Online Adaptive Model-Free MIMO Control of Lighter-Than-Air Dirigible Airship

by

Derek Boase

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Applied Sciencein
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Examining Committee

The following served on the Examining Committee for this thesis.

Carleton Member: Dr. Shichao Liu
Associate Professor, Department of Electronics
Carleton University, Canada

Internal Member: Dr. Davide Spinello
Full Professor, Department of Mechanical Engineering
University of Ottawa, Canada

Supervisor(s): Dr. Wail Gueaieb
Professor, School of Electrical Engineering & Computer Science
University of Ottawa, Canada

Dr. Suruz Miah
Adjunct Professor, School of Electrical Engineering & Computer Science
University of Ottawa, Canada

Associate Professor, Department of Electrical & Computer Engineering
Bradley University, Peoria, IL, 61625, USA

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

With the recent advances in the field of unmanned aerial vehicles, many applications have been identified. In tasks that require high-payload-to-weight ratios, flight times in the order of days, reduced noise and/or hovering capabilities, lighter-than-air vehicles present themselves as a competitive platform compared to fixed-wing and rotor based vehicles. The limiting factor in their widespread use in autonomous applications comes from the complexity of the control task. The so-called airships are highly-susceptible to aerodynamic forces and pose complex nonlinear system dynamics that complicate their modeling and control. Model-free control lends itself well as a solution to this type of problem, as it derives its control policies using input-output data, and can therefore learn complex dynamics and handle uncertain or unknown parameters and disturbances.

In this work, two multi-input multi-output algorithms are presented on the basis of optimal control theory. Leveraging results from reinforcement learning, a single layer, partially connected neural network is formulated as a value function appropriator in accordance with Weierstrass higher-order approximation theorem. The so-called critic-network is updated using gradient descent methods on the mean-squared error of the temporal difference equation. In the single-network controller, the control policy is formulated as a closed form equation that is parameterized on the weights of the critic-network. A second controller is proposed that uses a second single-layer partially connected neural network, the actor-network, to calculate the control action. The actor-network is also updated using gradient descent on the squared error of the temporal difference equation.

The controllers are employed in a highly realistic simulation airship model in nominal conditions and in the presence of external disturbances in the form of turbulent wind. To verify the validity and test the sensitivity of the algorithms to design parameters (the initialization of certain terms), ablation studies are carried out with multiple initial parameters. Both of the proposed algorithms are able to track the desired waypoints in both the nominal and disturbed flight tests. Furthermore, the performance of the controllers is compared to a modern, state-of-the-art multi-input multi-output controller. The two proposed controllers outperform the comparison controller in all but one flight test, with up to four fold reduction in the integral absolute error and integral time absolute error metrics. On top of the quantitative improvements seen in the proposed controllers, both controllers demonstrate a reduction in system oscillation and actuator chattering with respect to the comparison algorithm.

Acknowledgements

The completion of this work has many layers and has been made possible through the contributions of many individuals.

Firstly, I'd like to acknowledge my thesis supervisors, Dr. Wail Gueaieb and Dr. Suruz Miah, for their support throughout my thesis and during the publication process. They both displayed limitless patience and guidance during the sometimes daunting endeavour and always made it clear that my success was as much a priority to them as anything else. Their unwavering standards and passion for the subject are second to none.

This thesis's development sometimes felt more like a war of attrition than technical research. Algorithms breaking although they once worked, a single missed negation nested among 1000+ lines of code, compilation errors, etc. To push beyond these bumps in the road, the motivation to continue needs to overcome the motivation to quit. For me, this motivation to stay the course came by way of childlike curiosity, a curiosity that is reinforced every day through my children's desire to learn. Whether it's documenting every type of fungus they fall into while mountain biking or having to stop everything to look at any aircraft that goes overhead, their obsession with knowing why is wonderfully contagious.

Curiosity and ambition are only small parts of the recipe for success, without opportunity, they are all for not. This opportunity came from the sacrifices of my wife, Marie. Without her commitment and constant trust and belief in me, this process would not have even begun. With all that she has done over the many years, any accomplishment that is achieved or milestone that is reached is done together.

Lastly, I'd like to thank the authors of [58], Price et al., for contributing the simulation model used in this work to the research community, and I'd like to thank the financial contributions of NSERC and the University of Ottawa.

Table of Contents

List of Tables

List of Figures

Acronyms

BIBO Bounded-Input Bounded-Output 10, 23

BSMC Backstepping Sliding Mode Control 16–18

C.G. Center of Gravity 8, 13

C.V. Center of Volume 8, 13

CFDL Compact Form Dynamic Linearization 21–24, 62

DDC Data-Driven Control 21, 25, 41, 46, 47, 55, 91–93, 123, 124

DLT Dynamic Linearization Techniques 21, 23, 24, 60, 62, 118

DoF Degree of Freedom 12, 56, 91, 93

FFDL Full Form Dynamic Linearization ix–xi, 23, 32, 61, 62, 67–70, 72, 76, 77, 82, 84, 86–88, 90, 91, 103, 104, 106–108, 115, 118, 120, 121

HJB Hamilton-Jacobi-Bellman 27–29, 31, 48

i-PID intelligent-PID 24, 25, 32, 33, 36

IAE Integral Absolute Error 32, 63, 64, 70–72, 86–88, 90, 91, 100, 106–108, 118, 120, 121, 123

IMU Inertial Measurement Unit 57, 59

ITAE Integral Time Absolute Error 32, 63, 64, 70–72, 86–88, 90, 91, 100, 106–108, 118, 120, 121, 123

LMI Linear Matrix Inequalities 26

LQR Linear Quadratic Regulator 22, 26, 32, 33, 44, 49

LTAVs Lighter Than Air Vehicles ix, 1, 2, 5, 7, 8, 10, 13, 19, 38, 41, 67, 124

MET Main Error Tunnel 12, 13

MFC Model-Free Control 4, 5, 7, 21, 24, 28, 30, 36, 38–42, 44, 45, 47, 60, 62, 67, 72, 123

MIMO Multi-Input Multi-Output 4, 5, 7, 10, 21, 22, 25, 36, 41, 42, 47, 51, 62, 67, 92, 93, 108, 123, 124

P Proportional 38

PD Proportional Derivative 9, 12, 13

PFDL Partial Form Dynamic Linearization 21–24, 29

PI Proportional Integral 12, 38

PID Proportional Integral Derivative 9, 15, 18, 19, 24, 30, 32, 35, 36, 39–41, 57, 70

PPD Pseudo-Partial Derivative 21, 22, 24, 25

PPJM Pseudo-Partitioned Jacobian Matrix 23, 61, 75

RMS Root-Mean-Square 33, 37

ROS Robot Operating System 3, 5, 38, 40, 56, 57, 90, 124, 139

SISO Single-Input Single-Output xii, 18–22, 25, 28, 29, 36, 41

SMC Sliding Mode Control 10, 15–18, 41

TET Temporary Error Tunnel 12, 13

UAVs Unmanned Aerial Vehicles 1, 7, 21, 32, 41

Chapter 1

Introduction

This Chapter introduces the reader to the high-level ideas that follow in this work, outlining the motivation for the research, the research contributions and scholarly outcomes from this research period. Section 1.1 offers a summarized history of Lighter Than Air Vehicles (LTAVs) research and uses. The motivation for seeking a model independent control scheme for dirigible airships is introduced in Section 1.2 with the objective and contributions being discussed in Section 1.3. The thesis organization is outlined in Section 1.4 with a tabulated list of the scholarly outcomes of this research being available in Section 1.5.

1.1 Background

In this section, the reader is introduced to a brief history of the development of the state-of-the-art for airships research, with a focus on the research interest surrounding autonomous flight. Benefits of dirigibles over their unmanned counterparts, as well as tasks for which LTAVs are well suited are introduced.

Remark 1.1. *Herein, for readability, the terms airship, dirigible and LTAVs are used interchangeably.*

In modern times the term Unmanned Aerial Vehicles (UAVs) is ubiquitous in technological industries with multirotor aircraft finding a great deal of application. The popularity of these vehicles has followed into the research field. This trend shared between industry and academia has accelerated the understanding and development of drone flight [58]. The

popularity of these vehicles has bred the development of open-sourced models and simulation environments that allow researchers to more easily investigate control strategies. In more recent years, research into using LTAVs has begun its resurgence. LTAVs generate their lifting force by using gasses such as Helium to reduce the effective weight of the vehicle [49]. This design philosophy provides numerous advantages compared to the multirotor or fixed-wing alternatives that make dirigibles best suited for certain tasks. A brief comparison of their performance characteristics for aerial sensing and monitoring missions is provided in [20] and summarized with slight modification in Table 1.1.

Table 1.1: Comparison between fixed-wing, rotor and LTAVs for aerial sensing applications. The performance markers are given by ×, where × = Poor, ×× = Mediocre, × × × = Best

Characteristic	Fixed-Wing	Rotor	Lighter-Than-Air
Hovering	×	× × ×	× × ×
Long flight times	××	×	× × ×
Low electrical noise	×	×	× × ×
Low cost	××	×	× × ×
Low vibration	××	×	× × ×
Low speed flight	×	× × ×	× × ×
Maneuverability	××	× × ×	×
Payload-to-weight ratio	××	×	× × ×
Simplicity	××	×	× × ×
Vertical Take-Off and Landing	×	× × ×	× × ×

Expanding on the advantages outlined above, it is clear to see that there are a range of tasks for which LTAVs are ideally suited, including but not limited to [19, 44, 58]:

- Transport
- Surveillance
- Freight Carrier
- Monitoring (traffic, wildlife, vegetation)
- Adverts
- Urban planning
- Inspection (pipeline, power lines, architectural development, railways)
- Law enforcement
- Telecommunications relay
- Agricultural studies
- Atmospheric and limnological pollutants research

- Characterization of plants and animals • Cinematography

Published in 1998, [19] introduce a paper targeted at advancing the state-of-the-art of autonomous flight for airships. The paper discusses project AURORA (Autonomous Unmanned Remote mOnitoring Robotic Airship) with the mission statement of decomposing the task into research surrounding the navigation, control, sensing and inference for semi-autonomous airships. The ultimate goal is the transformation of, what was at the time, teleoperation of dirigibles, to telemonitoring of autonomous vehicles. The work in [19] and subsequent works in [8, 13, 20, 26, 52] pushed this goal forward delivering different philosophies of airship designs, and optimizing the perception task of the vehicles. During this time empirically derived aerodynamic coefficients for common airship designs were inferred through wind tunnel testing and application. These advances allowed for the development of classical control techniques implemented using linearization techniques on uncertain nonlinear models [44, 74].

With the development of project AURORA autonomous airship control gained research popularity. With modern advances in small-form computational power and intelligent control, autonomous airship control seemed more attainable. In 2022, Price et al. developed and released an open-source Robot Operating System (ROS) compatible simulation environment, in [58], that is shown to behave in a highly realistic manner, taking into account the non-rigid structure of a volume airship, and incorporated turbulent wind using a Dryden wind filter. This development, as well as others like it, allow for low-cost, rapid-iterative prototyping of control algorithms specific to this vehicle architecture, that may be implemented without risk to any physical components. The availability of these types of simulation models and environments also makes offline training and dirigible integration of multi-robot systems, including airships as seen in [57].

1.2 Motivation

In Section 1.1, the use of autonomous aerial vehicles as a tool in a subset of aerial tasks is presented, with a brief history leading up to the modern state-of-the-art. In this section, the need for the algorithms developed in this work is justified.

At the time of this work, the autonomous control of airships remains an open issue with the most successful approaches coming recently in [43, 44, 57, 58]. These works leverage intuition of the vehicle dynamics through virtual axes used in the LibrePilot autopilot software, although they are not explicitly represented with any type of vehicle-specific modeling. In [43, 44], the techniques that are used to control the airship are

trained offline for multiple days with high-powered and costly graphical processing units and therefore require a great deal of computational power on top of having limitations in their ability to generalize, which is a common downfall of offline techniques [62]. With this considered, there is room for an online adaptive airship control strategy that is independent of vehicle dynamics, whether modeled explicitly or implemented via-virtual axes. This type of formulation does not constrain its application to a singular vehicle type or structure and allows it to generalize well, given its online nature [62], while allowing for the possibility of controlling airships with a range of designs, requiring only minimal modification by the designer(s).

1.3 Objectives and Contributions

In Section 1.2, the need for a vehicle-agnostic online adaptive airship control strategy is outlined. In the work presented herein, the objective is to develop a Multi-Input Multi-Output (MIMO) Model-Free Control (MFC) strategy that learns online through interactions with its environment, by way of measurements gathered along its trajectories. The control algorithm should be lightweight to allow for onboard computations, respecting the desired frequency requirements of the designer. The controller must be robust to external turbulent wind disturbances, successfully tracking the desired waypoints, despite the presence of wind. The contributions of this work are:

- Two online MIMO MFC control structures using single and double neural-network architectures for value and action function approximation.
- A framework for applying both controllers to a highly realistic airship simulation model for altitude and altitude and velocity control tasks.
- An open-source GitHub repository for the application of the controller to the simulation model. [1]

1.4 Thesis Outline

As stated in Section 1.3, two algorithms are developed herein and tested in both nominal and windy test flights. In Chapter 2 related work on the topic of airship control and machine

[1] Found here

learning theory are discussed, with a focus on MFC and it's application to LTAVs. Previous classical control attempts are discussed in Section 2.1, while Section 2.2 introduces previous works on the mathematical development of machine-learning control, Section 2.2.1, and it's application to dirigible control, Section 2.2.2.

In Chapter 3, the mathematical preliminaries are introduced, reviewing the system definition in Section 3.1 and laying the foundation for reinforcement learning and it's application to optimal control theory in Section 3.2. Here the problems to be solved by the algorithms presented in Chapters 4 and 5 are explicitly discussed.

The structure of Chapters 4 and 5 is similar with the controller developments for the single-network (critic) and double network controllers being outlined in Sections 4.1 and 5.1, respectively.

In Sections 4.2 and 5.2 the experimental details and simulation results for the altitude control using the single-network controller are discussed. Section 4.2.1 highlights the ROS/Gazebo simulation environment and the experimental parameters are introduced for the altitude control task for both proposed controllers in Section 4.2.2 and the comparison controller, Section 4.2.3.

In Section 4.2.4 for the single-network controller and Section 5.2.1 for the double-network controller, the learning rate parameter is tested for the altitude control task to determine the sensitivity of the controllers to the critic and actor learning rates, respectively. Taking the learning rates from the previous sections, Sections 4.2.5 and 5.2.2 consider the single waypoint altitude control task for the single and double-network controllers, respectively. Sections 4.2.5 and 5.2.2 contains an ablation study performed over many initial values to determine the sensitivity of the algorithms to their starting conditions. From there, two sets of initial values are used to perform nominal and wind disturbed test flights to verify the robustness of the algorithms to turbulent wind. The nominal and wind disturbed experiments are repeated in Section 4.2.6 for the single-network controller and Section 5.2.3 for the double-network controllers, with the addition of the results from the comparison algorithms for a multiple waypoint trajectory. This experiment tests the ability of the controller to track multiple altitude waypoints, and quantifies the performance against a state-of-the-art MIMO MFC strategy. The results of altitude control experiments are then summarized with concluding remarks for each controller in Sections 4.2.7 and 5.2.4.

The complexity of the task is augmented in Sections 4.3 and 5.3 where the altitude and velocity, along with their time derivatives are controlled using two actuators. The experimental parameters are introduced in Sections 4.3.1 and 4.3.2 for the proposed single-network controller and the comparison controller, respectively.

The structure of this section is similar to its predecessor with Sections 4.3.3 and 5.3.1 performing an ablation studies over multiple initial conditions for the single and double-network controllers, before testing a subset of the responses from the study in a nominal and disturbed test setting. The single-network controller is tested with a multi-waypoint desired trajectory task in Section 4.3.4 and the double-network controller is tested in Section 5.3.2, quoting results for both nominal and windy experiments and comparing the trajectories to those of a comparison controller. Concluding remarks for the altitude and velocity controllers are offered in Sections 4.3.5 and 5.3.3 before concluding the chapters with final remarks in Sections 4.4 and 5.4.

Chapter 6 offers concluding remarks recounting the performance of both controllers at a high-level, and offering some areas where the controllers may be expanded.

1.5 Scholarly Outcome

Some of the thesis research findings has been disseminated in the following publications:

- [12]: Derek Boase, Wail Gueaieb, and Suruz Miah. "Underactuated MIMO Airship Control Based on Online Data-Driven Reinforcement Learning". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. (Accepted: Jun. 2023). Detroit, MI, USA, Oct. 2023, pp. 1–8
- [1]: Mohamed Abouheaf, Derek Boase, Wail Gueaieb, Davide Spinello, and Salah Al-Sharhan. "Real-Time Measurement-Driven Reinforcement Learning Control Approach for Uncertain Nonlinear Systems". In: *Engineering Applications of Artificial Intelligence (Elsevier)* 122 (June 2023), pp. 1–18. DOI: 10.1016/j.engappai.2023.106029

Chapter 2

Literature Review

This chapter discusses the most relevant literature for control strategies regarding Unmanned Aerial Vehicles (UAVs) with a focus on Lighter Than Air Vehicles (LTAVs). Section 2.1 considers fundamental approaches using classical model-based control schemes for different models of dirigibles. This section outlines the assumptions and results with a brief description of the high-level modelling philosophies. In Section 2.2, machine learning techniques are introduced and broken down to a theoretical and practical capacity. Section 2.2.1, outlines the mathematical formulation of relevant machine learning control schemes with a heavy focus on Multi-Input Multi-Output (MIMO) Model-Free Control (MFC) strategies, while Section 2.2.2 outlines the current state-of-the-art of MFC as it is applied to UAVs. Section 2.2 dives deeper into the mathematical structure of the control schemes as well as the practical results quoted in the most relevant literature.

Remark 2.1. *The terms model-free and data-driven are used interchangeably.*

2.1 Model-Based Airship Control

In this section, model-dependent airship control strategies are discussed. With the resurgence in popularity of LTAVs, autonomous control of such vehicles has been the focus of many research teams with initial attempts consisting of classical methods. To implement such classical methods it is necessary to first model the airships as well as their dynamics [51]. The Newton-Euler method is commonly used for modelling dirigible dynamics in the literature [3, 26, 35–37, 49, 61, 79]. In doing so, authors develop equations of motion

accounting for the actual and apparent mass, centrifugal and Coriolis forces, and gravitational forces, to represent the dynamics of the airship in response to different forces acting upon it. This means that, in general, different airship designs require different models and therefore must be examined on an individual basis before a control scheme may be derived. This is an expensive and challenging consequence as designs are complex and often contain unique coefficients that are only available through a large number of empirical tests, as illustrated by Paiva et al. in [52]. It is also the case that there exists high variability in the design of airships, specifically with respect to thrusting techniques and the inclusion and orientation of control surfaces [58]. In addition, the inability to accurately mathematically model external disturbances and process variation leads to underperformance of the system when using classical control techniques [7, 47, 51].

Models are presented in [3, 26, 61, 79] for dirigible designs that rely on vector thrusting for pitch and altitude control and control surfaces for pitch and yaw control. An alternative model is proposed for smaller low-speed airships that achieve pitch and altitude control by translating the gondola along the longitudinal axis of the airship [4, 35–37, 49]. By moving the gondola the alignment between the Center of Gravity (C.G.) with respect to the fixed Center of Volume (C.V.)[1] is changed thereby affecting the pitch.

For all of the model-based control strategies that are considered, assumptions are made to simplify the dynamics irrespective of the design of the vehicle. It is commonly assumed that the airship is perfectly rigid and that the C.V. and C.G. are coplanar, these are referred to as the rigidity and symmetry assumptions, respectively. It is also important to note that for each model highly specific aerodynamic, inertial and mass coefficients are required. At the time of writing, the only means of establishing these values with any degree of accuracy is by empirically deriving them. In fact, in much of the literature, coefficients are taken from other papers despite using different models and LTAVs designs.

In [52], Paiva et al. develop an autonomous hierarchical flight controller for the model developed in [26]. The unknown coefficients from the model are estimated using empirically derived data that was acquired from over 600 [h] of wind tunnel experimentation on an AS800 airship. It is interesting to note that although the model is developed with the assumption of rigidity, the airship that is used for data acquisition has a non-rigid structure. A high-level block diagram of the control scheme used in [52] is shown in Figure 2.1. The controller aims to control the airship during take-off, cruise, turn, hover and landing flight modes. The supervision block is responsible for determining which flight mode the

[1] It is noted that the center of volume is not strictly fixed as leaks will affect its location, but it is taken for granted that the rate of change in position of the C.G. is much quicker than the rate of change in position of the C.V. therefore justifying this statement.

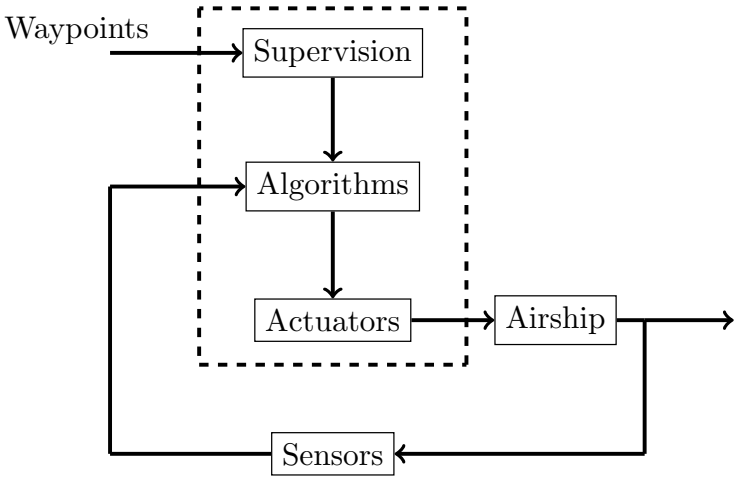


Figure 2.1: High-level block diagram of the control hierarchy used in [52]. The supervision block oversees trajectory and flight mode information, the algorithm block calculates the control action for the three control loops used in the model and the actuator block turns the control action input into low-level commands.

airship is currently operating in and monitoring the errors in cartesian space between the commanded pose and the actual pose of the airship. The algorithm block consists of three independent control loops for the velocity, height and heading control, respectively. The velocity controller is implemented using a Proportional Integral Derivative (PID) controller whose output is the control action sent to the propeller actuators for the thruster drives. The height and heading of the airship are both governed by Proportional Derivative (PD) controllers that calculate the control action sent to the elevator actuators and the rudder actuators, respectively. The actuator block turns the control actions sent from the respective controllers to low-level signals invoking the appropriate response for the associated hardware.

To validate the control scheme in [52] numerical simulations are performed on a trimmed model over an entire flight using Simulink. The model is trimmed around a longitudinal velocity of 12 [m/s] and a height of 100 [m]. The control gains for each controller are tuned via trial and error, employing a gain scheduling technique for the heading and velocity controllers. The results presented show that the controller is capable of achieving the desired performance from the model and demonstrates an intuitive response to constant perturbations that are meant to represent wind.

During the take-off region, the simulation reaches the target height of 100 [m] with overshoot in the order of 1 [m]. During the cruise flight, the simulation is disturbed by the

constant wind causing the controller to demand a lateral component of vehicle velocity of roughly 2 [m/s]. This lateral velocity component is removed at the 120 [s] mark when the airship is commanded to turn into the wind, which it successfully does. The hover mode is commanded at 180 [s] forcing the longitudinal component of velocity to decrease from 12 [m/s] to 0 [m/s] over a period of roughly 40 [s]. The final flight mode is the vertical landing performed over a time of 100 [s] bringing the altitude of the simulated airship from approximately 100 [m] to 0 [m].

While the controller proposed in [52] successfully eliminated the steady-state error in a Simulink environment, there are two important observations; despite the development of a complex and intricate model and use of 600 [h] (25 [d]) of wind tunnel data, the controllers were still tuned by trial and error. In addition, the validation of the controller was implemented on simulation of a trimmed model and therefore may not map to reality or extend to useful general flight characteristics as the performance of the vehicle is restricted to the knowledge of the model.

Building off of the results in [52], Carneiro de Paiva, Benjovengo, and Siqueira Bueno develop a robust MIMO Sliding Mode Control (SMC) controller in [13] by linearizing the model presented in [26] around a constant height of 100 [m] and constant airspeed of 5 [m/s]. The linearization of the dirigible model is common in airship control given the model complexity and nonlinearities [13] and has the consequence of decoupling the vertical and lateral motions of the airship. The authors argue that the model linearlization is justified based on the inherent robustness of SMC controller. This assumption is tested by exposing the nonlinear dynamic model of an AS800 airship to a constant unidirectional gust of wind with a speed of 3 [m/s], thereby introducing a range of airspeeds in the interval [2, 8] [m/s]. Despite having four states, the decoupled model of the horizontal motion of the airship is approximated as a second-order system which the authors argue is a common simplification used in this field [13]. The longitudinal controller controls the height and airspeed of the airship and the lateral controller controls the 2D trajectory following behaviour of the LTAVs. The SMC controller that is proposed is shown to be Bounded-Input Bounded-Output (BIBO) stable using the Lyapunov stability theory as outlined in [7] and is employed through the use of a soft-switching technique to lessen the prolonged high magnitude and high-frequency chattering that can accompany SMC controllers.

The SMC controller was tested using a nonlinear model environment as in [52]. The airship is commanded to fly at its trim point with a 2D rectangular trajectory with side lengths of 300 [m] and 200 [m]. A constant uniformly distributed wind is applied flowing parallel to the 300 [m] side length and perpendicular to the 200 [m] side length of the nominal path. The simulation begins with a headwind where the controller can respond

well to the tracking and velocity references. In the cross-wind sections, the controller begins to introduce overshoots of approximately 25 [m] turning into the tailwind and 10 [m] turning into the headwind. The authors argue that the tracking error at the corners of the rectangular trajectory could be significantly reduced by smoothing out the corners of the desired path. It is also observed that the rudder deflection begins to experience decaying oscillation, similar to the chattering phenomenon, in the cross-wind regions as the lateral controller attempts to correct the airship heading angle. In the tailwind region of the simulation, the airship experiences an overshoot of approximately 50 [m] while turning into the lower cross-wind section.

The simulations demonstrate success concerning the problem statement that the authors put forth, however, the simplifying assumption required for linearization, that is there exist only small perturbations from the trim point for the height and airspeed, limit the effectiveness of the controller in all flight modes. This linearization further decouples the vertical and horizontal flight dynamics, which may not be representative of a real system, despite the success of the controller in simulation.

An independent control scheme is presented in [60], where the authors also decouple the longitudinal and lateral system dynamics and control the speed, heading, pitch and height with independent controllers shown in Figure 2.2. The longitudinal controller is responsible

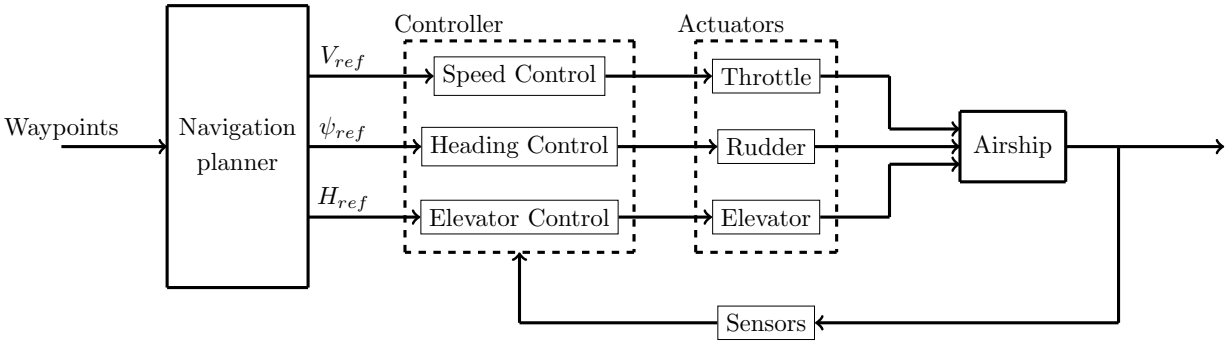


Figure 2.2: High-level block diagram of the control hierarchy used in [60]. The low-level controllers control the control action of the actuators and the high-level controller calculates the setpoints for the low-level controllers.

for controlling the longitudinal and vertical displacements and velocities as well as the pitch and pitch rates, using these as the outputs and having the elevator deflection and vector thrust angle as the control inputs. The lateral controller uses the lateral displacement and velocity and the roll and pitch, as well as their rates for system outputs, and the rudder deflection angle as the system input.

Control of the longitudinal dynamics is decomposed into two parts, velocity control and height control. The velocity control is realized with a Proportional Integral (PI) controller that is tuned based on the linearized model about a trim point. The altitude controller leverages a cascade control structure with the first block taking the height as the output and calculating the desired pitch which is then fed into the pitch controller. The output of the pitch controller is the required elevator deflection. The structure is shown in Figure 2.3 with the height controller and pitch controller both containing a fuzzy logic structure. The heading control is realized by leveraging a fuzzy logic controller in parallel with an integrator, where the integrator is included for robustness to noise and is reset every time a new waypoint is tracked. Both fuzzy logic controllers are tuned using improved genetic algorithms as an optimization technique for the controllers.

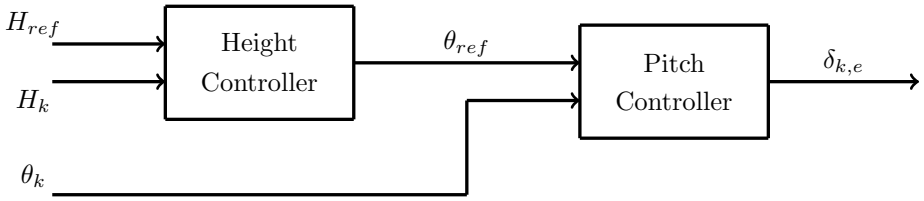


Figure 2.3: High-level block diagram of the control hierarchy used in [60]. The control scheme for the elevator controller in [60].

The controllers described above are tested in numerical simulation around a trimmed operating longitudinal velocity of 8 [m/s]. The airship is given four waypoints and successfully reduces the heading and altitude error in general, however, there were large deviations up to 55 [m] in the tracking error for the sharp corners in the trajectory path.

Adamski et al. propose a trajectory tracking algorithm in 3D space that uses error tunneling to control a modified model of an AS500 airship along a trajectory using six independent PD control loops, one for each Degree of Freedom (DoF). The model assumes that the rudders and elevators, herein the control surfaces, are fixed and that the two thrusters can operate independently. Additionally, it is assumed that the thrusters, which are initially close to the axis of symmetry are moved away from the airship by 1 [m] to allow for a larger range of turning torques. The error tunneling technique that is employed defines a Main Error Tunnel (MET) as a cylinder whose central axis is the nominal trajectory and whose radius is the acceptable error. A Temporary Error Tunnel (TET) is also defined which describes the trajectory that the airship is meant to take to return to the MET when the airship is not within it. The airship uses the TET if the error becomes too large during the flight, or if the initial position of the airship is not within the MET. The TET

is derived by using a 3rd order polynomial for the central axis and again having a radius equal to the acceptable temporary error.

Due to the complexity of the LTAVs model, the control gains for the PD controllers are determined numerically in Matlab using a built-in multicriteria genetic algorithm and selecting the gains for the linearized model. It is noted that while this is a valid approach for accuracy on the numerical model, translation to a physical structure may not yield comparable results. The validation of the control scheme is performed via simulation in a MATLAB environment with and without external disturbances. In both cases the airship is started at the origin of the world frame and the initial and terminal points of the trajectory to be followed are located at [300, 50, −80] [m] and [500, −50, −50] [m], respectively. In the simulation of the undisturbed flight, the algorithm successfully finds the final point of the nominal trajectory using a MET while following the nominal trajectory with the absolute value of the position error not exceeding 1.2 [m] in any direction. This error is well below the MET radius of $\delta_m = 10$ [m]. In the disturbed simulation the vehicle again follows the nominal trajectory within the MET until a spatially uniform gust of wind is simulated. The 3 [s] gust pushes the airship model outside of the MET so a TET is created bringing the airship back to the nominal path before it successfully finds the terminal point. The errors along the trajectory for the wind disturbed test case are not quoted in this reference.

In both cases, the airship found the initial and terminal trajectory points despite beginning outside of the error radius. While the trajectory tracking algorithm was successful in these simulations, it is once again important to note that the autonomous control of the airship was validated against a simplified model of the vehicle dynamics.

An adaptive variation to a fuzzy sliding mode control structure is proposed in [80] whereby the authors overcome the lack of systemic design techniques that restrict the use of fuzzy logic strategies by proposing an online adaptation law for the fuzzy sliding mode control. The controller that is proposed is based on a simplified airship attitude dynamic model that depends upon the parameters of an inertial matrix and the C.G. and C.V.. As mentioned previously, these terms are not static in the application, although they are often assumed as such. The stability of the proposed control strategy is proved mathematically using the Lyapunov candidate function.

The experimental validation is performed via numerical simulation using a Runge-Kutta method with variable steps. Two simulations are performed; one with constant nominal model parameters and no external disturbances, and another where the inertial coefficients are varied by 10% with respect to their nominal values and an external disturbance is simulated in the form of a torque applied to the attitude axes. In both cases, the controller can track the desired model trajectory and produce sliding functions that converge to 0.

An autonomous path tracking algorithm is proposed in [31], where the airship is treated as a point mass located at the center of the volume of the vehicle. In doing this, Kahale, Garcia, and Bestaoui consider only the equations of motion of the airship, significantly simplifying the problem formulation. The equations of motion for the guidance control system are derived from Newton's laws of motion allowing the authors to neglect the rotational velocity of the Earth.

The controllability and accessibility are proved for all flight modes for which the velocity is non-zero. The authors use robust control Lyapunov functions for tracking by using the pitch and roll rates and the 3D acceleration of the airship as control inputs. To verify the robustness of the controller, a linearized version of the airship dynamic model is numerically simulated in a nominal case and again in the presence of wind gusts produced using the von Karman power spectral density wind generator. In this paper, the authors discuss the use of both the Dryden wind model and the von Karman model but decide to use the latter as it is more easily implemented, although the Dryden model is more realistic. The simulation results show that the proposed controller is effective in eliminating the steady-state tracking error, in the presence of wind for both a linear and curved trajectory. It is noted that the trajectories are consistently increasing and show no convergence. This seems to be an unrealistic application of dirigibles as the hover and flight regimes of the z target should be static, at minimum, whereas in these tests, the altitude continuously increases. Nonetheless, the desired trajectories were successfully tracked.

In [41] the authors create a controller for a fin-less airship with four independently controllable thrusters. The model used for validation is based upon the Quanser MkII airship model presented in [48, 53]. The fin-less strategy is invoked to compensate for the stalling effect around the fins at high angles of attack and sideslip angles. Additionally, for hovering procedures and low-speed operations, the control surfaces have minimal influence compared to vector thrusting.

The design is composed of a high and low-level controller where the high-level control is responsible for calculating the desired velocity and attitude and the low-level controller calculates the thrust magnitude and thrust angles of each thruster. The velocity controller is an integral Lyapunov controller taking the desired velocity as an output and calculating the transnational acceleration as a control action. The attitude control is an integral backstepping structure that calculates the angular accelerations required to achieve the desired rotation described in quaternions. The thrust and thrust angles are then calculated from the desired accelerations using a static model that is developed.

The controller is tested using both numerical simulation and physical experimentation. In both the numerical simulation and the physical testing the controller was able to follow

the desired attitude with acceptable results. The altitude control did not map between the simulation and experimental results with the simulation showing little tracking error and the physical testing showing consistent deviations of $\pm$ 0.5 [m] with some of the errors exceeding 2 [m] of the target altitude of 4 [m]. The dirigible spends approximately 25 [s] on the ground during the final 35 [s] (representing an error of 100%) of the flight testing and seems to begin to diverge before the end of the experiment. The velocity had a similar but reduced error dynamic with the error being as high as 1.5 [m/s] (again, representing an error of 100%) and also appearing to begin to diverge toward the end of the experiment.

In the work presented by Recoskie in [61], a long-range hybrid airship is discussed. A variation of the airship model presented in [26] is presented in which the model is adapted to consider modelling and parametric uncertainties to add validity to the results. An input state matrix is also added to the model to adapt the model to the configuration of the dirigible that is used. Two control schemes, PID and backstepping controller, are compared in simulation with external disturbances simulated using a Dryden turbulence model.

The numerical experiments require the airship to travel through predefined waypoints, minimizing the error along the trajectory. The results show that the total travel time and distances are similar for both controllers with the backstepping controller having higher maximum errors in the altitude and trajectory deviation, despite yielding lower average errors in both metrics. This reduction of approximately 50% in the average error metrics came at the expense of a 40% increase in fuel consumption over the total flight time. A unique contribution of this work was the controller's ability to reduce error amongst a variety of flight conditions. As an example, the PID controller that is used for trajectory tracking is also used in hover flight with 7 m/s wind gusts, achieving an average error of 15.5 [m] and maximum error of 51.7 [m], which are comparable to other works that employ scheduling techniques.

A SMC controller is proposed in [81] in which the stability is proved using Lyapunov theory and the error is shown to converge asymptotically to 0. While this is desirable in the sense of error elimination, examination of the control actions demonstrates a high degree of chattering to achieve this control. To address this phenomenon Yang and Yan utilize a radial basis function neural network, taking the current sliding window and its derivative to the input layer and scheduling the control law between the different surfaces. The kernel layer of the network uses a Gaussian activation function whose weights are updated using backpropagation.

A numerical simulation is performed in MATLAB using a variable step Runge-Kutta integrator and disturbing the parameter with a random variable representing a 20% of the

parameters nominal value. The trajectory results are nearly indistinguishable between the SMC controller with and without the RBFNN. The difference is observed by looking into the smoothness of the actuation signal. The actuators for the controller with the neural network gain-scheduling were much smoother with small amplitude oscillations occurring far away from the saturation points of the control actions. This is in contrast to the original controller which seemed to oscillate between the high and low saturation values for each actuator.

In [79], Yang, Wu, and Zheng use a simplified dynamic equation that accounts for the mass and inertial mass, centrifugal and Coriolis forces and the damping effects of an underactuated airship that is controlled using control surfaces and propulsive motors. Two position controllers are implemented using the SMC and Backstepping Sliding Mode Control (BSMC) strategies. The objectives of the controllers are to asymptotically approach a target point. In developing the model, the authors assume for simplicity, a static equilibrium between the buoyancy and gravitational forces, and therefore exclude gravitational effects from the dynamic model. As a consequence of this, the control problem is restricted to controlling the 2D pose of the airship at a constant height of 500 [m]. Figure 2.4 shows the top view of the 2D space with the initial and final points. Furthermore, in the disturbed test cases, there is a spatially uniform time-varying wind blowing parallel to the x-axis. Both control schemes are simulated in MATLAB with and without disturbance

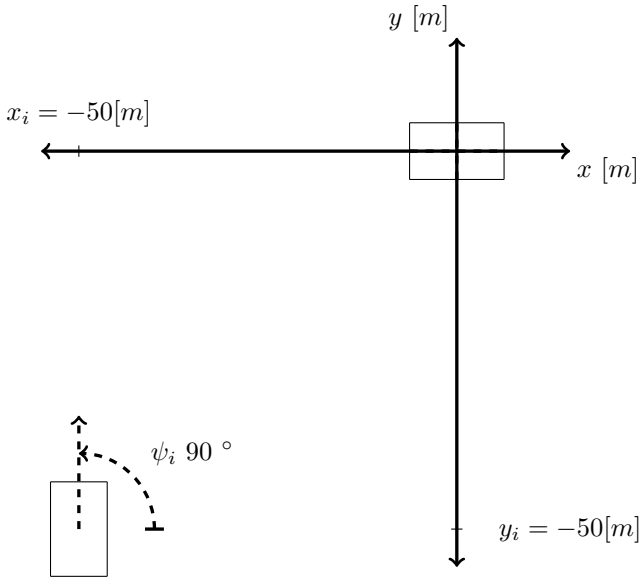


Figure 2.4: Initial and final point of the airship for the MATLAB simulation performed in [79].

and parametric uncertainties using a variable-step Runge-Kutta integrator. The parametric uncertainties are simulated by adjusting the model parameters to $\pm 10\%$ their nominal values and the disturbance is simulated by applying a lateral wind with a magnitude that is described by $d_{\text{wind}} = 10\cos(t)$ [m/s].

The effectiveness of the controller is evaluated based on the quality of the commanded control signal and convergence speed, with reduced chattering and smaller convergence durations being desired. The convergence results are summarized in Table 2.1.

Table 2.1: Convergence time for the simulated sliding mode control and backstepping sliding mode control controllers proposed in [79].

	Settling Times [s]		
Controller	x-axis	y-axis	ψ
SMC	500	250	50
disturbed SMC	N/A	300	30
BSMC	80	80	10
disturbed BSMC	N/A	300	50

From Table 2.1 it is observed that the position does not converge along the x-axis for either of the disturbed cases. In the case of the SMC controller the lateral velocity becomes bounded on the interval $[-3,\ 3]$ [m/s] and for the BSMC the lateral velocity becomes bounded on the interval $[-0.5,\ 0.5]$ [m/s], indicating that the controller is attempting larger error-driven adjustments for the SMC leading to higher-amplitude oscillations. A similar comparison is made for the chattering phenomenon of the two controllers. A tabulated comparison of the magnitude of the chattering effect for the control actions of each controller is demonstrated in Table 2.2.

While the BSMC controller shows promising improvement in the simulations compared to the SMC, it is noted that the validation method was again a limited simulation of a control strategy that is developed under simplifying assumptions and the rigidity and symmetry assumptions. As an example, the assumption of equilibrium between gravitational and buoyancy forces limits the controller in the event of a desired elevation change and removes control complexities associated with coupled behaviour between the controller constraints. With this in mind, it is unlikely that repeatable results would be obtained on a physical prototype.

More recently, different design philosophies for airships have presented themselves. In [4, 5, 35, 36, 49] a sliding ballast design is presented. In this approach, the pitch of the

Table 2.2: Chattering amplitudes for the simulated sliding mode control and backstepping sliding mode control controllers proposed in [79].

	Chattering Magnitude [N m]		
Controller	τ_u	τ_v	τ_r
SMC	10	25	400
disturbed SMC	16	35	460
BSMC	≈ 0	10	100
disturbed BSMC	≈ 0	14	110

airship is controlled by moving the gondola towards the front or back of the vehicle, thereby inducing a pitching moment on the airship. This addresses the physical constraints placed on traditional airship designs that prevent maneuverability in the vertical axis [4].

In [35] a Single-Input Single-Output (SISO) PID controller is developed to control the pitch of a sliding gondola airship model under the rigidity assumption. The controller is tested in numerical simulation by trimming the model presented in [36] around 180 [m] altitude and assuming constant thrust of 0.1 [N] is applied to each thruster. The linear analysis tool in MATLAB is used to deduce a trimmed model for the altitude, longitudinal velocity and pitch and a tuning tool from the control toolbox is used to find the controller gains for the given model. The trimmed model is exposed to low-frequency sinusoidal and square pulse reference trajectories for the pitch which it can track with minimal error. In this work a physical model is available, however, only open loop tests are performed as the purpose was to validate the pitch control methodology of the sliding gondola method.

In [4, 5] both numerical simulation and physical prototypes are controlled with a PID controller. The derivative term of the controller is replaced with a low pass filter, which is a common technique to avoid learning the high-frequency error that accompanies the system noise[7, 51].

A simulation is performed for a pitch controller with a constant wind gust with at an angle of -45 [°] from vertical. For a low-frequency sinusoidal reference pitch trajectory the tracking error of the SISO controller is sub 2 [°]. Using a square pulse trajectory, the model tracks the step changes in pitch, converging to the desired reference signal after approximately 15 [s].

The controllers are then tested empirically in two test cases; one with no applied thrust and one with a constant thrust input of 0.1 [N]. The PID gains from the simulation are applied to the controller of the physical system however the results were unstable. This

is likely because the empirical test cases are conducted too far away from the trimmed point, invalidating the model that was used to develop and validate the controller gains. The model derived gains will not be able to compensate for the modeling and parameter uncertainties associated with the transfer function. The tracking for the sinusoidal trajectory appeared to have an acceptable shape, with a significant phase shift. The tracking of multiple pulses and a single 20 [°] step struggled much more compared to the numerical simulation. The total average error is much larger and the response times to the step inputs are longer. It is noted that the focus of the paper was the theoretical and practical pitching mechanism associated with the sliding gondola and that was demonstrated effectively. The results from simulation to physical implementation, however, show that there is a disconnect between the two, likely due to the modeling complexity and the required linearization required to implement the controller.

The work presented in [49] presents an adaptive self-tuning PID controller with a supervisory loop and applies it to a LTAVs with a sliding ballast structure. The control problem is formulated as a SISO control problem where the displacement of the gondola is the control input and the output is the pitch of the airship. The system is modelled as a multi-body system, where the gondola and hull are independent bodies. The block diagram for this controller is shown in Figure 2.5 with the supervisory loop being a high-level loop that guarantees the stability of the response and the adaptation law being added for robustness against disturbances and uncertainties. The update laws for the adaptation block are based on a sliding mode condition. It was found that the supervisory loop induced chattering in the control actions and thus is only used when the error in the pitch exceeds some user-defined error limit otherwise the loop disconnected. This may be thought of a type of scheduling approach where the supervisory loop is used when the error exceeds a threshold.

The reference trajectory for the numerical simulations includes a low-frequency sinusoidal pitch trajectory and a multi-step input with linear transitions. In both cases, the model seems to smoothly track the reference trajectory without much chattering in the control action and the gains update as expected.

Navajas implements a pitch controller on a physical prototype developed at the University of Ottawa. Following an iterative trial and error approach for tuning the parameters, the best-suited controller is tested with a step trajectory and a multistep linear transition trajectory. The controller is also studied against external disturbances and partial actuator failure. The step experiment shows the pitch struggling to find the reference trajectory for the first 65 [s] of the 80 [s] experiment, with oscillations being present for the majority of the test. The multistep experiment is run for 220 [s]. Again the controller struggles for the first 65 [s] of the experiment and shows oscillations in the order of 7 [°] for a period of 20 [s].

