ABSTRACT

COBB, MITCHELL. Economic Iterative Learning Control with Application to Tethered Energy Systems. (Under the direction of Dr. Chris Vermillion.)

A multitude of engineered systems in the manufacturing, robotics, and energy systems communities operate in a cyclical or repetitive fashion. The preponderance of traditional tools from repetitive control and its cousin, iterative learning control (ILC), has focused solely on the tracking of a prescribed path. However, many recent and even not-so-recent repetitive processes stand to benefit from control strategies that consider some economic objective (e.g., manufacturing time, metabolic cost, or net energy generation, depending on the application).

This dissertation presents two iterative learning control frameworks for repetitive, path-following systems, both of which are tailored to explicitly consider economic performance objectives. In the first, we adapt the update law from traditional ILC so that it applies to the path shape, thereby allowing for the adaptation of the path itself rather than merely tracking a prescribed path. In the second, we adapt the lifted system representation and the update law in order to directly modify control inputs to a system in a way that allows the iteration time to vary from one iteration to the next. These methodologies differ from preexisting work in that i) allow for optimization of an economic objective, ii) allow the timing properties of a system to vary from one iteration to the next, and iii) do not require that the initial conditions be reset between iterations. Of these contributions, the second and third not only facilitate the consideration of economic performance goals (which are severely hampered by the need to keep iterating timing fixed and reset a system between repetitions) but also amount to lifting fundamental restrictions that are inherent to much of the preexisting work on ILC.

The methodologies developed in this work have been validated on a tethered energy system, whereby a high-lift tethered “kite” executes periodic flight patterns that, when optimized, produce more than an order of magnitude more power than an equivalently-sized stationary system. To this end, we present a sequence of four dynamic models of tethered energy systems. These models represent a sequence of incrementally increasingly complexity and realism while also capturing the key elements of the economic optimization problem that are important to this work. This dissertation then validates the ILC algorithms on each of these models in succession, demonstrating increased economic performance in each case.
Economic Iterative Learning Control with Application to Tethered Energy Systems

by

Mitchell Cobb

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Mechanical Engineering

Raleigh, North Carolina

2020

APPROVED BY:

Dr. Andre Mazzoleni
Dr. Kira Barton
External Member

Dr. Kenneth Granlund
Dr. Scott Ferguson

Dr. Chris Vermillion
Chair of Advisory Committee
DEDICATION

To my father, and to my mother.
ACKNOWLEDGEMENTS

I first owe a great debt of gratitude to my advisor, Dr. Chris Vermillion. There are a many people out there who understand control theory on a very deep level. However, I don't know any of them who have also mastered teaching, communication, organizational leadership, and the entire canon of South Park jokes. Chris has the unique combination of all these talents, matched with an unbeatable work ethic. He hired me when I had not taken a single course in control. I realize now that that was either an incredible leap of faith, or an extremely perceptive decision. Because he was willing to take the risk to hire me, and then take the time to teach me, I have gained so much. For that, I will be forever grateful.

I am also indebted to Dr. Kira Barton at the University of Michigan. Over the course of my work as a graduate student, she has become almost a second advisor. Although we don't meet as often as I do with Chris, she has always made time to help me. She has provided everything from critical guidance on my research, to professional guidance on getting a job in industry. Thank you.

Although at this point in my life I have probably taken courses with over one hundred professors, a few stand out in my memory and I would like to mention them specifically. Dr. Andre Mazzoleni’s course on advanced dynamics was hands-down the clearest and most concise treatment of the subject I have ever seen. I mean it when I say this, he should write a textbook. It’s a fairly unique experience as a student to take a course when you already know the first half of the subject material very well. However, that’s exactly the position I was in when I took Dr. Ferguson’s design optimization course. Rather than making the subject boring, it turned out to be a master-class on teaching a subject that often proves to be dense and nebulous in the hands of lesser instructors. Of course, I did also learn a great deal on heuristic optimization techniques in the second half of the class.

I also want to thank my lab mates, current and former. In alphabetical order, they are Alireza Bafandeh, Ali Baheri, John Borek, Josh Daniels, Joe Deese, Nihar Deodhar, Christian Earnhardt, Ben Groelke, Ben Haydon, Kartik Naik, James Reed, Blake Shepherd, and Ayaz Siddiqui. At some point along the way, each of you have helped me, whether it was edits to papers, suggestions to improve presentations, or more practical things that you’ve done to help keep the lab running, I have been able to rely on each of you. I’ve thought many times over the years about how this group of people really are the ones most familiar with my work on a day-to-day basis. As such, they have always been available for long, wandering, technical questions that may ultimately prove to only be tangentially related to our research, but are always interesting for both of us. I am grateful to have been part of a group that shared my interests and also had the knowledge, enthusiasm, and positivity to pursue those interests and support my work.

To Kasey, you know how important you have been in making this happen. With the benefit of hindsight, I can now see that meeting Kasey was a pivotal moment in my life. I went back to school to get the degree I really wanted because she motivated me to do it. She was the reason I took the risk, quit my job, bought a foreclosure (without heat or even running water), and enrolled as an undergraduate at UNCC, which felt like a step backwards at the time. It wasn’t. It was the first steps
in building the life that I wanted. Every day since then she has helped me build that life. I am so grateful.

To my parents, you have given me all of this. Nothing would have been possible without the millions of decisions you made in raising me. I am who I am because of you, and I have what I have because of you.

To everyone, thank you.
# TABLE OF CONTENTS

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

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

Chapter 1  Introduction ..................................................................... 1

Chapter 2  Prerequisites ................................................................. 7
  2.1 Notation and Dynamic Models .................................................. 7
  2.2 Iterative Learning Control ....................................................... 9
    2.2.1 Norm Optimal Iterative Learning Control ......................... 10
    2.2.2 Point to Point Iterative Learning Control ....................... 11
  2.3 Path Definitions and Pure Pursuit Algorithm ........................... 12
  2.4 Tethered Energy Systems ....................................................... 14

Chapter 3  Dynamic Models and Lower Level Controllers .................. 20
  3.1 Low Fidelity Models ............................................................. 21
    3.1.1 Four State Model ......................................................... 23
    3.1.2 Sailboat Model .......................................................... 24
    3.1.3 Path Following Algorithm ........................................... 27
  3.2 Unifoil Model ......................................................................... 28
    3.2.1 Dynamic Model .......................................................... 28
    3.2.2 Ground-Gen Spooling Controller and Path Following Flight Controller ... 32
  3.3 High Fidelity Model ............................................................. 34
    3.3.1 Preliminaries: Coordinate Systems & Important Quantities ... 34
    3.3.2 6 DoF Lifting Body MHK Model ................................... 35
    3.3.3 Tether Model ............................................................. 37
    3.3.4 Winch Model ............................................................ 38
    3.3.5 Flow Environment Modeling ......................................... 39
    3.3.6 Flight and Power Take-Off Controllers ............................ 41
    3.3.7 Power Take-off (Winch) Controller ................................. 45

Chapter 4  Economic ILC for Path Optimization ................................. 48
  4.1 Metamodel Identification ........................................................ 52
  4.2 Convergence Analysis - Ideal Case ....................................... 52
    4.2.1 Convergence Analysis - Discussion ............................... 55
  4.3 Applications .......................................................................... 56
    4.3.1 Sailboat Model .......................................................... 56
    4.3.2 Unifoil Model ............................................................ 58
    4.3.3 High Fidelity Model ..................................................... 63

Chapter 5  Economic ILC For Path Following .................................... 69
  5.1 Path Domain Lifted Model ...................................................... 71
  5.2 Receding Horizon Norm-Optimal Economic ILC ....................... 74
    5.2.1 Lifting in the Iteration Domain ....................................... 74
    5.2.2 Derivation of Optimal Learning Filters ............................ 76
  5.3 Linear Programming Based Economic ILC ............................... 78
5.4 Combined Flexible Time and Receding Horizon ILC ................................. 79
5.5 Application to Four State Model ......................................................... 80
5.6 Application to Unifoil Model ............................................................. 81

Chapter 6 Conclusions and Future Work ...................................................... 90
  6.1 Conclusions .................................................................................... 90
  6.2 Future Work .................................................................................. 91

BIBLIOGRAPHY .................................................................................... 93

APPENDIX ............................................................................................. 97
Appendix A Mathematical Tools ............................................................... 98
  A.1 Golden Section Algorithm ............................................................... 98
  A.2 Rodrigues’ Rotation Formula ............................................................ 98
  A.3 ILC Performance Weighting Matrices ............................................... 98
LIST OF TABLES

Table 4.1 Parameters used in simulation of the unifoil model when subjected to the path optimization ILC algorithm. ........................................ 60
Table 4.2 Important plant model parameters used in the high fidelity model when subjected to the path optimization ILC algorithm. ........................................ 64
Table 4.3 Important controller parameters used in the high-fidelity model when subjected to the path optimization ILC algorithm. ........................................ 65

Table 5.1 Parameter values used in simulation subjecting the four state model to the linear programming-based path following ILC algorithm. .................. 81
Table 5.2 Unifoil model parameter values used in simulation results for the receding horizon path following ILC algorithm. ................................. 84
Table 5.3 Controller parameter values used in simulation results of the unifoil model as subjected to the norm optimal path following ILC algorithm. Note that the mathematical expressions that utilize the penalty weights are given in Appendix A.3 ................................................................. 86
LIST OF FIGURES

Figure 1.1 Examples of systems that perform cyclic or iterative tasks wherein the repetitive nature of the system provides significant opportunity for learning and improvement from cycle to cycle both in the overall shape of the achieved path, and in the intra-waypoint behavior. ..................................................... 2

Figure 2.1 Generic block diagrams of different ILC control structures. In the simplest form on the top, the ILC update calculates an entire sequence of control inputs for every instant in time. In the more complex version on the bottom, the ILC update is applied in parallel with a closed loop, time-domain feedback controller. ................................................................. 10

Figure 2.2 Graphical depiction of an example path shape, $\vec{p}(\sigma, b)$, the path position, $s$, and the pure pursuit path following algorithm. The path position, $s$, is the value of the generic parametric variable, $\sigma$, corresponding to the closest point on the path. The desired direction of travel (heading) of the system is described by the angle that the vector $\vec{p}(s + \sigma_c, b) - \vec{p}(t)$ makes with the axis $\vec{x}$. 13

Figure 2.3 Examples of tethered energy systems, both the AWE system from Windlift [29] (left) and the MHK system from Minesto [30] (right) employ on-board turbines to harvest energy. Image credits: Windlift (left) and Minesto (right). 15

Figure 2.4 A top-down view of a two-dimensional tethered energy system showing the relationship of key geometric quantities involved in crosswind or cross-current flight. The apparent flow, $\vec{v}_{app} = \vec{v}_f - \vec{v}$, is the flow vector actually presented to the kite. Under steady flight, the tether tension, $\vec{F}_T$, is aligned so as to perfectly counteract the net hydrodynamic force on the system, which is a combination of the lift force, $\vec{F}_L$, and the drag force, $\vec{F}_D$. ........................................... 15

Figure 2.5 Different options for spooling in ground-gen tethered energy systems. Intercycle spooling (left) performs multiple spool-out crosscurrent laps before switching to spool in whereas intercycle (right) spools in and out during the course of a single lap. ................................................................. 16

Figure 2.6 Two possible platform configurations for a marine hydrokinetic energy (MHK) system. While the platform fixed to the ocean floor (right) is dynamically simpler, it presents significant logistical problems when accessing important components, relative to the floating platform (left). .................. 17

Figure 3.1 Important quantities in a generic model of a tethered energy system, the ground-fixed (inertial) coordinate system, $\vec{G}$, the body fixed system $\vec{B}$, the spherical coordinate system (composed of the radius, $r$, azimuth, $\Theta$, and elevation, $\Phi$), flow velocity vector, $\vec{v}_f$, and zenith angle, $\zeta$. .................. 22

Figure 3.2 Four state model approximation of a tethered energy system. The unit vectors $\hat{x}_\Theta$ and $\hat{y}_\Phi$ correspond to increasing azimuth and elevation angles in Figure 2.1, respectively and the heading of the system is measured as positive in the counter-clockwise direction starting at $\hat{x}_\Phi$. .................. 23

Figure 3.3 Velocity polar for a 3 m/s wind speed: Position along the radial axis corresponds to the maximum achievable flight speed in m/s, and position along the angular axis corresponds to the heading, $\psi$, in degrees. .................. 24
Figure 3.4 The two-dimensional sailboat model approximation of the generic tethered energy system shown in Figure 3.1. The top image depicts the boat traversing an example path in a spatially-varying flow profile chosen to mimic the flow profile experiences by a full three-dimensional system. The bottom image depicts important system quantities such as the velocity vector, $\vec{v}$, apparent wind vector, $\vec{v}_{app}$, angle of attack, $\alpha$, the angle between the sailboat centerline and the apparent wind vector, $\psi_{app}$ and the sail angle and rudder angle control inputs, $u_s$ and $u_r$................................. 25

Figure 3.5 Closed loop, path-following algorithm as applied to the sailboat model model. Note that the block diagram describing the path following algorithm in the four state model is almost exactly the same but does not include the model reference controller or the sail angle controller. ...................... 27

Figure 3.6 Schematic of unifoil model. The top figure shows a close up of the system including the wing, fuselage, horizontal stabilizers, and vertical stabilizer with rudder, body-fixed unit vectors, $\vec{x}_B$, $\vec{y}_B$, $\vec{z}_B$, and on-board turbines. The bottom figures show the relationship between the rudder control input, $u_r$, and $\alpha_r$, and the wind control input, $u_w$ and $\alpha_w$. The apparent flow velocity vector, $\vec{v}_{a}$ is defined in equation (3.23). Note that a horizontal stabilizer is also modeled, and the angles associated with it follow the same pattern as the wing. ......................... 29

Figure 3.7 Lower level path following and spooling controllers for the unifoil model. Although this preserves much of the same structure as the controllers for the low level model, as shown in Figure 3.5, a tether speed controller is introduced to determine the tether release rate if one chooses to model a ground-gen system. ................................................................. 32

Figure 3.8 State machine used to determine the raw tether speed, $u_t$. The parameters $r_{min}$ and $r_{max}$ are min and max tether lengths set by the designer, and the third switching condition, $\text{sgn}(\theta(t)) \neq \text{sgn}(\theta(t-T_d))$ triggers the transition from spool-out to spool-in when the system passes back over the zero-azimuth position, or the $\vec{x}_G$-$\vec{z}_G$ plane. The time $T_d$ is a unit time delay. 33

Figure 3.9 High level block diagram of the high-fidelity model showing the main system components, the path following flight controller, the power take-off (PTO) spooling controller, the plant model, and the environmental flow profile model. ................................................................. 42

Figure 3.10 Detailed block diagram showing the hierarchical structure of the path-following flight controller shown in Figure 3.9. ................................. 43

Figure 4.1 A generic block diagram depicting the concept of operation for path optimization iterative learning control. In this configuration, the ILC update serves as an upper level controller that prescribes a flight path geometry to the lower level path following controller in the form of basis parameters, which are encoded in the vector $b$. ................................................................. 49

Figure 4.2 Block diagram of the path-optimization ILC update structure showing three main components, the calculation of the performance over the last iteration, the recursive least squares estimation of the response surface, and the ILC-based update law. ................................................................. 50

Figure 4.3 Evolution of path geometry for a single initial condition of the sailboat model as subjected to the path-optimization ILC algorithm. ................................. 57
Figure 4.4 Evolution of basis parameters for a variety of initial conditions on the basis parameters of the sailboat model when subjected to the path-optimization ILC algorithm. ................................................................. 58
Figure 4.5 Evolution of performance index given in equation (4.27) for a variety of initial conditions of the sailboat model when subjected to the path-optimization ILC algorithm. ................................................................. 59
Figure 4.6 Three-dimensional path of the system over a single iteration both before and after optimization for the kite-based, ground-gen marine hydrokinetic (MHK) unifoil model as subjected to the path-optimization ILC. ................................................................. 61
Figure 4.7 Convergence of course geometry for the kite-based, ground-gen marine hydrokinetic (MHK) unifoil model as subjected to the path-optimization ILC. 61
Figure 4.8 Tether speed, \( \dot{r} \), tether tension, \( T \), and estimated power production over the first iteration, as well as the last iteration for the kite-based, ground-gen marine hydrokinetic (MHK) energy system as subjected to the path-optimization ILC. ................................................................. 62
Figure 4.9 Convergence of the performance index, given in (4.29), over the course of optimization for the kite-based, ground-gen marine hydrokinetic (MHK) energy system as subjected to the path-optimization ILC. ................................................................. 62
Figure 4.10 Increase in mean power production, quantified by the first term in equation (4.29), over the course of optimization for the kite-based, ground-gen marine hydrokinetic (MHK) energy system as subjected to the path-optimization ILC. 63
Figure 4.11 Summary results obtained by applying the path-optimization ILC algorithm to the high-fidelity model under a spatially and temporally constant flow profile at both 1 m/s and 2 m/s. ................................................................. 67
Figure 4.12 Summary results obtained by applying the path-optimization ILC algorithm to the high-fidelity model under a flow profile containing both a low frequency component and high-frequency, turbulent component. ................................................................. 68

Figure 5.1 Block diagram of the generic feedback + feedforward control structure used in many works on iterative learning control. Note that the ILC update law produces a control sequence for an entire iteration, not just a single time step. Thus the control input, \( u_{j+1}^{ILC} \), is really one element of the entire sequence produced by the ILC update. ................................................................. 69
Figure 5.2 Block diagram of the control structure used for the path-parameterized ILC update law. ................................................................. 80
Figure 5.3 Performance index vs. iteration number for a four-state model of an AWE system as subjected to the linear programming-based path optimization ILC algorithm. ................................................................. 82
Figure 5.4 EAR vs. iteration number, as calculated by ((2.27)) for a four-state model of an AWE system as subjected to the linear programming-based path optimization ILC algorithm. ................................................................. 82
Figure 5.5 Example figure 8 path at 1st, 11th, and 21st iteration for a four-state model of an AWE system as subjected to the linear programming-based path optimization algorithm. ................................................................. 83
Figure 5.6 Instantaneous PAR at 1st, 11th, and 21st iteration for a four-state model of an AWE system as subjected to the linear programming-based path optimization algorithm. ................................................................. 83
Figure 5.7 Path shape over the first and last iteration. Note that the achieved flight path over the last iteration tracks the waypoints much more closely than the first iteration. ................................................................. 85

Figure 5.8 Performance index over the course of 50 iterations. Note that lower values indicate better performance. Also, the negative values are only possible because of the presence of the economic, $S_x$ term in (5.22). ......................... 85

Figure 5.9 Waypoint tracking performance achieved by the unifoil model as subjected to the receding horizon path following ILC algorithm. Note that a low waypoint tracking penalty indicates better performance. ................................................................. 87

Figure 5.10 Iteration duration, mean speed and economic incentive term in $J$, as achieved by the unifoil model when subjected to the receding horizon path following ILC algorithm. For the top and bottom plot, a low value indicates better performance, while for the middle plot, a high value indicates better performance. ................................................................. 88

Figure 5.11 Power factor, $P_f$ versus iteration number as achieved by the unifoil model as subjected to the receding horizon path following ILC algorithm. A larger power factor indicates better system performance. ......................... 89
There are a huge variety of systems that perform cyclic or highly repetitive tasks. For many of these systems, performance is specified as a combination of a path tracking requirement and an economic objective. The path tracking requirement is fairly intuitive. In many systems, it is desirable to track some spatially-defined path such as a tool-path or racetrack. The second objective is fundamentally very different from the first. When maximizing (or minimizing) some economic performance objective such as profit, energy production, or efficiency, there is often no notion of a predefined reference trajectory or path, or the path is defined only in terms of starting and ending points, leaving considerable freedom for optimization in between. Instead, one wishes to find the best control inputs or system behavior, not drive the output to a predefined value.

Some example systems that fit this description include pick-and-place robots, actively controlled exoskeletons used for therapeutic rehabilitation, and tethered energy systems, examples of which are shown in Figure 1.1. In the case of the robotic manipulators, the economic objective may be control energy expenditure or the total time required to complete an operation while also following a spatially defined path that describes the desired motion of the manipulator through space. In the autonomous racing example from [1], the objective is to minimize the time required to complete a lap, while also satisfying the constraint that the system remain on the track. If an active exoskeleton like the one in [2] is being used for rehabilitation, then the objective may be to minimize the energy expended or force exerted by the machine. If it is being used for active assistance for a task, then minimizing the energy or force from the user would be a more appropriate objective. Finally, if one considers the tethered energy system on the bottom right of Figure 1.1, then the economic objective may be to maximize the energy captured by the system.

Traditional repetitive control and iterative learning control (ILC) methodologies focus mostly
Figure 1.1 Examples of systems that perform cyclic or iterative tasks wherein the repetitive nature of the system provides significant opportunity for learning and improvement from cycle to cycle both in the overall shape of the achieved path, and in the intra-waypoint behavior.

on tracking a predefined reference trajectory. That is, they seek to match the output of the system to some predefined sequence of outputs. However, for the type of systems shown in Figure 1.1 and considered in this work, reference tracking is not the primary objective of the system. A common characteristic of the economic performance indices of the examples in Figure 1.1, and other applications, is that the performance index is ultimately expressed as a single scalar value that characterizes the overall performance of the system and ultimately relates to the value (which could be expressed literally in units of dollars and cents or could simply relate directly to profitability/utility). Minimizing or maximizing this economic performance measure is often far more important than perfectly tracking a specified reference.

Several possible options exist in the literature for optimizing the performance of repetitive
systems such as these. One possibility is iterative learning control. This control technique was initially proposed in [5] to improve the tracking performance of systems that perform repetitive tasks by using information from previous attempts to improve performance on future iterations. The mathematical relationship between these past control, input, and error signals and the control input sequence for the next iteration is referred to as an update law. This initial work formed the basis of a whole branch of control theory that is still active today. Although a comprehensive treatment of the entire topic, such as those found in [6] or [7], is outside the scope of this thesis, this work relies on two key developments and therefore we address both of them in some detail.

In the initial work on iterative learning control, the focus was solely on learning in order to eliminate transient error relative to a predefined reference trajectory. In the first key development that is relevant to this work, norm optimal ILC, the focus of the algorithm was expanded in works such as [8], [9], [10], and [11], to balance reference tracking and the resulting required control effort. Then, in the second key relevant development for this work, point-to-point ILC, developed in works such as [9, 12–16], and [17], the performance index from norm-optimal ILC was adapted to consider tracking of the reference signal at a few instants in the iteration. By only considering tracking at a few key instants, where the points to be tracked are termed waypoints, the system gains additional control freedom that can be used to consider additional objectives. Because this work relies heavily on these two developments, this dissertation now discusses each in slightly more detail.

The first key development, norm-optimal ILC, recast the iterative learning control problem as an unconstrained quadratic optimization problem, where the objective function is quadratic in the state, error, and control input sequences. It was shown that for this quadratic performance index and a linear system, it was possible to derive analytical, closed-form expressions for the learning gains of the update law in terms of the weights used in the performance index and the dynamic model of the system.

The second key development, point-to-point ILC, lifts the requirement that the system tracks the reference trajectory at all instants in time and only requires the system to track a reference trajectory at a few specific waypoints. This results in additional control freedom which can be leveraged to further improve performance.

Despite these significant developments, ILC has still relied on two key assumptions, which ultimately amount to limitations on the capabilities of ILC. Specifically, most work has assumed that:

- the timing properties of an iteration, such as trial duration and waypoint arrival times, remain fixed from one iteration to the next, and

- the initial conditions of the system are reset between iterations.

When considering how to optimize systems such as those shown in Figure 1.1, these constraints prove to be highly restrictive, often times preventing any significant improvement in performance. If one would like to maximize the profitability of the robotic system, then preventing it from performing its task any faster (decreasing iteration time) is clearly problematic. When considering the
optimization of the robotic exoskeleton, allowances must be made for natural variations in human gait that occur for different tasks (shuffling, walking, jogging, etc) and different terrain (the gait cycle for flat terrain may be much faster than the gait cycle for climbing stairs). In the case of a robotic race car, the first constraint would prevent us from reducing the lap time, and the second constraint would require us to stop between each lap and rest the cars position and velocity to match the previous lap. This is obviously suboptimal for the autonomous racing application. Finally, in the case examined in this dissertation, maximizing the energy capture of a tethered energy system requires that the system fly faster. Thus, one would naturally like to complete the prescribed flight path as fast as possible, requiring one to allow the total iteration time to vary.

Nonetheless, some progress has been made in lifting these constraints. Work in [18], [19], [20], and [21] addresses aspects of the flexible-time problem via a two-stage update process. In one stage, the set of time stamps describing the system’s motion along a path are assumed constant, and the algorithm solves for the optimal control sequence. In the other stage, the path timing is optimized under the assumption that the control sequence is held constant. While the first stage takes the form of a more traditional ILC tracking problem, the second stage can be quite complex and may require solving a computationally complex nonlinear optimization problem. In contrast, this work does not require any computationally intensive nonlinear optimizations. Furthermore, work in [18–20] does not consider economic objectives, instead focusing on the traditional reference tracking and control energy expenditure objectives.

In response to the need to vary the total trial duration, work in [22] proposes an ILC framework wherein the trial duration is unknown beforehand and therefore allowed to vary from iteration to iteration. However, the work still relies on a pre-defined reference signal with known timing characteristics and does not consider economic objectives. In contrast, this work does not require a reference signal parameterized as a function of time and instead use a curve in space, referred to as a path, that is defined with respect to a parametric variable that is independent of time. Furthermore, none of the aforementioned work considers systems wherein an initial condition reset is suboptimal or infeasible.

There have been other efforts within the broader learning control community to address the aforementioned features (timing flexibility, continuous operation, and economic optimization). One leading technique is learning model predictive control (LMPC), developed in [1] and [23]. Here, an update law is used to enlarge “safe” constraint sets at each iteration, based on what is learned form the previous iteration. By enlarging the safe sets, more control options become available, and performance can be improved. This strategy has proven effective on the autonomous racing example of Figure 1.1 (bottom left), which is in fact characterized by an economic objective (lap time), flexible cycle time, and continuous operation. While shown to be tremendously effective for the autonomous racing application, the methodology dispenses with the lifted system representation, closed-form update law, and learning filters that are common to point-to-point ILC. This makes it difficult to build on an extensive body of methodological and theoretical results from point-to-point ILC in generalizing the work.
In direct response to the aforementioned challenges, this work presents two novel iterative learning control strategies that allow for variable iteration timing (waypoint arrival time and iteration duration), directly address economic optimization, and also do not require that the system be reset between iterations.

The first algorithm, referred to as *path optimization ILC*, seeks to optimize the path geometry itself. This algorithm is then composed of two key steps:

1. metamodel identification, in which a recursive least squares fit is performed to approximate the relationship between path shape and performance, and
2. an iterative learning update, in which information from past iterations (characterized via the metamodel) is used to select the path shape used in the next cycle.

The second ILC algorithm proposed in this work, referred to as *path following ILC*, seeks to directly modify the control inputs to the plant in order to improve both path tracking and the economic performance objective. This algorithm is comprised of three key steps:

1. First, the dynamic model is re-parameterized from a system that evolves with respect to time, to a system that evolves with respect to a parametric variable that describes position along a spatial curve. Note that the range spanned by this parametric variable is constant from one iteration to the next and waypoints are then intuitively defined in terms of spatial position, not time-step.

2. The path-parameterized dynamic model from step 1 is then discretized and lifted according to standard practice in iterative learning control. Here, it is not assumed that the initial conditions are reset between iterations and thus the impact of iteration-varying initial conditions are explicitly included in the lifted model.

