parameterizations' performance in the context of the ball-catching task and REINFORCE learning
- **Section VI** A discussion of extensions of the YP to broader classes of systems
- **Section VII** A brief conclusion highlighting the critical points of this chapter

B.2 Linear Feedback Controller Parameterizations

The general task of designing a linear feedback controller may be thought of in the context of Fig. B-1, where $K(s)$ and $P(s)$ are linear transfer functions in the laplace variable s. The designer must choose a $K(s)$ that, from measurements of the output y produces an input u that results in good performance by some designer-specified metric. In this section we consider four representations of $K(s)$ in the context of learning: Directly learning a linear transfer function for $K(s)$ in § B.2.1, representing $K(s)$ as a Kalman filter combined with learned state feedback gains (§ B.2.2), representing $K(s)$ as a Kalman filter then finding LQR feedback gains by learning LQR cost matrices Q and R (§ B.2.3) and finally, representing the controller in the Youla Parameterization (§ B.2.4). In § B.5 we present their performance using REINFORCE learning on the example ball-catching task described in § B.3.

B.2.1 Feedback Transfer Function

Perhaps the simplest parameterization is to learn the transfer function of a feedback controller $K(s)$ (see Fig. B-1) directly. As in theory $K(s)$ can be of arbitrary order, one must choose a set of $K(s)$ over which to actually search. In this chapter, the

$K(s)$ considered were of the form:

$$K(s) = \frac{\prod_{i=0}^{n} \beta_i s^i}{\prod_{i=1}^{n} (s - \alpha_i)}, \tag{B.1}$$

where the β_i are learned and the α_i are fixed. In effect, the poles of $K(s)$ are set by the designer while the numerator coefficients (and thus the zeros) are learned.

The α_i used for $K(s)$ are given below:

- α_1 through α_{15} are distributed logarithmically between $-10^{-0.5}$ and $-10^{0.5}$
- α_{16} and α_{17} are $-0.5 \pm 0.1i$
- α_{18} and α_{19} are $-1.1 \pm 0.05i$
- α_{20} and α_{21} are $-1.1 \pm 0.5i$
- α_{22} and α_{23} are $-2.3 \pm 1i$

These α_i were chosen because they covered a range of time-scales around the time-scale of the system, as well as several oscillatory modes with frequencies near the open-loop oscillation frequency of the system.

Learning $K(s)$ directly effectively allows the possibility of an arbitrary observer structure and the behavior of additional controller states (e.g., for integral control) to be learned. This parameterization is quite rich, with any linear controller possessing the poles specified with the α_i capable of being represented. However, less knowledge of the system is encoded in the structure of the parameterization than in the other parameterizations explored in this chapter and thus achievable performance can depend greatly on the set of transfer functions $K(s)$ that are actually considered. Also, stability is not guaranteed in this case, with the relationship between the structure of $K(s)$ and the stability of the resulting system quite complicated.

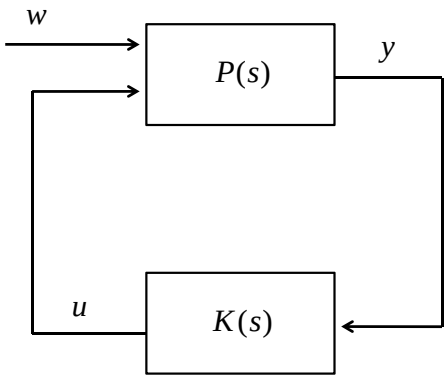


Figure B-1: Block diagram of the system used for Direct $K(s)$ Learning. Controllers are parameterized by selecting a linear system $K(s)$, with the specific form shown in Eq. (B.1).

B.2.2 State Feedback Gains with Fixed Observer

Fixing a high-performance observer (e.g., a Kalman Filter) and learning state-feedback gains seems like a very natural parameterization for learning a feedback controller (see Fig. B-2), with LQR optimal controllers being within the set of available controllers. The dimension of the learned policy is the same dimension as the state, and it is easy to initialize the policy to LQR gains found for reasonably chosen Q and R matrices. However, as §B.5.2 shows, the performance of this parameterization during learning is actually very poor.

B.2.3 LQR Cost Matrices

Directly parameterizing controllers in the space of LQR cost matrices and using a fixed observer (see Fig. B-3) has a number of advantages. Perhaps foremost among these is that by restricting one's attention to the cone of positive-definite matrices for Q and R (a convex set) only stabilizing controllers are considered. However, the set of possible controllers that can be represented in this parameterization is actually quite limited due to being restricted to scalar gains on the state variables. Furthermore, the set of controllers is heavily overparameterized. While symmetric positive definite Q

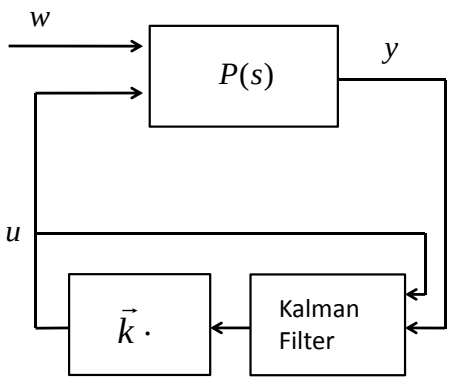


Figure B-2: Block diagram of the State Feedback Gain parameterization with a fixed observer. Controllers are parameterized by selecting n scalar gains k_1 through k_n where n is the number of states of the system.

and R matrices have $n(n+1)/2$ and $m(m+1)/2$ free parameters respectively for an n state system with m inputs, the space of controllers is of dimension nm. Thus, the set of controllers will always be over-parameterized, making for a possibly inefficient representation as learning must operate in an unnecessarily high-dimensional space. As learning performance has been shown to degrade with the number of learned parameters (see [80]), this is best avoided. Furthermore, dealing with delays in continuous time is non-trivial, and as the stability margins of LQR controllers with Kalman filters can be very small [33], even a small delay can be destabilizing.

B.2.4 The Youla Parameterization

As the Youla Parameterization (YP) is not widely known outside of the controls literature, we will first briefly describe the YP for single-input single-output linear systems, for which the main arguments are very simple, although similar parameterizations exist for much more complex systems (see Section B.6).