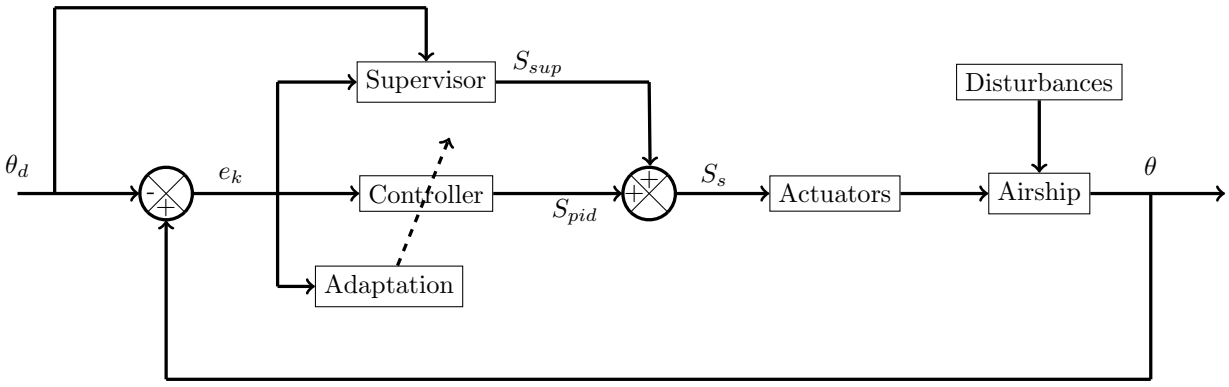


Figure 2.5: High-level overview of the SISO block diagram structure for the controller used in [49].

Following this high oscillation period, the controller tracks the pitch reference with little error, demonstrating only a minor lag compared to the reference. The two disturbance rejection tests are performed with step input reference trajectories for the pitch. The first is implemented with the addition of upwards wind gusts on the stationary dirigible and the second is done by disconnecting one of the actuators to the gondola. In both cases, the controller can adapt to the situation and minimize the pitch error. The notable difference between the two cases is that the latter introduces an overshoot in the order of 20%.

It is clear that the task of autonomous airship control is a widely studied and open problem in the research community. From the literature surrounding model-based control strategies for autonomous airship control, three things become evident:

- Airship models are highly complex, nonlinear and time-varying
- There is a great deal of uncertainty surrounding the parameters of the airship models and the external disturbances
- There is a high degree of variation across the design of airships

With these points in mind, machine learning lends itself well as a possible solution to autonomous dirigible navigation. In the following section the state-of-the-art in model-free machine learning control theory is introduced in Section 2.2.1 and applications to aerial vehicles are discussed in Section 2.2.2.

2.2 Machine Learning Based Control

With more modern developments in the field of machine learning and robotic control, popularity in the field of MFC has grown. In parallel, the advancements in computational speed and power accessible on single board computers, system memory and microcontrollers have made high dimensional online tasks more realistic for real-time robotic control [47]. Machine learning-based control strategies encompass a lot of benefits compared to their classical counterparts, namely in tracking performance, robustness and adaptability to uncertain parameters. MFC, also commonly referred to as Data-Driven Control (DDC) or Measurement-Driven Control, is used in many applications discussed in this section to control plants with no a priori knowledge of the system dynamics or environmental parameters. A limiting factor holding back this type of controller in industrial applications is the fact that, except for certain applications operating under specific assumptions, MFC strategies lack stability proofs guaranteeing efficient and repeatable tracking in all situations in the workspace[47]. A great deal of the proposed algorithms demonstrate their effectiveness through numerical and sometimes physical empirical testing [47]. This section is broken down by first discussing the state-of-the-art of MFC in Section 2.2.1 followed by case studies of this type of control strategy on UAVs in Section 2.2.2.

2.2.1 Model-Free Controllers: State-of-the-Art

In this section, the theoretical foundations of the state-of-the-art MFC strategies are outlined with a focus on the theory and mathematical development. This work discusses MIMO robotic control in detail, however, there is a great deal of literature for SISO robotic control and multi-agent systems that are introduced as the theoretical developments are relevant to the work presented in this work.

In 1994, Hou proposed a MFC in [29] relying on Dynamic Linearization Techniques (DLT). This idea relies on the change in output of a system being proportionally related to the change in input to the system. This work is expanded in [30] by introducing MFC based on Compact Form Dynamic Linearization (CFDL) and Partial Form Dynamic Linearization (PFDL), which are outlined below.

In CFDL the diagonally dominant square proportionality matrix, referred to as the Pseudo-Partial Derivative (PPD), is used to relate the changes in inputs and changes in outputs. The formulation of the CFDL over the time step is structured as

$$\Delta \mathbf{y}_{k+1} = \mathbf{\Phi}_k^{\text{cfdl}} \Delta \mathbf{u}_k \quad \text{where} \tag{2.1}$$

$$\Delta \mathbf{y}_{k+1} = \mathbf{y}_{k+1} - \mathbf{y}_k \in \mathbb{R}^n \quad \text{and} \quad \Delta \mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1} \in \mathbb{R}^n.$$

Note that, $\mathbf{\Phi}_k^{\text{cfdl}} \in \mathbb{R}^{n \times n}$. The dynamics of the plant are incorporated in the time-varying PPD matrix which is updated using the input-output data on each discrete-time step.

It is empirically noted in [30] that certain systems possess dynamics that are too complex to be captured in this structure of the CFDL. In such cases, Hou and Jin propose a variation to the CFDL technique that uses the previous L input samples in the estimate of the future output. As a result, the PPD and control action update laws also depend on the past L input values. With this adjustment, Equation (2.1) becomes

$$\Delta \mathbf{y}_{k+1} = \bar{\mathbf{\Phi}}_k^{\text{pfdl}} \Delta \bar{\mathbf{U}}_k \quad \text{with} \tag{2.2}$$

$$\bar{\mathbf{\Phi}}_k^{\text{pfdl}} = [\mathbf{\Phi}_{i,k}^{\text{cfdl}} \dots \mathbf{\Phi}_{L,k}^{\text{cfdl}}] \in \mathbb{R}^{n \times nL} \quad \text{and} \quad \bar{\mathbf{U}}_k = [\Delta \mathbf{u}_k^T, \dots, \Delta \mathbf{u}_{k-L+1}^T]^T \in \mathbb{R}^{nL}.$$

This windowed approach allows the input and output data to consider a designer-specified window of the input data, allowing the algorithm to take weighted past data into consideration in the updates.

In both algorithms, to allow the PPD to track time-varying dynamics and remain diagonally dominant, a resetting algorithm is introduced in which each element of the matrix is reset to its initial value if certain criteria are met. This is to ensure that $\|\Delta \mathbf{u}_k\| \neq 0$, $\|\Delta \bar{\mathbf{U}}_k\| \neq 0$ and that the control actions don't become too large.

Both CFDL and PFDL techniques are tested using numerical examples. Both algorithms are successful in tracking trajectories for numerically simulated MIMO nonlinear systems of order $n = 2$, with the CFDL showing larger amplitude oscillations in the response. It is emphasized here that the formulation of these algorithms requires the size of the inputs and outputs to be the same.

The update laws for both algorithms presented in [30] are derived based on a quadratic objective function in the inputs and outputs consistent with the Linear Quadratic Regulator (LQR) theory.

In [78], the CFDL approach is modified to add a weighting factor to the inputs for the previous L input terms in the update rule for the control action only. The so-called High-Order Model-Free Adaptive Control strategy then has the implementation simplicity and computational efficiency of the CFDL, without the complexity of the PFDL. In the same work Xu, Lin, and Chi also introduce the Improved High-Order Model-Free Adaptive Control strategy which takes the same form as the former, however, it includes a weighting term to the past L values of the PPD. In both algorithms introduced in [78] the sum of the L weighting terms must be equal to 1. These algorithms are introduced in the SISO

case and the numerical examples are restricted as such.

It is worth noting that the High-Order Model-Free Adaptive Controller and the Improved High-Order Model-Free Adaptive Controller are equivalent to the CFDL algorithm when the choice of $L = 1$ is applied, as required by their development.

In [77] a more recent DLT is presented with both numerical and simulation results to speak of. In the so-called Full Form Dynamic Linearization (FFDL) algorithm, the PFDL is augmented in a similar way to how the CFDL was augmented in [30]. In the FFDL the dynamically linearized model is written in the same form as (2.2), with the previous L_u inputs and L_y outputs being considered in the linearized models. That is

$$\Delta \mathbf{y}_{k+1} = \bar{\mathbf{\Phi}}_k^{\text{ffdl}} \Delta \bar{\mathbf{H}}_k \quad \text{with} \tag{2.3}$$

$$\Delta \bar{\mathbf{H}}_k = [\Delta \mathbf{y}_k^T, \ldots, \Delta \mathbf{y}_{k-L_y+1}^T, \Delta \mathbf{u}_k^T, \ldots, \Delta \mathbf{u}_{k-L_u+1}^T]^T$$

and $\Delta \mathbf{y}_{k+1}$ being defined as before. In the case of the FFDL, the proportionality term is referred to as the Pseudo-Partitioned Jacobian Matrix (PPJM) and is defined by

$$\bar{\mathbf{\Phi}}_k^{\text{ffdl}} = [\mathbf{\Phi}_{i,k}^{\text{ffdl}}, \ldots \mathbf{\Phi}_{L_y,k}^{\text{ffdl}}, \mathbf{\Phi}_{L_y+1,k}^{\text{ffdl}}, \ldots, \mathbf{\Phi}_{L_y+L_u,k}^{\text{ffdl}}].$$

The introduction of additional terms in the PPJM allows the updates to access a larger amount of input-output data from the interaction of the agent with the environment, but it also has the added advantage of allowing the order of the input and outputs to differ, making it a more general algorithm compared to those presented in [30, 78]. It was observed while reproducing the results from the work that there is a large sensitivity to the initial conditions, and thus while the larger number of parameters yields more adaptable parameters for the algorithm to learn, it was also significantly more difficult to find a set of initial conditions that lead to admissible results. The case study presented in [77] is done on a simulation quadrotor where the four control inputs are related to the angular velocities of the rotors and the outputs are the Euler angles of the simulator. This will be discussed in Section 2.2.2.

In all of the above DLT, the BIBO stability is guaranteed for discrete-time systems of the form

$$\mathbf{y}_{k+1} = \mathbf{f}\big(\mathbf{y}_k, \ldots, \mathbf{y}_{k-n_y}, \mathbf{u}_k, \ldots, \mathbf{u}_{k-n_u}\big) \tag{2.4}$$

under the following assumptions:

Assumption 2.1. $\frac{\partial \mathbf{f}(\cdot)}{\partial \mathbf{u}_{k-i}} \in C^n \quad \forall \quad i \in 0, \ldots, L-1$

Assumption 2.2. *The input-output model* (2.4) *is generalized Lipschitz*

The above control strategies are implemented in a strictly online fashion. The works in [15, 86] utilize DLT for iterative learning control by using the CFDL and PFDL and updating the proportionality term, PPD, offline between iterations as well as online. The offline updates allow the trend of the system to be updated if the plant has a repetitive task application, and the online adaptation allows the controller to be robust to disturbances and process variations of all sorts. This control strategy has found application in repetitive tasks with only minor process variation and uncertainty. Given that this application is formulated with a heavily offline component and that this work outlines a strictly online control strategy, the details are neglected. It is noted for completeness that the required assumptions for convergence and stability guarantees are the same as in the previously mentioned DLT cases.

Another common MFC strategy was proposed by Fliess and Join in [23]. This control strategy, named intelligent-PID (i-PID), uses a controller with the same structure as the classical PID family controllers to control a system by approximating an online time-varying term referred to as the ultra-local model. The v^{th} derivative of the unknown system output is given as

$$y^{v}(t) = F(t) + \alpha u(t). \tag{2.5}$$

The justification of this equation follows from the implicit function theorem stating that for some continuous input-output function, E, of the derivatives of the inputs and outputs of the system,

$$E(t, y(t), \ldots, y^{\iota}(t), u(t), \ldots, u^{\kappa}(t)) = 0 \tag{2.6}$$

the existence of the non zero partial derivative of (2.6) implies the existence of a function for the v^{th} derivative, that is,

$$\frac{\partial^{v} E}{\partial y^{v}(t)} \neq 0 \implies \exists\ \mathfrak{C}(t, y(t), \ldots, y^{\iota}(t), u(t), \ldots, u^{\kappa}(t)) \quad \forall \quad \{\iota, \kappa \mid \iota \neq v\} \tag{2.7}$$

It then follows that,

$$y^{v}(t) = \mathfrak{C}(t, y(t), \ldots, y^{\iota}(t), u(t), \ldots, u^{\kappa}(t)) = F(t) + \alpha u(t).$$

Fliess and Join argue that the dynamics of the system may be learned and encapsulated in the F term, herein the ultra-local model, from measurements by numerically differentiating the output and rearranging (2.5) for the ultra-local model

$$F(t) = \alpha u(t) - y^{v}(t). \tag{2.8}$$

In (2.5) and (2.8) α is used to scale the input to match the order of magnitude of the

ultra-local model and is a design parameter that need not be known exactly. The authors propose a numerical derivation technique whereby the measured outputs of the system are represented as an infinite power series, by leveraging the algebraic derivative in the Laplace domain, the derivative operators are removed and the series truncated, leaving a finite sum. Since this method removes the derivatives, the numerical approximation is robust to noise and high-frequency fluctuations in the inputs and outputs that typically plague derivative terms of measured values [7, 51].

Remark 2.2. *An interesting observation that is argued empirically in the i-PID papers is that the order of system output need not be greater than $v = 2$, with $v = 1$ satisfying most cases.*

At the time of writing this paper, i-PID control has only been developed for SISO systems with MIMO numerical and experimental testing being done by independently stacking SISO controllers together [34, 73]. This methodology does not allow the controllers to learn coupling between actuators or be extended to cases where the number of inputs and outputs are different. It was further observed numerically that for non-minimum phase systems and cases where $F \to 0$ the control inputs of the i-PID controllers tended to diverge [22, 24]. The use of this algorithm is further complicated by the condition outlined in (2.7). Given that the dynamics of the model are not claimed to be known, there is no way of verifying that the system is not non-minimum phase or that $\frac{\partial^v E}{\partial y^v(t)} \neq 0$.

In [22–24] the authors show a multitude of numerical examples using transfer functions for linear and nonlinear and stable and unstable systems. In some cases, the models are disturbed with non-idealities, such as friction from a Tustin friction model and adding Gaussian noise to the measurements. In all cases the systems shown in the paper respond well, eliminating the steady state error and tracking well in the transient region.

In addition two the PPD and ultra-local model based controllers, other DDC methods are proposed as well by learning the system's behaviour along the trajectories. Consider the controllable and observable discrete-time linear system represented by

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \tag{2.9a}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k. \tag{2.9b}$$

In [76], the authors prove that a finite set of trajectories of a linear time-invariant system spans the entire trajectory space, given that some signal of the system is persistently exciting of order $T_{\text{p.e}} + n$, where $T_{\text{p.e}}$ is the length of the trajectory windows and n is the order of the state space of the system. It is further shown that satisfaction with the

persistent excitation condition in the input signal is guaranteed for sufficiently large choices of trajectory lengths.

In [56] the authors utilize the so-called *Willems et al.'s fundamental lemma* to formulate control strategies based on input-output data. By using *Willems et al.'s fundamental lemma*, Persis and Tesi offer both open-loop and closed-loop representations of the system. By defining the vectors

$$\mathbf{u}_{0,T_{\text{p.e}}} = \begin{bmatrix} u_0, & u_1, & \dots, & u_{T_{\text{p.e}}} \end{bmatrix} \quad \text{and} \quad \mathbf{x}_{i,T_{\text{p.e}}} = \begin{bmatrix} x_i, & x_{i+1}, & \dots, & x_{T_{\text{p.e}}} \end{bmatrix} \quad \forall \;\; i < T_{\text{p.e}}$$

The open-loop and closed-loop data-driven representation of the system described in (2.9) is given as

$$\mathbf{x}_{k+1} = \mathbf{x}_{1,T_{\text{p.d}}} \begin{bmatrix} \mathbf{u}_{1,T_{\text{p.d}}} \\ \hline \mathbf{x}_{0,T_{\text{p.d}}} \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{u}_k \\ \mathbf{x}_k \end{bmatrix} \quad \text{and} \tag{2.10}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_{1,T_{\text{p.d}}} \mathbf{G}_K \quad \text{respectively,} \tag{2.11}$$

where $\mathbf{G}_K$ is dependent on the state-feedback gain vector, $\mathbf{k}$ and the input-output data. This is an important result as (2.10) is a system identification method. In fact Persis and Tesi demonstrates the minimizer of the least-square problem is derived directly from (2.10) and following that, the system and input matrices are recovered from the input-output data. (2.11) offers a way for the gain matrix to be parametrized through measurements.

A stabilizing optimal control formulation is found based on (2.11) using Linear Matrix Inequalities (LMI). By finding a matrix $\mathbf{Q}$ satisfying a given LMI a closed form equation for state-feedback gain vector is given as

$$\mathbf{k} = \mathbf{u}_{0,T_{\text{p.e}}} \mathbf{Q} (\mathbf{x}_{0,T_{\text{p.e}}} \mathbf{Q})^{-1}. \tag{2.12}$$

It turns out that (2.12) is also the closed form solution for a LQR.

The authors motivate the use of LMI in the development of the state-feedback gains by arguing that the stability developed in the above cases holds for sufficiently small perturbations in the input-output data. Essentially the argument boils down to selecting the matrix $\mathbf{Q}$ in such a way that variations of it don't compromise the conditions of the LMI, therefore don't affect the stability of the closed-loop system. Persis and Tesi extend the robustness to noise in small perturbations to nonlinear systems. The argument is developed by saying that if a nonlinear system were approximated with a first-order approximation, the higher order nonlinearity contributions could be considered as disturbances about the linearized model and thus the controller is robust to it based upon the previous argument.

Reinforcement learning together with optimal control is commonly used in robotic

control. Reinforcement learning is employed to learn the value of states or state action pairs, as in the case of Q-learning [25, 32, 62, 67], whereas optimal control employed to select an appropriate policy by minimizing an objective function.

In [71] a synchronous online actor-critic structure is developed for nonlinear time-invariant continuous-time affine systems. Whilst Vamvoudakis and Lewis present a modelled approach with the full dynamics assumed known, the development of the critic and actor neural networks are related to the work presented in this thesis. The cost of the function associated with the optimal control problem is presented as

$$V(\mathbf{x}_0) = \int_0^\infty \big(\mathbf{Q}(\mathbf{x}) + \mathbf{u}^T\mathbf{R}\mathbf{u}\big)d\tau.$$

The optimal policy for this system is derived by minimizing the cost function over $\mathbf{u}$. This optimized function is referred to as the value function as given by

$$V^*(\mathbf{x}_0) = \min_{\mathbf{u}} \int_0^\infty \big(\mathbf{Q}(\mathbf{x}) + \mathbf{u}^T\mathbf{R}\mathbf{u}\big)d\tau.$$

The optimal control action is then given as

$$\mathbf{u}^*(\mathbf{x}) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{x})V^*(\mathbf{x}). \tag{2.13}$$

The partial derivative of the value function can be found as a solution to the Hamilton-Jacobi-Bellman (HJB) equation, but for practical systems, this is typically nonlinear making the solution difficult or impossible. This issue is addressed using the Weierstrass higher-order approximation theorem such that the value function is expressed exactly as a composition of polynomial sums, as below

$$V^*(\mathbf{x}) = \sum_{i=1}^{\infty} \mathbf{c}_i\varphi_i(\mathbf{x}) = \mathbf{c}_1^T\phi_1(\mathbf{x}) + \sum_{i=N+1}^{\infty} c_i\varphi_i(\mathbf{x}) \tag{2.14}$$

where $\mathbf{c}_1 = [c_1, \ldots, c_N]^T$ and $\phi_1(\mathbf{x}) = [\phi_1, \ldots, \phi_N]^T$. Then,

$$\frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{c}_1^T\frac{\partial \phi_1(\mathbf{x})}{\partial \mathbf{x}} + \sum_{i=N+1}^{\infty} c_i\frac{\partial \varphi_i(\mathbf{x})}{\partial \mathbf{x}}$$

Using this, there exists a vector of polynomial coefficients that may be approximated

by neural network weights, $\mathbf{W}_1 \in \mathbb{R}^N$ such that

$$V(\mathbf{x}) = \mathbf{W}_1^T \phi(\mathbf{x}) + \epsilon(\mathbf{x}).$$

Here $\phi_1(x) : \mathbb{R}^n \to \mathbb{R}^N$ is the activation function, $\epsilon(\mathbf{x})$ is the neural network approximation error and N is the number of neurons in the hidden layer. It has been shown that $N \to \infty$, $\epsilon \to 0, \nabla\epsilon \to 0$ [21]. It is also true that for finite values of N the neural network error and its derivatives are bounded [28]. The critic network weights are updated online using normalized gradient descent.

The actor network is designed supposing that the cost function is given in the form of (2.14). Then (2.13) becomes

$$\mathbf{u}^*(\mathbf{x}) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{x})\frac{\partial \phi^*(\mathbf{x})}{\partial \mathbf{x}}\hat{\mathbf{W}}_2 \tag{2.15}$$

where $\hat{\mathbf{W}}_2$ is the approximation of the weights of the critic network weights, $\mathbf{W}_1$. The network error is then given by $\tilde{\mathbf{W}}_2 = \mathbf{W}_1 - \hat{\mathbf{W}}_2$ which is used to tune the actor network online again using gradient descent. In this way, the critic and actor networks are trained online in a synchronous fashion as opposed to other policy gradient methods which at the time were performed by updating the neural networks one at a time, holding the other constant during the updates. It is emphasized again that while this formulation presented an attractive way to synchronously update both neural networks in an online fashion, the control action depends on the partial system dynamics, namely the drift dynamics.

In [1], a SISO real-time integral reinforcement learning structure is presented with stability guarantees for slow time-varying processes. The model-free structure is done using a critic network to approximate the value function, which is commonplace in MFC [2, 39, 75]. This structure leverages a proposed kernel structure

$$S(\mathbf{x}(t), \boldsymbol{\eta}^\pi(t)) = \frac{1}{2}\mathbf{V}^{\pi T}(t)\boldsymbol{\mathcal{H}}\mathbf{V}^\pi(t)$$

with $\mathbf{V}^\pi = [\mathbf{X}(t)^T, \boldsymbol{\eta}(t)^T]^T$ and $\boldsymbol{\mathcal{H}} = \begin{bmatrix} \boldsymbol{\mathcal{H}}_{\mathbf{XX}} & \boldsymbol{\mathcal{H}}_{\mathbf{X}\eta} \\ \boldsymbol{\mathcal{H}}_{\eta\mathbf{X}} & \boldsymbol{\mathcal{H}}_{\eta\eta} \end{bmatrix}$. The kernel structure is shown to be an optimal solution for the HJB equation and represents a Lyapunov function. The real-time component of the algorithm is attributed to the model-free strategy given as

$$\boldsymbol{\eta}^{\pi*}(t) = \boldsymbol{\omega}^{*T}\mathbf{X}(t)$$

for $\boldsymbol{\omega}^{*T} = [\omega_0^*, \omega_2^*, \omega_3^*]^T = -\boldsymbol{\mathcal{H}}_{\eta\eta}^{-1}\boldsymbol{\mathcal{H}}_{\eta\mathbf{X}}$. These equations are used to solve an integral

Bellman optimality equation

$$S^*(\mathbf{X}(t), \boldsymbol{\eta}^{\pi*}(t)) = \int_t^{t+T} = (\mathbf{X}(\zeta), \boldsymbol{\eta}^{\pi*}(\zeta))d\zeta + S^*(\mathbf{X}(t+T), \boldsymbol{\eta}^{\pi*}(t+T))$$

for a time step T and optimal policy $\boldsymbol{\pi}^* = [\omega_1^*, \omega_2^*, \omega_3^*]^T$. Given the impossibility of solving this equation analytically, reinforcement learning techniques are leveraged in the form of actor-critic structure for approximation of the unknown kernel solution.

The kernel function approximator is designed as

$$\hat{S}(\mathbf{X}(t), \hat{\boldsymbol{\eta}}(t)) = \frac{1}{2}\hat{\mathbf{V}}^T(t)\boldsymbol{\Omega}_c\hat{\mathbf{V}}(t)$$

with $\hat{\mathbf{V}} = [\mathbf{X}(t)^T, \hat{\boldsymbol{\eta}}(t)^T]^T$ and $\boldsymbol{\Omega}_c = \begin{bmatrix} \boldsymbol{\Omega}_{c\mathbf{XX}} & \boldsymbol{\Omega}_{c\mathbf{X}\boldsymbol{\eta}} \\ \boldsymbol{\Omega}_{c\boldsymbol{\eta}\mathbf{X}} & \boldsymbol{\Omega}_{c\boldsymbol{\eta}\boldsymbol{\eta}} \end{bmatrix}$. The structure of $\boldsymbol{\Omega}_c$ matches that of $\boldsymbol{\mathcal{H}}$ by design and is updated using gradient descent. The actor network is structured to mimic the optimal correction term

$$\hat{\boldsymbol{\eta}}(t) = \boldsymbol{\Omega}_a\mathbf{X}(t)$$

with $\boldsymbol{\Omega}_a$ representing the actor weights which are again updated using the gradient descent method.

In this work, the so-called integral reinforcement learning, adaptive critics technique is implemented by independently controlling four joints of a manipulator arm. The strategy outperforms the SISO PFDL technique discussed in [78] on the basis of trajectory tracking and reduced oscillation in the response.

In [65] an optimal control strategy is proposed using virtual states and controllers and forgoing the need to use optimal reinforcement learning. The structure of the augmented system, although unknown, is assumed to be able to fit the form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{D}\mathbf{g}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{D} \in \mathbb{R}^{n\times n}$ represent the state system matrix, the augmented input gain matrix and the nonlinearity gain matrix. The vectors $\mathbf{x}, \mathbf{u}, \mathbf{g} \in \mathbb{R}^n$ represent the augmented state vector, the augmented input vector and the nonlinearity vector, where the elements of the two last vectors are either 0 or 1.

The authors propose a representation for the cost function that is the optimal solution to the HJB equation. By choosing the parameters of the cost function to satisfy the

continuous time Riccati equation, an optimal control equation is given, depending on the state input gain matrix, the nonlinearity gain matrix and the nonlinearity vector. The terms in the control vector are approximated and updated online using the input-output data. This approach represents an indirect control parameterization where the system matrices are updated without a priori knowledge of the dynamics.

A limitation to using reinforcement learning for MFC is the unknown future implications of actions required to known the value function and its gradient. In [70, 83], this problem is overcome using a neural network for model identification and parameterizing the critic network and policy (action) on the weights of the network model.

In [70] a MFC structure is presented based on actor-critic theory. The authors design L control structures each comprising three neural networks; one for approximating the model parameters, one for the critic network and one for the actor network. The model network is based on a recurrent neural network and is shown to be uniformly ultimately bounded. The parameters of this network represent the system dynamics and therefore may be used in the critic and actor networks without a priori knowledge. The critic and actor networks are set up similarly to [71] as single-layer partially connected networks and use the same Weierstrass high-order approximation theorem. Both networks are updated using gradient descent methods. A hierarchical shunt inhibitor artificial neural network is used to choose which of the L control structures is used in a given scenario. This network is designed to mimic the orbitofrontal and anterior cingulate cortexes of the brain to make fast and efficient use of previous experience. This classifies the input-output data and designates the control structure associated with each. This approach is reminiscent of the gain-scheduling approach for PID control, with the selection of the system region being intelligent as opposed to predefined.

In [83] Yin, Fu, and Lu design an optimal control strategy leveraging a critic neural network for value function approximation and a deep neural network for model approximation. Both the critic network and the optimal control policy are parameterized through the model estimator. It is shown in [40] that a deep neural network of the form

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{M}\mathbf{x}(t) + \mathbf{N}\boldsymbol{\vartheta}$$

can approximate nonlinear system dynamics. The actual nonlinear system is then represented in the form

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}^*\mathbf{x}(t) + \mathbf{W}_1^*\sigma(\mathbf{V}_1(\mathbf{x}(t))) + \mathbf{W}_2^*\phi(\mathbf{V}_2(\mathbf{x}(t)))\mathbf{u}(t) + \boldsymbol{\zeta}_1$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$ are the system states and inputs, respectively, and $\mathbf{W}_i$ and $\mathbf{V}_i$ are the deep neural network input weights and hidden layer weights, with $\cdot^*$ indicating the true, unknown, values. $\sigma(\cdot)$ and $\phi(\cdot)$ are sigmoidal activation functions and $\boldsymbol{\zeta}_1$ is the disturbance term. The approximation of the system dynamics is then

$$\frac{d\hat{\mathbf{x}}(t)}{dt} = \mathbf{A}\hat{\mathbf{x}}(t) + \mathbf{W}_1\sigma(\mathbf{V}_1(\hat{\mathbf{x}}(t))) + \mathbf{W}_2\phi(\mathbf{V}_2(\hat{\mathbf{x}}(t)))\mathbf{u}(t).$$

The approximated terms are then updated using the following update laws:

$$\dot{\mathbf{A}} = -k_1\tilde{\mathbf{x}}(t)\hat{\mathbf{x}}^T(t) \tag{2.16a}$$

$$\dot{\mathbf{W}}_1 = -k_2\tilde{\mathbf{x}}(t)\sigma(\hat{\mathbf{x}}(t)) \tag{2.16b}$$

$$\dot{\mathbf{W}}_2 = -k_3\tilde{\mathbf{x}}(t)\mathbf{u}^T(t) \tag{2.16c}$$

for $\tilde{\mathbf{x}} = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$. Under the conditions that $\mathbf{W}_i$ and $\boldsymbol{\zeta}_1$ are bounded, the update laws in (2.16) lead to exact identification in the case that $\boldsymbol{\zeta}_1 = 0$ and bounded estimation error for $\boldsymbol{\zeta}_1 \leq \bar{\boldsymbol{\zeta}}_1$.

The control input is decomposed as $\mathbf{u}(t) = \mathbf{u}_r(t) + \mathbf{u}_e(t)$, where $\mathbf{u}_r(t)$ is responsible for eliminating the steady-state tracking error and $\mathbf{u}_e(t)$ is responsible for minimizing the performance index optimally. $\mathbf{u}_r(t)$ is from the error dynamics as

$$\mathbf{u}_r(t) = \mathbf{W}_2^+\left[\frac{d\mathbf{x}_d(t)}{dt} - \mathbf{A}\mathbf{x} - \mathbf{W}_1\sigma(\mathbf{x}) - \mathbf{K}\mathbf{e}(t)\right]$$

with $\mathbf{W}_2^+$ being the generalized inverse or $\mathbf{W}_2$, $\mathbf{x}(t)_d$ being the desired trajectory and $\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{x}_d(t)$ being the tracking error.

To find $\mathbf{u}_e(t)$ the usual approach is taken defining the cost function over an infinite horizon and minimizing the HJB to find the optimal control parameter. Note that the definition of the cost function is specific to the transient component of the control action, i.e.

$$V(e(t)) = \int_t^\infty \mathbf{e}(t)^T\mathbf{Q}\mathbf{e}(t) + \mathbf{u}_e(t)^T\mathbf{R}\mathbf{u}_e(t).$$

Then in the usual way

$$\mathbf{u}_e(t) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{W}_2^T\frac{\partial V(\mathbf{e}(t))}{\partial \mathbf{e}(t)}. \tag{2.17}$$

Again, the unavailability of $\frac{\partial V(\mathbf{e}(t))}{\partial \mathbf{e}(t)}$ inhibits finding this parameter so a single layer neural

network is used with

$$V(\mathbf{e}(t)) = \mathbf{W}_3^T \phi(\mathbf{e}(t))$$

where $\mathbf{W}_3$ are the input weights. Then (2.17) becomes

$$\mathbf{u}_e(t) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{W}_2^T \nabla \phi^T(\mathbf{e})\mathbf{W}_3.$$

The algorithms introduced above are taken predominantly from theoretical work that are validated usually with mathematical justification and highly simplified numerical simulation. To better understand the machine learning control algorithms that have seen practical use in aerial vehicle control, the current literature is presented in Section 2.2.2.

2.2.2 Model-Free Control of Aerial Vehicles

This section continues the discussion of machine learning control, specifically discussing algorithms that have been implemented on aerial vehicles.

To demonstrate the validity of the of the FFDL technique, Xiong and Hou apply it to a UAVs simulation tool, the Quad-Rotor Aircraft Simulator. The control outputs are the average voltages across the actuators for each of the four propellers and the outputs are the simulated Euler angles of the simulated aircraft, namely the roll, pitch and yaw. The results of the FFDL controller are compared to the results of PID controller for step input reference trajectories of $[10, 10, 10]$ [°][2] The authors quantify the performance of each controller by using the Integral Absolute Error (IAE) and Integral Time Absolute Error (ITAE). The pitch is controlled in such a way that the error for both types of controllers is comparable with the FFDL marginally outperforming the PID in both metrics. The FFDL controller shows approximately 3.5 and 1.5 fold improvements concerning the PID controller in the roll and pitch, respectively. Xiong and Hou attribute this to the algorithm's ability to capture the coupling characteristics.

There are many examples of the i-PID controllers being used in UAVs applications [9–11, 14, 16, 72, 84, 85]. They are listed here for completeness, but only three interesting cases are discussed. In [84] the authors modify (2.5) to include a known and unknown component of the control law. Two controllers are then proposed based on the theories of nonlinear integral backstepping and LQR. This novel controller uses the intuition from the approximately modelled controllers, whilst relying on the adaptability of the i-PID

[2]The authors claim that the reference trajectories are in radians, but that doesn't seem to make sense given that 10 [rad] $\approx$ 573 [°].

controller to compensate for the unknown or uncertain components of the dynamics. The nonlinear integral backstepping and LQR controllers are run as baselines following a three-dimensional square pathway counter-clockwise as seen from above. Halfway through the third side of the square, a 15% loss of power is introduced to the front rotor. Both controllers can follow the path with minimal error, until the actuation failure where they both diverge. With the addition of the i-PID controller, the quadrotor can follow the trajectory with reduced error, and efficiently return to the path when the actuator fails. The Root-Mean-Square (RMS) error of the in-plane trajectory follower before the divergence of the non-augmented controllers is reduced by approximately twofold with the addition of the i-PID controller. It is further observed that the RMS error of the augmented controllers is not drastically affected by the introduction of the actuator failure. It is noted that these results are from a physically implemented system.

In their work on airship control using model-free reinforcement learning approaches in [50], Nie, Zheng, and Zhu develop a model-free online Q-learning three-dimensional path following flight controller. The controller is decomposed into an altitude controller and a planar motion controller using a decoupling assumption. The high-dimensionality of the state space is addressed by using parameter reduction and coordinate transformation.

The authors assume that the longitudinal component of the velocity, u is significantly higher than the lateral and vertical components and therefore assume that the longitudinal component of the 3D velocity vector is approximately equal to the total velocity, $u \approx \mathbf{V}$. It is also required for this assumption that the roll is very small, which is often the case given the high restoring torques of dirigibles [50]. With these assumptions, the 12 parameters representing the pose and velocity of the airship in the base reference frame and Earth reference frame is reduced to three parameters for the altitude state space,

$$\mathbf{S}_{\text{alt}} = \left[h, \dot{h}, \mathbf{V}\right]^T,$$

for the height, h, vertical speed $\dot{h}$ and velocity $\mathbf{V}$.

Augmenting from the state space reduction for the altitude controller, the authors encapsulate the location of the airship with respect to the location of the next waypoint using error based polar coordinate. Define,

$$l_r = \sqrt{(x_{\text{airship}} - x_{\text{target}})^2 + (y_{\text{airship}} - y_{\text{target}})^2}$$

for the planar planar error from the airship position (x_{airship}, y_{airship}) to the waypoint

$(x_{\text{target}},\ y_{\text{target}})$. Similarly, the relative required yaw is defined with respect to the yaw in the Earth reference frame, ψ_a, as

$$\psi_r = \psi_a - \arctan\left(\frac{y_{\text{airship}} - y_{\text{target}}}{x_{\text{airship}} - x_{\text{target}}}\right).$$

The state space for the planar controller is then

$$\mathbf{S}_{\text{plane}} = [l_r, \psi_r, \mathbf{V}]^T.$$

The value of each action for the given state vector is approximated using a cerebellar model articulation controller neural network. This network groups states that are similar while segregating states that are dissimilar. This is done to create a more efficient value function computation while generalizing for similar experiences.

The Q-function is defined as,

$$Q(\mathbf{s}, \mathbf{a}) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t r_t \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right\}.$$

In the Q-function γ is the discount factor and the reward function, r_t is meant to advise the policy. Although no specific reward structure is discussed in the paper, the authors reward positively for finding the waypoint and punish for every second the waypoint is not found. Given the unknown nature of the Q-function, the authors estimate it using,

$$\tilde{Q}(\mathbf{s}, \mathbf{a}) = \mathbf{W}_a^T \mathbf{F}(\mathbf{s}),$$

for the binary matrix $\mathbf{F}(\mathbf{s})$. The values of the weights, $\mathbf{W}_a$ are updated using gradient descent based on the estimated value at the next time step and the calculated value. The action transition probability equation for the next state is approximated by the Boltzmann distribution on the Q-function that is structured with an exploration coefficient, T_{temp}. This time-varying coefficient is reduced over time gradually shifting the strategy from favouring exploration, to favouring exploitation as more experience is required,

$$\mathbf{p}(\mathbf{a}_t \mid \mathbf{s}_t) = \frac{e^{\frac{Q(\mathbf{s}_t, \mathbf{a}_t)}{T_{\text{temp}}}}}{\sum_{\mathbf{a} \in \mathbf{A}} e^{\frac{Q(\mathbf{s}_t, \mathbf{a}_t)}{T_{\text{temp}}}}}$$

The algorithm is tested using MATLAB simulation on a numerical model of a LS-S1200. The weights of the Q-function for the altitude controller are trained for 6000 [s] with randomly initialized waypoints close to the current location of the airship to stabilize the success rate of the airship, which converged around 1000 [s]. The altitude controller is used to control a staircase altitude task by controlling the elevator control surface of the airship. Two simulations are conducted; the first with the rudder angle at 0 [°] and thrust vector constant and the second with the rudder angle and thrust vector randomly changing every 15 [s]. For target altitudes given by,

$$h_{\text{target}} = \begin{cases} 100 \text{ [m]}, & \text{if} \quad t \leq 250 \text{ [s]} \\ 110 \text{ [m]}, & \text{if} \quad 250 \text{ [s]} < t \leq 450 \text{ [s]} \\ 120 \text{ [m]}, & \text{if} \quad 450 \text{ [s]} < t \leq 650 \text{ [s]} \\ 110 \text{ [m]}, & \text{if} \quad 650 \text{ [s]} < t \leq 850 \text{ [s]} \\ 100 \text{ [m]}, & \text{if} \quad 850 \text{ [s]} < t \end{cases}$$

as trial-and-error tuned PID controller outperformed the reinforcement learning controller, with both controllers performing well. In the experiment with random actuator disturbance, the reinforcement learning controller outperforms the PID controller, showing a reduced error amplitude. In both cases, the reinforcement learning algorithm shows low amplitude, high-frequency oscillations.

To test the planar controller, the height is held constant and a path with sharp and high radius curvatures is chosen in a plane. The controllers influence the rudder deflection angle and the thrust vector. Following roughly 11000 [h] of training the weights, the reinforcement learning controller outperforms a trial-and-error tuned PID controller in the average and maximum distances to the next target and average and maximum distance to the desired path by a factor up to approximately 4.2. From the actual path graph, it is observed that although the proposed controller outperforms the PID controller, it tends to have a less smooth trajectory and even loops back around at one point.

The two controllers are stacked to follow a 3D spiral trajectory, although this experiment is not discussed in great detail and is compared to another controller. Although the proposed controller is successful in numerical simulation, there is no physical testing or high realism experimentation. In the process of reducing the state space the assumption that $\dot{h} \approx 0$ is made to assume that the longitudinal velocity is equal to the 3D velocity vector. This assumption is contradicted by setting $\dot{h} = \sin\theta \mathbf{V}$ to find the vertical velocity. In addition, in realistic environments crosswinds can induce high degrees of side slip making the assumption that the lateral velocity is negligible invalid. Finally, the high frequency

oscillations in the altitude controller output seem to negate some of the stability and low noise benefits that are inherent to the vehicle type.

In [85] Younes, Rabhi, and Al-Wedyan extends the work done in [22–24] by developing a MIMO formulation for the i-PID controller. The controller is implemented via a twin rotor MIMO system using the MATLAB/Simulink interface. The authors chose the dc motor voltages as the controller's input and the pitch and yaw as the outputs. The pitch is asked to follow an abstract trajectory, while the reference signal for the yaw is a constant 0.2 [rad]. In the first test, a 0.2 [kg] mass is dropped on the boom connecting the rotors and in the second a multiplicative actuator failure is simulated. In both cases, the controllers track the desired trajectories well, with the system correcting the external disturbances and actuation failures. The PID strategy showed significantly higher oscillation than the i-PID controllers did and the actuator failures caused a larger and longer-lasting deviation from the desired trajectory. It is observed that this MIMO structure of the i-PID strategy suffers the same limitation as the independent SISO formulation, in that the number of inputs and outputs must be the same. Perhaps a more limiting observation is the number of matrix inversions that exist in the control law. This, along with the necessary numerical derivation following every measurement might lead to computational burdens that limit this algorithm from use in real-time or high-dimensional tasks.

In [16] the authors control the rates of the Euler angles based on remote controller input from a user. To do this multiple i-PID controllers run independently, and the control actions are chosen carefully to respect the condition that the number of inputs and outputs must match. To do this, the authors define control actions based on high-level intuition. For example, to introduce roll rates, one control action controls the difference in angular speeds of the propellers on the right and left sides of the vehicle. This requires the designers to have a good understanding of the effect of each of the actuators, although a specific model is not required. To validate the formulation they implement the controller on a DJI drone, fitted with their autopilot microcontroller board and software. The robustness is then confirmed by cutting off half of each propeller and then further confirmed by putting the autopilot on a different drone altogether. While the paper lacked proper quantitative results as the vehicle was being flown manually, the multi-rotor vehicle appeared stable and responsive in all tests. The interested reader can view a video of their flight tests here.

Reinforcement learning is ubiquitous with modern MFC techniques with a great deal of focus being on how to approximate the value function and, in the case of some policy gradient techniques, the action function. Along with neural networks and kernel functions, Gaussian processes are used in the literature [33, 63, 64]. In [33] a hybrid of modelled and model-free approaches are used where the authors start by designing a nonlinear model of a dirigible and use the Gaussian processes to approximate and correct the model error.

The objective of the controller is to control the yaw, ψ and yaw rate, $\dot{\psi}$, driving them to their targets, $\psi^* = 0$ [°], $\dot{\psi}^* = 0$ [°/s]. The initial conditions are $\psi = 90$ [°], $\dot{\psi} = 0$ [°/s]. Two experiments are performed comparing the performance of the Runge-Kutta numerical approximation of the system dynamics based on the nonlinear model and the performance, again using the Runge-Kutta approximation, of the Gaussian process enhanced model. The enhanced controller showed a 4.3 and 4.8 fold decrease in RMS error when using the Gaussian process.

The works in [63, 64] use reinforcement learning techniques enhanced with Gaussian processes as well, but the function approximation in these works is the value functions rather than the system dynamics. The objective of both papers is to control the altitude of a physical dirigible using Kalman-filtered height and velocity signals derived from readings from an ultrasonic sensor. In [64] the reinforcement problem is formulated over the continuous action and state spaces using episodic Monte-Carlo methods with decaying ϵ-greedy policy selection. The objective is to develop a policy $\pi : S \times A \rightarrow A \mid A \in [a_{\min}, a_{\max}]$ to drive the height to a desired setpoint, z^d by rewarding the action a_t in the following manner

$$R^{\pi} = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{2.18}$$

where the reward $r_{t+i} = - \mid z - z^d \mid$ is the negative of the error in the altitude. The Gaussian process is utilized to approximate the state-action function, $Q(s_t, a_t) : S \times A \rightarrow \mathbb{R}$ such that the approximation is given by, $q_i = Q(s_i, a_i) + \zeta_i$, where ζ_i is the independent and identically, normally distributed noise. To perform the predictions online without a priori knowledge or modeling, the flight is episodic and performed in two stages. For $t < \bar{T}$ the airship is in the exploration stage with the action to be selected being pseudo-random[3], otherwise, the airship is in the exploitation stage, where the probability of exploration decreases exponentially, that is, $\epsilon \rightarrow 0$ as $t \rightarrow \infty \quad \forall \quad t \geq \bar{T}$. The episodes are selected as $e_t = \{(s_t, a_t), \ldots, (s_{t+p}, a_{t+p})\}$, where p is chosen such that $\gamma^{p+1} < \theta_{\text{threshold}}$, for some user-defined threshold $\theta_{\text{threshold}}$, in this case $\theta_{\text{threshold}} = 0.1$.

The use of Gaussian processes for convergence is validated in simulation by numerical experimentation, where blimp dynamics are controlled with strict Monte-Carlo methods, approximating the state-action function strictly from gathered data, and Gaussian process-enhanced Monte-Carlo methods. The results show that the convergence with enhancement is orders of magnitude faster than without enhancement and the variance among the policy decisions are reduced as well. The tracking performance on the physical dirigible is

[3]The authors manually override any action that could cause damage to the airship in the empirical tests by keeping the airship away from the ground and ceiling.

also shown to be successful with the airship successfully reducing the steady-state error in altitude with only 150 [s] of exploration and 150 [s] of exploitation. With the best policy extracted from this experiment, the policy is compared to a manually tuned PD^2-T$_2$ controller. Both controllers are able to track the desired altitude of $z^d = 2$ [m] with the online controller reaching the target approximately 15 [s] sooner at the cost of a 20% overshoot. It is mentioned in this paper as well as [43] that this technique is acceptable for low dimensional cases, but struggles as the number of states increases due to the sample-inefficiencies.