3. Learning filters are then derived from a performance index that considers multiple future iterations. This provides control input sequences for multiple future iterations. However, only a single iteration is implemented before updating the control sequence again, thus mimicking the structure of time-domain model predictive control (MPC), also referred to as receding horizon control.

It is important to note here that processes described in step 1 and step 3 can be used independently of each other. By reparameterizing the dynamic model into path domain, as described in step 1, and using that model in an ILC update, the timing behavior of the system is inherently allowed to vary from iteration to iteration. In addition, it is possible to apply the process described in step 3 to a traditional time-domain model, without performing step 1. Thus, the path-parameterization of step 1 is the key innovation in enabling flexible time ILC, whereas the MPC-like structure introduced in step 3, referred to as *receding horizon iterative learning control* (RHILC) is the key innovation that enables continuous operation of the system, without resorting to an initial condition reset between iterations.
In order to test the validity of these two algorithms, I developed a suite of dynamic models of tethered energy systems. These models range from the relatively simple and straightforward, to more realistic models incorporating many degrees of freedom, nonlinear dynamics, and complex features such as catenary tether geometry. By validating these algorithms on a full spectrum of models, this dissertation demonstrates the potential for i) wide applicability of the algorithms and ii) applicability to realistic system models.

The contributions of this work are:

1. Development of iterative learning control for economically optimal path planning, wherein the overall path geometry is optimized to maximize an economic metric.

2. Development of flexible time iterative learning control for optimal path following, wherein the time domain behavior such as waypoint arrival times and total iteration time are permitted to vary between iterations. Of the preexisting work that does attempt to address aspects of this problem, no result addresses both the problem of varying waypoint arrival times and total iteration time. Furthermore, many of them require that the system solve complex nonlinear optimization problems. In contrast, this work only requires the traditional tools of ILC, the calculation of a lifted system matrix, and the resulting learning filters.

3. Development of receding horizon iterative learning control, which inherently considers the impact of one iteration on the next, thus eliminating the requirement that the system be reset after each iteration. None of the preexisting work on ILC derives optimal learning filters without the assumption that the system is reset between iterations. Thus, lifting this requirement in a rigorous manner constitutes a significant contribution.

4. A suite of four dynamic models of tethered energy systems of incrementally increasing complexity and fidelity.

5. The validation of the aforementioned ILC algorithms on the suite of simulation models. This serves to demonstrate the efficacy of the algorithms in optimizing an economic objective (primarily energy generation), while also following a path in an efficient manner. This validation is performed for both simple and complex system models.
This chapter introduces several key mathematical tools that are foundational for this work. Section 2.1 introduces the basic generic dynamic models and the notation used to define them. Then, Section 2.2 introduces the basic ideas of iterative learning control (ILC) and its two important variants, norm optimal ILC and point-to-point ILC. Section 2.3 details the path following algorithm that is used in the lower level flight controller of each of the dynamic models examined in this work. Finally, Section 2.4 provides a detailed explanation of operational concepts and performance characterization of tethered energy systems.

2.1 Notation and Dynamic Models

This section sets up a standard notation and defines the forms of the generic dynamic models that will be used throughout the document. In this work, the standard nonlinear, continuous-time, time-invariant dynamic model will be denoted as

\[
\dot{x}(t) = f(x(t), u(t)). \tag{2.1}
\]

Here, \(x(t) \in \mathbb{R}^{n_x}\) is the state vector, and \(u(t) \in \mathbb{R}^{n_u}\) is the control input vector. One operation used frequently throughout this work is linearization. This converts the nonlinear model of equation (2.1) to a linear model of the form

\[
\dot{x}(t) = Ax(t) + Bu(t), \tag{2.2}
\]
where $A \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_x \times n_u}$ are constant matrices. This model can then be converted to discrete time via a process called discretization. The resulting model relates the state and input vectors at the current discrete instant in time, denoted by $x(i)$ and $u(i)$, to the state vector at the next discrete instant in time, $x(i+1)$, according to the relationship

$$x(i + 1) = A_d x(i) + B_d u(i)$$

(2.3)

where the constant, discrete time matrices $A_d \in \mathbb{R}^{n_x \times n_x}$ and $B_d \in \mathbb{R}^{n_x \times n_u}$ can be calculated via a zero-order hold defined by the relationships

$$A_d = e^{A \Delta_T}$$

(2.4)

$$B_d = \int_0^{\Delta_T} e^{A \tau} B d \tau.$$  

(2.5)

Here, $\Delta_T$ is the discretization time-step. The process described here, which results in a linear, time-invariant discrete time system model, can be extended to i) the case of time-varying systems and ii) nonuniform discretization time steps, both of which will be necessary in later work in this dissertation. In the first case, $f(x(t), u(t)) \rightarrow f(x(t), u(t), t)$ in equation (2.1), therefore $A \rightarrow A(t)$ and $B \rightarrow B(t)$ in equation (2.2), which will result in $A_d \rightarrow A_d(i)$ and $B_d \rightarrow B_d(i)$ in equation (2.3). In the second case, $\Delta_T \rightarrow \Delta_i(i)$. This allows the discretization step, $\Delta_T$ to be different at every step in time, denoted by the index $i$.

We now consider one very important form of dynamic model, called the lifted system model or simply the lifted system. This dynamic model is based on an affine relationship, an initial condition state vector, $x(0) \in \mathbb{R}^{n_x}$, and a complete discrete sequence of control input vectors, $\{u(0), u(1), \ldots, u(n_t - 2), u(n_t - 1)\}$, to a complete sequence of state vectors $\{x(0), x(1), \ldots, x(n_t - 1), x(n_t)\}$.

Beginning from the time-invariant, discrete-time system of the form shown in equation (2.3), one can construct a lifted system model by simply marching forwards in time. By doing so, one can observe that an entire set of state vectors can be related to an entire sequence of input vectors via the block matrix structure

$$\begin{bmatrix}
  x(0) \\
  x(1) \\
  x(2) \\
  \vdots \\
  x(n_t - 1) \\
  x(n_t)
\end{bmatrix} =
\begin{bmatrix}
  0 & 0 & 0 & \ldots & 0 & 0 \\
  B_d & 0 & 0 & \ldots & 0 & 0 \\
  A_d B_d & B_d & 0 & \ldots & 0 & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
  A_d^{n_t - 2} B_d & A_d^{n_t - 3} B_d & A_d^{n_t - 4} B_d & \ldots & 0 & 0 \\
  A_d^{n_t - 1} B_d & A_d^{n_t - 2} B_d & A_d^{n_t - 3} B_d & \ldots & B_d & 0
\end{bmatrix}
\begin{bmatrix}
  u(0) \\
  u(1) \\
  u(2) \\
  \vdots \\
  u(n_t - 2) \\
  u(n_t - 1)
\end{bmatrix} +
\begin{bmatrix}
  1 \\
  A_d \\
  A_d^2 \\
  \vdots \\
  A_d^{n_t - 1} \\
  A_d^{n_t}
\end{bmatrix} x(0).$$

(2.6)

By defining variables for the concatenated sequence of $x$ and $u$ vectors as well as the block matrices, one can re-write this expression much more concisely as

$$x = G u + F x(0),$$

(2.7)
where \( x \in \mathbb{R}^{n_x n_t}, u \in \mathbb{R}^{n_u (n_t - 1)}, G \in \mathbb{R}^{n_x n_t \times n_u (n_t - 1)}, \) and \( F \in \mathbb{R}^{n_x n_t \times n_x} \) are defined so as to make equations (2.6) and (2.7) equivalent. Here, \( G \) and \( F \) are referred to as the “lifted” matrices, or simply as the lifted model, \( x \) is referred to as the lifted state vector, and \( u \) is referred to as the lifted control input vector. This form of dynamic model proves to be quite useful in many contexts because it is a direct, affine mapping from the control input sequence to the resulting state sequence. Furthermore, when the initial conditions are zero (i.e. \( x(0) = \vec{0} \)) or the statespace coordinate system is defined so that \( x(0) \) lies at the origin, it becomes linear. Additionally, note that one can define a lifted reference vector \( r \) similarly to \( x \) and \( u \) by concatenating the value of the reference signal \( r(t) \in \mathbb{R}^{n_r} \) at discrete instances in time.

It is important to note here that the lifted system model requires two key quantities be specified by the designer: i) the discretization time step, \( \Delta T \), and ii) the total number of time steps, \( n_t \). These two numbers effectively determine the duration spanned by the lifted vectors \( x \) and \( u \) (and thus determine the duration of an iteration). Furthermore, the manner in which they appear (\( \Delta T \) in the exponential of equations (2.4) and (2.5), and \( n_t \) in the dimensions of \( x, u, G \) and \( F \)) presents significant challenges when attempting to vary the timing characteristics of an iteration.

2.2 Iterative Learning Control

The core idea at the heart of iterative learning control is that information from previous iterations or attempts can be used to improve performance at future iterations. Two possible control structures that achieve this are depicted generically in generic block diagrams in Figure 2.1. The main idea is that in between iterations of the system (the portion of Figure 2.1 denoted as operating in time domain), the elements in the iteration domain operate to calculate a new control input sequence to be used during the next iteration.

The mathematical structure to accomplish this was initially proposed in [5]. That work developed a closed-form mathematical equation called the update law. Supposing that the index \( j \) denotes the iteration that has just finished, then \( j + 1 \) denotes the next iteration, which has not yet begun. The update law then relates:

- the control input signal from the iteration that was just completed, \( u_j(t) \),
- the state sequence obtained in the previous iteration, \( x_j(t) \),
- the reference signal that the system should track, \( r(t) \), and
- the control input signal for the next iteration, denoted as \( u_{j+1} \) on the top of Figure 2.1 and \( u_{j+1}^{\text{ilc}} \) on the bottom of Figure 2.1.

By using the lifted forms of these signals, \( x_j, u_j, r, \) and \( u_{j+1} \), one basic form of an ILC update law can be written as

\[
  u_{j+1} = u_j + k(r - x_j),
\]

(2.8)
where \( k \in \mathbb{R}^{n_u \times (n_t-1) \times n_x} \) is a matrix-valued learning gain. Note that here, the term in parentheses is the lifted error signal from the previous iteration, sometimes denoted as \( e_j \). Additionally, it should be emphasized that this update relates the actual measured signals, \( u_j \) and \( x_j \) to the next control input. Thus the system “learns” by observing the actual system, not a model of the system.

Following the seminal work, a natural question arose in the research: is there some systematic way to determine a “good” or “optimal” value for the learning gain matrix \( k \)? In order to address this question, work in publications such as \([8],[10]\), and \([11]\) developed norm-optimal ILC. This dissertation next addresses norm optimal ILC as well as another extension, point-to-point ILC, which seeks only drive the error in the system to zero at specified instants in time, where the target states at those points in time are referred to as “waypoints”.

2.2.1 Norm Optimal Iterative Learning Control

In order to derive an expression for the “best” or “optimal” learning gain in equation \((2.8)\), norm-optimal ILC first defines a performance index to express what constitutes good or optimal performance. This performance index usually takes the form of a quadratic norm of various combinations of the input, state and error signals. For example, one common performance index is

\[
J_{j+1} = u_{j+1}^T Q_u u_{j+1} + (u_{j+1} - u_j)^T Q_{\delta u} (u_{j+1} - u_j) + u (r - x_{j+1})^T Q_e (r_{j+1} - x_{j+1}),
\]  
(2.9)
where $Q_u \in \mathbb{R}^{n_u(n_t-1) \times n_u(n_t-1)}$, $Q_\delta u \in \mathbb{R}^{n_u(n_t-1) \times n_u(n_t-1)}$, and $Q_e \in \mathbb{R}^{n_u(n_t-1) \times n_u(n_t-1)}$ are diagonal, positive semidefinite matrices of tuneable weights chosen to encode the relative importance of each component. Thus, this performance index captures the relative importance of minimizing the error, the size of the control input, and the deviation in the control input signal between iterations.

Here, one can use the expression of (2.7) to relate $x_{j+1}$ to $u_{j+1}$. The result can be used in equation (2.9), thus yielding an expression for the performance of the next iteration, $J_{j+1}$ in terms of the control input over the next iteration, $u_{j+1}$,

$$J_{j+1} = u_{j+1}^T Q_u u_{j+1} + (u_{j+1} - u_j)^T Q_\delta u (u_{j+1} - u_j) + u_j^T (r - G u_{j+1})^T Q_e (r_{j+1} - G u_{j+1}).$$  (2.10)

For simplicity, I have assumed that the initial conditions, $x(0)$ are all zero here. By differentiating this expression with respect to the elements of $u_{j+1}$, setting the result equal to the zero vector and solving for $u_{j+1}$, one obtains the relationship

$$u_{j+1} = L_u u_j + L_e (r - x_j),$$  (2.11)

where the matrix-valued learning filters, $L_u \in \mathbb{R}^{n_u(n_t-1) \times n_u(n_t-1)}$ and $L_e \in \mathbb{R}^{n_u(n_t-1) \times n_u(n_t-1)}$ are given by

$$L_u = (Q_u + Q_\delta u + G^T Q_e G)^{-1} (Q_u + G^T Q_e G),$$  (2.12)

$$L_u = (Q_u + Q_\delta u + G^T Q_e G)^{-1} G^T Q_e.$$  (2.13)

Thus, the update law of equation (2.11) is explicitly based the performance index specified in equation (2.9) and first order optimally criteria. Therefore, this method provides an intuitive method to specify the performance of the system and also relates that performance specification to learning gains (in matrix form) in the ILC update law.

This work, originally pioneered in works such as [8], is integral to the work presented later in this thesis. The derivation of optimal learning filters in Section 5.2 applies exactly the same methodology used here.

### 2.2.2 Point to Point Iterative Learning Control

As mentioned previously, it is possible to gain additional control freedom by only requiring that the system track a subset of points along the prescribed reference signal. This set of points, referred to as “waypoints”, encode critical states that the system designer would like to track. For example, in pick-and-place operations, it is far more important that the system track specified states at the pick-up and place times than at intermediate times.

To incentivize this waypoint tracking behavior, the performance index of equation (2.9) can be modified slightly via the introduction of the waypoint selection matrix, $\Psi \in \mathbb{R}^{n_t \times n_{w}}$. This waypoint selection matrix is a diagonal matrix of ones and zeros that is designed to “pick off” the value of the error at points in time corresponding to waypoints. Thus, if $T_\Psi = \{i_1, i_2, ..., i_{n_w-1}, i_{n_w}\}$ is the set of time indices corresponding to the waypoints, then the block-element in the $i^{th}$ block row
and $j^{th}$ block column of $\Psi$, denoted as $[\Psi]_{i,j}$ are given by

$$[\Psi]_{i,j} = \begin{cases} \mathbb{I}_{n \times n} & i = j \text{ and } i \in T_l \\ \mathbf{0}_{n \times n} & \text{otherwise.} \end{cases} \quad (2.14)$$

Note that in the event that the designer does not want to track all states at the waypoint, which is the case when you would like the system to hit a position, but allow our algorithm to optimize the velocity, then this expression can be modified slightly. In this case, rather than using an identity matrix in the first case of equation (2.14), one can use a diagonal matrix with a one on the diagonal in positions corresponding to tracked states, and zeros elsewhere.

The resulting performance index

$$J_{j+1} = u_{j+1}^T Q_u (u_{j+1} - u_j) + u (r - x_{j+1})^T \Psi^T Q_e (r_{j+1} - x_{j+1}), \quad (2.15)$$

can then be used to derive optimal learning filters following the exact same methods presented in Section 2.2.1. The result is learning filters that are extremely similar to equations (2.12) and (2.13) except that they only seek to track the reference signal at specified points, and seek to optimize the $Q_u$ and $Q_{\delta u}$ terms in the performance index elsewhere. This method is employed later in this thesis in Chapter 5 as well as works such as [14], [15], [16], and [24].

### 2.3 Path Definitions and Pure Pursuit Algorithm

In this work, I adopt tools from iterative learning control, such as the ones introduced in the previous section, and use them to address economic optimization in path-following systems. Having introduced the prerequisite tools from iterative learning control, this dissertation now turns to three critical mathematical definitions for path-following systems, namely, path shape, path position, and waypoints. Each of these key mathematical quantities are depicted graphically in Figure 2.2. Finally, this section details a basic path-following algorithm referred to as “pure pursuit” that is used throughout this work.

The path shape is encoded by a vector-valued function, $\vec{p}(\sigma, b) \in \mathbb{R}^{2n+3}$, that is a parametric curve in two or three dimensional space. This function describes the shape of the path that one would like the system to follow. It is defined in terms of two inputs, a generic parametric variable, $\sigma \in \mathbb{R}$, and a set of scalar values, $b \in \mathbb{R}^n$, referred to as basis parameters, that define one specific instance of more general form. Different values of that path variable, $\sigma$, describe different points in space along the path. It is defined to lie in a finite range, $[\sigma_0, \sigma_f]$, where $\sigma_0 < \sigma_f$ and $\sigma_0$ corresponds to the beginning of the path and $\sigma_f$ corresponds to the end. This work assumes that $\sigma$ increases monotonically along the path from beginning to end. It also assumes that the function $\vec{p}$ is smooth and twice-differentiable with respect to $\sigma$. Note that $\vec{p}(\sigma, b)$ is a non-injective function, meaning that multiple values of $\sigma$ may correspond to the same point in space. This is the case for self-intersecting paths or closed-loop paths that start and end at the same point, such as a figure eight.
Figure 2.2 Graphical depiction of an example path shape, $\vec{p}(\sigma, b)$, the path position, $s$, and the pure pursuit path following algorithm. The path position, $s$, is the value of the generic parametric variable, $\sigma$, corresponding to the closest point on the path. The desired direction of travel (heading) of the system is described by the angle that the vector $\vec{p}(s + \sigma_c, b) - \vec{r}(t)$ makes with the axis $\hat{x}$.

In this work, the dependence on $b$ is often suppressed in the notation for clarity or if the values in $b$ do not change.

A **waypoint** is a critical point along the path that one would like the system to track. For example, in pick-and-place robotics, a waypoints might be the points in space where the robot picks up or puts down an item. A waypoint is then calculated mathematically by evaluating the path shape function $\vec{p}(\sigma, b) \in \mathbb{R}^{2\text{or}3}$ at a particular value of the path variable, $\sigma_i$. Therefore, one defines a set of waypoints, $\mathcal{S}_w$, to be the set of path variables corresponding to critical points in space, $\mathcal{S}_w = \{\sigma_1, \sigma_2, \ldots, \sigma_{n_w-1}, \sigma_{n_w}\}$. Note that through the definition of the path shape, this also defines a set of points in space, $\{\vec{p}(\sigma_1, b), \vec{p}(\sigma_2, b), \ldots, \vec{p}(\sigma_{n_w-1}, b), \vec{p}(\sigma_{n_w}, b)\}$ which can be notated using the shorthand $\{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_{n_w-1}, \vec{p}_n\}$ as shown in Figure 2.2.

The **path position**, $s$, is the value of $\sigma$ corresponding to the position along the path that is closest to the current position of the system. Thus, if at a particular time the system is located at position $\vec{r}(t) \in \mathbb{R}^{2\text{or}3}$, then the path position, $s$, is defined as

$$s(t) = \arg\min_{\sigma} d(\vec{r}(t), \vec{p}(\sigma, b)), \quad (2.16)$$

where $d(\cdot, \cdot) : (\mathbb{R}^{2\text{or}3}, \mathbb{R}^{2\text{or}3}) \rightarrow \mathbb{R}$ is an arbitrary distance metric. Note that in practice there may be no analytical solution for $s$, necessitating a numerical solution. However, because this problem is relatively simple (a single-parameter minimization), this work assumes that it can be solved quickly and efficiently. To solve this problem, this work utilizes the golden section minimization technique detailed in Appendix A.1. The relationship between $\vec{r}(t)$, $\vec{p}(\sigma, b)$, and $s$ is depicted graphically in Figure 2.2.

In order to follow the prescribed path, $\vec{p}(\sigma, b)$, an algorithm for path-following is required. In
this work, this algorithm is responsible for determining the desired direction of travel for the system. The basic idea is to determine a point ahead of our path position (described by \( s \)) and then calculate the direction necessary to head directly at that point. The original derivation of this algorithm was provided in \([25]\) but the planar, two dimensional case is described here. The extension to three dimensions is in general, system-specific and thus it is detailed for our specific system later.

Given a system moving in a plane spanned by the orthonormal basis vectors \( \vec{x} \) and \( \vec{y} \), the shape of the desired path, \( \vec{p}(\sigma, b) \), and the current path position, \( s \), then the pure pursuit path following algorithm calculates the desired heading, \( \psi_{des} \) via

\[
\psi_{des}(t) = \arctan \left( \frac{(\vec{p}(s + \sigma_c, b) - r(t)) \cdot \vec{y}}{(\vec{p}(s + \sigma_c, b) - r(t)) \cdot \vec{x}} \right),
\]

where \( \sigma_c \) is a tuneable parameter called the “lead length”. This relationship is also shown graphically in Figure 2.2.

### 2.4 Tethered Energy Systems

Traditional devices for capturing energy from wind or ocean currents consist of a turbine mounted on a large tower. While these have proven effective and reliable, they are often suboptimal. Because turbines are fixed in space, they are usually not located at the best point in the flow. Often, the winds or currents with the most available energy are located at very high altitudes or close to the surface in very deep parts of the ocean. It is these areas that represent the greatest potential for energy generation. Work in \([26]\) showed that the wind energy available at 600m was often five times that of the wind available to most towered systems. A study in \([27]\) showed that there was more energy present in the Gulf Stream off the coast of North Carolina than the entire energy demand of the state. Unfortunately, the locations with the most available Gulf Stream energy often have depths greater than 200 m. At these depths, the capital costs associated with tower construction render installation of towered systems infeasible.

Tethered energy systems, examples of which are shown in Figure 2.3, represent one possible solution to these challenges. In both airborne wind energy (AWE) systems and tethered marine hydrokinetic energy (MHK) systems, the tower is replaced by a tether (or multiple tethers) and the turbine is replaced with a high lift-to-drag kite. In addition to reducing capital costs by eliminating the need to build a rigid tower, tethered energy systems present two opportunities. First, the system can be repositioned to the optimal location in the flow profile. Second, the kite can be flown perpendicular to the prevailing flow. This \textit{crosswind} or \textit{crosscurrent} flight, initially examined in \([28]\), has been shown to significantly increase power generation. This is due to the increased magnitude of the flow velocity vector that is actually presented to the kite. This \textit{apparent} flow velocity, \( \vec{v}_{app} \), results from a combination of the free stream flow, \( \vec{v}_f \), and the velocity of the kite, \( \vec{v} \), according to the simple vector relationship

\[
\vec{v}_{app} = \vec{v}_f - \vec{v}.
\]
The geometric relationship between these three quantities is depicted graphically in Figure 2.4.

Figure 2.3 Examples of tethered energy systems, both the AWE system from Windlift [29] (left) and the MHK system from Minesto [30] (right) employ on-board turbines to harvest energy. Image credits: Windlift (left) and Minesto (right).

Figure 2.4 A top-down view of a two-dimensional tethered energy system showing the relationship of key geometric quantities involved in crosswind or crosscurrent flight. The apparent flow, \( \vec{v}_{app} = \vec{v}_f - \vec{v} \), is the flow vector actually presented to the kite. Under steady flight, the tether tension, \( \vec{F}_T \), is aligned so as to perfectly counteract the net hydrodynamic force on the system, which is a combination of the lift force, \( \vec{F}_L \), and the drag force, \( \vec{F}_D \).

Energy capture is then accomplished via either the fly-gen configuration, where turbines are mounted on the kite, or the ground-gen configuration, where the tension in the tether is used to drive a generator. In the latter method, the tether is spooled out under high tension, generating positive power, and then spooled in under low tension, consuming power. Because the power generated during spool out is much larger than the power consumed during spool in, the result is net positive energy generation over one cycle.
Additionally, there exists design freedom to choose how to schedule the spool-in and spool-out periods. Two possible options are

- **intracycle** spooling, where the tether is spooled out under multiple high-tension crosswind or crosscurrent flight cycles and then the system is transitioned into a low-tension configuration and spooled in, and

- **multi-cycle** spooling, where the tether is spooled in and out multiple times over the course of a single crosswind or crosscurrent flight cycle.

These are depicted conceptually in Figure 2.5. Each of these options present their own challenges and opportunities. While intercycle spooling does require that the system transition into and out of a high-tension flight configuration more often than intracycle spooling, it does not require that the system stop and restart the crosscurrent flight patterns, which is a complex control challenge. Additionally, intracycle spooling may result in large variations in altitude of the system. If the system is operating in an environment with flow shear, then this may present the system with wildly different flow environments over the course of one power generation cycle. This can result in a challenging aerodynamic control problem.

![Figure 2.5](image)

**Figure 2.5** Different options for spooling in ground-gen tethered energy systems. Intercycle spooling (left) performs multiple spool-out crosscurrent laps before switching to spool in whereas intercycle (right) spools in and out during the course of a single lap.

One unique feature for marine hydrokinetic energy (MHK) systems, such as the system from Minesto shown in Figure 2.3, is the opportunity to attach the system to *either* a base station anchored to the sea floor or a platform floating on the sea surface. Both of these options are depicted schematically in Figure 2.6. While the fixed platform is dynamically simpler, the floating platform may offer easier access to important mechanical components of the system such as winches and power-takeoff components. Work throughout this thesis considers systems attached to both a floating platform and an ocean-floor fixed platform.
In order to characterize the performance of these systems, initial work in [28] derived a theoretical upper bound on the power production by considering a two-dimensional quasistatic approximation of crosscurrent flight as shown in Figure 2.4. Based on this diagram, [28] analyzed the theoretical upper limits of power generation from both the ground-gen and fly-gen operation modes (referred to in that work as “lift” and “drag” modes, respectively). Because the analysis in [28] is extremely instructive in understanding the principles of operation for tethered energy systems, key elements of that publication are reproduced here.