Given a stable plant $P(s)$ and a feedback controller $K(s)$, both finite-dimensional

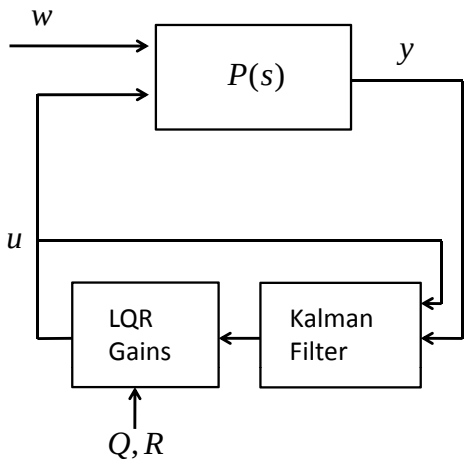


Figure B-3: Block diagram of the LQR Cost Matrix parameterization. Controllers are parameterized by two positive definite cost matrices Q and R, as is done in the design LQR controllers.

SISO systems, the closed-loop system from u to y is:

$$H(s) := \frac{Y(s)}{U(s)} = \frac{P(s)}{1 - P(s)K(s)}. \tag{B.2}$$

Note that the relationship between the parameters of the feedback controller $K(s)$ and the closed loop system are highly nonlinear. The Youla parameter associated with this controller is defined as

$$Q(s) := \frac{K(s)}{1 - K(s)P(s)}. \tag{B.3}$$

Clearly for any $K(s)$ this $Q(s)$ exists.

Alternatively, given a plant $P(s)$ and a Youla parameter $Q(s)$, it is simple to construct the controller $K(s)$:

$$K(s) = \frac{Q(s)}{1 + Q(s)P(s)} \tag{B.4}$$

and the closed-loop system is

$$H(s) = P(s)[1 + Q(s)P(s)] \tag{B.5}$$

which is affine in $Q(s)$. Furthermore, since $P(s)$ is stable it is clear that $H(s)$ is stable if and only if $Q(s)$ is stable.

In summary, every controller $K(s)$ can be represented equivalently by a $Q(s)$, and the closed loop system is affine in $Q(s)$ and stable if and only if $Q(s)$ is, so the set of stable $Q(s)$ is complete affine parameterization of *all* stabilizing linear controllers for $P(s)$. Given a Youla parameter $Q(s)$, one can easily construct the feedback controller from (B.4). Note that it is not necessarily the case that $K(s)$ itself is stable considered as an isolated system.

Furthermore, since the closed-loop system $H(s)$ is affine in $Q(s)$, many important cost functions are convex in the $Q(s)$, including LQG, H^∞, and time-domain bounds on overshoot, rise-time, and settling time. For this reason it has long been a central tool for designing feedback controllers via optimization methods (see, e.g, [113, 115, 17, 6, 38] and many others).

There are many cases in which, even if the system model $P(s)$ is known well, online learning can be advantageous over model-based design. For example, if reference or disturbance commands have characteristics which are changing over time, or do not fit a mathematical framework which is easy to optimize over, such as Gaussian noise with a known spectral density, or bounded energy signals. Another case is when the cost/reward for the learning process is not easily represented mathematically, e.g. qualitative ranking by human teachers. Furthermore, the results proposed in this chapter can be easily extended to cases in which $P(s)$ is only approximately known, and clear advantage could be gained by optimizing on hardware-in-the-loop experiments (see Section B.6).

The most intuitive view of the YP is to consider it as a stable system $P(s)$ (pos-

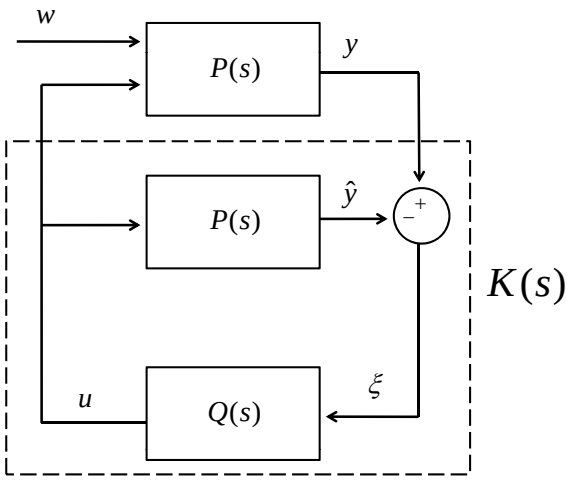


Figure B-4: Block diagram of the Youla Parameterization (YP). Controllers are parameterized by selecting a stable linear system $Q(s)$ of the form Eq. (B.1).

sibly with a disturbance input w) in feedback with controller consisting of a copy of the system dynamics $P(s)$ (without the disturbance input) and a second arbitrary stable system $Q(s)$ acting on the difference (see Fig. B-4a). As $Q(s)$ varies over all stable linear systems, the combination of $P(s)$ and $Q(s)$ will produce only stabilizing controllers.

While theoretically $Q(s)$ can be any stable linear system, in practice one must choose some limited subset of systems to study (much as was done in the case of representing $K(s)$). While the literature has proposed a number of affine representations of $Q(s)$ (see [17]), this chapter will focus on choosing a $Q(s)$ of the form shown in Eq. (B.1). This representation of $Q(s)$ is clearly affine, is relatively rich and has shown good performance in practice.

B.3 Catching with a Flexible Arm: An Example System

The simulations in this chapter are based on a simulated repeated ball-catching task for a flexible arm (see Fig. B-5). The task is as follows: A flexible manipulator starts at rest at the zero position (i.e., $y = \dot{y} = 0$). A ball enters at a distance x_b traveling at a speed v_b and at a height such that it will be caught if the manipulator's end effector reaches an angle y_b. The speed v_b and target angle y_b are drawn from uniform random distributions, with the distance x_b fixed. The arm reaches for the ball, inducing flexural modes in the beam that can not be directly sensed or controlled. The frequency and magnitude of the flexural modes are of the same order as those of the reaching task, causing them to have non-negligible effects on end-effector position, and thus catching performance.

The arm is modeled as a homogeneous rectangular beam of length L, Young's Modulus E, linear density λ and cross-sectional moment of inertia I_y. The total rotational inertia of the arm and actuator about the revolute joint is I. The actuator acts with a torque u on the base of the arm while the end of the arm is measured, giving an angular measurement y. A spring and very weak damper with constants k and b respectively work to hold the arm at zero, making the system weakly open-loop stable. A structured disturbance acts on the base of the beam with a torque w. This disturbance is sinusoidal with fixed frequency ω_d and amplitude f_d, as shown below:

$$w(t) = f_d \sin(\omega_d t) \tag{B.6}$$

The system is modeled in the linear regime (i.e., small deformations and neglecting the centrifugal component of the dynamics) using the first three resonant modes of the system. The parameters may be seen in Table B.1.