A major drawback of simulating control algorithms is the dissociation of numerical simulations to the physical systems, the so-called real-to-sim error [6, 42, 54, 66]. The results from this work show that the controller used in the simulation can be applied to the physical prototype without the need for changing the controller weights, which implies good mapping from simulation to reality. This is the inability of the simulation to encapsulate realistic dynamics, coupling and behaviour that should be expected to observe when a plant is operated in real life. In the case of airships, common assumptions are that the airship represents a rigid body and that the wind disturbance is either constant or zero. These assumptions are almost required given the complexity of modelling non-rigidity and turbulent wind. In [58] a Robot Operating System (ROS) compatible, open-source airship model is designed allowing by attaching rigid shapes and modifying spring coefficients between the substructures to simulate non-rigidity. While this is not a direct mapping to reality, it is a significant improvement from neglecting these terms. In addition, a turbulent Dryden wind model [55, 82] is used to simulate external wind disturbances which is an important consideration given the high degree of aerodynamic susceptibility of LTAVs. In their work Price et al. leverage the LibrePilot autopilot software [17] to create virtual axes on some of the actuators that allow them to control the pitch, yaw and thrust. To advise the low-level vehicle agnostic control signals, a high-level path planning algorithm is designed to determine the setpoints for the low-level control structures. The path-planner algorithm takes the desired 3-dimensional velocity vector, the 3-dimensional airspeed vector, the actual velocity vector and the Euler angles as inputs and returns the desired setpoints for the yaw rate, pitch, thrust and thrust angle vector. These are calculated through a series of Proportional (P) and PI controllers. The gains for the controllers are tuned empirically through trial and error. Although this is not a MFC approach it is mentioned here as it provides a platform to validate control approaches without risking damage to a physical prototype. that is leveraged for use in other MFC approaches [43, 44].

In [44] a reinforcement learning approach is employed offline using a quantile regression deep Q-learning network for the value function approximation [45]. This work uses the simulation environment in [58] and formulates the problem as a continuous state, discrete action-space problem to reduce the computational complexity of the continuous action-

spaces and improve the convergence times. In the paper, the state space is given as

$$\mathbf{x} = [\mathbf{x}^{\text{act}}, l_r, \psi_r, z_r, \mid V \mid, w]^T \tag{2.19}$$

where $\mathbf{x}^{\text{act}}$ are the current normalized states of the actuators, l_r is the 2-dimensional Euclidean distance to the target in the longitudinal/lateral plane of the blimp frame, ψ_r is the angle from the lateral axis of the blimp to the target, z_r is the vertical distance to the target, $\mid V \mid$ is the magnitude of the 3-dimensional velocity or the airship and $w = V \sin\theta$ is the vertical speed. The reward structure of the agent is

$$r_k = \mathbf{w}^T \mathbf{r} \tag{2.20}$$

with $\mathbf{w} = [1.0, 0.95, 0.05]^T$ being reward weighting terms and $\mathbf{r} = [r_k^{\text{success}}, r_k^{\text{track}}, r_k^{\text{act}}]^T$. The rewards are chosen such that $r_k^{\text{success}} = 1$ if the Euclidean distance to the target is below a set threshold and $r_k^{\text{success}} = 0$, r_k^{track} is determined based on the distance r_k^{act} punishes high motor demands. The authors inject intuition into the MFC scheme in the selection of the actions for the discrete action-space choosing the changes in the actuator signal based on the desired change in the blimp. The policy is chosen based on the output of the quantile regression deep Q-learning network with the inputs being the states from (2.19) and the output is an array representing the discretized action-space with the best action being selected based on a greedy policy.

Following seven days of training, the reinforcement learning agent is tested by completing three laps of a square trajectory. The agent shows good results in the navigation component without wind disturbance not deviating more than 30 [m] from the desired trajectory while carrying approximately 10 [m] altitude error for the majority of the flight. The PID controller proposed in [58] outperforms the proposed controller in error reduction in both the trajectory following and altitude tracking. The second test is a hover test with the point to hover around being at the origin with a target altitude of 50 [m]. The reinforcement learning agent again misses the altitude target by approximately 5 [m] while the PID controller is less than 1 [m] from the desired height. The distance to the hover point reinforcement learning agent reduces the error considerably compared to the PID controller and keeps the point of interest at the center of the trajectory. When wind is added to the trajectory tracking simulations, the response of the reinforcement learning agent is not significantly altered for a wind speed of 2 [m/s], however when the speed is increased to 4 [m/s] the agent is no longer able to track the desired trajectory.

Building on the approach in [44], the work in [43] proposes a mixed classical and MFC scheme that leverages the stability benefits of the classical approach with the improved response of the reinforcement learning response. The paper outlines a modified deep resid-

ual reinforcement learning [68] approach similar to that in [44] that allows for continuous action-space and incorporates the time-delay effects of the of the airship into a long short-term memory layer, [27] at the output of the neural network. The continuous action-space is realized with low chattering by scaling down the outputs from the neural network and adding them to the current states, as opposed to having pre-set correction terms. The long short-term memory is implemented as a final hidden layer before the output. The mix of classical control and MFC is governed by:

$$f^{\text{mix}}_{\text{absolute}}(a^{\text{pid}}_t, a^{\text{RL}}_t) = (1-\beta)a^{\text{pid}}_t + \beta a^{\text{RL}}_t \tag{2.21a}$$

$$f^{\text{mix}}_{\text{relative}}(a^{\text{pid}}_t, a^{\text{RL}}_t) = a^{\text{pid}}_t(1+\beta a^{\text{RL}}_t) \tag{2.21b}$$

$$f^{\text{mix}}_{\text{hybrid}}(a^{\text{pid}}_t, a^{\text{RL}}_t) = (1-\alpha)f^{\text{mix}}_{\text{absolute}}(a^{\text{pid}}_t, a^{\text{RL}}_t) + \alpha f^{\text{mix}}_{\text{relative}}(a^{\text{pid}}_t, a^{\text{RL}}_t) \tag{2.21c}$$

for some $(\alpha, \beta) \in [0, 1]$ which are manually tuned parameters. The absolute scheme in (2.21a) favours the reinforcement learning agent, the relative scheme, (2.21b), favours the PID controller and the hybrid and the hybrid shceme, (2.21c), shifts the authority from one strategy to the other based on the selection of the hyperparameters.

In the testing, the trajectories to follow are a square path with four waypoints each 80 m apart and a so-called coil trajectory with waypoints at 42.4 [m] with an angular deviation of 45 [°]. This control strategy is trained offline, with a total training time of 28 days in the ROS/Gazebo framework presented in [58]. To compare the differences, the maximum total rewards are compared for the three mixing strategies for a poor-PID and good-PID controller. The results showed that in each case the absolute strategy had the lowest minimum reward value, showing up for a small number of iterations, and the highest maximum value, for a large number of iterations. This is intuitive as the absolute favours the reinforcement learning strategy whose task is to maximize the rewards function. With too few iterations, the policy doesn't have enough samples to effectively exploit the best policy. The long short-term memory addition shows an improvement in the performance by 14% when used with a good-PID controller, compared to 13% when used with a poor-PID controller. It is also noted that the performance boost of the reinforcement learning agent is more so on a well-tuned PID controller, meaning that in order to leverage this control strategy, a well-working controller is a prerequisite. In the trajectory tracking tests on the trained agent, the reinforcement learning controller is able to track the square trajectory well, reducing the overshoot from approximately 25 [m] for the baseline-PID test, to approximately 10 [m]. Similar results are seen in the coil test, although it is observed that the reinforcement learning agent is never able to eliminate the tracking error. The authors attribute this to the reduced length between the waypoints, leaving the agent less distance to learn the desired policy for that segment. The control strategy is

implemented in the physical dirigible and used in an outdoor setting with an average wind speed of approximately 6 [m/s], with peaks measured up to 8 [m/s]. Despite the training only including 3 [m/s] wind, the agent is able to adapt well and appropriately track four waypoints. There is an overshoot in the order of 40 [m] on multiple occasions during that flight test.

2.3 Summary

In summary, both classical and machine learning control strategies have been developed along the journey to airship autonomy. Among the classical strategies controllers from the PID family seem to be most prominent in conjunction with simplifying assumptions and model linearization techniques. In most cases, such heuristic controllers are not implemented in practice and simply validated using numerical models, which are known to be uncertain. In the few cases that these controllers are applied to a physical vehicle, the controller gains are tuned via trial-and-error, re-emphasizing the uncertainties and complexities surrounding the modeling task. Likewise, nonlinear controllers, such as Lypunov and SMC methods are applied to numerical simulations, however their application to real dirigibles is less common.

General machine learning structures are introduced in this chapter, analyzing the theoretical state-of-the-art for MFC. It is observed that the MIMO MFC algorithms are lacking, and the ones that do exist have dimensional restrictions that limit their usage. In fact, it is common to see MIMO MFC achieved by stacking independent SISO controllers. Applications of these controllers to UAVs are outlined in Section 2.2.2, where a range of online and offline techniques applied to mutli-rotor vehicles and LTAVs, with dirigibles being less common among the literature, with no applications of online adaptive MIMO DDC being present at the time of writing.

Chapter 3

Mathematical Preliminaries

This chapter outlines mathematical preliminaries for reinforcement learning and optimal control theory. General controller design methodologies for determining the state feedback control using optimal control theory and reinforcement learning are derived and specific attention is drawn to limitations preventing such approaches from being applied as an online Model-Free Control (MFC) strategy. The problem statement is formulated, and the objective function used for online controller evaluation and adaptation is introduced.

Notations 3.1. *We shall denote vectors and matrices with bold lowercase and uppercase letters, respectively. Let* $\mathbb{N} = \{1, 2, \ldots\}$ *with* $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$*. The set of nonnegative real numbers is denoted by* $\mathbb{R}_+$*, where* $\mathbb{R}$ *represents the set of all real numbers. Nonbold letters denote scalar quantities.*

3.1 System Definition

In this section, a general description of an unknown system to be controlled is presented and the error dynamics are introduced. This mathematical formulation represents the framework upon which the problem formulation and controller developments are based on.

Without loss of generality, we consider a nonlinear n-dimensional, partially observable, Multi-Input Multi-Output (MIMO) state-space representation of an airship. It is assumed that the unknown model can be approximated using a discrete-time state-space model

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\zeta}_k) \tag{3.1a}$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}_k) \tag{3.1b}$$

where $k \in \mathbb{N}_0$ denotes the discrete-time index at time $t = kT$ for some small sampling period $T \in \mathbb{R}_+$. The process and measurement noise are denoted with $\boldsymbol{\zeta}_k$ and $\boldsymbol{\xi}_k$ with appropriate dimensions, respectively.

It is emphasized that the state transition function $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, which contains the system input and drift dynamics is unknown. The output of the airship, $\mathbf{y}_k \in \mathbb{R}^n$, is mapped by some output function that is also assumed unknown, $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ for some $p \leq n$. In this work, the states of the airship at time instant $k \in \mathbb{N}_0$ are defined as

$$\mathbf{x}_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,n}]^T \in \mathbb{R}^n,$$

where $x_{k,i}$ is the ith state variable and the control action vector is represented as

$$\mathbf{u}_k = [u_{k,1}, u_{k,2}, \ldots u_{k,m}]^T \in \mathbb{R}^m. \tag{3.2}$$

The input vector in (3.2) can be decomposed into the previous state of the actuators, $\mathbf{u}_{k-1}$, and a regulation term, $\tilde{\mathbf{u}}_k$ as in [1]. Then

$$\mathbf{u}_k = \mathbf{u}_{k-1} + \tilde{\mathbf{u}}_k.$$

Remark 3.1. *It is important to note that, unlike a common assumption in the literature [4, 13], the state variables in $\mathbf{x}_k$ are not assumed to be independent or decoupled.*

To advise the performances of the proposed controllers online, a cost function is used over an infinite time horizon, that allows the designer to weigh the controller's ability to eliminate error while minimizing its actuation energy. To this end, the error dynamics are introduced such that

$$\mathbf{e}_{k+1} = \mathbf{h}(\mathbf{e}_k, \mathbf{u}_k), \quad \text{where} \tag{3.3}$$

$$\mathbf{e}_k = [e_{k,1}, e_{k,2}, \ldots e_{k,n}]^T \in \mathbb{R}^n \ \forall k \in \mathbb{N}_0. \tag{3.4}$$

Similar to (3.1a), the function $\mathbf{h} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, representing the error dynamics of the outputs is not known a priori. Here $e_{k,i}$ represents the normalized and saturated tracking error at time instant $k \in \mathbb{N}_0$ of the i^{th} state variable. It is important that the error signals, $e_{k,i}$, are conditioned to be in the range of $[-1, 1]$ to help avoid divergence and avoid states with larger error ranges from dominating the system response [18, 59, 69]. The normalized, unsaturated error is defined as

$$\tilde{e}_{k,i} = \frac{x_{k,i} - x_i^*}{x_i^{\max}}, \quad \forall i \in \{1, 2, \ldots, n\} \tag{3.5}$$

where $x_i^{\max} > 0$ is a designer-specified maximum value for the i^{th} output and x_i^* is the target value of the state. The normalized saturated error term is then

$$e_{k,i} = \text{sign}(\tilde{e}_{k,i}) \min(1, |\tilde{e}_{k,i}|), \tag{3.6}$$

where the function sign$(\cdot)$ returns $-1(+1)$ for negative (positive) argument $(\cdot)$, $\forall i \in \{1, 2, \ldots, n\}$. The optimal control problem is formulated as a discrete-time, deterministic regulation problem such that, given the unknown dynamics (3.1) of the system model, the controller determines the required correction term, $\tilde{\mathbf{u}}_k$ such that the states of the system, $\mathbf{x}_k$ track the desired state, $\mathbf{x}_k^*$ of the airship. That is

$$\mathbf{x}_k \to \mathbf{x}_k^*, \text{ as } k \to \infty \quad \text{or equivalently} \quad \mathbf{e}_k \to \mathbf{0} \in \mathbb{R}^n, \text{ as } k \to \infty.$$

With these definitions in place, a general control strategy is considered in Section 3.2 that leverages optimal control theory to derive the desired control action based on a proposed online objective function. Concepts from reinforcement learning are used in deriving the Bellman equation which is used to parameterize the desired control action.

3.2 Optimal Control Formulation

In this section, the optimal control problem is formulated and an equation for the desired control action is developed based on optimal control theory. The controller derivation follows from the works outlined in [1, 2, 38, 39, 46, 71]. The section ends with a discussion concerning the challenges of implementing the controller, as derived, in a MFC setting.

To evaluate the performance of the proposed controllers, a scalar quadratic objective function, the cost function, is defined over an infinite time horizon as

$$J(\tilde{\mathbf{u}}) = \sum_{k=0}^{\infty} r(\mathbf{e}_k, \tilde{\mathbf{u}}_k) \quad \text{where} \quad r(\mathbf{e}_k, \tilde{\mathbf{u}}_k) = \frac{1}{2}\mathbf{e}_k^T \mathbf{Q} \mathbf{e}_k + \tilde{\mathbf{u}}_k^T \mathbf{R} \tilde{\mathbf{u}}_k \tag{3.7}$$

denotes the running cost of the proposed airship control scheme. The matrices $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\mathbf{R} \in \mathbb{R}^{m \times m}$ are symmetric positive-definite weighting matrices that are left as design parameters. Motivated by the linear-quadratic regulator Linear Quadratic Regulator (LQR) strategy in the optimal control literature it is known that the $\mathbf{Q}$ matrix is responsible for weighing the controller's ability to eliminate error, while the $\mathbf{R}$ matrix is responsible for weighing the controller's ability to conserve energy [38].

Towards the development of the controller, a discrete-time value function is defined to be quadratic in the error dynamics, $\mathbf{e}_k$, and the control action, $\mathbf{u}_k$. The value function is defined as

$$V(\mathbf{e}_k, \tilde{\mathbf{u}}_k) = \sum_{\iota=k}^{\infty} r(\mathbf{e}_\iota, \tilde{\mathbf{u}}_\iota) = \frac{1}{2}\left[\mathbf{e}_k^T \mathbf{Q} \mathbf{e}_k + \tilde{\mathbf{u}}_k^T \mathbf{R} \tilde{\mathbf{u}}_k\right] + V(\mathbf{e}_{k+1}, \tilde{\mathbf{u}}_{k+1}) \tag{3.8}$$

Remark 3.2. *The choice of (3.8) being quadratic is made to simplify the analysis, but this could be extended to any order so long as care is taken to make the appropriate changes to the objective function, (3.7), and the basis function, which is introduced in Chapter 4.*

Applying Bellman's principle of optimality, the optimal value function is [38, 62]:

$$V^*(\mathbf{e}_k, \tilde{\mathbf{u}}_k^*) = \min_{\tilde{\mathbf{u}}_k} \left\{ \frac{1}{2}\left[\mathbf{e}_k^T \mathbf{Q} \mathbf{e}_k + \tilde{\mathbf{u}}_k^T \mathbf{R} \tilde{\mathbf{u}}_k\right] + V^*(\mathbf{e}_{k+1}, \tilde{\mathbf{u}}_{k+1}^*) \right\} \tag{3.9}$$

where $\mathbf{u}_k^*$ denotes the optimal control action at time instant $k \in \mathbb{N}_0$. Performing the minimization in (3.9) yields

$$\frac{\partial V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)}{\partial \tilde{\mathbf{u}}_k} = \mathbf{R}\tilde{\mathbf{u}}_k + \left(\frac{\partial \mathbf{e}_{k+1}}{\partial \tilde{\mathbf{u}}_k}\right)^T \nabla V^* = \mathbf{0}, \tag{3.10}$$

where $\nabla V^* = \frac{\partial V^*\left(\mathbf{e}_{k+1}, \tilde{\mathbf{u}}_{k+1}^*\right)}{\partial \mathbf{e}_{k+1}}$ is a column vector of dimension n. By algebraic manipulation of (3.10), the equation for the optimal control action for this development is found to be

$$\tilde{\mathbf{u}}_k^* = -\mathbf{R}^{-1}\left(\frac{\partial \mathbf{e}_{k+1}}{\partial \tilde{\mathbf{u}}_k}\right)^T \nabla V^*. \tag{3.11}$$

The representation given in (3.11) presents an issue for both online and MFC techniques as it assumes a priori knowledge of the error dynamical function, $\mathbf{h}(\mathbf{e}_k, \mathbf{u}_k)$, which is unknown. Even in model-based control strategies, knowledge of the required functions requires exhaustive experience and/or accurate system modelling [7, 51, 62] and often depend on the solutions of nonlinear partial differential equations [43].

Definition 3.1. *A control action $\mathbf{u}_k$, $\forall\ k \in \mathbb{N}_0$, is admissible if:*

1. *The system is stabilized under the control of $\mathbf{u}_k$, and*
2. *The system has finite cost over its trajectory*

By requiring that $\mathbf{Q}$ and $\mathbf{R}$ are symmetric positive-definite, it is expected that the value function is monotonically decreasing, thus for an admissible initial error and control action, the overall value remains finite, so long as $\mathbf{Q}$ and $\mathbf{R}$ remain symmetric positive definite.

Assumption 3.1. *The system can be stabilized by at least one policy* $\mathbf{u}_k$, $\forall\ k \in \mathbb{N}_0$. *In other words, the closed-loop system is asymptotically stable for* $\mathbf{x}_k$ *under the control policy* $\mathbf{u}_k$.

This assumption is justified given that the work in [43, 44, 58]. Therein, control systems are guaranteed to be stable in the sense of Lyapunov.

3.3 Summary

In this chapter, a state-space representation of a nonlinear system model for a general unknown dynamic system is introduced and approximated using model (3.1), and the normalized and saturated error dynamics are defined for the system (*i.e.*, airship) considered in this thesis. From there, the optimal control problem has been formulated. After that, the derivation of control actions following the theory of conventional optimal control has been outlined. The results from this chapter are unsuitable for the Data-Driven Control (DDC) case due to the unknown parameter, namely the future value function, the gradient of the value function and the error dynamical function. Solutions to these limitations are offered in Chapters 4 and 5 by using value function approximations that allow the unknown parameters to be estimated online using the data gathered along the system trajectories, thereby bypassing the limitations by using adaptive approximation methods based on neural networks.

Chapter 4

Suboptimal Control with Critic Neural Network

This chapter outlines a suboptimal online Data-Driven Control (DDC) strategy building upon the optimal control and reinforcement learning techniques outlined in Chapter 3. By designing a value function approximator as a partially connected neural network, herein known as the critic network, solutions are offered for the limitations outlined in Chapter 3, namely the unknown error dynamical function (3.3), and therefore the future value function and its gradient in (3.8). It is emphasized that the formulation of this strategy is developed entirely based on the data collected along the trajectories and therefore required no a priori knowledge of any of the system dynamics or disturbance models. The controller development is outlined in detail in Section 4.1 and simulation results are quoted with discussions for the altitude control and altitude and velocity control tasks in Sections 4.2 and 4.3, respectively. Section 4.4 offers concluding remarks on the suboptimal Model-Free Control (MFC) strategy.

4.1 Suboptimal Controller Development

In this section, a generalized mathematical development of a Multi-Input Multi-Output (MIMO) MFC strategy is presented. The derivation addresses the problems of the unknown error mapping in (3.3) and the unknown future value function and its gradient in (3.8). The solution to the unknown equations is to construct a value function approximator as a partially connected neural network, called herein critic network, whose output is the current

value function approximation. The proposed controller is structured as a regulator whose task is to calculate the optimal correction term to current actuator state given the current state-error and actuator states. Leveraging a Q-learning like structure [38], the algorithm uses a normalized and saturated error term, (3.5), to encapsulate the state information. In contrast to the traditional Q-learning approach, the structure that is proposed in this work doesn't use a discount factor in the Hamilton-Jacobi-Bellman (HJB) equation to effectively limit the time horizon.

Motivated by Weierstrass higher order approximation theorem [28], the input to the critic network is a quadratic basis function constructed from the input-output data gathered online along the trajectories. The weights of the critic network are then used to calculate the control action following the theory of optimal control from the literature. This makes the control action optimal over the time interval from which the data were collected.

Consider the vector $\mathbf{w}_k$ which is the augmented error dynamical vector, (3.4), and the control action vector, (3.2). That is

$$\mathbf{w}_k \equiv [w_{k,1}, w_{k,2}, \ldots, w_{k,\bar{m}}]^T \equiv [\mathbf{e}_k^T \ \tilde{\mathbf{u}}_k^T]^T \in \mathbb{R}^{\bar{m}} \tag{4.1}$$

where $\bar{m} = n + m$. The value function (3.8) can be equivalently represented in terms of $\mathbf{w}_k$ as

$$V(\mathbf{w}_k) = \sum_{\iota=k}^{\infty} r(\mathbf{w}_\iota) = \frac{1}{2}\mathbf{w}_k^T \bar{\mathbf{P}} \mathbf{w}_k + V(\mathbf{w}_{k+1}) \tag{4.2}$$

by initializing $\bar{\mathbf{P}}$ as a block diagonal matrix whose upper-left and lower-right elements are $\mathbf{Q}$ and $\mathbf{R}$, respectively. That is

$$\bar{\mathbf{P}} = \begin{bmatrix} \mathbf{Q} \in \mathbb{R}^{n\times n} & \mathbf{0} \in \mathbb{R}^{n\times m} \\ \mathbf{0} \in \mathbb{R}^{m\times n} & \mathbf{R} \in \mathbb{R}^{m\times m} \end{bmatrix}.$$

Let $\hat{V}(\mathbf{w}_k)$ denote an approximation of the value function, $V(\mathbf{w}_k)$, given by

$$\hat{V}(\mathbf{w}_k) \equiv \frac{1}{2}\mathbf{w}_k^T \mathbf{P}^{[r]} \mathbf{w}_k \approx V(\mathbf{w}_k) \tag{4.3}$$

with $\mathbf{P}^{[r]}$ being an adaptable matrix that is updated online via the critic neural network. The elements of $\mathbf{P}^{[r]}$ are

$$\mathbf{P}^{[r]} = \begin{bmatrix} \mathbf{P}_{\text{ee}}^{[r]} \in \mathbb{R}^{n\times n} & \mathbf{P}_{\text{eu}}^{[r]} \in \mathbb{R}^{n\times m} \\ \mathbf{P}_{\text{ue}}^{[r]} \in \mathbb{R}^{m\times n} & \mathbf{P}_{\text{uu}}^{[r]} \in \mathbb{R}^{m\times m} \end{bmatrix}.$$

The form of (4.3) is consistent with the form of the value function for the Linear Quadratic Regulator (LQR) which has been used in the literature [38]. In the LQR case, the value function is quadratic in the states, whereas in the work presented here, the value function is quadratic in the augmented vector, $\mathbf{w}_k$, and thus indirectly quadratic in the states and the control actions, which represent the measurements taken along the trajectory.

Comparing the forms of (4.2) and (4.3), it can be seen that this proposal is justified. The diagonal elements $\mathbf{P}_{\text{ee}}^{[r]}$ and $\mathbf{P}_{\text{uu}}^{[r]}$ represent the weighting matrices for the terms that are quadratic in the states and control actions, respectively, while the off-diagonal matrices $\mathbf{P}_{\text{eu}}^{[r]}$ and $\mathbf{P}_{\text{ue}}^{[r]}$ can adapt to estimate the future value functions.

To find the optimal value function associated with the approximation, Bellman's principle of optimality is applied to (4.3). This yields

$$\hat{V}^*(\mathbf{w}_k) = \frac{1}{2} \min_{\tilde{\mathbf{u}}_k} \left\{ \mathbf{w}_k^T \mathbf{P}^{[r]} \mathbf{w}_k \right\}. \tag{4.4}$$

Evaluating (4.4) and solving for $\tilde{\mathbf{u}}_k^*$, the optimal control action is then given by

$$\hat{\mathbf{u}}_k^* = -\mathbf{P}_{\text{uu}}^{[r]^{-1}} \mathbf{P}_{\text{ue}}^{[r]} \mathbf{e}_k. \tag{4.5}$$

Here, $\hat{\mathbf{u}}_k^*$ is used to clearly represent that the control action is derived from the approximation of the value function while satisfying the criteria for optimality.

Remark 4.1. *Given that $\mathbf{P}_{uu}^{[r]}$ is a principal submatrix of the symmetric positive definite matrix $\mathbf{P}^{[r]}$, it is guaranteed that $\mathbf{P}_{uu}^{[r]}$ is also symmetric positive definite, thus $\mathbf{P}_{uu}^{[r]^{-1}}$ is guarenteed to exist.*

Although (4.5) is consistent with the framework of optimal control, the matrix $\mathbf{P}^{[r]}$ still needs to be constructed in such a way that the dynamics of the value function are encapsulated. To this end, we introduce Weierstrass higher-order approximation theorem.

Theorem 4.1 (Weierstrass higher-order approximation theorem)**.** *Let $f(x) \in \mathcal{C}^m(\Xi)$, for $\Xi \subseteq \mathbb{R}^n$, then there exists a polynomial $P(x)$ such that it converges uniformly to $f(x)$, and all its partial derivatives up to order m converge uniformly. [2, 21]*

Theorem 4.1 says that any function that is continuous of order m can be approximated in a uniformly convergent manner by a polynomial.

Using Theorem 4.1 and following the developments outlined in [38], augmented value

function approximation model (4.2) can be represented exactly as an infinite polynomial

$$V(\mathbf{w}_k) = \frac{1}{2}\mathbf{w}_k^T \bar{\mathbf{P}} \mathbf{w}_k + V(\mathbf{w}_{k+1}) = \sum_{i=0}^{\infty} \omega_i^* \varphi_i = \sum_{i=0}^{i=L} \omega_i^* \varphi_i + \sum_{j=L+1}^{\infty} \omega_j^* \varphi_j \tag{4.6}$$

for some $L \in \mathbb{N}$. In (4.6) ω_i^* represents the true ith coefficient of the polynomial vector. Therefore, the value function defined in (4.3) is approximated by truncating the second sum in (4.6). This is given in compact form as

$$\hat{V}(\mathbf{w}_k) = {\boldsymbol{\Omega}^*}^T \boldsymbol{\varphi}(\mathbf{w}_k) \quad \text{where} \tag{4.7}$$

$$\boldsymbol{\Omega}^* = [\omega_1^*, \omega_2^*, \ldots, \omega_{\bar{m}}^*]^T \in \mathbb{R}^{\bar{m}}$$

is the vector of true parameters that minimize the approximation error. The vector $\boldsymbol{\varphi}(\cdot) \in \mathbb{R}^{\bar{m}}$ is composed of basis functions that represent activation functions for the critic neural network. In the control literature, this is generally a quadratic function but it has also been used in the neural network literature as functions including Gaussian radial basis, sigmoidal, hyperbolic tangent [38]. The basis function represents a mapping such that

$$\boldsymbol{\varphi}(\mathbf{w}_k) : \mathbb{R}^{\bar{m}} \to \mathbb{R}^L.$$

The approximation error associated with the truncation of the polynomial sum is,

$$\epsilon^{\text{approx}} = \sum_{j=L+1}^{\infty} \omega_j^* \varphi_j.$$

It is clear from Theorem 4.1 that as $L \to \infty$, $\epsilon^{\text{approx}} \to 0$. Motivated by Theorem 4.1, a single-layer partially connected neural network, the critic network, is proposed. By introducing a vector $\mathbf{p}$ consisting of stacked columns of $\mathbf{P}^{[r]}$ at policy index $r \in \{0, 1, \ldots\}$

$$\mathbf{p}^{[r]} = [p_{1,1}^{[r]}, p_{1,2}^{[r]}, \ldots, p_{i,j}^{[r]}]^T \quad \forall \quad (i,j) \in \{1, 2, \ldots, \bar{m}\}^2$$

the approximate value function (4.3) is written as

$$\hat{V}(\mathbf{w}_k) = \frac{1}{2}{\mathbf{p}^{[r]}}^T (\mathbf{w}_k \otimes \mathbf{w}_k) = \frac{1}{2}{\mathbf{p}^{[r]}}^T \bar{\rho}(\mathbf{w}_k). \tag{4.8}$$

The approximation model given by (4.8) defines a critic network whose input is $\mathbf{w}_k$ and

output is $\hat{V}(\mathbf{w}_k)$. The output weights are $\mathbf{p}^{[r]}$ and the activation function is given by

$$\bar{\rho}(\cdot) : \mathbb{R}^{\bar{m}} \to \mathbb{R}^{\bar{m}^2} = \mathbf{w}_k \otimes \mathbf{w}_k. \tag{4.9}$$

The $\otimes$ operator represents the Kronecker product. Notice that the weights of the neural network correspond to the matrix $\mathbf{P}^{[r]}$, allowing the value function to learn from the agent's interaction with the environment.

The enforced symmetry of $\mathbf{P}^{[r]}$ and the repeated terms in the Kronecker product are taken advantage of to further simplify (4.8) by removing the redundant terms and reduce the order and the computational cost of (4.8). Let

$$\mathbf{\Omega_c}^{[r]} = [p_{1,1}^{[r]}, p_{1,2}^{[r]}, \dots, p_{i,j}^{[r]}]^T, \quad \forall i \in \{1, 2, \dots, \bar{m}\},\ j \leq i \quad \text{and} \tag{4.10}$$

$$\rho(\mathbf{w}_k) = [C_{1,1} w_{k,1}^2, C_{1,2} w_{k,1} w_{k,2}, \dots, C_{i,j} w_{k,i} w_{k,j}]^T \quad \forall i \in \{1, 2, \dots, \bar{m}\},\ j \leq i \quad \text{where}$$

$$C_{i,j} = \begin{cases} \frac{1}{2}, & \text{if} \quad i = j \quad \text{and} \\ 1, & \text{if} \quad i \neq j. \end{cases}$$

It follows that the activation function for the critic network is a mapping such that

$$\rho(\cdot) : \mathbb{R}^{\bar{m}} \to \mathbb{R}^{\eta}$$

where $\eta = \frac{1}{2}(\bar{m})(\bar{m}+1)$. (4.8) is equvalently written as

$$\hat{V}(\mathbf{w}_k) = \mathbf{\Omega_c}^{[r]^T} \rho(\mathbf{w}_k). \tag{4.11}$$

It is clear that

$$\lim_{n,m \to \infty} \frac{\eta}{\bar{m}^2} = \frac{1}{2}.$$

So for MIMO systems with $n, m \gg 1$, the computational efficiency is approximately doubled using the reduced order representation. The structure of the reduced order critic neural network modeled using (4.11) is shown in Figure 4.1.

Remark 4.2. *For Weierstrass higher-order approximation theorem as $L \to \infty$, $\epsilon^{approx} \to 0$, however, given that $\mathbf{P}^{[r]}$ is a Hankel matrix, no new information is added by the exclusion of the upper right elements. In fact, (4.8) and (4.11) are identical expressions, therefore the bound ϵ^{approx} is unaffected by the reduction in order.*

The weights of (4.11) are updated using the gradient descent method applied on the

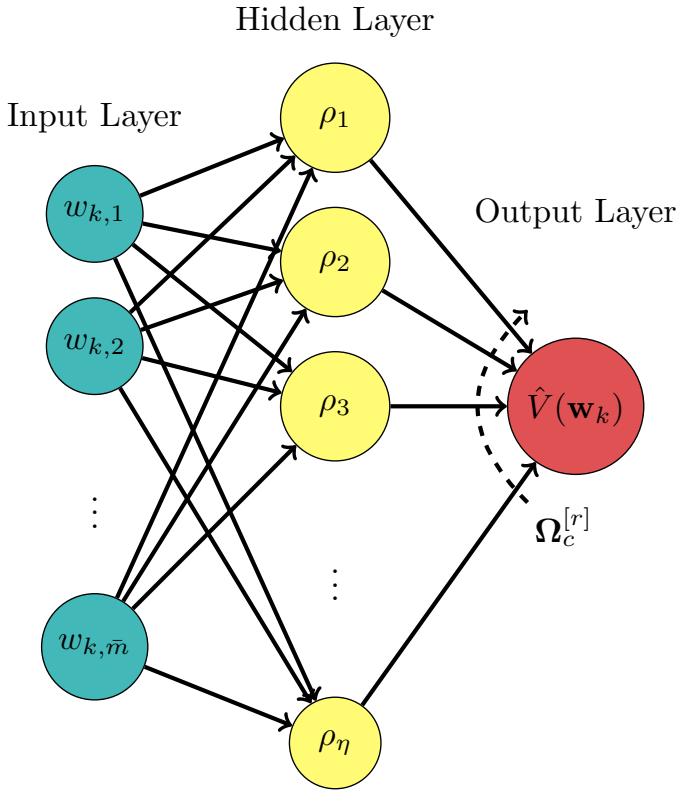


Figure 4.1: Structure of the partially connected critic neural network.

squared temporal difference error between the desired value error function, defined as

$$\tilde{V}_{k,k+1}^{[d]} = V(\mathbf{w}_k) - V(\mathbf{w}_{k+1}) = \frac{1}{2}\mathbf{w}_k^T \bar{\mathbf{P}} \mathbf{w}_k$$

and the estimated value error function, which is written compactly as

$$\tilde{V}_{k,k+1}^{[r]} = \mathbf{\Omega_c}^{[r]^T} \tilde{\rho}_{k,k+1} \quad \text{by defining} \quad \tilde{\rho}_{k,k+1} = \rho(\mathbf{w}_k) - \rho(\mathbf{w}_{k+1}).$$

To advise the critic neural network the squared error of the desired value error function and estimated value error function is defined as

$$\delta_c = \frac{1}{2} \sum_{k=0}^{\eta} \left[\tilde{V}_{k,k+1}^{[d]} - \tilde{V}_{k,k+1}^{[r]} \right]^2. \tag{4.12}$$

Introduce the batch matrices that are used for every update to the critic network weights at each policy evaluation step, r.

$$\boldsymbol{\Upsilon} = [\boldsymbol{\Upsilon}_0, \boldsymbol{\Upsilon}_1, \dots, \boldsymbol{\Upsilon}_{\bar{\eta}-1}]^T \in \mathbb{R}^{\bar{\eta}\times\eta} \quad \text{with}$$

$$\boldsymbol{\Upsilon}_\kappa = \tilde{\rho}_{k,k+1}^T \quad \forall \quad \kappa \in \{0, 1, \dots, \bar{\eta}-1\} \mid \kappa = k \text{ modulo } \eta \quad \text{and}$$

$$\mathbf{v}^{[d]} = \left[v_0^{[d]}, v_1^{[d]}, \dots, v_{\bar{\eta}-1}^{[d]}\right] \in \mathbb{R}^{\bar{\eta}}, \quad \text{where} \quad v_\kappa^{[d]} = \frac{1}{2}\mathbf{w}_k^T \bar{\mathbf{P}} \mathbf{w}_k \quad \forall \quad \kappa \in \{0, 1, \dots, \bar{\eta}-1\}.$$

Note that inequality $\bar{\eta} \geq \eta$ must be satisfied in order for the data gathered in the batch matrices to be persistently exciting [7]. The squared error modelled in (4.12) can be written in compact form as

$$\delta_c = \frac{1}{2}\left\|\mathbf{v}^{[d]} - \boldsymbol{\Upsilon}{\boldsymbol{\Omega}_\mathbf{c}}^{[r]}\right\|^2. \tag{4.13}$$

The critic network weights are updated on every policy iteration $r \in \{0, 1, \dots\}$ such that η is divisible by $k+1$. Weights of the critic neural network are updated using gradient descent for some learning rate parameter $\alpha_c \ll 1$. The update law is chosen for some $\Omega_c^{[r]}$ that minimizes (4.13). That is

$$\boldsymbol{\Omega}_c^{*[r]} = \min_{\Omega_c^{[r]}} \delta_c, \quad \text{which follows} \quad \frac{\partial \delta_c}{\partial \boldsymbol{\Omega}_c} = 0 = \boldsymbol{\Upsilon}^T\left(\mathbf{v}^{[d]} - \boldsymbol{\Upsilon}{\boldsymbol{\Omega}_\mathbf{c}}^{[r]}\right).$$

The minimizing expression for $\boldsymbol{\Omega}_c^{[r]}$ over the sampled and stored data is then given by

$$\boldsymbol{\Omega}_c^{*[r]} = \left(\boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon}\right)^{-1}\mathbf{v}^{[d]}\boldsymbol{\Upsilon}. \tag{4.14}$$

Therefore, to satisfy (4.14), $\left(\boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon}\right)^{-1}$ must exist. Using gradient descent for the update law of the weight vector

$${\boldsymbol{\Omega}_\mathbf{c}}^{[r+1]} = {\boldsymbol{\Omega}_\mathbf{c}}^{[r]} - \alpha_c \frac{\partial \delta_c}{\partial {\boldsymbol{\Omega}_\mathbf{c}}^{[r]}} = {\boldsymbol{\Omega}_\mathbf{c}}^{[r]} - \alpha_c \boldsymbol{\Upsilon}^T\left(\boldsymbol{\Upsilon}{\boldsymbol{\Omega}_\mathbf{c}}^{[r]} - \mathbf{v}^{[d]}\right). \tag{4.15}$$

The matrix $\mathbf{P}^{[r+1]}$ is reconstructed using the inverse of (4.10).

The control action (4.5) is then computed using the updated weight matrix $\mathbf{P}^{[r+1]}$. It is noted that the control action is considered optimal only in the range over which the input-output data is collected and when the weight matrices have converged. That is the

control action is

$$\tilde{\mathbf{u}}_k = -\mathbf{P}_{\text{uu}}^{[r]^{-1}}\mathbf{P}_{\text{ue}}^{[r]}\mathbf{e}_k = \begin{cases} \tilde{\mathbf{u}}_k & \text{if } \left\|\boldsymbol{\Omega}_c^{[r]} - \boldsymbol{\Omega}_c^{[r-1]}\right\| \geq \epsilon^{\text{conv}} \\ \tilde{\mathbf{u}}_k^* & \text{if } \left\|\boldsymbol{\Omega}_c^{[r]} - \boldsymbol{\Omega}_c^{[r-1]}\right\| < \epsilon^{\text{conv}} \end{cases} \tag{4.16}$$

for some small convergence threshold, $\epsilon^{\text{conv}} \ll 1$. The overall structure of the suboptimal state-feedback controller is depicted in Figure 4.2.

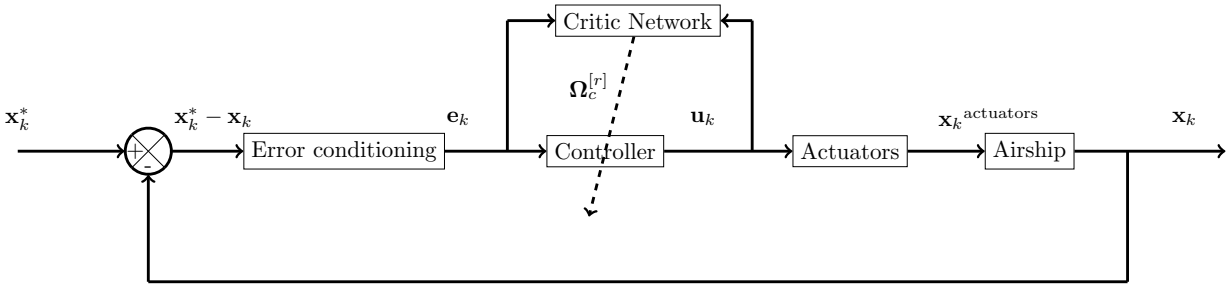


Figure 4.2: Block diagram for the suboptimal controller.

The error conditioning block takes the true error as its input and has the normalized and saturated errors, (3.5) and (3.6), as its output. The controller block represents the control policy, (4.16), which is parameterized from the critic neural network. The output for the controller block is passed to actuators of the airship. The resulting states are treated as the outputs. The network weights, $\boldsymbol{\Omega}_c^{[r]}$ are updated on each policy iteration by the critic network block, thereby changing the parameter of the controller. This structure represents an adaptive control system. The algorithm is summarized in Algorithm 4.1.

4.2 Altitude Control with Single Network Controller

This section outlines a single-input 2-output control task for the altitude, z [m], and vertical velocity, $\dot{z}$ [m/s], of a dirigible simulation model. This preliminary experimental test is used to validate the controller in a low-dimensional task. The primary objectives are to investigate the sensitivities of the controller, find acceptable initial conditions and perform a robustness study against external disturbances. The simulation model that is used as the plant for this task is introduced in detail in Section 4.2.1 with the controller parameters for the suboptimal controller being introduced in Section 4.2.2. To offer the reader a

Algorithm 4.1 Algorithm 1: Suboptimal Control with Critic Neural Network

Input:

$\mathbf{P}^{[0]}$: Symmetric Positive Definite
$\mathbf{Q} \in \mathbb{R}^n$: Symmetric Positive Definite
$\mathbf{R} \in \mathbb{R}^m$: Symmetric Positive Definite
K: Maximum value of the discrete-time variable

Output:

Initialize the following variables:
$r \leftarrow 0$
$\boldsymbol{\Upsilon} \in \mathbb{R}^{\eta \times \eta}$ as an empty matrix
$\mathbf{v}^{[d]} \in \mathbb{R}^{\eta}$ as an empty column vector
Measure the initial error vector $\mathbf{e}_0$
for $(k \leftarrow 0,\ k \leq K,\ k \leftarrow k+1)$ **do**
    Measure the states $\mathbf{x}_{k+1}$
    Calculate the error $\mathbf{e}_{k+1}$ using (3.5) and (3.6)
    Calculate the control action $\tilde{\mathbf{u}}_{k+1}$ using (4.16)
    Calculated the augmented vector $\mathbf{w}_{k+1} \leftarrow \left[\mathbf{e}_{k+1}^T \tilde{\mathbf{u}}_{k+1}^T\right]^T$
    $\kappa \leftarrow k$ modulo $\eta - 1$
    Set $\boldsymbol{\Upsilon}_{\kappa} \leftarrow \tilde{\rho}_{\kappa,\kappa+1}$
    Set $\mathbf{v}_{\kappa}^{[d]} \leftarrow \frac{1}{2}\mathbf{w}_k^T \bar{\mathbf{P}} \mathbf{w}_k$
    if $(k+1)$ is divisible by η **then**
        Increase the policy index $r \leftarrow r+1$
        Update the weights of the critic matrix $\boldsymbol{\Omega}_c^{[r]} \leftarrow \boldsymbol{\Omega}_c^{[r-1]} + \alpha_c \boldsymbol{\Upsilon}^T \left(\boldsymbol{\Upsilon} \boldsymbol{\Omega_c}^{[r-1]} - \mathbf{v}^{[d]}\right)$
    end if
end for

reference against a state-of-the-art DDC strategy, a comparison controller is introduced in Section 4.2.3 with the initial conditions and implementation restrictions being discussed.

To test the performance of the suboptimal controller an ablation study is performed in Section 4.2.4, over different initial conditions of the critic learning rate parameters, α_c. Section 4.2.5 uses nominal single waypoint control experiments over a range of initial critic values as an ablation study to investigate the sensitivity of the controller to the initial critic weights, $\boldsymbol{\Omega}_c^{[r]}$ and then focuses on two specific controllers to investigate the responses in depth. Finally, Section 4.2.6 extends the work in Section 4.2.5, to multiple waypoint tests to and explores the adaptability of the proposed algorithms. A summary of the findings is presented in Section 4.2.7.

4.2.1 ROS/Gazebo Model

In this section the Robot Operating System (ROS) airship simulation model[1] is presented with discussions of the state and action spaces for the altitude control task.

In this work, the Gazebo physics engine is leveraged as a backend for the ROS interface for the control and perception of a deformable airship simulation model, Figure 4.3 with and without external wind disturbance generated using a Dryden turbulent wind model. The ROS/Gazebo platform enables highly realistic physical interaction. The representation of non-rigid structures from constituent rigid elements that are interconnected with spring and damper style connections is used to simulate the true dynamics of an airship, relaxing the need for rigidity assumptions that are common in the literature, as outlined in Chapter 2. The rigidity of the entire structure is influenced by the spring and damping coefficients of the interconnections. The stiffness of the joints between the constituent objects is adjusted to accommodate variable translational and rotational Degree of Freedom (DoF) for each member of the dirigible. The internal relationships of the constituent rigid elements are also considered by the simulation model.