First, a ground-gen system is analyzed using the quasisteady approximation of Figure 2.4. In this case, the tether is spooled out at a speed of $v_t$, making the effective free-stream flow speed $v_f - v_t$ (i.e. $\vec{v}_f \rightarrow \vec{v}_f - \vec{v}_t$ in Figure 2.4). Next, because the two right triangles of Figure 2.4 are similar to one another, the magnitudes of the vectors in the diagram can be related according to

$$\frac{v}{v_f - v_t} = \frac{F_L}{F_D}. \tag{2.19}$$

In the event that the kite has a large lift-to-drag ratio, $\vec{v}$ and $\vec{v}_{app}$ are similar in magnitude. Therefore,

$$v_{app} = (v_f - v_t) \frac{C_L}{C_D}, \tag{2.20}$$

where the ratio of forces from equation 2.19 has been written using the ratio of the lift coefficient, $C_L$, and drag coefficient $C_D$, which is equivalent. Using this expression for the apparent flow speed in the standard equation of lift yields the magnitude of the overall lift force, $F_L$,

$$F_L = \frac{1}{2} \rho C_L A_{ref} (v_f - v_t)^2 \left( \frac{C_L}{C_D} \right)^2, \tag{2.21}$$

where $A_{ref}$ is the fluid dynamic reference area. Next, because the lift force vector is (approximately) colinear with the tether, the tether tension is equal to the lift force. Therefore, the mechanical power
produced by spooling, \( P_G \), is given by this lift force multiplied by the tether release speed, \( v_t \),

\[
P_G = \frac{1}{2} \rho C_L A_{ref} \left( v_f - v_t \right)^2 \left( \frac{C_L}{C_D} \right)^2 v_t.
\]  

(2.22)

By differentiating this expression with respect to the tether release rate, \( v_t \), setting the result equal to zero, and solving for the spooling rate, \( v_t \), [28] shows that the optimal tether release rate is \( v_t^* = \frac{1}{3} v_f \) which results in a maximum theoretical power production for a ground-gen system, \( P_G^* \), of

\[
P_G^* = \frac{2}{27} \rho A_{ref} \frac{C_L^3}{C_D^2} v_f^3.
\]  

(2.23)

Next, work in [28] analyzed a fly-gen system with on-board turbines that exert a simple drag on the system. In that case, the tether release rate, \( v_t \), is zero, and equation 2.20 is modified slightly to include both the drag coefficient of the kite, \( C_{D,K} \), and the drag coefficient of the turbine \( C_{D,T} \). This yields the expression

\[
v_{app} = v_f \frac{C_L}{C_{D,K} + C_{D,T}}.
\]  

(2.24)

The power captured via the on-board turbine is then modeled by simply multiplying the apparent flow speed with the drag force. This yields

\[
P_F = \frac{1}{2} \rho A_{ref} C_D \left( v_f \frac{C_L}{C_{D,K} + C_{D,T}} \right)^3.
\]  

(2.25)

Differentiating this expression with respect to the turbine drag coefficient, \( C_{D,T} \), setting the result equal to zero and solving for the turbine drag coefficient yields the optimal turbine drag coefficient, \( C_{D,T}^* = \frac{1}{4} C_{D,K} \). Using this expression in equation 2.25 yields the theoretical maximum power generation of a fly-gen system,

\[
P_F^* = \frac{2}{27} \rho A_{ref} \frac{C_L^3}{C_{D,K}^2} v_f^3.
\]  

(2.26)

Interestingly, this is exactly the same as for the ground-gen system, which is given in equation 2.23. Thus, the theoretical upper limit of power production is exactly the same for a fly-gen and a ground gen system (i.e. \( P_G^* = P_F^* = P^* \)).

To quantify energy production in a non-dimensional manner that lends insight into the benefit of crosswind motion, this work uses a metric that is termed the energy augmentation ratio (EAR). This number is a ratio of the amount of energy generated by the system under crosswind flight, to the amount of energy that the same system would have generated if it had been motionless with a fixed power coefficient. Over a single lap, denoted by the index \( z \), it is calculated as:

\[
EAR = \frac{\int_{t_{i,z}}^{t_{f,z}} \lVert \bar{v}_{app} \rVert^3 \, dt}{\int_{t_{i,z}}^{t_{f,z}} \lVert \bar{v}_f(t) \rVert^3 \, dt}.
\]  

(2.27)
where the $t_i$ and $t_f$ are the initial and final times of the iteration, respectively. The term in the denominator is proportional to the amount of energy generated by a turbine pointed directly into the wind in a stationary system, whereas the term in the numerator is proportional to (with the same proportionality constant) the amount of energy produced by the system under crosswind flight. Note that this work also occasionally refers to the quantity

$$\text{PAR} = \frac{\|\vec{v}_{app}\|^3}{\|\vec{v}_f\|^3},$$

(2.28)

termed the power augmentation ratio (PAR) or also the instantaneous power factor.
Successful crosswind flight for a tethered energy system requires addressing four challenges, specifically,

1. generating an efficient figure-eight crosswind path,

2. controlling the lifting body’s position and attitude (orientation) so that it tracks the prescribed path,

3. controlling the lifting body’s attitude via control surfaces so that it maximizes the power production, and

4. in the case of a ground-gen system, selecting a sequence of spooling speeds in order to maximize power production.

While all of these are critical for a successful implementation, each of them independently represents a challenging control problem. Therefore, rather than attempting to address the full problem initially, this thesis details a sequence of models and associated control strategies of increasing complexity and fidelity. Thus, the solution of the full complex problem is reduced to a series of smaller problems with incremental increases in difficulty.

The low fidelity models of the subsequent section have been developed in order to simplify challenges 2-3, thus allowing us to focus on challenge 1. Then the unifoil model presented in the subsequent section increases the model fidelity and controller complexity incrementally by
introducing select parts of challenges 2-3. Finally, the highest fidelity model, detailed at the end of
this chapter, includes all of the elements associated with all of the challenges above.

A generic diagram representing a tethered energy system is shown in Figure 3.1. This diagram
depicts several important quantities that will be referenced throughout this section. These include:

- The ground-fixed coordinate system, $\mathcal{G}$, composed of the point $G$ and the orthonormal unit
  vectors $\vec{x}_G$, $\vec{y}_G$, and $\vec{z}_G$.

- The body-fixed coordinate system, $\mathcal{B}$, composed of the point $B$, located at the center of mass
  of the kite, and the orthonormal unit vectors $\vec{x}_B$, $\vec{y}_B$, and $\vec{z}_B$.

- The flow velocity vector, $\vec{v}_f$. Although it is shown as a single vector in Figure 3.1, in some
  models the flow field is spatially and temporally varying, thus in those cases it would be better
  represented by a different vector at every location in space and every instant in time. This
  would be denoted mathematically as $\vec{v}_f(t, \vec{r})$, where $\vec{r}$ is the position in the flow field being
  considered.

- The spherical coordinate system, comprised of a radius, $r$, an azimuth angle, $\Phi$, and an
  elevation angle, $\Theta$. Note that each of these coordinates have a unit vector associated with
  them. The set of these three unit vectors along with the point $B$ comprise a coordinate frame
  referred to as the tangent frame and described in more detail later.

- The zenith angle $\zeta$, which is the complement of the elevation angle.

Based on this generic diagram, I have developed a sequence of four dynamic models of increasing
complexity:

- two lower-fidelity models, termed the four-state and sailboat models, respectively,

- the medium-fidelity “unifoil” model, and

- a higher fidelity model that does not use any simplifying assumptions about the kinematics
  that are present in the previous models.

We now turn to a detailed description of each of these models and their associated lower-level
controllers.

### 3.1 Low Fidelity Models

The lower fidelity simulation models used in this work have been developed as simplified analogs
that capture the essential features of a generic tethered energy system, such as the one shown
in Figure 3.1. However, these models simplify many of the nonlinear dynamics found in more
complicated models. This then allowed us to focus on developing the iterative learning optimization
algorithms which form the core contributions of this work, without having to simultaneously address
the important but very complex lower level control problems of a real system.
In developing these low fidelity models, two critical simplifying assumptions are applied to the generic model shown in Figure 3.1. Specifically, it is assumed that:

1. The model is kinematically constrained to move on the surface of a sphere centered on the base station. This is the “taut-tether” assumption commonly found in the literature.

2. The system moves in the direction that it is pointing on the surface of the sphere (i.e., the familiar “no slip” condition). Note that although this prevents geometric sideslip, it does not prevent hydrodynamic sideslip.

In the simplest model, referred to as the four state model, full, three-dimensional system is distilled down to a planar system with simple longitudinal dynamics. While these dynamics are qualitatively correct, they do not fully reflect a real system. Therefore, I have developed a second lower fidelity model, termed the sailboat model. While both models are planar approximations of a three dimensional tethered energy system, the key difference between them is how mobility is achieved. In the four-state model, achievable steady-state velocity in any direction is characterized via a velocity polar, and velocity dynamics are described through a first-order filter driven by the steady-state velocity from the velocity polar. In the sailboat model, the actual fluid-dynamic forces on the kite are computed and used to derive second order dynamics.
3.1.1 Four State Model

In developing this highly simplified analog of a three-dimensional system, I utilize the taut-tether approximation to restrict the motion of the system to lie only on the surface of a sphere. I then consider motion in two directions: the azimuthal and elevation directions, denoted in Figure 3.1 by the angles $\Theta$ and $\Phi$. By considering motion only in these directions, the system can be approximated as planar, as shown in Figure 3.2. Note that because only motion in the $\Theta$ and $\Phi$ directions is considered, the velocity vector lies in this plane and the direction of travel, of the system can be described by a single angle, $\psi$, which is also shown in Figure 3.2.

![Figure 3.2 Four state model approximation of a tethered energy system. The unit vectors $\vec{x}_\Theta$ and $\vec{y}_\Phi$ correspond to increasing azimuth and elevation angles in Figure 2.1, respectively and the heading of the system is measured as positive in the counter-clockwise direction starting at $\vec{x}_\Phi$.](image)

We then model the AWE system’s flight speed according to a velocity polar, which encodes the steady-state flight speed as a function of the system’s heading relative to the prevailing flow. An example velocity polar for an AWE system, depicted for a wind speed of $3 \text{ m/s}$ is shown in Figure 3.3. The radial distance of the polar curve represents the steady-state flight velocity for any particular heading angle. Because the wing under consideration is symmetric, the polar curve is symmetric about the vertical axis.

The control input for the system is then the desired direction of travel, or heading setpoint, denoted by $u$. It is assumed here that a lower-level controller manipulates control surfaces to drive this heading to its setpoint.

Ultimately, the above approximations result in a simplified 2D model, consisting of four states: two for position, one for the speed of the system, and one for the heading of the system:

$$
\begin{align*}
\dot{r}_x &= v \cos(\psi), \\
\dot{r}_y &= v \sin(\psi), \\
\dot{v} &= \frac{1}{\tau_v} (V_{SS}(\psi) - v), \\
\dot{\psi} &= \frac{1}{\tau_\psi} (u - \psi).
\end{align*}
$$ (3.1)
Figure 3.3 Velocity polar for a 3 m/s wind speed: Position along the radial axis corresponds to the maximum achievable flight speed in m/s, and position along the angular axis corresponds to the heading, $\psi$, in degrees.

In these expressions, $v$ represents the speed of the AWE system, $V_{SS}(\psi)$ represents the steady-state speed for a particular heading, as obtained from the velocity polar, $\tau_v$ represents a lumped time constant associated with the translational dynamics, and $\tau_{\psi}$ represents a lumped time constant associated with the closed-loop rotational dynamics.

3.1.2 Sailboat Model

This model again applies the taut-tether and no-slip approximations, therefore considering motion in two directions: the azimuthal and elevation directions, denoted in Figure 3.1 by the angles $\Theta$ and $\Phi$. As with the four-state model, this reduces the system to two degrees of translational motion, which can be described by latitude and longitude. Multiplying these angular coordinates by $r$, a two dimensional system is obtained, as shown in Figure 3.4.

To appropriately model the wind in our two dimensional system, one must account for a key difference between the orientation of the wind relative to a sailboat and the orientation of the wind relative to an actual AWE system. Specifically, while the relevant components of wind for a sailboat are the components parallel to the plane of motion, the relevant component of the wind for an AWE system is the component in the radial direction. The magnitude of this component dissipates with increasing values of $\Theta$ or $\Phi$, which gives rise to the term “wind window” in tethered energy systems. This describes the area of the azimuth-elevation space where the radial component of the wind is strongest. Thus, to model the wind in the sailboat model in a way that correctly reflects the dynamics of a tethered energy system, the components of the wind in the $x_\Theta$ and $y_\Phi$ directions, $v_{w,\Phi}$ and $v_{w,\Theta}$ should be taken as the radial component of the wind, which varies with the azimuthal
Figure 3.4 The two-dimensional sailboat model approximation of the generic tethered energy system shown in Figure 3.1. The top image depicts the boat traversing an example path in a spatially-varying flow profile chosen to mimic the flow profile experiences by a full three-dimensional system. The bottom image depicts important system quantities such as the velocity vector, $\vec{v}$, apparent wind vector, $\vec{v}_{app}$, angle of attack, $\alpha$, the angle between the sailboat centerline and the apparent wind vector, $\psi_{app}$ and the sail angle and rudder angle control inputs, $u_s$ and $u_r$. 
position of the system, \( x_\Theta \), as follows:

\[
v_{w,\phi}(x_\Theta, y_\Phi) = \begin{cases} 
v_{\max} \cos\left(\frac{x_\Theta}{l_t}\right) \cos\left(\frac{y_\Phi}{l_t}\right), & x_\Theta < l_t \cos(\Phi) \\
0, & \text{otherwise}, \end{cases}
\]

(3.2)

\[
v_{w,\theta}(x_\Theta, y_\Phi) = \begin{cases} 
-v_{\max} \sin\left(\frac{x_\Theta}{l_t}\right) \cos\left(\frac{y_\Phi}{l_t}\right), & x_\Theta < l_t \cos(\Phi) \\
0, & \text{otherwise} \end{cases}
\]

(3.3)

where \( l_t \) is the length of the tether, which is a tuneable parameter. The orientation of the hull, which is specified by \( \psi \), evolves according to the following dynamic model:

\[
\dot{\psi} = \omega \tag{3.4}
\]

\[
\dot{\omega} = \frac{k_r}{I_R} v^2 u_r, \quad -\frac{\pi}{2} \leq u_r \leq \frac{\pi}{2}. \tag{3.5}
\]

Here, \( \omega \) represents the hull’s angular velocity, \( v \) represents the speed of the AWE system (sailboat), and \( u_r \) represents the rudder angle. The constant parameters \( k_r \) and \( I_R \) represent a lumped rotational fluid dynamic damping coefficient and the hull’s inertia, respectively.

Because the taut-tether constraint forces the system move on the surface of a sphere, only the components of lift and drag vectors that are tangent to the sphere play a role in the translational dynamics. These lift and drag forces, denoted by \( F_L \) and \( F_D \), respectively, are modeled to depend upon the the angle of attack, \( \alpha \), as follows:

\[
F_L(\alpha) = k_L \alpha \| \vec{v}_{\text{app}} \|^2, \tag{3.6}
\]

\[
F_D(\alpha) = \left( k_D + k_D \alpha^2 \right) \| \vec{v}_{\text{app}} \|^2. \tag{3.7}
\]

As shown in the bottom of Figure 3.4, \( \alpha \) represents the angle between the apparent wind vector, \( \vec{v}_{\text{app}} \), and the sail (whose orientation, specified by the control input \( u_s \), is analogous to the orientation of the AWE lifting body).

Through the no-slip constraint, the hull velocity vector is aligned with its heading \( \psi \). Thus, the translational equations of motion can be written as

\[
\dot{v} = \frac{1}{m} \left( F_L(\alpha) \sin(\psi_{\text{app}}) - F_D(\alpha) \cos(\psi_{\text{app}}) \right), \tag{3.8}
\]

\[
\dot{x}_\Theta = v \cos(\psi), \tag{3.9}
\]

\[
\dot{y}_\Phi = v \sin(\psi), \tag{3.10}
\]

where the apparent wind heading, \( \psi_{\text{app}} \), is measured relative to the sailboat centerline, in the two dimensional plane, as illustrated in Figure 3.4. Because of the constraint that the hull velocity is aligned with the heading, \( \psi \) is analogous to the velocity angle introduced for AWE systems in [31].
3.1.3 Path Following Algorithm

In the prior sections, I detailed two mathematical models of simplified versions of tethered energy systems. I now describe critical components of the closed-loop path following algorithm used to make the system track a figure-eight path.

In the case of the path-following algorithm for the four-state model, the problem is greatly simplified by the asymptotically stable rotational dynamics of the plant model. Thus, in order to track a path, the path following controller is simply comprised of two calculations:

- the calculation of the current path position, which is obtained in this work by applying the golden section minimization technique to equation (2.16) at every time step, and
- the calculation of the desired heading angle, $\psi_{des}$, which is obtained by directly applying equation (2.17).

In order to address the sailboat model, I have extended this algorithm to accommodate the more complex rotational dynamics seen in equations (3.8), (3.9), and (3.10) by introducing a linearizing model reference feedback controller. The resulting block diagram is seen in Figure 3.5.

![Figure 3.5](image)

**Figure 3.5** Closed loop, path-following algorithm as applied to the sailboat model. Note that the block diagram describing the path following algorithm in the four state model is almost exactly the same but does not include the model reference controller or the sail angle controller.

The collection of all of the blocks in this diagram (other than the plant model) are collectively referred to as the lower-level controller. This lower-level controller must adjust the rudder, $u_r$, in order to follow the path described by the path shape function, $\bar{p}(\sigma, b)$, and must adjust the sail angle, $u_s$, in order to efficiently generate longitudinal motion. The sail angle, $u_s$, is chosen to maximize net longitudinal force, according to equation (3.8), which is accomplished by controlling the sail angle of attack to:

$$\alpha_{opt} = \frac{k_{L1}}{2k_{D1}} \tan \psi_{sp}.$$  \hspace{1cm} (3.11)

Through the direct relationship between $\alpha$ and $u_s$, the optimal sail angle is given by $u_s = \psi_{app} - \alpha_{opt}$.
Waypoint following is accomplished by first calculating a desired heading, $\psi_{\text{des}}$, according to equation (2.17) from Section 2.3, then using a simple feedback linearizing model reference controller to regulate the hull’s orientation to that setpoint. Specifically, I use a second-order, relative degree two reference model describing desirable heading angle tracking performance is given by:

$$\frac{\psi_m(s)}{\psi_{\text{des}}(s)} = \frac{1}{(\tau s + 1)^2},$$  \hspace{1cm} (3.12)

where $\psi_m(t)$ is the reference model output, which is designed to follow $\psi_{\text{sp}}(t)$ with time constants of $\tau$. This results in a control law given by:

$$u_r(t) = \frac{I_R}{k_r \nu^2} \left( \frac{1}{\tau^2} (\psi_{\text{des}} - \psi) - \frac{2}{\tau} \omega \right).$$ \hspace{1cm} (3.13)

Here, $k_r$ is the lumped coefficient from equation (3.5). This control law then renders the closed-loop system linear and the error dynamics asymptotically stable about the origin.

### 3.2 Unifoil Model

The Unifoil model was developed as an extension to simpler models to include more realistic, three-dimensional effects. Although the model preserves the taut-tether and no-slip kinematic constraints, which ultimately results in kinematics that match those developed in [32], it does not map the system into two dimensions. Unlike the work in [32], however, the dynamics are driven by fluid dynamic forces and moments, although they are more complex than the previous four-state and sailboat models.

#### 3.2.1 Dynamic Model

This model is based on the design of a rigid wing system, similar to the system shown in the right of Figure 2.3. It is shown schematically in Figure 3.6. This tethered energy system is characterized by five elements:

1. a single large wing,
2. a horizontal stabilizer, whose lift and drag properties are lumped with the main wing,
3. a vertical stabilizer (with a rudder), positioned several meters behind and perpendicular to the wing,
4. a fuselage that rigidly connects the wing and the vertical stabilizer, and
5. counter-rotating on-board turbines.

This model can be (and is) adapted to model a ground-gen system by leaving out the effects of the counter-rotating turbines, and adding dynamics in the radial direction.
Figure 3.6 Schematic of unifoil model. The top figure shows a close up of the system including the wing, fuselage, horizontal stabilizers, and vertical stabilizer with rudder, body-fixed unit vectors, $\vec{x}_B$, $\vec{y}_B$, $\vec{z}_B$, and on-board turbines. The bottom figures show the relationship between the rudder control input, $u_r$, and $\alpha_r$, and the wind control input, $u_w$, and $\alpha_w$. The apparent flow velocity vector, $\vec{v}_a$, is defined in equation (3.23). Note that a horizontal stabilizer is also modeled, and the angles associated with it follow the same pattern as the wing.
As before with the low-fidelity models, the taut-tether and no-slip constraints, are preserved. However, the no slip constraint is now applied to a system moving on the surface of a sphere. The resulting kinematics are reflective of a system with (i) a stiff tether, (ii) limited sideslip, and (iii) a high lift/weight ratio.

It should be emphasized that this model does account for a non-zero hydrodynamic sideslip angle (i.e., the angle between the apparent flow vector and the longitudinal plane); however, it does not allow for geometric slip (i.e., the system only moves in the direction that it is pointing). Under these constraints, the system can be completely described by three variables and their first derivatives. These are the azimuth angle, \( \Phi \), elevation angle, \( \Theta \), and the heading angle, \( \psi \), which is the complement of the velocity angle used in [31]. These quantities evolve according to the dynamic equations

\[
\dot{r} = \frac{1}{\tau} (u_t - r), \tag{3.14}
\]

\[
\dot{\Phi} = \frac{v \cos(\psi)}{r \cos(\Theta)}, \tag{3.15}
\]

\[
\dot{\Theta} = \frac{v}{r} \sin(\psi), \tag{3.16}
\]

\[
\dot{v} = \frac{1}{m + m_{\text{add}}} F(x, u), \tag{3.17}
\]

\[
\dot{\psi} = \omega, \tag{3.18}
\]

\[
\dot{\omega} = \frac{1}{I_{zz} + I_{zz,\text{add}}} M_z(x, u), \tag{3.19}
\]

where the first equation amounts to a lumped-parameter winch model, capturing the rotational inertia properties of a winch. Additionally, \( r \) is the length of the tether, \( m \) is the mass of the system, \( m_{\text{add}} \) is the added mass of the system, \( I_{zz} \) is the rotational inertia about the body \( z \) axis, \( F_x \), is the net force in the \( \vec{x}_{B} \) direction, and \( M_z \) is the net moment about \( \vec{z}_{B} \). A schematic depiction of the system is given in Figure 3.6.

Note that the first state, the radial distance, \( r \), is constant in the case of a fly-gen system (because the tether is not spooled in and out) and thus the \( r \)-dynamics are omitted when modeling a system with on-board turbines. Furthermore, in the case of a ground-gen system, turbine drag effects are not included in the calculation of longitudinal force, \( F_x \).

The dynamics of this model are driven by forces and moments that result from hydrodynamic lift and drag. In order to obtain expressions for these forces and moments, I first define a body-fixed coordinate system. This coordinate system is assumed to be located at the center of mass, as depicted previously in Figure 3.1.

In order to constrain the system to the sphere, I define the body-fixed unit vectors according to
where \( \vec{\Phi} = -\sin(\Phi)\vec{x}_G + \cos(\Phi)\vec{y}_G \) is the unit vector in the direction of increasing azimuth. Note that the function \( R: (\mathbb{R}^3, \mathbb{R}^3, \mathbb{R}) \rightarrow \mathbb{R}^3 \) is Rodrigues’ rotation formula, provided in appendix A.2, which describes the rotation of one vector about a unit vector by an angle. Here, the azimuthal unit vector is rotated about the outwards unit vector, \( \vec{z}_B \), by the heading angle, \( \psi \), to obtain \( \vec{x}_B \).

To calculate the longitudinal force, \( F_x \), and turning moment, \( M_z \), I model three fluid dynamic surfaces: a wing, a horizontal stabilizer, and a vertical stabilizer. These are denoted by the superscripts \( w, h, \) and \( v \), respectively. Each of these three surfaces is then subjected to the apparent flow that results as a combination of the free stream flow of speed \( v_f \) in the \( \vec{x}_G \) direction:

\[
\vec{v}_{app} = v_f \vec{x}_G - v \vec{x}_B. \tag{3.23}
\]

This apparent flow is then used to calculate the angle of attack of each of the three surfaces:

\[
\alpha_w = \text{atan2}(-\vec{v}_{app} \cdot \vec{x}_B, \vec{v}_{app} \cdot \vec{z}_B) + u_w(t) \tag{3.24}
\]

\[
\alpha_r = \text{atan2}(-\vec{v}_{app} \cdot \vec{x}_B, \vec{v}_{app} \cdot \vec{y}_B) + u_r(t) \tag{3.25}
\]

where \( \text{atan2} \) is the two-argument inverse tangent. Note that \( \alpha_w = \alpha_h \). The net force in the \( \vec{x}_B \) direction is then given by

\[
F_x = \frac{1}{2} \rho \| \vec{v}_a \|^2 (A^w_{ref} (C^w_L(\alpha_w)\vec{u}^w_L + C^w_D(\alpha_w)\vec{u}^w_D) + A^h_{ref} (C^h_L(\alpha_h)\vec{u}^h_L + C^h_D(\alpha_h)\vec{u}^h_D)) + A^v_{ref} (C^v_L(\alpha_v)\vec{u}^v_L + C^v_D(\alpha_v)\vec{u}^v_D)) \cdot \vec{x}_B, \tag{3.26}
\]

where \( \rho \) is the fluid density, \( A^w,h,v_{ref} \) are the fluid dynamic reference areas, \( C^w,h,v_L(\cdot) \) and \( C^w,h,v_D(\cdot) \) are the lift and drag coefficients as a function of angle of attack, \( \vec{u}^w,r \) are the lift direction unit vectors for the wing and vertical stabilizer respectively, and \( \vec{u}_D \) is the drag direction unit vector. The lift direction unit vectors are defined using Rodrigues’ rotation formula as

\[
\hat{\vec{u}}^w_L = R\left( \hat{\vec{u}}^x_D, -\vec{y}_B, \frac{\pi}{2} \right), \tag{3.27}
\]

\[
\hat{\vec{u}}^r_L = R\left( \hat{\vec{u}}^y_D, \vec{x}_B, \frac{\pi}{2} \right)
\]

where \( \vec{v}_{app} \) and \( \vec{u}_D^x \) and \( \vec{u}_D^z \) denote the result of projecting \( \vec{u}_D \) into the \( \vec{x}_B - \vec{z}_B \) or \( \vec{x}_B - \vec{y}_B \) plane and normalizing the result.

The net moment about \( \vec{z}_B, M_z \), is defined similarly to equation (3.26) but includes cross products.
of the force vectors with their respective moment arms, \( \vec{r}_w \), \( \vec{r}_h \), and \( \vec{r}_r \), which are the vectors from the center of mass to the aerodynamic centers of each surface. Note that I assume the fluid dynamic center of the main wing to be at the center of mass. Therefore it does not exert a moment.

Finally, note that all of the fluid dynamic coefficients are implemented as lookup tables. The entries are calculated based on rectangular planform wings. The lift and drag coefficients are first obtained from XFOIL \[33\], and then the drag coefficients are corrected using an Oswald efficiency factor to account for lift-induced drag. In this work, the main wing and horizontal stabilizer were modeled with a NACA 2412 airfoil, and the vertical stabilizer was modeled with an HT05 airfoil.

### 3.2.2 Ground-Gen Spooling Controller and Path Following Flight Controller

The lower level controllers applied to the unifoil model retain many of the individual components from the pure pursuit path following algorithm of Section 2.3 and the resulting controller of Section 3.1.3. The resulting complete controller is shown in Figure 3.7. This lower level controller is comprised of individual controllers for the tether spool speed, main wing, stabilizer, and rudder. Because this model introduces the ability to spool tether in and out, modeling the behavior of a ground-gen tethered energy system, these components are now embedded in a switched structure. This switched structure is shown in the state machine of Figure 3.8.

The tether speed command, \( u_t(t) \), switches between a constant spool-in speed, \( u_{in} \), and a spool-out speed calculated as

\[
 u_{out}(t) = \lambda \frac{v_f(t)}{\sin(\Phi(t))},
\]

where \( \lambda \) is a tuneable parameter between 0 and 1, \( v_f \) is the known flow speed of the surrounding fluid. Here, \( u_{out}(t) \) is chosen so that the horizontal component of the spool-out speed matches the...
optimal spool-out speed derived in [28].

Switching between spool-out and spool-in is controlled via the state machine shown in Figure 3.8. Because the raw speed command, $u_t$, produced by this state machine is passed through a low pass filter in the plant, unrealistically sudden changes in the spooling speed are avoided.

![Figure 3.8 State machine used to determine the raw tether speed, $u_t$. The parameters $r_{min}$ and $r_{max}$ are min and max tether lengths set by the designer, and the third switching condition, $\text{sgn}(\theta(t)) \neq \text{sgn}(\theta(t - T_d))$ triggers the transition from spool-out to spool-in when the system passes back over the zero-azimuth position, or the $\hat{x}_G - \hat{z}_G$ plane. The time $T_d$ is a unit time delay.]