This system is underactuated and does not have full-state information, making it

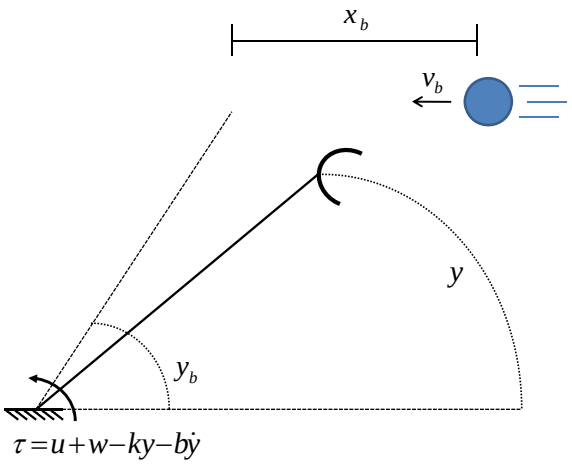


Figure B-5: Diagram of the simulated system. A ball enters at a distance x_b traveling at a speed v_b and at a height such that it will be caught if the manipulator's end effector reaches an angle y_b. The speed v_b and target angle y_b are drawn from uniform random distributions, with the distance x_b fixed. The input is a torque u, the disturbance a torque w and the output a measured angle y. The arm attempts to catch the ball by driving y to y_b in time to catch the ball, while minimizing arm velocity at the time the ball arrives. Costs are also imposed for torque, as well as additional costs for exceeding specified torque and angle values. Catching performance is judged by Eq. (B.10).

a non-trivial control task. While it is open-loop stable, it is easily capable of being driven unstable with a poorly chosen control law.

A catching trajectory was considered "good" if the end of the arm was at the desired position and a low velocity when the ball arrives. A trajectory was penalized for using torque and exceeding specified torque and angle thresholds. The specific function used to quantitatively evaluate a trajectory is shown in Eq. (B.10). An example of a successful catching trajectory can be seen in Fig. B-7.

B.4 The Learning Algorithm: Episodic REINFORCE

The algorithm used for learning is Episodic REINFORCE, in which the policy $\pi(y; \phi)$ is a function of the output of the system y and a set of parameters ϕ. The learning is

Table B.1: Parameters of flexible-arm ball-catching task.

L	1.5 m	I_y	2.77×10^{-7} m^4
E	3 GPa	b	0.01 Ns/m
k	0.5 kN/rad	v_b	60 ± 3 m/s
y_b	1 ± 0.05 rad	x_b	100 m
f_d	10 Nm	ω_d	π rad/s
λ	0.1 kg/m	I	1 kg m^2

performed episodically (i.e., the system states are reset between policy evaluations), and the stochasticity of the policy is on the policy parameters, rather than the outputs. The update appears in [109] for learning the mean of a Gaussian element, but we will briefly derive here the specific update used in this chapter, in which a vector of parameters is learned with identical noise and learning rate.

Consider the REINFORCE update (formulated for cost instead of reward) in which a vector of parameters share the same learning rate *eta*:

$$\phi_{i+1} = \phi_i - \eta \left(J(\phi_i') - b\right) \frac{\partial}{\partial \phi_i'} \ln(g(\phi_i')), \tag{B.7}$$

with ϕ_i' the actual parameters used on trial i, b a cost baseline, $J(\phi_i')$ the cost associated with the policy parameters ϕ_i' and $g(\phi_i')$ the probability of using parameters ϕ_i' on trial i. If $g(\phi_i')$ is a multivariate Gaussian distribution with mean ϕ_i and independent noise on each element with covariance σ^2, the eligibility $\frac{\partial}{\partial \phi_i} \ln(g(\phi_{p_i}))$ can be computed as:

$$\frac{\partial}{\partial \phi_i} \ln(g(\phi_{p_i})) \propto (\phi_i' - \phi_i)\,. \tag{B.8}$$

Clearly, $\phi_i' = \phi_i + \phi_{p_i}$, thus, after including all scalars in the learning rate, the update may be written:

$$\phi_{i+1} = \phi_i - \eta \left(J(\phi_i + \phi_{p_i}) - J(\phi_i)\right) \phi_{p_i}. \tag{B.9}$$

Note that while an averaged baseline is common in the literature, we have here used

a second policy evaluation (i.e., a "nominal" policy evaluation). This was found to learn using fewer total policy evaluations on the system, even though two evaluations were needed per update. Thus, the resulting learning algorithm was performed as follows: at iteration i the policy ϕ_i is executed and used to find the baseline cost b. The policy is then perturbed by a small amount ϕ_{p_i}. This perturbed policy is executed, and the difference in performance is used to compute the policy update, which is shown above in Eq. (B.9):

The cost function $J(\phi)$ used for this experiment is shown below:

$$\begin{aligned} J(\phi) &= \int_0^{x_b/v_b} ru^2 dt + c_1 \left((y-y_b)|_{x_b/v_b}\right)^2 + c_2 \left(\dot{y}|_{x_b/v_b}\right)^2 \\ &\quad + c_{sat} \max\left(0, \max((u/u_{sat})^2 - 1)\right) \\ &\quad + c_{crash} \max\left(0, \max((y/y_{crash})^2 - 1)\right). \end{aligned} \tag{B.10}$$

It rewards achieving the desired angle and velocity $y = y_b, \dot{y} = 0$ when the ball arrives at $t = x_b/v_b$, while minimizing applied torque u and avoiding angles greater than y_{crash} and torques greater than u_{sat}. The constants r, c_1, c_2, c_{sat} and c_{crash} are used to weight the relative importance of these costs.

B.5 Results

The four parameterizations previously introduced in § B.2 were used for REINFORCE learning of controllers for the ball-catching example task. The specific results for each parameterization are discussed below, but the primary result is that the Youla Parameterization provides the best learning performance as well as the best performance from the resulting controller. While clearly the performance for both the YP and directly learning $K(s)$ depend on the selection of the the roots of $K(s)$ (or $Q(s)$), several reasonable selections of the α_i produced effectively the same behavior.

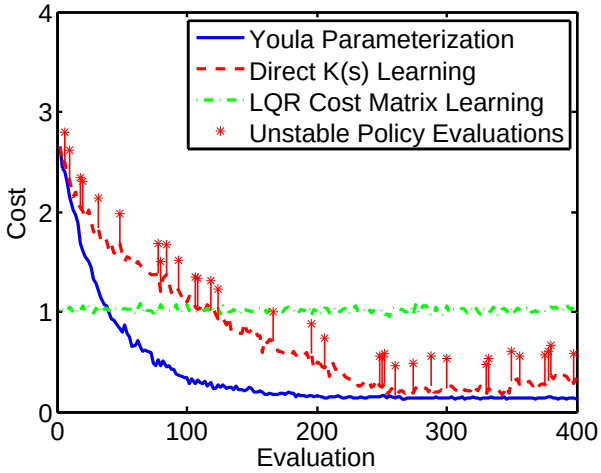


Figure B-6: Performance of YP and learning $K(s)$ directly for zero initial policy parameters and using tuned policy gradient. Ten trials of 200 iterations (400 evaluations) were performed for each curve, then averaged. The Y-axis is normalized by the cost of the initial LQR controller performing this task. Note how the weakly stable nature of the task and system result in many unstable policy evaluations even as the Direct $K(s)$ learning converges.