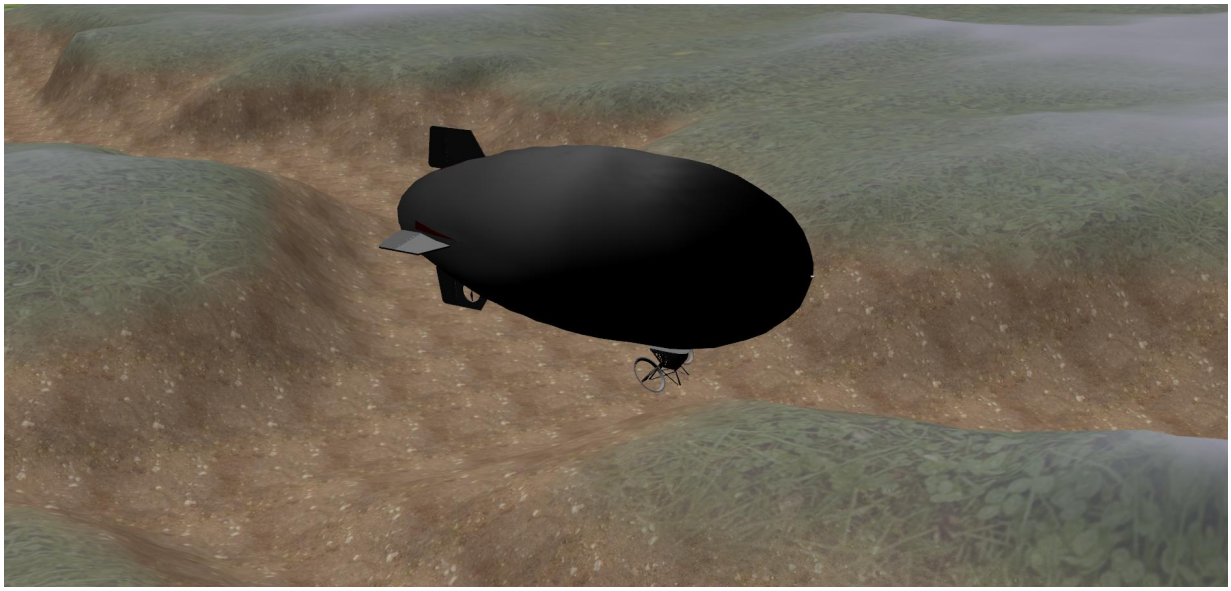


Figure 4.3: Simulation model of the dirigible developed in [58] that is used for the experimental results.

[1] This model is a result of Price et al. who give a detailed breakdown and preliminary control results in [58]. Installation and use instructions are available here.

Pseudo real-time capabilities are provided using the ROS platform. This is to say that real-time control and sensing are possible so long as the wall time for the calculation of the control algorithm doesn't supersede the discrete-time step size, $T \in \mathbb{R}^+$. The odometry readings that are available through the various ROS topics are calculated using a Kalman filter on the noisy Inertial Measurement Unit (IMU) and positioning readings. It is reassuring that in [43, 44, 58] the simulation model and ROS/Gazebo environment are used to train reinforcement learning and Proportional Integral Derivative (PID) controllers which are then applied to the physical dirigible that the model is based on, without needing to be retrained on the physical system. This is a sign that the simulation model and real prototype are indeed reflective of reality.

Remark 4.3. *Given the model-free nature of the work presented in this paper, the controllability is not verified, as an accurate system model is required. It is taken for granted herein that the system is controllable given the success of the authors in [43, 44, 58]. That is there exist admissible control policies.*

4.2.2 Altitude Controller Implementation with Single Network Controller

This section outlines the control of the dirigible simulation model that is used in the altitude control tasks. In this task, the vertical position, z [m], and vertical velocity, $\dot{z}$ [m/s], are controlled by regulating the angular speed, ω [°/s], of the propellers. The states, shown in Figure 4.4, are given by

$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \end{bmatrix} \in \mathbb{R}^2.$$

The angular velocities of the propellers, shown in Figure 4.5, of the left and right propellers are equal in magnitude and opposite in direction. This causes a thrust in the longitudinal direction as the geometry of the blades is inverted. The values of the angular speed is saturated in the range of [-2864.8, 2864.8][°/s]

$$\omega_k = \omega_{k,r} = -\omega_{k,l} \in [-2864.8, 2864.8] \text{ [°/s]}.$$

For all experiments, the initial angular velocities of the propellers are $\omega_0 = 658.9$ [°/s]. This corresponds to an initial, upwards, thrust vector acting along the positive z-axis.

For all experiments the matrix $\mathbf{Q} \in \mathbb{R}^{\bar{m}}$ is held constant as a symmetric positive definite

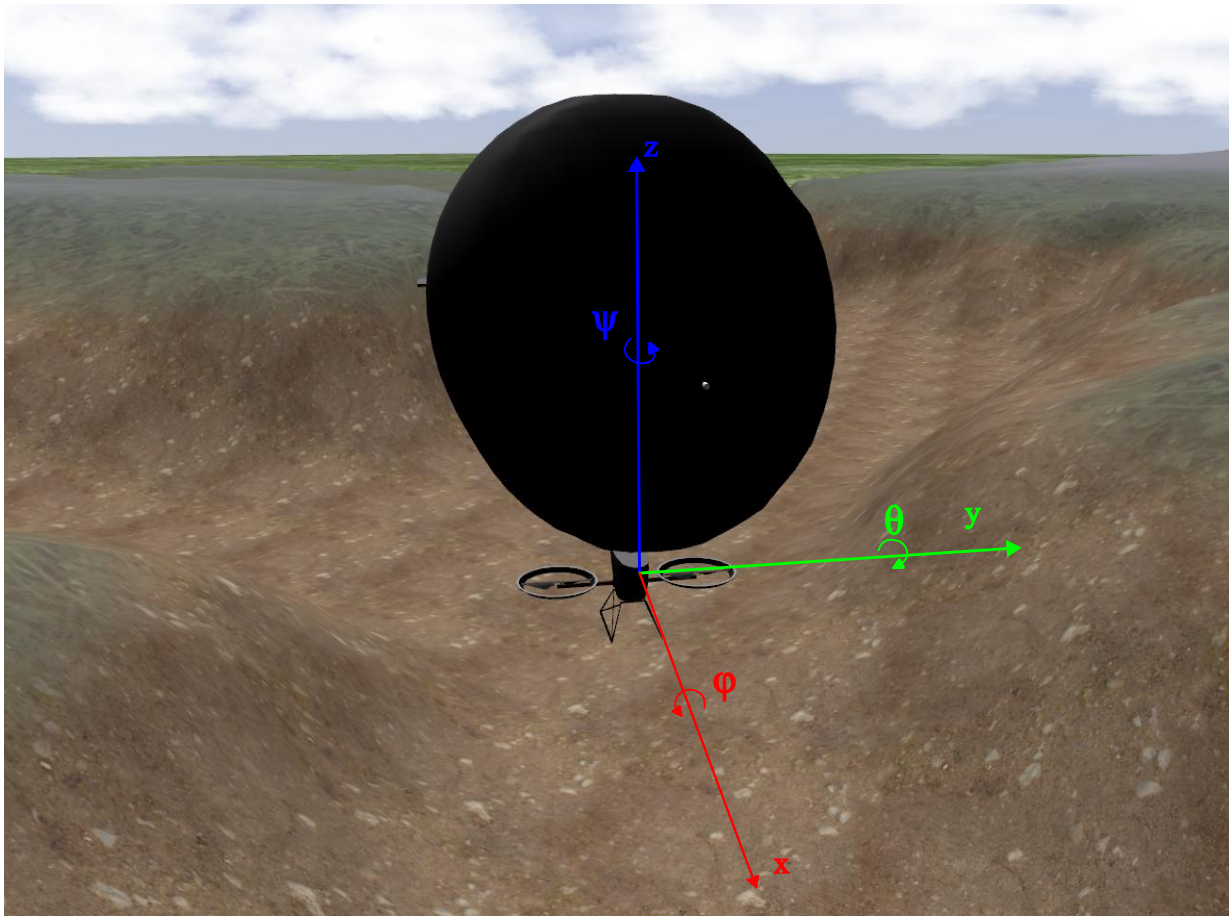


Figure 4.4: Vertical position, z [m] and vertical velocity, $\dot{z}$ [m/s] axis definitions. Note that z and $\dot{z}$ are measured with respect to the World reference frame.

matrix whose element for the i^{th} row and j^{th} column are

$$q_{i,j} = \begin{cases} 10^{-1} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}, \quad \forall (i,j) \in \{1, 2, \ldots, \bar{m}\}^2.$$

Matrix $\mathbf{R} \in \mathbb{R}^m$ is also held constant over the different experiments as a symmetric

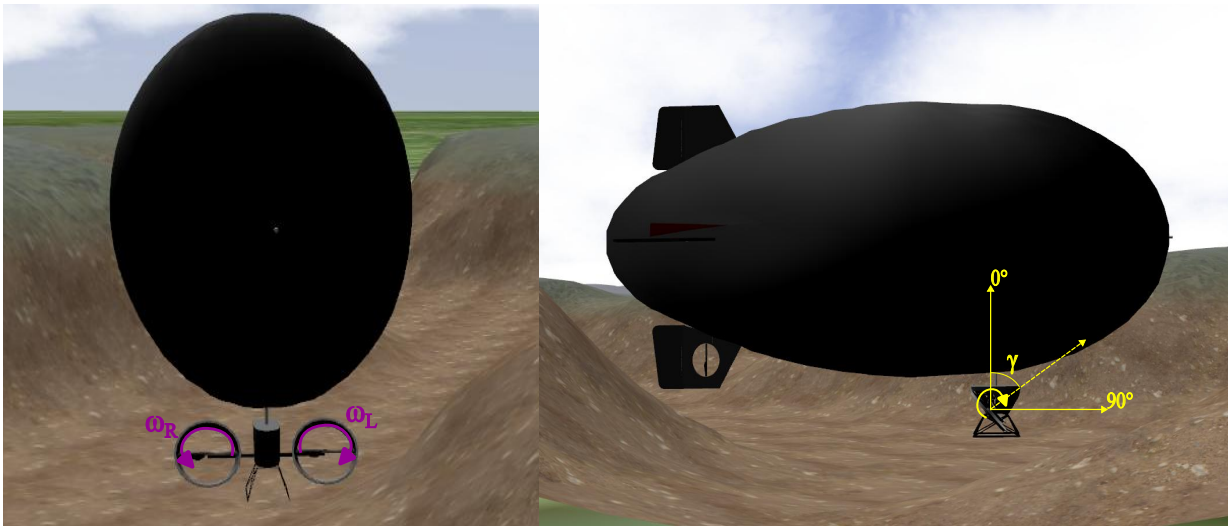


Figure 4.5: Angular velocities for the left, ω_L, and right, ω_R propellers of the airship, where $\omega_{k,l} = -\omega_{k,r} = \omega_k$ [°/s].

positive definite matrix whose element for the i^{th} row and j^{th} column are

$$r_{i,j} = \begin{cases} 10^{-1} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}, \quad \forall (i,j) \in \{1, 2, \ldots, n\}^2.$$

The values of the states are updated with frequency $f_{\text{sample}} = 150$ [Hz] and conditioned using an extended Kalman filter which calculates its values from global positioning sensor and IMU sensor fusion. The altitude at time k, z_k [m], is read as an Odometry message from the "/blimp/ground_truth/odometry" ROS topic. In contrast, the current vertical velocity at time k, $\dot{z}_k$ [m/s], is read as a TwistedStamped message from the "/blimp/ground_speed" topic. The altitude control script creates a node called "/alt_controller" in the "/blimp" namespace that updates a list of motor speeds as Actuator messages that are published to the airship over the "/blimp/command/motor_speed" topic with frequency $f_{\text{publish}} = 15$ [Hz]. The rqt graph for the altitude controller is shown in Figure 4.6,

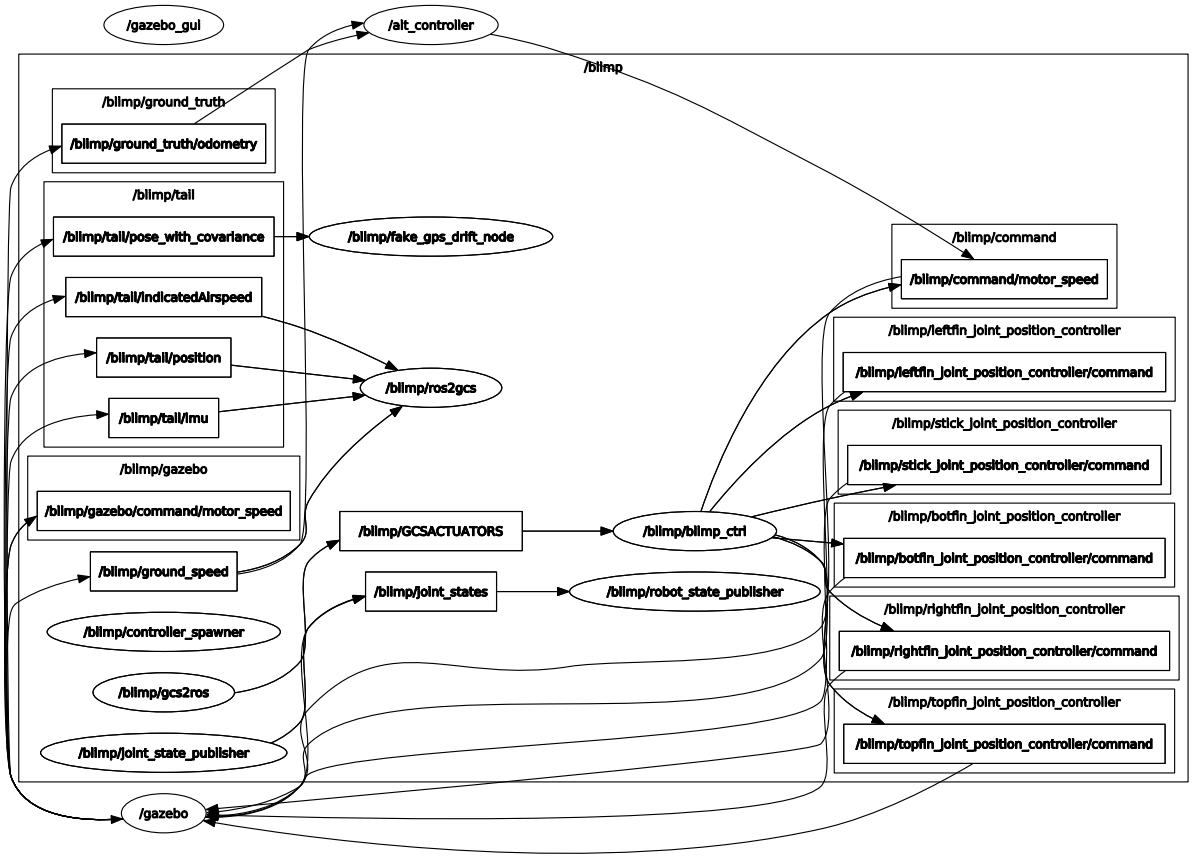


Figure 4.6: rqt graph for the proposed altitude controller.

4.2.3 Altitude Controller Implementation with Comparison Controller

The controllers presented in this work are compared to a state-of-the-art MFC strategy, presented in [77]. The comparison controller is introduced in Section 2.2.1 as a Dynamic Linearization Techniques (DLT) whose output transition equation is given in (2.3), and repeated here for self containment

$$\Delta \mathbf{y}_{k+1} = \bar{\mathbf{\Phi}}_k^{\text{ffdl}} \Delta \bar{\mathbf{H}}_k. \tag{4.17}$$

In (4.17), the input-output history is captured by the $\Delta\bar{\mathbf{H}}_k$ term, which is defined as

$$\Delta\bar{\mathbf{H}}_k = [\Delta\mathbf{y}_k^T, \ldots, \Delta\mathbf{y}_{k-L_y+1}^T, \Delta\tilde{\mathbf{u}}_k^T, \ldots, \Delta\tilde{\mathbf{u}}_{k-L_u+1}^T]^T \in \mathbb{R}^{mL_y+nL_u}.$$

The so-called Pseudo-Partitioned Jacobian Matrix (PPJM), $\bar{\mathbf{\Phi}}_k^{\text{ffdl}}$, is an adaptive term that is updated by the input-output data gathered online and is meant to encapsulate the input-output behaviour of the system. The equation for updating the PPJM is

$$\Delta\hat{\mathbf{\Phi}}_k^{\text{ffdl}} = \frac{\eta(y_k - y_{k-1})\Delta\mathbf{H}_{k-1}^T}{\mu + \|\Delta H_{k-1}\|^2} + \frac{\eta\hat{\mathbf{\Phi}}_{k-1}^{\text{ffdl}}\Delta\mathbf{H}_{k-1}\Delta\mathbf{H}_{k-1}^T}{\mu + \|\Delta H_{k-1}\|^2} \in \mathbb{R}^{m\times(mL_y+nL_u)}.$$

Note that the PPJM is decomposed as

$$\hat{\mathbf{\Phi}}_0^{\text{ffdl}} = \left[\hat{\phi}_1, \ldots, \hat{\phi}_{L_y}, \hat{\phi}_{L_y+1}, \ldots, \hat{\phi}_{L_y+L_u}\right].$$

To ensure diagonal dominance of the PPJM the authors propose a resetting algorithm, which resets elements of the matrix that become too small, too large or change signs. The mathematical formulation of these rules are neglected here for readability, but they are available in [77]. The values that are used for the resetting algorithm are, $a = 1$, $b_1 = 5.0$ and $b_2 = 10^{-2}$.

The control action for the Full Form Dynamic Linearization (FFDL) controller is given as

$$\Delta\mathbf{u}_k = \frac{\hat{\phi}_{k,L_y+1}}{\lambda + \left\|\hat{\phi}_{k,L_y+1}\right\|^2}\left[\rho_{L_y+1}\left(y_{k+1}^* - y_k\right) - \sum_{i=1}^{L_y}\rho_i\hat{\phi}_k \cdot \Delta y_{k-i+1} - \sum_{i=L_y+2}^{L_y+L_u}\rho_i\hat{\phi}_{k,i}\Delta\mathbf{u}_{k+L_y-i+1}\right]. \tag{4.18}$$

The initial value of the PPJM is chosen to be

$$\hat{\mathbf{\Phi}}_k^{\text{ffdl}} = \left[0.5\right].$$

A few notes on the comparison controller are required. In both the altitude controller and the altitude and velocity controller, discussed later, various values of all the controller parameters are attempted, looping through different parameters using a grid search method, until acceptable responses were observed. The ranges for each parameter were taken from the previous works, namely [30, 77, 78]. The only successful iterations were ob-

Table 4.1: Parameter values for the FFDL comparison controller for the altitude control task.

Parameter	Value
η	1
μ	1
ρ	10^{-1}
λ	10^{-2}
L_y	0
L_u	1

served when $L_y = 0$ and $L_u = 1$, which correspond to the structure of the Compact Form Dynamic Linearization (CFDL), which is a DLT method. Furthermore, the algorithms showed extreme sensitivity to the initial conditions. An attempt was made to discuss the algorithm with the authors, but a communication channel was not able to be formed. It should also be mentioned that the formulation of the suboptimal controllers proposed in this work requires that the states to be controlled, and their derivatives, both be present in the state-space. The FFDL doesn't have that restriction, rather it requires that $m \leq n$, barring the algorithm from including the derivatives in the experiments presented in this work. These limitations prevent the comparison from being one-to-one with the proposed controllers, however, the FFDL was the only MIMO MFC technique whose results from their works could be successfully reproduced and applied to the airship.

It is noted that in order to avoid divergence in the FFDL controller, the actuator signals had to be restricted to $\omega^{\text{ffdl}} \in [-572.96, 1718.87]$ [°/s] which is a subset of the action-space that the proposed controllers are able to manage.

4.2.4 Learning Rate Ablation Study

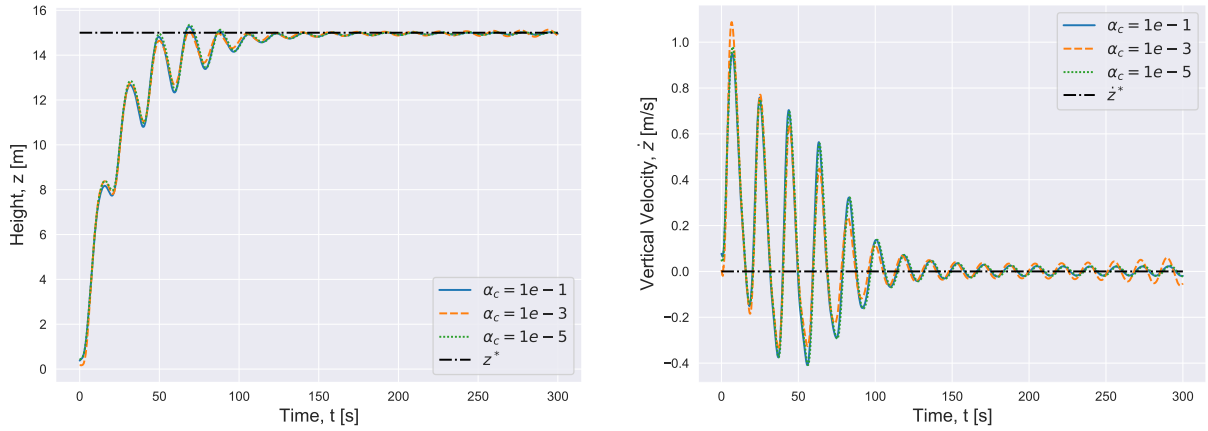
In this section three critic learning rates, $\alpha_c \in \{10^{-1}, 10^{-3}, 10^{-5}\}$, are applied to the suboptimal controller, while holding all other design variables constant, to observe the sensitivity of the system to the choice of learning rate. In all cases the target states, $\mathbf{x}^*$, and the maximum values of for the states used to calculate the error in (3.5), $\mathbf{x}^{\max}$, for the controller are

$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \end{bmatrix} = \begin{bmatrix} 15 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\max} = \begin{bmatrix} z^{\max} \\ \dot{z}^{\max} \end{bmatrix} = \begin{bmatrix} 22.5 \\ 1.5 \end{bmatrix}. \tag{4.19}$$

The initial value of the weights of the critic weights in all three cases are

$$\Omega_c^{[0]} = [0.50, 0.65, 1.00, 0.36, 0.55, 0.36]^T.$$

The results from the learning rate ablation study are shown in Figures 4.7 and 4.8.



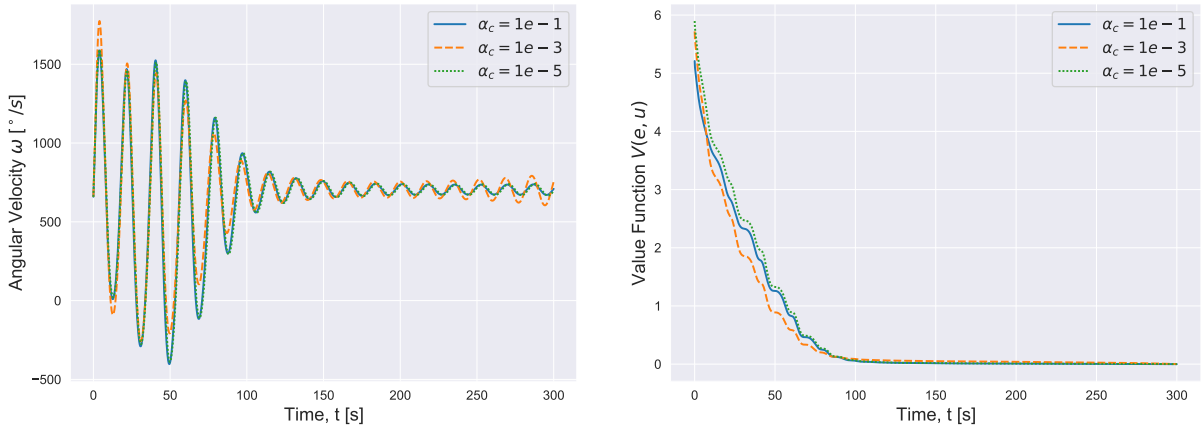
(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

Figure 4.7: State responses for the learning rate ablation study for $\alpha_c \in \{10^{-1}, 10^{-3}, 10^{-5}\}$ for a single waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

From the plots, very little difference is observed between the three learning rates. Both the altitude and vertical velocities successfully track the desired waypoints and oscillate about them with small amplitudes. Additionally, convergence is observed in the value functions around the 100 [s] period in all three cases, with the transient regions of the value functions also showing very similar responses. To quantify the system responses, the Integral Absolute Error (IAE) and Integral Time Absolute Error (ITAE) metrics are quoted in Table 4.2(a) for the systems vertical position and Table 4.2(b) for the vertical velocity.

Very few differences in the error metrics are observed. All three choices, marginally outperform the others in at least one metric. This shows that the algorithm is relatively insensitive to the choice in learning rate in the nominal case. It is noted however that when learning rates above $\alpha_c = 10^{-1}$ were chosen, the updates in some policy iteration steps became too large, and the adapted network weights violated the symmetric positive definite constraint imposed on $P^{[r]}$. Divergence was observed in these cases. For the



(a) Angular velocity, ω [°/s]

(b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 4.8: Actuator signals and value function responses from the learning rate ablation study for $\alpha_c \in \{10^{-1}, 10^{-3}, 10^{-5}\}$ for a single waypoint experiment. (a) actuator signal, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

Table 4.2: Error metrics for altitude controllers with learning rates in the range $\alpha_c \in \{10^{-1}, 10^{-3}, 10^{-5}\}$

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
$\alpha_c = 10^{-1}$	3587.08	1058045.39
$\alpha_c = 10^{-3}$	3572.25	1097684.99
$\alpha_c = 10^{-5}$	3505.54	1027536.64

(b) Error metrics for altitude control $\dot{z}$ [m/s]

Controller	IAE	ITAE
$\alpha_c = 10^{-1}$	337.09	188462.52
$\alpha_c = 10^{-3}$	330.30	212779.86
$\alpha_c = 10^{-5}$	340.07	192794.84

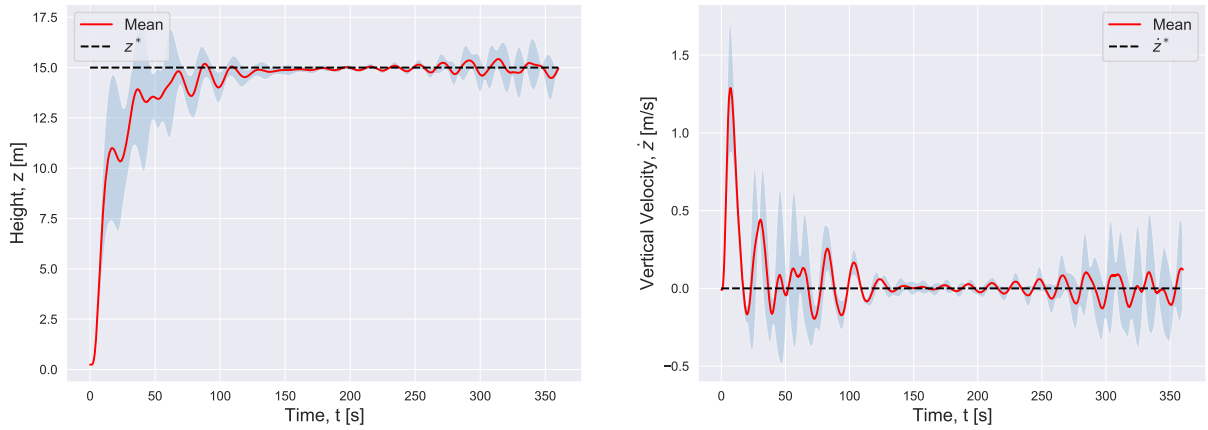
remainder of this section, the critic learning rate $\alpha_c = 10^{-1}$ is used.

4.2.5 Altitude Controller with Single Waypoint

In this section an ablation study is performed using different initial values of the critic weights, $\mathbf{\Omega}_c^{[0]}$, to determine the sensitivity of the system response with respect to the initial weights. In all cases in this section, the target states, $\mathbf{x}^*$, and the maximum values are the same as in (4.19).

Five initial values of the critic weights are randomly initialized and tested using random seeds. The initial weights are given in Appendix A.1 for completeness. It must be

mentioned, that six sets of critic weights were attempted, however, one of them diverged during the experiments, so it was excluded from the study as an outlier. It is mentioned here strictly for transparency. The results of the ablation study over the different random seeds are plotted in Figure 4.9 with the solid red line indicating the mean of each controller for a given time interval and the shaded blue area indicating a vertical range of $\pm$ 1 standard deviation from the mean.



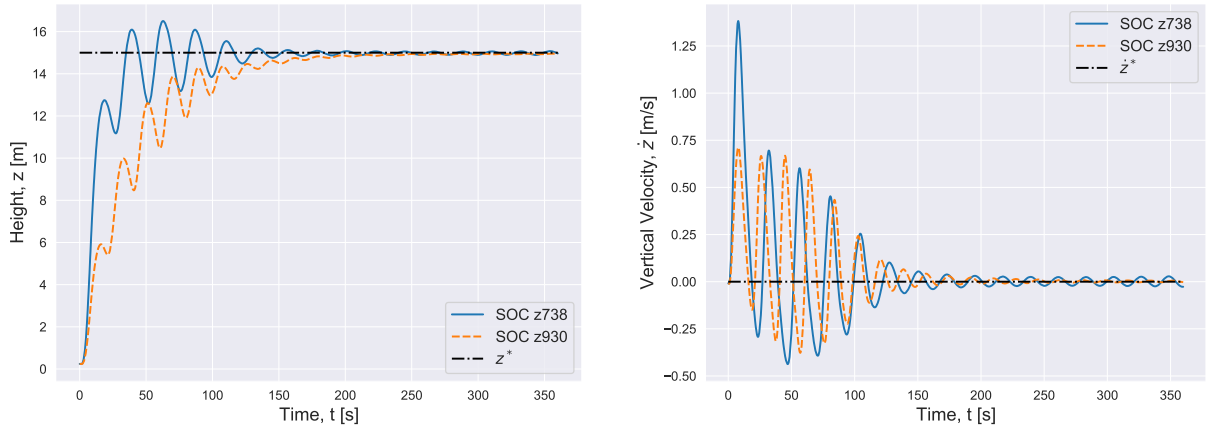
(a) z [m] (b) $\dot{z}$ [m/s]

Figure 4.9: Ablation study for altitude control, varying the initialization of the critic weights $\Omega_c^{[0]}$ for a single waypoint. The red line indicates the mean response and the shaded area represents a vertical width of one standard deviation from the mean system response. (a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s].

Notations 4.1. *Herein, SOC is used to refer to the suboptimal control structure with a single neural network, the critic network. A controller "SOC z###" refers to the suboptimal controller for states in* (4.19)*, with random seed ###.*

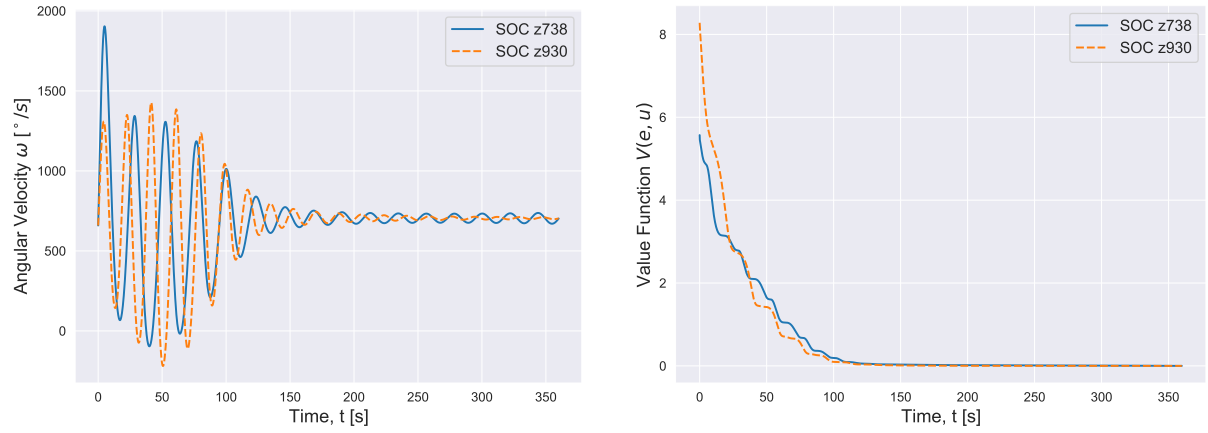
The state response from two of the controllers in the ablation study is plotted in Figure 4.10. The actuator signals and associated value functions are shown in Figure 4.11.

From the graph, it is clear that the two proposed suboptimal controllers, which vary only by the initial values of $\Omega_c^{[0]}$, are able to track the desired reference signals, with little to no fluctuations about the setpoints. As was the case in Section 4.2.4, both value functions converge shortly after 100 [s]. The angular speeds of the propellers vary in a smooth motion without any high-frequency fluctuations and settle with little oscillation about a



(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s]

Figure 4.10: State responses for suboptimal controllers with random seeds 738 and 930 for a single waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



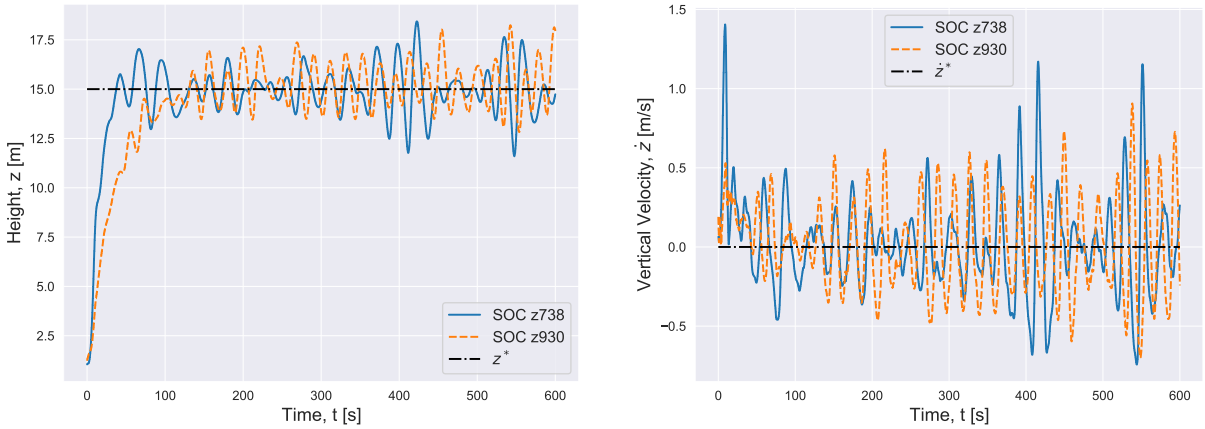
(a) Angular velocity, ω [°/s] (b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 4.11: Actuator signals and value function for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

fixed signal. The SOC z930 controller shows less oscillation at steady-state at the expense of a slower response.

The simulation is repeated with the addition of turbulent wind gusts, simulated using

a Dryden wind model [58]. The results for the vertical position and vertical velocity are shown in Figure 4.12 and the results for the control action and the value function are shown in Figure 4.13.



(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

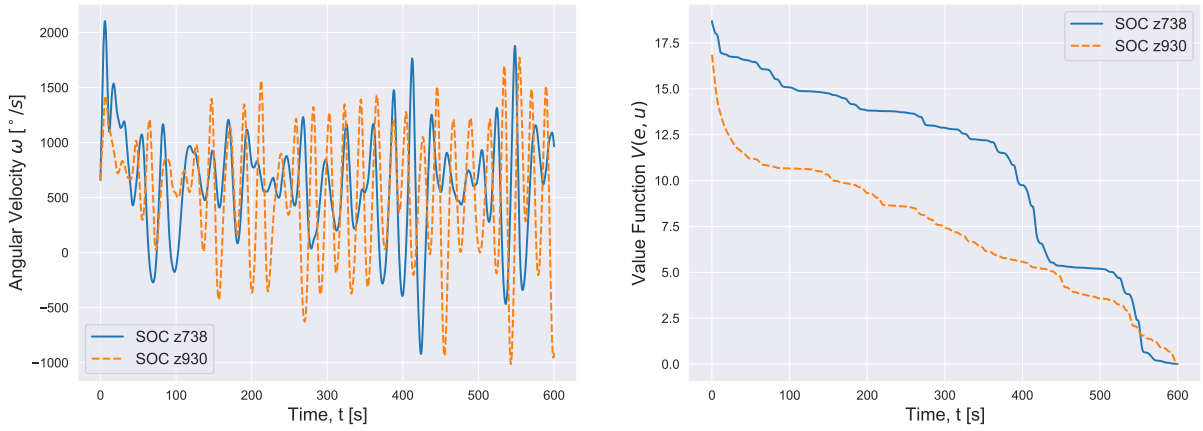
Figure 4.12: State responses for suboptimal controllers with random seeds 738 and 930 for a single waypoint experiment in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

From Figure 4.12 it is observed that the suboptimal controllers are able to hold a close envelope around the target altitude, showing maximum oscillations of approximately ± 2 [m] from the setpoint. Both the SOC z738 and SOC z930 respond desirably with the former reaching the altitude target more slowly, but staying more closely to the vertical velocity reference.

The results from Section 4.2.5, show that the proposed suboptimal controller is capable of controlling the altitude, z [m] and vertical velocity, $\dot{z}$ [m/s] of a Lighter Than Air Vehicles (LTAVs) in both nominal and disturbed single waypoint experiments. In Section 4.2.6, the controllers are tested using multiple waypoints, and are compared to the performance of a FFDL MIMO MFC strategy.

4.2.6 Altitude Controller with Multiple Waypoints

In this section the same two controllers from Section 4.2.5, SOC z738 and SOC z930, are tested in comparison to the FFDL in a multiple waypoint simulation in both nominal and



(a) Angular velocity, ω [°/s]

(b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 4.13: Actuator signals and value function for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

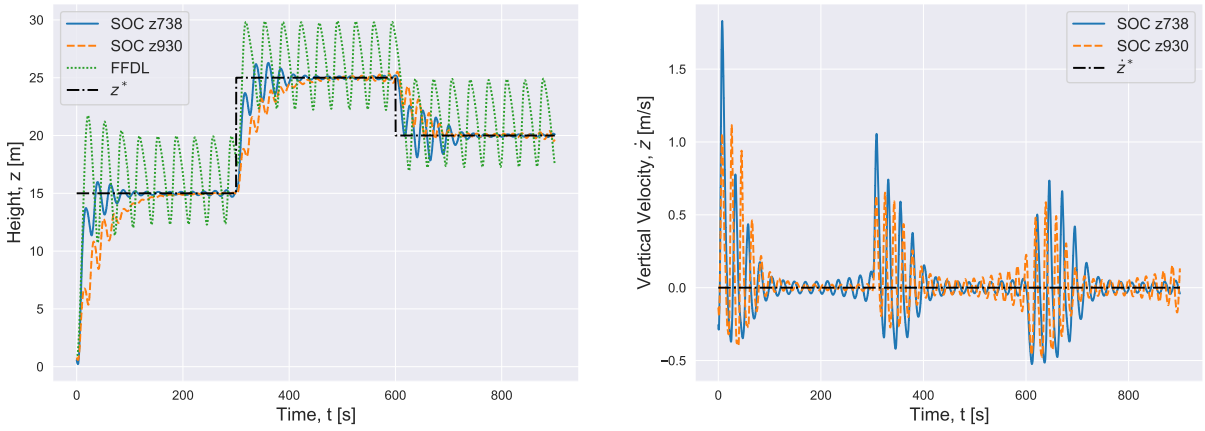
windy environments, to verify the adaptability of the proposed controller. In both tests the target states, $\mathbf{x}^*$, and the maximum values of for the states used in (3.5), $\mathbf{x}^{\max}$, for the controller are,

$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \end{bmatrix} = \begin{bmatrix} T_z \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\max} = \begin{bmatrix} z^{\max} \\ \dot{z}^{\max} \end{bmatrix} = \begin{bmatrix} 22.5 \\ 1.5 \end{bmatrix} \quad \text{for}$$

$$T_z = \begin{cases} 15, & \text{if } t \leq 300 \text{ [s]}, \\ 25, & \text{if } 300 \text{ [s]} < t \leq 600 \text{ [s]} \\ 20, & \text{if } 600 \text{ [s]} < t. \end{cases} \quad \text{and}$$

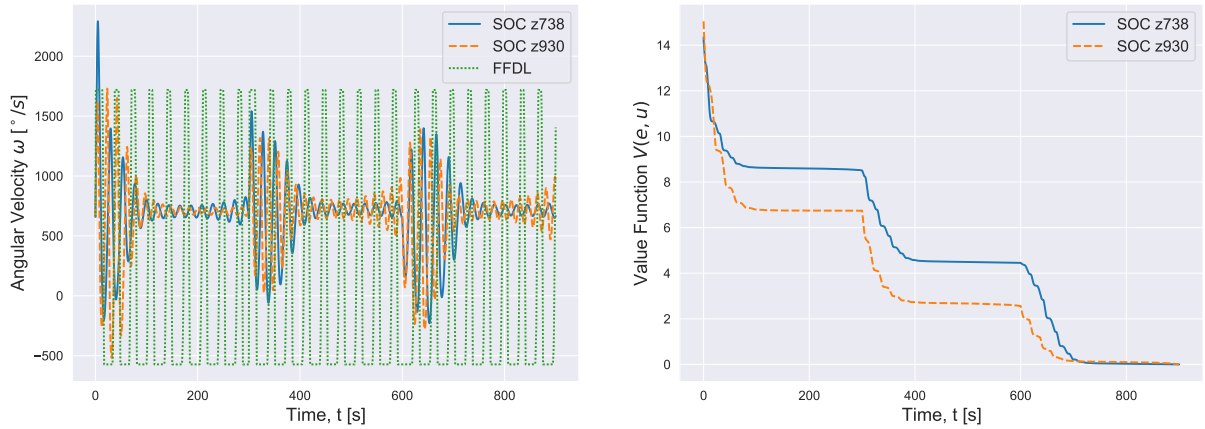
The state responses for the multiple waypoint testing are shown in Figure 4.14 with the actuator signals and the value functions for each controller being shown in Figure 4.15.

Both suboptimal controllers are able to overcome the change in waypoints and track them in a similar way to Figure 4.10, reducing the steady-state error in each of the 300 [s] trajectory segments. The FFDL controller is able to track the changes in waypoints, but is unable to stabilize the error, showing a marginally stable response with oscillations about the desired altitude reference with peak-to-peak value of roughly, 7 [m]. Comparing the actuation signals in Figure 4.15(a), the suboptimal controllers never reach the saturation



(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s]

Figure 4.14: State responses for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



(a) Angular velocity, ω [°/s] (b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 4.15: Actuator signals and value function for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

values of the actuator, while the FFDL controller bounces from one saturation point to the next. The value functions of the suboptimal controllers begin to level off within 100 [s]

of the change in waypoint, which is consistent with the single waypoint results.

The error metrics confirm the results from the plots, with the proposed SOC z738 controller outperforming the FFDL in the ITAE score by a factor of approximately 3.8 and the proposed SOC z930 controller outperforming the FFDL by a factor of 2.5. It is interesting to note that while the SOC z738 controller outperforms the SOC z930 in the error metrics for the altitude, the opposite is true for the vertical velocity. This suggests that the SOC z738 controller is more aggressive in eliminating the altitude error and does so by allowing higher fluctuations of the velocity. The difference in response between the two controllers is reminiscent of under and over damping in the classical PID controller family.

Table 4.3: Error metrics for suboptimal controllers with random seeds 738 and 930 and FFDL for multiple waypoint experiments. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
SOC z738	9520.17	52221542.63
SOC z930	17532.26	78748614.08
FFDL	29234.15	198600419.79

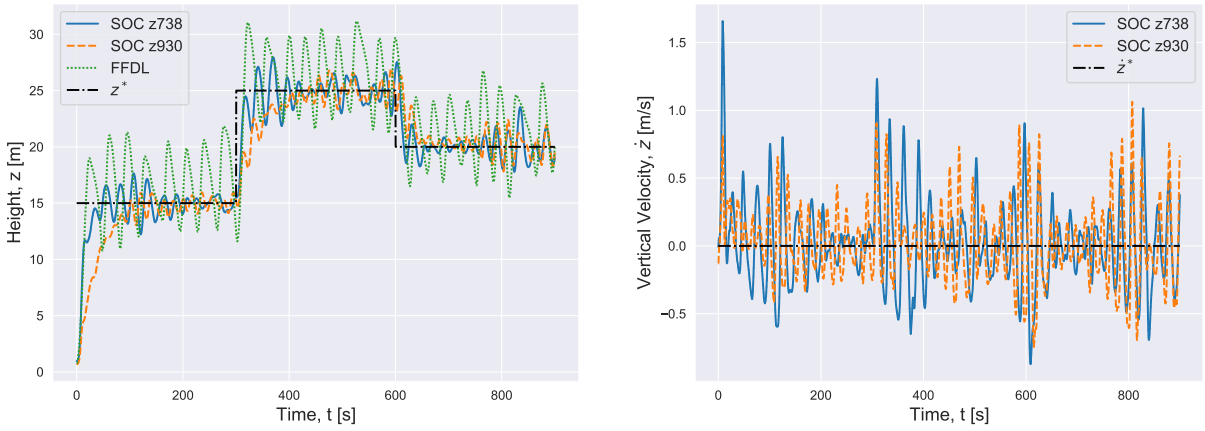
(b) Error metrics for altitude control $\dot{z}$ [m/s]

Controller	IAE	ITAE
SOC z738	1725.08	9638492.92
SOC z930	1471.97	8582524.20

The tests are repeated with the addition of turbulent wind to challenge the controller's ability to reject external disturbances. All the controller configurations and desired waypoints are the same as in the nominal case. The state responses for the multiple waypoint testing in the presence of turbulent wind are shown in Figure 4.16 with the actuator signals and the value functions for each controller being shown in Figure 4.17.

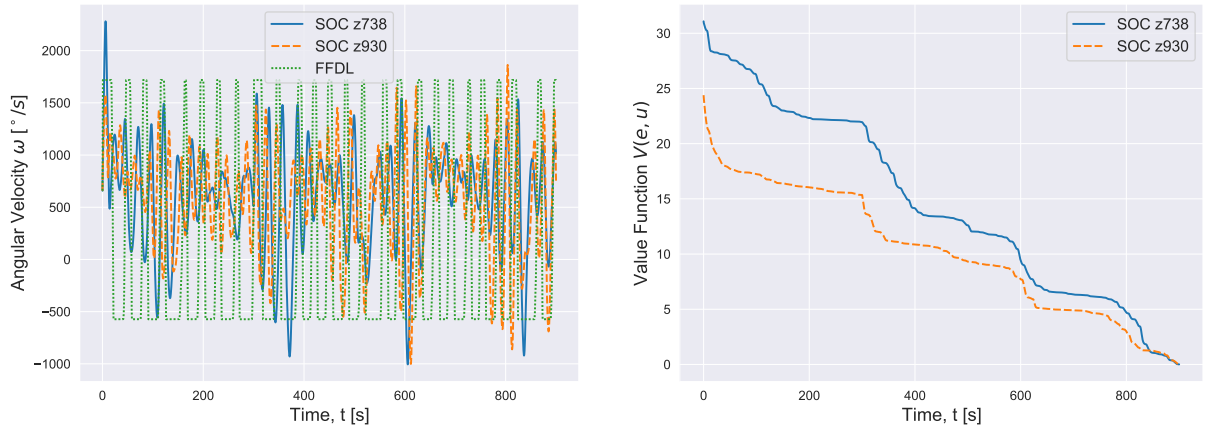
The responses of the two suboptimal controllers are enveloped by the comparison controller. The response times of the controllers seem to be relatively unaffected by the addition of the wind, however, there is a noticeable addition of a high-frequency component to the states and the actuator signals. It is also observed that the value function no longer converges, which makes intuitive sense, as the error is never fully eliminated. As was the case in the nominal experiment, the actuator commands of the FFDL controller toggled between saturation values without much time away from the extrema. Conversely, the actuator responses of the suboptimal controllers never appear to reach the limits and are also clearly enveloped by the comparison response.