The controller dictating the angle of the main wing, relative to the plane tangent to the sphere, $u_{w}(t)$, switches between a controller designed to hold a constant angle of attack during spool-out, and a controller that attempts to reduce the tether tension resulting from the main wing to nearly zero during spool-in.

During spool-in, the target angle of attack is selected to maintain a small tether tension. Therefore, it is necessary to estimate the relationship between tether tension and angle of attack. This controller estimates the tether tension resulting from the main wing by approximating the lookup tables used to calculate lift and drag coefficients in the plant, $C_{l_{w}}(\alpha)$ and $C_{d_{w}}(\alpha)$, with closed form linear and quadratic approximations. Specifically, it approximates tether tension as

$$T_W = \frac{1}{2} \rho v^2_{\text{App}} A_{\text{ref}} (k_{l0} + k_{l1} \alpha_{w}) \cos(\gamma_{w}) + (k_{d0} + k_{d1} \alpha_{w} + k_{d2} \alpha_{w}^2) \sin(\gamma_{w}), \quad (3.29)$$

where $k_{l0}$, $k_{l1}$, $k_{d0}$, $k_{d1}$, and $k_{d2}$ are the coefficients of the best fit line to the linear region of the wing $C_{l}(\alpha)$ table and the coefficients of the quadratic fit to the wing $C_{d}(\alpha)$ table. The variables $\gamma_{w}$ and $\gamma_{r}$ are related to their respective angles of of attack and control inputs by

$$\gamma_{w} = \alpha_{w} - u_{w}, \quad (3.30)$$

$$\gamma_{r} = \alpha_{r} - u_{r}. \quad (3.31)$$

The value $\alpha^*$ that will produce no tether tension is calculated as the smallest of the two solutions to equation (3.29). The commanded wing angle during reel-in is calculated from the relationship shown in $u_{w}^{\text{in}} = \gamma_{w} - \alpha^*$.
The angle of attack during spool-out, $u_{w}^{out}$, is heuristically chosen to provide efficient flight, quantified by a large lift-to-drag ratio, while ensuring that the system maintains positive tether tension, quantified by a large lift coefficient. The full form of the switched wing angle controller is then

$$
\begin{align*}
\ u_{w}(t) = & \begin{cases} 
\ u_{w}^{in}(\gamma_{w}(t)), & u_{t}(t) \geq 0 \\
\ u_{w}^{out}(\gamma_{w}(t)), & \text{otherwise.}
\end{cases}
\end{align*}
$$

(3.32)

The stabilizer controller operates in a similar fashion during spool-in, rotating the stabilizer about its span to minimize drag. However, during spool-out it aligns the stabilizer to the fuselage (i.e. zero incidence angle).

The rudder controller follows the a pure pursuit feedback linearizing model reference control strategy discussed the previous sections, with one small difference. The heading setpoint, $\psi_{des}$, is calculated using the great-sphere heading between the current position of the system and the point ahead of the system on the path, $\vec{p}(s+\sigma_{c},b)$. This calculation follows the standard great-circle navigation mathematics as provided in [34] and is given by

$$
\psi_{des} = \frac{\pi}{2} - \text{atan2}(-\cos(\theta_{p})\sin(\theta) + \sin(\theta_{p})\cos(\theta)\cos(\phi_{p} - \phi), \cos(\theta)\sin(\phi_{p} - \phi)),
$$

(3.33)

where $\phi_{p}$ and $\theta_{p}$ are the azimuth and elevation angles associated with the target point along the path, $\vec{p}(s+\sigma_{c},b)$.

### 3.3 High Fidelity Model

The high-fidelity system model seeks to realistically characterize all six degrees of freedom of a tethered energy system without any of the simplifying assumptions used in the previous models. It includes three core elements:

1. A rigid lifting body wherein forces and moments are calculated from lift, drag, buoyancy, and gravity.

2. A tether model comprised of a series of non-compressive spring-dampers subject to fluid dynamic drag, buoyancy, and gravity. One end of the tether is attached to the lifting body and exerts a force on it, and the other is fixed to a platform, termed the ground station.

3. A winch, modeled as a simple combination of first order filters, rate limiters, integrators and lumped efficiency factors, which is used to estimate instantaneous power production.

#### 3.3.1 Preliminaries: Coordinate Systems & Important Quantities

The plant model and lower-level controller are described using three coordinate systems. Two of these systems, the ground and body systems, $\overrightarrow{G}$ and $\overrightarrow{B}$, are shown in Figure 3.1. The third system, the tangent frame coordinate system $\overrightarrow{T}$, is centered at the point $B$ but uses the spherical unit vectors
associated with the ground frame coordinate system. Specifically, the unit vectors of this coordinate system are calculated at a generic spatial position, \( \vec{r} \in \mathbb{R}^3 \), according to

\[
\vec{i}_T = \frac{\vec{r}_T}{\|\vec{r}_T\|}, \quad \vec{j}_T = \frac{d\vec{r}_T}{d\vec{r}_G} \times \frac{\vec{r}_G}{\|\vec{r}_G\|}, \quad \vec{k}_T = \frac{d\vec{r}_G}{d\vec{r}_G} \times \frac{\vec{r}_G}{\|\vec{r}_G\|},
\]

where \( \vec{r} \in \mathbb{R}^3 \) is the vector pointing from \( G \) to \( B \) which describes the location of the system.

The design of a path-following flight controller for this system, presented later, relies heavily on two geometric quantities that are derived from the coordinate systems. These are the velocity angle, \( \gamma \in \mathbb{R} \), and tangent roll angle, \( \xi \in \mathbb{R} \). Both of these quantities are calculated using the tangent plane, which is defined to be the plane spanned by \( \vec{j}_T \) and \( \vec{k}_T \) with \( \vec{i}_T \) and \( \vec{k}_T \) as unit vectors.

Intuitively, the velocity angle describes the direction that a system is moving, on the sphere centered at \( G \), with radius \( \|\vec{x}\| \). Mathematically, it is the angle between the projection of the velocity vector onto the tangent plane, and \( \vec{k}_T \) (or local north). This is consistent with the velocity angle used in other works, such as [35]. Therefore, the velocity angle, \( \gamma \), corresponding to a three-dimensional velocity vector, \( \vec{v} \in \mathbb{R}^3 \) is given by

\[
\gamma(\vec{v}) = \tan\left(\frac{\vec{v} \cdot \vec{k}_T(\vec{x})}{\vec{v} \cdot \vec{j}_T(\vec{x})}\right).
\]

Note that once again, the dependence on \( \vec{x} \) has been omitted for clarity.

The tangent roll angle, \( \xi \), is the angle that determines the component of hydrodynamic lift that contributes to turning in the tangent plane. Conceptually, \( \xi \) is the angle between the body unit vector, \( \vec{j}_B \) and the tangent plane. This is calculated as

\[
\tan(\xi(\vec{x})) = \frac{\vec{j}_B \cdot \vec{k}_T(\vec{x})}{\sqrt{(\vec{j}_B \cdot \vec{j}_T(\vec{x}))^2 + (\vec{j}_B \cdot \vec{k}_T(\vec{x}))^2}}.
\]

Note that once again, the dependence on \( \vec{x} \) has been omitted for clarity.

### 3.3.2 6 DoF Lifting Body MHK Model

The state variables describing the position and orientation (and rates of change of the position and orientation) of coordinate system \( \vec{B} \) relative to coordinate system \( \vec{G} \) evolve according to standard nonlinear equations of motion:

\[
\dot{\vec{\mu}} = h(\vec{\mu}, \vec{\Omega}) \tag{3.37}
\]

\[
J \dot{\vec{\Omega}} = \vec{M}_{N,e}(t) - \vec{\Omega} \times J \vec{\Omega} \tag{3.38}
\]

\[
\dot{\vec{x}} = R(\vec{\mu}) \vec{v} \tag{3.39}
\]

\[
M \ddot{\vec{v}} = \left( \vec{F}_{N,e}(t) - \vec{\Omega} \times \vec{v} \right) \tag{3.40}
\]
Here, the orientation of the kite coordinate system, $\mathcal{B}$, relative to the ground coordinate system, $\mathcal{G}$, is described by the vector of conventional Tait-Bryan angles, $\vec{\mu} \triangleq [\phi \ \theta \ \psi]^T$, where $\phi$ is roll, $\theta$ is pitch, and $\psi$ is yaw. The matrix $J \in \mathbb{R}^{3 \times 3}$ is the inertia matrix, and $\vec{\lambda}_{Net}$ is the sum of all applied moments expressed in the $\mathcal{B}$ frame. The position vector, $\vec{x}$, is expressed in the $\mathcal{G}$ frame. The vector $\vec{v}$ is the associated velocity, expressed in the $\mathcal{B}$ frame. The matrix $R \in \mathbb{R}^{3 \times 3}$ is the rotation matrix, calculated based on $\vec{\mu}$, that describes the relative orientation of $\mathcal{B}$ with respect to $\mathcal{G}$. The variable $M \in \mathbb{R}^{3 \times 3}$ is the diagonal mass matrix, $\vec{A}_{Net}$ is the sum of all forces applied to the kite expressed in the $\mathcal{B}$ frame, and $\vec{\omega} \triangleq [\omega_x \ \omega_y \ \omega_z]^T$ is the angular velocity of $\mathcal{B}$ relative to $\mathcal{G}$. Finally, the function $\vec{h}(\vec{\mu}, \vec{\omega})$ is the standard nonlinear function that relates the angular velocity $\vec{\omega}$ and Euler angles $\vec{\mu}$ to the rates of change of the Euler angles. This can be found in standard reference texts such as [36].

The kite is subject to forces and moments resulting from four fluid dynamic surfaces (a port wing, starboard wing, horizontal stabilizer and vertical stabilizer), buoyancy, gravity, and the tether. These forces and moments are calculated as:

$$
\vec{F}_{Net} = \vec{F}_{Thr} + (V \rho - m) g \vec{v}_G + \frac{1}{2} \rho A_r \sum_{i_a=1}^{4} ||\vec{v}_{a,i_a}||^2 \left(C_{L,i_a} \vec{u}_{L,i_a} + C_{D,i_a} \vec{u}_{D,i_a}\right), \tag{3.41}
$$

$$
\vec{M}_{Net} = \frac{1}{2} \rho A_r \sum_{i_a=1}^{4} ||\vec{v}_{a,i_a}||^2 \vec{v}_{a,i_a} \times \left(C_{L,i_a} \vec{u}_{L,i_a} + C_{D,i_a} \vec{u}_{D,i_a}\right), \tag{3.42}
$$

where in equation (3.41), the first term is the force exerted at the center of mass by the tether on the lifting body, the second term describes the net buoyant force, and the last term describes the fluid dynamic forces. Here, $V$ is the volume of the kite, $\rho$ is the fluid density, $m$ is the mass of the kite, and $g$ is the acceleration due to gravity. Here, it is assumed that both the center of buoyancy and the tether attachment point are close to the center of mass, and thus do not contribute an appreciable moment in equation (3.42).

The index $i_a \in \{1, 2, 3, 4\}$ refers to each of the four independent fluid dynamic surfaces, namely the port wing, starboard wing, horizontal stabilizer and vertical stabilizer. Therefore, the resulting force depends on the apparent flow at the aerodynamic center of each surface, $\vec{v}_{a,i_a}$, which is calculated as:

$$
\vec{v}_{a,i_a}(t) = \vec{v}_f(t, \vec{x} + \vec{r}_{a,i_a}) - (\vec{v} + \vec{\omega} \times \vec{r}_{a}), \tag{3.43}
$$

where $\vec{v}_f(t, \cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^3$ is the spatially and temporally varying flow profile discussed in detail later in Section 3.3.5, and $\vec{r}_{a,i_a}$ is the vector from the center of mass of the kite to the fluid dynamic center of the $i_a$th surface. The fluid dynamic coefficients of equations (3.41) and (3.42) are obtained by modeling each fluid dynamic surface independently in the Athena Vortex Lattice (AVL) software of [37] and are parameterized as functions of the associated control surface deflections, $\delta_{i_a}$, as:

$$
C_{L,i_a}(\vec{v}_{a,i_a}, \delta_{i_a}) = C_{L_0,i_a}(\vec{v}_{a,i_a}) + C_{L_1,i_a} \delta_{i_a} + C_{L_2,i_a} \delta_{i_a}^2
$$

$$
C_{D,i_a}(\vec{v}_{a,i_a}, \delta_{i_a}) = C_{D_0,i_a}(\vec{v}_{a,i_a}) + C_{D_1,i_a} \delta_{i_a} + C_{D_2,i_a} \delta_{i_a}^2
$$

where the control sensitivity coefficients, $C_{L_1,i_a}$, $C_{L_2,i_a}$, $C_{D_1,i_a}$, and $C_{D_2,i_a}$, are obtained from a quadratic
fit of the discrete results from AVL. Note that the control surface deflections are limited to a range of ±30° to reflect actuator limitations. The spanwise lift coefficient distributions, \( C_{L,i}(y) \), obtained from the software, are heuristically corrected to account for nonlinear stall behavior that is not accurately accounted for in AVL.

Finally, the unit vectors describing the direction of the lift and drag forces are calculated from the apparent flow direction vector at the \( i_a \)th hydrodynamic center:

\[
\vec{u}_{D,i_a} = \frac{\vec{v}_{a,i_a}}{\|\vec{v}_{a,i_a}\|},
\]

(3.45)

\[
\vec{u}_{L,i_a} = \begin{cases} 
0 & 0 & -1 \\
0 & 0 & 0 \\
1 & 0 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 0 
\end{cases} \begin{bmatrix} \vec{a}_{D,i_a}^x \\ 0 \\ \vec{a}_{D,i_a}^z \end{bmatrix}^T \frac{1}{\|\vec{a}_{D,i_a}^x \times \vec{a}_{D,i_a}^z\|} \quad i_a \neq 4,
\]

(3.46)

\[
\vec{u}_{D,i_a} = \begin{cases} 
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 
\end{cases} \begin{bmatrix} \vec{a}_{D,i_a}^x \\ \vec{a}_{D,i_a}^y \\ 0 \end{bmatrix}^T \frac{1}{\|\vec{a}_{D,i_a}^x \times \vec{a}_{D,i_a}^y\|} \quad i_a = 4,
\]

where the components of the drag direction vector are given by the dot product with the appropriate unit vector of the \( \hat{B} \) coordinate system, \( \vec{u}_{D,i} = \vec{u}_{D,i_a} \cdot (\cdot)_{\hat{B}} \). Note that \( i_a = 4 \) refers to the vertical stabilizer, thus requiring the case structure of equation (3.46).

### 3.3.3 Tether Model

In this work, I adopt the tether model used in [38] and [39]. The tether is modeled as a series of \( N_n - 1 \) massless, cylindrical, non-compressive spring-damper links that connect \( N_n \) lumped nodal point masses, referred to as “nodes”. Each of these links is subjected to buoyancy, gravity, and fluid drag. The forces acting on the tether are first calculated at the center of each link and then distributed to each node to model the dynamics. The net buoyant force at the center of the \( i_l \)th link, \( \vec{F}_{B,i_l} \), is given by

\[
\vec{F}_{B,i_l} = \frac{1}{2} \left( \rho - \rho_T \right) \pi r_T^2 \frac{I_T}{N_n - 1} \vec{v}_{a,i_l} \cdot \vec{a}_{i_l} \quad (3.47)
\]

where \( i_l \in \{1, 2, \ldots, N_n - 2, N_n - 1\} \) refers to the link being considered, \( \rho_T \) is the density of the tether, \( r_T \) is the radius of the tether cross section, \( I_T \) is the unspooled tether length, \( N_n \) is the number of nodes, and \( g \) is the acceleration of gravity. The fluid drag at the center of the \( i_l \)th link, \( \vec{F}_{D,i_l} \), is calculated as

\[
\vec{F}_{D,i_l} = \frac{1}{2} \rho C_{D,T} A_{P,i_l} \vec{v}_{a,i_l} \cdot \vec{v}_{a,i_l} \quad (3.48)
\]

where \( C_{D,T} \) is the drag coefficient of the tether, \( A_{P,i_l} \) is the projected area of the \( i_l \)th link, and \( \vec{v}_{a,i_l} \) is the apparent flow velocity at the center of the \( i_l \)th link. In calculating the projected areas, \( A_{P,i_l} \), apparent flow velocities, \( \vec{v}_{a,i_l} \), and spring-damper forces, it first useful to calculate the unit vector...
pointing from the \(i_l\)th node to the \((i_l + 1)\)th node, \(\vec{u}_{T,i_l}\), where

\[
\vec{u}_{T,i_l} = \frac{x_{T,i_l+1} - x_{T,i_l}}{\|x_{T,i_l+1} - x_{T,i_l}\|}.
\]  

(3.49)

The apparent flow velocities and projected areas can then be calculated as

\[
\vec{v}_{a,i_l}(t) = \vec{v}_f(t, \frac{1}{2} (x_{T,i_l+1} + x_{T,i_l})) - \frac{1}{2} (\vec{v}_{T,i_l+1} + \vec{v}_{T,i_l}),
\]  

(3.50)

\[
A_{P,i_l} = 2 r_T \frac{l_T}{N_n-1} \|\vec{u}_{T,i_l} \times \frac{\vec{v}_{a,i_l}}{\|\vec{v}_{a,i_l}\|}\|
\]  

(3.51)

where \(x_{T,i_l}\), \(\vec{v}_{T,i_l}\), \(x_{T,i_l+1}\), and \(\vec{v}_{T,i_l+1}\) are the positions and velocities of the \(i_l\)th and \((i_l + 1)\)th nodes respectively.

Finally, the nonlinear spring-damper force in the \(i_l\)th link is given by

\[
\vec{F}_{T,i_l}^2 = \begin{cases} \vec{0} & \|x_{T,i_l+1} - x_{T,i_l}\| < \frac{l_T}{N_n-1}, \\ \vec{F}_{T,i_l}^2 & \text{otherwise}, \end{cases}
\]  

(3.52)

where

\[
\vec{F}_{T,i_l}^2 = \left(-E_y \frac{\pi r_T^2}{l_T} (N_n-1) \left(\|x_{T,i_l+1} - x_{T,i_l}\| - \frac{l_T}{N_n-1}\right) - 2 \zeta \sqrt{E_y \frac{\pi r_T^2}{l_T} m (\vec{v}_{T,i_l+1} - \vec{v}_{T,i_l}) \cdot \vec{u}_{T,i_l}} \right) \vec{u}_{T,i_l},
\]  

(3.53)

where \(E_y\) is the Young’s modulus of the tether, and \(m\) is a tuneable mass, combined with the damping ratio \(\zeta\) define the size of the damping force. Note that the node attached to the ground is assumed to be fixed, and thus does not have any dynamics associated with it, and the dynamics of the node attached to the kite are dictated by kite. Thus, the positions and velocities of the \(N_n-2\) intermediate nodes, indexed by \(i_n \in \{2, 3, \ldots, N_n-2, N_n-1\}\) where \(i_n = 2\) refers to the node adjacent to the ground node and \(i_n = N_n-1\) refers to the node adjacent to the kite, are given by:

\[
\dot{x}_{T,i_n} = \vec{v}_{T,i_n},
\]  

(3.54)

\[
\dot{\vec{v}}_{T,i_n} = \frac{N_n}{\pi r_T^2 l_T} \left(\vec{F}_{i_n+1}^{Net} - \vec{F}_{i_n}^{Net}\right),
\]  

(3.55)

where \(\vec{F}_{i_n}^{Net} = \vec{F}_{i_n}^B + \vec{F}_{i_n}^D + \vec{F}_{i_n}^T\).

3.3.4 Winch Model

The total length of unspooled tether, \(l_T\), is calculated according to a filtered saturation-plus-integrator lumped parameter model. First, the commanded tether speed \(u_T(t)\) is saturated to be within the range from \(u_{min}\) to \(u_{max}\). The result is then passed through a first order filter with time constant \(\tau_w\) to obtain the achieved tether release speed, \(\tilde{u}_T\).
The instantaneous estimate of power production, \( P(t) \), is calculated from the achieved tether speed, \( \tilde{u}_T(t) \), and the net force on the ground node, as

\[
P(t) = \begin{cases} 
\eta_m \| \vec{F}_{Net} \| \tilde{u}_T(t), & \tilde{u}_T(t) \leq 0, \\
\frac{1}{\eta_{gen}} \| \vec{F}_{Net} \| \tilde{u}_T(t), & \text{otherwise}
\end{cases}
\]  

(3.56)

where \( \eta_m \) is a lumped motor efficiency and \( \eta_{gen} \) is a lumped generator efficiency.

### 3.3.5 Flow Environment Modeling

The flow field \( \vec{v}_f(t, \vec{r}) : (\mathbb{R}, \mathbb{R}^3) \rightarrow \mathbb{R}^3 \), which is characterized as a function of time, \( t \), and spatial position, \( \vec{r} \), is computed as the superposition of a low-frequency flow profile and high-frequency turbulence model, as:

\[
\vec{v}_f(t, \vec{r}) = \vec{v}_{hi}(t, \vec{r}) + \vec{v}_{lo}(t, \vec{r}),
\]  

(3.57)

where \( \vec{v}_{hi}(t, \vec{r}) \) represents the high-frequency, turbulent-flow component and \( \vec{v}_{lo}(t, \vec{r}) \) represents prevailing flow profile, which varies according to much longer time constants. In both of the following sections, data is calculated or provided at set of discretized spatial points. Whenever the flow profile is evaluated at intermediate points, the relevant data is linearly interpolated.

#### 3.3.5.1 Low-Frequency Ocean Modeling

The low frequency velocity profile, which accurately characterizes the prevailing flow over long time durations, was obtained from the Mid-Atlantic Bight South-Atlantic Bight Regional Ocean Modeling System (MSR) hindcast model. The MSR model was generated by North Carolina State University’s Ocean Observing and Modeling Group [40] and provides current profiles at 42 different locations in the Gulf Stream at 25-meter vertical resolution. At each location, flow velocity vectors, \( \vec{v}_{lo} \), are provided along a single vertical column of water (i.e., the \( \vec{k}_G \) direction).

#### 3.3.5.2 Modeled High-Frequency Turbulent Variability

The turbulent high-frequency components of the ocean currents are calculated based on a discretization of the flow velocity’s power spectral density (PSD) equation. Specifically, the model leverages fundamental techniques described in [41] to generate a spatiotemporally varying turbulence profile that can be applied to the hydrodynamic center of each component in the dynamic model. Based on inputs of turbulence intensity, time-averaged flow velocity profile (obtained from the MSR model), a specified frequency range, standard deviations, spatial correlation coefficients for the flow velocities, and a grid of 3D positions, the model outputs a set of time-varying velocity vectors, one at each of the specified grid points. The continuous PSD of the flow velocity, \( G(f) \), where \( f \) represents frequency, is given as:

\[
G(f) \propto f^{-3/2},
\]  

(3.58)
which implies $G^m$, the one-sided PSD, is equal to:

$$G^m(f) = A_m f^{\frac{-5}{3}}.$$  \hspace{1cm} (3.59)

Here, $A_m$ is a constant defined by the equation:

$$A_m = \frac{2 \bar{U}^2 T_m^2}{3 \left[ \frac{1}{f_{\text{min}}} - \frac{1}{f_{\text{max}}} \right]^3},$$  \hspace{1cm} (3.60)

where $m$ is an index to the $u$, $v$, or $w$ velocity components, $f_{\text{min}}$ and $f_{\text{max}}$ define the frequency range of the turbulence, $\bar{U}$ is the magnitude of the time-averaged flow velocities, $\bar{u}$, $\bar{v}$, and $\bar{w}$, defined as:

$$\bar{U} = \sqrt{\bar{u}^2 + \bar{v}^2 + \bar{w}^2},$$  \hspace{1cm} (3.61)

and where turbulence intensity, denoted by $T_m$, is equal to:

$$T_m = \frac{\sigma_m}{\bar{U}},$$  \hspace{1cm} (3.62)

where the standard deviations in the flow velocity components, $\sigma_m$, (one each for the the axial, cross-current, and down directions) are calculated as:

$$\sigma_u = \frac{T_u}{\sqrt{1 + P^2 + Q^2}}, \sigma_v = P \sigma_u, \text{ and } \sigma_w = Q \sigma_u.$$  \hspace{1cm} (3.63)

Here, $P$ and $Q$ are constants.

Correlated velocity components are then generated by a discretized one-sided PSD equation, $\vec{s}^m(\vec{f}) = G^m(\vec{f}) \delta_f$, where $\vec{f}$ is a vector of tuneable frequencies, chosen to capture the characteristic frequencies of the flow field, $\vec{f} = [f_1 \ f_2 \ ... \ f_n]$, which are assumed to be evenly spaced at an interval of $\delta_f$. A coherence function, $C_{pq}$, defining the flow component’s correlation between any two grid nodes (indexed by the variables $p, q \in \{1, 2, \ldots, N_g - 1, N_g\}$ where $N_g$ is the total number of grid points), is defined by:

$$C_{pq}(\vec{f}) = \exp\left( -\frac{R_c \Delta r_{pq} \vec{f}}{\bar{U}} \right),$$  \hspace{1cm} (3.64)

where $\Delta r_{pq}$ is the distance between any grid points and $R_c$ is a coherence decay constant. The amplitude of the turbulent velocity component, $S^m$, is written as:

$$S^m_{pq}(\vec{f}) = 2C_{pq}(\vec{f}) A_m f^{\frac{-5}{3}} \delta_f.$$  \hspace{1cm} (3.65)

We then calculate a velocity weighting matrix for each flow component, $m \in \{u, v, w\}$. This matrix,
\( \mathbf{H}^m(\vec{f}) \in \mathbb{R}^{N_g \times N_g} \), has component elements \( H_{pq}^m \) that are calculated according to

\[
\begin{align*}
H_{11}^m(\vec{f}) &= S_{11}^m(\vec{f}) = S_{21}^m(\vec{f}) = H_{11}^m(\vec{f})^2, \\
H_{22}^m(\vec{f}) &= (S_{22}^m(\vec{f}) - H_{21}^m(\vec{f})^2)^2, \\
H_{pq}^m(\vec{f}) &= \left( S_{pq}^m(\vec{f}) - \sum_{l=1}^{q-1} H_{pl}^m(\vec{f}) H_{ql}^m(\vec{f}) \right) / H_{qq}^m(\vec{f}), \\
H_{qq}^m(\vec{f}) &= (S_{qq}^m(\vec{f}) - \sum_{l=1}^{q-1} H_{ql}^m(\vec{f})^2)^2.
\end{align*}
\] (3.66)

Here, \( H_{pq}^m \) is the element in the \( p \)th row and the \( q \)th column of \( \mathbf{H}^m \). The velocity weighting matrix, \( \mathbf{H}^m(\vec{f}) \), is then used to calculate analytical expressions for the velocity components, \( u \), \( v \), and \( w \), as functions of time. The amplitude of the turbulent velocity component contributed by each frequency component, \( i_f \in \{1, 2, \ldots, n_f - 1, n_f\} \) can be represented as

\[
m_{pq}^* = \sum_{p=1}^{a} H_{pq}^m(f_i) e^{i \theta_{i_f} p},
\] (3.67)

where \( \theta_{i_f} \) is a randomly chosen phase angle between 0 and 2\( \pi \). Because \( m_{pq}^* = |m_{pq}^*| e^{i \theta_{i_f} p} \), where \( \theta_{i_f} \) is the phase angle associated with each frequency component at each grid point, \( p \), \( m_{ij}^* \) can be converted from frequency domain to time domain, where each fluctuating velocity component is denoted as:

\[
m_q(t) = \sum_{i_f=1}^{n_f} |m_{ij}^*| \sin(2\pi f_i^* t + \theta_{ij} q).
\] (3.68)

The net result of this calculation is a set of \( N_g \) three-dimensional time-varying flow velocity vectors. The grid points used in the calculation, along with the time-varying velocity vectors, are then used as a lookup table to form the function \( \vec{v}_{hi}(t, \vec{r}) \).