B.5.1 Direct $K(s)$ Learning Results

Experiments on the flexible arm ball-catching task have shown that although learning converges reasonably well (see Fig. B-6) for direct $K(s)$ learning, numerous unstable controllers are applied to the system throughout the learning process, even as the controller converges. This is due to the task effectively rewarding behavior on the margin of stability. Moreover, the convergence was significantly slower than that of the YP. Finally, while the resulting cost was similar to that of the YP (due to being capable of avoiding saturations and actually reaching the target point reasonably well), the actual trajectory followed was somewhat erratic. Thus, while this parameterization is reasonably well-suited to learning when instability is not a significant issue, and in theory a sufficiently rich $K(s)$ can produce the same controllers as any of the parameterizations examined here, in practice performance is worse for $K(s)$ with natural selections of the α_i than for similarly reasonable selections of $Q(s)$.

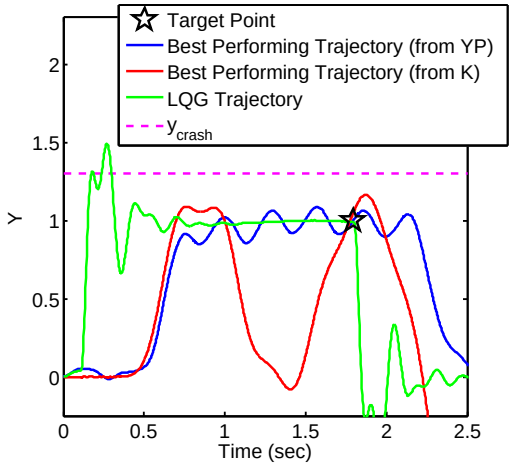


Figure B-7: Tracking performance of resulting controllers from the YP, LQR Cost Matrices Learning and Direct $K(s)$ Learning. Note that while all three parameterizations shown here hit the target point well for the nominal values, randomization of v_b (as well as greater use of actuation) degraded the performance of the controller from Direct $K(s)$ Learning and overshoot that could not be eliminated by learning caused the LQR Cost Matrix controller to incur unacceptable cost. As a result, the Youla Parameterization controller achieved the best performance.

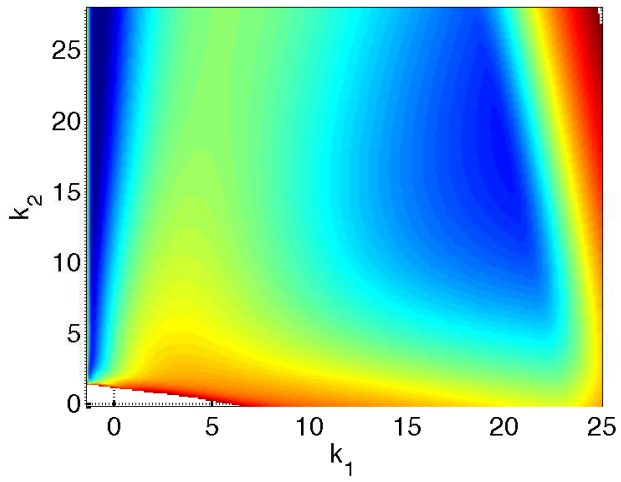


Figure B-8: Cost landscape for state feedback gains with a fixed observer for k_1 and k_2 with k_3 through k_8 set to their LQR values. Colors correspond to the log of the cost normalized to the LQR cost, and policies with with a cost greater than five times that of LQR are omitted. Note the very non-uniform gradients and non-convexity.

B.5.2 State Feedback Gain Learning Results

When applied to the example system, it was found that the sensitivity of cost to the state feedback gains is very non-uniform, resulting in either unacceptably slow learning (by learning slowly enough to identify the steep gradients warning of instability) or frequent unstable policy evaluations (by using perturbations and updates too large to identify oncoming instability). Both of these will result in unacceptably poor learning performance. Figure B-8 shows the cost landscape for two of the eight feedback gains on the flexible arm catching task. The non-convexity and non-uniformity evident in the figure makes learning in this parameterization completely impractical. Because of this the other limitations of this approach (e.g., a limited set of representable controllers) are of minor importance compared with the fundamental difficulty of excessively non-uniform cost sensitivity to parameters.

B.5.3 LQR Cost Matrix Learning Results

When the LQR Cost Matrix parameterization was used for learning, it was found to improve policy performance very little, if at all, from its initial value. This is likely due to the quite limited set of controllers available in this parameterization, and the very large parameter changes required to effect reasonable change in the output. No unstable controllers were applied to the system, and reasonable trajectory tracking was achieved, but performance with respect to the given cost function did not improve. As the cost function was not quadratic on state and action, is is quite likely that the true optimal controller was not within the set of controllers parameterized by LQR cost matrices, and indeed it could be that no controller of performance comparable to that attained by other parameterizations was capable of being represented. Thus, the ultimately achievable performance is inferior to that obtainable by representations that parameterize a more general controller set.

B.5.4 Youla Parameterization Learning Results

As Figures B-6 and B-7 show, the YP provided the best overall performance, with good learning performance (i.e., quick, predictable convergence) and good ultimate controller performance. The fact that no intermediate trials were unstable is an added benefit, as it would allow for the application of learning using the YP even to sensitive hardware-in-the-loop learning systems that could be too delicate to experience excessively violent unstable controllers. The ultimate ball-catching performance is clearly quite good (though with imperfect noise rejection), and the control and output saturations that are problematic in the context of the LQR Cost Matrix learning are easily dealt with through learning in the YP. While rise-time and noise rejection were not as strong as for the LQR cost matrix control, these performance measures depend upon the selection of the representation for $Q(s)$, as with appropriate $Q(s)$ *any* stabilizing linear controller can be represented, including the LQR controller. While

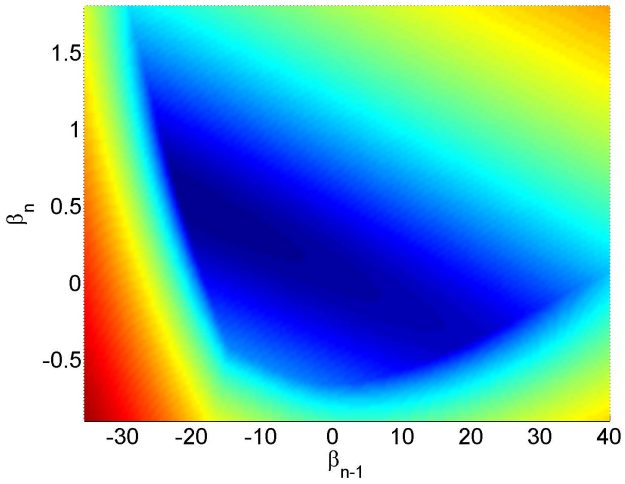


Figure B-9: Cost landscape for the Youla Parameterization for β_{n-1} and β_n with the rest of the β_i set to their converged to values. Colors correspond to the log of the cost normalized to cost of the converged to policy. Note the uniform gradients and convexity.

this chapter has not addressed in detail how $Q(s)$ should be represented, a wealth of work in the controls literature deals with this very problem, and exploring these different representations in the context of learning is an interesting future direction.