The error metrics are tabulated in Table 4.4. Again the proposed suboptimal controllers demonstrate adaptability to changing waypoints and robustness to external disturbances.



(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s]

Figure 4.16: State responses for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



(a) Angular velocity, ω [°/s] (b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 4.17: Actuator signals and value function for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

In all cases, both suboptimal controllers produce lower IAE and ITAE scores than the comparison algorithm.

Table 4.4: Error metrics for suboptimal controllers with random seeds 738 and 930 and FFDL for multiple waypoint experiments in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
SOC z738	16623.48	109927863.10
SOC z930	21993.40	123361645.56
FFDL	31134.09	212580242.89

(b) Error metrics for altitude control $\dot{z}$ [m/s]

Controller	IAE	ITAE
SOC z738	3300.78	21707872.12
SOC z930	2971.90	21941284.24

4.2.7 Summary of Altitude Control

These results work towards the validation of the single network optimal controller developed in Section 4.1. It is demonstrated that the proposed algorithm can successfully control the altitude of a dirigible, a nonlinear time-delayed system, in a reduced dimensional control problem without a priori knowledge of the dynamics. The algorithm showed adaptability and robustness, outperforming a modern state-of-the-art MFC strategy by every considered metric. In addition to the satisfactory results observed by the controller, the ablation study demonstrated that the implementation of the suboptimal controller was able to stabilize the system response for five out of six randomly initialized actor weights with very little sensitivity to the learning rate parameter, given it was selected to be small enough. In the next section, this work is extended by testing the algorithm in a higher dimensional experimental test setting by controlling the longitudinal velocity as well as the altitude.

4.3 Altitude and Velocity Control with Single Network Controller

This section extends the complexity of the altitude control experiments to a 2-input 4-output control task for the altitude, z [m], vertical velocity, $\dot{z}$ [m/s], longitudinal velocity u [m/s] and longitudinal acceleration $\dot{u}$ [m/s^2], of the airship model described in Section 4.2.1. The controller parameters for the suboptimal controller are introduced in Section 4.3.1 and the updated parameters for the comparison, FFDL controller are given in Section 4.3.2 with the initial conditions and implementation restrictions being discussed.

The structure of this section is similar to that of Section 4.2, with an ablation study being performed over a range of initial critic values and then selecting two specific con-

trollers to investigate more rigorously in Section 4.3.3. Section 4.3.4 extend the work in Section 4.3.3, to multiple waypoint tests to investigate the adaptability of the proposed algorithms. A summary of the findings is offered in Section 4.3.5.

4.3.1 Altitude and Velocity Controller Implementation

The work in the previous section is augmented by increasing the dimension of the control task to 2-input 4-ouputs. In this task, the vertical position, z [m], vertical velocity, $\dot{z}$ [m/s], longitudinal velocity u [m/s] and longitudinal acceleration $\dot{u}$ [m/s^2] are controlled by regulating the angular velocity of the thrusters, ω [°/s] and the vector thrusting angle, γ [°]. The states, shown in Figure 4.18, are given by

$$\mathbf{x} = \begin{bmatrix} z \\ \dot{z} \\ u \\ \dot{u} \end{bmatrix} \in \mathbb{R}^4.$$

The angular velocities of the propellers, shown in Figure 4.19, of the left and right propellers are equal in magnitude and opposite in direction, causing a longitudinal thrust, and are again saturated with maximum and minimum values of ± 2864.8 [°/s]

$$\omega_k = \omega_{k,r} = -\omega_{k,l} \in [-2864.8, 2864.8]°/\text{s}$$

and have an initial value of $\omega_0 = 658.9$ [°/s] in all experiments performed. This corresponds to an initial, upwards, thrust vector acting along the positive z-axis. The vector thrust actuator is the angular position of the thrust vector with respect to the vertical as shown in Figure 4.19

$$\gamma_k \in [0, 90] \text{ [°]}$$

For all experiments the matrix $\mathbf{Q} \in \mathbb{R}^{\bar{m}}$ is held constant as a symmetric positive definite matrix whose element for the i^{th} row and j^{th} column are

$$q_{i,j} = \begin{cases} 10^{-1} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}, \quad \forall (i,j) \in \{1, 2, \ldots, n\}^2.$$

Matrix $\mathbf{R} \in \mathbb{R}^m$ is also held constant as a symmetric positive definite matrix whose

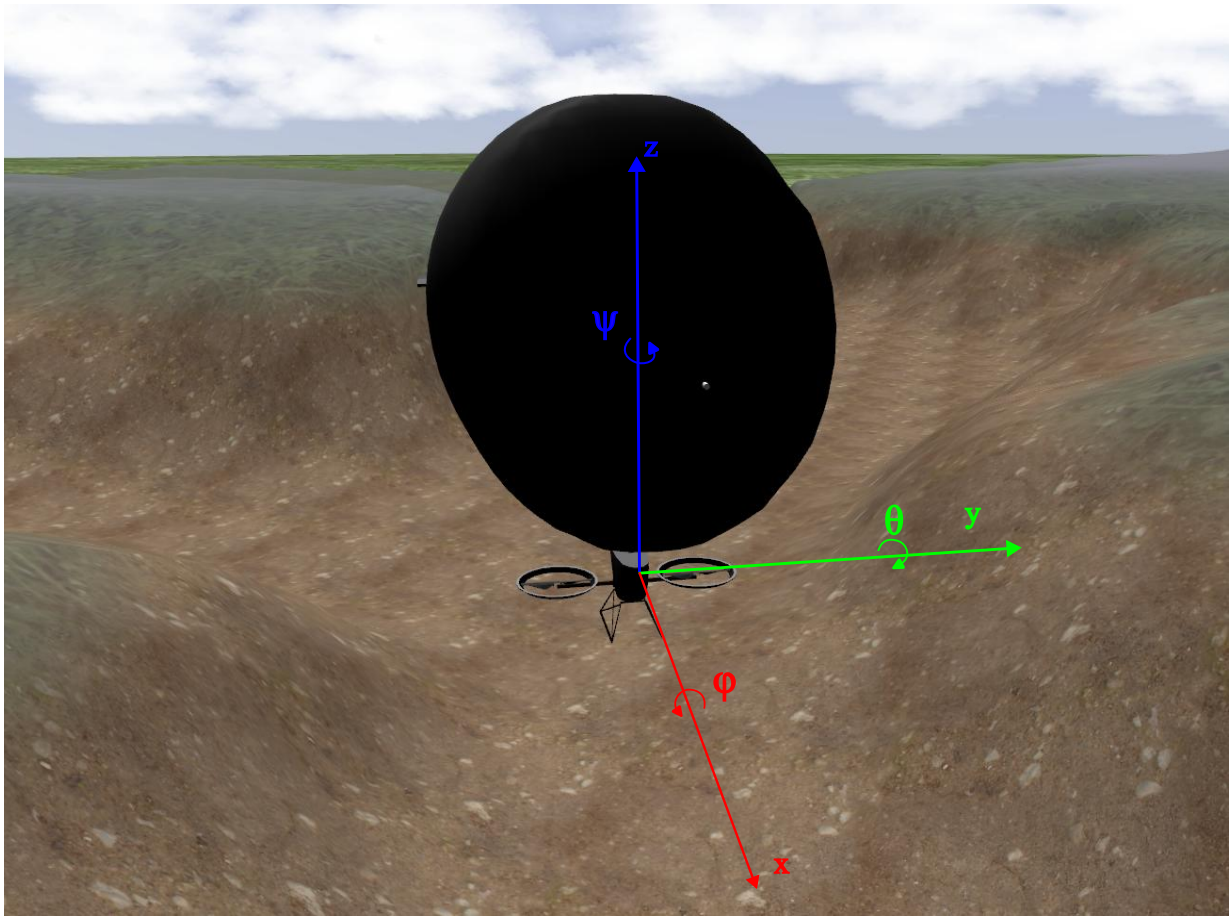


Figure 4.18: Vertical and longitudinal axes definitions. Note that z and $\dot{z}$ are defined with respect to the World reference frame, whereas u and $\dot{u}$ are defined on the vehicles body frame.

whose element for the i^{th} row and j^{th} column are

$$r_{i,j} = \begin{cases} 10^{-1} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}, \quad \forall (i,j) \in \{1, 2, \ldots, n\}^2.$$

The position, velocity and acceleration readings from the dirigible simulation are all updated with sampling $f_{\text{sample}} = 150$ [Hz]. Along with the altitude measurements, the longitudinal velocity at time k, u_k [m/s] is read from the "/blimp/ground_truth/odometry" topic while the longitudinal acceleration at time k,

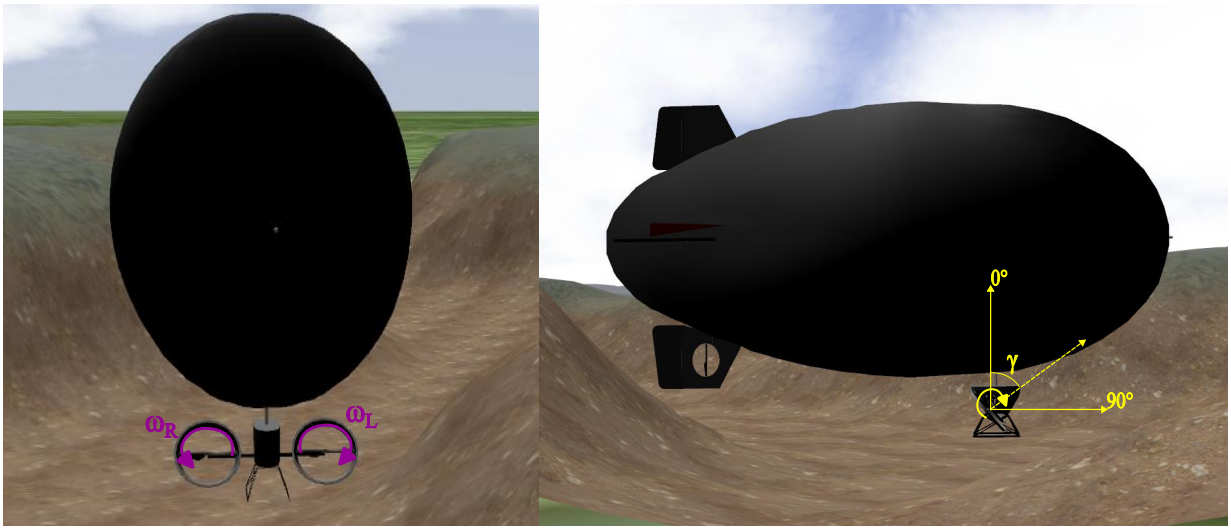


Figure 4.19: Angular velocities for the left, ω_L, and right, ω_R propellers of the airship, where $\omega_{k,l} = -\omega_{k,r} = \omega_k$ [°/s].

$\dot{u}_k$ [m/s^2] is measured from the "/blimp/ground_truth/imu" topic. The controller script creates a node called "/alt_vel_controller" in the "/blimp" namespace that updates a list of motor speeds as Actuator messages and vector thrust angles as Float64 messages that are published to the airship over the "/blimp/command/motor_speed" and "/blimp/stick_joint_position_controller/command" topics, respectively, with frequency $f_{\text{publish}} = 15$ [Hz]. The rqt graph for the altitude controller is shown in Figure 4.20.

4.3.2 Altitude and Velocity Controller Implementation with Comparison Controller

The comparison controller that is used for the altitude and velocity control task is the same one introduced in Section 4.2.3 with slightly different parameters, and modifications to the dimensions of the variables as required. For readability, only the parameters are quoted here with the reader encouraged to look at Section 4.2.3, for application details.

The values that are used for the resetting algorithm are, $a = 10^{-3}$, $b_1 = 5$ and $b_2 = 10^{-2}$. The initial value of the PPJM is chosen to be

$$\hat{\mathbf{\Phi}}_0^{\text{ffdl}} = \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}.$$

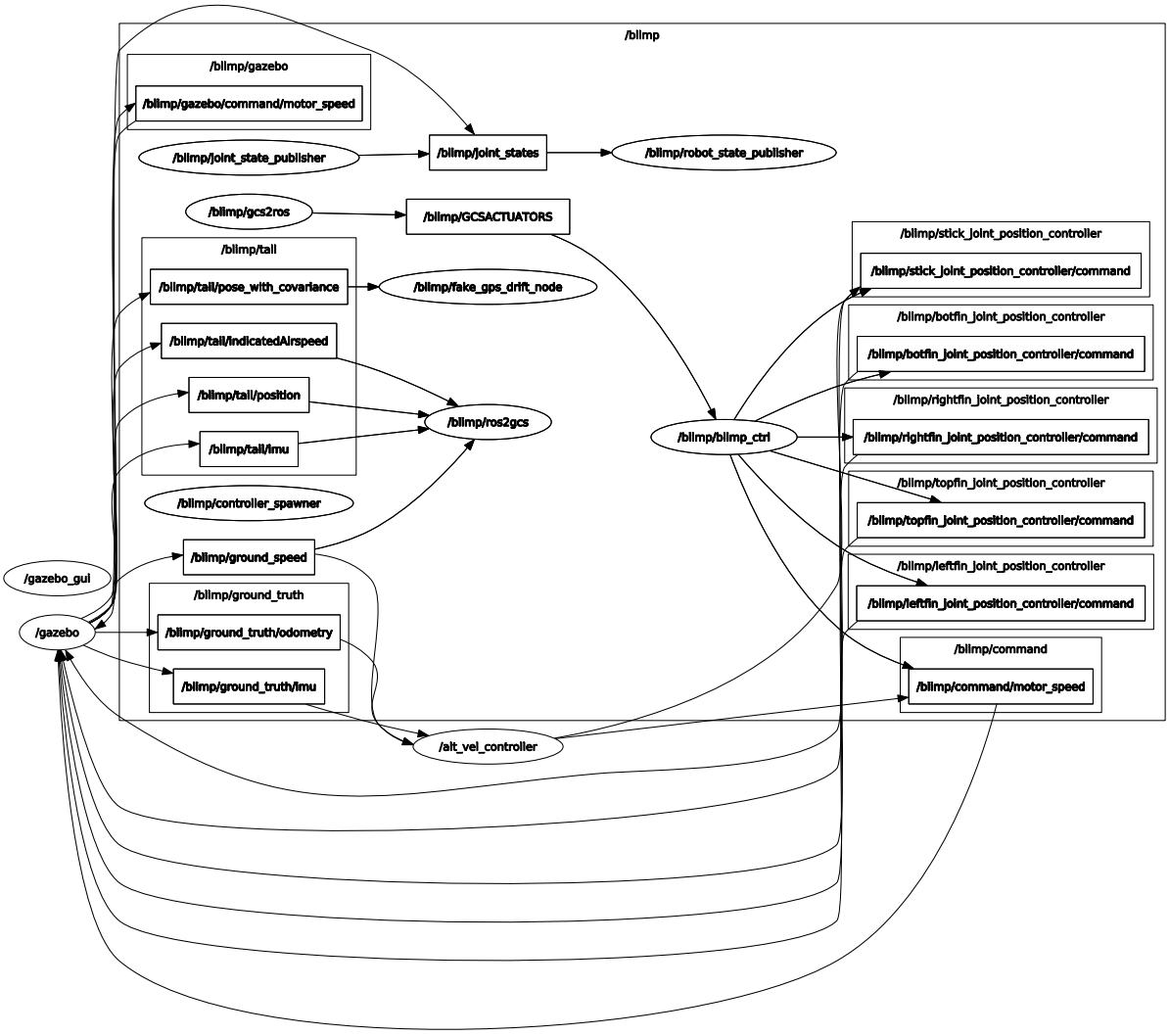


Figure 4.20: RQT graph for the proposed altitude and velocity controller.

The same comments from the altitude controller in Section 4.2.3 apply to the altitude and velocity comparison controller. It is noted again that in order to avoid divergence in the FFDL controller, the actuator signals had to be restricted to $\omega^{\text{ffdl}} \in [-572.96, 1718.87]$ [°/s], however the range of the vector thrust angle, γ is equal to that of the proposed suboptimal

Table 4.5: Parameter values for the FFDL comparison controller for the altitude control task.

Parameter	Value
η	1
μ	1
ρ	5×10^{-1}
λ	10^{-2}
L_y	0
L_u	1

controller.

In the following section, the suboptimal controller is implemented for altitude and velocity control of the airship by controlling the magnitude and direction of the thrust generated by the propellers. Both single and multiple waypoint experiments are conducted with and without the presence of turbulent wind, with the multiple waypoint experiments being compared to the FFDL controller.

4.3.3 Altitude and Velocity Controller with Single Waypoint

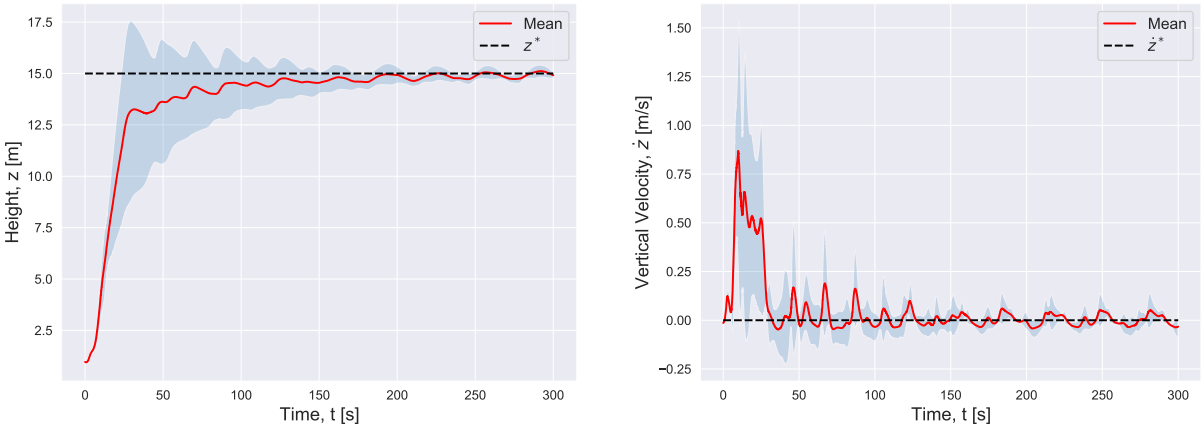
In this section an ablation study is performed on the suboptimal controller using different initial values of the critic weights $\mathbf{\Omega}_c^{[0]}$ to determine the sensitivity of the altitude and velocity controller with respect to the initial weights. The values of the five critic weights are randomly initialized using random seeding. The values are given in Appendix A.2 for completeness. As was the case in Section 4.2.5 six initial values are attempted, however, one of them diverged during the experiments, so it is excluded from the study as an outlier. The target and maximum values of the states in this experiment are

$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \\ u^* \\ \dot{u}^* \end{bmatrix} = \begin{bmatrix} 15 \\ 0 \\ 2 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\max} = \begin{bmatrix} z^{\max} \\ \dot{z}^{\max} \\ u^{\max} \\ \dot{u}^{\max} \end{bmatrix} = \begin{bmatrix} 7.5 \\ 1.5 \\ 2.5 \\ 0.5 \end{bmatrix} \tag{4.20}$$

Notations 4.2. *Modifying the notation from the previous section, a controller "SOC zu###" refers to the suboptimal controller for states in (4.20), with random seed ###.*

The results of the ablation study over the different random seeds are plotted in Figures 4.21 and 4.22 with the solid red line indicating the mean of each controller for a given

time interval and the shaded blue area indicating a vertical range of $\pm$ 1 standard deviation from the mean.



(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

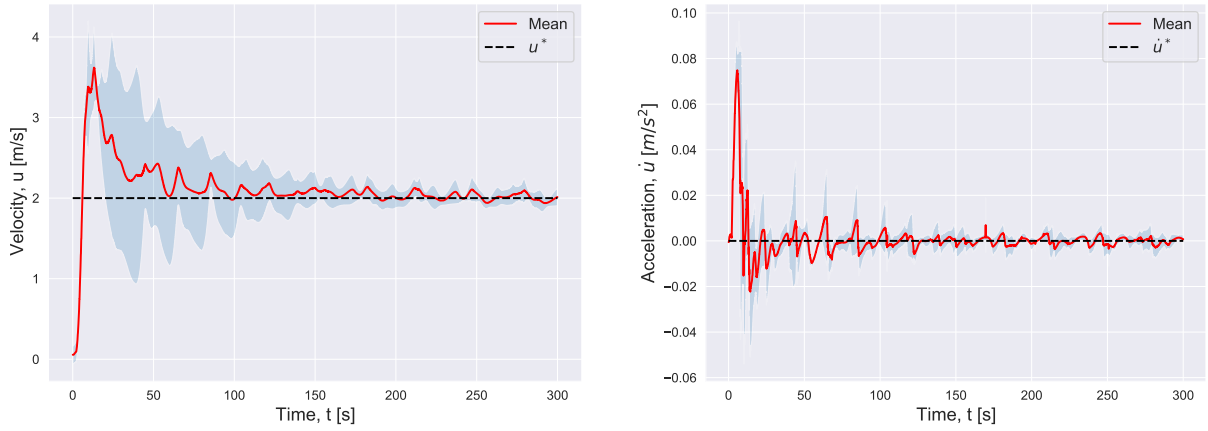
Figure 4.21: Ablation study for altitude control, varying the initialization of the critic weights $\mathbf{\Omega}_c^{[0]}$ for a single waypoint. The red line indicates the mean response and the shaded area represents a vertical width of one standard deviation from the mean system response. (a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s].

It is reassuring to see that in all cases, the controller tracks the desired references, with the standard deviation envelope closing in towards the mean. This shows that while the learning phase of the controller is sensitive to the initialization of the critic weights, the tracking performance for $t > 200$ [s] seems to vary only slightly. Once again, the fact that the algorithm produced stabilizing outputs for five of the six initial critic weights show promise in easily selecting an admissible initial value.

To visualize the response of two of the controllers, SOC zu392 and SOC zu738, the state responses are plotted in Figures 4.23 and 4.24.

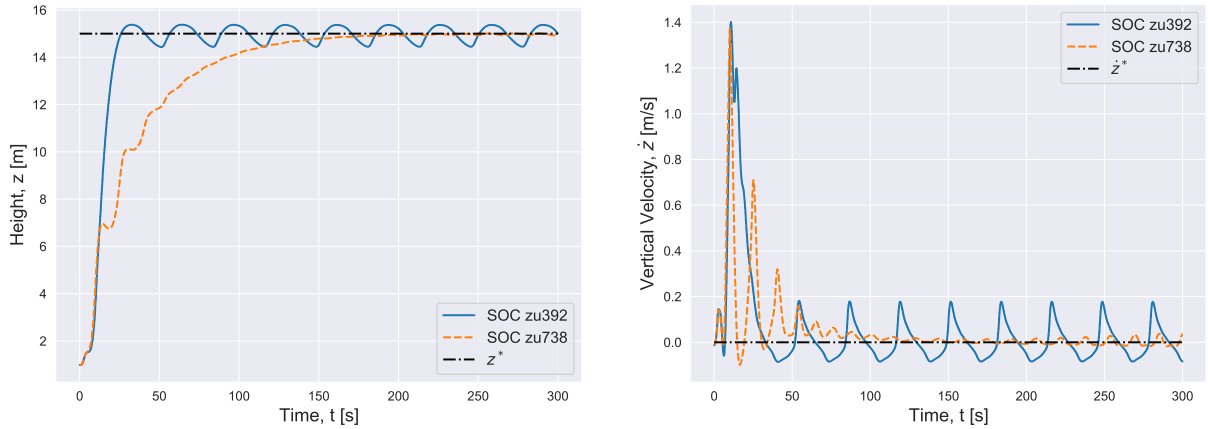
For all of the state control tasks, the SOC zu392 controller more aggressively reduces the state errors, reaching the reference targets first, however this controller in turn has higher oscillation amplitudes about the set points. Conversely, the SOC zu738 controller is slower to achieve the desired setpoint, but has a more stable steady-state response. The same attributes are noticed in the actuator responses, Figure 4.25. The SOC zu392 controller has larger oscillations in the actuator signals than the SOC zu738.

The value functions, Figure 4.26, for the two proposed suboptimal controllers, both show a steep decline in the initial phase. The value function for the SOC zu392 controller



(a) Longitudinal Velocity, u [m/s] (b) Longitudinal Acceleration, $\dot{u}$ [m/s^2]

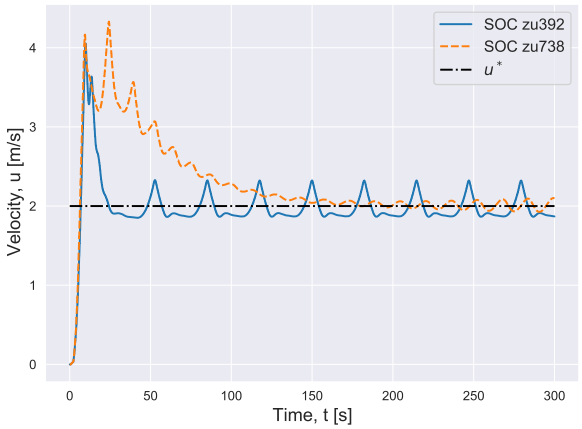
Figure 4.22: Ablation study for altitude control, varying the initialization of the critic weights $\mathbf{\Omega}_c^{[0]}$ for a single waypoint. The red line indicates the mean response and the shaded area represents a vertical width of one standard deviation from the mean system response. (a) Longitudinal Velocity, u [m/s] (b) Longitudinal Acceleration, $\dot{u}$ [m/s^2].



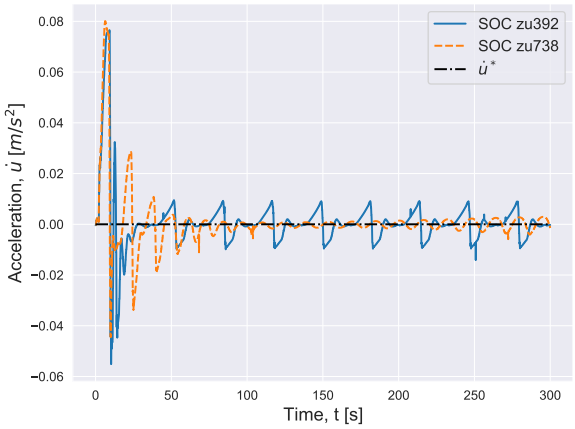
(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s]

Figure 4.23: State responses for suboptimal controllers with random seeds 392 and 738 for a single waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

initially declines more quickly than its counterpart, before slowly levelling off, without ever converging. The value function for the SOC zu738 controller, however, declines steadily for

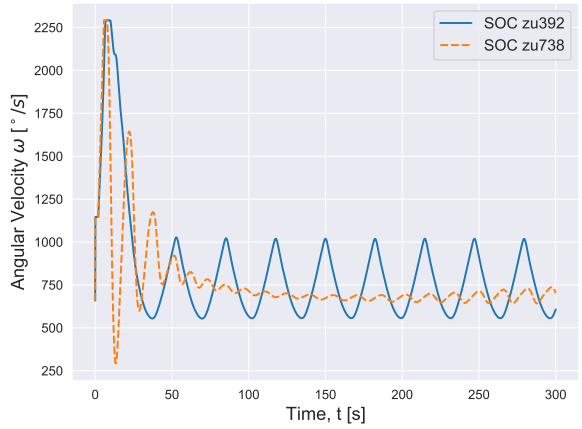


(a) Longitudinal Velocity, u [m/s]

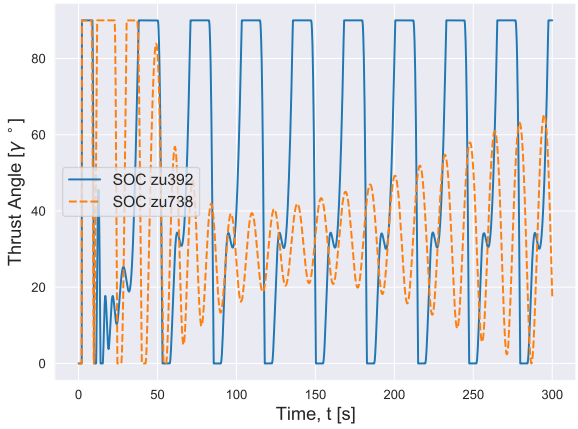


Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 4.24: State responses for suboptimal controllers with random seeds 392 and 738 for a single waypoint experiment. (a) longitudinal velocity u [m/s] and (a) longitudinal acceleration $\dot{u}$ [m/s^2].



(a) Angular velocity, ω [°/s]



(b) Thrust angle, γ [°]

Figure 4.25: Actuator signals for suboptimal controllers with random seeds 392 and 738 for a single waypoint experiment. (a) angular speed, ω [°/s] and (b) thrust angle, γ [°].

the first 75 [s] before converging smoothly around the 100 [s] mark. The lack of convergence in the SOC zu930 is likely correlated to the oscillations observed in the state responses.

The nominal single waypoint altitude and velocity control experiments are repeated in

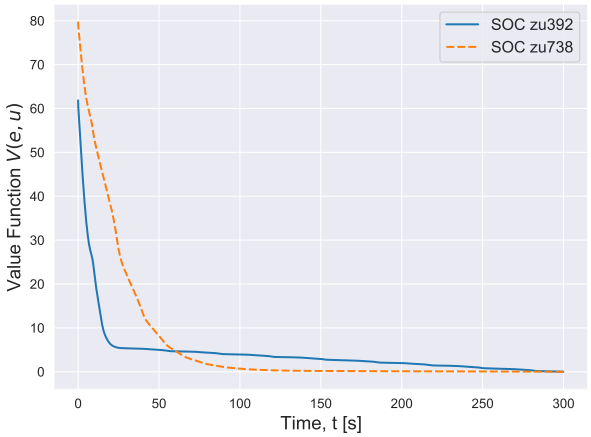
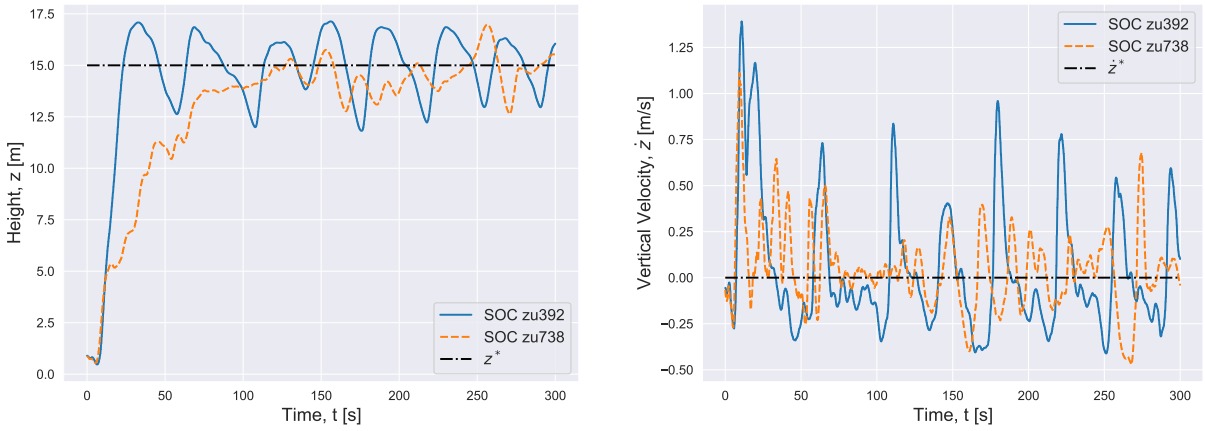


Figure 4.26: Value function for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment.

the presence of turbulent wind to verify the robustness of the proposed algorithms. The state responses from these tests are shown in Figure 4.27.

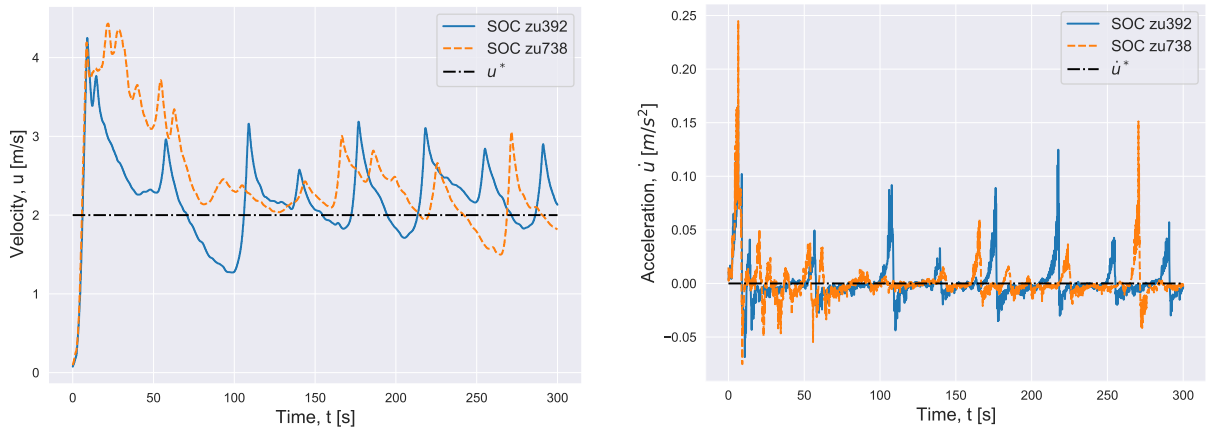


(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

Figure 4.27: State responses for suboptimal controllers with random seeds 392 and 738 for a single waypoint experiment in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

Firstly, it is noted that in Figures 4.27 and 4.28, both controllers are able to successfully track the desired setpoints and oscillate about them, despite the disturbance. The SOC



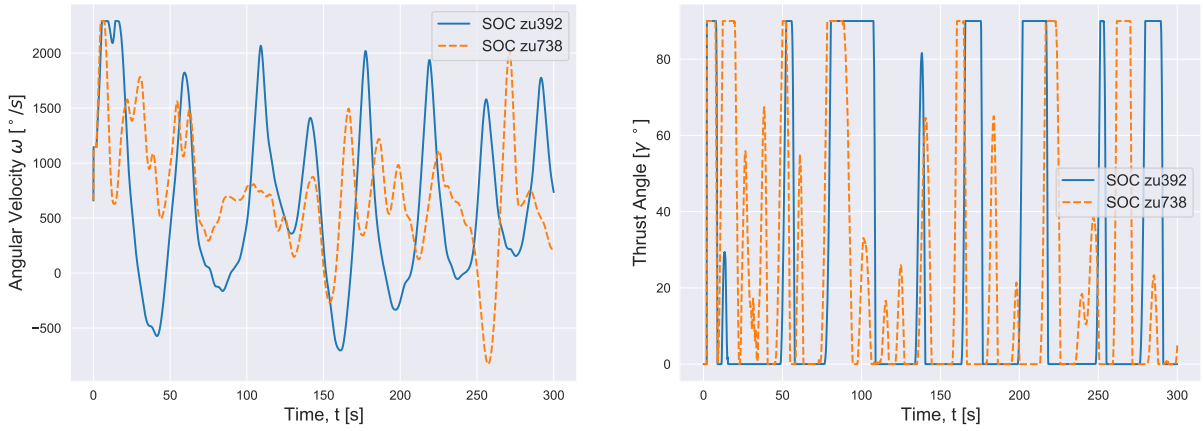
(a) Longitudinal Velocity, u [m/s]

Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 4.28: State responses for suboptimal controllers with random seeds 392 and 738 for a single waypoint experiment in the presence of turbulent wind. (a) longitudinal velocity u [m/s] and (a) longitudinal acceleration $\dot{u}$ [m/s^2].

zu392 controller is again more aggressive in tracking the errors, leading to faster tracking times, in exchange for higher amplitude oscillations. Similar behaviour is observed in Figure 4.29. The SOC zu392 controller shows larger peak-to-peak amplitudes that the SOC zu738 controller for the angular velocities of the propellers, Figure 4.29(a), and spends more time in the extrema of the thrust vector control, Figure 4.29(b), whereas the SOC zu738 is capable of controlling the system with values outside of the saturation values.

The value functions for the two proposed suboptimal controllers are shown in Figure 4.30. Whilst neither converge, the response of the SOC zu738 controller shows a steep decline in the first 60 [s] range, followed by a relatively smooth decline for the remainder of the experiment. Both of the studied controllers show promising results for the single waypoiont altitude and velocity control task in the nominal and disturbed test cases. Both controllers successfully tracked the desired waypoints for all states while displaying smooth actuator signals that don't show any signs of chattering. To better understand the characteristics of the contrrollers, a multiple waypoint experiment is performed in both the nominal and disturbed scenarios and a comparison is made to the FFDL controller.



(a) Angular velocity, ω [°/s]

(b) Thrust angle, γ [°]

Figure 4.29: Actuator signals for suboptimal controllers with random seeds 392 and 738 for a single waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) thrust angle, γ [°].

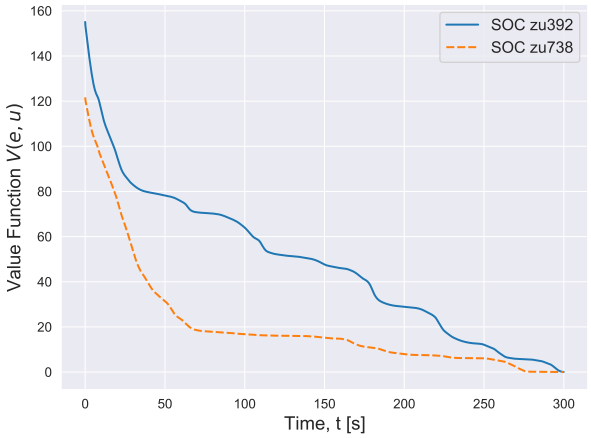


Figure 4.30: Value function for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment.

4.3.4 Altitude and Velocity Controller with Multiple Waypoints

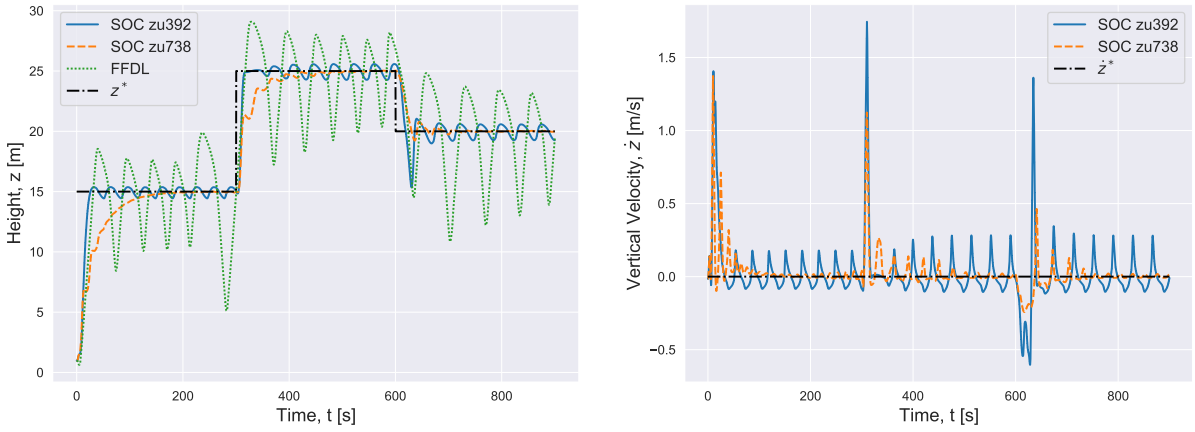
This section investigates the adaptability of the proposed algorithm by adding to the complexity of the implementation from Section 4.3.3. The same two controllers, SOC

zu392 and SOC zu738, are tested in comparison to the FFDL in a multiple waypoint simulation in both nominal and windy environments. In all cases the target states, $\mathbf{x}^*$, and the maximum values of for the states used in (3.5), $\mathbf{x}^{\max}$, for the controller are

$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \\ u^* \\ \dot{u}^* \end{bmatrix} = \begin{bmatrix} T_z \\ 0 \\ 2 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\max} = \begin{bmatrix} z^{\max} \\ \dot{z}^{\max} \\ u^{\max} \\ \dot{u}^{\max} \end{bmatrix} = \begin{bmatrix} 7.5 \\ 1.5 \\ 2.5 \\ 0.5 \end{bmatrix} \quad \text{for}$$

$$T_z = \begin{cases} 15, & \text{if } t \leq 300 \text{ [s]}, \\ 25, & \text{if } 300 \text{ [s]} < t \leq 600 \text{ [s]} \\ 20, & \text{if } 600 \text{ [s]} < t. \end{cases} \quad \text{and}$$

The state responses for the nominal experiment are shown in Figures 4.31 and 4.32, with the FFDL being used for a comparison controller.

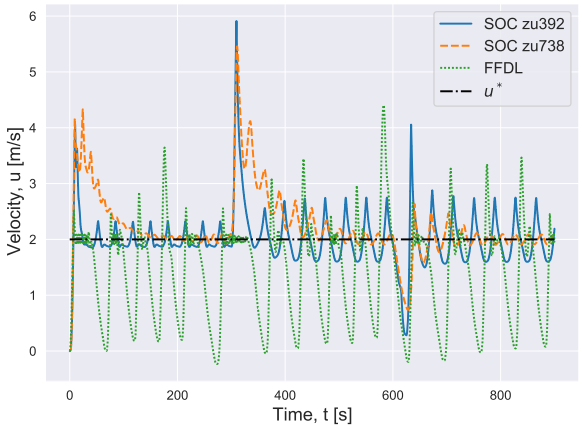


(a) Vertical position, z [m]

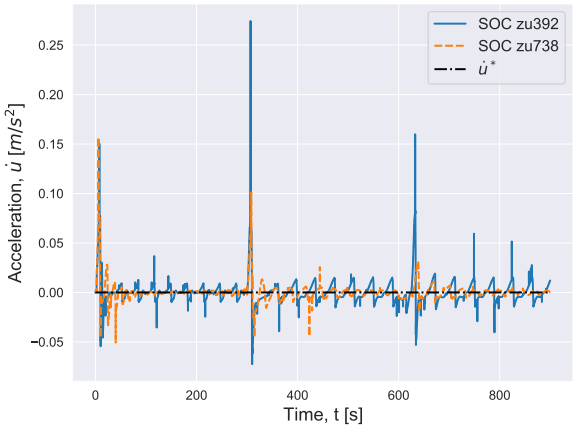
(b) Vertical velocity, $\dot{z}$

Figure 4.31: State responses for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

The trajectory plots for the nominal multiple waypoint experiment show the two suboptimal controllers successfully tracking the desired setpoints with the SOC zu738 controller having a slower response but showing less oscillation for all four feedback states, similarly



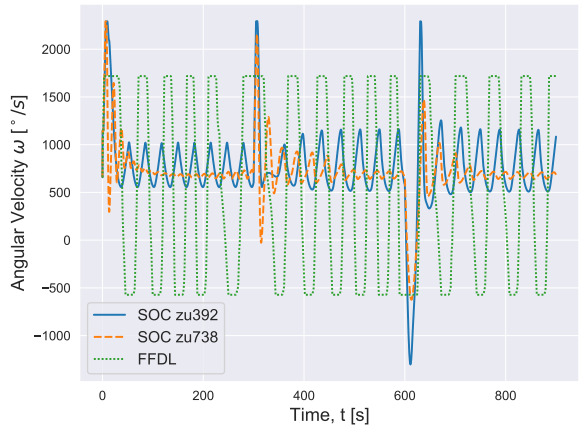
(a) Longitudinal Velocity, u [m/s]



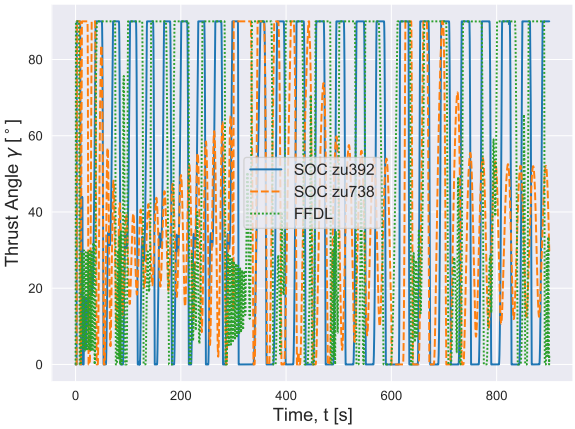
(b) Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 4.32: State responses for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment. (a) longitudinal velocity u [m/s] and (b) longitudinal acceleration $\dot{u}$ [m/s^2].

to the single waypoint simulations. The comparison controller is also able to track the desired setpoints but does so at the expense of high-amplitude oscillations.



(a) Angular velocity, ω [°/s]



(b) Thrust angle, γ [°]

Figure 4.33: Actuator signals and value function for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) thrust angle, γ [°].

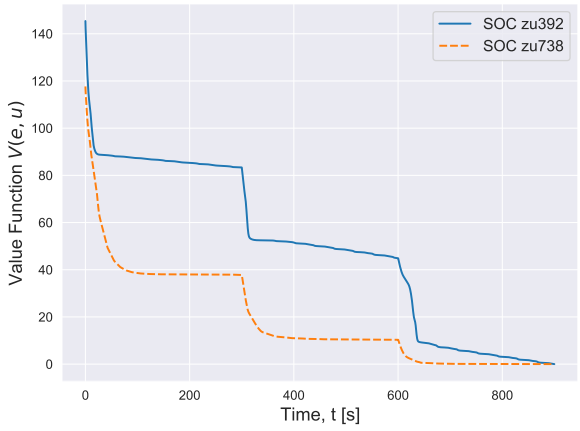


Figure 4.34: Value function for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment.