### 3.3.6 Flight and Power Take-Off Controllers

The flight control strategy is responsible for ensuring that the kite robustly tracks a prescribed figure-eight crosscurrent flight path. A block diagram of the complete system is shown here in Figure 3.9. This controller is composed of a two critical components:

- a path following flight controller that articulates the control surfaces in order to follow the prescribed path, \( \vec{p}(\sigma, \vec{b}) \), and
- a power take-off (PTO) controller that selects a tether release or retraction speed at every instant in order to extract power from the system.
3.3.6.1 Path Following Flight Controller

This controller contains four levels, each of which accepts feedback from the plant in calculating its outputs. This modular, hierarchical control structure is based on work in [42] and is partitioned into the following blocks:

1. A path-following block that accepts the path geometry as defined by the basis parameters, \( b \), and outputs a desired velocity angle, \( \gamma_{\text{des}} \), as defined in equation (3.35).

2. A tangent roll angle selection block, which accepts a desired velocity angle and outputs a desired tangent roll angle, which is the angle that dictates the component of hydrodynamic lift that contributes to turning in order to follow the path.

3. A desired moment selection block, which accepts the tangent roll setpoint and side slip angle setpoint, and outputs a desired moment vector represented in the body frame \( \overrightarrow{B} \).

4. A control allocation module, which accepts the desired moment vector and computes the required control surface deflections to be actuated by the ailerons, elevators, and rudder of the kite.

3.3.6.2 Path-Following

Given a prescribed path, \( \vec{p}(\sigma, b) \), the path following block first finds the current path position, \( s \), according to the same method used previously in the unifoil model. It then calculates a three dimensional vector that is taken to represent the desired velocity of the system. This desired velocity vector is taken to be a weighted average between the perpendicular vector, \( \vec{p}_\perp^* \), and the parallel
Figure 3.10 Detailed block diagram showing the hierarchical structure of the path-following flight controller shown in Figure 3.9.

vector, $\hat{p}^*_\parallel$. Intuitively, the parallel vector is the vector that perfectly aligns with the path, and the perpendicular vector is the vector from the current position, to the closest point on the path, which is by definition, perpendicular to the path. Mathematically, the perpendicular vector is given by

$$\hat{p}^*_\perp = \frac{\hat{p}_\perp}{\|\hat{p}_\perp\|} \quad \text{where} \quad \hat{p}_\perp = \begin{bmatrix} (\hat{p}(s, b) - \bar{x}) \cdot \hat{j}_T(\bar{x}) \\ (\hat{p}(s, b) - \bar{x}) \cdot \hat{k}_T(\bar{x}) \\ 0 \end{bmatrix},$$

(3.69)

where $\hat{j}_T(\bar{x})$ and $\hat{k}_T(\bar{x})$ are defined in equation (3.34). The parallel vector, $\hat{p}^*_\parallel$, is a unit vector that lies parallel to the path at the path variable corresponding to the closest point on the path, $s$ and is calculated by

$$\hat{p}^*_\parallel = \frac{\hat{p}_\parallel}{\|\hat{p}_\parallel\|}, \quad \text{where} \quad \hat{p}_\parallel = \frac{d\hat{x}}{d\sigma}{|}_{s=\sigma}.$$

(3.70)

The desired velocity unit vector, $\vec{v}_{\text{des}}$, is then calculated as the linearly weighted sum of the perpendicular and parallel vectors according to

$$\vec{v}_{\text{des}} = \left(1 - \frac{\overline{a}(s)}{a_0}\right)\hat{p}^*_\parallel + \frac{\overline{a}(s)}{a_0}\hat{p}^*_\perp.$$

(3.71)

Here, $a_{\text{int}}(s)$ is the internal angle between current position of the system, $\bar{x}$, and the closest point on the path, $\hat{p}(s, b)$. Here, $a_0$ serves as an upper limit on the possible angle used in the weighting. Intuitively, this means that if the angle between the system and the path is more than $a_0$, then the weighting will be entirely on the second term, making the system head directly toward the closest point on the path.

The desired velocity angle, which is the output of the leftmost block in Figure 3.10, is then given by $\gamma(\vec{v}_{\text{des}})$, where $\gamma(\cdot)$ is the velocity angle calculated according to equation (3.35).
3.3.6.3  Tangent Roll Angle Selection

The next stage of the flight controller maps this desired velocity angle to a desired tangent roll angle, \( \xi_{des} \). Recall that the tangent roll angle, \( \xi \), describes the orientation of the system relative to the tangent plane and determines the amount of fluid dynamic force that contributes to turning on a sphere of radius \( \|\vec{x}\| \) centered at \( G \).

The desired tangent roll angle is calculated using saturated proportional control based on the error in the velocity angle, specifically:

\[
\xi_{des} = \min\{\max\{k_\gamma (\gamma(t) - \gamma(\vec{v}_{des})), \xi_{min}, \xi_{max}\}, \]

where \( k_\gamma \) is the proportional gain. An error signal is then calculated as \( e_\xi(t) = \xi(\vec{j}_B(t)) - \xi_{des} \), where the current tangent roll angle, \( \xi(\vec{j}_B(t)) \), is calculated using equation (3.36).

3.3.6.4  Desired Moment Vector Selection

The basic premise behind the selection of a desired moment vector is that the rolling moment can be utilized to control tangent roll angle, \( \xi \), and yawing moment can be utilized to drive aerodynamic side slip angle, \( \beta \), to zero. The PTO controller articulates the elevator to passively trim the system to a high angle of attack, resulting in large tether tension during spool out and a low angle of attack, resulting in low tether tension, during spool in. Therefore, it is desirable that the deflection of the ailerons and rudder contribute negligible or zero pitching moment. This helps ensure that the PTO controller will be solely responsible for the pitch behavior. Ultimately, the desired moment vector is given by two PID controllers,

\[
\hat{M}_{des} = \begin{bmatrix}
k_{p_\xi} e_\xi(t) + k_{i_\xi} \int_0^t e_\xi(\tau)d\tau + k_{d_\xi} \dot{e}_\xi(t)
0
k_{p_\beta} \beta(t) + k_{i_\beta} \int_0^t \beta(\tau)d\tau + k_{d_\beta} \dot{\beta}(t)
\end{bmatrix},
\]

where \( \beta(t) \) is the fluid dynamic side slip angle.

3.3.6.5  Control Allocation Module

In order to map the desired moment vector to control surface deflections, a linearized approximation of the nonlinear mapping from deflections to hydrodynamic moments is inverted. This approximation is calculated by neglecting the effect of angular velocity on the apparent flow at each fluid dynamic surface, then linearizing to obtain an expression in the following form:

\[
\hat{M}_{net} = \hat{M}_o + A \begin{bmatrix}
\delta_a \\
\delta_x \\
\delta_r
\end{bmatrix},
\]

(3.74)
where $\delta_a$, $\delta_e$, and $\delta_r$ represent the deflection angles of the ailerons, elevator, and rudder, respectively. The variable $\vec{M}_o$ is given by:

$$\vec{M}_o = \frac{1}{2} \rho \bar{A}_r \|\vec{v}_a\|^2 \sum_{i_a=1}^{4} \vec{r}_{a,i_a} \times \left( C_{L_a,i_a} \vec{u}_{L,i_a} + C_{D_a,i_a} \vec{u}_{D,i_a} \right),$$  

(3.75)

and the matrix $A$ is formed by re-arranging the cross products and deflection angles in equations (3.42) and (3.44) into a matrix where the results of the cross products form the columns of the matrix:

$$A = \frac{1}{2} \rho \bar{A}_r \|\vec{v}_a\|^2 \left[ \vec{a}_1 - \vec{a}_2, \vec{a}_3, \vec{a}_4 \right] \quad \text{where} \quad \vec{a}_i = \vec{r}_{a,i_a} \times \left( C_{L_a,i_a} \vec{u}_{L,i_a} + C_{D_a,i_a} \vec{u}_{D,i_a} \right).$$  

(3.76)

The resulting control surface deflections, $\delta_a$ and $\delta_r$, are computed by solving equation (3.74) for $\vec{M}_{net} = \vec{M}_{des}$ where $\vec{M}_{des}$ is given by equation (3.73). Note that the result for $\delta_e$ obtained from this method is not used. Instead, the value of $\delta_e$ obtained from the PTO controller is passed to the plant.

### 3.3.7 Power Take-off (Winch) Controller

The power take-off controller used in this work seeks to satisfy the net-zero spooling constraint by approximating the behavior over the next lap using information from the previous lap. The commanded rate of tether release, $u_T(t)$, is set by a spooling controller that seeks to spool tether out at a high angle of attack during the portions of the lap in which large tensions are possible, then spool tether in at a low angle of attack during the remainder of the lap. The intra-cycle spooling algorithm in this work is designed to maintain a consistent tether length each lap, represented by the constraint:

$$\int_{t_0,j}^{t_{f,j}} \vec{u}_j(\tau) d\tau = 0,$$  

(3.77)

where the index $j$ refers to the lap or iteration number. This ensures that the kite remains in a consistent depth within the ocean velocity profile. In attempting to find the command sequence that satisfies this constraint, several key simplifying assumptions are made:

- The winch is capable of achieving the commanded speed.

- The winch is capable of achieving that speed quickly, relative to the rate of change of the command.

- The commanded spooling speed is piecewise constant over each of $N_R$ “spooling regions”, and alternates between spooling in and spooling out at the maximum speed, $u_{spool}$.

The first two approximations should hold for a well designed winch/generator system, meaning that $\vec{u}_T(t) \approx u_T(t)$. The three approximations together mean that the constraint equation of (3.77) can be written as:

$$0 = 1^{1 \times N_R} J_{-1}^{\Delta f} \Delta f, $$  

(3.78)
where the matrix $U^{j-1}_{p,q} \in \mathbb{R}^{N_R \times N_R}$ is a diagonal matrix where the element in the $p^{th}$ and $q^{th}$ column is given by:

$$U^{j-1}_{p,q} = \begin{cases} u^{j-1}_{spool} & p = q = \text{odd} \\ -u^{j-1}_{spool} & p = q = \text{even} \\ 0 & p \neq q. \end{cases}$$  \hspace{1cm} (3.79)$$

As derived in [28], $u^{j-1}_{spool}$ is taken to be one third of the mean flow speed at the vehicle center of mass (point $B$) over the last lap of the system. The vector $\Delta^{j}_T \in \mathbb{R}^{N_R}$ is a vector containing the time durations required to traverse each section of the path during next ($j^{th}$) lap. Because the timings of the next lap are not known beforehand, it is desirable to define our tether spooling controller in terms of the path variable, $s$, not time. Therefore, I transform the time-domain constraint of equation (3.78) to a path-domain constraint by using a numerical approximation of the time derivative of the path variable from the previous lap in each spooling region. Here, I denote the spooling region with the index $i_R = 1, 2, \ldots, N_R$. Specifically, I approximate the $i^{th}$ element of $\Delta^{j}_T$, written as $\Delta^{j}_T, i_{R}$ in terms of the path variable using logged data from the previous lap, $j - 1$. Specifically,

$$\Delta^{j}_T, i_{R} \approx \frac{s^{j-1}_{i_R+1} - s^{j-1}_{i_R}}{\delta s^{j-1}_{i_R}}.$$  \hspace{1cm} (3.80)$$

Note that $s^{j-1}_{i_R}$ refers to the value of the path variable at the end of the $i^{th}$ region during the previous lap, $j - 1$. Additionally, $\delta s^{j-1}_{i_R}$ is the mean of the time derivative of $s(t)$ over the $i^{th}$ section of the path. Therefore,

$$\Delta^{j}_T = \begin{bmatrix} \frac{1}{\delta s^{j}_{1}} & 0 & \ldots & 0 \\ 0 & \frac{1}{\delta s^{j}_{2}} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{1}{\delta s^{j}_{N_R+1}} \end{bmatrix} D \begin{bmatrix} s^{j-1}_{1} \\ s^{j-1}_{2} \\ \vdots \\ s^{j-1}_{N_R+1} \end{bmatrix},$$  \hspace{1cm} (3.81)$$

$$\Delta^{j}_T = \delta s^{j-1} DS^{j-1}.$$  \hspace{1cm} (3.82)$$

Furthermore, because the path is defined using a path variable $s \in \{0, 1\}$, $s^{j}_{N_R+1} = 1 \forall j$. The discrete difference operation matrix $D \in \mathbb{R}^{N_R \times N_R}$ is a matrix with ones along the main diagonal and negative ones on the diagonal underneath the main diagonal. Thus, after every lap, the problem of satisfying our approximation of the net-zero spooling constraint becomes one of solving an approximated version of the constraint equation,

$$0 = 1^{1 \times N_R} U^{j-1}_{p,q} \delta s^{j-1} D S^{j},$$  \hspace{1cm} (3.83)$$

for the vector $S^{j} \in \mathbb{R}^{N_R+1}$, the elements of which define the spooling regions for the next lap. Note that in general, this is a single scalar equation and cannot be solved uniquely for the elements of
$S^j$. However, if one prescribes a structure to the spooling regions, then the number of parameters defining the spooling regions can be reduced to one, resulting in a unique solution. In the case of the figure-eight path geometry, it is known that the tension profile over the course of a figure-eight exhibits two local minima, which occur roughly at $s = 0.25$ and $s = 0.75$. Therefore, in this work the vector $S^j$ is parameterized in terms of a single scalar variable, $\sigma^j_w$, which defines the width of the spooling region. Therefore, $S^j$ takes the form

\[
S^j = \begin{bmatrix}
0.25 \\
0.25 \\
0.75 \\
0.75 \\
1 \\
\end{bmatrix} + \begin{bmatrix}
-1 \\
1 \\
-1 \\
1 \\
0 \\
\end{bmatrix} \sigma^j_w.
\] (3.84)

By substituting this expression into equation (3.83), one can solve directly for the width of the spool-in regions, $\sigma^j_w$. This then defines a simple, switched spooling control structure:

\[
\mathbf{u}^j_T(s(t)) = \begin{cases} 
\mathbf{u}_{in} & 0.25 - \sigma^j_w \leq s(t) \leq 0.25 + \sigma^j_w \\
\mathbf{u}_{out} & \text{otherwise.}
\end{cases}
\] (3.85)

While equation (3.85) will yield zero net spooling under nominal conditions, it is not robust to disturbances that cause the actual flight speed (and therefore the time required to traverse a particular section of the figure-eight) to differ from that which was used in computing $\mathbf{u}^j_T(s(t))$. To add robustness to the spooling strategy, I utilize a simple feedback controller to track a target tether length, $l^j_{T,SP}(s(t))$, which is obtained by integrating (3.85) over the path as follows:

\[
l^j_{T,SP}(s(t)) = l^j_{T,0} + \int_{0}^{s(t)} \frac{\mathbf{u}^j_T(\sigma)}{\delta s^j} d\sigma
\] (3.86)

where $l^j_{T,0}$ is the tuneable target tether length.
As introduced in Section 2.3, the shape of a path is defined by the form of the function $\vec{p}(\sigma, b)$, and a specific value of the basis parameters, $b \in \mathbb{R}^{n_b}$. The purpose of the path optimization ILC algorithm of this chapter is to use previous iterations’ path shapes and corresponding performances to adjust future path shapes to achieve convergence to an optimal path. The problem of following the prescribed path is then left to a lower-level controller. In this way, the problem is partitioned into an upper-level path optimization controller, which is the contribution detailed in this chapter, and a lower level path-following controller, which is detailed in Section 2.3. This structure is extremely similar to the generic hierarchical ILC structure shown in the bottom of Figure 2.1. The equivalent block diagram depicting the general idea of this strategy is shown in Figure 4.1. One can see from the figure that the ILC update to the path geometry is calculated by adjusting the set of basis parameters, $b$, that define the overall shape of the path. The path-optimization ILC algorithm is then implemented as an outer loop, upper level controller to the combined plant and lower-level controller.

As introduced in Chapter 1, the key features of this algorithm (and the algorithm detailed in Chapter 5) are that it i) achieves economic optimization, not reference tracking, ii) allows for continuous operation by lifting the requirement of an initial condition reset, and iii) allows for variable timing behavior from one iteration to the next. The first item of this list is accomplished by characterizing a metamodel of the economic performance index as a function of the path geometry (basis parameters) and then basing our ILC update on this characterization. The second and third
items on this list are enabled by the hierarchical controller structure shown in Figure 4.1. In this section, the path shape optimization is addressed by the higher-level controller, while the path following is addressed by the lower level controller. Note that in this work, the lower level controller does not consider how to optimally follow a prescribed path. However, it does enable continuous operation while allowing for variable timing behavior. The work in this chapter is included in [43], [44], and [45].

![Figure 4.1: A generic block diagram depicting the concept of operation for path optimization iterative learning control. In this configuration, the ILC update serves as an upper level controller that prescribes a flight path geometry to the lower level path following controller in the form of basis parameters, which are encoded in the vector $b$.](image)

The economic path shape optimization problem is solved in this chapter by applying the following iterative adaptation law after each iteration:

$$b_{j+1} = b_j + K_L \Delta b_j,$$  \hspace{1cm} (4.1)

In this update law, $b_j \in \mathbb{R}^{n_b}$ represents the basis parameters during iteration $j$, the matrix $K_L \in \mathbb{R}^{n_b \times n_b}$ represents the learning gain and $\Delta b_j \in \mathbb{R}^{n_b}$ is an adjustment to the basis parameters calculated to optimize the overall economic performance index, $J$.

The structure of this update parallels the structure of many ILC update laws, with $b_j$ taking the place of the control input sequence and $\Delta b_j$ taking the place of the tracking-error based update to the control signal. This chapter details two possible options for $\Delta b_j$ that I have developed. Both of these options seek to maximize or minimize an economic performance index, $J(b) \in \mathbb{R}$, by basing the update adjustment, $\Delta b$, on an estimate of the true performance index at the current iteration,
These two update options are

- the **gradient-based update**, wherein the update term, $\Delta b$ is given at every iteration $j$ by the gradient of the estimated response surface, $\nabla \hat{J}_j(b_j)$, and

- the **optimizer-based update**, wherein the update term is given at every iteration by the vector that points from the current point in basis-parameter space, $b_j$ to the optimizer of the estimated response surface,

\[
\Delta b_j = b^*_j - b_j
\]

where

\[
b^*_j = \arg \max_b \hat{f}_j(b).
\]

While the first update amounts to an iterative update law that is analogous to gradient descent in the iteration domain, the second one amounts to an update law that is similar to unconstrained successive programming techniques such as sequential quadratic programming (SQP).

Because each of these possible update laws are based on the estimation of the response surface, referred to as a metamodel, three mathematical operations must take place at each iteration:

1. The estimated response surface, $\hat{f}(b_j)$, must be updated based on performance over the previous iteration.
2. The gradient or optimizer of the estimated response surface must be calculated.
3. The path geometry for the next iteration, encoded as the next set of basis parameters, $b_{j+1}$, must be updated according to the iterative update law.

The generic block named “path optimization ILC update” in Figure 4.1 can now be drawn in more detail in terms of these operations, as shown in Figure 4.2.

**Figure 4.2** Block diagram of the path-optimization ILC update structure showing three main components, the calculation of the performance over the last iteration, the recursive least squares estimation of the response surface, and the ILC-based update law.

In practice, it is often necessary to modify both of the basic update laws described by equation (4.1) in two small ways. Both of these modifications are motivated by the realization that the update
is based on an estimate of the true performance index. Therefore, i) the estimated response may not be globally accurate, and ii) some consideration should be given to maintaining an accurate estimate of the response surface. In order to address the first potential challenge, one can modify the learning gain matrix, $K_l$, to implement a trust region, effectively restricting the size of the change (in basis parameter space) that can be implemented between iterations.

In order to address the second challenge, it is necessary in some applications to artificially induce small perturbations to the update law. This effectively injects noise into the system in order to ensure adequate exploration of the design space, such that the estimated response surface, $\hat{J}$, ultimately converges to within a finite error of the true response surface. Thus, it may be useful to utilize a slightly modified version of the update law that incorporates this persistent excitation, $p_j \in \mathbb{R}^{n_b}$ term:

$$b_{j+1} = b_j + K_L \Delta b_j + p_j.$$ \hspace{1cm} (4.4)

In order to guarantee convergence of the RLS estimator, one must choose $p_j$ such that the following *uniform persistent excitation* condition holds [46]:

**Definition 1.** *(Uniform persistent excitation)* The signal $\hat{b}$ with $b_i \in \hat{b}$, is uniformly persistently exciting if there exists an integer $T > 0$ such that:

$$\sum_{i=k}^{k+T} (h(b_i)(h(b_i)))^T > 0,$$ \hspace{1cm} (4.5)

for all $k$.

Here, the symbol $>$ indicates that the left hand side is a positive definite matrix. In the update law of (4.4), $p_j$ is the only term that can be freely specified at each iteration. Thus, $p_j$ must be carefully designed to ensure that the above persistent excitation condition holds.

Therefore, the complete statement of the update law, incorporating all of the features listed above, is given as

$$b_{j+1} = b_j + K_L(\Delta b_j)\Delta b_j + p_j.$$ \hspace{1cm} (4.6)

Here, the matrix valued function $K_L : \mathbb{R}^{n_b} \rightarrow \mathbb{R}^{n_b}$ implements a trust region. Assuming the vector $\Delta b_j$ is given by $[\Delta b_1^j \Delta b_2^j \ldots \Delta b_{n_b-1}^j \Delta b_{n_b}^j]$ and $\delta b$ is a vector of tuneable bounds on the size of the trust region, given by $[\delta b_1 \delta b_2 \ldots \delta b_{n_b-1} \delta b_{n_b}]$, then the $K_L$ is given by

$$K_L(\Delta b_j) = \min \left\{ 1, \frac{\delta b_1}{\Delta b_1^j}, \frac{\delta b_2}{\Delta b_2^j}, \ldots, \frac{\delta b_{n_b-1}}{\Delta b_{n_b-1}^j}, \frac{\delta b_{n_b}}{\Delta b_{n_b}^j} \right\} \mathbb{R}^{n_b \times n_b}.$$ \hspace{1cm} (4.7)

We now consider a quick and computationally efficient method for characterizing the aforementioned estimate of the performance index, then address convergence properties of the algorithm.
4.1 Metamodel Identification

To estimate the response surface at each iteration, \( \hat{J}_j(b) \), I model the performance index as the inner product of a regressor vector, \( h(b) \in \mathbb{R}^q \), and a coefficient vector, \( c \in \mathbb{R}^q \), as follows:

\[
\hat{J}_j(b) = h(b)_j^T c_j. \tag{4.8}
\]

Here, \( \hat{J}_j(b) \) represents an approximation of the performance index, and the regressor vector structure, \( h(b)_j \), is selected to encode the anticipated dependency of \( J \) on the basis parameters (e.g., if I expect that \( J \) is quadratic with respect to the basis parameters, then terms \( h(b)_j \) should include the squares of the basis parameters). The coefficients to the estimated response surface, \( c \) are then identified at each iteration, \( j \), using recursive least squares (RLS), with an exponential forgetting factor, \( \lambda \), as follows:

\[
V_j = \frac{1}{\lambda} \left( \frac{V_{j-1} - h(b)_j h(b)_j^T V_{j-1}}{1 + h(b)_j^T V_{j-1} h(b)_j} \right), \quad \lambda \leq 1 \\
c_j = c_{j-1} + V_j h(b)_j \left( J(b)_j - h(b)_j^T c_{j-1} \right) \tag{4.9}
\]

where \( V_j \) is the inverse of the weighted sample covariance matrix.

Given a parameter estimate, \( c_j \), the corresponding gradient of \( h(b)_j^T c_j \), denoted by \( \nabla \hat{J}(b)_j \), is computed either analytically or numerically.

Note that this method of parameterizing the response surface implicitly assumes that the initial condition, \( x(0) \), and environmental conditions, \( d_e(t) \), are constant from iteration to iteration, as mentioned earlier. This realization further motivates the introduction of the forgetting factor, \( \lambda \), which heavily weights recently acquired data in the estimate, \( c_j \), thus ensuring that the estimated response surface is weighted towards data acquired using similar flight path geometries and under similar flow conditions.

4.2 Convergence Analysis - Ideal Case

This subsection presents convergence analyses based on both forms of the basis parameter update law of equation (4.6) (gradient-based update and optimizer-based update). The analysis provides sufficient conditions under which \( b_j \) converges to a finite set containing the true optimizer, \( b^* \). This work is published in [43]. For simplicity, I consider the case where the learning gains take the form \( K_L = k_L I \).

In performing the analysis, a few assumptions are made regarding the structure of \( h(b) \) and \( p_j \):

Assumption 1: The initial condition, \( x(0) \), and external disturbance, \( d_e(t) \), are iteration-invariant, and the only uncertainties are parametric:

\[
\exists \beta^* \text{ such that } J(b, x(0), d_e(t)) = J(b) = h(b)^T \beta^*. \tag{4.10}
\]
Assumption 2: The performance index, $J(b)$, is convex and differentiable everywhere, possesses a unique maximizer, and the gradient is Lipschitz continuous with constant $L$, that is:

$$
\|\nabla J(b_{j+1}) - \nabla J(b_j)\| \leq L\|b_{j+1} - b_j\|. \quad (4.11)
$$

Assumption 3: The excitation signal at each iteration, $p_j$, is chosen such that there exists an integer $T > 0$ for which:

$$
\sum_{i=k}^{k+i} h(\xi + \sum_{k=0}^{i} (D_k + p_{j+k} + p_j)) [h(\xi + \sum_{k=0}^{i} (D_k + p_{j+k} + p_j))]^T > 0, \quad (4.12)
$$

$\forall \xi \in \mathbb{R}^n, D_k \in \mathbb{R}^{n_b} : \|D_k\| \leq \Delta_{\text{max}}, k = 0 \ldots T$.

Assumption 1 guarantees that the regressor vector structure ($h(b)$) characterizes the actual performance index. Assumption 2 bounds the iteration-to-iteration variation in the performance index. Satisfaction of Assumption 3 ensures the necessary persistent excitation, which guarantees that $\beta_j$ converges by the definition of persistent excitation.

Under the aforementioned assumptions, it is possible to guarantee convergence of $\hat{J}(b_j)$ to $J(b)$ and $b_j$ to a finite set containing $b^*$, through the following propositions, which correspond to the optimizer-based and gradient-based adaptation laws, respectively.

**Proposition 1.** (Convergence of optimizer-based iterative path adaptation law) Suppose that Assumptions 1, 2, and 3 are satisfied, and that $\|p_j\| \leq p_{\text{max}}, \forall j \geq 0$. Then under the basis parameter update law of equation (4.6) with $K_L = k_L I$, and $k_i < 1$, the following results hold:

- $\lim_{j \to \infty} \|b^*_j - b_j\| = 0$.

- The set $B = \{b : \|b^* - b\| \leq \frac{p_{\text{max}}}{k_e} \}$ is attractive, (i.e., $b$ converges to a ball of radius $\frac{p_{\text{max}}}{k_e}$ around $b^*$).