Furthermore, the convexity of many common cost functions in the YP (including Linear Quadratic costs and saturation costs in the time domain as were used here) can result in the convexity of the value function learning must descend (see Fig. B-9). While this convexity is not guaranteed for arbitrary cost functions, if regions of the value function have this property it can still be advantageous to both the rate at which learning converges and to the avoidance of local minima.

B.6 Extensions of the YP to More Complex Systems

As the Youla Parameterization clearly demonstrated the best performance, we will focus here on the possible extensions to the simple YP described in this chapter. Due

to their utility in feedback optimization problems, finding convex parameterizations of stabilizing controllers has been a major focus of control theory research for many decades. In this section we briefly overview a few major results related to the Youla parameterization which may also be applied in a policy learning architecture.

B.6.1 Unstable and Multivariable Systems

In this chapter we have considered a stable single-input single-output system, mainly for simplicity's sake. The Youla parameterization for unstable and multivariable systems is classical and is widely used in control design, however the construction is somewhat more involved (see, e.g., [113, 17, 25]).

B.6.2 Rapid Switching Between Controllers

In this chapter we have assumed that the learning is performed over repeated performances of a task, all starting from the same initial conditions. If, on the other hand, learning is performed by switching between candidate controllers during continuous operation (as in supervisory adaptive control) then stability under the switching process becomes an issue [53].

It is well known that even if two or more feedback controllers are stabilizing for a given system when considered in isolation, it is possible destabilize the system by rapidly switching between them. Giving tight conditions on stability of switched systems is intrinsically difficult – NP-Hard or undecideable, depending on the exact formulation [15, 102].

When the switching is chosen by the controller, stability can be ensured by enforcing a dwell-time between switching [53]. Particularly relevant to the work in this chapter are designs in which switching can be performed arbitrarily rapidly. One such design is to represent all controllers in the Youla parameterization, with a reset on the states of $Q(s)$ at the moment of controller switch [39].

B.6.3 Uncertain Models

A natural application of learning or adaptive control is where the system model is not exactly known in advance. In this case, common methods include online system identification and control design, or direct policy search.

An alternative, made feasible in the framework proposed here, is to enforce that all control policies searched over are robustly stabilizing, but use the learning procedure to evaluate performance. In the control literature it is common to enforce robust stability via a small-gain condition by limiting the H^∞ norm of the closed-loop transfer function [115, 25, 17]. This constraint is convex in the Youla Parameterization. Characterizating uncertainty via a small-gain bound may be crude if more is known about the modelling errors. Convex parameterizations are also available for some tighter definitions of uncertainty [73].

B.6.4 Nonlinear Systems

For certain classes of nonlinear systems there exists parameterizations of all stabilizing controllers, similar to the Youla parameterization, for a variety of definitions of stability (see, e.g., [23, 70, 55, 8] and many others). In these cases, the Youla parameterization still gives a complete parameterization of all stabilizing controllers in terms of a stable nonlinear operator Q. However the closed-loop dynamics of the feedback system is no-longer affine in Q and the optimization process is much more difficult.

B.6.5 Decentralized Systems

For a long time, the problem of decentralized control - in which each controller/agent has access to data from a limited subset of sensors - has been a very difficult problem. Witsenhausen's classic counterexample showed that even in the linear-quadratic-gaussian case, the optimal decentralized controller may not be linear, and searching

for the optimal linear controller can be a nonconvex problem with many local minima [111]. Recently, very interesting results have emerged describing conditions under which the search for an optimal linear decentralized control is convex in the Youla parameter [82, 51]. An extension to the nonlinear case has also been considered [112].

Note that for these results to be applied directly to our method, it would be necessary that all controller locations have access to the reward signal. A substantially more challenging problem would be one where each individual controller can only see a subcomponent of the total reward.

B.7 Conclusion

This chapter has explored the possibility of learning feedback controllers in the Youla Parameterization to avoid the issue of intermittent instability while learning, and to make learning perform better via convexification of the underlying problem. The system studied in this appendix is not a fluid-body system, but it is a complicated system which can suffer from instability if improperly controlled. Deploying this technique on a fluid-body system would require more accurate modeling than that performed in this thesis on the HCP, as well as trustworthy bounds on the error. However, the advantages of learning directly in the policy space, and of using RL on experimental systems without the risk of applying unstable controllers, could justify the additional difficulties if the modeling is possible for the system of interest.

This work as has shown that the significance of appropriately choosing a parameterization for learning is clear, with seemingly natural parameterizations making learning seem impossible, while well-chosen parameterizations provide high-performance both in the context of learning convergence and the resulting controller. For the flexible arm system used as an example in this chapter, the Youla Parameterization convincingly performed the best. Its flexibility, convexity in the case of many standard cost functions, guaranteed stability and richness make it very well-suited to

learning in situations when the system model is known.

The extensions to the YP present in the controls literature offer a wealth of opportunities for study. From the appropriate representation of $Q(s)$ to applications in nonlinear and uncertain systems, many possibilities exist for improving learning through further study of the YP. With the stability guarantees offered by the Youla Parameterization, fluid-body systems which are open-loop unstable could be explored without fear of damage to an experimental apparatus, or a controlled system.

Bibliography

[1] Abbeel, P., Coates, A., Ng, and A.Y. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 2010.

[2] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. In *Proceedings of the Neural Information Processing Systems (NIPS '07)*, volume 19, December 2006.

[3] Pieter Abbeel, Varun Ganapathi, and Andrew Y. Ng. Learning vehicular dynamics, with application to modeling helicopters. *NIPS*, 2006.

[4] R. J. Adrian. Twenty years of particle image velocimetry. *Experiments in Fluids*, (39):159–169, July 2005.