From Figure 4.33, it is observed that the SOC zu738 controller shows much less fluctuation than either of the other controllers, having a lower rate of change in the actuator signal. Additionally, the SOC zu738 controller tends to converge in the angular velocity signal in the time interval of constant altitude. Figure 4.34 shows the value function of the SOC zu392 and SOC zu738 controllers. It is observed that the time it takes for the value function of the SOC zu738 controller to level off is less than 100 [s], which shows a quicker response than what was observed in the altitude controller in Section 4.2.4.

The IAE and ITAE metrics for the four state control tasks are tabulated in Table 4.6. From these data two interesting observations are made, the first being that both of the proposed SOC controllers outperform the FFDL controller in every metric. The second observation that is drawn is that the SOC zu738 controller outperforms both others in the ITAE metric for all experiments, despite being outscored in the IAE category for the altitude and velocity states. This reinforces the qualitative observation that while the SOC zu392 is capable of addressing the trajectory error more quickly than the SOC zu738, the latter can more readily reduce the steady-state error of the system.

The multiple waypoint experiment is repeated using the same three controllers, target waypoints and maximum values as in the nominal case to investigate the noise rejection abilities of the controllers. The state responses for the airship in the presence of turbulent wind are shown in Figures 4.35 and 4.36.

The altitude response of the two proposed suboptimal controllers qualitatively appears to track the desired setpoints more closely, with their responses being enveloped by the

Table 4.6: Error metrics for suboptimal controllers with random seeds 392, 738 and FFDL for multiple waypoint experiments. (a) vertical position z [m], (b) vertical velocity $\dot{z}$ [m/s], (c) longitudinal velocity u [m/s] and (d) longitudinal acceleration $\dot{u}$ [m/s^2].

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
SOC zu392	10714.24	67480593.43
SOC zu738	13086.45	47549997.10
FFDL	32970.12	227937040.44

(b) Error metrics for vertical velocity control $\dot{z}$ [m/s]

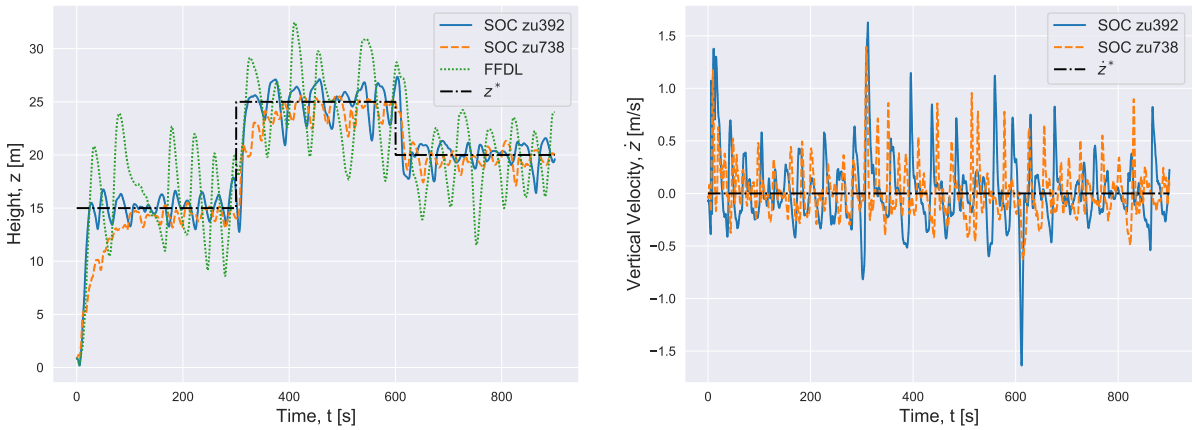
Controller	IAE	ITAE
SOC zu392	1437.32	9095317.34
SOC zu738	665.96	3355108.55

(c) Error metrics for longitudinal velocity control u [m/s]

Controller	IAE	ITAE
SOC zu392	4329.93	36044176.41
SOC zu738	4774.32	25866692.65
FFDL	8103.60	61605695.66

(d) Error metrics for longitudinal acceleration control $\dot{u}$ [m/s^2]

Controller	IAE	ITAE
SOC zu392	92.90	631104.52
SOC zu738	57.65	315721.23

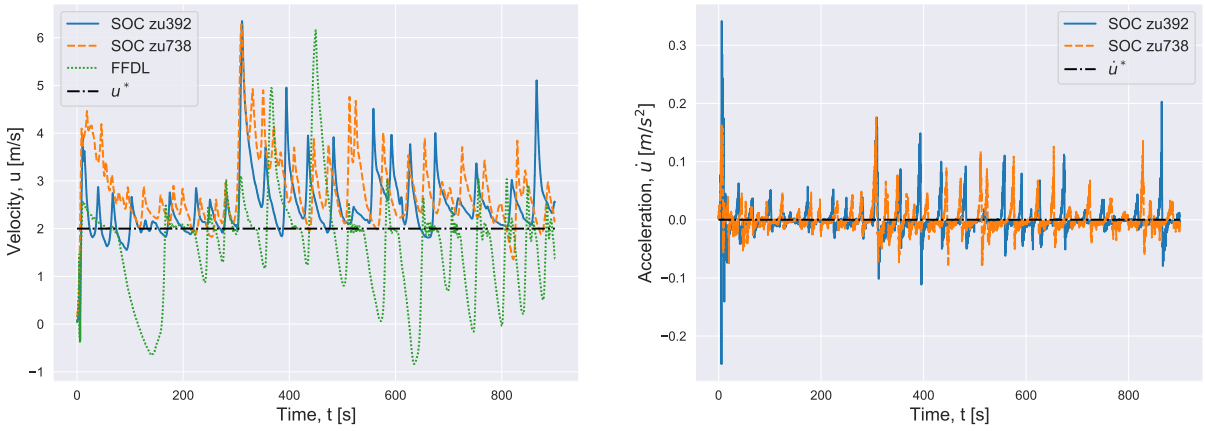


(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

Figure 4.35: State responses for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

response of the comparison controller. It is noted that the error of the comparison controller once it has tracked the desired setpoint, seems to reach a peak absolute value close to 10 [m], whereas the peak absolute error of the two SOC controllers is less than 5 [m]. Looking at the longitudinal velocity response in Figure 4.36(a), all three controllers show high-frequency



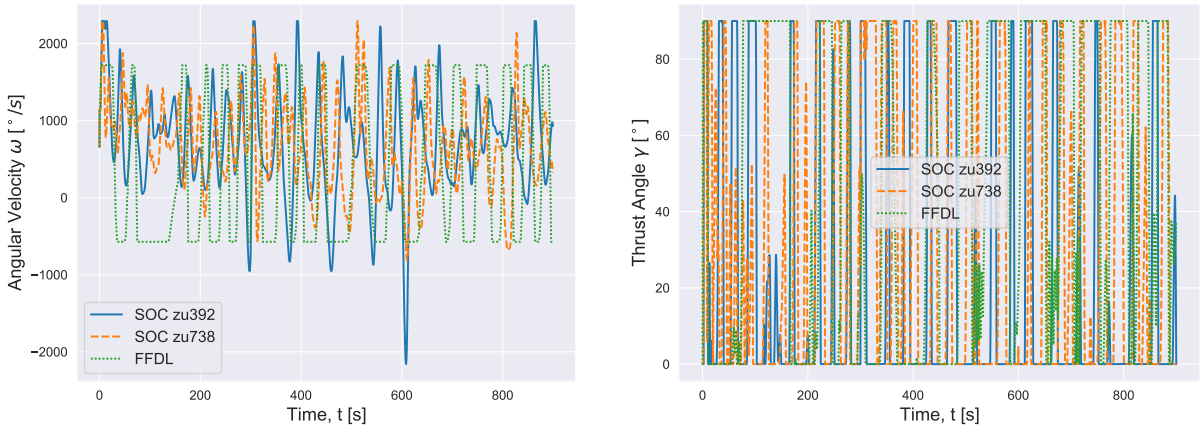
(a) Longitudinal Velocity, u [m/s] (b) Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 4.36: State responses for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment in the presence of turbulent wind. (a) longitudinal velocity u [m/s] and (b) longitudinal acceleration $\dot{u}$ [m/s^2].

fluctuations, with the positive peaks in the velocity control happening around 0 [s] and 300 [s], which correspond to the two step changes in the altitude targets. It is interesting to note that the two proposed controllers seem to be biased towards positive errors in the longitudinal velocity response, while the comparison controller is biased towards a negative error. In fact the FFDL controller approaches or even achieves, negative velocities on five occasions.

From the actuator responses in Figure 4.37, it is clear that the actuators have to work significantly harder to achieve the target outputs when compared to the nominal case. High frequency components are observed in both actuator signals in the proposed controllers, with the vector thrust control tending to saturate for all three controllers. In Figure 4.37(a) it is observed that for most of the experiment, the angular velocities fo the proposed controllers are, again, sandwiched between the angular velocity of the FFDL controller, with a few brief exceptions being noted. It is also observed that neither of the proposed controllers reach the saturation limits of the angular velocity actuators for extended periods of time, and in fact, the SOC zu738 controller doesn't appear to reach it at all.

From the error metrics in Table 4.7, both SOC controllers outperform the FFDL controller in the altitude and longitudinal velocity control tasks for both the IAE and ITAE metrics. In the disturbed case, the error metrics of the proposed suboptimal controllers



(a) Angular velocity, ω [°/s] (b) Thrust angle, γ [°]

Figure 4.37: Actuator signals and value function for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) thrust angle, γ [°].

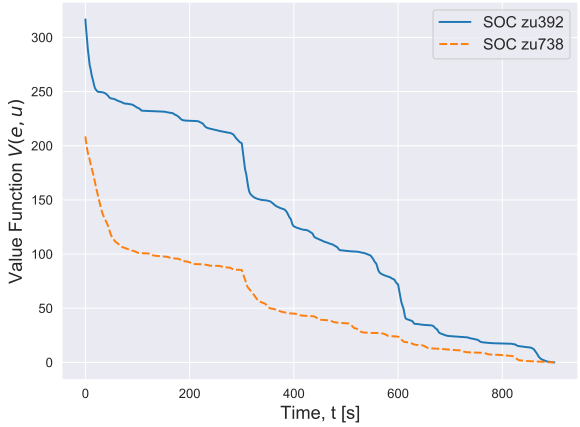


Figure 4.38: Value function for suboptimal controllers with random seeds 392 and 738 for a multiple waypoint experiment in the presence of turbulent wind.

are relatively comparable, with all the scores being fairly close.

During the experimental testing, the time it takes for the control computations to be completed is calculated and stored to ensure that the real-time capabilities are being respected. The times for the multiple waypoint nominal and windy control loop calculations

Table 4.7: Error metrics for suboptimal controllers with random seeds 392, 738 and FFDL for multiple waypoint experiments in the presence of turbulent wind. (a) vertical position z [m], (b) vertical velocity $\dot{z}$ [m/s], (c) longitudinal velocity u [m] and (d) longitudinal acceleration $\dot{u}$ [m/s^2].

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
SOC zu392	18272.60	119307638.60
SOC zu738	20846.64	105978791.30
FFDL	34864.63	229404076.05

(b) Error metrics for vertical velocity control $\dot{z}$ [m/s]

Controller	IAE	ITAE
SOC zu392	3120.92	19144023.62
SOC zu738	2384.43	15656360.47

(c) Error metrics for longitudinal velocity control u [m/s]

Controller	IAE	ITAE
SOC zu392	7240.84	60747943.10
SOC zu738	10891.62	78058177.06
FFDL	16062.03	108808995.30

(d) Error metrics for longitudinal acceleration control $\dot{u}$ [m/s^2]

Controller	IAE	ITAE
SOC zu392	150.88	845406.01
SOC zu738	146.74	955962.47

for the single network controller are,

Table 4.8: Control times for the SOC zu738 control algorithm in the nominal and windy tests.

Controller	Mean [ms]	Std. Dev. [ms]	Max [ms]
SOC zu738 Nominal	1.17	0.48	7.18
SOC zu738 Wind	0.94	0.31	4.13

The results from Table 4.8 confirm that the f_{publish} frequency is respected in the ROS publications. In fact, taking the maximum time form the SOC zu738 Nominal case, a frequency of $f_{\text{publish}} = \frac{1}{7.18}$ [Hz] ≈ 139.3 [Hz] could be used without issue.

4.3.5 Summary of Altitude and Velocity Control

As in the previous section, these results work towards validating the single-network controller in a higher-dimensional scenario. In this section the altitude and velocity are controlled, successfully tracking the desired setpoints in both nominal and disturbed test cases. In all test flights, the controllers outperformed the comparison algorithms both in system response and in reduced chattering in the actuator signals.

4.4 Summary

The experimental results from the suboptimal control strategy presented in Algorithm 4.1, being applied as a controller for the states in (4.19) and (4.20) showed a successful ability to track multiple desired waypoints both in the nominal case and when there is turbulent wind with maximum velocity amplitudes of 3 [m/s]. In this chapter, it is demonstrated that the suboptimal control algorithm is relatively insensitive to the choice of learning rate parameter and more notably, can produce admissible control trajectories for a large range of initial values of the critic weights.

The performance of the suboptimal controller, compared to that of a FFDL controller, is more desirable when evaluated using the IAE and ITAE metrics in the limited testing that is quoted in this chapter. This is a promising conclusion as the FFDL controller represents the state-of-the-art in DDC and has been implemented successfully in many test cases. In addition to outperforming the comparison controller, the suboptimal controller is not restricted to $m \leq n$ in the same way the FFDL algorithm, or $m = n$ like the algorithm proposed by Clouâtré, Thitsa, and Join.

While the results from this chapter are promising, more understanding is needed as to how the critic weights influence the system response. A better understanding this relationship could allow a designer to dictate more readily how the system will respond and might lead to stability formulations or methodological selection of initial conditions in future extensions of the algorithm.

Expanding on the results from this chapter, Chapter 5 uses the suboptimal controller as a starting point and adds a second partially connected neural network, namely the actor network, to offer the algorithm more DoF in its ability to adapt based on the data measured along the trajectory.

Chapter 5

Suboptimal Controller with Critic and Actor Neural Networks

This chapter develops a second Multi-Input Multi-Output (MIMO) Data-Driven Control (DDC) suboptimal controller with a double neural network structure, herein the actor-critic controller, by using the results developed in Chapter 4 as a framework. To develop this controller another neural network, the actor network, is added to calculate the control action. The addition of the actor network gives the controller more degrees of freedom in terms of weights that can be adjusted to eliminate the transient and steady state errors. The actor network is updated online using input output data and, therefore addresses the same problems as the single layer network.

The outline of this Chapter is as follows. Section 5.1 introduces the mathematical development of the actor network, outlining the network structure, the update laws and how the critic network is used to advise the actor network. Following an ablation study for the learning rate parameter for the actor weights, the two controllers showing admissible control policies are tested in single and multiple waypoint tests for an altitude control task in Section 5.2. In Section 5.3 the control task is complicated by adding longitudinal velocity and acceleration to the states to be controlled. Concluding remarks are added in Section 5.4 discussing the performance of the proposed controller and future considerations for the development of the algorithm and it's application to dirigibles.

5.1 Actor-Critic Controller Development

In this section, MIMO DDC scheme presented in Section 4.1 is extended by adding a second neural network, namely the actor network, to allow the controller more Degree of Freedom (DoF) in tracking the desired trajectories. As in Chapter 4, the controllerdesign addresses the problems of the unknown system dynamics by leveraging a critic network to approximate the functions that depend on these terms. Borrowing from optimal control theory, the critic network is used to derive a desired control action that is parameterized through the critic weights, which update based on the measurements gathered along the trajectories. Following the Weierstrass higher order approximation theorem, an actor network is developed whose inputs are the normalized and saturated errors and whose output is the control action. The updates of the actor network are advised by the desired suboptimal control action from the controller developed in Chapter 4.

With slight abuse of notations, the augmented vector from (4.1) is redefined using the control action calculated from the actor network, $\tilde{\mathbf{u}}_{k,a}$, in place of the control action that is parameterized from the critic network, $\tilde{\mathbf{u}}_k$. The augmented vector is redefined as

$$\mathbf{w}_k \equiv [w_{k,1}, w_{k,2}, \ldots, w_{k,n+m}]^T \equiv [\mathbf{e}_k^T \ \tilde{\mathbf{u}}_{k,a}^T]^T \in \mathbb{R}^{n+m}. \tag{5.1}$$

Denote the control action recommended by the suboptimal controller presented in Chapter 4, whether optimal or not, by

$$\tilde{\mathbf{u}}_k^{[d]} = -\mathbf{P}_{\text{uu}}^{-1}\mathbf{P}_{\text{ue}}\mathbf{e}_k. \tag{5.2}$$

Motivated again by Theorem 4.1, (5.2) is approximated as

$$\hat{\mathbf{u}}_{k,a} = \bar{\mathbf{\Omega}}_a^{[r]}\bar{\sigma}(\mathbf{e}_k). \tag{5.3}$$

To structure this as a neural network define the activation function to be a quadratic basis function of the input vector $\mathbf{e}_k$

$$\bar{\sigma}(\mathbf{e}_k) : \mathbb{R}^n \to \mathbb{R}^{n^2} = \mathbf{e}_k \otimes \mathbf{e}_k \tag{5.4}$$

and the weights of neural network as

$$\bar{\mathbf{\Omega}}_a^{[r]} = \begin{bmatrix} \bar{\omega}_{1,1} & \ldots & \bar{\omega}_{1,m} \\ \vdots & \ddots & \vdots \\ \bar{\omega}_{n^2,1} & \ldots & \bar{\omega}_{n^2,m} \end{bmatrix} \in \mathbb{R}^{n^2 \times m}.$$

Note that (5.3) is valid as presented, but it is also possible to reduce the order in a similar way as Chapter 4. In reducing the order of (5.3), only the redundant terms from the activation function for the actor network, $\sigma(\cdot)$, are removed as there is no symmetry imposed on the actor weights, $\bar{\mathbf{\Omega}}_a^{[r]}$, and hence no redundant terms. The reduced order modified Kronecker product is then defined as

$$\boldsymbol{\sigma}(\mathbf{e}_k) = [S_{1,1}e_{k,1}^2, S_{1,2}e_{k,1}e_{k,2}, \ldots, S_{i,j}e_{k,i}e_{k,j}]^T, \quad \forall i \in \{1, 2, \ldots, n\},\ j \leq i \tag{5.5}$$

$$S_{i,j} = \begin{cases} \frac{1}{2} & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$$

With this construction, the activation function for the actor network is a mapping such that

$$\boldsymbol{\sigma}(\cdot) : \mathbb{R}^n \to \mathbb{R}^{\bar{n}}$$

for $\bar{n} = \frac{n(n+1)}{2}$. Note that the term $S_{i,j}$ in (5.5) is added to ensure that the sign of the error is considered by the network. Without the addition of this term, the diagonal elements of the activation function, corresponding to the uncoupled error, would not have any way of differentiating between positive and negative errors. The associated actor network weights are

$$\mathbf{\Omega}_a^{[r]} = \begin{bmatrix} \omega_{1,1} & \ldots & \omega_{1,m} \\ \vdots & \ddots & \vdots \\ \omega_{\bar{n},1} & \ldots & \omega_{\bar{n},m} \end{bmatrix} \in \mathbb{R}^{\bar{n} \times m}.$$

The actor neural network then has the structure shown in Figure 5.1.

The closed form equation for the control action for the actor critic framework is

$$\tilde{\mathbf{u}}_{k,a} = \mathbf{\Omega}_a^{[r]} \sigma(\mathbf{e}_k). \tag{5.6}$$

Herein, it is assumed that the reduced order structure is implemented, that is $\mathbf{\Omega}_a^{[r]} \in \mathbb{R}$. Defining $\hat{\kappa} = k$ modulo η. To develop the update law, two matrices are defined

$$\mathbf{\Sigma} = [\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_{\eta-1}]^T \quad \text{for,} \quad \Sigma_\kappa = \sigma^T(\mathbf{e}_{\hat{\kappa}}) \quad \text{and}$$

$$\mathbf{U}^{[\mathbf{d}]} = [\tilde{\mathbf{u}}_0, \tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_{\eta-1}]^T \quad \text{for,} \quad \tilde{\mathbf{u}}_\kappa = \tilde{\mathbf{u}}_{\hat{\kappa}}^{[d]}.$$

As was the case in Chapter 4, η points are collected along the trajectory per batch. The

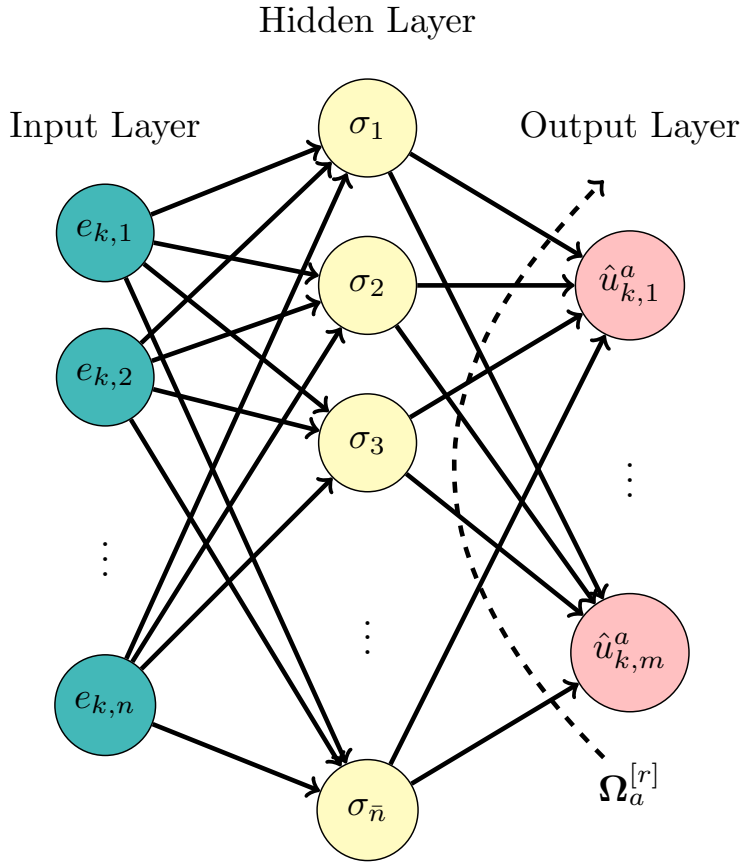


Figure 5.1: Structure of the actor neural network.

compact form of (5.3) over the batch for policy iteration $r \in \{1, 2, \ldots\}$ is

$$\hat{\mathbf{u}} = \mathbf{\Sigma}\mathbf{\Omega}_a^{[r]}. \tag{5.7}$$

The temporal difference equation is then derived as the squared error between the desired control action, (5.2), and the actor control action approximation, (5.6), over η samples. That is

$$\delta_a = \frac{1}{2}\left\|\tilde{\mathbf{u}}^{[d]} - \hat{\mathbf{u}}\right\|^2. \tag{5.8}$$

The weights of the actor network are updated using gradient descent for some learning rate parameter $\alpha_a \ll 1$. The update law is chosen such that (5.8) is minimized with respect to

$\mathbf{\Omega}_a^{[r]}$. Then

$$\mathbf{\Omega_a}^{[r+1]} = \mathbf{\Omega_a}^{[r]} - \alpha_a \frac{\partial \delta_a}{\partial \mathbf{\Omega_a}^{[r]}} = \mathbf{\Omega_a}^{[r]} - \alpha_a \mathbf{\Sigma}^T \Big(\mathbf{\Sigma} \mathbf{\Omega_a}^{[r]} - \tilde{\mathbf{u}}^{[d]} \Big). \tag{5.9}$$

It is noted that the control action is considered optimal only for the case where the weight matrices for the actor and critic networks have converged. That is the control action is

$$\tilde{\mathbf{u}}_{k,a} = \mathbf{\Omega}_a^{[r]} \sigma(\mathbf{e}_k) = \begin{cases} \tilde{\mathbf{u}}_{k,a}, & \text{if } \left\| \mathbf{\Omega}_c^{[r]} - \mathbf{\Omega}_c^{[r-1]} \right\| \geq \epsilon^{\text{conv}} \text{ or } \left\| \mathbf{\Omega}_a^{[r]} - \mathbf{\Omega}_a^{[r-1]} \right\| \geq \epsilon^{\text{conv}} \text{ or} \\ \tilde{\mathbf{u}}_{k,a}^*, & \text{if } \left\| \mathbf{\Omega}_c^{[r]} - \mathbf{\Omega}_c^{[r-1]} \right\| < \epsilon^{\text{conv}} \text{ and } \left\| \mathbf{\Omega}_a^{[r]} - \mathbf{\Omega}_a^{[r-1]} \right\| < \epsilon^{\text{conv}} \end{cases}. \tag{5.10}$$

The overall structure of the suboptimal state-feedback controller is depicted in Figure 5.2.

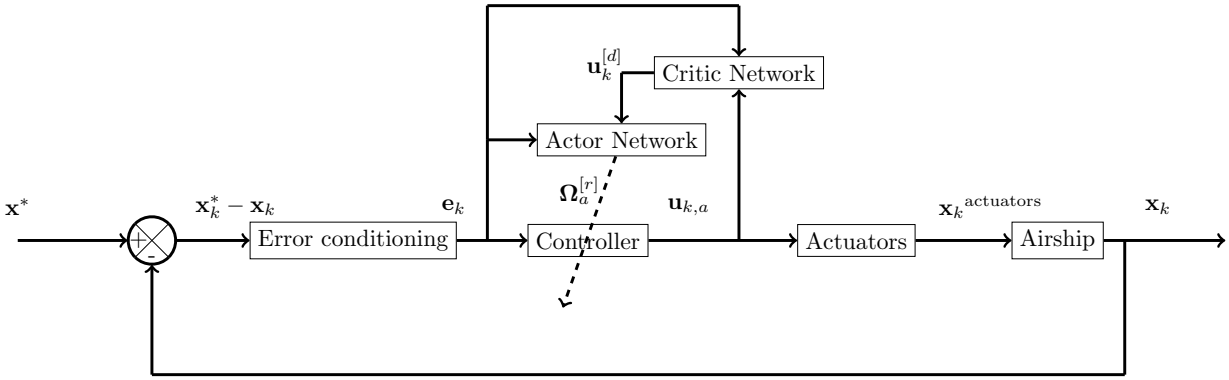


Figure 5.2: Block diagram for the actor-critic controller.

As in Figure 4.2, the error conditioning block takes the true state error and passes the normalized and saturated error to the controller and both the actor and critic neural networks. The controller block represents the control policy, however the controller is parameterized on the weights of the actor network, which is advised by the critic network. The output for the controller block is passed to the actuators of the airship. The resulting states are treated as the outputs. The critic network weights, $\mathbf{\Omega}_c^{[r]}$, and actor network weights, $\mathbf{\Omega}_a^{[r]}$, are updated on each policy iteration. This structure represents an actor-critic adaptive control system. The algorithm is summarized in Algorithm 5.1.

Algorithm 5.1 Algorithm 2: Suboptimal Control with Actor and Critic Neural Networks

Input:

$\mathbf{P}^{[0]}$: Symmetric positive definite
$\mathbf{\Omega}_a^{[0]}$: Random initialization
$\mathbf{Q} \in \mathbb{R}^n$: Symmetric Positive Definite
$\mathbf{R} \in \mathbb{R}^m$: Symmetric Positive Definite
K: Maximum value of the discrete-time variable

Output:

Initialize the following variables:
$r \leftarrow 0$
$\Upsilon \in \mathbb{R}^{\eta \times \eta}$ as an empty matrix
$\Sigma \in \mathbb{R}^{\eta \times \bar{n}}$ as an empty matrix
$\tilde{\mathbf{u}}^{[d]} \in \mathbb{R}^{\eta \times m}$ as an empty matrix
$\mathbf{v}^{[d]} \in \mathbb{R}^{\eta}$ as an empty column vector
Measure the initial normalized error vector $\mathbf{e}_0$
for $(k \leftarrow 0,\ k \leq K,\ k \leftarrow k+1)$ **do**
    Measure the states $\mathbf{x}_{k+1}$
    Calculate the normalized error $\mathbf{e}_{k+1}$
    Calculate the control action $\tilde{\mathbf{u}}_{k,a}$ using (5.10)
    Calculated the augmented vector $\mathbf{w}_{k+1} \leftarrow \left[\mathbf{e}_{k+1}^T \tilde{\mathbf{u}}_{k+1,a}^T\right]^T$
    $\kappa \leftarrow k$ modulo $\eta - 1$
    Set $\Upsilon_{\bar{k}} \leftarrow \tilde{\rho}_{\bar{k},k\bar{+}1}$
    Set $\mathbf{v}_{\bar{k}}^{[d]} \leftarrow \frac{1}{2}\mathbf{w}_k^T \bar{\mathbf{P}} \mathbf{w}_k$
    if $(k+1)$ is divisible by η **then**
        Increase the policy index $r \leftarrow r+1$
        Update the weights of the critic matrix $\mathbf{\Omega}_c^{[r]} \leftarrow \mathbf{\Omega}_c^{[r-1]} + \alpha_c \Upsilon^T \left(\Upsilon \mathbf{\Omega_c}^{[r-1]} - \mathbf{v}^{[d]}\right)$
        Update the weights of the actor matrix $\mathbf{\Omega}_a^{[r]} \leftarrow \mathbf{\Omega}_a^{[r-1]} + \alpha_a \Sigma^T \left(\Sigma \mathbf{\Omega_a}^{[r]} - \tilde{\mathbf{u}}^{[d]}\right)$
    end if
end for

5.2 Altitude Control with Two Network Controller

As was the case in Section 4.2, the actor-critic structure is first validated with a single-input 2-output altitude and vertical velocity control task to allow for the exploration of the different elements of the controller. The actor-critic controller is implemented with the

same measurements and sends the same actuation signals as the controller in Chapter 4 and therefore the implementation details are excluded here instead, the reader is urged to review Section 4.2.1 for details, noting that the same comparison controller is used as well. In Section 5.2.1 different learning rates for the actor network are tested in an ablation study to observe the difference and determine which is best suited for this controller. In Section 5.2.2 both nominal and wind disturbed flight test simulations are performed to qualitatively evaluate the performance of the controller for a single waypoint. In Section 5.2.3, the single waypoint experiments are repeated with multiple altitude waypoints to observe the controllers adaptability.

5.2.1 Learning Rate Ablation Study

In this section three learning rates, α_a

where are three learning rates? Need to change this sentence highlighting that you used three different values for α_a.

for the actor network are applied to the update rule, (5.9), for the actor-critic controller while holding all other design variables constant in an ablation study. The objective is to observe the sensitivity of the system to the choice of learning rate parameter for the actor weights. In all cases the target state, $\mathbf{x}^*$, and the maximum values of for the states used in (3.5), $\mathbf{x}^{\max}$, for the controller are

$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \end{bmatrix} = \begin{bmatrix} 15 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x}^{\max} = \begin{bmatrix} z^{\max} \\ \dot{z}^{\max} \end{bmatrix} = \begin{bmatrix} 22.5 \\ 1.5 \end{bmatrix}, \tag{5.11}$$

respectively. For all experiments in Section 5.2 the critic weights are held constant as

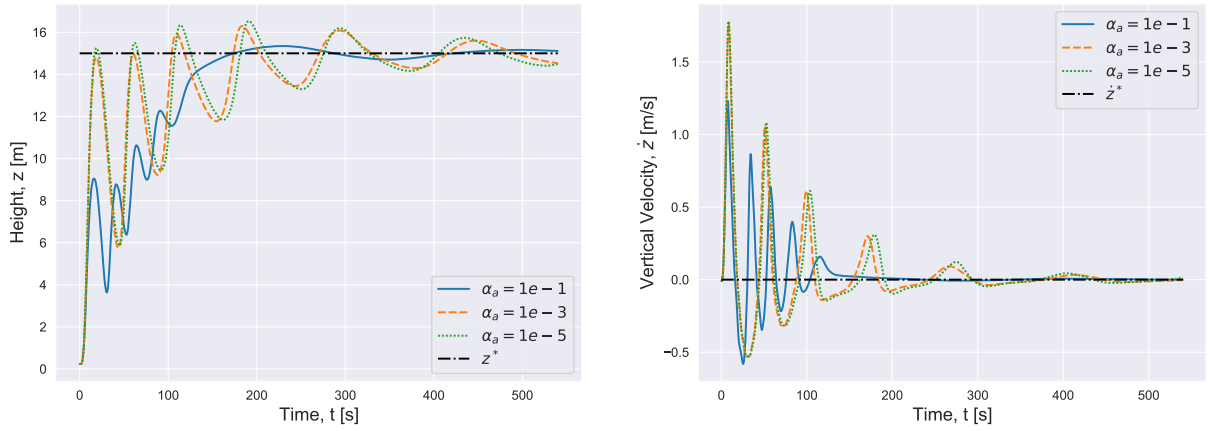
$$\mathbf{\Omega}_c^{[0]} = [0.30, 0.43, 0.93, 0.36, 0.94, 1.00]^T.$$

The initial value of the actor weights in all three controllers in Section 5.2.1 are

$$\mathbf{\Omega}_a^{[0]} = [-0.69, -0.96, -0.81]^T.$$

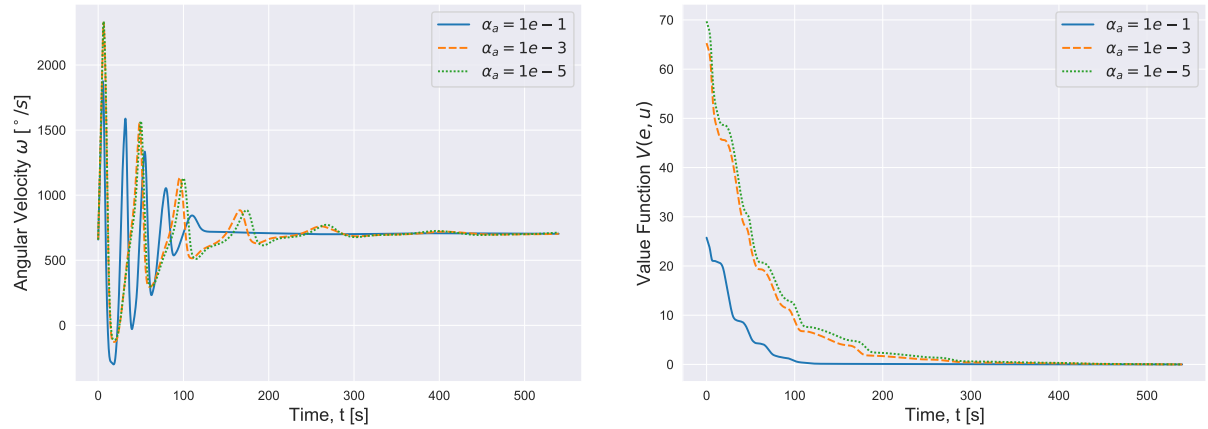
The results from the learning rate ablation study are shown in Figures 5.3 and 5.4.

Unlike the suboptimal controller, the responses show a high sensitivity to the learning rates for the actor network. Both the vertical position and vertical velocity plots in Figure 5.3 show that the reduced learning rates for the actor network lead to a longer tracking time, and show larger, higher frequency transient oscillations compared to the higher learning rates. The value function responses in Figure 5.4(b) show that the controller with the



(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s]

Figure 5.3: State responses for the learning rate ablation study for $\alpha_a \in \{10^{-1}, 10^{-3}, 10^{-5}\}$ for a single waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



(a) Angular velocity, ω [°/s] (b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 5.4: Actuator signals and value function responses from the learning rate ablation study for $\alpha_a \in \{10^{-1}, 10^{-3}, 10^{-5}\}$ for a single waypoint experiment. (a) actuator signal, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

highest learning rate, $\alpha_a = 10^{-1}$ converges in the actor weights approximately 200 [s] faster than the other two controllers. Further, it is observed that the controller with $\alpha_a = 10^{-3}$

is sandwiched between the controllers with higher and lower learning rates. This indicates that, in general, the larger the learning rate the better. The error metrics for this ablation study are tabulated in Table 5.1.

Table 5.1: Error metrics for altitude controllers with learning rates in the range $\alpha_a \in \{10^{-1}, 10^{-3}, 10^{-5}\}$

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
$\alpha_c = 10^{-1}$	12669.37	13353325.31
$\alpha_c = 10^{-3}$	12147.90	23492321.17
$\alpha_c = 10^{-5}$	12523.09	26293978.29

(b) Error metrics for vertical velocity control $\dot{z}$ [m/s]

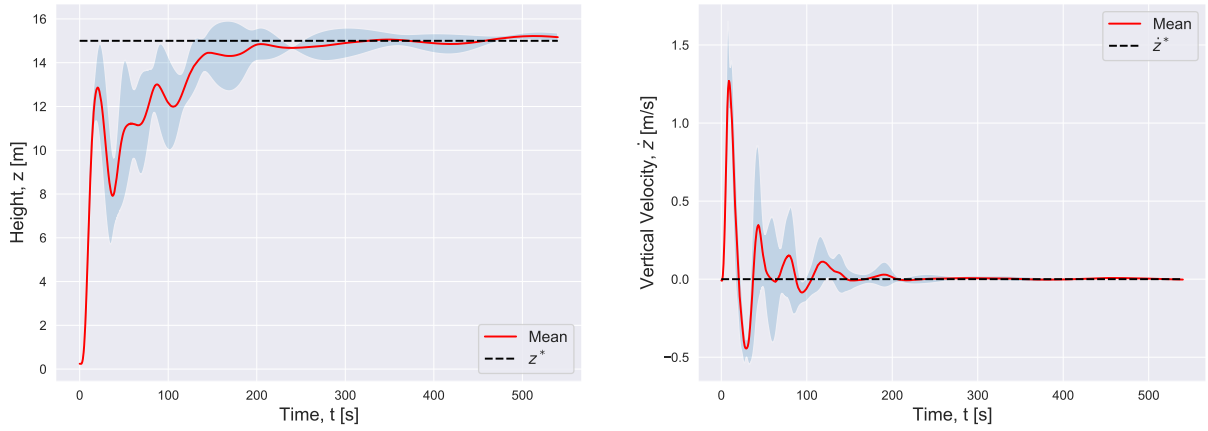
Controller	IAE	ITAE
$\alpha_c = 10^{-1}$	556.83	492598.13
$\alpha_c = 10^{-3}$	961.23	1406323.57
$\alpha_c = 10^{-5}$	1016.38	1583936.69

Interestingly, the Integral Absolute Error (IAE) score for the vertical position of the controller with $\alpha_a = 10^{-1}$ has a marginally higher score compared to the others, while its Integral Time Absolute Error (ITAE) scores are approximately half, compared to the other controllers. This is explained by noting that the IAE score weights the errors across the trajectory equally, whereas the ITAE score puts more emphasis on the ability of the controller to reduce the steady-state error. This implies that the higher learning rate accepts larger errors in the transient, learning, phase, but has the ability to reduce the error more rapidly, and perhaps more effectively for larger values of k. For the, remainder of the altitude control section, learning rates of $alpha_c = 10^{-2}$ and $\alpha_a = 10^{-1}$ are used.

5.2.2 Altitude Controller with Single Waypoint

In this section an ablation study is performed using different initial values of the critic weights $\mathbf{\Omega}_a^{[0]}$ to determine the sensitivity of the system response with respect to the initial weights. Five different initial values are used. In all cases in this section the target states, $\mathbf{x}^*$ and the maximum values, $\mathbf{x}^{\max}$ are the same as in (5.11).

The initial values of the actor weights are randomly initialized with random seeds. Their values are given in Appendix A.3 for completeness. The results of the ablation study over the different random seeds are plotted in Figure 5.5 with the solid red line indicating the mean of each controller for a given time interval and the shaded blue area indicating a vertical range of $\pm$ 1 standard deviation from the mean.



(a) z [m] (b) $\dot{z}$ [m/s]

Figure 5.5: Ablation study for altitude control, varying the initialization of the critic weights $\mathbf{\Omega}_c^{[0]}$ for a single waypoint. The red line indicates the mean response and the shaded area represents a vertical width of one standard deviation from the mean system response. (a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s].

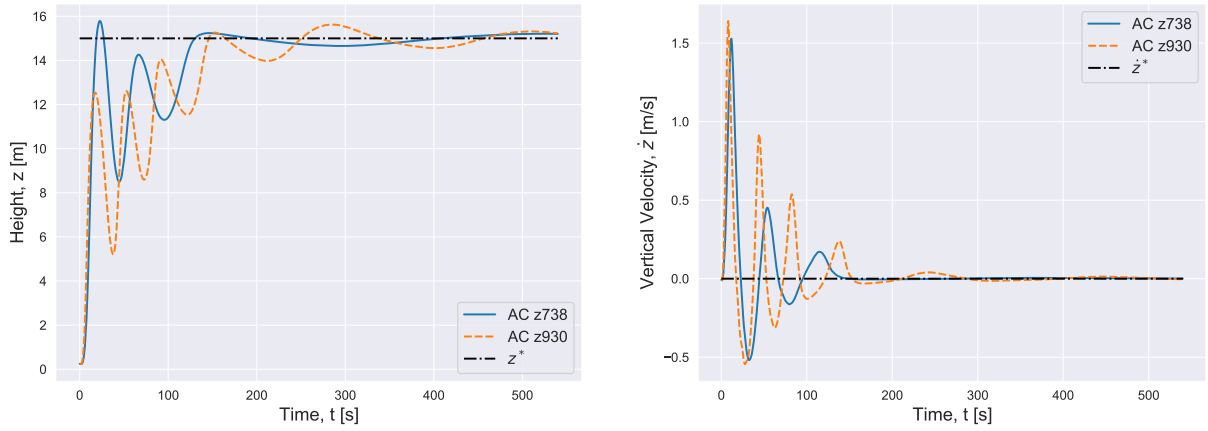
Notations 5.1. *Herein, AC is used to refer to the suboptimal control structure with a two neural networks, the critic and actor networks. A controller "AC z###" refers to the actor-critic controller for states in* (5.11)*, with random seed ###.*

Both z and $\dot{z}$ responses shown in Figure 5.5 ultimately seem to converge such that the standard deviation tightens around the mean as k increases, which is a positive result. The shaded area in Figure 5.5(a) shows larger deviation from the mean for approximately 300 [s], before seeming to tighten, whereas the vertical velocity response, Figure 5.5(b) begins to close in towards the mean response around 200 [s].

The state response for two controllers, AC z738 and AC z930 are plotted in Figure 5.6. The actuator signals and associated value functions are shown in Figure 5.7.

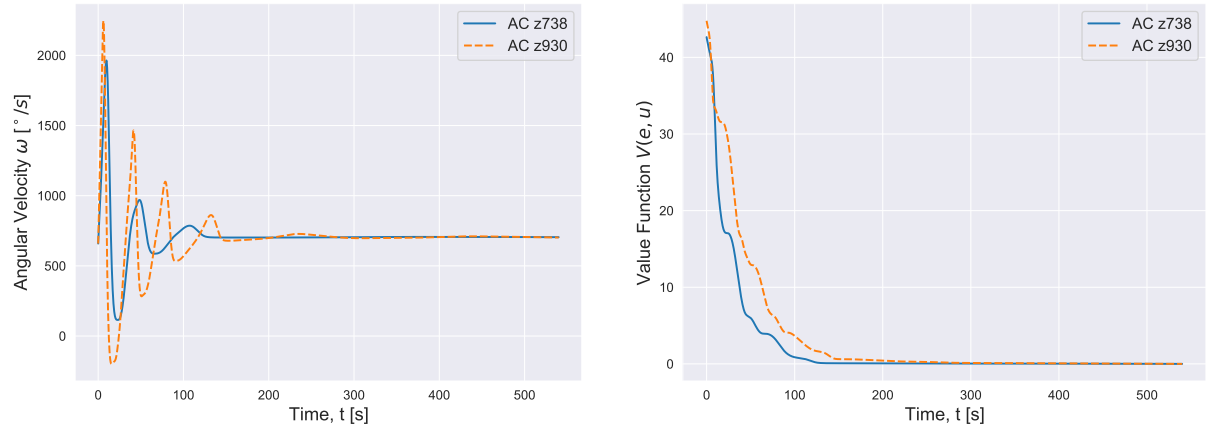
Two controllers, AC z738 and AC z930 are used for further testing. Figures 5.6 and 5.7 show the state responses and actuator signals and value functions, respectively.

The actor-critic controllers show a smooth response in reducing the steady-state error for both the vertical position and the vertical velocity. It is observed that the actuator signals and value functions converge after the 100 [s] mark, showing delayed convergence compared to the results in the suboptimal controller from Section 4.2.5. It is supposed that this is due to the additional parameters that are required to learn, namely the weights



(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s]

Figure 5.6: State responses for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



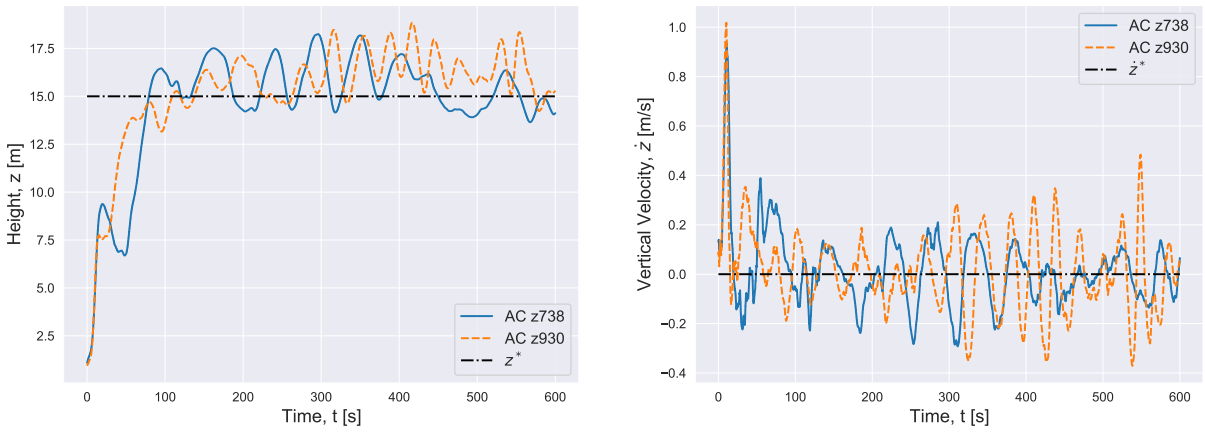
(a) Angular velocity, ω [°/s] (b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 5.7: Actuator signals and value function for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment. (a) angular speed, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

in the actor network. Qualitatively, the results from the plots seem to be smoother in the steady-state region compared to the suboptimal controller.