**Proof.** Defining $D_i \triangleq k_e (b^*_j - b_j)$, $b_{j+1}$ is related to $b_j$ by:

$$
b_{j+i} = b_j + \sum_{k=0}^{i} (D_{j+k} + p_{j+k}). \quad (4.13)
$$

Taking $\xi = b_j$ and $D_k = D_{j+k}$ for $k = 0 \ldots i$, it follows from Assumption 3 that there exists $T > 0$ for which equation (4.5) holds for all $j$. Thus, uniform persistent excitation is achieved, which by Assumption 3 guarantees that $\beta_j$ converges. This fact, in combination with Assumption 1 guarantees that:

$$
\lim_{j \to \infty} \|\beta_j - \beta^*\| = 0. \quad (4.14)
$$
According to Assumption 2, there exists a unique maximizer, \( b^* \), that can be computed from \( \beta^* \); therefore, it immediately follows that:

\[
\lim_{j \to \infty} \| b^* - b \| = 0. \tag{4.15}
\]

Next, note that the update law of (4.6) can be rewritten as:

\[
b_{j+1} = (1 - k_e) b_j + k_e (b^* + \epsilon_j) + p_j,
\]

where \( \epsilon_j = \hat{b}_j^* - b^* \). This can be represented in the z domain by:

\[
b(z) = \frac{k_e}{z + k_e - 1} (b^*(z) + \epsilon(z)) + \frac{1}{z + k_e - 1} p(z). \tag{4.17}
\]

Given that \( \lim_{j \to \infty} \epsilon_j = 0 \), \( b^* \) is a constant, and the poles in (4.17) are located at \( z = 1 - k_e \), (which for \( 0 < k_e < 1 \) indicates input-output stable and overdamped dynamics), the steady state-dynamics satisfy:

\[
\| b_{ss} - b^* \| \leq \frac{p_{max}}{k_e}. \tag{4.18}
\]

Thus, \( b \) converges to a set \( B = \{ b : \| b^* - b \| \leq \frac{p_{max}}{k_e} \} \) at steady state.

**Proposition 2.** (Convergence of gradient-based iterative path adaptation law) Suppose that Assumptions 1, 2, and 3 are satisfied, and that \( \| p_j \| \leq p_{max}, \forall j \geq 0 \). Then under the basis parameter update law of (4.6) with \( K_L = k_L I \), and \( 0 < k_L < \frac{2}{L} \), the following results hold:

- \( \lim_{j \to \infty} \| \hat{J}_j(b_j) - J(b) \| = 0 \).

- The set \( B = \{ b : \| \nabla J(b) \| \leq \frac{1}{L_{min}} \frac{p_{max}}{k_e - k_L} \} \) is attractive.

**Proof.** By the same logic as the proof of Proposition 1, if I define \( D_i = k_e I \hat{J}_j(b_j) \), \( b_{j+1} \) then it follows from Assumption 3 that uniform, persistent excitation is achieved, and (4.14) holds in this case as well. Furthermore, according to Assumption 1, the only uncertainties are parametric, therefore it immediately follows that:

\[
\lim_{j \to \infty} \hat{J}(b_j) - J(b) = 0 \quad \forall b. \tag{4.19}
\]

This implies from continuity that:

\[
\lim_{j \to \infty} \| \nabla \hat{J}(b_j) - \nabla J(b) \| = 0. \tag{4.20}
\]

In fact, Assumption 1 guarantees that \( \nabla \hat{J}(b_j) = \nabla J(b) \) for some finite value of \( j \). Assumption 2 provides a bound on the change in performance index between two successive iterations, \( \Delta J \equiv \)
\[ J(b_{j+1}) - J(b_j) : \]

\[ \Delta J \geq \nabla J(b_j)^T (b_{j+1} - b_j) - \frac{L}{2} \| b_{j+1} - b_j \|^2. \]  
(4.21)

Substituting the gradient-based update law for \( b_{j+1} \), with \( K_b = I \) and \( K_e = k_e I \), gives:

\[ \Delta J \geq \nabla J(b_j)^T (k_e \nabla \hat{J}(b_j) + p_j) - \frac{L}{2} \| k_e \nabla \hat{J}(b_j) + p_j \|^2. \]  
(4.22)

Noting from before that \( \nabla \hat{J}(b_j) = \nabla J(b_j) \) for a finite value of \( j \), then for sufficiently large \( j \), this can be algebraically rearranged to obtain:

\[ \Delta J \geq \frac{L}{2} \left( k_e \left( \frac{2}{L} - k_e \right) \| \nabla J(b_j) \|^2 + 2 \left( \frac{1}{L} - k_e \right) p_j^T \nabla J(b_j) - p_j^T p_j \right). \]  
(4.23)

This is lower bounded once again by using the largest magnitude of the excitation signal, \( p_{max} \):

\[ \Delta J \geq \frac{L}{2} \left( k_e \left( \frac{2}{L} - k_e \right) \| \nabla J \|^2 + 2 \left( \frac{1}{L} - k_e \right) p_{max} \| \nabla J \| - p_{max}^2 \right), \]  
(4.24)

where the dependence of \( J \) on \( b \) has been suppressed for succinctness. Sufficient conditions on \( k_e \) and \( \| \nabla J \| \) for the quadratic expression to be greater than or equal to zero are:

\[ 0 < k_e < \frac{2}{L} \quad \text{and} \quad \| \nabla J \| \geq \frac{p_{max}}{\frac{2}{L} - k_e}. \]  
(4.25, 4.26)

Defining \( B = \{ b : \| \nabla J(b) \| \leq \frac{p_{max}}{\frac{2}{L} - k_e} \} \), \( J' \) as the maximum scalar such that \( B \subset \{ b : J(b) \geq J' \} \), and \( S = \{ b : J(b) \geq J' \} \), then \( S \) is invariant and \( \Delta J > 0 \) whenever \( b \not\in S \). Thus the performance index will be non-decreasing (i.e. \( \Delta J \geq 0 \)) whenever the true response surface has been identified, the learning gain is chosen appropriately and \( b \not\in S \).

\[ 4.2.1 \quad \text{Convergence Analysis - Discussion} \]

As mentioned previously, the convergence analysis given above is based on a set of three assumptions. The most severe deviations from the ideal case occur because, in real-world systems, Assumption 1 will typically be violated to some extent.

In the tethered energy systems examined in this work, inconsistent initial conditions arise from a combination of two sources. The first is the optimization itself. By varying the shape of the path between iterations, one changes (at least) the heading angle at which the system finishes iteration \( j \). This is then the heading angle at which the system starts iteration \( j + 1 \). Second, the external disturbance (the flow speed) is varying from iteration to iteration. Because the flow speed determines the system’s achievable flight speed, it influences the state of the system at the end of iteration \( j \),
which is then the initial state at iteration \( j + 1 \). These iteration-to-iteration variations must be kept manageable in size through careful tuning of the learning gains.

Furthermore, in this application there is no reason to expect that the true, global response surface satisfies Assumption 1. Rather, the estimated response surface should be viewed as a local approximation.

In order to increase the robustness of the optimization against the effects of violating these assumptions, three methods for adjusting the optimization algorithm can be pursued:

- First, a trust region to limit the amount that the basis parameters are allowed to vary between iterations can be implemented.
- Second, the learning gain, \( K_L \), can be reduced.
- Third, the forgetting factor, \( \lambda \), can be set to a value close to but less than 1. For the following results, it was found that \( \lambda \) in the range of 0.95 to 0.99 was appropriate.

The first and second adjustments are helpful in mitigating the effects of both the inconsistent initial conditions and the local nature of the response surface estimation. The last adjustment is helpful in mitigating the effects of the inconsistent external disturbance. The idea here is that by heavily weighting the more recent data in the response surface estimation, the algorithm will fit the response surface to data measured under a similar external disturbance.

### 4.3 Applications

This section details the application of the path-optimization ILC method of the previous sections to each of the three dynamic models detailed in Section 2.1. It also presents the results from simulations on each of these models.

#### 4.3.1 Sailboat Model

This section presents a path optimization ILC algorithm that is tailored to the sailboat model. This work, and the resulting analysis, is published in [43].

With a tethered energy system, the ultimate objective is to maximize power output in a manner that can be repeated from one iteration to the next. This means that the performance index, \( J \), should be chosen primarily to reflect average power output but should also disallow figure-eight paths that cannot be followed adequately (which will lead to situations where the figure-eight paths cannot be repeated because they do not finish at the same initial condition that they start at). To capture these desires, the following two-term performance index is chosen and calculated for each iteration:

\[
J = \frac{k}{T_f} \int_0^{T_f} v_{app}^3(t) dt - \frac{1}{n_w d_t} \sum_{m=1}^{n_w} \min_t \left\{ ||\vec{r}(t) - \vec{p}_i|| \right\}
\]  

(4.27)
where $k$ is a normalization constant, $T_f$ is the total duration of a single lap, $\dot{r}(t) = [x_\phi(t) \ y_\phi(t)]^T$, $n_w$ is the number of waypoints, $d_i$ is a normalization length, and $\vec{p}_i$ is the spatial position of waypoint $i$. Here, the first term is proportional to the average power generated over an iteration. The second term acts as a soft constraint that ensures that paths can be tracked within sufficiently tight tolerance. Specifically, the term $\min_t \{||\dot{r}(t) - \vec{p}_i||\}$ characterizes the minimum error between waypoint $i$ and the path that the tethered energy system actually follows.

For this application, the path shape function, $\vec{p}(\sigma, \mathbf{b}_j)$, was chosen as the Lemniscate of Gerono:

$$
\vec{p}(\sigma, \mathbf{b}_j) = \begin{bmatrix} W_j \cos\left(\left(2\sigma + \frac{3}{2}\right)\pi\right) \\ H_j \sin\left(\left(2\sigma + \frac{3}{2}\right)\pi\right) \end{bmatrix},
$$

(4.28)

where the basis parameters $W_j$ and $H_j$ describe the width and height of the course.

In order for this ILC-based waypoint optimization to be successful, it should produce paths that converge to a fixed geometry. Furthermore, the performance index, $J_j$, should increase as iteration number increases, as should the average power output. Figure 4.3 shows the course geometry over one optimization at several iterations. This figure demonstrates that the figure-eight path converges as iterations increase. In this case, the figure-eight path takes on a larger width and smaller height as iterations progress.

To assess the robustness of our adaptation approach, I have also explored the sensitivity of the final, converged basis parameters to initial conditions. Figure 4.4 shows the evolution of the basis parameters from iteration to iteration, for three candidate initial conditions. Figure 4.5 shows the corresponding performance index at each iteration. These figures suggest that both the basis parameters and performance do indeed converge to a relatively small domain around their optimal
values. For these simulation-based results, the optimal basis parameters were calculated through an exhaustive grid search, which enabled a direct comparison with the results from the ILC-based basis parameter adaptation. It is important to note that while an exhaustive search can be done to ensure convergence of the ILC-based basis parameter adaptation in simulation, the real merit of the ILC-based basis parameter adaptation arises from the fact that simulation models are in fact imperfect. Online adjustment of the figure-eight path is essential in those cases.

![Sailboat Model Iterative Convergence of Basis Parameters](image)

**Figure 4.4** Evolution of basis parameters for a variety of initial conditions on the basis parameters of the sailboat model when subjected to the path-optimization ILC algorithm.

### 4.3.2 Unifoil Model

In the following simulation results, I model a ground-gen marine hydokinetic energy system. The analysis and results of this section are included in [44]. The dynamic model and iterative optimization were tested under a spatially and temporally constant flow profile. In these simulations, the course geometry was held constant over three full iterations, in order to allow initial transients to dissipate. The RLS estimator used to estimate the response surface was then initialized by testing five points in the design space and solving for the best fit surface. The following results represent the evolution of the system under the ILC update law but do not show the initial transient settling and initialization phases.

As with the sailboat model, the performance index is chosen to capture two simultaneous goals. The first term of this performance index characterizes the mean power generation over a single iteration, and the second term serves as a penalty on paths that cannot be followed accurately, helping to ensure robust and repeatable flight. Specifically, the performance index at each iteration
Figure 4.5 Evolution of performance index given in equation (4.27) for a variety of initial conditions of the sailboat model when subjected to the path-optimization ILC algorithm.

is calculated as

$$J_j = \frac{1}{t_{e,j} - t_{s,j}} \int_{t_{s,j}}^{t_{e,j}} [T(t)u_r(t) - kd_{\text{min},j}(t)] dt,$$

(4.29)

where $t_{s,j}$ and $t_{e,j}$ are the start and end times of the $j$-th iteration, $T(t)$ is the estimated tether tension, $k$ is a scalar weight, and $d_{\text{min},j}$ is the minimum distance from the system to the path at each instant over iteration $j$.

In three dimensions, the path shape function, $\vec{p}(\sigma, b_j)$ is given in spherical coordinates by:

$$r_j(\sigma, b_j) = r_j(t) \quad \forall j$$

(4.30)

$$\Theta_j(\sigma, b_j) = W_j \cos\left(2\sigma + \frac{3}{2}\pi\right)$$

(4.31)

$$\Phi_j(\sigma, b_j) = -H_j \sin\left(2\sigma + \frac{3}{2}\pi\right) + \Phi_0$$

(4.32)

where $r$ is the distance from point $G$ to point $B$, $W_j$ and $H_j$ are the basis parameters width and height as before, and $\Phi_0$ is the mean course elevation angle.

It was observed during simulation that this system was much more sensitive to one basis parameter than the other. Accordingly, the ILC update used in simulation was performed on a set of normalized basis parameters, $\vec{b}_j = [W_i, H_i]_I$ where the normalization constants, $w_n$ and $h_n$ were chosen to reflect the relative sensitivity of the performance index, $J$ to each basis parameter.

Although in the previous results of Section 4.3.1 it was not necessary to utilize the small persistent excitation to the update law, I found that for this model, this was necessary. This perturbation, $p_j$, was implemented as two, independent unit white noise generators, indicated by the functions $W_1(j)$
and $W_j$, according to:

$$p_i = \left[ 2\bar{w} \left( W_1(i) - \frac{1}{2} \right) - 2\bar{h} \left( W_2(i) - \frac{1}{2} \right) \right]^T \tag{4.33}$$

where $\bar{w}$ and $\bar{h}$ are normalized excitation amplitudes. Specific values of the parameters used in simulation are given in Table 4.1.

Table 4.1 Parameters used in simulation of the unifoil model when subjected to the path optimization ILC algorithm.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_r$</td>
<td>Winch model time constant</td>
<td>1</td>
<td>s</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Forgetting factor</td>
<td>0.99</td>
<td>-</td>
</tr>
<tr>
<td>$\Phi_0$</td>
<td>Mean path elevation angle</td>
<td>30</td>
<td>deg</td>
</tr>
<tr>
<td>$r_{min}$</td>
<td>Tether speed transition point, lower</td>
<td>40</td>
<td>m</td>
</tr>
<tr>
<td>$r_{max}$</td>
<td>Tether speed transition point, upper</td>
<td>100</td>
<td>m</td>
</tr>
<tr>
<td>$u_{in}$</td>
<td>Tether spool-in speed</td>
<td>3</td>
<td>m s$^{-2}$</td>
</tr>
<tr>
<td>$\alpha_{out}$</td>
<td>Wing angle of attack during spool-out</td>
<td>5.7</td>
<td>deg</td>
</tr>
<tr>
<td>$k$</td>
<td>Performance index weight</td>
<td>10</td>
<td>kW m$^{-1}$</td>
</tr>
<tr>
<td>$\omega_n$</td>
<td>Width sensitivity normalization constant</td>
<td>5</td>
<td>deg</td>
</tr>
<tr>
<td>$\delta_n$</td>
<td>Height sensitivity normalization constant</td>
<td>0.5</td>
<td>deg</td>
</tr>
<tr>
<td>$\bar{w}$</td>
<td>Normalized width excitation amplitude</td>
<td>0</td>
<td>deg</td>
</tr>
<tr>
<td>$\bar{h}$</td>
<td>Normalized height excitation amplitude</td>
<td>0</td>
<td>deg</td>
</tr>
<tr>
<td>$W_1$</td>
<td>Initial course height</td>
<td>90</td>
<td>deg</td>
</tr>
<tr>
<td>$H_1$</td>
<td>Initial course width</td>
<td>20</td>
<td>deg</td>
</tr>
</tbody>
</table>

The full three-dimensional shape of the achieved path over the full spool-in spool-out cycle both before and after the optimization is shown in Figure 4.6. This figure shows that the optimization decreased the path width, $W_j$, significantly. Although the change in the zenith basis parameter, $H_j$, is not obvious from Figure 4.6, Figure 4.7 shows that it was also reduced, although to a lesser degree.

The primary components contributing to energy generation are shown in Figure 4.8. These include the rate of change of tether length, $u_r$ or $\dot{r}$, tether tension, and the estimated power output or consumption. During spool-in the tether tension is small, and during spool-out, the tether tension is much higher, producing a large positive power output. Furthermore, it is apparent that the overall duration of the iteration was reduced between the first and last iteration. The data shown in Figure 4.9 suggests that the optimization converges to an optimum. Finally, Figure 4.10 shows that this optimization strategy increases the mean power output of the system by approximately 35% compared to the initial course geometry.
Figure 4.6 Three-dimensional path of the system over a single iteration both before and after optimization for the kite-based, ground-gen marine hydrokinetic (MHK) unifoil model as subjected to the path-optimization ILC.

Figure 4.7 Convergence of course geometry for the kite-based, ground-gen marine hydrokinetic (MHK) unifoil model as subjected to the path-optimization ILC.
Figure 4.8 Tether speed, $\dot{r}$, tether tension, $T$, and estimated power production over the first iteration, as well as the last iteration for the kite-based, ground-gen marine hydrokinetic (MHK) energy system as subjected to the path-optimization ILC.

Figure 4.9 Convergence of the performance index, given in (4.29), over the course of optimization for the kite-based, ground-gen marine hydrokinetic (MHK) energy system as subjected to the path-optimization ILC.
4.3.3 High Fidelity Model

In applying the path-optimization ILC algorithm to high fidelity model, I again chose to model a marine hydrokinetic energy system. The analysis and results of this section are included in [47]. To investigate the efficacy of our control and optimization structure two test cases were simulated:

- a spatially and temporally constant flow environment where the flow velocity vector is independent of spatial location and time, and
- a spatially and temporally varying flow environment where the flow velocity vectors were calculated using an ocean flow model that combines both high and low frequency flow components as detailed in Section 3.3.5.

To properly characterize the performance of this system, I again use a two-term performance index where the first term captures the energy generation, and the second term captures the path tracking performance. Specifically, the performance index at each iteration is calculated as

\[ J_j = \frac{1}{t_{e,j} - t_{s,j}} \int_{t_{s,j}}^{t_{e,j}} \left[ P(\tau) - k_w d(\tau) \right] d\tau, \]  

(4.34)

where \( t_{s,j} \) and \( t_{e,j} \) are the start and end times of the \( j \)-th iteration, \( P(t) \) is the estimated power production, as calculated from equation (3.56). In this expression, \( k_w \) is a scalar weight, and \( d(t) \) is the penalty that describes our secondary design objectives. In this work, we use \( d(t) = \alpha(t) \) where \( \alpha(t) \) is the interior angle between the current position of the system and the closest point on the path.

In both the constant and varying flow simulation results, the system was deployed from a fixed floating platform located 200 m above the sea floor. The path under consideration had a mean...
azimuth angle of $0^\circ$ and a mean elevation angle of $-20^\circ$. Note that the negative elevation angle corresponds to deployment from a floating platform, as depicted on the left of Figure 2.6. A summary of plant parameters used in simulation is given in Table 4.2 and a summary of controller parameters used in simulation is given in Table 4.3.

Table 4.2 Important plant model parameters used in the high fidelity model when subjected to the path optimization ILC algorithm.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kite</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M$</td>
<td>Mass</td>
<td>945</td>
<td>kg</td>
</tr>
<tr>
<td>—</td>
<td>Density</td>
<td>1000</td>
<td>kg/m³</td>
</tr>
<tr>
<td>$I_{xx}$</td>
<td>Inertia tensor xx element</td>
<td>6303</td>
<td>kg/m²</td>
</tr>
<tr>
<td>$I_{yy}$</td>
<td>Inertia tensor yy element</td>
<td>2080</td>
<td>kg/m²</td>
</tr>
<tr>
<td>$I_{zz}$</td>
<td>Inertia tensor zz element</td>
<td>8320</td>
<td>kg/m²</td>
</tr>
<tr>
<td>—</td>
<td>Fuselage length</td>
<td>8</td>
<td>m</td>
</tr>
<tr>
<td>$A_r$</td>
<td>Reference area</td>
<td>10</td>
<td>m²</td>
</tr>
<tr>
<td>—</td>
<td>Port wing chord</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>—</td>
<td>Port wing span</td>
<td>5</td>
<td>m</td>
</tr>
<tr>
<td>—</td>
<td>Starboard wing chord</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>—</td>
<td>Starboard wing span</td>
<td>5</td>
<td>m</td>
</tr>
<tr>
<td>—</td>
<td>Horizontal stabilizer chord</td>
<td>0.5</td>
<td>m</td>
</tr>
<tr>
<td>—</td>
<td>Horizontal stabilizer span</td>
<td>4.0</td>
<td>m</td>
</tr>
<tr>
<td>—</td>
<td>Vertical stabilizer chord</td>
<td>0.6</td>
<td>m</td>
</tr>
<tr>
<td>—</td>
<td>Vertical stabilizer span</td>
<td>2.0</td>
<td>m</td>
</tr>
<tr>
<td><strong>Tether</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_n$</td>
<td>Number of nodes</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$r_T$</td>
<td>Radius</td>
<td>7.2</td>
<td>mm</td>
</tr>
<tr>
<td>$E_y$</td>
<td>Youngs modulus</td>
<td>50</td>
<td>GPa</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>Damping ratio</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>$m$</td>
<td>Damping mass</td>
<td>945</td>
<td>kg</td>
</tr>
<tr>
<td>$C_{D,T}$</td>
<td>Drag coefficient</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>$\rho_T$</td>
<td>Density</td>
<td>1300</td>
<td>kg/m³</td>
</tr>
<tr>
<td><strong>Winch</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_w$</td>
<td>Time constant</td>
<td>1</td>
<td>s</td>
</tr>
<tr>
<td>$\eta_{gen}$</td>
<td>Generator efficiency</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>$\eta_m$</td>
<td>Motor efficiency</td>
<td>0.95</td>
<td></td>
</tr>
</tbody>
</table>
Table 4.3 Important controller parameters used in the high-fidelity model when subjected to the path optimization ILC algorithm.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Path-Optimization Controller</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k_1^L$</td>
<td>Learning gain for 1 m/s flow</td>
<td>$21 \times 10^{-6}$</td>
<td>$rad^2/W$</td>
</tr>
<tr>
<td>$k_1^p$</td>
<td>Penalty weight for 1 m/s flow</td>
<td>43</td>
<td>$kW/rad$</td>
</tr>
<tr>
<td>$k_2^L$</td>
<td>Learning gain for 2 m/s flow</td>
<td>$3.3 \times 10^{-6}$</td>
<td>$rad^2/W$</td>
</tr>
<tr>
<td>$k_2^p$</td>
<td>Penalty weight for 2 m/s flow</td>
<td>274</td>
<td>$kW/rad$</td>
</tr>
<tr>
<td>$k_v^L$</td>
<td>Learning gain variable flow</td>
<td>$7.8 \times 10^{-6}$</td>
<td>$rad^2/W$</td>
</tr>
<tr>
<td>$k_v^p$</td>
<td>Penalty weight variable flow</td>
<td>116</td>
<td>$kW/rad$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Forgetting factor</td>
<td>0.95</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Flight Controller</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha_0$</td>
<td>Angle weighting limit</td>
<td>6</td>
<td>$deg$</td>
</tr>
<tr>
<td>$k_\gamma$</td>
<td>Tangent roll proportional gain</td>
<td>0.2</td>
<td>–</td>
</tr>
<tr>
<td>$\xi_{min}$</td>
<td>Min prescribed tangent roll</td>
<td>$-20$</td>
<td>$deg$</td>
</tr>
<tr>
<td>$\xi_{max}$</td>
<td>Min prescribed tangent roll</td>
<td>20</td>
<td>$deg$</td>
</tr>
<tr>
<td>$k_{pL}$</td>
<td>Roll moment proportional gain</td>
<td>$6.3 \times 10^5$</td>
<td>$Nm/rad$</td>
</tr>
<tr>
<td>$k_{iL}$</td>
<td>Roll moment integral gain</td>
<td>0</td>
<td>$Nm/rad/s$</td>
</tr>
<tr>
<td>$k_{dL}$</td>
<td>Roll moment derivative gain</td>
<td>$6.3 \times 10^5$</td>
<td>$Nms/rad$</td>
</tr>
<tr>
<td>$\tau_L$</td>
<td>Roll moment filter time constant</td>
<td>0.001</td>
<td>$s$</td>
</tr>
<tr>
<td>$k_{pN}$</td>
<td>Yaw moment proportional gain</td>
<td>5730</td>
<td>$Nm/rad$</td>
</tr>
<tr>
<td>$k_{iN}$</td>
<td>Yaw moment integral gain</td>
<td>0</td>
<td>$Nm/rad/s$</td>
</tr>
<tr>
<td>$k_{dN}$</td>
<td>Yaw moment derivative gain</td>
<td>0</td>
<td>$Nms/rad$</td>
</tr>
<tr>
<td>$\tau_N$</td>
<td>Yaw moment filter time constant</td>
<td>1</td>
<td>$s$</td>
</tr>
<tr>
<td></td>
<td>PTO Controller</td>
<td></td>
<td></td>
</tr>
<tr>
<td>–</td>
<td>Spool in elevator deflection</td>
<td>23</td>
<td>$deg$</td>
</tr>
<tr>
<td>–</td>
<td>Spool out elevator deflection</td>
<td>0</td>
<td>$deg$</td>
</tr>
</tbody>
</table>

4.3.3.1 Spatiotemporally Constant Flow Profile

The results in this section were obtained for two spatiotemporally constant flow profiles, one in which $\vec{v}_{f,i}(t, \vec{r}_i) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \forall \vec{r}_i, t$, and one in which $\vec{v}_{f,i}(t, \vec{r}_i) = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^T \forall \vec{r}_i, t$. Both flow speeds were compared to baseline simulations wherein the course geometry did not change from the initial conditions. Note that changes in performance for the baseline simulation are due to transient behavior in the iteration domain of the spooling controller. Similarly, the momentary peak in mean power in Figure 4.11d is also due to iteration domain transients in the PTO controller.

Results are shown in Figure 4.11. Figures 4.11a and 4.11e show how the basis parameters change over the course of the 100 minute simulation, and figures 4.11c and 4.11g show the shape of the path at the beginning and end of the optimization for both simulations. It is interesting to note that under both flow speeds, $b_1$ converged to nearly the same value, while $b_2$ converged to different values. This suggests that the optimal value of $b_1$ may be relatively insensitive to flow speed, whereas
the optimal value of $b_2$ may vary with the flow speed. Also, because the optimized path shape is significantly larger than the initial path, the optimized path resulted in fewer iterations than the unoptimized path, as noted in the legend of figures 4.11b, 4.11d, 4.11f, and 4.11h.

Furthermore, the final converged values of power production shown in Figures 4.11d and 4.11h appear to nearly follow the idealized relationship between flow speed and power production. Finally, note that the optimized path geometry produced approximately 100% to 200% more power than the unoptimized path geometry.

### 4.3.3.2 Spatiotemporally Varying Flow Profile

Simulations in the realistic flow scenarios were compared against a baseline wherein the course geometry was fixed at the initial course geometry over the entire duration of the simulation.