[5] Alekseenko, SV, Kuibin, PA, Okulov, and VL. *Theory of concentrated vortices*. Springer-Verlag Berlin Heidelberg, 2007.

[6] Anderson, J. M. Streitlien, K. Barrett, D.S. Triantafyllou, and M.S. Oscillating foils of high propulsive efficiency. *Journal of Fluid Mechanics*, 360:41–72, 1998.

[7] A. Anderson, U. Pesavento, and Z. Jane Wang. Analysis of transitions between fluttering, tumbling and steady descent of falling cards. *J. Fluid Mech.*, 541:91–104, 2005.

[8] Brian D.O. Anderson. From Youla-Kucera to Identification, Adaptive and Nonlinear Control. *Automatica*, 34(12):1485 – 1506, 1998.

[9] Barbagallo, A., Sipp, D., Schmid, and P.J. Closed-loop control of an open cavity flow using reduced-order models. *Journal of Fluid Mechanics*, 641:1, 2009.

[10] D.N. Beal, F.S. Hover, M.S. Triantafyllou, J.C. Liao, and G. V. Lauder. Passive propulsion in vortex wakes. *Journal of Fluid Mechanics*, 549:385402, 2006.

[11] Abderrahmane Bennis, Miriam Leeser, Gilead Tadmor, and Russ Tedrake. Implementation of a Highly Parameterized Digital PIV System On Reconfigurable Hardware. In *Proceedings of the Twelfth Annual Workshop on High Performance Embedded Computing (HPEC)*, Lexington, MA, September 2008.

[12] Bergmann, M., Cordier, and L. Optimal control of the cylinder wake in the laminar regime by trust-region methods and pod reduced-order models. *Journal of Computational Physics*, 227(16):7813–7840, 2008.

[13] Betts and J.T. *Practical methods for optimal control and estimation using nonlinear programming*. Society for Industrial & Applied, 2010.

[14] Thomas R. Bewley, Parviz Moin, and Roger Temam. DNS-based predictive control of turbulence: an optimal benchmark for feedback algorithms. *Journal of Fluid Mechanics*, 447:179–225, 2001.

[15] Vincent D. Blondel and John N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems & Control Letters*, 41(2):135 – 140, 2000.

[16] Bock, H.G., Plitt, and K.J. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.

[17] Stephen P. Boyd and Craig H. Barratt. *Linear Controller Design: Limits of Performance*. Prentice Hall, NJ, 1991.

[18] D. A. Bristow, M. Tharayil, and A. G. Alleyne. A Survey of Iterative Learning Control: A Learning-Based Method for High-Performance Tracking Control. *IEEE Control Systems Magazine*, 26(3):96–114, Jun 2006.

[19] E. F. Camacho and Carlos Bordons. *Model Predictive Control*. Springer-Verlag, 2nd edition, 2004.

[20] Louis N. Cattafesta, Qi Song, David R. Williams, Clarence W. Rowley, and Farrukh S. Alvi. Active control of flow-induced cavity oscillations. *Progress in Aerospace Sciences*, 44(78):479 – 502, 2008.

[21] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *CURRENT SCIENCE*, 78(7):808–817, April 2000.

[22] Couplet, M., Basdevant, C., Sagaut, and P. Calibrated reduced-order POD-Galerkin system for fluid flow modelling. *Journal of Computational Physics*, 207(1), 2005.

[23] C. A. Desoer and R. W. Liu. Global parametrization of feedback systems with nonlinear plants. *Systems & Control Letters*, 1(4):249 – 251, 1982.

[24] Donoho and D.L. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[25] John C. Doyle, Bruce A. Francis, and Allen R. Tannenbaum. *Feedback Control Theory*. Macmillan Publishing Company, New York, 1992.

[26] Fawcett and T. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[27] Fureby and C. Large eddy simulation of combustion instabilities in a jet engine afterburner model. *Combustion science and technology*, 161(1):213–243, 2000.

[28] Gerhard, J., Pastoor, M., King, R., Noack, B.R., Dillmann, A., Morzynski, M., Tadmor, and G. Model-based control of vortex shedding using low-dimensional Galerkin models. *AIAA paper*, 4262:2003, 2003.

[29] Shlomo Geva and Joaquin Sitte. Performance of temporal differences and reinforcement learning in the cart-pole experiment. *Proceedings of 1993 International Joint Conference on Neural Networks*, pages 2835–2838, 1993.

[30] Ghosh, J., Paden, and B. Pseudo-inverse based iterative learning control for nonlinear plants with disturbances. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 5, pages 5206–5212. IEEE, 1999.

[31] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1):99–131, 2005.

[32] Philip E. Gill, Walter Murray, and Michael A. Saunders. *User's Guide for SNOPT Version 7: Software for Large -Scale Nonlinear Programming*, February 12 2006.

[33] M. Green and D.J.N. Limebeer. *Linear Robust Control*. Prentice Hall, 1995.

[34] Gursul, I., Wang, Z., Vardaki, and E. Review of flow control mechanisms of leading-edge vortices. *Progress in Aerospace Sciences*, 43(7-8):246–270, 2007.

[35] Hart and D.P. Super-resolution PIV by recursive local-correlation. *Journal of Visualization*, 3(2):187–194, 2000.

[36] Douglas P. Hart. High-Speed PIV Analysis Using Compressed Image Correlation. *Journal of Fluids Engineering*, 120:463–470, 1998.

[37] J.-W. He, R. Glowinski, R. Metcalfe, A. Nordlander, and J. Periaux. Active Control and Drag Optimization for Flow Past a Circular Cylinder: I. Oscillatory Cylinder Rotation. *Journal of Computational Physics*, 163(1):83 – 117, 2000.

[38] Joao P. Hespanha. *Linear Systems Theory*. Princeton Press, 2009.

[39] Joao P. Hespanha and A. Stephen Morse. Switching between stabilizing controllers. *Automatica*, 38(11):1905 – 1917, 2002.

[40] Hoffmann, G.M., Huang, H., Waslander, S.L., Tomlin, and C.J. Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control Engineering Practice*, 2011.

[41] C. Homescu, I. M. Navon, and Z. Li. Suppression of vortex shedding for flow around a circular cylinder using optimal control. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS*, (38):4369, 2002.

[42] Horowitz, M., Williamson, and CHK. The effect of Reynolds number on the dynamics and wakes of freely rising and falling spheres. *Journal of Fluid Mechanics*, 651(1):251–294, 2010.

[43] Husain, H.S., Shtern, V., Hussain, and F. Control of vortex breakdown by addition of near-axis swirl. *Physics of Fluids*, 15:271, 2003.

[44] S. R. Hutchison, P.A. Brandner, J. R. Binns, A. D. Henderson, and G. J. Walker. Development of a CFD model for an oscillating hydrofoil. *17th Australasian Fluid Mechanics Conference*, December 2010.