The single waypoint undisturbed experiment for the actor-critic controller is repeated,

exposing the airship to turbulent wind. The results for the vertical position and vertical velocity are shown in Figure 5.8 and the results for the control action and the value function are shown in Figure 5.9.



(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

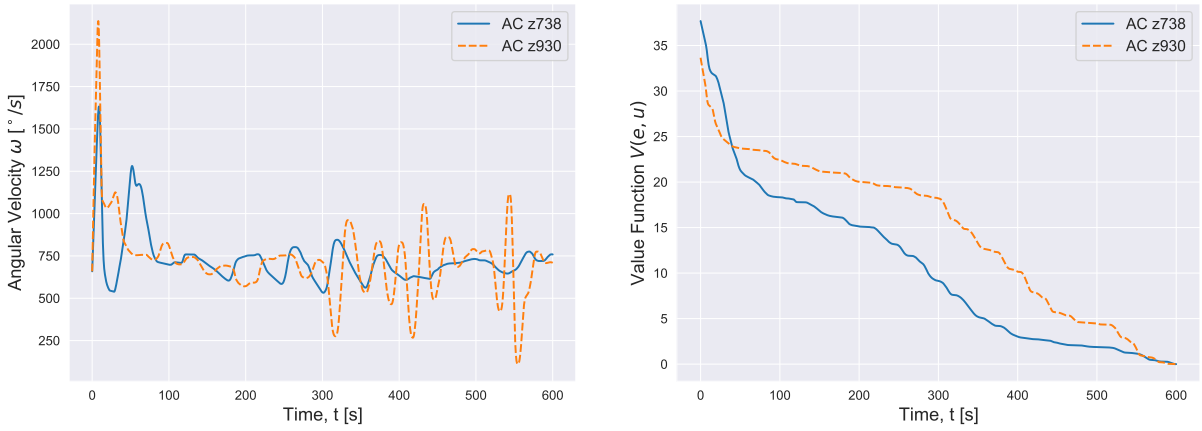
Figure 5.8: State responses for suboptimal controllers with random seeds 738 and 930 for a single waypoint experiment in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

From Figure 5.8 it is observed that the actor-critic controllers are able to track the desired reference altitude with both controllers seeming to be marginally biased towards positive errors. In Figure 5.9(a), it is observed that the actuator signals seem to have a much lower amplitude fluctuation when compared to the results of the suboptimal controller.

While the results for the single waypoint experiments are only qualitative, it is confirmed that in this experimental environment and tracking task, the actor-critic structure is capable of controlling the states in a desirable fashion and is robust to turbulent wind gusts of up to 2 [m/s].

5.2.3 Altitude Controller with Multiple Waypoints

In this section the same two controllers from Section 5.2.2, AC z738 and AC z930, are tested in comparison to the Full Form Dynamic Linearization (FFDL) in a multiple waypoint simulation in both nominal and windy environments, to verify the adaptability of the proposed controller. In both tests the target states, $\mathbf{x}^*$, and the maximum values of for the states, $\mathbf{x}^{\max}$ are,



(a) Angular velocity, ω [°/s]

(b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

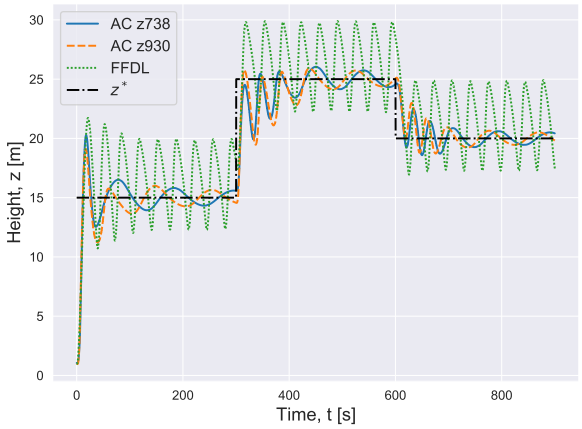
Figure 5.9: Actuator signals and value function for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \end{bmatrix} = \begin{bmatrix} T_z \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\max} = \begin{bmatrix} z^{\max} \\ \dot{z}^{\max} \end{bmatrix} = \begin{bmatrix} 22.5 \\ 1.5 \end{bmatrix} \quad \text{for}$$

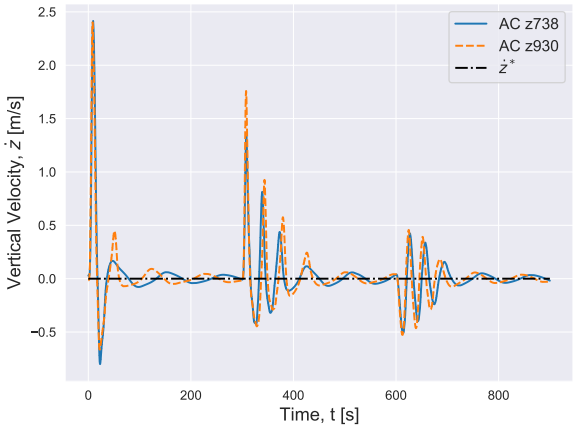
$$T_z = \begin{cases} 15, & \text{if } t \leq 300 \text{ [s]}, \\ 25, & \text{if } 300 \text{ [s]} < t \leq 600 \text{ [s]} \\ 20, & \text{if } 600 \text{ [s]} < t. \end{cases} \quad \text{and}$$

The state responses for the multiple waypoint testing are shown in Figure 5.10 with the actuator signals and the value functions for each controller being shown in Figure 5.11.

Both actor-critic controllers are able to overcome the change in waypoints and show a similar response when tracking positive and negative errors in both the transient and steady-state responses, tracking the altitude much more efficiently than the FFDL controller which shows consistent peak-to-peak oscillations of roughly, 7 [m], compared to the oscillations of the actor-critic network which appear to be roughly, ± 1 [m]. Comparing the actuation signals in Figure 5.11(a), the actor-critic controllers show very smooth actuation trajectories, while the FFDL controller bounces from one saturation point to the next. The value functions of the actor-critic controllers begin to level off in approximately 75 [s] of the change in waypoint, which seems to be an improvement from the earlier suboptimal

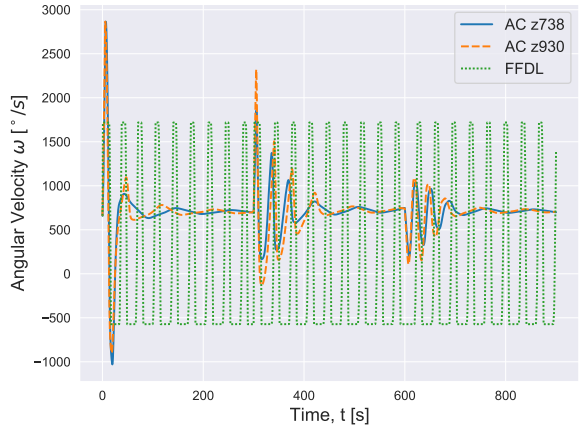


(a) Vertical position, z [m]

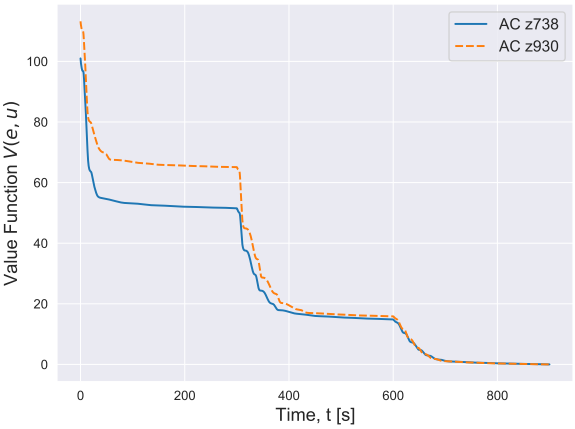


(b) Vertical velocity, $\dot{z}$ [m/s]

Figure 5.10: State responses for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



(a) Angular velocity, ω [°/s]



(b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 5.11: Actuator signals and value function for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment. (a) angular speed, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

controllers.

From the error metrics in Table 5.2, the qualitative analysis of the system responses is

Table 5.2: Error metrics for the actor-critic controllers with random seeds 738 and 930 and FFDL for multiple waypoint experiments. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

(a) Error metrics for altitude control z [m]

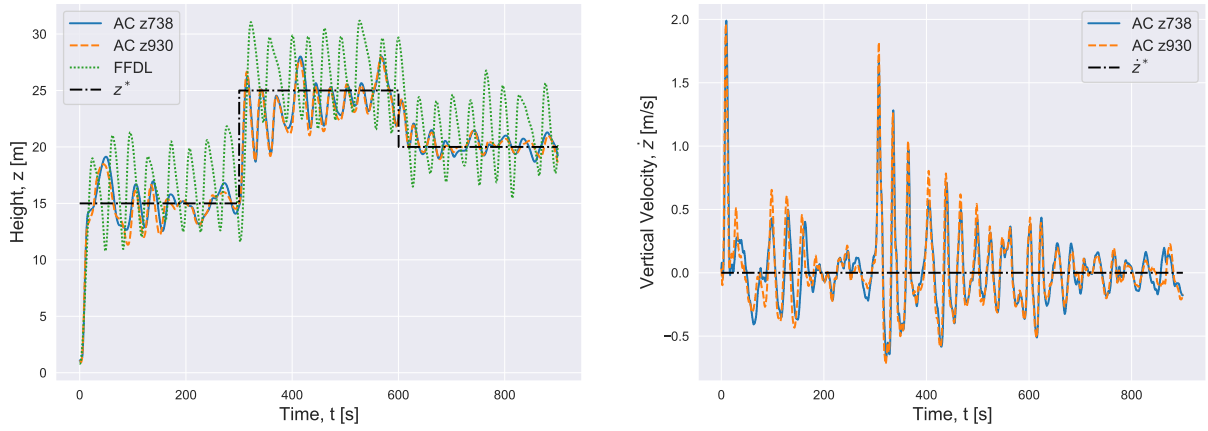
Controller	IAE	ITAE
AC z738	13905.25	82201544.10
AC z930	14333.04	84510302.60
FFDL	29234.15	198600419.79

(b) Error metrics for altitude control $\dot{z}$ [m/s]

Controller	IAE	ITAE
AC z738	1534.32	7510896.91
AC z930	1665.09	8233856.77

confirmed. The two actor-critic controllers respond in very similar ways and receive similar scores in the IAE and ITAE evaluations for both states. By all four metrics, the AC z738 actor-critic controller outperforms the other two, being smaller than the FFDL response by a factor of 2.10 and 2.42 for the IAE and ITAE scores, respectively.

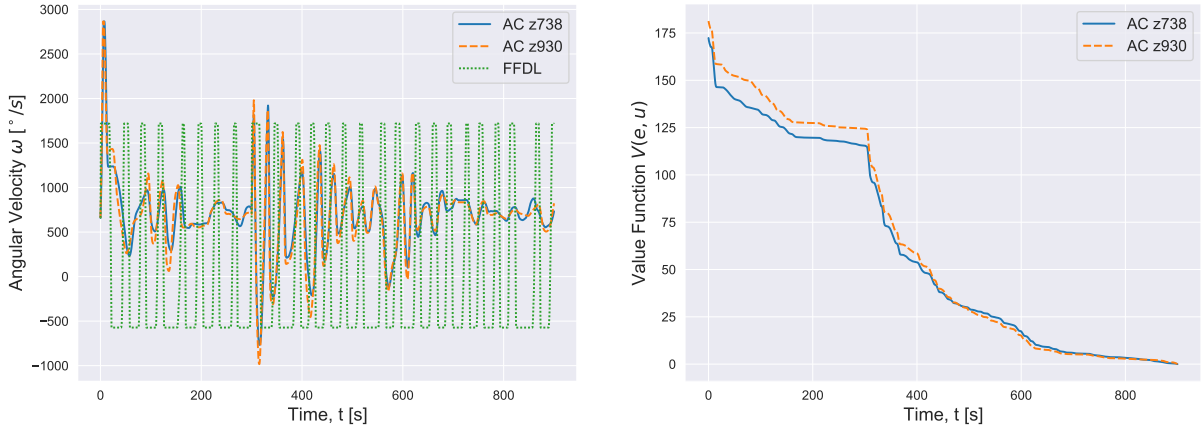
The multiple waypoint tests are repeated with the addition of turbulent wind to challenge the controller's ability to reject external disturbances. All the controller configurations and desired waypoints are the same as in the nominal case. The state responses for the multiple waypoint testing in the presence of turbulent wind is shown in Figure 5.12 with the actuator signals and the value functions for each controller being shown in Figure 5.13.



(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

Figure 5.12: State responses for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



(a) Angular velocity, ω [°/s]

(b) Value function, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$

Figure 5.13: Actuator signals and value function for suboptimal controllers with random seeds 738 and 930 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) value functions, $V(\mathbf{e}_k, \tilde{\mathbf{u}}_k)$ at $t = kT$.

Considering the altitude response of the airship in Figure 5.12(a) and the propeller speed in Figure 5.13(a), it is seems true that the proposed actor-critic controllers have reduced the error more effectively throughout the span of the trajectory. The altitude and actuator responses of the AC z738 and AC z930 controllers are again enveloped by the FFDL controller throughout the whole trajectory with brief exceptions in the angular velocity response around the 0 [s] and 300 [s] intervals. Intuitively this makes sense as these are the times where the altitude trajectories increase. Qualitatively, the altitude response is smoother, with less deviation for the actor-critic controllers seeming to imply a better tracking performance. The error metrics are available in Table 5.2 and show a similar trend to the nominal case. The two actor-critic controllers behave similarly and thus score evenly, both of them outperforming the comparison controller by factors of 1.6 and 1.85 in the IAE and ITAE metrics, respectively.

5.2.4 Summary of Altitude Control

In a similar way to Section 4.2, the system response of the controllers to the altitude control task works towards the empirical validation of the actor-critic structure for the airship model. The actor-critic outperformed the comparison algorithm and showed qualitative improvements to the smoothness of the results. The controller was able to track the

waypoints in the nominal and wind disturbed cases. In Section 5.3, the control task is augmented by added the velocity and it's derivative to be controlled.

Table 5.3: Error metrics for suboptimal controllers with random seeds 738 and 930 and FFDL for multiple waypoint experiments in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
AC z738	19092.40	115320976.65
AC z930	19199.49	113329800.70
FFDL	31134.09	212580242.89

(b) Error metrics for vertical velocity control $\dot{z}$ [m/s]

Controller	IAE	ITAE
AC z738	2727.82	15688486.68
AC z930	2759.04	15524673.34

5.3 Altitude and Velocity Control with Two Network Controller

This section extends the complexity of the altitude control experiments to a 2-input 4-output control task for the altitude, z [m], vertical velocity, $\dot{z}$ [m/s], longitudinal velocity u [m/s] and longitudinal acceleration $\dot{u}$ [m/s^2], of the airship model described in Section 4.2.1. The controller parameters for the MIMO actor-critic and glsffdl controllers in this section are the same as outlined in Sections 4.3.1 and 4.3.2, respectively.

The structure of this section is similar to that of Section 4.2, with an ablation study being performed over a range of initial critic values and then selecting two specific controllers to investigate more rigorously in Section 5.3.1. Section 5.3.2 extend the work in Section 5.3.1, to multiple waypoint tests to investigate the adaptability of the proposed algorithms. A summary of the findings is offered in Section 5.4.

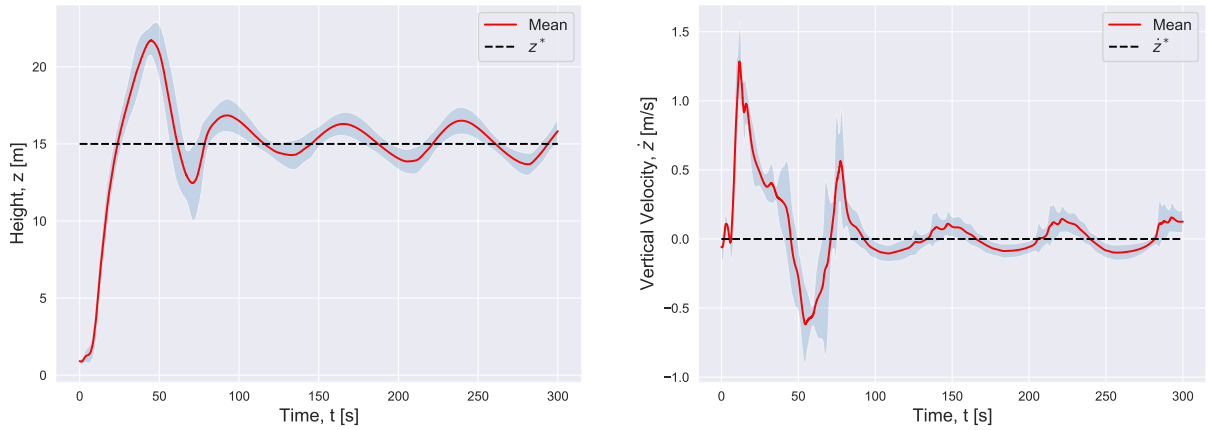
5.3.1 Altitude and Velocity Controller with Single Waypoint

In this section an ablation study is performed on the actor-critic controller using different initial values of the actor weights $\mathbf{\Omega}_a^{[0]}$ to determine the sensitivity of the altitude and velocity controller with respect to the initial actor weights. The values of the five actor weights are randomly initialized using random seeding. The values are given in Appendix A.4. The target and maximum values of the states in this experiment are

$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \\ u^* \\ \dot{u}^* \end{bmatrix} = \begin{bmatrix} 15 \\ 0 \\ 2 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\max} = \begin{bmatrix} z^{\max} \\ \dot{z}^{\max} \\ u^{\max} \\ \dot{u}^{\max} \end{bmatrix} = \begin{bmatrix} 7.5 \\ 1.5 \\ 2.5 \\ 0.5 \end{bmatrix} \tag{5.12}$$

Notations 5.2. *Modifying the notation from the previous section, a controller "AC zu###" refers to the actor-critic controller for states in* (5.12)*, with random seed ###.*

The results of the ablation study over the different random seeds are plotted in Figures 5.14 and 5.15 with the solid red line indicating the mean of each controller for a given time interval and the shaded blue area indicating a vertical range of ± 1 standard deviation from the mean.



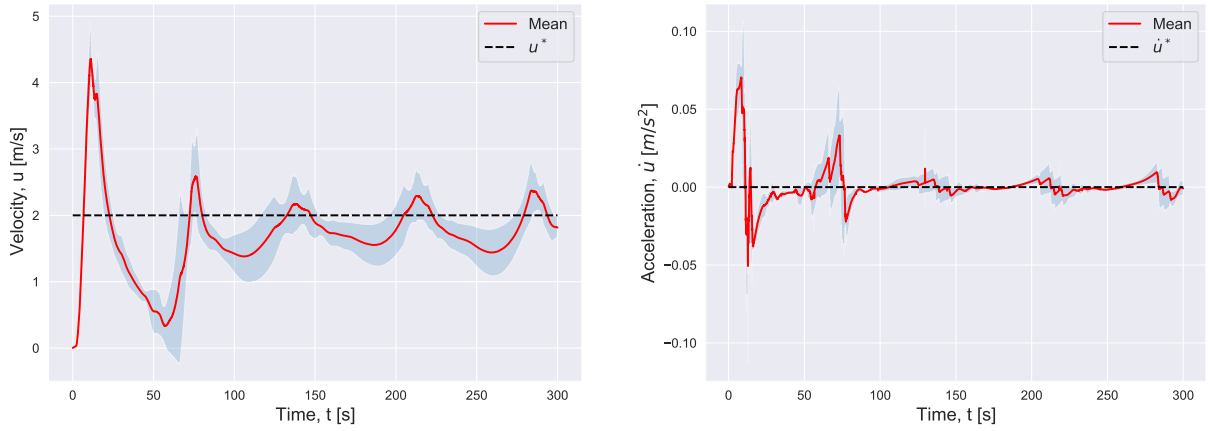
(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

Figure 5.14: Ablation study for altitude control, varying the initialization of the critic weights $\mathbf{\Omega}_a^{[0]}$ for a single waypoint. The red line indicates the mean response and the shaded area represents a vertical width of one standard deviation from the mean system response. (a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s].

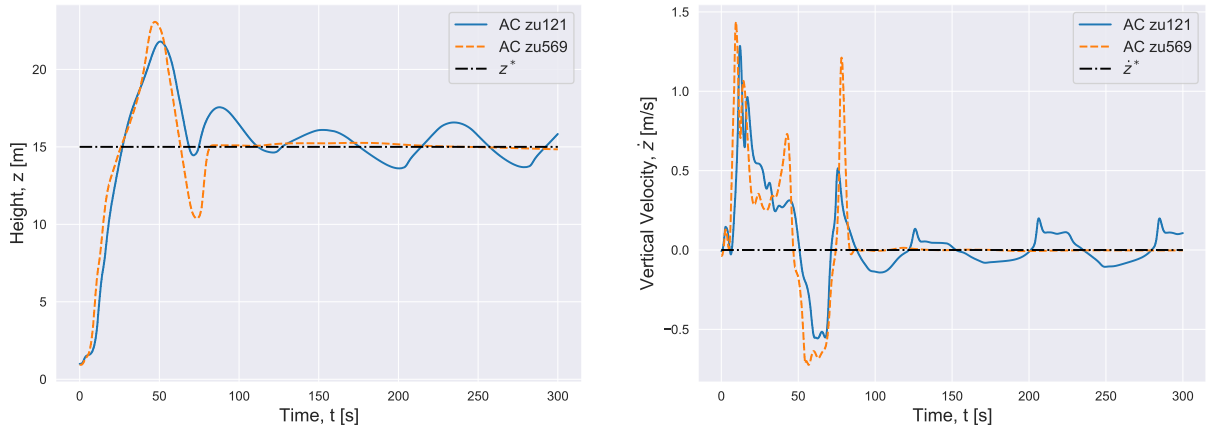
The interval of the actor-critic responses representing ±1 standard deviation from the mean seems to be constant throughout the trajectory, with the mean response oscillating about the desired reference signal. Two controllers, AC zu121 and AC zu569 are chosen to investigate further in a similar way to the previous sections. To visualize the response of two of the controllers, the state responses are plotted in Figures 5.16 and 5.17.

The state responses for the AC zu121 controller seem to track the desired reference trajectories, however, accept a fair amount of oscillation in both altitude and velocity



(a) Longitudinal Velocity, u [m/s] (b) Longitudinal Acceleration, $\dot{u}$ [m/s^2]

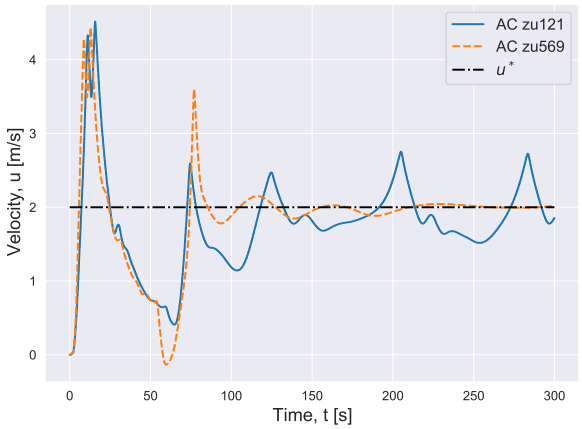
Figure 5.15: Ablation study for altitude control, varying the initialization of the critic weights $\mathbf{\Omega}_a^{[0]}$ for a single waypoint. The red line indicates the mean response and the shaded area represents a vertical width of one standard deviation from the mean system response. (a) longitudinal velocity, u [m/s] (b) longitudinal acceleration, $\dot{u}$ [m/s^2].



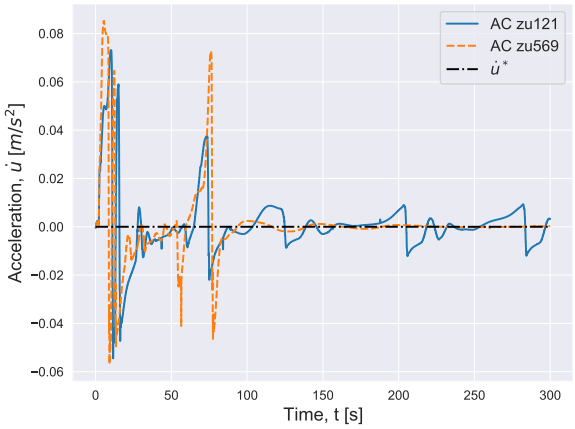
(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s]

Figure 5.16: State responses for suboptimal controllers with random seeds 121 and 569 for a single waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].

responses and their derivative. Conversely, the AC zu569 updates undistinguishably from the other test cases for the first 75 [s] and then seemingly adjusts to perfectly track z and



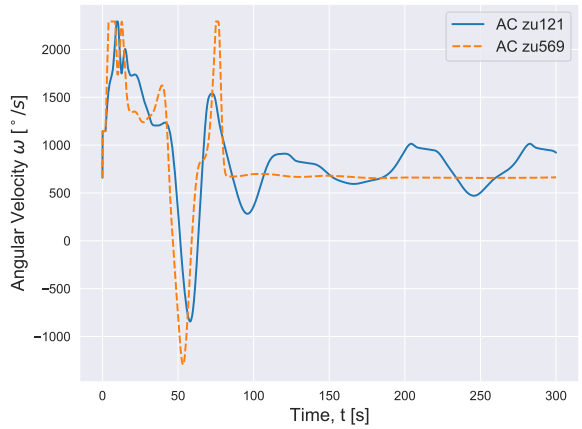
(a) Longitudinal Velocity, u [m/s]



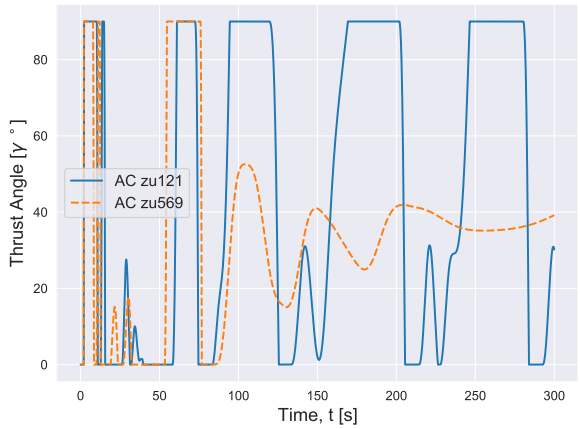
Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 5.17: State responses for suboptimal controllers with random seeds 121 and 569 for a single waypoint experiment. (a) longitudinal velocity u [m/s] and (a) longitudinal acceleration $\dot{u}$ [m/s^2].

$\dot{z}$, while accepting small errors in u and $\dot{u}$ for the remainder of the trajectory.



(a) Angular velocity, ω [°/s]



(b) Thrust angle, γ [°]

Figure 5.18: Actuator signals for suboptimal controllers with random seeds 121 and 569 for a single waypoint experiment. (a) angular speed, ω [°/s] and (b) thrust angle, γ [°].

The responses are similar in the actuator signals, with the fluctuations being observed for both actuators being controlled by the AC zu121 controller and a sudden leveling off

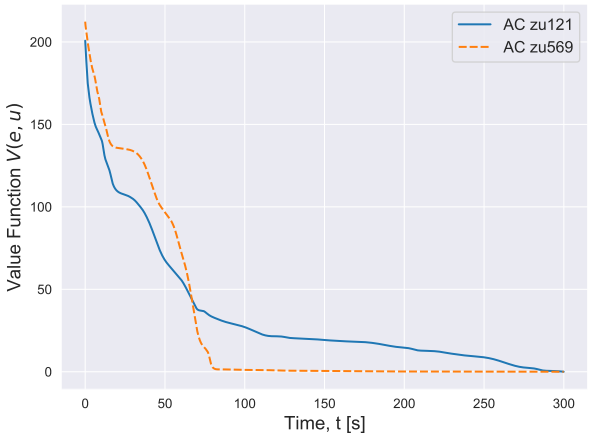


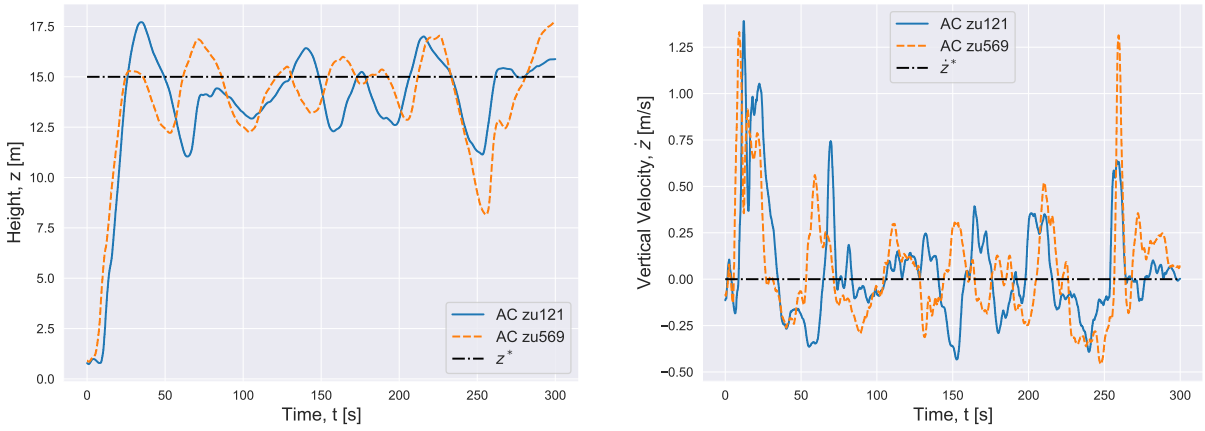
Figure 5.19: Value function for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment.

for the angular velocities of the AC zu569 controller and a decaying response for the vector thrusting.

The nominal single waypoint altitude and velocity control experiments are repeated in the presence of turbulent wind to verify the robustness of the proposed algorithms. The state responses from these tests are shown in Figure 5.20.

Firstly, it is noted that in Figures 5.20 and 5.21, both controllers are able to successfully track the desired setpoints and oscillate about them, despite the disturbance, although the error for the longitudinal velocity reaches 2 [m/s]. Observing the angular velocity response in Figure 5.22(a), the plots seem smoother than in the suboptimal controller responses, with fewer high-frequency oscillations, but the peak-to-peak deviation is larger. There doesn't seem to be much difference in response between the two actor-critic controllers.

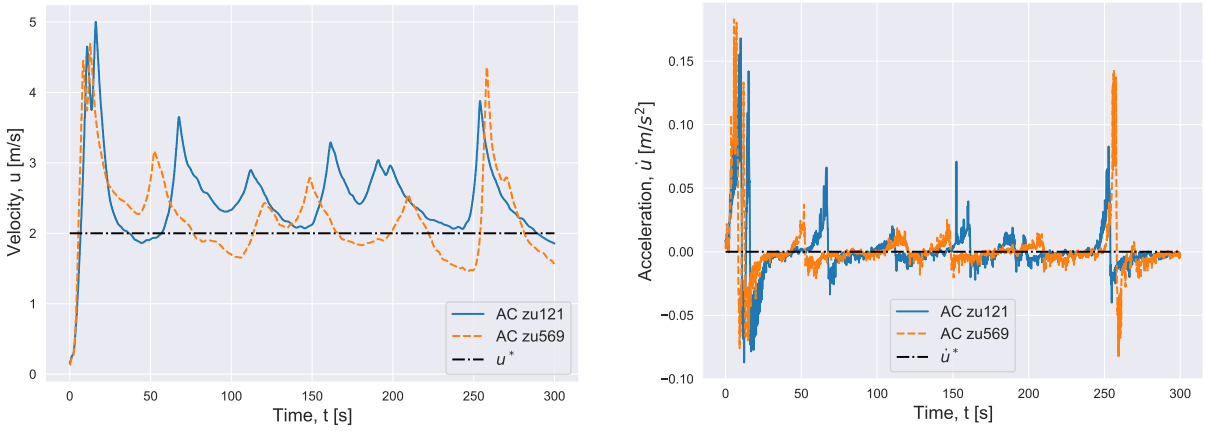
The value functions for the two proposed actor-critic controllers are shown in Figure 5.23. Whilst neither converge, the AC zu121 controller shows a steep decline in the first 40 [s] range, followed by a relatively smooth decline for the remainder of the experiment before tapering off in the final 40 [s] of the experiment. The AC zu569 controller, on the other hand, shows a smooth decline up until approximately 240 [s] where there is a sudden decrease followed by another smooth decline. This sudden drop in $t \in [240, 260]$ [s] range maps to a sharp decrease in altitude, leading to a sudden increase in angular velocity in the propellers, causing sudden increases in both vertical and longitudinal velocities. It is expected that this effect is the result of a sudden downward gust of wind that the system must respond too, which Price et al. and all observe in their simulation experiments in



(a) Vertical position, z [m]

(b) Vertical velocity, $\dot{z}$ [m/s]

Figure 5.20: State responses for suboptimal controllers with random seeds 121 and 569 for a single waypoint experiment in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



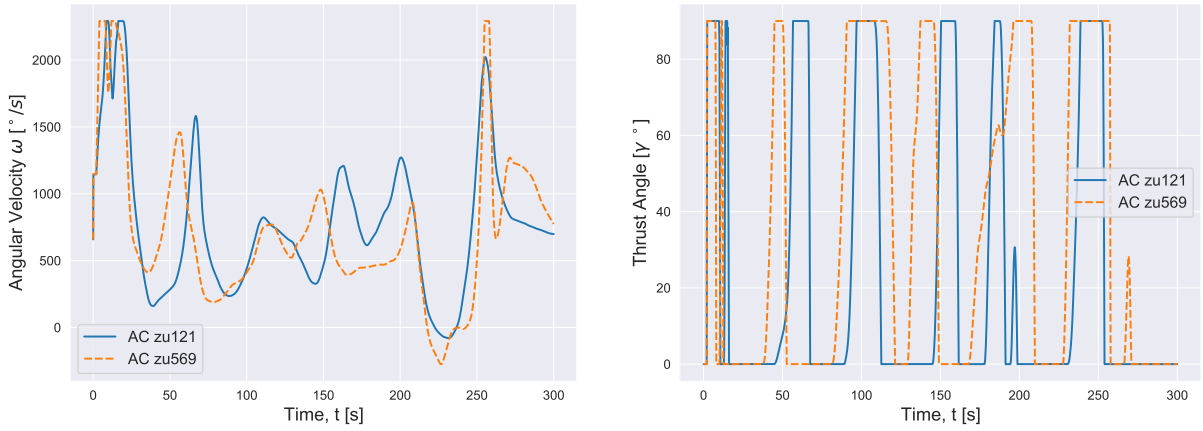
(a) Longitudinal Velocity, u [m/s]

Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 5.21: State responses for suboptimal controllers with random seeds 121 and 569 for a single waypoint experiment in the presence of turbulent wind. (a) longitudinal velocity u [m/s] and (a) longitudinal acceleration $\dot{u}$ [m/s^2].

[58].

Both of the studied controllers show promising results for the single waypoiont alti-



(a) Angular velocity, ω [°/s] (b) Thrust angle, γ [°]

Figure 5.22: Actuator signals for suboptimal controllers with random seeds 121 and 569 for a single waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) thrust angle, γ [°].

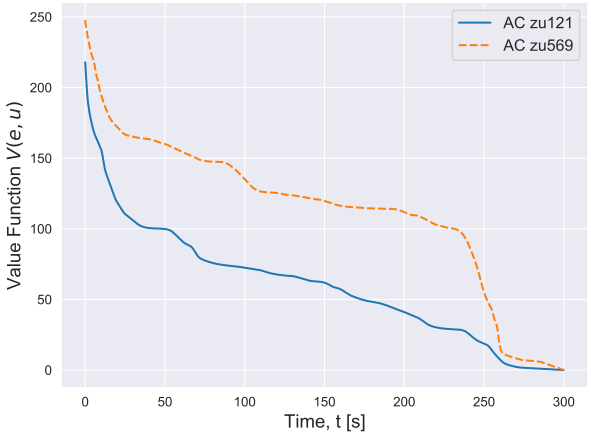


Figure 5.23: Value function for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment.

tude and velocity control task in the nominal and disturbed test cases. Both controllers successfully tracked the desired waypoints for all states while displaying smooth actuator signals that don't show any signs of chattering. To better understand the characteristics of the controllers, a multiple waypoint experiment is performed in both the nominal and

disturbed scenarios and a comparison is made to the FFDL controller.

5.3.2 Altitude and Velocity Controller with Multiple Waypoints

This section investigates the adaptability of the proposed algorithm by adding to the complexity of the implementation from Section 5.3.1. The same two controllers, AC zu121 and AC zu569, are tested in comparison to the FFDL in a multiple waypoint simulation in both nominal and windy environments. In all cases the target states, $\mathbf{x}^*$, and the maximum values of for the states used in (3.5), $\mathbf{x}^{\max}$, for the controller are,

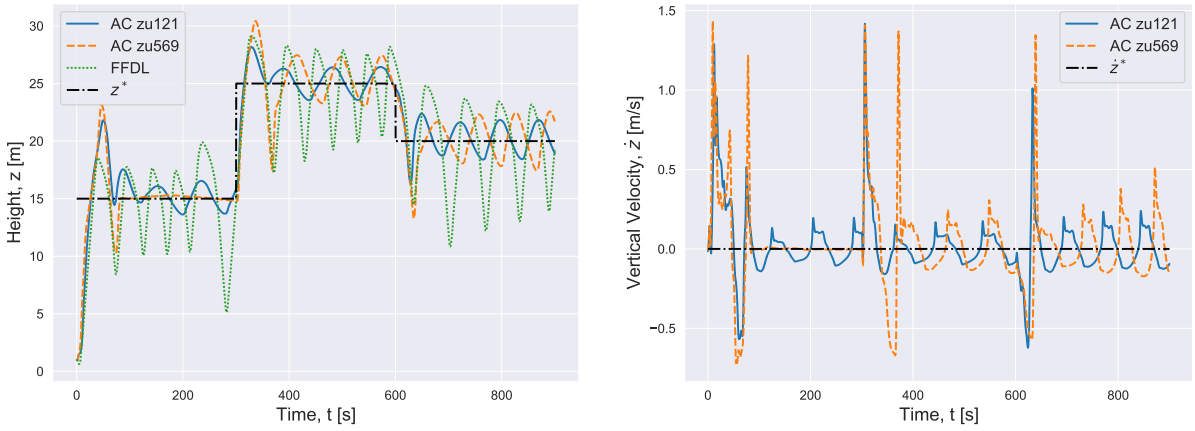
$$\mathbf{x}^* = \begin{bmatrix} z^* \\ \dot{z}^* \\ u^* \\ \dot{u}^* \end{bmatrix} = \begin{bmatrix} T_z \\ 0 \\ 2 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\max} = \begin{bmatrix} z^{\max} \\ \dot{z}^{\max} \\ u^{\max} \\ \dot{u}^{\max} \end{bmatrix} = \begin{bmatrix} 7.5 \\ 1.5 \\ 2.5 \\ 0.5 \end{bmatrix} \quad \text{for} \tag{5.13}$$

$$T_z = \begin{cases} 15, & \text{if } t \leq 300 \text{ [s]}, \\ 25, & \text{if } 300 \text{ [s]} < t \leq 600 \text{ [s]} \\ 20, & \text{if } 600 \text{ [s]} < t. \end{cases} \quad \text{and}$$

he state responses for the nominal experiment are shown in Figures 5.24 and 5.25, with the FFDL being used for a comparison controller.

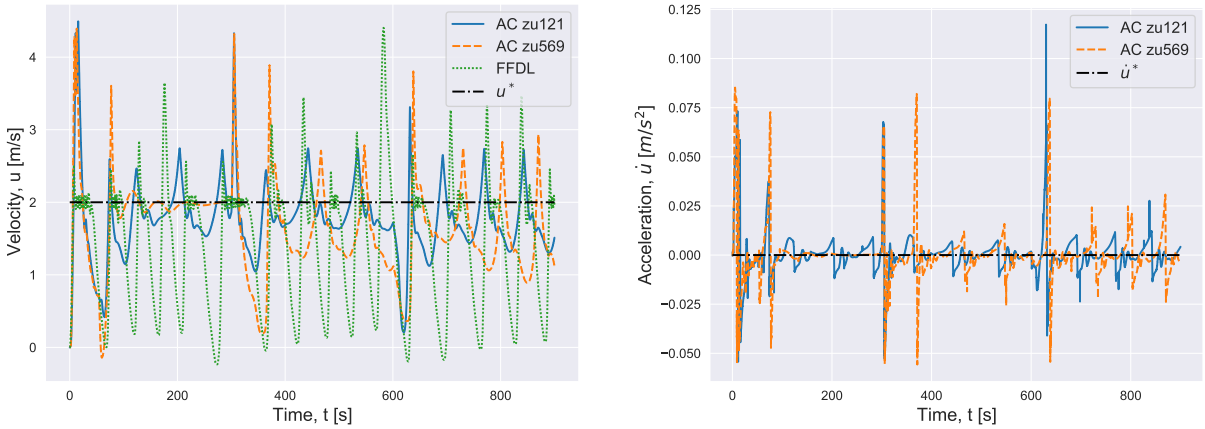
The trajectory plots for the nominal multiple waypoint experiment show the two actor-critic controllers successfully tracking the desired trajectories with the SOC zu569 controller seeming to respond incredibly well in the interval $t \in [100, 300]$ [s] for all feedback states. Outside of that range, the two actor-critic controllers respond very similarly, with their responses being enveloped by the comparison controller, in general. The proposed controllers display lower amplitude and lower frequency oscillations compared to the FFDL. Again the regions where the velocity and altitude responses deviate the most from the desired trajectories for the proposed actor-critic controllers correspond to the step changes in altitude.

As in the previous controllers the angular velocities of the propellers show much smoother responses for the actor-critic controller than for the comparison algorithm. Interestingly, for the vector thrust response, the AC zu569 controller is able to oscillate around the 35 [°] mark for the region where it's most successful in reducing the error. Unfortunately, follow-



(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$

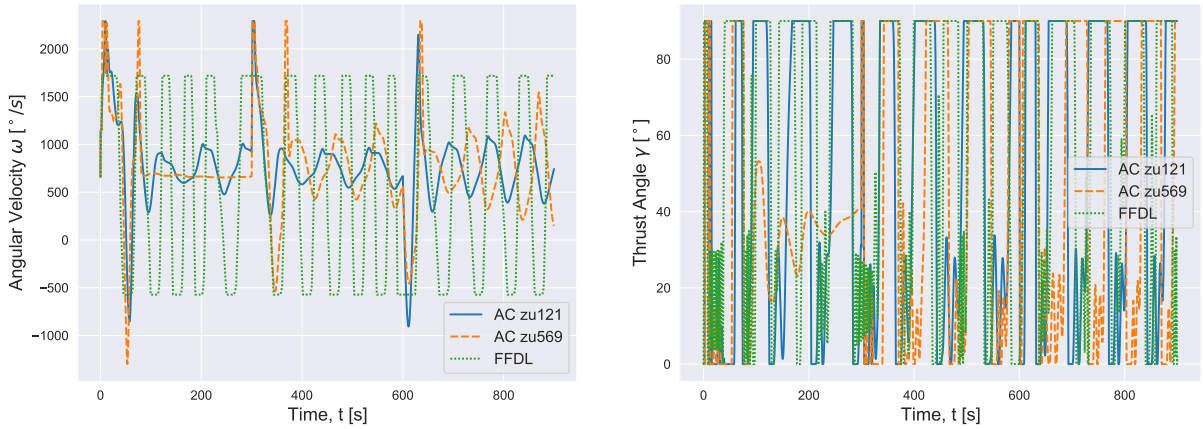
Figure 5.24: State responses for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



(a) Longitudinal Velocity, u [m/s] (b) Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 5.25: State responses for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment. (a) longitudinal velocity u [m/s] and (b) longitudinal acceleration $\dot{u}$ [m/s^2].

ing the step input change in the desired altitude, the response toggles back and forth from the actuator saturation limits, similar to that of AC zu121 and the comparison algorithm.



(a) Longitudinal Velocity, u [m/s] (b) Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 5.26: Actuator signals and value function for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) thrust angle, γ [°].

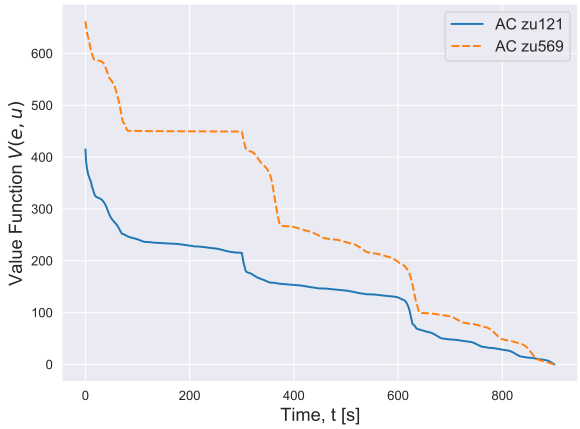


Figure 5.27: Value function for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment.

It is interesting to note, and somewhat unexpected that the AC zu121 controller outscores the other two controllers in every metric for the z and u control, and outscores the AC zu569 in the $\dot{z}$ and $\dot{u}$ performance. This might not have been expected given the ability of the AC zu569 at the end of the first segment of the experiments. This seems to

Table 5.4: Error metrics for actor-critic controllers with random seeds 121, 569 and FFDL for multiple waypoint experiments. (a) vertical position z [m], (b) vertical velocity $\dot{z}$ [m/s], (c) longitudinal velocity u [m] and (d) longitudinal acceleration $\dot{u}$ [m/s^2].