Figure 4.12a shows the evolution of the basis parameters over the course of 100 minutes of simulation, and Figure 4.12c shows the shape of the target path for both the first and last iteration. One can see from the plots that the parameters change significantly from the initial conditions, resulting in a significantly larger course geometry. Similar to the constant flow results in the previous section, the optimized path geometry was larger than the initial path geometry, resulting in fewer iterations over the whole simulation.

Figures 4.12b and 4.12d show how the performance index and mean power vary over the course of the 100 minute simulation time. One can see from Figure 4.12b that the ILC-based path optimization significantly increases the performance index. In fact, Figure 4.12d shows that it produces just under 25 kW on average whereas the unoptimized course produces approximately 12.5 kW on average, an increase of almost 100%. Additionally, this plot demonstrates that the algorithm is not sensitive to high-frequency, spatiotemporal variations in flow speed.
Figure 4.11 Summary results obtained by applying the path-optimization ILC algorithm to the high-fidelity model under a spatially and temporally constant flow profile at both 1 m/s and 2 m/s.
Figure 4.12 Summary results obtained by applying the path-optimization ILC algorithm to the high-fidelity model under a flow profile containing both a low frequency component and high-frequency, turbulent component.
In this chapter, I formulate an iterative learning control structure with several key features based on the feedback+feedforward ILC control structure shown in Figure 5.1.

This section first describes a framework that implicitly allows the time-domain behavior (waypoint arrival times and total iteration duration) to vary without resorting to computationally intensive nonlinear optimization problems. In doing so, I reparameterize the system model from one which evolves with respect to time, to one that evolves with respect to path position, \( s \). This
is significant because in this new domain, waypoint arrival behavior and trial duration are fixed between iterations.

This section then describes a framework that does not require the initial conditions to be reset between iterations. For many systems, the requirement that initial conditions be reset may be infeasible or suboptimal. Therefore, lifting this requirement is critical when optimizing some systems such as tethered energy systems. In this work, I eliminate the need to reset the initial conditions by combining i) a lifted system model that explicitly includes the impact of nonzero initial conditions with ii) learning filters based on a performance index that includes predicted performance over multiple future iterations. The resulting iterative update law then provides control sequences for multiple future iterations. However, due to the need to adjust for disturbances, modeling uncertainties, and other inaccuracies, only one iteration is implemented before updating the control sequence again. The overall control structure is then extremely similar to time-domain model predictive control with the notable exception that our prediction and control horizons are expressed in terms of entire iterations instead of time steps.

The third feature that I incorporate is the ability to handle economic optimization. Previous work in iterative learning control has concentrated heavily on minimizing tracking error. While this is a appropriate objective for a large number of applications, it may be inappropriate for others. Thus, work presented in this section has resulted in a framework capable of optimizing economic objectives such as power production. This is accomplished in this work in one of two ways:

- Receding horizon norm-optimal economic iterative learning control, referred to in this work simply as receding horizon ILC, where the ILC update is calculated based on optimal learning filters that are calculated to minimize or maximize a quadratic performance index.

- Linear programming based economic iterative learning control, referred to as linear programming based ILC, where the ILC update is formulated as the solution to a linear programming problem that is based on a linearized approximation of the performance index.

Here, the first method follows traditional methods developed in norm-optimal ILC literature for deriving learning filters as detailed in Section 2.2.1. This iterative update structure then effectively amounts to unconstrained sequential quadratic programming. In contrast, the second constructs an iterative update law where the deviation in the control sequence between iterations is calculated as the solution to an optimization (linear programming) problem. This ILC update structure then amounts to successive linear programming (SLP). Therefore, a key difference between these methods is then the ability to explicitly incorporate constraints. While the norm-optimal ILC method requires that constraints be implemented as soft constraints, that is penalties on constraint violation in the performance index, the linear-programming based update allows the designer to explicitly define constraint functions (which then must be linearly approximated in the linear program).

In the following subsections, I first demonstrate how to obtain a lifted model in the path domain, and then demonstrate how that lifted model can be used in both the receding horizon ILC framework and the linear programming based ILC framework.
5.1 Path Domain Lifted Model

Obtaining a path-domain lifted model requires that the dynamic model and the state and control trajectories from the previous iteration be reparameterized from depending on time, \( t \), to depending on path position, \( s \). Recall that Section 2.3 explicitly defines the path position, \( s \), as the value of the path variable, \( \sigma \), that corresponds to the closest point along the path, which is defined by the path shape function, \( \vec{p}(\sigma, \mathbf{b}) \), and a distance metric, \( d(\vec{r}, \vec{p}) \). Note that throughout this section, the basis parameters are assumed to be constant, and thus \( \vec{p}(\sigma, \mathbf{b}) \) is often simply written as \( \vec{p}(\sigma) \). The work included in this section is included in [48].

To begin, suppose that we are given a dynamic model of the form \( \frac{dx}{dt} = f(x, u) \) with state vector \( x \in \mathbb{R}^n_x \) and control input vector \( u \in \mathbb{R}^n_u \). Then the time-domain dynamic model can be converted to a path-domain dynamic model by applying a simple calculus chain rule. This results in the relationship:

\[
\frac{dx}{ds} = f(x, u) \left( \frac{ds}{dt} \right)^{-1}, \tag{5.1}
\]

where \( \left( \frac{ds}{dt} \right)^{-1} \) is inverse of the rate of change of the path position. An analytical expression for this term is provided by a corollary of the implicit function theorem and given by:

\[
\frac{ds}{dt} = -\frac{\partial_{\sigma} \partial_t d(\vec{r}(t), \vec{p}(\sigma))}{\partial_{\sigma}^2 d(\vec{r}(t), \vec{p}(\sigma))}, \tag{5.2}
\]

where \( \partial_{\sigma, t} \) denotes the partial derivative with respect to \( \sigma \) or \( t \), and \( \partial^2 \) is the corresponding second partial derivative. One can then define the vector-valued function \( g(x, u) : (\mathbb{R}^n_x, \mathbb{R}^n_u) \rightarrow \mathbb{R}^n_x \), as

\[
g(x, u) \triangleq -f(x, u) \left( \frac{\partial_{\sigma} \partial_t d(\vec{r}(t), \vec{p}(\sigma))}{\partial_{\sigma}^2 d(\vec{r}(t), \vec{p}(\sigma))} \right)^{-1}, \tag{5.3}
\]

then our path-domain dynamic model of equation (5.1) can be written concisely as \( x' = g(x, u) \) where \( ' \) denotes the derivative with respect to path position, \( s \). Note that \( g(x, u) \) implicitly depends on both the dynamic model, \( f(x, u) \), the choice of path shape, \( \vec{p}(\sigma) \), and the chosen distance metric, \( d \).

Given that \( g(x, u) \) will almost certainly be nonlinear for all but the most trivial examples, I first consider how to linearize and discretize this dynamic model before performing the lifting operation.

Beginning with an arbitrary sequence of state vectors and control input vectors parameterized with respect to the path position, \( x_l(s) \) and \( u_l(s) \), an approximation of our path-parameterized dynamic model is then calculated by linearizing \( g(x, u) \) about the sequences \( x_l(s) \) and \( u_l(s) \). The resulting linearized model is given by:

\[
x'(s) = x'_l(s) + A(s)(x(s) - x_l(s)) + B(s)(u(s) - u_l(s)), \tag{5.4}
\]
and the path-varying matrices $A(s)$ and $B(s)$ are defined as:

$$A(s) \triangleq \left( \nabla_x g(x, u) \right)_{x=x_l(s), \ u=u_l(s)} , \quad B(s) \triangleq \left( \nabla_u g(x, u) \right)_{x=x_l(s), \ u=u_l(s)}. \quad (5.5)$$

To obtain a lifted system representation, it is necessary to discretize the model of equations (5.4) and (5.4). Using a zero-order hold, the resulting sequences of $n_s$ discrete path variable matrices, $\{A_1, A_2, ..., A_{n_s}\}$ and $\{B_1, B_2, ..., B_{n_s}\}$, where $n_s$ is the number of discrete steps along the path, are then given by:

$$A_q = e^{A(s_q)\Delta_s}, \quad (5.6)$$
$$B_q = \int_0^{\Delta_s} e^{A(s_q)\tau_s} B(s_q) \, d\tau_s, \quad (5.7)$$

where $q \in \{1, 2, ..., n_s\}$ refers to a discrete index along the path, $\Delta_s$ is the path discretization increment and $s_q$ represents the path position at path step $q$, i.e. $s_q = q\Delta_s$. The result is a discrete path-parameterized dynamic model given by:

$$x_{q+1} = x_q + A_q (x_{q+1} - x_q) + B_q (u_{q+1} - u_q), \quad (5.8)$$

Note that this relationship is derived assuming linearization around a generic set of two sequences, $x_l(s)$ and $u_l(s)$. In subsequent sections, I will always choose to linearize our system around the sequences achieved during the previous iteration, $x^j(s)$ and $u^j(s)$. Thus, the discrete path dynamics of (5.8) become

$$x_{q+1}^j = x_q^j + A_q^j (x_{q+1}^j - x_q^j) + B_q^j (u_{q+1}^j - u_q^j), \quad (5.9)$$

where the superscript, $j$ refers to the iteration that was just completed, and $j+1$ refers to the next iteration, which has not yet begun. Thus, this model predicts the state sequence for the next iteration as a function of the state sequence over the last iteration.

Given these discrete-path dynamics, and these definitions of the lifted vectors from the previous iteration, $j$,

$$x_j \triangleq \left[ (x_0^j)^T \ x_1^j \ \cdots \ x_{n_s-1}^j \ x_{n_s}^j \right]^T, \quad u_j \triangleq \left[ (u_0^j)^T \ u_1^j \ \cdots \ u_{n_s-1}^j \ u_{n_s}^j \right]^T, \quad (5.10)$$

it is now straightforward to derive a lifted system representation. This lifted system model expresses the state sequence at the next iteration, $x_{j+1}$, (defined similarly to equation (5.10)) as a function of

- the state sequence from the previous iteration, $x_j$,
- the control sequence from the previous iteration, $u_j$,
- the control sequence for the next iteration, $u_{j+1}$ (defined similarly to equation (5.10)), and
• the initial condition of the previous iteration $x_0^j$ and the initial condition of the iteration, $x_0^{j+1}$.

The lifted system model is then written concisely as:

$$\mathbf{x}_{j+1} = \mathbf{x}_j + G_j (u_{j+1} - u_j) + F_j (E_F - E_I) \mathbf{x}_j,$$

(5.11)

where the $n_x \times n_u$ block element in the $m^{th}$ block row and $p^{th}$ block column of $G_j$, $[G_j]_{m,p}$, is given by:

$$[G_j]_{m,p} = \begin{cases} 
0_{n_x \times n_u} & m < p \\
B_m^j & m = p \\
A_m^j A_{m-1}^j \ldots A_p^j B_m^j & \text{otherwise}.
\end{cases}$$

(5.12)

Similarly, the block element in the $m^{th}$ block row of $F_j$, $[F_j]_{m,}$, is given by

$$F_m^j = \begin{cases} 
I_{n_x \times n_x} & m = 1 \\
A_m^j A_{m-1}^j \ldots A_2^j A_1^j & \text{otherwise}.
\end{cases}$$

(5.13)

Last, note that in equation (5.11), $E_F$ and $E_I$ are designed to select the first or last state vector from the lifted sequence, $\mathbf{x}_j$. Thus they are defined as

$$E_F \triangleq \begin{bmatrix} 0_{n_x \times n_x (n_s-1)} & I_{n_x \times n_x} \\
I_{n_x \times n_x} & 0_{n_x \times n_x (n_s-1)}
\end{bmatrix},$$

(5.14)

$$E_I \triangleq \begin{bmatrix} I_{n_x \times n_x} & 0_{n_x \times n_x (n_s-1)} \\
0_{n_x \times n_x (n_s-1)} & I_{n_x \times n_x}
\end{bmatrix}.$$  

(5.15)

Note here that because the initial conditions are not reset between iterations, the final condition from iteration $j$ is the initial condition of iteration $j+1$. Thus, the last term in (5.11) expresses the deviation in initial conditions between iteration $j$ and iteration $j+1$.

There are a few points worth emphasizing here:

• Everything in this dynamic model is described or parameterized with respect to $s$, the position along the path, not time. Utilizing this lifted system model to derive an ILC update law will then lead to a formulation where the timing characteristics can vary. However, the path-domain characteristics such as waypoint arrival position and duration are constant from one iteration to the next. Although the range of the variable $t$ may change from one iteration to the next, the total range spanned by $s$ will not. Furthermore, the location of waypoints in terms of $t$ may change, but the location of waypoints in terms of $s$ will not.

• Specification of waypoints is straightforward and intuitive. The designer selects specific values of $\sigma$ that correspond to points in space that they would like to track. This lifts the requirement that waypoints be specified in time and instead allows them to be specified in space.

• This lifted system model explicitly considers the impact of nonzero initial conditions. This is critical for deriving an ILC formulation that allows the initial conditions to vary from one
5.2 Receding Horizon Norm-Optimal Economic ILC

For many systems, it is suboptimal or infeasible to reset the initial conditions of the system between iterations. In systems without an initial condition reset, the control sequence chosen for one iteration has a direct impact on subsequent iterations. Therefore, in order to optimize the control sequence, one must account for this interdependence. To capture this interdependence, I first developed a dynamic model to capture the behavior of the system over multiple future iterations. This model can then be used to estimate the performance of the system over multiple future iterations. That estimate can then be used to estimate performance in order to derive learning filters using the same techniques as Section 2.2.

The procedure for deriving the update law and learning filters is similar to other works such as [17]. First, a performance index is written in terms of the lifted model, then the gradient of that performance index is set it equal to the zero vector. Then, by rearranging the result an update law and expressions for learning filters are obtained.

To clarify the following math and notation, I wish to define a standard of notation and nomenclature to differentiate symbols and quantities. So far, \( x_j \) and \( u_j \) have been referred to as “lifted” vectors. That is, they are formed by concatenating a set of state or control input vectors ordered according to their path position (or time stamp) as in equation (5.10). Similarly, the matrices \( G_j \) and \( F_j \) have been referred to as “lifted” system matrices, lifted model matrices, or just simply lifted matrices.

In this section, it will be necessary to concatenate multiple instances of vectors and matrices that are already lifted. These further concatenated vectors and block matrices will be referred to as “super-lifted” to emphasize that they are different from, but dependent upon, vectors and matrices from the previous section which themselves are already lifted. These super-lifted vectors and matrices will be notated with a bold-faced symbol.

5.2.1 Lifting in the Iteration Domain

In general, the lifted model of equation (5.11) can be used to predict the state sequence resulting from any control input. Thus, if one wished to predict the state sequence over the \((j + n_i)\)th iteration, they could write

\[
x_{j+n_i} = x_j + G_j (u_{j+n_i} - u_j) + F_j (E_F - E_I) x_{j+n_i-1}.
\]  

(5.16)

Noting the recursive form of this relationship, \( x_{j+n_i} \) depends on \( x_{j+n_i-1} \), it can use it to derive a relationship between the super-lifted state and control vectors, \( u_{j+1} \), and \( u_{j+1} \) which are defined
via a similar structure to equation (5.10) as

\[
x_{j+1} \triangleq \begin{bmatrix} x_{j+1} \\ x_{j+2} \\ \vdots \\ x_{j+n_i-1} \\ x_{j+n_i} \end{bmatrix}, \quad \text{and} \quad u_{j+1} \triangleq \begin{bmatrix} u_{j+1} \\ u_{j+2} \\ \vdots \\ u_{j+n_i-1} \\ u_{j+n_i} \end{bmatrix},
\]

(5.17)

respectively. The relationship between \(x_{j+1}\), and \(u_{j+1}\) is then described by

\[
x_{j+1} = (I_x + F_{j}) x_j + G_{j} (u_{j+1} - I_u u_j)
\]

(5.18)

where

\[
F_{j} \triangleq \begin{bmatrix} I \\ I + F_j E_F \\ \vdots \\ I + \sum_{k=1}^{n_i-1} \prod_{m=1}^{k} F_j E_F \\ \vdots \\ I + \sum_{k=1}^{n_i-1} \prod_{m=1}^{k} F_j E_F \end{bmatrix},
\]

(5.19)

\[
G_{j} \triangleq \begin{bmatrix} G_{j} & \cdots & \cdots \\ F_j E_F G_{j} & G_{j} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{m=1}^{n_i} F_j E_F G_{j} & \prod_{m=1}^{n_i-1} F_j E_F G_{j} & \cdots & F_j E_F G_{j} G_{j} \end{bmatrix},
\]

(5.20)

and \(I_j\) is defined as an appropriately sized identity matrix, \(I^{n_i \times n_i}_{n_i \times n_i}\), repeated vertically, \(n_i\) times. Thus, this super-lifted dynamic model then maps

- the state sequence from the previous iteration, \(x_j\), and
- the control input sequences of multiple future iteratons, \(u_{j+1}\),


to the state sequence of multiple future iterations, \(x_{j+1}\).

Similarly to the super-lifted state and control sequences, one can define a super-lifted reference signal, which can then be used to calculate a super-lifted error sequence as

\[
e_{j+1} = I_r \overline{r} - x_{j+1}
\]

(5.21)

where \(\overline{r}\) is the lifted reference signal vector constructed from an ordered set of reference states, \(\{r_0, r_1, \ldots, r_{n_i-1}, r_{n_i}\}\) where \(r_q \in \mathbb{R}^{n_x}\) according to the same lifted structure as shown in equation (5.10).
Here, it is assumed that at every point along the path (or in time) there is a reference state, \( r \in \mathbb{R}^{n_x} \). Although it may appear initially that this requires the prescription of all states at every point along the path, that is not the case. In practice, one can build a reference state vector that contains the desired values of the states for some states at some points along the path, and zeros (or any other values) in the places corresponding to other states or points along the path. Then, by careful construction of the weighting matrices in the performance index (and subsequent learning filters), error relative to these zero-valued states will not be penalized. Thus “padding” the reference state vector with zeros or specifying a value at non-waypoint locations has no effect on the end result.

### 5.2.2 Derivation of Optimal Learning Filters

In order to address a general form of the norm optimal ILC problem, the following general performance index is used:

\[
J_{j+N} = \sum_{k=1}^{N} \left[ u_{j+k}^T Q_u u_{j+k} + \left( u_{j+k} - u_{j+k-1} \right)^T Q_\delta u \left( u_{j+k} - u_{j+k-1} \right) \right. \\
+ \left. x_{j+k}^T Q_x x_{j+k} + \left( x_{j+k} - x_{j+k-1} \right)^T Q_\delta x \left( x_{j+k} - x_{j+k-1} \right) \right] \\
+ \left. e_{j+k}^T Q_e e_{j+k} + \left( e_{j+k} - e_{j+k-1} \right)^T Q_\delta e \left( e_{j+k} - e_{j+k-1} \right) + S_x x_{j+k} \right].
\]  

(5.22)

The diagonal positive definite weighting matrices of the quadratic terms, \( Q_u, Q_\delta u, Q_x, Q_\delta x, Q_e, \) and \( Q_\delta e \), are defined in detail in appendix A.3. They follow the standard norm-optimal point-to-point structure detailed in other work such as [17].

The last term of equation (5.22) is introduced in order to accommodate economic objectives. The quantity \( S_x \in \mathbb{R}^{n_x \times n_x} \) is taken to be a weighted linear approximation (or exact representation) of the true economic objective that one would like to optimize. If the economic objective is described by \( J_e(x, u) \), then

\[
S_x = \left[ \begin{array}{c} \nabla_x J_e(x, u)_{x=x_l(s_1)}^{u=u_l(s_1)} \\ \vdots \\ \nabla_x J_e(x, u)_{x=x_l(s_{ns})}^{u=u_l(s_{ns})} \end{array} \right]^T Q_{sx},
\]  

(5.23)

where \( Q_{sx} \) is a diagonal weighting matrix designed to weight and also normalize the states in the overall performance. Although it may appear restrictive to require a linear economic objective, this is not the case, since any economic objective can be locally approximated as linear.

The performance index of equation (5.22) can now be written in an equivalent super-block...
structure using the super-lifted vectors $x_{j+1}$, and $u_{j+1}$:

$$J_{j+N} = u_{j+1}^T Q_u u_{j+1} + u_{j+1}^T D_u^T Q_\delta u_d u_{j+1} + (E_u u_{j+1} - u_j)^T Q_\delta u (E_u u_{j+1} - u_j)$$

$$+ x_{j+1}^T Q_x x_{j+1} + x_{j+1}^T D_x^T Q_\delta x D_x x_{j+1} + (E_x x_{j+1} - x_j)^T Q_\delta x (E_x x_{j+1} - x_j)$$

$$+ e_{j+1}^T Q_e e_{j+1} + e_{j+1}^T D_e^T Q_\delta e D_e e_{j+1} + (E_e e_{j+1} - e_j)^T Q_\delta e (E_e e_{j+1} - e_j) + S x_{j+1}$$

(5.24)

where $D_u \in \mathbb{R}^{(n-1)i_n \times n_i n_s n_s}$, $D_x \in \mathbb{R}^{(n-1)i_n \times n_i n_s n_s}$, and $D_e \in \mathbb{R}^{(n-1)i_n \times n_i n_s n_s}$ are difference operator matrices designed to calculate the difference between sequences during subsequent iterations according to

$$D_{(i)} = \begin{bmatrix} I & -I & \cdots & \cdots \\ I & -I & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & I \\ \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}$$

where the identity matrix has dimensions of $n_{(i)} n_i \times n_{(i)} n_s$. Note that $D_{(i)}$ has $n_i - 1$ block rows and $n_i$ block columns. Additionally, the matrices $E_u$, $E_x$, and $E_e$ are defined in such a way as to select the first lifted vector from the super-vector. Specifically,

$$E_{(i)} = \begin{bmatrix} T_{n_{(i)} n_i} n_i & 0_{n_{(i)} n_i \times (n_i - 1)n_i} \\ \end{bmatrix}$$

(5.26)

Then, by using the super-lifted dynamic model of equation (5.18) in equations (5.21) and (5.24), one can obtain an expression for the performance index completely in terms of the known quantities from the last iteration, $u_j$ and $x_j$, along with the control sequences over the next several iterations, $u_{j+1}$. By then differentiating this expression for performance, $J_{j+1}$, with respect to the elements of $u_{j+1}$, setting the result equal to the zero vector, and re-arranging the resulting expression, one obtains an update law of the form

$$u_{j+1} = L_u u_j + L_e e_j + L_x x_j + L_c$$

(5.27)

(5.28)

where

$$L_u = L_0 \left( Q_u + G_j^T (Q_x + \hat{Q}_x) G_j \right)^{-1},$$

(5.29)

$$L_e = L_0 \left( E^T u Q_\delta u + G_j^T (Q_x + \hat{Q}_x) G_j I_u \right),$$

(5.30)

$$L_x = L_0 G_j^T \left( E^T_x Q_\delta x - \hat{Q}_x I_x - \hat{Q}_x F_j - \hat{Q}_e F_j \right),$$

(5.31)

$$L_e = L_0 G_j^T \left( \hat{Q}_e I_e - E^T_e Q_\delta e \right),$$

(5.32)

$$L_c = - \frac{1}{2} L_0 G_j^T S x,$$

(5.33)
and
\[
\hat{\mathbf{Q}}_u \triangleq \mathbf{Q}_u + \mathbf{D}_u^T \mathbf{Q}_u \mathbf{D}_u + \mathbf{E}_u^T \mathbf{Q}_u \mathbf{E}_u, \tag{5.34}
\]
\[
\hat{\mathbf{Q}}_x \triangleq \mathbf{Q}_x + \mathbf{D}_x^T \mathbf{Q}_x \mathbf{D}_x + \mathbf{E}_x^T \mathbf{Q}_x \mathbf{E}_x, \tag{5.35}
\]
\[
\hat{\mathbf{Q}}_e \triangleq \mathbf{Q}_e + \mathbf{D}_e^T \mathbf{Q}_e \mathbf{D}_e + \mathbf{E}_e^T \mathbf{Q}_e \mathbf{E}_e. \tag{5.36}
\]

It should be emphasized here that this update law produces the super-lifted vector \( \mathbf{u}_{j+1} \), which contains control sequences for multiple future iterations. Thus, while it would be technically possible to perform multiple iterations based on this single update law, I choose to only run a single iteration before performing this update again. This is justified by the same reasons that are usually cited for using different control and prediction horizons in model predictive control (MPC). By updating the control sequence after every iteration, the algorithm incorporates the most recent information, making the system more robust to disturbances and modeling uncertainties. I describe the resulting control structure as receding horizon iterative learning control (RHILC).

Finally, note that by considering multiple future iterations in equations (5.18) and (5.24), combined with the lifted model of equation (5.11), which explicitly considers the impact of nonzero initial conditions, the algorithm inherently considers the impact of one iteration on the next. This helps to ensure that one iteration does not end in a state that negatively impacts subsequent iterations.

### 5.3 Linear Programming Based Economic ILC

In this methodology, the ILC update to the control input sequence is calculated as an explicit solution to an economic optimization problem. This work is included in [48]. The ILC update law is then given as
\[
\mathbf{u}^{ILC}_{j+1}(s) = \mathbf{u}^{ILC}_j(s) + \delta \mathbf{u}^*(s), \tag{5.37}
\]
where \( \delta \mathbf{u}^*(s) \) serves the role of a correction to the previous iteration’s feedforward control sequence and is based on the solution to a linear program (LP). To compute \( \delta \mathbf{u}^*(s) \), the objective function is linearized with respect to both the control and state sequences, resulting in the approximation:
\[
\delta J(\delta \mathbf{x}_j) = \nabla_x J(\delta \mathbf{x}_j) + \nabla_u J(\delta \mathbf{u}_j), \tag{5.38}
\]
where:
\[
\nabla_x J(\delta \mathbf{x}_j) = \psi_x \delta \mathbf{x} + \psi_u \delta \mathbf{u}, \tag{5.39}
\]
\[
\nabla_u J(\delta \mathbf{u}_j) = \psi_u \delta \mathbf{u}. \tag{5.40}
\]
The optimized correction in the control sequence is then computed according to the following LP:

\[
\delta u^* = \arg \max_{\delta u} \Psi_x \delta \dot{x}_j + \Psi_{uu} \delta u_j
\]

subject to:

\[
-\Delta_{tol} \leq \Psi_w \delta \dot{x} \leq \Delta_{tol}
\]

\[
b_l \leq \delta u \leq b_u,
\]

where \(\Psi_w\) is a diagonal matrix with ones in diagonal entries corresponding to values of \(s\) where waypoint tracking inequality constraints are to be imposed and zeros everywhere else.

The imposition of inequality constraints, rather than equality constraints, for waypoint tracking, is significant. Because the lifted system representation depends upon the rate at which the path is traversed \((\frac{d}{dt})\), the lifted system matrix implicitly (but not explicitly) depends on waypoint arrival times. In order to render waypoint arrival times flexible, the waypoints themselves must be made flexible. Furthermore, because it is well known that an LP will, in all but limiting special cases, result in a boundary solution, saturation constraints on \(\delta u\) are important. The result of this implementation is an ILC update that “nudges” the control sequence in the economically optimal direction at each iteration.

5.4 Combined Flexible Time and Receding Horizon ILC

Section 5.1 presented a dynamic model in the form of a lifted system representation that was independent of time. Then Section 5.2 presented the derivation of optimal learning filters based on a performance index that includes economic objectives and also considers multiple future iterations, thus inherently accommodating systems without an initial condition reset. Finally, Section 5.3 presented the formulation of an iterative update law as an explicit solution to an economic optimization problem that was capable of handling economic optimizations. It should be noted that the results of Section 5.1 can be used independently of the results in Sections 5.2 and 5.3 and vice versa.