[45] Sham Kakade. A Natural Policy Gradient. In *Advances in Neural Information Processing Systems (NIPS14)*, 2002.

[46] Kim, S.J., Koh, K., Lustig, M., Boyd, S., Gorinevsky, and D. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):606–617, 2007.

[47] J. Kober, B. Mohler, and J. Peters. Imitation and Reinforcement Learning for Motor Primitives with Perceptual Coupling. In *From Motor to Interaction Learning in Robots*. Springer, 2009.

[48] J. Zico Kolter and Andrew Y. Ng. Policy Search via the Signed Derivative. *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[49] Konle, M., Kiesewetter, F., Sattelmayer, and T. Simultaneous high repetition rate PIV–LIF-measurements of CIVB driven flashback. *Experiments in Fluids*, 44(4):529–538, 2008.

[50] George V. Lauder and Peter G. A. Madden. Learning from Fish: Kinematics and Experimental Hydrodynamics for Roboticists. *International Journal of Automation and Computing*, 4:325–335, 2006.

[51] L. Lessard and S. Lall. Quadratic Invariance is Necessary and Sufficient for Convexity. In *submitted to 2011 American Control Conference*, 2011.

[52] James C. Liao, David N. Beal, George V. Lauder, and Michael S. Triantafyllou. The Kármán gait: novel body kinematics of rainbow trout swimming in a vortex street. *Journal of Experimental Biology*, 206(6):1059–1073, 2003.

[53] Daniel Liberzon. *Switching in Systems and Control*. Birkhauser, Boston, 2003.

[54] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 3 edition, 1999.

[55] Wei-Min Lu. A state-space approach to parameterization of stabilizing controllers for nonlinear systems. *IEEE Transactions on Automatic Control*, 40(9):1576 –1588, sep. 1995.

[56] Luenberger and D. An introduction to observers. *Automatic Control, IEEE Transactions on*, 16(6):596–602, 1971.

[57] Lupashin, S., DAndrea, and R. Adaptive fast open-loop maneuvers for quadrocopters. *Autonomous Robots*, pages 1–14, 2012.

[58] Lupashin, S., Schollig, A., Sherback, M., D'Andrea, and R. A simple learning strategy for high-speed quadrocopter multi-flips. In *2010 IEEE International Conference on Robotics and Automation*, pages 1642–1648. IEEE, 2010.

[59] S. Lupashin and R. DAndrea. Adaptive Open-Loop Aerobatic Maneuvers for Quadrocopters. In *IFAC World Congress*, volume 18, pages 2600–2606, 2011.

[60] Mellinger, D., Kumar, and V. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[61] D. Mellinger, N. Michael, and V. Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. In *Proceedings of the 12th International Symposium on Experimental Robotics (ISER 2010)*, 2010.

[62] Daniel Mellinger, Nathan Michael, Michael Shomin, and Vijay Kumar. Recent Advances in Quadrotor Capabilities. *2011 IEEE International Conference on Robotics and Automation*, May 2011.

[63] Moore, K.L., Johnson, M., Grimble, and M.J. *Iterative learning control for deterministic systems*. Springer-Verlag New York, Inc., 1993.

[64] Natarajan and B.K. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995.

[65] Newman and J.N. *Marine hydrodynamics*. The MIT press, 1977.

[66] Noack, B.R., Tadmor, G., Morzynski, and M. Low-dimensional models for feedback flow control. Part I: Empirical Galerkin models. In *2nd AIAA Flow Control Conference, Portland, Oregon, USA*, 2004.

[67] Noack, Bernd R., Schlegel, Michael, Morzynski, Marek, Tadmor, and Gilead. Galerkin Method for Nonlinear Dynamics. In Noack, Bernd R., Morzynski, Marek, Tadmor, Gilead, Pfeiffer, Friedrich, Rammerstorfer, Franz G., Salencon, Jean, Schrefler, Bernhard, Serafini, and Paolo, editors, *Reduced-Order Modelling for Flow Control*, volume 528 of *CISM Courses and Lectures*, pages 111–149. Springer Vienna, 2011. 10.1007/978-3-7091-0758-4 3.

[68] Oppenheim, A.V., Schafer, and R.W. *Discrete-time signal processing.* Pearson Education, third edition, 2010.

[69] Peter Van Overschee and Bart De Moor. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75 – 93, 1994. Special issue on statistical signal processing and control.

[70] A. D. B. Paice and J. B. Moore. On the Youla-Kucera parametrization for nonlinear systems. *Systems & Control Letters*, 14(2):121 – 129, 1990.

[71] Protas, B., Styczek, and A. Optimal rotary control of the cylinder wake in the laminar regime. *Physics of Fluids*, 14(7):2073–2087, 2002. cited By (since 1996) 33.

[72] Reni Raju and Rajat Mittal. Towards Physics Based Strategies for Separation Control over an Airfoil using Synthetic Jets. In *Proceedings of the 45th AIAA Aerospace Sciences Meeting and Exhibit*, page 13. AIAA, 8 - 11 January, Reno, Nevada 2007.

[73] A. Rantzer and A. Megretski. A convex parameterization of robustly stabilizing controllers. *Automatic Control, IEEE Transactions on*, 39(9):1802 –1808, sep. 1994.

[74] S.S. Ravindran. Reduced-Order Adaptive Controllers for Fluid Flows Using POD. *Journal of Scientific Computing*, 15(4), 2000.

[75] S.S. Ravindran. Adaptive Reduced-Order Controllers for a Thermal Flow System using Proper Orthogonal Decomposition. *Journal of Scientific Computing*, 23(6):1924–1942, 2002.

[76] Martin Riedmiller, Jan Peters, and Stefan Schaal. Evaluation of Policy Gradient Methods and Variants on the Cart-Pole Benchmark. In *Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 254–261. IEEE, 2007.

[77] John Roberts, Ian Manchester, and Russ Tedrake. Feedback Controller Parameterizations for Reinforcement Learning. In *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2011.

[78] John Roberts, Lionel Moret, Jun Zhang, and Russ Tedrake. Motor Learning at Intermediate Reynolds Number: Experiments with Policy Gradient on the Flapping Flight of a Rigid Wing. *From Motor to Interaction Learning in Robots*, pages 293–309, 2010.

[79] John W. Roberts. Motor Learning on a Heaving Plate via Improved-SNR Algorithms. Master's thesis, Massachusetts Institute of Technology, February 2009.

[80] John W. Roberts and Russ Tedrake. Signal-to-Noise Ratio Analysis of Policy Gradient Algorithms. In *Advances of Neural Information Processing Systems (NIPS) 21*, page 8, 2009.