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
AC zu121	19866.63	122688312.94
AC zu569	22566.50	162780943.92
FFDL	32970.12	227937040.44

(b) Error metrics for altitude control $\dot{z}$ [m/s]

Controller	IAE	ITAE
AC zu121	1739.61	9886001.35
AC zu569	2258.62	13947164.43

(c) Error metrics for altitude control u [m/s]

Controller	IAE	ITAE
AC zu121	5961.39	43655096.29
AC zu569	7533.04	62473254.72
FFDL	8103.60	61605695.66

(d) Error metrics for altitude control $\dot{u}$ [m/s^2]

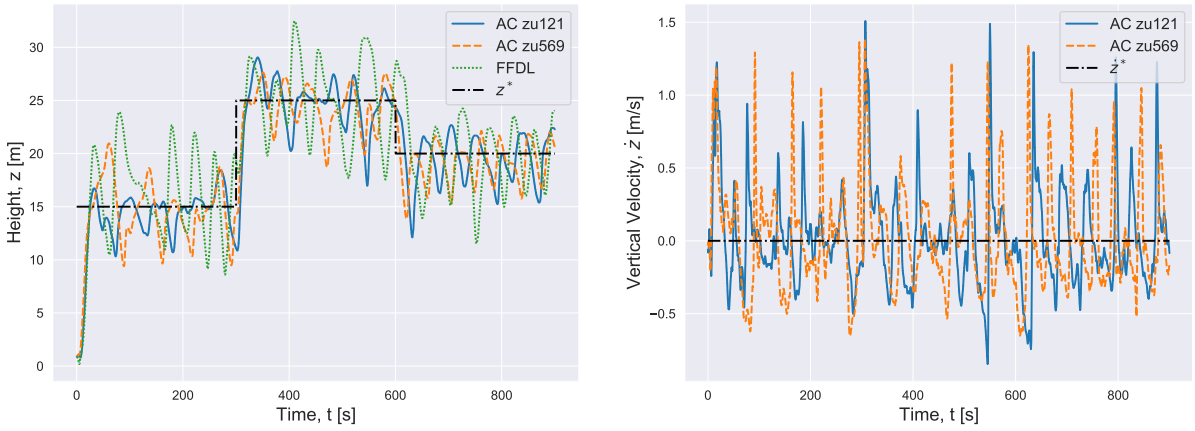
Controller	IAE	ITAE
AC zu121	77.65	429254.93
AC zu569	82.42	491262.77

indicate that the response of the AC zu121 controller is much better outside of this range.

The multiple waypoint experiment is repeated using the same three controllers, target waypoints and maximum values as in the nominal case to investigate the noise rejection abilities of the controllers. The state responses for the airship in the presence of turbulent wind are shown in Figures 5.28 and 5.29.

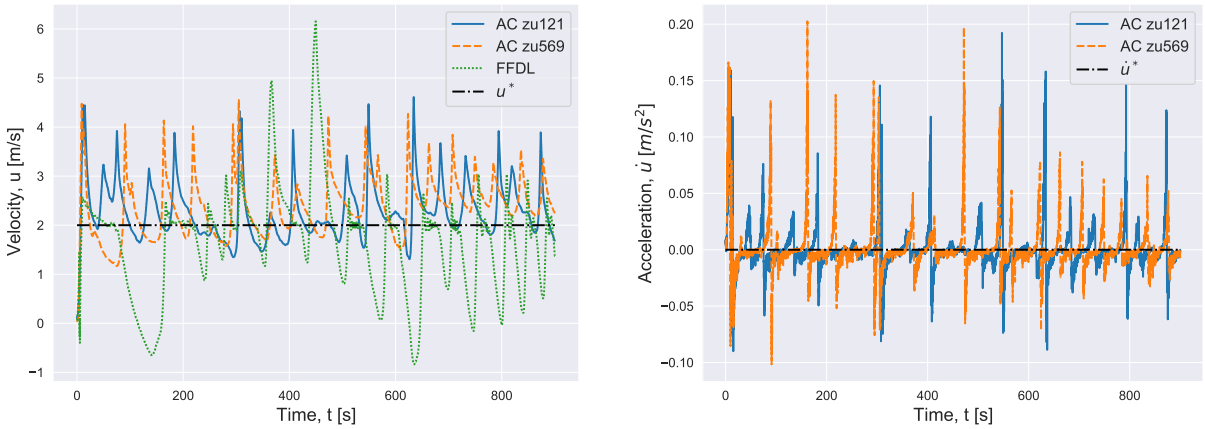
The altitude response of the two proposed actor-critic controllers qualitatively appears to track the desired setpoints while accepting high amplitude oscillation. The response of the actor-critic controller does not seem to match the performance of the single network controller in the presence of wind, but qualitatively seems to compete with the FFDL still. Figure 5.20(a) seems to show the actor-critic controllers both tracking the desired altitude more effectively than the Dynamic Linearization Techniques (DLT) based controller, but the responses are much closer in Figure 5.21(a). One qualitative remark that isn't told in the error metrics in Table 5.5 is that the maximum positive and negative errors in the responses for the vertical position and the longitudinal velocity are all from the FFDL. So while the performances might be close in an overall sense, if a controller is required never to leave a certain range about the desired states, the actor-critic controller seems to perform better than the FFDL.

From the actuator responses in Figure 5.30, it is clear that the actuators have to work significantly harder to achieve the target outputs when compared to the nominal case. High frequency components are observed in both actuator signals in the proposed controllers, with the vector thrust control tending to saturate for all three controllers. All



(a) Vertical position, z [m] (b) Vertical velocity, $\dot{z}$ [m/s]

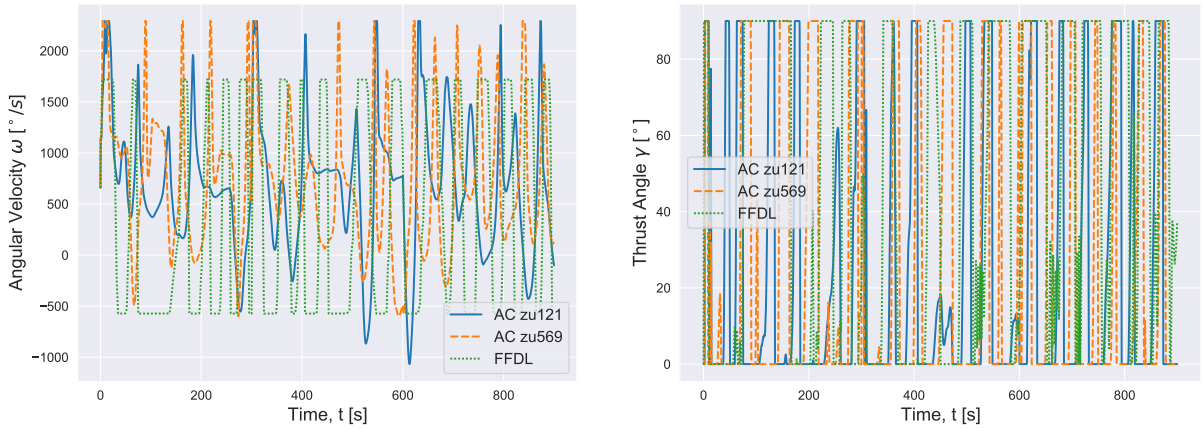
Figure 5.28: State responses for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment in the presence of turbulent wind. (a) vertical position z [m] and (b) vertical velocity $\dot{z}$ [m/s].



(a) Longitudinal Velocity, u [m/s] (b) Longitudinal Acceleration, $\dot{u}$ [m/s^2]

Figure 5.29: State responses for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment in the presence of turbulent wind. (a) longitudinal velocity u [m/s] and (b) longitudinal acceleration $\dot{u}$ [m/s^2].

three controllers seem to struggle to find a control pattern that is able to consistently reduce the error in the states.



(a) Angular velocity, ω [°/s]

(b) Thrust angle, γ [°]

Figure 5.30: Actuator signals and value function for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment in the presence of turbulent wind. (a) angular speed, ω [°/s] and (b) thrust angle, γ [°].

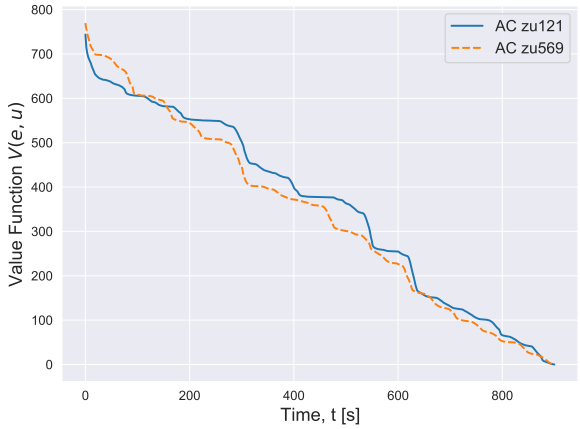


Figure 5.31: Value function for suboptimal controllers with random seeds 121 and 569 for a multiple waypoint experiment in the presence of turbulent wind.

From the error metrics in Table 5.5, the AC zu121 controller outperforms the FFDL controller in the altitude and longitudinal velocity control tasks for both the IAE and ITAE metrics. The AC zu569 controller scores marginally higher than the comparison controller in the ITAE metric for the longitudinal control task, but it is again mentioned that it

Table 5.5: Error metrics for actor-critic controllers with random seeds 121, 569 and FFDL for multiple waypoint experiments. (a) vertical position z [m], (b) vertical velocity $\dot{z}$ [m/s], (c) longitudinal velocity u [m] and (d) longitudinal acceleration $\dot{u}$ [m/s^2].

(a) Error metrics for altitude control z [m]

Controller	IAE	ITAE
AC zu121	26716.75	196735412.27
AC zu569	27834.53	188906019.88
FFDL	34864.63	229404076.05

(b) Error metrics for altitude control $\dot{z}$ [m/s]

Controller	IAE	ITAE
AC zu121	3377.91	22953659.13
AC zu569	3882.76	25544666.89

(c) Error metrics for altitude control u [m/s]

Controller	IAE	ITAE
AC zu121	6135.88	46272761.61
AC zu569	7482.42	60595305.25
FFDL	8062.03	58808995.30

(d) Error metrics for altitude control $\dot{u}$ [°/s^2]

Controller	IAE	ITAE
AC zu121	123.04	786358.03
AC zu569	121.47	725715.21

has the advantage of never having gone into the negative. Between the two actor-critic controllers they both score rather similarly in the metrics for the z, $\dot{z}$, and $\dot{u}$ control, but the AC zu121 controller outperforms the AC zu569 controller for both IAE and ITAE metrics for the longitudinal velocity control.

5.3.3 Summary of Altitude and Velocity Control

Building on the results of the previous section, the actor-critic structure showed promise in its ability to control the altitude and velocity of the airship in simulation. In one of the responses, the airship almost entirely eliminated the steady-state error around the 100 s mark of the flight test. This demonstrates that the algorithm has potential to be improved and substantially outperform similar algorithms in specific cases. While the results in the nominal case were comparable and sometimes superior to the single-network algorithm, the actor-critic did not perform as well in the wind disturbed flight tests.

5.4 Summary

The results in Sections 5.2 and 5.3, show promising performance and improvements in the nominal case compared to the single network controller and the comparison FFDL controller. Specifically, the actor-critic network more effectively addresses the steady-state

error of the airship. The ablation studies show that the controller is capable of overcoming a multitude of initial conditions, and produce admissible control strategies in most cases. The drawback of this controller, when applied to a dirigible seems to be the noise rejection. While the actor-critic is able to compensate for the addition of the turbulent wind, its performance is not as capable as the single network controller in this setting. The reason for this is unclear, however, it could be that the additional parameters introduced by the actor network take longer to adapt and therefore struggle with high-frequency noise rejection. None-the-less, the proposed actor-critic controller is able to outperform the comparison controller in most cases, tracking the desired waypoints in a reasonable time period.

Chapter 6

Conclusion

In this chapter, concluding remarks are offered, briefly summarizing the work in this thesis, and touching on some of the high-level ideas that are introduced. The reader is left with avenues in which this work can be expanded to further investigate and possibly improve the results seen in the experimental setup.

In this work, two online Multi-Input Multi-Output (MIMO) Model-Free Control (MFC) algorithms are developed on the basis of optimal control theory, leveraging results from reinforcement learning techniques. The online nature of the algorithms is sought to allow the plant to be able to generalize its control policy such that it is robust against external disturbances. The model-free structure of the proposed single-network controllers is supported by the parameterization of the control action on the critic-network, which is tuned online through measurements taken along the vehicle's trajectory. Likewise, the double-network controller is developed without a priori knowledge of the system dynamics by updating the parameters of both the critic and actor-networks based on the interaction of the vehicle with its environment.

The efficacy of these algorithms is realized through highly realistic simulations of an airship and compared to a state-of-the-art Data-Driven Control (DDC) strategy. In all nominal test flights and all but one windy test flights, both algorithms outperform the comparison controller, showing improvements in the Integral Absolute Error (IAE) and Integral Time Absolute Error (ITAE) error metrics by up to a factor of four. Qualitatively, the proposed algorithms were able to track the desired waypoint, with minimal oscillations and overshoot compared to the comparison controller and showed, in general, a reduced chattering effect in the actuator signals. Due to dimensionality reduction in the formulation, the computational savings for the value function approximation network (the

critic-network) were approximately 50%, ensuring that the total control loop calculations did not impede the real-time control of the system.

Although the results of this work show promise in the development of an online MIMO DDC strategy for autonomous Lighter Than Air Vehicles (LTAVs), there are areas that deserve additional consideration. Firstly, the intention of this work was to explore potential controllers with simulated validation for airship control. As a result, the stability, convergence and robustness considerations are not discussed at this time and could be useful in expanding this work or finding closed-form solutions to the initialization of the network parameters. Secondly, rather than limiting the task to altitude and velocity control, expansion to higher dimensional control tasks is left open. Specifically, the addition of yaw control could prove to be a significant step on the way to full autonomy. Lastly, pushing the envelop of the vehicle agnostic component of this formulation could yield interesting results. Using the same algorithms, with minor adjustments as needed, on different Robot Operating System (ROS)/Gazebo airship models, or more meaningfully on a physical prototype stands to strengthen the empirical validation of the algorithm's effectiveness.

References

[1] Mohamed Abouheaf, Derek Boase, Wail Gueaieb, Davide Spinello, and Salah Al-Sharhan. "Real-Time Measurement-Driven Reinforcement Learning Control Approach for Uncertain Nonlinear Systems". In: *Engineering Applications of Artificial Intelligence (Elsevier)* 122 (June 2023), pp. 1–18. DOI: 10.1016/j.engappai.2023.106029.

[2] Murad Abu-Khalaf and Frank L. Lewis. "Nearly Optimal Control Laws for Nonlinear Systems with Saturating Actuators Using a Neural Network HJB Approach". In: *Automatica* 41.5 (May 2005), pp. 779–791. DOI: https://doi.org/10.1016/j.automatica.2004.11.034.

[3] Wojciech Adamski, Przemyslaw Herman, Yasmina Bestaoui, and Krzysztof Kozłowski. "Control of Airship in Case of Unpredictable Environment Conditions". In: *2010 Conference on Control and Fault-Tolerant Systems (SysTol)*. 2010, pp. 843–848. DOI: 10.1109/SYSTOL.2010.5675976.

[4] Ahmad Alsayed and Eric Lanteigne. "Experimental Pitch Control of an Unmanned Airship with Sliding Ballast". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2017, pp. 1640–1646. DOI: 10.1109/ICUAS.2017.7991326.

[5] Ahmed Alsayed. "Pitch and Altitude Control of an Unmanned Airship with Sliding Gondola". MA thesis. University of Ottawa, 2017.

[6] Rika Antonova, Jingyun Yang, Priya Sundaresan, Dieter Fox, Fabio Ramos, and Jeannette Bohg. "A Bayesian Treatment of Real-to-Sim for Deformable Object Manipulation". In: *IEEE Robotics and Automation Letters* 7.3 (July 2022), pp. 5819–5826. DOI: 10.1109/LRA.2022.3157377.

[7] Karl J. Astrom and Bjorn Wittenmark. *Adaptive Control*. Ed. by Dover. 2nd. Addison-Wesley Publishing Company, 2008. ISBN: 9780486462783.

[8] J.R. Azinheira, P. Rives, J.R.H. Carvalho, G.F. Silveira, E.C. de Paiva, and S.S. Bueno. "Visual Servo Control for the Hovering of all Outdoor Robotic Airship". In: *Proceedings 2002 IEEE ICRA*. Vol. 3. May 2002, 2787–2792 vol.3. DOI: 10.1109/ROBOT.2002.1013654.

[9] Jacson M. Olszanecki Barth, Jean-Philippe Condomines, Murat Bronz, Gautier Hattenberger, Jean-Marc Moschetta, Cédric Join, and Michel Fliess. "Towards a Unified Model-Free Control Architecture for Tailsitter Micro Air Vehicles: Flight Simulation Analysis and Experimental Flights". In: (Jan. 2020). DOI: https://doi.org/10.2514/6.2020-2075.

[10] Jacson MO Barth, Jean-Philippe Condomines, Murat Bronz, Jean-Marc Moschetta, Cédric Join, and Michel Fliess. "Model-Free Control Algorithms for Micro Air Vehicles with Transitioning Flight Capabilities". In: *International Journal of Micro Air Vehicles* 12 (Jan. 2020), p. 175682932091426. DOI: https://doi.org/10.1177/1756829320914264.

[11] Maria Bekcheva, Cedric Join, and Hugues Mounier. "Cascaded Model-Free Control for Trajectory Tracking of Quadrotors". In: (June 2018). DOI: 10.1109/ICUAS.2018.8453339.

[12] Derek Boase, Wail Gueaieb, and Suruz Miah. "Underactuated MIMO Airship Control Based on Online Data-Driven Reinforcement Learning". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. (Accepted: Jun. 2023). Detroit, MI, USA, Oct. 2023, pp. 1–8.

[13] Ely Carneiro de Paiva, Fabio Benjovengo, and Samuel Siqueira Bueno. "Sliding Mode Control For The Path Following of an Unmanned Airship". In: *IFAC Proceedings Volumes* 40.15 (2007), pp. 221–226. DOI: https://doi.org/10.3182/20070903-3-FR-2921.00040.

[14] Z. Chekakta, M. Zerikat, Y. Bouzid, and M. Abderrahim. "Model-Free Control applied for position control of Quadrotor using ROS". In: (Apr. 2019). DOI: 10.1109/CoDIT.2019.8820614.

[15] Ronghu Chi, Zhongsheng Hou, Shangtai Jin, and Biao Huang. "Computationally Efficient Data-Driven Higher Order Optimal Iterative Learning Control". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.12 (2018), pp. 5971–5980. DOI: 10.1109/TNNLS.2018.2814628.

[16] Maison Clouâtré, Makhin Thitsa, and Michel Fliess Cédric Join. "A Robust but Easily Implementable Remote Control for Quadrotors: Experimental Acrobatic Flight Tests". In: *arXiv preprint arXiv:2008.00681* (2020).

[17] Emad Ebeid, Martin Skriver, Kristian Husum Terkildsen, Kjeld Jensen, and Ulrik Pagh Schultz. "A Survey of Open-Source UAV Flight Controllers and Flight Simulators". In: *Microprocessors and Microsystems* 61 (Sept. 2018), pp. 11–20. DOI: https://doi.org/10.1016/j.micpro.2018.05.002.

[18] Magnus Ekman. *Learning Deep Learning. Theory and Practice of Neural Networks, Computer Vision, Nlp, and Transformers Using Tensorflow*. Pearson Education, Limited, 2021, p. 752. ISBN: 978-0137470358.

[19] A. Elfes, S. Siqueira Bueno, M. Bergerman, and J.G. Ramos. "A Semi-Autonomous Robotic Airship for Environmental Monitoring Missions". In: *International Conference on Robotics and Automation*. Vol. 4. 1998, 3449–3455 vol.4. DOI: 10.1109/ROBOT.1998.680971.

[20] Alberto Elfes, Samuel Bueno, JJG Ramos, Ely Paiva, Marcel Bergerman, JRH Carvalho, Silvio Maeta, LGB Mirisola, BG Faria, and José Azinheira. "Modelling, Control and Perception for an Autonomous Robotic Airship". In: Jan. 2002, pp. 216–244. ISBN: 3-540-43399-6. DOI: 10.1007/3-540-45993-6_13.

[21] Bruce A. Finlayson. *The method of weighted residuals and variational principles*. SIAM, Society for Industrial and Applied Mathematics, 2014, p. 412. ISBN: 978-1-61197-323-5.

[22] Michel Fliess and Cédric Join. "Model-Free Control and Intelligent PID Controllers: Towards a Possible Trivialization of Nonlinear Control?" In: *IFAC Proceedings Volumes (IFAC-PapersOnline)* 15 (May 2009). DOI: 10.3182/20090706-3-FR-2004.00256. URL: https://arxiv.org/abs/0904.0322.

[23] Michel Fliess and Cedric Join. "Intelligent PID Controllers". In: *2008 16th Mediterranean Conference on Control and Automation*. 2008, pp. 326–331. DOI: 10.1109/MED.2008.4601995.

[24] Michel Fliess and Cedric Join. "Model-Free Control". In: *International Journal of Control* 86.12 (Dec. 2013), pp. 2228–2252. DOI: 10.1080/00207179.2013.810345. URL: https://hal-polytechnique.archives-ouvertes.fr/hal-00828135.

[25] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. "An Introduction to Deep Reinforcement Learning". In: *Foundations and Trends® in Machine Learning* 11.3-4 (2018), pp. 219–354. DOI: 10.1561/2200000071.

[26] Sergio B. V. Gomes and Josue G. Ramos. "Airship Dynamic Modeling for Autonomous Operation". In: *Proceedings. 1998 IEEE ICRA*. Vol. 4. 1998, 3462–3467 vol.4. DOI: 10.1109/ROBOT.1998.680973.

[27] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[28] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks". In: *Neural Networks* 3.5 (Jan. 1990), pp. 551–560. DOI: https://doi.org/10.1016/0893-6080(90)90005-6.

[29] Zhongsheng Hou. "The Parameter Identification, Adaptive Control and Model-Free Learning Adaptive Control for Nonlinear Systems". PhD thesis. Northeastern University, 1994.

[30] Zhongsheng Hou and Shangtai Jin. "Data-Driven Model-Free Adaptive Control for a Class of MIMO Nonlinear Discrete-Time Systems". In: *IEEE Transactions on Neural Networks* 22.12 (2011), pp. 2173–2188. DOI: 10.1109/TNN.2011.2176141.

[31] Elie Kahale, Pedro C. Garcia, and Yasmina Bestaoui. "Autonomous Path Tracking of a Kinematic Airship in Presence of Unknown Gust". In: *Journal of Intelligent & Robotic Systems* 69.1-4 (Aug. 2012), pp. 431–446. DOI: https://doi.org/10.1007/s10846-012-9709-2.

[32] Bahare Kiumarsi, Kyriakos G. Vamvoudakis, Hamidreza Modares, and Frank L. Lewis. "Optimal and Autonomous Control Using Reinforcement Learning: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.6 (June 2018), pp. 2042–2062. DOI: 10.1109/TNNLS.2017.2773458.

[33] Jonathan Ko, Daniel J. Klein, Dieter Fox, and Dirk Haehnel. "Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp". In: (Apr. 2007). DOI: 10.1109/ROBOT.2007.363075.

[34] Frédéric Lafont, Jean-François Balmat, Nathalie Pessel, and Michel Fliess. "A Model-Free Control Strategy for an Experimental Greenhouse with an Application to Fault Accommodation". In: *Computers and Electronics in Agriculture* 110 (Jan. 2015), pp. 139–149. DOI: https://doi.org/10.1016/j.compag.2014.11.008.

[35] Eric Lanteigne, Ahmad Alsayed, Dominic Robillard, and Steven G. Recoskie. "Modeling and Control of an Unmanned Airship with Sliding Ballast". English. In: *Journal of Intelligent & Robotic Systems* 88.2-4 (Mar. 2017), pp. 285–297. ISSN: 1573-0409. DOI: 10.1007/s10846-017-0533-6.

[36] Eric Lanteigne, Wail Gueaieb, and Dominic Robillard. "Unmanned Airship Design with Sliding Ballast: Modeling and Experimental Validation". In: *International Conference on Unmanned Aircraft Systems*. 2016, pp. 1246–1253.

[37] Eric Lanteigne and Joshua O'Reilly. "Multibody Dynamic Modeling and Control of an Unmanned Aerial Vehicle under Non-Holonomic Constraints". In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2020, pp. 316–321. DOI: 10.1109/ICUAS48674.2020.9213950.

[38] Frank Lewis, Vrabie L. Draguna, and Symos L. Vassilis. *Optimal Control*. Ed. by John Wiley Sons. 3rd. John Wiley Sons, 2012.

[39] Frank L. Lewis and Derong Liu. *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. John Wiley & Sons, Inc., Dec. 2012. DOI: 10.1002/9781118453988.

[40] Xiaoou Li and Wen Yu. "Dynamic System Identification Via Recurrent Multilayer Perceptrons". In: *Information Sciences* 147.1-4 (Nov. 2002), pp. 45–63. DOI: https://doi.org/10.1016/S0020-0255(02)00207-4.

[41] Torsten Liesk, Meyer Nahon, and Benoit Boulet. "Design and Experimental Validation of a Nonlinear Low-Level Controller for an Unmanned Fin-Less Airship". In: *IEEE Transactions on Control Systems Technology* 21.1 (Jan. 2013), pp. 149–161. ISSN: 1558-0865. DOI: 10.1109/TCST.2011.2178415.

[42] Fei Liu, Zihan Li, Yunhai Han, Jingpei Lu, Florian Richter, and Michael C. Yip. "Real-to-Sim Registration of Deformable Soft Tissue with Position-Based Dynamics for Surgical Robot Autonomy". In: (May 2021). DOI: 10.1109/ICRA48506.2021.9561177.

[43] Yu Tang Liu, Eric Price, Michael J. Black, and Aamir Ahmad. "Deep Residual Reinforcement Learning Based Autonomous Blimp Control". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 12566–12573. DOI: 10.1109/IROS47612.2022.9981182.

[44] Yu Tang Liu, Eric Price, Pascal Goldschmid, Michael J. Black, and Aamir Ahmad. *Autonomous Blimp Control Using Deep Reinforcement Learning*. 2021. DOI: https://doi.org/10.48550/arXiv.2109.10719. URL: https://arxiv.org/abs/2109.10719.

[45] Borislav Mavrin, Shangtong Zhang, Hengshuai Yao, Linglong Kong, Kaiwen Wu, and Yaoliang Yu. "Distributional Reinforcement Learning for Efficient Exploration". In: (2019). DOI: https://doi.org/10.48550/arXiv.1905.06125.

[46] Md Suruz Miah, Amr Elhussein, Fazel Keshtkar, and Mohammed Abouheaf. "Model-Free Reinforcement Learning Approach for Leader-Follower Formation Using Nonholonomic Mobile Robots". In: *The Thirty-Third International Flairs Conference*. Vol. 4. 2020.

[47] Signe Moe, Anne M. Rustad, and Kristian G. Hanssen. "Machine Learning in Control Systems: An Overview of the State of the Art". In: *Artificial Intelligence XXXV*. Ed. by Max Bramer and Miltos Petridis. Cham: Springer International Publishing, 2018, pp. 250–265. ISBN: 978-3-030-04191-5.

[48] Meyer Nahon, Hashim Mazhar, and Torsten Liesk. "Validation of a Dynamics Model and Controller for an Umanned, Finless Airship". In: (Mar. 2013). DOI: 10.2514/6.2013-1300.

[49] Gilmar T. Navajas. "Modeling and Pitch Control of a Re-Configurable Unmanned Airship". MA thesis. University of Ottawa, 2021.

[50] Chunyu Nie, Zewei Zheng, and Ming Zhu. "Three-Dimensional Path-Following Control of a Robotic Airship with Reinforcement Learning". In: *International Journal of Aerospace Engineering* 2019 (Mar. 2019), pp. 1–12. DOI: 10.1155/2019/7854173.

[51] Norman S. Nise. *Control Systems Engineering*. Ed. by Jen Brady. Wiley, 2019.

[52] E.C. de Paiva, S.S. Bueno, S.B.V. Gomes, J.J.G. Ramos, and M. Bergerman. "A Control System Development Environment for AURORA's Semi-Autonomous Robotic Airship". In: *Proceedings 1999 IEEE ICRA*. Vol. 3. May 1999, 2328–2335 vol.3. DOI: 10.1109/ROBOT.1999.770453.

[53] Prashant Peddiraju, Torsten Liesk, and Meyer Nahon. "Dynamics Modeling for an Unmanned, Unstable, Fin-Less Airship". In: (May 2009). DOI: 10.2514/6.2009-2862.

[54] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: (May 2018). DOI: 10.1109/ICRA.2018.8460528.

[55] Alexander R. Perry. "The FlightGear Flight Simulator". In: *Proceedings of the Annual Conference on USENIX Annual Technical Conference*. ATEC '04. Boston, MA: USENIX Association, 2004, p. 31.

[56] Claudio De Persis and Pietro Tesi. "Formulas for Data-Driven Control: Stabilization, Optimality, and Robustness". In: *IEEE Transactions on Automatic Control* 65.3 (Mar. 2020), pp. 909–924. DOI: 10.1109/TAC.2019.2959924.

[57] Eric Price, Michael J. Black, and Aamir Ahmad. *Viewpoint-driven Formation Control of Airships for Cooperative Target Tracking*. 2023. arXiv: 2209.13040 [cs.RO].

[58] Eric Price, Yu Tang Liu, Michael J. Black, and Aamir Ahmad. "Simulation and Control of Deformable Autonomous Airships in Turbulent Wind". In: *Intelligent Autonomous Systems 16*. Ed. by Marcelo H. Ang Jr, Hajime Asama, Wei Lin, and Shaohui Foong. Cham: Springer International Publishing, 2022, pp. 608–626. ISBN: 978-3-030-95892-3.

[59] Michal Puheim and Ladislav Madarasz. "Normalization of Inputs and Outputs of Neural Network Based Robotic Arm Controller in Role of Inverse Kinematic Model". In: (Jan. 2014). DOI: 10.1109/SAMI.2014.6822439.

[60] Jinjun Rao, Zhenbang Gong, Jun Luo, and Shaorong Xie. "A Flight Control and Navigation System of a Small Size Unmanned Airship". In: *IEEE International Conference Mechatronics and Automation, 2005.* Vol. 3. 2005, 1491–1496 Vol. 3. DOI: 10.1109/ICMA.2005.1626776.

[61] Steven Recoskie. "Autonomous Hybrid Powered Long Ranged Airship for Surveillance and Guidance". PhD thesis. University of Ottawa, 2014.

[62] Andrew G. Barto Richard S. Sutton. *Reinforcement Learning: An Introduction. An Introduction*. Ed. by Francis Bach. 2nd. The MIT Press, 2018, p. 552. ISBN: 9780262039246.

[63] Axel Rottmann and Wolfram Burgard. "Adaptive Autonomous Control Using Online Value Iteration with Gaussian Processes". In: *2009 IEEE ICRA*. May 2009, pp. 2106–2111. DOI: 10.1109/ROBOT.2009.5152660.

[64] Axel Rottmann, Christian Plagemann, Peter Hilgers, and Wolfram Burgard. "Autonomous Blimp Control Using Model-Free Reinforcement Learning in a Continuous State and Action Space". In: (Oct. 2007). DOI: 10.1109/IROS.2007.4399531.

[65] Ali Safaei and Muhammad Nasiruddin Mahyuddin. "An Optimal Adaptive Model-Free Control with a Kalman-Filter-Based Observer for a Generic Nonlinear MIMO System". In: *2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*. 2017, pp. 56–61. DOI: 10.1109/I2CACIS.2017.8239033.

[66] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. "Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning". In: *IEEE Access* 9 (2021), pp. 153171–153187. DOI: 10.1109/ACCESS.2021.3126658.

[67] Mohit Sewak. "Deep Reinforcement Learning". In: (2019). DOI: 10.1007/978-981-13-8285-7.

[68] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. "Residual Policy Learning". In: (2018). DOI: https://doi.org/10.48550/arXiv.1812.06298.

[69] J. Sola and J. Sevilla. "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems". In: *IEEE Transactions on Nuclear Science* 44.3 (June 1997), pp. 1464–1468. DOI: 10.1109/23.589532.

[70] Ruizhuo Song, Frank Lewis, Qinglai Wei, Hua-Guang Zhang, Zhong-Ping Jiang, and Dan Levine. "Multiple Actor-Critic Structures for Continuous-Time Optimal Control Using Input-Output Data". In: *IEEE Transactions on Neural Networks and Learning Systems* 26.4 (2015), pp. 851–865. DOI: 10.1109/TNNLS.2015.2399020.

[71] Kyriakos G. Vamvoudakis and Frank L. Lewis. "Online Actor-Critic Algorithm to Solve the Continuous-Time Infinite Horizon Optimal Control Problem". In: *Automatica* 46.5 (May 2010), pp. 878–888. DOI: https://doi.org/10.1016/j.automatica.2010.02.018.

[72] Haoping Wang, Xuefei Ye, Yang Tian, Gang Zheng, and Nicolai Christov. "Model-free Based Terminal SMC of Quadrotor Attitude and Position". In: *IEEE Transactions on Aerospace and Electronic Systems* 52.5 (Oct. 2016), pp. 2519–2528. DOI: 10.1109/TAES.2016.150303.

[73] Yuchen Wang, Hongmei Li, Rundong Liu, Liguo Yang, and Xianling Wang. "Modulated Model-Free Predictive Control With Minimum Switching Losses for PMSM Drive System". In: *IEEE Access* 8 (2020), pp. 20942–20953. DOI: 10.1109/ACCESS.2020.2968379.

[74] Yubin Wen, Li Chen, Kuankuan Liang, and Dengping Duan. "Nonlinear MPC for a Sensorless Multi-Vectored Propeller Airship Based on Sliding Mode Observer with Saturation". In: *Asian Journal of Control* 21.1 (Aug. 2018), pp. 248–263. DOI: 10.1002/asjc.1879.

[75] Paul Werbos. "Approximate Dynamic Programming for Real-Time Control and Neural Modeling". In: (Jan. 1992).

[76] J.C. Willems, I. Markovsky, P. Rapisarda, and B.L.M. De Moor. "A Note on Persistency of Excitation". In: *2004 43rd IEEE Conference on Decision and Control (CDC)*. Vol. 3. 2004, 2630–2631 Vol.3. DOI: 10.1109/CDC.2004.1428856.

[77] Shuangshuang Xiong and Zhongsheng Hou. "Model-Free Adaptive Control for Unknown MIMO Nonaffine Nonlinear Discrete-Time Systems With Experimental Validation". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.4 (Apr. 2022), pp. 1727–1739. DOI: 10.1109/TNNLS.2020.3043711.

[78] Jian Xu, Na Lin, and Ronghu Chi. "Improved High-Order Model Free Adaptive Control". In: *2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS)*. 2021, pp. 704–708. DOI: 10.1109/DDCLS52934.2021.9455488.

[79] Yueneng Yang, Jie Wu, and Wei Zheng. "Positioning Control for an Autonomous Airship". In: *Journal of Aircraft* 53 (May 2016), pp. 1–9. DOI: 10.2514/1.C033709.

[80] Yueneng Yang, Jie Wu, and Wei ZhengWei. "Adaptive Fuzzy Sliding Mode Control for Robotic Airship with Model Uncertainty and External Disturbance". In: *Journal of Systems Engineering and Electronics* 23.2 (2012), pp. 250–255. DOI: 10.1109/JSEE.2012.00032.

[81] Yueneng Yang and Ye Yan. "Neural Network Gain-Scheduling Sliding Mode Control for Three-Dimensional Trajectory Tracking of Robotic Airships". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 229.6 (Feb. 2015), pp. 529–540. DOI: 10.1177/0959651815570353.

[82] Jessie C. Yeager. *Implementation and Testing of Turbulence Models for the F18-HARV Simulation*. Tech. rep. National Aeronautics and Space Administration, 1998.

[83] Yuming Yin, Zhiiun Fu, and Yan Lu. "Online Adaptive Optimal Tracking Control for Model-Free Nonlinear Systems Via a Dynamic Neural Network". In: *Automatika* 64.3 (Feb. 2023), pp. 431–440. DOI: https://doi.org/10.1080/00051144.2023.2170058.

[84] Younes Al Younes, Ahmad Drak, Hassan Noura, Abdelhamid Rabhi, and Ahmed El Hajjaji. "Robust Model-Free Control Applied to a Quadrotor UAV". In: *Journal of Intelligent & Robotic Systems* 84 (Dec. 2016). DOI: 10.1007/s10846-016-0351-2.

[85] Younes Al Younes, Abdelhamid Rabhi, and Hussien Al-Wedyan. "Intelligent Controller Design for MIMO Systems using Model-Free Control and LMI Approaches Applied on a Twin Rotor MIMO System". In: (Mar. 2019). DOI: 10.1109/ICASET.2019.8714219.

[86] Xian Yu, Zhongsheng Hou, Marios M. Polycarpou, and Li Duan. "Data-Driven Iterative Learning Control for Nonlinear Discrete-Time MIMO Systems". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.3 (Mar. 2021), pp. 1136–1148. DOI: 10.1109/TNNLS.2020.2980588.

APPENDICES

Appendix A

Experiment Initial Conditions

This section summarizes the initial conditions of the controllers used in the ablation study experiments.

A.1 Ablation Study for Altitude Controller for Single Network Algorithm

Initial values of the non-divergent initial critic weights $\boldsymbol{\Omega}_c^{[0]}$ in Section 4.2.5.

Initial weights for controller SOC z000:

$$\boldsymbol{\Omega}_c^{[0],z000} = [0.50, 0.65, 1.00, 0.36, 0.55, 0.36]^T$$

Initial weights for controller SOC z295:

$$\boldsymbol{\Omega}_c^{[0],z295} = [0.98, 0.93, 1.00, 0.87, 0.83, 0.88]^T$$

Initial weights for controller SOC z569:

$$\boldsymbol{\Omega}_c^{[0],z569} = [0.98, 0.54, 0.79, 0.90, 0.68, 1.00]^T$$

Initial weights for controller SOC z738:

$$\boldsymbol{\Omega}_c^{[0],z738} = [0.47, 0.46, 0.46, 0.64, 0.60, 1.00]^T$$

Initial weights for controller SOC z930:

$$\mathbf{\Omega}_c^{[0],z295} = [0.30, 0.43, 0.93, 0.36, 0.94, 1.00]^T$$

A.2 Ablation Study for Altitude and Velocity Controller for Single Network Controller

Initial weights for controller SOC z290:

$\mathbf{\Omega}_c^{[0],zv290} =$

$$[1.00,\ 0.60,\ 0.44,\ 0.65,\ 0.36,\ 0.78,\ 0.81,\ 0.51,\ 0.53,\ 0.91,\ 0.51,\ 0.28,\ 0.51,\ 0.45,\ 0.43,\ 0.53,\ 0.29,\ 0.38,\ 0.48,\ 0.35,\ 0.34]^T$$

Initial weights for controller SOC z295:

$\mathbf{\Omega}_c^{[0],zv295} =$

$$[0.61,\ 0.62,\ 1.00,\ 0.46,\ 0.79,\ 0.84,\ 0.40,\ 0.49,\ 0.51,\ 0.46,\ 0.45,\ 0.68,\ 0.61,\ 0.37,\ 0.61,\ 0.42,\ 0.55,\ 0.58,\ 0.49,\ 0.42,\ 0.60]^T$$

Initial weights for controller SOC z392:

$\mathbf{\Omega}_c^{[0],zv392} =$

$$[0.99,\ 0.77,\ 0.98,\ 0.67,\ 0.50,\ 0.87,\ 0.36,\ 0.56,\ 0.28,\ 0.58,\ 0.75,\ 0.70,\ 0.42,\ 0.41,\ 0.90,\ 0.84,\ 0.63,\ 0.63,\ 0.48,\ 0.73,\ 1.00]^T$$

Initial weights for controller SOC z738:

$\mathbf{\Omega}_c^{[0],zv738} =$

$$[0.66,\ 0.72,\ 0.98,\ 0.50,\ 0.67,\ 0.72,\ 0.35,\ 0.51,\ 0.56,\ 0.58,\ 0.58,\ 0.85,\ 0.52,\ 0.44,\ 0.82,\ 0.67,\ 0.85,\ 0.62,\ 0.56,\ 0.78,\ 1.00]^T$$

A.3 Ablation Study for Altitude Controller for Double Network Algorithm

Initial values of the non-divergent initial critic weights $\mathbf{\Omega}_a^{[0]}$ in Section 5.2.2.

Initial weights for controller AC z115:

$$\mathbf{\Omega}_a^{[0],z115} = [-0.20, -0.70, -0.41]^T$$

Initial weights for controller AC z295:

$$\mathbf{\Omega}_a^{[0],z295} = [-0.62, -0.33, -0.73]^T$$

Initial weights for controller AC z569:

$$\mathbf{\Omega}_a^{[0],z569} = [-0.88 - 0.28 - 0.51]^T$$

Initial weights for controller AC z738:

$$\mathbf{\Omega}_a^{[0],z738} = [-0.34, -0.35, -0.08]^T$$

Initial weights for controller AC z930:

$$\mathbf{\Omega}_a^{[0],z295} = [-0.69, -0.96, -0.81]^T$$

A.4 Ablation Study for Altitude Controller for Double Network Algorithm

Initial values of the non-divergent initial critic weights $\mathbf{\Omega}_a^{[0]}$ in Section 5.3.1.

Initial weights for controller AC zv115:

$$\mathbf{\Omega}_a^{[0],zv115} = \begin{bmatrix} -0.20 & -0.41 & -0.73 & -0.57 & -0.54 & -0.54 & -0.08 & -1.00 & -0.15 & -0.94 \\ -0.70 & -0.58 & -0.19 & -0.61 & -0.79 & -0.40 & -0.72 & -0.14 & -0.21 & -0.80 \end{bmatrix}^T$$

Initial weights for controller AC zv121:

$$\boldsymbol{\Omega}_a^{[0],zv121} = \begin{bmatrix} -0.11 & -0.23 & -0.83 & -0.56 & -0.25 & -0.95 & -0.02 & -0.40 & -0.30 & -0.41 \\ -0.21 & -0.15 & -0.41 & -0.75 & -0.97 & -0.49 & -0.86 & -0.62 & -0.93 & -0.57 \end{bmatrix}^T$$

Initial weights for controller AC zv290:

$$\boldsymbol{\Omega}_a^{[0],zv290} = \begin{bmatrix} -0.27 & -0.74 & -0.33 & -0.97 & 0.00 & -0.11 & -0.53 & -0.45 & -0.20 & -0.73 \\ -0.26 & -0.03 & -0.08 & -0.58 & -0.73 & -0.44 & -0.57 & -0.99 & -0.20 & -0.28 \end{bmatrix}^T$$

Initial weights for controller AC zv295:

$$\boldsymbol{\Omega}_a^{[0],zv295} = \begin{bmatrix} -0.62 & -0.73 & -0.72 & -0.84 & -0.49 & -0.53 & -0.02 & -0.89 & -0.45 & -0.19 \\ -0.33 & -0.47 & -0.63 & -0.84 & -0.41 & -0.09 & -0.51 & -0.76 & -0.75 & -0.85 \end{bmatrix}^T$$

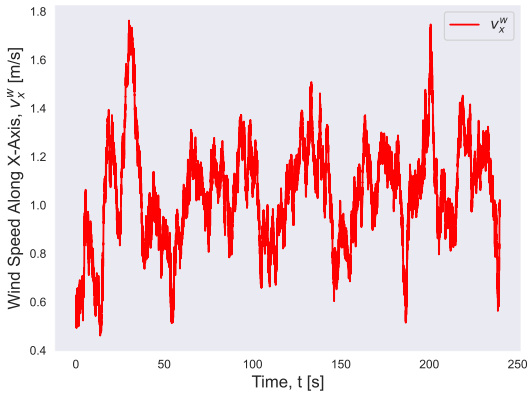
Initial weights for controller AC zv569:

$$\boldsymbol{\Omega}_a^{[0],zv569} = \begin{bmatrix} -0.88 & -0.51 & -0.95 & -0.71 & -0.92 & -0.85 & -0.88 & -0.52 & -0.78 & -0.34 \\ -0.28 & -0.29 & -0.53 & -0.32 & -0.32 & -0.65 & -0.47 & -0.28 & -0.43 & -0.80 \end{bmatrix}^T$$

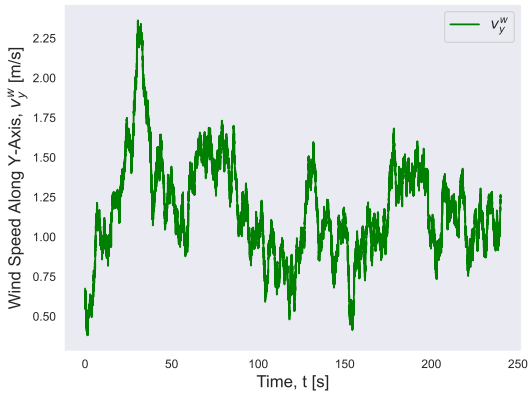
Appendix B

Wind Characterization

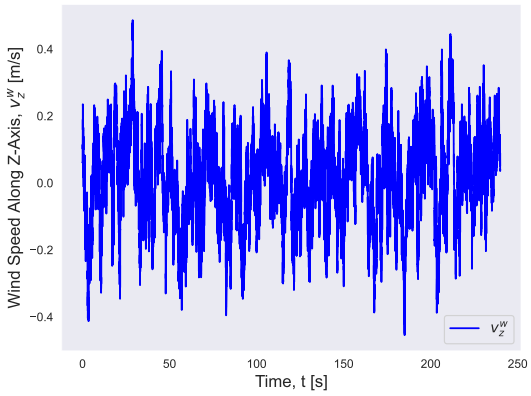
Wind response of empty Robot Operating System (ROS)/Gazebo simulation environment with Dryden turbulent wind filter used in the wind disturbed experiments.



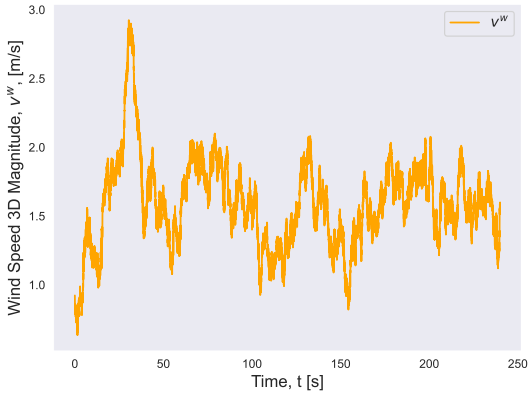
(a) X-component of wind speed vector in the world frame, v_x^w [m/s]



(b) Y-component of wind speed vector in the world frame, v_y^w [m/s]



(c) Z-component of wind speed vector in the world frame, v_z^w [m/s]



(d) Magnitude of wind speed vector in the world frame, $|\mathbf{v}^w|$ [m/s]

Figure B.1: Wind speed measurements over a 240 [s] period