If one wants to consider a system wherein the timing characteristics can vary from iteration to iteration, but the initial conditions are reset between iterations, this can be accomplished by using learning filters that are based on this path-parameterized model but do not consider multiple future iterations. Thus, the path-parameterized lifted model can be seen as the key feature in allowing for flexible-time behavior from iteration to iteration.

Conversely, if one wants to consider a system wherein the timing characteristics are fixed from one iteration to the next, but the initial conditions are not reset, then the designer can apply the results of Sections 5.2 or 5.3 to a more conventional lifted system model that is parameterized with respect to time. Thus, the performance index of (5.24), when combined with a lifted system model such as (5.11) that explicitly accounts for the impact of initial conditions, can be seen as the key feature that enables us to lift the requirement that initial conditions be reset between iterations.

One key point is that when the path-parameterized model is used in an ILC update law, the
result will be a control sequence parameterized with respect to the path position, \( s \), not \( t \). Therefore, it will be necessary to map all of the results (the control sequence for the next iteration) back into time-domain when performing an iteration or simulation. However, this is a relatively simple and straightforward task that can accomplished via a lookup table, as shown in Figure 5.2. This lookup table selects the correct control input, \( u^{ilc} \), from the lifted sequence, \( u_{j+1} \), produced by the ILC update.

**Figure 5.2** Block diagram of the control structure used for the path-parameterized ILC update law.

5.5 **Application to Four State Model**

In order to assess the performance of the linear-programming based path optimization ILC, I first applied the linear programming based path following algorithm to the four-state model as provided in Section 3.1.1. The analysis and results of this section are included in [48]. The values of critical system parameters have been tuned to reflect an airborne wind system, and values used in simulation are given in Table 5.1.

In order to assess the performance of the AWE system, two quantities are considered, namely the value of the performance index from iteration to iteration and a direct measure of energy capture as captured by the energy augmentation ratio in equation (2.27).

The performance index as a function of iteration (figure-eight lap) is shown in Figure 5.3. The progression of \( EAR \) as a function of iteration number is shown in Figure 5.4. A comparison of
Table 5.1  Parameter values used in simulation subjecting the four state model to the linear programming-based path following ILC algorithm.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
<td>Path width</td>
<td>50</td>
<td>m</td>
</tr>
<tr>
<td>$H$</td>
<td>Path height</td>
<td>15</td>
<td>m</td>
</tr>
<tr>
<td>$V_{ss}$</td>
<td>Max steady state speed</td>
<td>10</td>
<td>m/s</td>
</tr>
<tr>
<td>$\tau_v$</td>
<td>Speed time constant</td>
<td>1</td>
<td>s</td>
</tr>
<tr>
<td>$\tau_{\psi}$</td>
<td>Heading time constant</td>
<td>0.1</td>
<td>s</td>
</tr>
<tr>
<td>$S_w$</td>
<td>Waypoint path parameters</td>
<td>${0.25, 0.5, 0.75, 1}$</td>
<td>-</td>
</tr>
<tr>
<td>$\Delta_x$</td>
<td>Waypoint x tolerance</td>
<td>0.25</td>
<td>m</td>
</tr>
<tr>
<td>$\Delta_y$</td>
<td>Waypoint y tolerance</td>
<td>0.25</td>
<td>m</td>
</tr>
<tr>
<td>$\Delta_{\psi}$</td>
<td>Final allowable hdg. error</td>
<td>25</td>
<td>deg</td>
</tr>
<tr>
<td>$\delta u_{lim}$</td>
<td>Max allowable ctrl. chg.</td>
<td>1</td>
<td>deg</td>
</tr>
<tr>
<td>$\Delta_s$</td>
<td>Path discretization level</td>
<td>0.005</td>
<td>-</td>
</tr>
<tr>
<td>$v_{wind}$</td>
<td>Wind speed</td>
<td>4</td>
<td>m/s</td>
</tr>
<tr>
<td>$s_t$</td>
<td>Target lead distance</td>
<td>0.02</td>
<td>-</td>
</tr>
</tbody>
</table>

Figures 5.3 and 5.4 confirms that the average velocity-based performance index used as a surrogate economic metric in this work indeed correlates extremely well with the $EAR$. This is unsurprising considering that energy augmentation is driven by apparent wind speed, which is heavily driven by crosswind flight speed, $v$. Faster crosswind flight almost always correlates with greater energy generation. Most importantly, it can be seen from these figures that the performance is indeed improving (and converging) as the iterations progress.

Figure 5.5 shows the progression of flight paths at selected iterations. This figure shows that small changes in the flight path between waypoints can lead to significant (nearly 20 percent) additional energy augmentation.

Finally, Figure 5.6 compares the instantaneous energy augmentation ratio, given by equation (2.28), which can be termed the power augmentation ratio, over the course of a lap. In addition to the later laps exhibiting a higher average power augmentation ratio, it can be observed that these laps are completed in a shorter amount of time (approximately 90 percent of the time taken in the initial lap). This underscores the importance of a flexible-time ILC update strategy.

5.6 Application to Unifoil Model Model

To test the effectiveness of the receding horizon ILC method, I simulated an MHK unifoil system with a 10 meter tip-to-tip wingspan in a spatiotemporally constant flow environment. Table 5.2 shows the plant parameter values used in simulation.

In order to maximize power generation, the weights of the economic $S_x$ term of (5.22) are selected to incentivize maximizing the speed state of the model. Table 5.3 shows the controller parameter values used in simulation. Note that many of the variable names presented in Table 5.3 are defined in the appendix. For this work, waypoints were placed at $s = \frac{1}{4}$, $s = \frac{1}{2}$, $s = \frac{3}{4}$, and $s = 1$. 

81
Figure 5.3 Performance index vs. iteration number for a four-state model of an AWE system as subjected to the linear programming-based path optimization ILC algorithm.

Figure 5.4 EAR vs. iteration number, as calculated by (2.27) for a four-state model of an AWE system as subjected to the linear programming-based path optimization ILC algorithm.

Figure 5.7 shows how the shape of the achieved flight path changes from the first iteration to the last iteration. Additionally, the nominal path that is used to calculate path position and waypoint locations is shown as a finely dotted gray line. The waypoints are also indicated on the plot. One can see from this figure that the achieved flight path of the first iteration does not accurately track the waypoints whereas the final flight path does. One can also see that the final flight path starts and ends at nearly the same point, suggesting that the initial conditions of each iteration have stopped changing from iteration to iteration.

Figure 5.8 shows how the total performance index changes from iteration to iteration. Note that this performance index is the actual achieved performance, which is not the same as equations (5.22)
or (5.24), which capture predicted values of future performance over multiple iterations. Instead, this plot shows the actual value of the performance index as achieved by the system on the last iteration. This is given by

\[
J_j = u_j^T Q_u u_j + \left( u_{j-1} - u_j \right)^T Q_\delta u \left( u_{j-1} - u_j \right) + x_j^T Q_x x_j + \left( x_{j-1} - x_j \right)^T Q_\delta x \left( x_{j-1} - x_j \right)
\]

\[
+ e_j^T Q_e e_j + \left( e_{j-1} - e_j \right)^T Q_\delta e \left( e_{j-1} - e_j \right) + S_x x_j.
\]  

(5.44)

Note that this function does not have a value until the end of the second iteration because the \( \delta \) quantities are undefined until then.
Next, I examine some specific components of the performance that are of particular interest in this application. Figure 5.9 illustrates the convergence of the waypoint tracking term in the performance index, $e_j^T Q_e e_j$. This plot illustrates how over the first several iterations, the flown flight path achieves better and better waypoint tracking.

Next, to examine the economic performance and variable timing characteristics, Figure 5.10 shows the economic incentive term in the performance index, mean speed achieved over each iteration and the total iteration time of each iteration. From these plots one can infer that the economic incentive term correlates well with the mean speed of the system, suggesting that the choice to maximize the sum of each speed state at every point along the path captures the economic objective well. Finally, the bottom plot shows that the total iteration time decreased from approximately 57 seconds on the first iteration, to about 25 seconds on the final iteration, a decrease of 56%.

The last figure, Figure 5.11, shows the instantaneous energy augmentation ratio of equation (2.28).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>Mass</td>
<td>2857</td>
<td>kg</td>
</tr>
<tr>
<td>$I_{zz}$</td>
<td>Rotational inertia about $\vec{z}_B$</td>
<td>24675</td>
<td>kg m$^2$</td>
</tr>
<tr>
<td>$m_{add}$</td>
<td>Added mass</td>
<td>134</td>
<td>kg</td>
</tr>
<tr>
<td>$I_{zz}$</td>
<td>Added rotational inertia about $\vec{z}_B$</td>
<td>23320</td>
<td>kg m$^2$</td>
</tr>
<tr>
<td>$A_{w ref}^w$</td>
<td>Wing reference area</td>
<td>10</td>
<td>m$^2$</td>
</tr>
<tr>
<td>$A_{v ref}^v$</td>
<td>Vertical stabilizer reference Area</td>
<td>2</td>
<td>m$^2$</td>
</tr>
<tr>
<td>$A_{h ref}^h$</td>
<td>Horizontal stabilizer reference Area</td>
<td>3</td>
<td>m$^2$</td>
</tr>
<tr>
<td>-</td>
<td>Wing aspect ratio</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>Vertical stabilizer aspect ratio</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>Horizontal stabilizer aspect ratio</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>Oswald efficiency factor (all surfaces)</td>
<td>0.8</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>Sphere radius</td>
<td>100</td>
<td>m</td>
</tr>
<tr>
<td>$v_f$</td>
<td>Flow speed</td>
<td>1</td>
<td>m/s</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Fluid density</td>
<td>1000</td>
<td>kg/m$^3$</td>
</tr>
</tbody>
</table>
Figure 5.7 Path shape over the first and last iteration. Note that the achieved flight path over the last iteration tracks the waypoints much more closely than the first iteration.

Figure 5.8 Performance index over the course of 50 iterations. Note that lower values indicate better performance. Also, the negative values are only possible because of the presence of the economic, $S_x$ term in (5.22).
Table 5.3 Controller parameter values used in simulation results of the unifoil model as subjected to the norm optimal path following ILC algorithm. Note that the mathematical expressions that utilize the penalty weights are given in Appendix A.3

### Lower Level Path Following Controller

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_0$</td>
<td>Path $\phi$ sweep angle</td>
<td>60</td>
<td>deg</td>
</tr>
<tr>
<td>$\phi_1$</td>
<td>Path mean $\phi$ angle</td>
<td>0</td>
<td>deg</td>
</tr>
<tr>
<td>$\theta_0$</td>
<td>Path $\theta$ sweep angle</td>
<td>20</td>
<td>deg</td>
</tr>
<tr>
<td>$\theta_1$</td>
<td>Path mean $\theta$ angle</td>
<td>30</td>
<td>deg</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Reference model time constant</td>
<td>0.25</td>
<td>s</td>
</tr>
<tr>
<td>$s_0$</td>
<td>Target path point offset</td>
<td>0.05</td>
<td>-</td>
</tr>
<tr>
<td>$-\nu$</td>
<td>$\nu$ setpoint ($j=0$ only)</td>
<td>3.5</td>
<td>m/s</td>
</tr>
<tr>
<td>$-\nu$</td>
<td>$\nu$ feedback gain ($j=0$ only)</td>
<td>0.249</td>
<td>rad s/m</td>
</tr>
</tbody>
</table>

### Upper Level ILC

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_s$</td>
<td>Number of discretization points</td>
<td>400</td>
<td>-</td>
</tr>
<tr>
<td>$q_{u,w}$</td>
<td>$u_w$ penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{u,r}$</td>
<td>$u_r$ penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{\delta u,w}$</td>
<td>$u_w$ deviation penalty weight</td>
<td>8.207</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{\delta u,r}$</td>
<td>$u_r$ deviation penalty weight</td>
<td>8.207</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{e,\phi}$</td>
<td>$\phi$ error penalty weight</td>
<td>$8.2 \times 10^4$</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{e,\theta}$</td>
<td>$\theta$ error penalty weight</td>
<td>$8.2 \times 10^4$</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{e,\nu}$</td>
<td>$\nu$ error penalty weight</td>
<td>0</td>
<td>(m/s(^{-2}))</td>
</tr>
<tr>
<td>$q_{e,\psi}$</td>
<td>$\psi$ error penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{e,\omega}$</td>
<td>$\omega$ error penalty weight</td>
<td>0</td>
<td>rad(^{-4})</td>
</tr>
<tr>
<td>$q_{\delta e,\phi}$</td>
<td>$\phi$ error deviation penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{\delta e,\theta}$</td>
<td>$\theta$ error deviation penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{\delta e,\nu}$</td>
<td>$\nu$ error deviation penalty weight</td>
<td>0</td>
<td>(m/s(^{-2}))</td>
</tr>
<tr>
<td>$q_{\delta e,\psi}$</td>
<td>$\psi$ error deviation penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{\delta e,\omega}$</td>
<td>$\omega$ deviation error penalty weight</td>
<td>0</td>
<td>rad(^{-4})</td>
</tr>
<tr>
<td>$q_{x,\phi}$</td>
<td>$\phi$ penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{x,\theta}$</td>
<td>$\theta$ penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{x,\nu}$</td>
<td>$\nu$ penalty weight</td>
<td>0</td>
<td>(m/s(^{-2}))</td>
</tr>
<tr>
<td>$q_{x,\psi}$</td>
<td>$\psi$ penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{x,\omega}$</td>
<td>$\omega$ penalty weight</td>
<td>0</td>
<td>rad(^{-4})</td>
</tr>
<tr>
<td>$q_{\delta x,\phi}$</td>
<td>$\phi$ deviation penalty weight</td>
<td>8.207</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{\delta x,\theta}$</td>
<td>$\theta$ deviation penalty weight</td>
<td>8.207</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{\delta x,\nu}$</td>
<td>$\nu$ deviation penalty weight</td>
<td>0.0025</td>
<td>(m/s(^{-2}))</td>
</tr>
<tr>
<td>$q_{\delta x,\psi}$</td>
<td>$\psi$ deviation penalty weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
<tr>
<td>$q_{\delta x,\omega}$</td>
<td>$\omega$ deviation penalty weight</td>
<td>0</td>
<td>rad(^{-4})</td>
</tr>
<tr>
<td>$s_{x,\phi}$</td>
<td>$\phi$ incentive weight</td>
<td>0</td>
<td>rad(^{-1})</td>
</tr>
<tr>
<td>$s_{x,\theta}$</td>
<td>$\theta$ incentive weight</td>
<td>0</td>
<td>rad(^{-1})</td>
</tr>
<tr>
<td>$s_{x,\nu}$</td>
<td>$\nu$ incentive weight</td>
<td>$-0.0025$</td>
<td>(m/s(^{-1}))</td>
</tr>
<tr>
<td>$s_{x,\psi}$</td>
<td>$\psi$ incentive weight</td>
<td>0</td>
<td>rad(^{-1})</td>
</tr>
<tr>
<td>$s_{x,\omega}$</td>
<td>$\psi$ rate incentive weight</td>
<td>0</td>
<td>rad(^{-2})</td>
</tr>
</tbody>
</table>
Figure 5.9 Waypoint tracking performance achieved by the unifoil model as subjected to the receding horizon path following ILC algorithm. Note that a low waypoint tracking penalty indicates better performance.
Figure 5.10 Iteration duration, mean speed and economic incentive term in $J$, as achieved by the unifoil model when subjected to the receding horizon path following ILC algorithm. For the top and bottom plot, a low value indicates better performance, while for the middle plot, a high value indicates better performance.
**Figure 5.11** Power factor, $P_f$ versus iteration number as achieved by the unifoil model as subjected to the receding horizon path following ILC algorithm. A larger power factor indicates better system performance.
CONCLUSIONS AND FUTURE WORK

This chapter summarizes the key achievements of this work and also proposes possible avenues of investigation for future researchers.

6.1 Conclusions

The overarching goal of this work has been to develop an economic optimization structure based on iterative learning control in order to optimize the behavior of repetitive path following systems. In order to accomplish this goal, I developed iterative learning control-based tools that allow for i) economic optimization, ii) iteration-varying time-domain behavior (waypoint arrival times and iteration durations), and iii) continuous operation, without a system reset between iterations.

This work proposed two novel optimal control frameworks for repetitive, path-following systems. These are:

- **Path optimization ILC**, which considers how to optimize the shape of the desired path, which is then passed to a lower level controller that is tasked with following the prescribed path, and

- **Path following ILC**, which considers how to optimally follow a prescribed path. This method is then in turn composed of two key contributions. The first contribution, flexible time ILC, hinges on the reparameterization of the model into the path domain. The second contribution, receding horizon ILC, lifts the requirement that initial conditions be reset, enabling continuous operation of the system.

It then proved the effectiveness of these methodologies on a sequence of dynamic models of increasing complexity.
6.2 Future Work

The work presented in this document contains several novel contributions. As such, there exist significant opportunities for further investigation into the consequences and implications of these contributions. These include at least five clear opportunities:

- The fusion of path optimization and path following ILC into a single control structure.
- An investigation of the convergence and stability properties of path-domain ILC.
- An investigation of the convergence and stability properties of receding horizon ILC.
- An extension of the path-following ILC algorithm to non-smooth paths.
- An experimental verification of the path optimization ILC algorithm.

In order to fully optimize a repetitive path-following system such as a tethered energy system, one should consider not just how to optimize the path geometry or how to optimally follow a prescribed path, rather, one should consider how to do both simultaneously. In order to address this challenge, it may be possible to structure the combined controller (including elements from both Chapter 4 and Chapter 5) as a hierarchical controller. In this combined controller, the path optimization ILC would operate on an outer loop, and the path following ILC would operate in an inner loop. However, due to the fact that both algorithms are seeking to optimize the same performance index, this structure may lead to undesirable behavior if not implemented carefully. In order to avoid this, it may be possible to introduce an iteration-scale separation, analogous to the time-scale separation commonly found in hierarchical control. This iteration scale separation will ensure that the lower level path-following ILC is given enough iterations to fully converge before the shape of the path is changed by the upper level path optimization ILC algorithm.

The second item in the above list proposes to investigate theoretical properties of the path-domain ILC algorithm. As mentioned in Section 5.4, it is possible to implement ILC on a path-domain model, as detailed in Section 5.1. The resulting algorithm will inherently allow for flexible waypoint arrival times and iteration durations. However, this requires i) that the dynamic model be converted into the path-domain, and ii) that the resulting sequence be mapped back into the time-domain during the next iteration. It is not immediately clear at this point how these two operations will impact the stability and convergence of an ILC algorithm, and therefore this should be investigated.

The third item from the list above suggests an investigation into the properties of receding horizon ILC. Because the structure of this algorithm so closely parallels the structure of time-domain receding horizon control, it may be possible to adapt stability, convergence, and recursive feasibility proofs from work in receding horizon control. However, because of the heavy reliance on terminal penalties in MPC convergence analysis, this may prove to be nontrivial.

The fourth item from this list of potential future work is an extension to non-smooth paths. The work of Chapter 4 assumes that the path shape function, $\vec{p}(\sigma, b)$, is smooth and twice differentiable.
with respect to the path variable, \( \sigma \). In practice, many systems may follow paths that are not smooth and differentiable, thus the expression of equation (5.2) is undefined at some locations. To address this, it may be possible to first develop a strategy that addresses two adjoining straight-line path segments. One could then approximate any path (smooth or otherwise) as a sequence of many shorter straight-line paths.

Finally, having demonstrated the efficacy of the path following ILC algorithm on an unconstrained and unsimplified dynamic model in Section 4.3.3, it seems appropriate to consider real-world implementation of this algorithm. This work could build off works by this author such as [45] and [49] wherein a lab-scale tethered energy system control prototyping platform was developed and dynamic scaling laws were verified. However, this verification will likely prove nontrivial due to the fact that the structure of the controller used in experiments is different from the path following flight controller used in simulation. This reflects the fact that the experimental system is actuated using multiple tethers, while the simulation model is actuated using control surfaces. In order to reconcile this difference, future work can focus on developing a path following controller that is appropriate for a lab scale system that is actuated via tethers, and not via control surfaces.
BIBLIOGRAPHY


[3] Team, R. O. M. “Pick and Place Robots: What Are They Used For and How Do They Benefit Manufacturers?” Robotics.org ()


APPENDIX
APPENDIX

A

MATHEMATICAL TOOLS

A.1 Golden Section Algorithm

The golden section one-dimensional minimization algorithm is used extensively throughout this work. Whenever a path-following algorithm is implemented it relies on the golden section algorithm to calculate the path position, $s$, by solving the one-dimensional minimization problem of 2.16. Thus, we include a short summary of the method here.

Supposing that we are given a range, $[\sigma_1, \sigma_2]$, that is known to contain a minimum of the function, $J(\sigma)$, then the golden section algorithm is given in Algorithm 1.

A.2 Rodrigues' Rotation Formula

Rodrigues' rotation formula is used throughout this work to rotate one vector, $\vec{v} \in \mathbb{R}^3$, about an axis described by a unit vector, $\vec{u} \in \mathbb{R}^3$, by a prescribed angle, $\theta \in \mathbb{R}$, according to the right hand rule. The rotation formula that describes the new vector, $\vec{v}'$, is then

$$\vec{v}' = \vec{v} \cos(\theta) + (\vec{u} \times \vec{v}) \sin(\theta) + \vec{u} (\vec{u} \cdot \vec{v})(1 - \cos(\theta)) \quad (A.1)$$

A.3 ILC Performance Weighting Matrices

In the point-to-point weighting matrices of the performance index in equation (5.22), the weighting matrices $Q_u$, $Q_{\delta u}$, $Q_e$, $Q_{\delta e}$, $Q_x$, and $Q_{\delta x}$, are easily defined in terms of three functions,
Algorithm 1 Golden section minimization algorithm used throughout this work

\[
\gamma \leftarrow \frac{\sqrt{5}+1}{2} \\
J_1 \leftarrow f(\sigma_1) \\
J_4 \leftarrow f(\sigma_4) \\
\sigma_3 \leftarrow \sigma_1 + \frac{\sigma_4 - \sigma_1}{\gamma} \\
\sigma_2 \leftarrow \sigma_4 - \frac{\sigma_4 - \sigma_1}{\gamma} \\
J_3 \leftarrow f(\sigma_3) \\
J_4 \leftarrow f(\sigma_4) \\
\textbf{while } \sigma_4 - \sigma_1 > T \textbf{ do} \\
\quad \textbf{if then} J_2 < J_3 \\
\quad \quad \sigma_1 \leftarrow \sigma_2 \\
\quad \quad J_1 \leftarrow J_2 \\
\quad \quad \sigma_2 \leftarrow \sigma_3 \\
\quad \quad J_2 \leftarrow J_3 \\
\quad \quad \sigma_3 \leftarrow \sigma_1 + \frac{\sigma_4 - \sigma_1}{\gamma} \\
\quad \quad J_3 \leftarrow f(\sigma_3) \\
\quad \textbf{else} \\
\quad \quad \sigma_4 \leftarrow \sigma_3 \\
\quad \quad J_4 \leftarrow J_3 \\
\quad \quad \sigma_3 \leftarrow \sigma_2 \\
\quad \quad J_3 \leftarrow J_2 \\
\quad \quad \sigma_2 \leftarrow \sigma_4 - \frac{\sigma_4 - \sigma_1}{\gamma} \\
\quad \quad J_2 \leftarrow f(\sigma_2) \\
\quad \textbf{end if} \\
\textbf{end while}
\]
• the “diagonal” matrix function, \( f_D : \mathbb{R}^{n_a} \rightarrow \mathbb{R}^{n_a \times n_a} \) which takes in an input vector, \( v = [v_1, v_2, \ldots, v_{n_a-1}, v_{n_a}] \), and outputs a matrix with the entries of the input vector along the diagonal and zeros elsewhere.

\[
f_D(v) = \begin{bmatrix}
v_1 & 0 & \ldots & 0 & 0 \\
0 & v_2 & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \ldots & v_{n_a-1} & 0 \\
0 & 0 & \ldots & 0 & v_{n_a-1}
\end{bmatrix}
\] (A.2)

• the “repeated block diagonal” matrix function, \( f_\Delta : (n_b, \mathbb{R}^{n_c \times n_d}) \rightarrow \mathbb{R}^{n_b n_c \times n_b n_d} \) which accepts the positive integer, \( n_b \), and an arbitrary matrix, \( C \in \mathbb{R}^{n_c \times n_d} \), and returns a block matrix with the input matrix repeated \( n_b \) times along the block diagonal.

\[
f_\Delta(n_b, C) \triangleq \begin{bmatrix}
C & \ldots \\
\vdots & \ddots \\
C & \ldots \\
\vdots & \ddots \\
\end{bmatrix}
\] (A.3)

• the “waypoint selection” matrix function \( W : (n_s, n_x, S_{w,i}) \rightarrow \mathbb{R}^{n_s n_x \times n_s n_x} \) which accepts the number of steps along the path, \( n_s \), the number of states, \( n_x \), and the set of indices corresponding to waypoint locations along the path, \( S_{w,i} \). The block element, \( w \), in the \( i \)th block row and \( j \)th block column \( w_{i,j} \) is given by

\[
w_{i,j} = \begin{cases}
I_{n_x \times n_x} & \text{if } i = j \text{ and } i \in S_{w,i}, \\
0_{n_x \times n_x} & \text{otherwise}.
\end{cases}
\] (A.4)

If we then define the vectors of user-selected scalar weights as

\[
\begin{align*}
q_u & \triangleq [q_{u,w} \quad q_{u,r}]^T, \\
q_{\delta u} & \triangleq [q_{\delta u,w} \quad q_{\delta u,r}]^T, \\
q_e & \triangleq [q_{e,\phi} \quad q_{e,\theta} \quad q_{e,v} \quad q_{e,\psi} \quad q_{e,\omega}]^T, \\
q_{\delta e} & \triangleq [q_{\delta e,\phi} \quad q_{\delta e,\theta} \quad q_{\delta e,v} \quad q_{\delta e,\psi} \quad q_{\delta e,\omega}]^T, \\
q_x & \triangleq [q_{x,\phi} \quad q_{x,\theta} \quad q_{x,v} \quad q_{x,\psi} \quad q_{x,\omega}]^T, \\
q_{\delta x} & \triangleq [q_{\delta x,\phi} \quad q_{\delta x,\theta} \quad q_{\delta x,v} \quad q_{\delta x,\psi} \quad q_{\delta x,\omega}]^T, \\
s_x & \triangleq [s_{x,\phi} \quad s_{x,\theta} \quad s_{x,v} \quad s_{x,\psi} \quad s_{x,\omega}]^T.
\end{align*}
\] (A.5)

Then the gain matrices \( Q_u, Q_{\delta u}, Q_e, Q_{\delta e}, Q_x \) that encode the relative importance of each term in
the performance index are given by

\[
Q_u = f_\Delta(n_s, f_D(q_u)), \\
Q_{\delta u} = f_\Delta(n_s, f_D(q_{\delta u})), \\
Q_e = W(n_s, n_x, S_{u,i})f_\Delta(n_s, f_D(q_e)), \\
Q_{\delta e} = W(n_s, n_x, S_{u,i})f_\Delta(n_s, f_D(q_{\delta e})), \\
Q_x = f_\Delta(n_s, f_D(q_x)), \\
Q_{\delta x} = f_\Delta(n_s, f_D(q_{\delta x})).
\]

(A.6)