[81] John W. Roberts, Jun Zhang, and Russ Tedrake. Motor Learning at Intermediate Reynolds Number: Experiments with Policy Gradient on the Flapping Flight of a Rigid Wing. In *From Motor to Interaction Learning in Robots*. Springer, 2009.

[82] Rotkowitz, M., Lall, and S. A characterization of convex problems in decentralized Control. *IEEE Transactions on Automatic Control*, 51(2):274 – 286, feb. 2006.

[83] Rowley and C.W. Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation Chaos in Applied Sciences and Engineering*, 15(3):997–1014, 2005.

[84] Rowley, C.W., Ilak, and M. Reduced-order models of linearized channel flow using balanced truncation. *Control and Automation, 2006. MED '06. 14th Mediterranean Conference on*, pages 1–6, June 2006.

[85] C. W. Rowley, T. Colonius, and R. M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115–129, 2004.

[86] Rumsey, C.L., Ying, and S.X. Prediction of high lift: review of present CFD capability. *Progress in Aerospace Sciences*, 38(2):145–180, 2002.

[87] M. Samimy, M. Debiasi, E. Caraballo, A. Serrani, X. Yuan, J. Little, and J. H. Myatt. Feedback Control of Subsonic Cavity Flows Using Reduced-order Models. *Journal of Fluid Mechanics*, 2007.

[88] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational Approaches to Motor Learning by Imitation. *Philosophical Transactions: Biological Sciences*, 358(1431):537–547, March 2003.

[89] Schadow, KC, Gutmark, and E. Combustion instability related to vortex shedding in dump combustors and their passive control. *Progress in Energy and Combustion Science*, 18(2):117–132, 1992.

[90] Schatz, M., and B Gunther. Computational investigation of separation control for high-lift airfoil flows. *Active Flow Control*, pages 173–189, 2007.

[91] Schrijer, F.F.J., Scarano, and F. Effect of predictor–corrector filtering on the stability and spatial resolution of iterative PIV interrogation. *Experiments in fluids*, 45(5):927–941, 2008.

[92] Wei Shyy, Yongsheng Lian, Jian Teng, Dragos Viieru, and Hao Liu. *Aerodynamics of Low Reynolds Number Flyers*. Cambridge Aerospace Series. Cambridge University Press, 2008.

[93] Siegel, S., Cohen, K., McLaughlin, T., Myatt, and J. Real-time particle image velocimetry for closed-loop flow control studies. In *41 st AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV*, 2003.

[94] Stefan G. Siegel, Jrgen Seidel, Casey Fagley, D. M. Luchtenburg, Kelly Cohen, and Thomas McLaughlin. Low-dimensional modelling of a transient cylinder wake using double proper orthogonal decomposition. *Journal of Fluid Mechanics*, 610:1–42, 2008.

[95] Mark W. Spong. Energy Based Control of a Class of Underactuated Mechanical Systems. In *Proceedings of the 1996 IFAC World Congress*, 1996.

[96] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[97] Suzuki, H., Kasagi, N., Suzuki, and Y. Active control of an axisymmetric jet with distributed electromagnetic flap actuators. *Experiments in Fluids*, 36:498–509, 2004. 10.1007/s00348-003-0756-0.

[98] G. Tadmor, M. D. Centuori, B. R. Noack, M. Luchtenburg, O. Lehmann, and M. Morzynski. Low Order Galerkin Models for the Actuated Flow Around 2-D Airfoils. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, 8 - 11 January, Reno, Nevada 2007. Paper AIAA 2007-1313.

[99] G. Tadmor, O. Lehmann, B. R. Noack, L. Cordier, J. Delville, j. P. Bonnet, and M. Morzyński. Reduced order models for closed-loop wake control. *Phil. Trans. R. Soc. A*, 369:1513 – 1624, 2011.

[100] G. Tadmor, O. Lehmann, B. R. Noack, and M. Morzyński. Galerkin Models Enhancements for Flow Control. In B. R. Noack, M. Morzyński, and G. Tadmor, editors, *Reduced-Order Modelling for Flow Control*, volume 528 of *CISM Courses and Lectures*. Springer Verlag, 2011.

[101] Russ Tedrake, Teresa Weirui Zhang, and H. Sebastian Seung. Learning to Walk in 20 Minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, CT, 2005.

[102] Tsitsiklis, John N., Blondel, and Vincent D. The Lyapunov exponent and joint spectral radius of pairs of matrices are hardwhen not impossibleto compute and to approximate. *Mathematics of Control, Signals, and Systems (MCSS)*, 10:31–40, 1997. 10.1007/BF01219774.

[103] Venkatesh, S.R., Dahleh, and M.A. On system identification of complex systems from finite data. *Automatic Control, IEEE Transactions on*, 46(2):235 –257, feb 2001.

[104] Oskar von Stryk. Users Guide for DIRCOL: A Direct Collocation Method for the Numerical Solution of Optimal Control Problems, November 1999.

[105] Wang and D. *International Journal of Control*, 73(10):890–901, 2000.

[106] Z. Jane Wang. Aerodynamic efficiency of flapping flight: analysis of a two-stroke model. *The Journal of Experimental Biology*, 211:234–238, October 2007.

[107] Willert, C., Jarius, and M. Planar flow field measurements in atmospheric and pressurized combustion chambers. *Experiments in fluids*, 33(6):931–939, 2002.

[108] Willert[1], C.E., Munson, M.J., Gharib, and M. Real-time particle image velocimetry for closed-loop flow control applications.

[109] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[110] Williamson and C H K. Vortex Dynamics in the Cylinder Wake. *Annual Review of Fluid Mechanics*, 28(1):477–539, 1996.

[111] H. S. Witsenhausen. A Counterexample in Stochastic Optimum Control. *SIAM Journal on Control*, 6(1):131–147, 1968.

[112] J. Wu and S. Lall. Nonlinear Youla parametrization and information constraints for decentralized control. In *American Control Conference (ACC), 2010*, pages 5614 –5619, Baltimore, MD, Jun 2010.

[113] D. Youla, H. Jabr, and J. Bongiorno Jr. Modern Wiener-Hopf design of optimal controllers–Part II: The multivariable case. *IEEE Transactions on Automatic Control*, 21(3):319–338, 1976.

[114] Haiqian Yu, Miriam Leeser, Gilead Tadmor, and Stefan Siegel. Real-time Particle Image Velocimetry for Feedback Loops Using FPGA Implmentation. *Journal of Aerospace Computing, Information, and Communication*, 3(2):52–62, February 2006.

[115] G. Zames. Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *Automatic Control, IEEE Transactions on*, 26(2):301 – 320, apr. 1981.